Cryptographic Applications and Methods

Joel Jaeggli
Some liberal help from:
Brian Candler and Hervey Allen

Most recently updated for AFNOG 2006

Why use cryptographic systems?

- Cryptography can offer genuinely secure solutions to important security problems.
- Confidentiality – Can I ensure that no one can see my data in the event of interception or loss?
- Data Integrity – Has this data been modified?
- Authentication – Through the presentation of cryptographic credentials can I uniquely identify myself to a 3rd party?

Cryptographic system usage

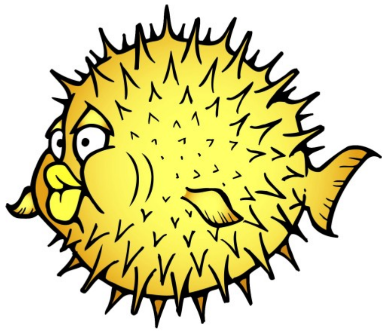
- It used to be be the case that the principle users of cryptography, were governments interested in protecting their own communications or spying on their enemies.
- Cryptography has become one of the fundamental technologies underpinning the late 20th and early 21st century economies. It secures transactions, protects legal agreements against repudiation, enforces access controls on media, and serves as a general purpose wrapper for communications.

Key approaches used in modern cryptographic systems

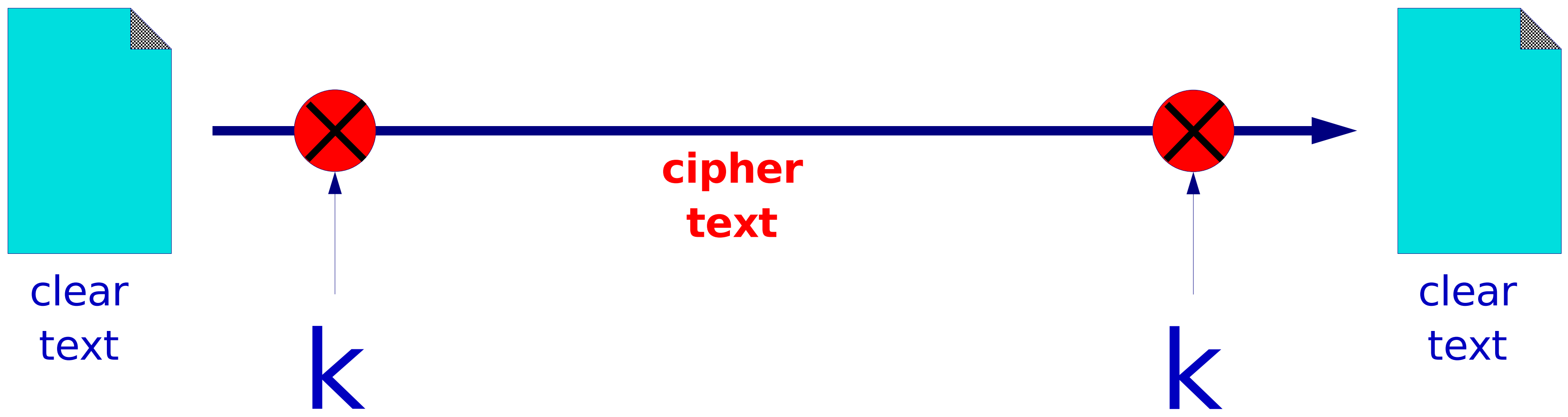
- Shared key symmetric ciphers
- One way encryption through hashing
- asymmetric (Public and Private) cipher systems

Symmetric ciphers

- In symmetric ciphers the same key is used to encrypt and decrypt the data.
- Examples of symmetric cyphers:
  - DES, 56bit key length
  - 3DES, effectively 112bits
  - AES, 128 – 256 bit key length
  - Blowfish, 128 bits
  - Idea, 128 bits



Symmetric cipher diagram



The same key is used to encrypt the document before sending and decrypt it at the far end.

Symmetric ciphers - continued

- Symmetric ciphers can be subdivided into:
- Block ciphers - Operate on chunks of data (64 bits in DES, 128 in AES). This kind transform is known as a pseudo-random function, and is in general unvaryingly applied to each block.
- Stream ciphers which operate on data either one bit or byte at a time (the rc4 cipher, and A5/1 used in encrypting GSM communications are stream cyphers). The transformation of successive bytes varies based on the current state.

Features of symmetric cipher systems

- Fast to encrypt and decrypt, suitable for large volumes of data, and real time applications.
- A well designed symmetric cipher can only be attacked through brute-force exhaustion of the key space.
  - 40, 48, 56, and 64bit ciphers have been successfully attacked by brute force means. ASIC based hardware costing on order of $100K can attack 56bit ciphers in minutes or hours.

Features - Continued

- Current recommendation would be to choose a cipher Based on the ability to secure data for at least 20 years. 90 bits would probably have been acceptable in 2000, but now?
- Symmetric ciphers do not solve the problem of key distribution.
- Reversibility – literally decryption in a symmetric cipher is the reverse of encryption.

Symmetric cipher examples

```
$ cat bank-note

This is some text that will be protected
through strong encryption.
----------------------------------------
The password for bank account number:

011 4444 7356 118

is:

rosebud
```

1

```
$ gpg --cipher-algo blowfish -c bank-note
Enter passphrase: xxxxxx
Repeat passphrase: xxxxxx
$
```

2

```
 $ less bank-note.gpg
"bank-note.gpg" may be a binary file.  See it anyway?
<8C>^M^D^D^C^B^@^?<84><9D><B6><9F>:^Y`ε<BB><F0>}1k<88>^E<93><Ĉ<9
4>^^e<A7>^W<U+07E0><FA>"<D8>L<U+076A>5<8F>b\<91><85>^D<90>J<E1>/
$<U+04F7>^H^\X.i<86>t<C0><8F>8mA<E6>J6 <DE>k}O<A9>2^?<98><D2>`^S
<C9><EB>h<BC>ESC^A<A0><96><E0><96>y<C6>b<96><C1><A5><E4>j<D4>ObO
S^Z<F4><FD><D5>tp<9D>s<AE><D1>^A^<C2>^_I<81><C5><E7>^V<D6>^E{<C0
>E<A7><F0><83>^H\<C5>4f<EB><E2>:^\<87>a<87>_<9C><FC><F6><A1>h+<F
E><96>O<DD>j<F6>[8^E<E9>L<AD>`Ɠ<9C>^]_
bank-note.gpg (END)
```

3

```
$ gpg -d bank-note.gpg
gpg: BLOWFISH encrypted data
Enter passphrase: xxxxxx
gpg: encrypted with 1 passphrase
This is some text that will be protected
through strong encryption.
----------------------------------------
The password for bank account number:

011 4444 7356 118

is:

rosebud
gpg: WARNING: message was not integrity protected
```

4

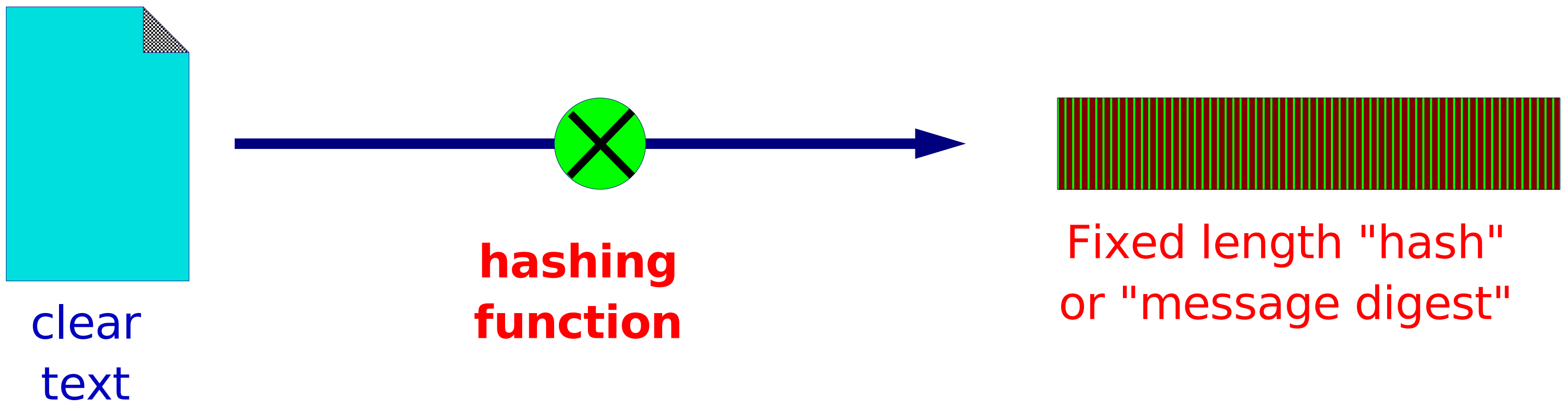
One-way encryption(hashing)

- A one way mathematical transform that cannot be reversed to product the original data.
- The message digest or checksum is always the same size for a given algorithm.
- Changing one bit in the original data would produce a completely different checksum.

One way hash functions

- Unix crypt (derived from DES)
- MD5 (message digest 5) 128bit hash
- SHA1 (secure hash algorithm) 160 bits
- SHA 256/512

Hash Function



Munging the document gives a short "message digest" (checksum). It is not possible to go back from the digest to the original document.

Warning!

- When we gave this talk in 2000 we said:
  - No two documents have yet been discovered which have the same MD5 digest!
  - No feasible method to create any document that has a given MD5 digest.
- The first statement is now categorically untrue.
- The second is now less true than it once was.
- SHA1 is now looking a little weak as well.
- The implication is clear, you need to periodically evaluate the threats to cryptographic systems based on current knowledge.

Applications of one-way hashes

- Integrity checks
  - No matter how much data you run through md5 you get a hash of the same size.
  - Despite recent changes in understanding, It is infeasible for an attacker to modify a file and leave it with the same MD5 checksum.
  - Using SHA1 or SHA256 instead of md5 is still probably a good idea.
- Password storage
  - Password files with clear text password in them are a very attractive target.

Applications - continued

  - When a user logs in, we hash the password they give us and compare it to the hash in the password or shadow file.
  - Can the password still be recovered? That depends.

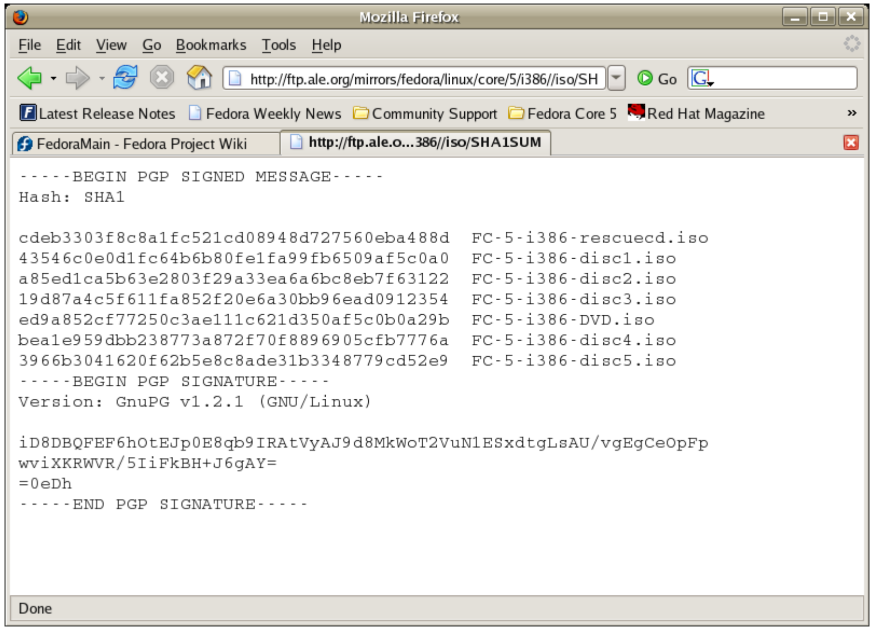
- Generating encryption keys
  - For some reason, users can't remember 128bit numbers.
  - a hash can be used to convert a password/phrase into a fixed length 128bit key.

Passwords - Digression

- How good does a password/phrase have to be in order to protect against brute-force or dictionary attacks against the password itself?
- Entropy in language.
  - A typical english sentence has 1.2 bits of entropy per character, you need 107 characters to get a statistically random md5 hash.
  - Using totally random english characters you need 28 characters.
  - Using a random distribution of all 95 printable ascii characters you need 20 characters.
- Observation, good passwords are hard to come by.

* From the PGP passphrase faq

Hash example


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://ftp.ale.org/mirrors/fedora/linux/core/5/i386//iso/SH
Go
Latest Release Notes
Fedora Weekly News
Community Support
Fedora Core 5
Red Hat Magazine
FedoraMain - Fedora Project Wiki
http://ftp.ale.o...386//iso/SHA1SUM
Done

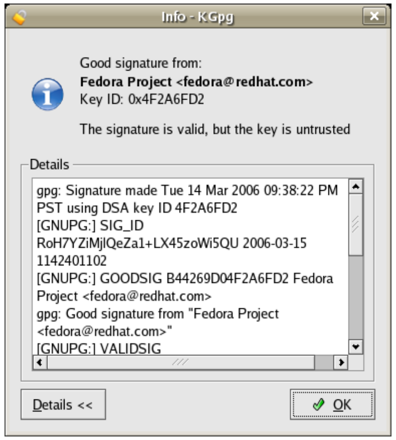
```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

cdeb3303f8c8a1fc521cd08948d727560eba488d  FC-5-i386-rescuecd.iso
43546c0e0d1fc64b6b80fe1fa99fb6509af5c0a0  FC-5-i386-disc1.iso
a85ed1ca5b63e2803f29a33ea6a6bc8eb7f63122  FC-5-i386-disc2.iso
19d87a4c5f611fa852f20e6a30bb96ead0912354  FC-5-i386-disc3.iso
ed9a852cf77250c3ae111c621d350af5c0b0a29b  FC-5-i386-DVD.iso
bea1e959dbb238773a872f70f8896905cfb7776a  FC-5-i386-disc4.iso
3966b3041620f62b5e8c8ade31b3348779cd52e9  FC-5-i386-disc5.iso
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.1 (GNU/Linux)

iD8DBQFEF6hOtEJp0E8qb9IRAtVyAJ9d8MkWoT2VuN1ESxdtgLsAU/vgEgCeOpFp
wviXKRWVR/5IiFkBH+J6gAY=
=0eDh
-----END PGP SIGNATURE-----
```

1

2



Good signature from:
Fedora Project <fedora@redhat.com>
Key ID: 0x4F2A6FD2

The signature is valid, but the key is untrusted

```
gpg: Signature made Tue 14 Mar 2006 09:38:22 PM
PST using DSA key ID 4F2A6FD2
[GNUPG:] SIG_ID
RoH7YZiMjlQeZa1+LX45zoWi5QU 2006-03-15
1142401102
[GNUPG:] GOODSIG B44269D04F2A6FD2 Fedora
Project <fedora@redhat.com>
gpg: Good signature from "Fedora Project
<fedora@redhat.com>"
[GNUPG:] VALIDSIG
```

```
[root@limestone root]# cd /var/ftp/fedora/5/i386/iso
[root@limestone iso]# ls -la
total 6403108
drwxr-xr-x  2 263 263       4096 Mar 14 21:39 .
drwxr-xr-x  5 263 263       4096 Mar 14 20:32 ..
-rw-r--r--  1 263 263  687235072 Mar 14 20:47 FC-5-i386-disc1.iso
-rw-r--r--  1 263 263  700618752 Mar 14 20:48 FC-5-i386-disc2.iso
-rw-r--r--  1 263 263  721016832 Mar 14 20:50 FC-5-i386-disc3.iso
-rw-r--r--  1 263 263  720910336 Mar 14 20:51 FC-5-i386-disc4.iso
-rw-r--r--  1 263 263  387753984 Mar 14 20:52 FC-5-i386-disc5.iso
-rw-r--r--  1 263 263 3253669888 Mar 14 20:49 FC-5-i386-DVD.iso
-rw-r--r--  1 263 263   79122432 Mar 14 20:31 FC-5-i386-rescuecd.iso
-rw-r--r--  1 263 263        671 Mar 14 21:38 SHA1SUM
[root@limestone iso]# sha1sum FC-5-i386-disc1.iso
43546c0e0d1fc64b6b80fe1fa99fb6509af5c0a0  FC-5-i386-disc1.iso
[root@limestone iso]#
```

3

Public key ciphers

- For most of the history of cryptography, it was necessary for the key to the encryption system, known to both parties, to be kept secret in order to protect the enciphered data.
- Through new methods pioneered in the 1970's that basically changed. Widespread adoption would wait until the advent of the commercial use of the Internet.
- RSA and DSA are the most common systems in use here.

Two keys for encryption

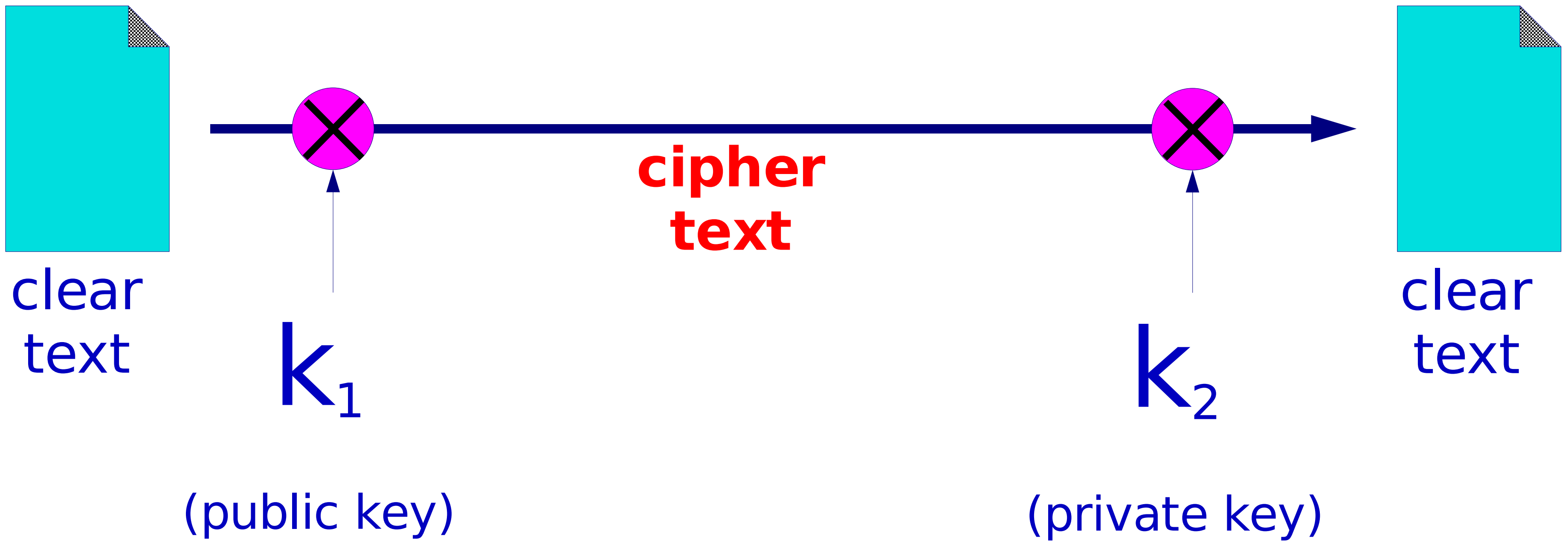
- Keys are generated in pairs, one public (don't have to protect it) the other private (keep this one secret).
- The keys are mathematically related. In the case of RSA, the public key is the product of two very large prime numbers and a public exponent, the private key contains both primes and a few other values related to the mathematical transform.

Two keys – continued

- The part of key distribution problem is solved. You can post the public key anywhere. People can encrypt messages to you, which can only be decrypted with the private key
- The system is based on the assumption that it easy to convert the private key in into the public but computationally infeasible to do the reverse.
- “No polynomial-time method for factoring large integers on a classical computer has yet been found, but it has not been proven that none exists.*

* Wikipedia - Public_key_encryption

Public Key diagram



One key is used to encrypt the document, a different key is used to decrypt it

How big are the keys? - digression

- As of 2005 the largest asymmetric key factored by conventional methods was 663 bits, 2^663 is 200 decimal digits. 1024 bits is 309 decimal digits, 2048 bits is 617 decimal digits.
- Every time you add a bit, the search space doubles in size.
- Numbers on this scale have little relevance outside mathematics, e.g. the number of atoms in the visible universe can be expressed as about 10^81.

- US national debt in USD*
  - 8355718000000
- 10^81 –number o f atoms in visible universe
  - 1000000000000000000000000000000000000000000000000000000000000000000000000000000000
- 2^663 –bi ggest asymmetric key factored to date
  - 382725258645104877886550812609503094103299358017333278221360589891900028618480487938625922563193922631543
    17107227530633921701753935367770195646260767912323717284188541295913442384799112073398408183808
- 2^1024 1024 bit key
  - 179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120
    113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005
    768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216
- 2^2048 2048 bit key
  - 323170060713110073007148766886699519604441026697154840321303454275246551388678908931972014115229134636887
    179609218980194941195591504909210950881523864482831206308773673009960917501977503896521067960576383840675
    682767922186426197561618380943384761704705816458520363050428875758915410658086075523991239303855219143333
    896683424206849747865645694948561760353263220580778056593310261927084603141502585928641771167259436037184
    618573575983511523016459044036976132332872312271256847108202097251571017269313234696785425806566979350459
    97268352998638215525166389437335543602135433229604645318478604952148193555853611059596230656

* as of april 30 2006

Another use, authentication

- Cipher some text using the private key. If you can decipher it with the public-key you can prove it was written by the private key holder.

Securing the private key

- Obviously the linchpin in the system is the proper handling of the private key. If access to it is compromised the system fails.
- For applications like PGP consider storing it off-line, for example, on a USB memory stick or smart-card.
- Consider storing it encrypted when it has to be on a machine, for example passphrase protected for SSH or SSL keys.

Securing the private key - continued

- Unfortunately some applications make it infeasible to use password protected private keys.
- Nobody likes to have to type their password every time a web server is rebooted.

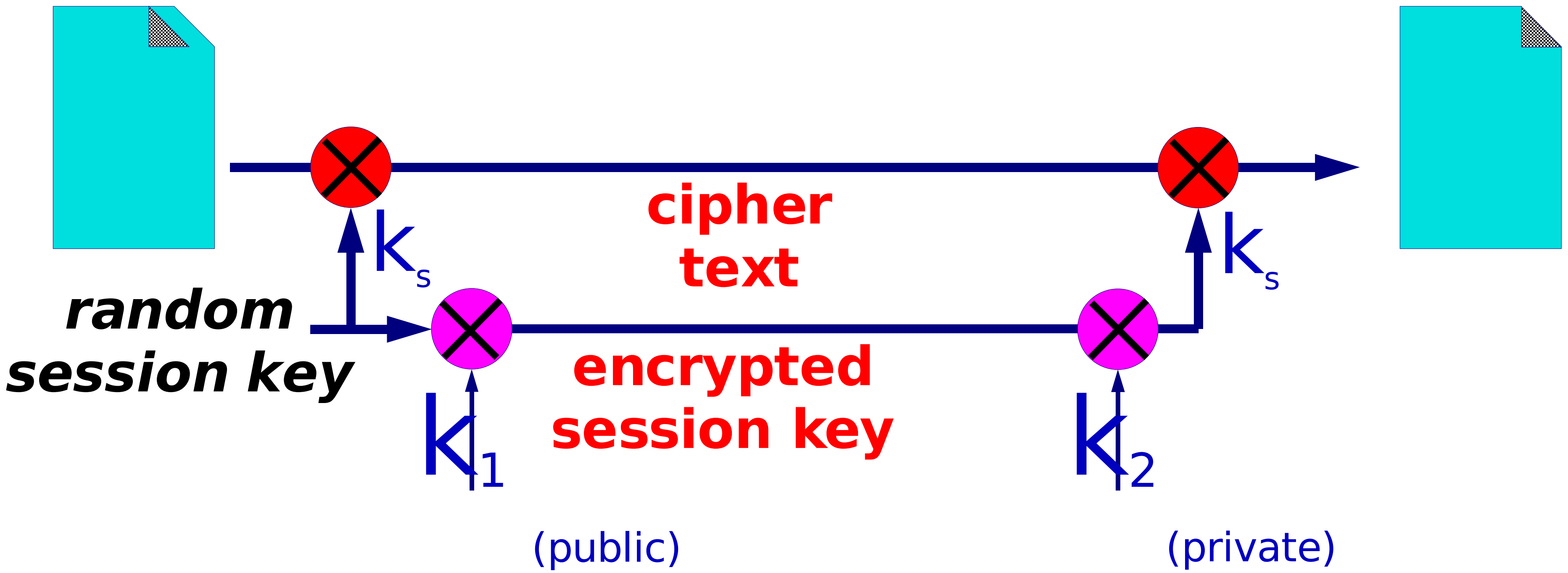
Application

- Public key cryptographic systems are The cornerstone of a number of now critical applications, however they have some problems...
- They can be very computationally expensive, a problem for which a relatively simple solution is to leverage the other two techniques...
- Key distribution from the perspective of trust is also a problem.

Application - encrypting

- To encrypt a stream of data, use a symmetric cipher.
- Select a random key for the session.
- Use a public key cipher to protect the session key.

Encrypting the shared key - diagram

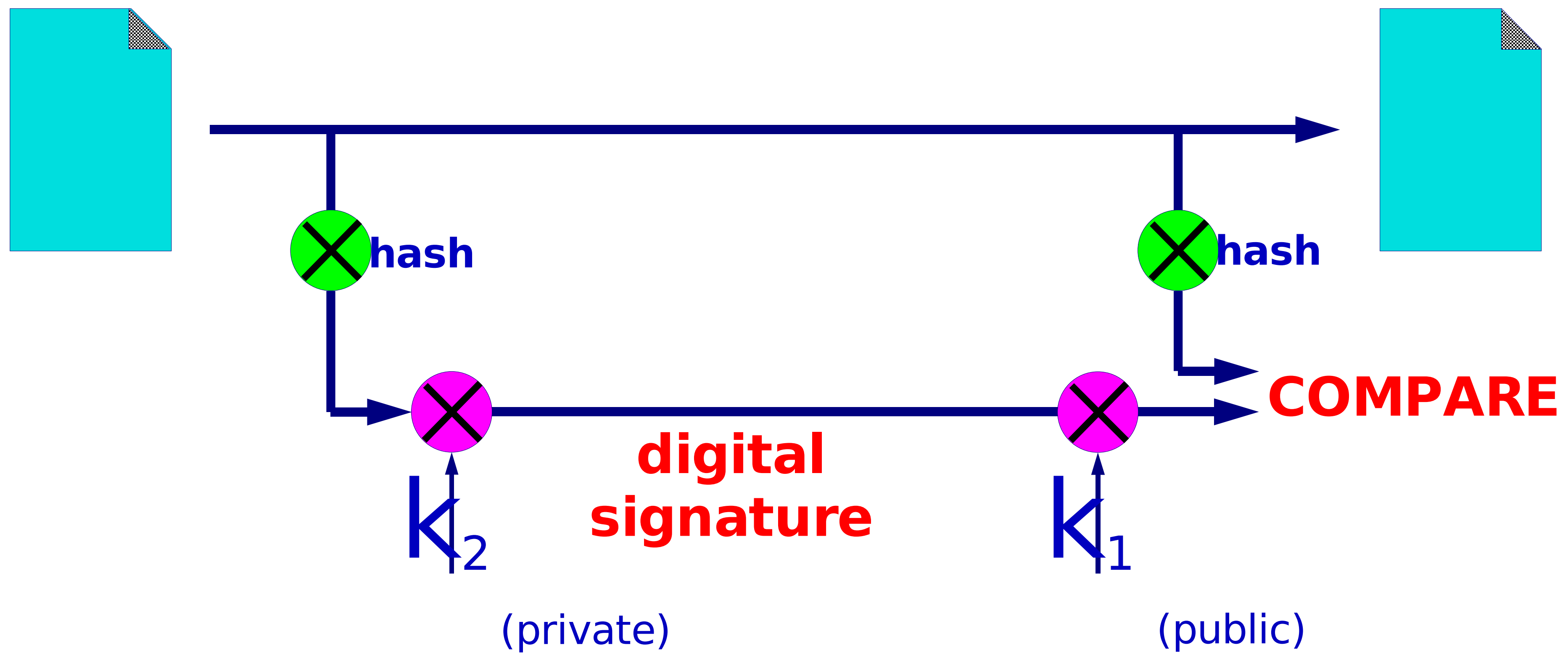


Use a symmetric cipher with a randomly generated key. Use a public key cipher to encrypt the session key, and send it along with the encrypted document.

Application –authenti cation revisited

- Rather than encrypting an entire document as a form of authentication, take a hash of the document and encrypt only that.
- This is known as a digital signature.
- Remember, only the holder of the private key can encipher a message which can be decrypted with the public key.
- Digital signatures have a number of applications including e-commerce and declarations of identity (certificates).

Authentication - diagram



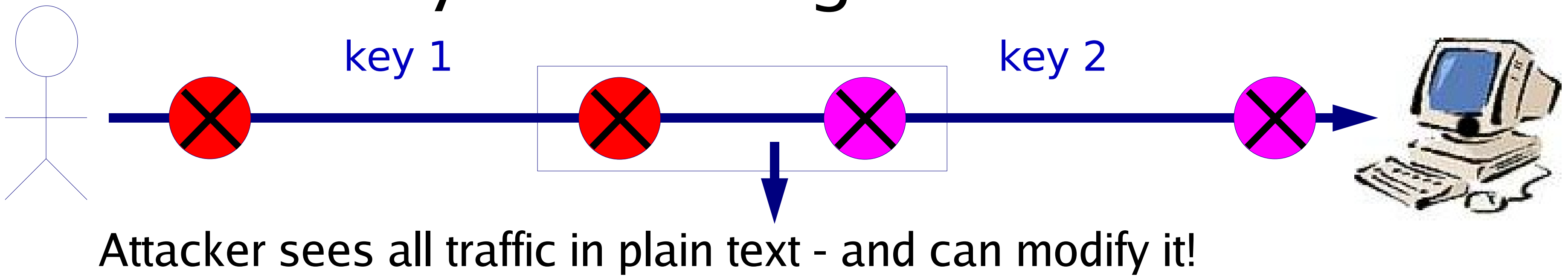
Take a hash of the document and encrypt only that. An encrypted hash is called a "digital signature"

Key Distribution

- The problem of distributing a shared key has been replaced by the problem of distributing the public key.
- Often we want to communicate with a 3rd party whose key we do not know.
- But what if there's someone in between intercepting our traffic.

Key Distribution - MITM

- Passive stiffing is no problem.
- If packets can be intercepted they can substitute a different key.
- The attacker uses a separate key to talk to each party.
- You think you're talking over a secure channel, but in fact you're talking to an attacker.


key 1
key 2
Attacker sees all traffic in plain text - and can modify it!

Key Distribution – digital certificates

- Digital certificates can solve the Man-In-The-Middle problem.
- I Have no knowledge of the remote side's key.
- But a party that I already trust can vouch for them.
- The trusted third-party can sign a certificate that contains the remote sides name and public key.
- I can validate the signature on the certificate using the public key of the party I already trust.

The TLS(SSL) approach

- I generate a private key on my system.
- I send my public key plus my identity in the form of a certificate signing request (CSR) to a certificate authority (CA).
- The CA performs some sort of validation to insure that I am who I say I am.
- They sign a certificate containing my public key, my domain name, and and expiration date.
- I install the certificate on my server.

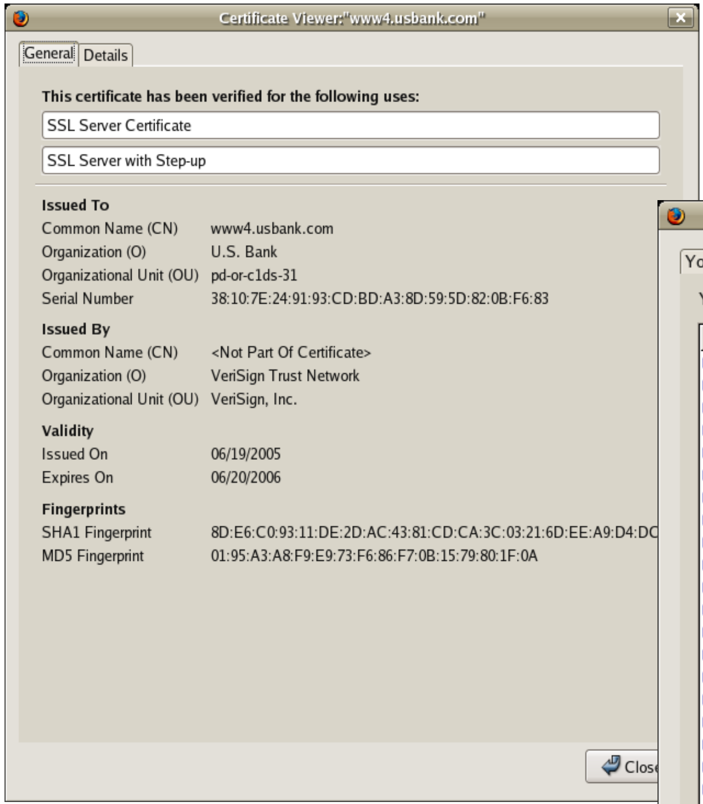
TLS – part 2

- When a client connects with SSL or TLS
  - They negotiate an encrypted session during which they learn the server's public key.
  - The server sends them the certificate
  - They validate the certificate using the CA's public key stored in a keyring on their machine.
  - If the certificate is valid, the domain name matches the domain in the cert and the expiration date has not passed, the client knows the connection is secure.

TLS - problems

- The security of SSL and TLS depends on:
  - The private key remaining secure on your server.
  - The CA's public key being installed on the client.
  - The CA being well managed.
  - The CA being trustworthy.

CA Cert - examples



This certificate has been verified for the following uses:

SSL Server Certificate

SSL Server with Step-up

Issued To

Common Name (CN)	www4.usbank.com
Organization (O)	U.S. Bank
Organizational Unit (OU)	pd-or-c1ds-31
Serial Number	38:10:7E:24:91:93:CD:BD:A3:8D:59:5D:82:0B:F6:83

Issued By

Common Name (CN)	<Not Part Of Certificate>
Organization (O)	VeriSign Trust Network
Organizational Unit (OU)	VeriSign, Inc.

Validity

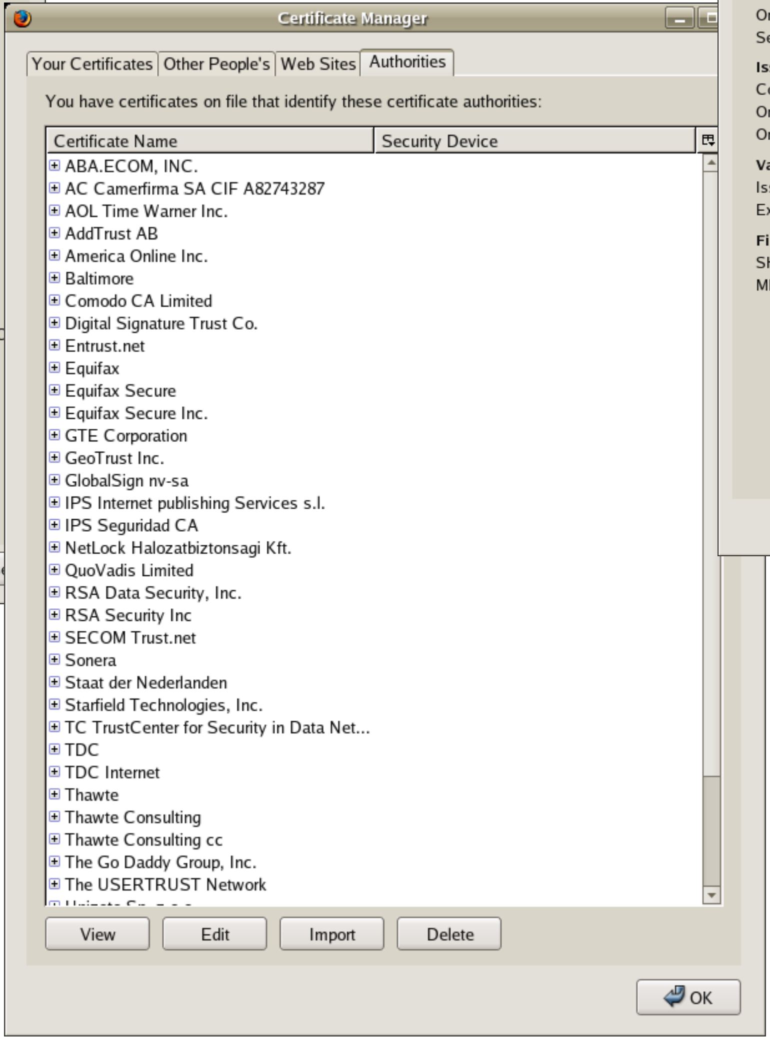
Issued On	06/19/2005
Expires On	06/20/2006

Fingerprints

SHA1 Fingerprint	8D:E6:C0:93:11:DE:2D:AC:43:81:CD:CA:3C:03:21:6D:EE:A9:D4:DC
MD5 Fingerprint	01:95:A3:A8:F9:E9:73:F6:86:F7:0B:15:79:80:1F:0A

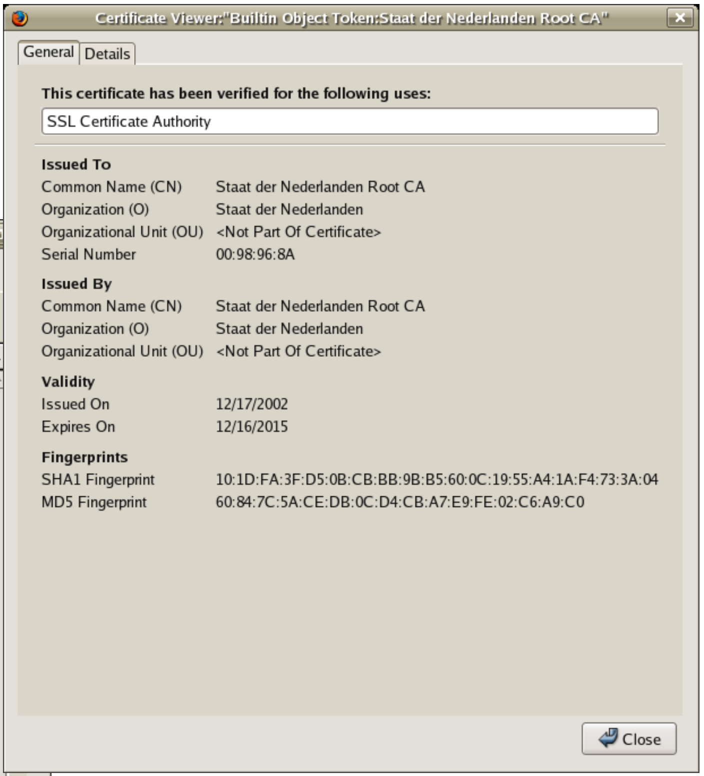
A CA signed cert

CA keyring in your browser



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
ABA.ECOM, INC.	
AC Camerfirma SA CIF A82743287	
AOL Time Warner Inc.	
AddTrust AB	
America Online Inc.	
Baltimore	
Comodo CA Limited	
Digital Signature Trust Co.	
Entrust.net	
Equifax	
Equifax Secure	
Equifax Secure Inc.	
GTE Corporation	
GeoTrust Inc.	
GlobalSign nv-sa	
IPS Internet publishing Services s.l.	
IPS Seguridad CA	
NetLock Halozatbiztonsagi Kft.	
QuoVadis Limited	
RSA Data Security, Inc.	
RSA Security Inc	
SECOM Trust.net	
Sonera	
Staat der Nederlanden	
Starfield Technologies, Inc.	
TC TrustCenter for Security in Data Net...	
TDC	
TDC Internet	
Thawte	
Thawte Consulting	
Thawte Consulting cc	
The Go Daddy Group, Inc.	
The USERTRUST Network	


Certificate Viewer:"Builtin Object Token:Staat der Nederlanden Root CA"
General
Details
Close

This certificate has been verified for the following uses:

SSL Certificate Authority

Issued To

Common Name (CN)	Staat der Nederlanden Root CA
Organization (O)	Staat der Nederlanden
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	00:98:96:8A

Issued By

Common Name (CN)	Staat der Nederlanden Root CA
Organization (O)	Staat der Nederlanden
Organizational Unit (OU)	<Not Part Of Certificate>

Validity

Issued On	12/17/2002
Expires On	12/16/2015

Fingerprints

SHA1 Fingerprint	10:1D:FA:3F:D5:0B:CB:BB:9B:B5:60:0C:19:55:A4:1A:F4:73:3A:04
MD5 Fingerprint	60:84:7C:5A:CE:DB:0C:D4:CB:A7:E9:FE:02:C6:A9:C0

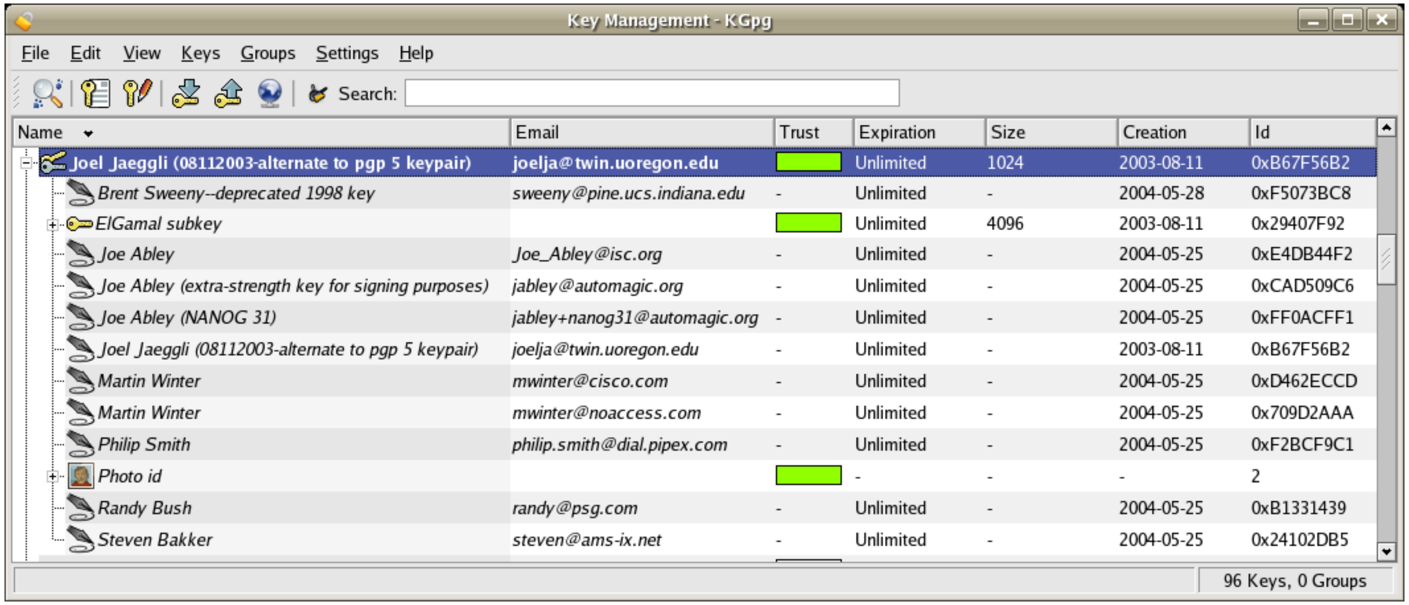
A CA certificate

The PGP approach

- We don't trust anyone (especially big corporate monopolies) expect people that we know.
- You sign the key's of people you know in order to vouch for them.
- Other people can choose to trust your signature as much as they trust you.
- A distributed web of trust is created.

PGP Examples

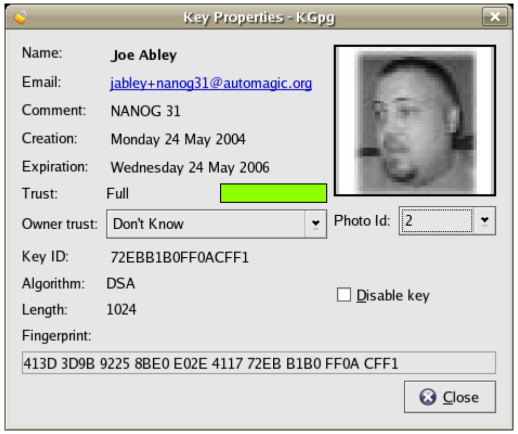
Joel's key signed by others



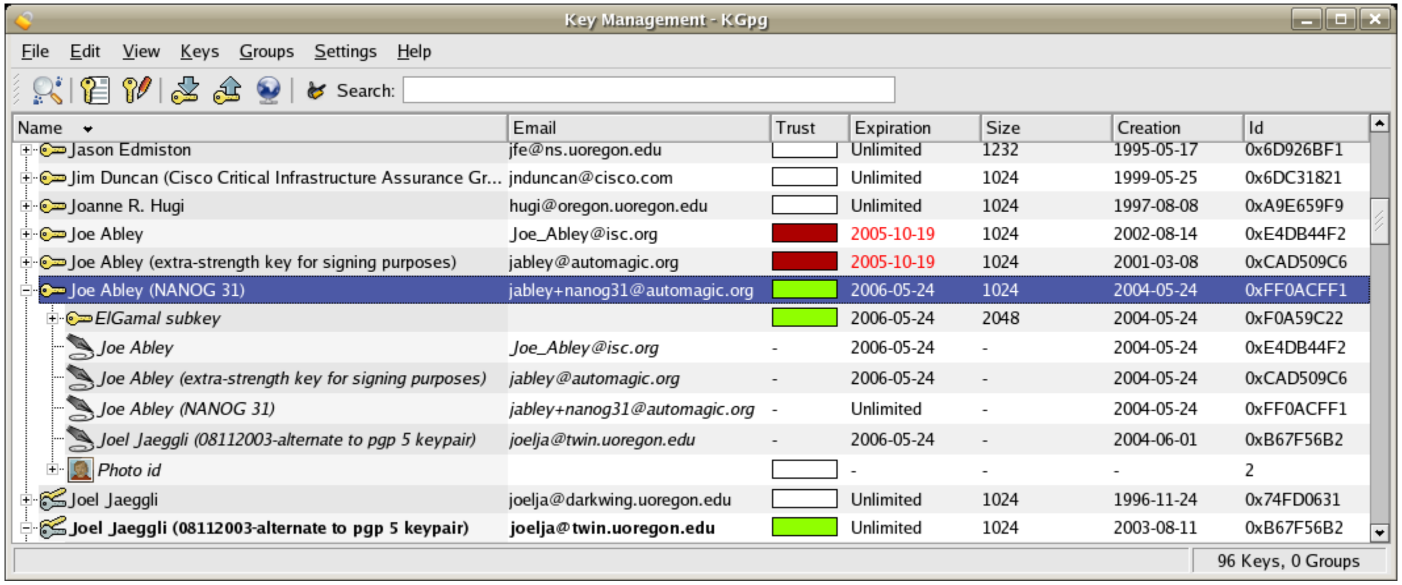
Name	Email	Trust	Expiration	Size	Creation	Id
Joel Jaeggli (08112003-alternate to pgp 5 keypair)	joelja@twin.uoregon.edu	🟩	Unlimited	1024	2003-08-11	0xB67F56B2
Brent Sweeny--deprecated 1998 key	sweeny@pine.ucs.indiana.edu	-	Unlimited	-	2004-05-28	0xF5073BC8
ElGamal subkey		🟩	Unlimited	4096	2003-08-11	0x29407F92
Joe Abley	Joe_Abley@isc.org	-	Unlimited	-	2004-05-25	0xE4DB44F2
Joe Abley (extra-strength key for signing purposes)	jabley@automagic.org	-	Unlimited	-	2004-05-25	0xCAD509C6
Joe Abley (NANOG 31)	jabley+nanog31@automagic.org	-	Unlimited	-	2004-05-25	0xFF0ACFF1
Joel Jaeggli (08112003-alternate to pgp 5 keypair)	joelja@twin.uoregon.edu	-	Unlimited	-	2003-08-11	0xB67F56B2
Martin Winter	mwinter@cisco.com	-	Unlimited	-	2004-05-25	0xD462ECCD
Martin Winter	mwinter@noaccess.com	-	Unlimited	-	2004-05-25	0x709D2AAA
Philip Smith	philip.smith@dial.pipex.com	-	Unlimited	-	2004-05-25	0xF2BCF9C1
Photo id		🟩	-	-	-	2
Randy Bush	randy@psg.com	-	Unlimited	-	2004-05-25	0xB1331439
Steven Bakker	steven@ams-ix.net	-	Unlimited	-	2004-05-25	0x24102DB5

96 Keys, 0 Groups

Joe's key signed by Joel



Name:	Joe Abley
Email:	jabley+nanog31@automagic.org
Comment:	NANOG 31
Creation:	Monday 24 May 2004
Expiration:	Wednesday 24 May 2006
Trust:	Full 🟩
Owner trust:	Don't Know
Photo Id:	2
Key ID:	72EBB1B0FF0ACFF1
Algorithm:	DSA
Length:	1024
Disable key	☐
Fingerprint:	413D 3D9B 9225 8BE0 E02E 4117 72EB B1B0 FF0A CFF1



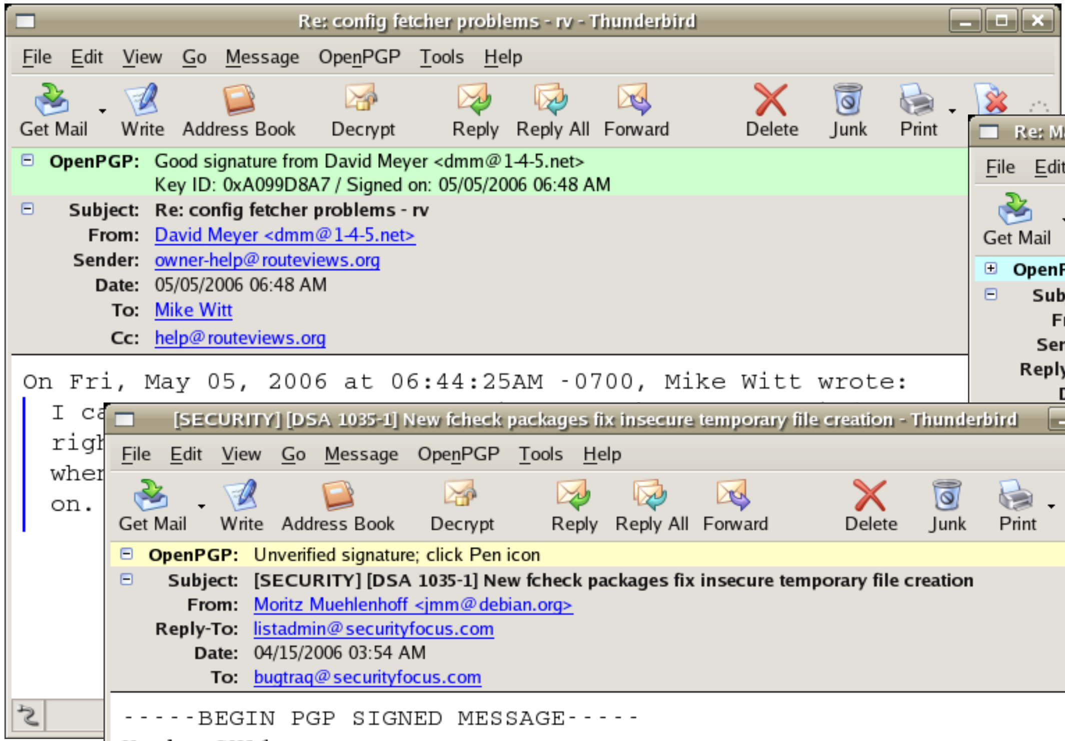
Name	Email	Trust	Expiration	Size	Creation	Id
Jason Edmiston	jfe@ns.uoregon.edu	⬜	Unlimited	1232	1995-05-17	0x6D926BF1
Jim Duncan (Cisco Critical Infrastructure Assurance Gr...	jnduncan@cisco.com	⬜	Unlimited	1024	1999-05-25	0x6DC31821
Joanne R. Hugi	hugi@oregon.uoregon.edu	⬜	Unlimited	1024	1997-08-08	0xA9E659F9
Joe Abley	Joe_Abley@isc.org	🟥	2005-10-19	1024	2002-08-14	0xE4DB44F2
Joe Abley (extra-strength key for signing purposes)	jabley@automagic.org	🟥	2005-10-19	1024	2001-03-08	0xCAD509C6
Joe Abley (NANOG 31)	jabley+nanog31@automagic.org	🟩	2006-05-24	1024	2004-05-24	0xFF0ACFF1
ElGamal subkey		🟩	2006-05-24	2048	2004-05-24	0xF0A59C22
Joe Abley	Joe_Abley@isc.org	-	2006-05-24	-	2004-05-24	0xE4DB44F2
Joe Abley (extra-strength key for signing purposes)	jabley@automagic.org	-	2006-05-24	-	2004-05-24	0xCAD509C6
Joe Abley (NANOG 31)	jabley+nanog31@automagic.org	-	Unlimited	-	2004-05-24	0xFF0ACFF1
Joel Jaeggli (08112003-alternate to pgp 5 keypair)	joelja@twin.uoregon.edu	-	2006-05-24	-	2004-06-01	0xB67F56B2
Photo id		⬜	-	-	-	2
Joel Jaeggli	joelja@darkwing.uoregon.edu	⬜	Unlimited	1024	1996-11-24	0x74FD0631
Joel Jaeggli (08112003-alternate to pgp 5 keypair)	joelja@twin.uoregon.edu	🟩	Unlimited	1024	2003-08-11	0xB67F56B2

96 Keys, 0 Groups

Joe's key detail

PGP Examples 2

Good signature from a trusted key

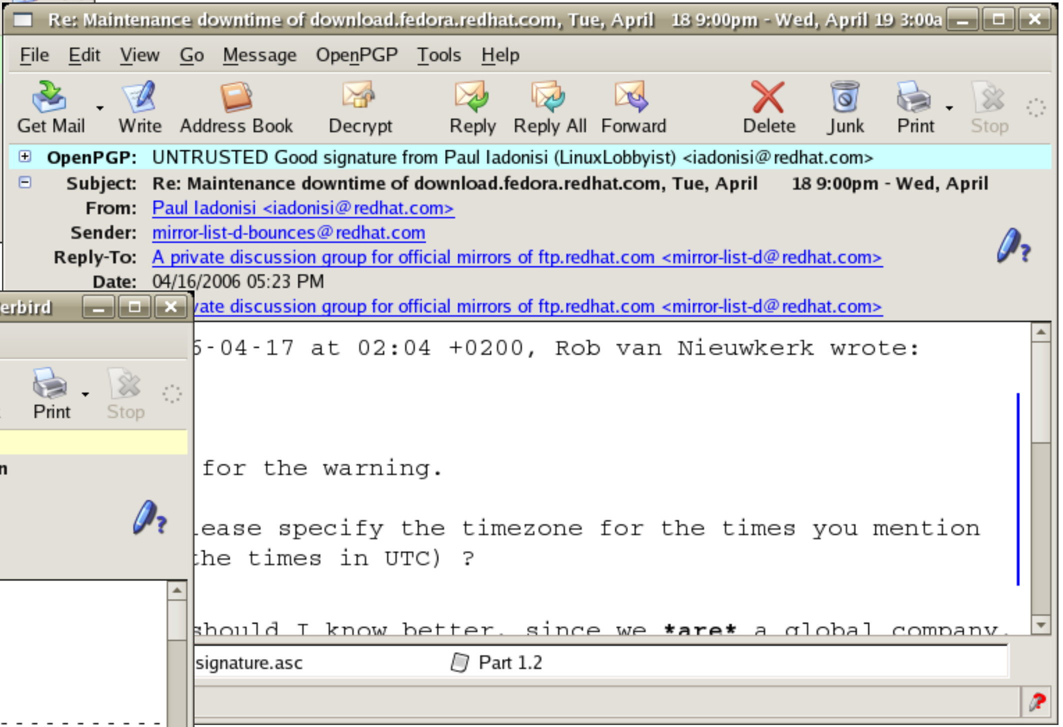

Re: config fetcher problems - rv - Thunderbird
File Edit View Go Message OpenPGP Tools Help
Get Mail Write Address Book Decrypt Reply Reply All Forward Delete Junk Print

OpenPGP: Good signature from David Meyer <dmm@1-4-5.net>
Key ID: 0xA099D8A7 / Signed on: 05/05/2006 06:48 AM

Subject: Re: config fetcher problems - rv
From: David Meyer <dmm@1-4-5.net>
Sender: owner-help@routeviews.org
Date: 05/05/2006 06:48 AM
To: Mike Witt
Cc: help@routeviews.org

```
On Fri, May 05, 2006 at 06:44:25AM -0700, Mike Witt wrote:
  I ca
  righ
  when
  on.
```

Good but untrusted signature


Re: Maintenance downtime of download.fedora.redhat.com, Tue, April 18 9:00pm - Wed, April 19 3:00a
File Edit View Go Message OpenPGP Tools Help
Get Mail Write Address Book Decrypt Reply Reply All Forward Delete Junk Print Stop

OpenPGP: UNTRUSTED Good signature from Paul Iadonisi (LinuxLobbyist) <iadonisi@redhat.com>

Subject: Re: Maintenance downtime of download.fedora.redhat.com, Tue, April 18 9:00pm - Wed, April
From: Paul Iadonisi <iadonisi@redhat.com>
Sender: mirror-list-d-bounces@redhat.com
Reply-To: A private discussion group for official mirrors of ftp.redhat.com <mirror-list-d@redhat.com>
Date: 04/16/2006 05:23 PM
ate discussion group for official mirrors of ftp.redhat.com <mirror-list-d@redhat.com>

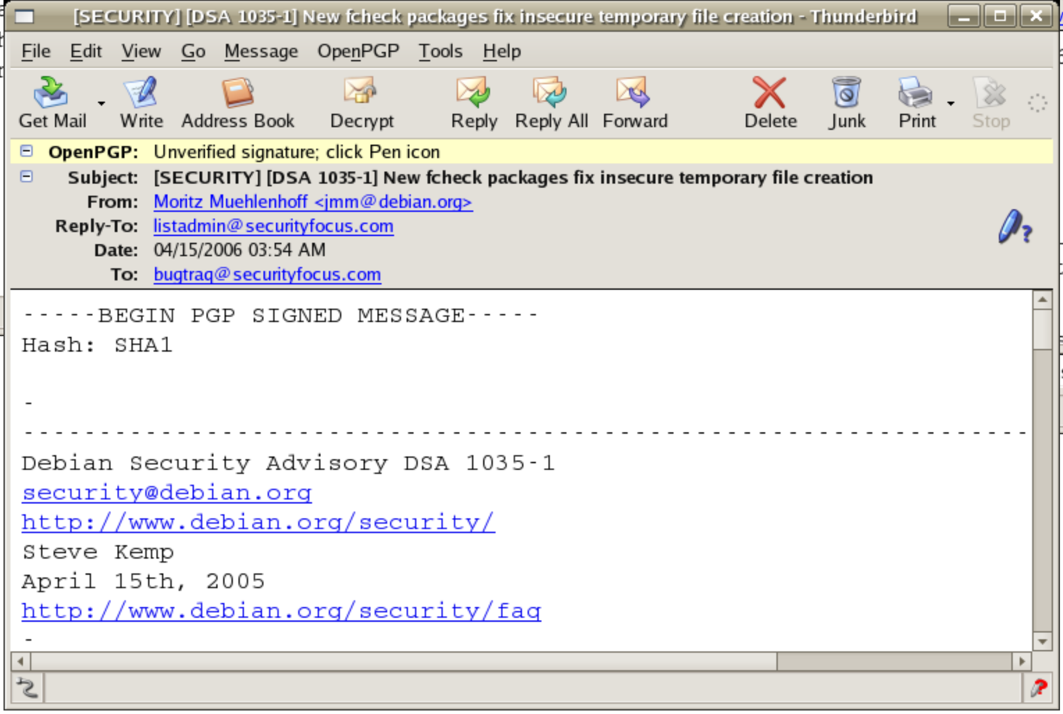
```
6-04-17 at 02:04 +0200, Rob van Nieuwkerk wrote:

for the warning.

lease specify the timezone for the times you mention
the times in UTC) ?

should I know better, since we *are* a global company.
```

signature.asc Part 1.2


[SECURITY] [DSA 1035-1] New fcheck packages fix insecure temporary file creation - Thunderbird
File Edit View Go Message OpenPGP Tools Help
Get Mail Write Address Book Decrypt Reply Reply All Forward Delete Junk Print Stop

OpenPGP: Unverified signature; click Pen icon

Subject: [SECURITY] [DSA 1035-1] New fcheck packages fix insecure temporary file creation
From: Moritz Muehlenhoff <jmm@debian.org>
Reply-To: listadmin@securityfocus.com
Date: 04/15/2006 03:54 AM
To: bugtraq@securityfocus.com

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

-
------------------------------------------------------------------------
Debian Security Advisory DSA 1035-1
security@debian.org
http://www.debian.org/security/
Steve Kemp
April 15th, 2005
http://www.debian.org/security/faq
-
```

Unverified signature (no key)

The SSH approach

- The first time a connection is made to a host the key has to be accepted.
  - Key is stored in ~/.ssh/known_hosts or the equivalent.
- On the next connection, if the key presented is different then maybe an attacker is a work.
  - Maybe the host just has a new key?

The SSH approach - part 2

- Use public key cryptography to prove user identity.
  - Generate a public/private key pair.
  - Install the private key locally and the public key on the remote host.
  - Connect from local host to remote, no password required, it has been replaced by something vastly stronger.
  - Still a good idea to protect the private a key with a password, but that password never has to touch the network, never has to be stored in any form on a remote system and is not subject to the whims of policy on the remote system.

Where can you apply these cryptographic methods

- At the link layer
  - PPP encryption
- At the network layer
  - IPSEC
- At the transport layer
  - TLS/SSL
- At the application layer
  - SSH
  - PGP and GPG
  - System integrity checking, package management, etc

Cryptography isn't really an option!

- Cryptographically enhanced systems offer the protection you need now, to protect your systems, your users and their data.
- Important first steps:
  - Use SSH exclusively for system administration.
  - Use SCP/SFTP for all file transfers not wrapped in SSL or done anonymously.
  - Install POP3/IMAP and SMTP servers with TLS support.
  - Use SSL/HTTPS for applications involving passwords or proprietary data (like webmail applications).

Bibliography

- Applied cryptography second edition
  - Schneier, Bruce., 1996. ISBN 0471117099
- PGP passphrase faq
  - http://www.iusmentis.com/security/passphrasefaq/
- Wikipedia cryptography root article
  - http://en.wikipedia.org/wiki/Cryptography