Introduction to PGP

Michael Howe

michael.howe@it.ox.ac.uk

Infrastructure and Hosting Team
IT Services, Universisty of Oxford

June 24th, 2016

Introduction to PGP

Michael Howe

michael.howe@it.ox.ac.uk

Infrastructure and Hosting Team
IT Services, Universisty of Oxford

June 24th, 2016

What this talk will cover

- What is PGP?
- Why might you use it?
- How does it work?
- How might you use it?
- No keysigning in this talk

So what is PGP?

> *Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication.*

`https://en.wikipedia.org/wiki/Pretty_Good_Privacy`

What do we mean by PGP?

- PGP
- GPG (or GnuPG, or GNU Privacy Guard)
- OpenPGP (RFC 4880, RFC 2440)

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography
1991 - US Senate Bill 266

> *"It is the sense of Congress that providers of electronic communications services and manufacturers of electronic communications service equipment shall insure that communications systems permit the Government to obtain the plain text contents of voice, data, and other communications when appropriately authorized by law."*

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

1996 - Case against Phil Zimmerman dropped

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

1996 - Case against Phil Zimmerman dropped

1997 - GnuPG first released

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

1996 - Case against Phil Zimmerman dropped

1997 - GnuPG first released

1999 - Why Johnny can't encrypt: a usability evaluation of PGP 5.0

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

1996 - Case against Phil Zimmerman dropped

1997 - GnuPG first released

1999 - Why Johnny can't encrypt: a usability evaluation of PGP 5.0

2013 - Edward Snowden

A brief history of PGP

1977 - Whitfield Diffie, Martin Hellman and Ralph Merkle develop and patent public key cryptography

1991 - US Senate Bill 266

1991 - Phil Zimmerman develops PGP

1993 - US Government starts a criminal investigation

1995 - PGP: Source Code and Internals

1996 - Case against Phil Zimmerman dropped

1997 - GnuPG first released

1999 - Why Johnny can't encrypt: a usability evaluation of PGP 5.0

2013 - Edward Snowden

2015 - Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP Client

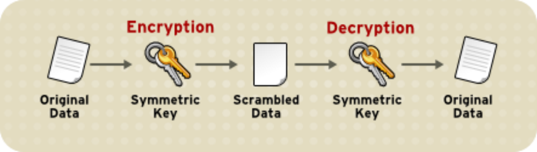
How it works: the building blocks

The building blocks

- Symmetric cryptography
- Asymmetric (public key) cryptography
- Hashing

Symmetric cryptography

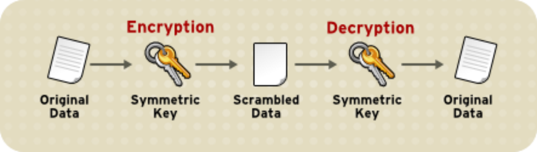
- The same key is used for encryption and decryption



- This has been with us for centuries...

Symmetric cryptography

- The same key is used for encryption and decryption



- This has been with us for centuries...

Plain:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:	XYZABCDEFGHIJKLMNOPQRSTUVW

Using to encrypt:

Plaintext	WELCOME TO THE ICTF CONFERENCE
Ciphertext	TBIZLJB QL QEB FZQC ZLKCBOBKZB

Symmetric encryption

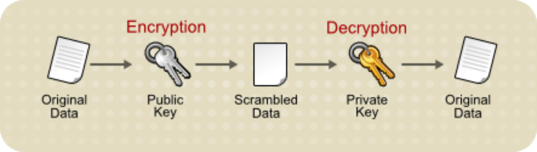
- Examples: AES, CAST5, Blowfish, Camellia, IDEA

Symmetric encryption

- Examples: AES, CAST5, Blowfish, Camellia, IDEA
- Problem: key distribution

Asymmetric cryptography

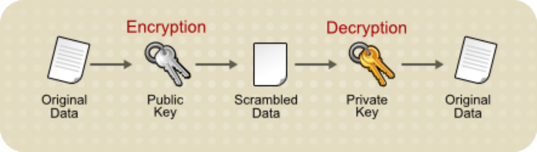
- Different (but linked) keys used for encryption and decryption: a `private` and a `public` key



- Only been around ≈ 50 years
- Uses mathematical properties to ensure security (eg prime number factorisation, discrete logarithm computation)

Asymmetric cryptography

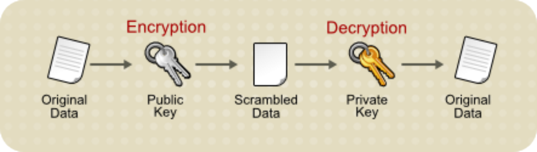
- Different (but linked) keys used for encryption and decryption: a `private` and a `public` key



- Only been around ≈ 50 years
- Uses mathematical properties to ensure security (eg prime number factorisation, discrete logarithm computation)
- Solves the key-sharing problem!
- But slower than symmetric encryption (larger keys)

Asymmetric cryptography

- Different (but linked) keys used for encryption and decryption: a `private` and a `public` key



- Only been around ≈ 50 years
- Uses mathematical properties to ensure security (eg prime number factorisation, discrete logarithm computation)
- Solves the key-sharing problem!
- But slower than symmetric encryption (larger keys)
- Examples: RSA, DSA, ElGamal, ECDSA

Asymmetric cryptography

Not quite as simple - but can be implemented in 3 lines of perl...

```
#!/bin/perl -sp0777i<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<j]dsj
$/=unpack(’H*’,$_);$_=‘echo 16dio\U$k"SK$/SM$n\EsN0p[lN*1
lK[d2%Sa2/d0$^Ixp"|dc‘;s/\W//g;$_=pack(’H*’,/((..)*)$/)
```

Usage:

```
rsa -k=public-key -n=rsa-modulus < msg > msg.rsa
rsa -k=private-key -n=rsa-modulus < msg.rsa > msg.out
```

Hashing

- Takes data of an arbitrary size (`message`) and maps it to a fixed size (`digest`)
- One-way

Hashing

- Takes data of an arbitrary size (`message`) and maps it to a fixed size (`digest`)
- One-way
- Useful to check that a message hasn't been modified

Hashing

- Takes data of an arbitrary size (`message`) and maps it to a fixed size (`digest`)
- One-way
- Useful to check that a message hasn't been modified

HELLO WORLD	`361fadf1c712e812d198c4cab5712a79`
HALLO WORLD	`fbb80bf0d72fb5ebf03c776db4e80fe8`

Hashing

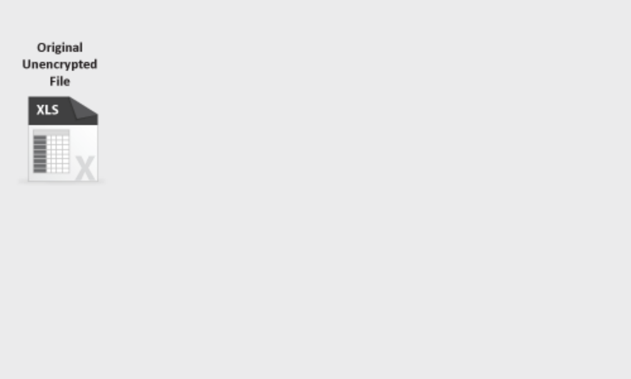
- Takes data of an arbitrary size (`message`) and maps it to a fixed size (`digest`)
- One-way
- Useful to check that a message hasn't been modified

HELLO WORLD	`361fadf1c712e812d198c4cab5712a79`
HALLO WORLD	`fbb80bf0d72fb5ebf03c776db4e80fe8`

- Examples: MD5, SHA-1, SHA-512

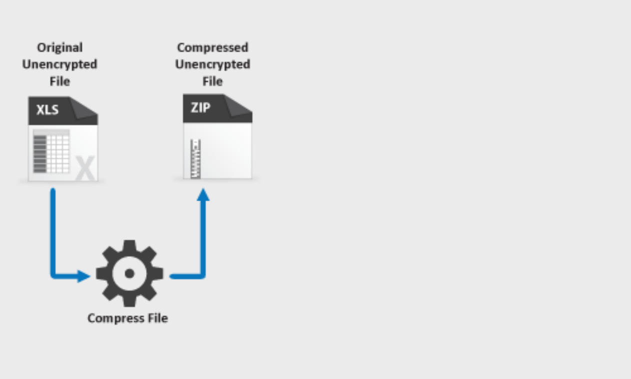
Putting it all together

PGP uses all of thase building blocks - symmetric and asymmetric encryption, and hashing (plus compression).



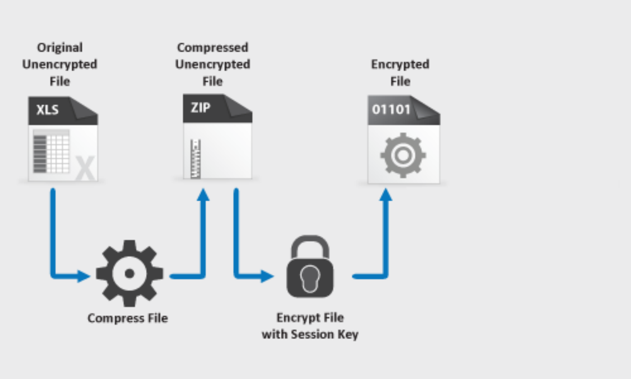
Putting it all together

PGP uses all of thase building blocks - symmetric and asymmetric encryption, and hashing (plus compression).



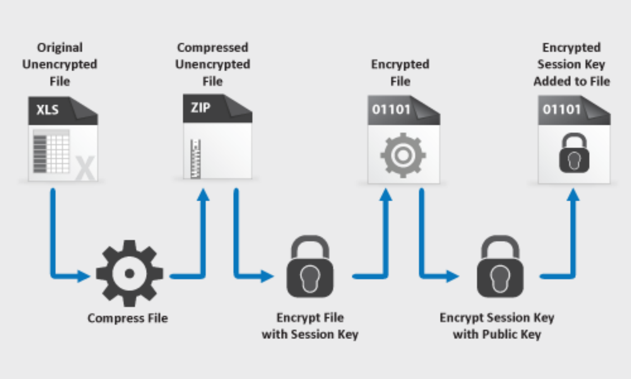
Putting it all together

PGP uses all of thase building blocks - symmetric and asymmetric encryption, and hashing (plus compression).


Original Unencrypted File
XLS
Compressed Unencrypted File
ZIP
Encrypted File
01101
Compress File
Encrypt File with Session Key

Putting it all together

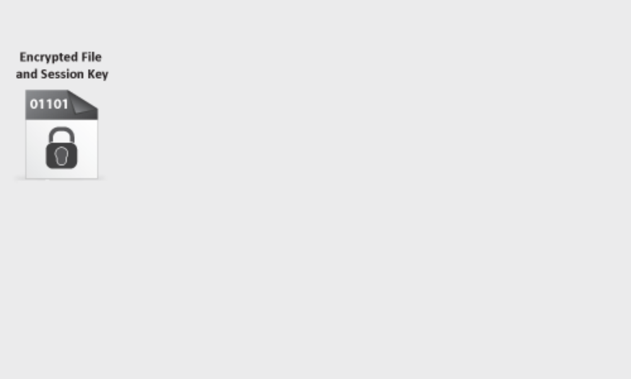
PGP uses all of thase building blocks - symmetric and asymmetric encryption, and hashing (plus compression).



Original Unencrypted File
XLS
Compressed Unencrypted File
ZIP
Encrypted File
01101
Encrypted Session Key Added to File
01101
Compress File
Encrypt File with Session Key
Encrypt Session Key with Public Key

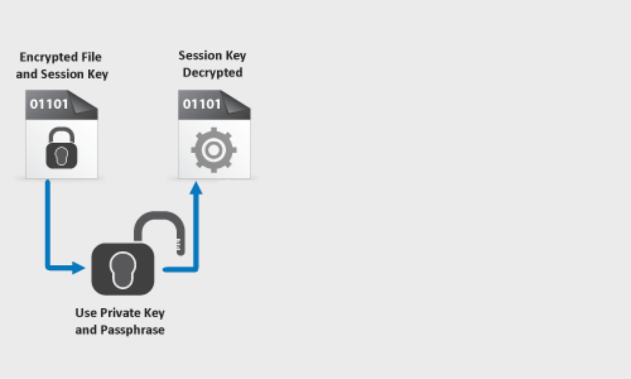
Putting it all together

Decryption is the same, just in reverse



Putting it all together

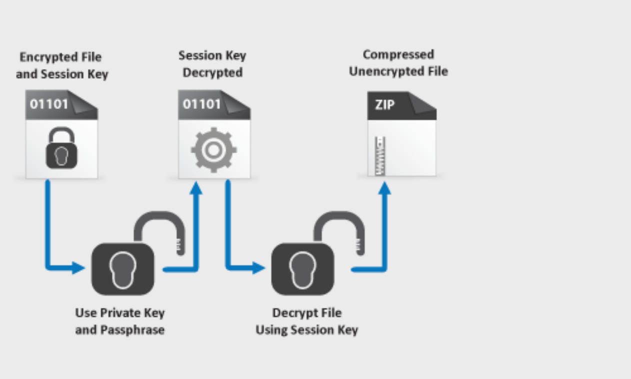
Decryption is the same, just in reverse



Encrypted File and Session Key
01101
Session Key Decrypted
01101
Use Private Key and Passphrase

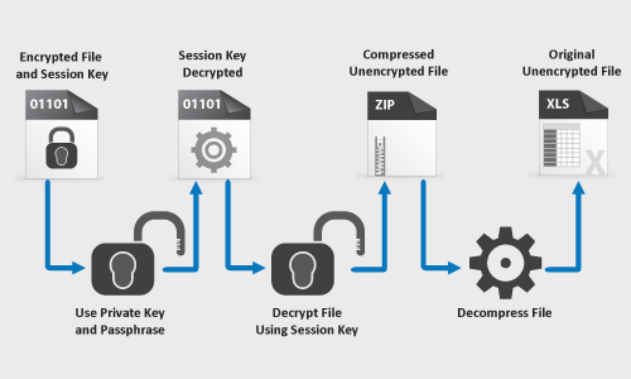
Putting it all together

Decryption is the same, just in reverse


Encrypted File and Session Key
01101
Session Key Decrypted
01101
Compressed Unencrypted File
ZIP
Use Private Key and Passphrase
Decrypt File Using Session Key

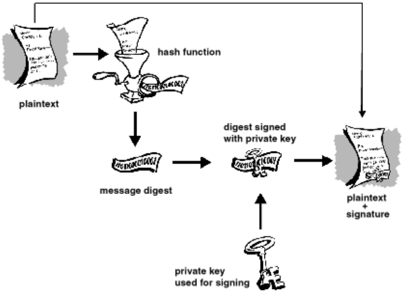
Putting it all together

Decryption is the same, just in reverse


Encrypted File and Session Key
01101
Session Key Decrypted
01101
Compressed Unencrypted File
ZIP
Original Unencrypted File
XLS
Use Private Key and Passphrase
Decrypt File Using Session Key
Decompress File

Wait, what about the hashing?

Hashing is used to sign messages.



These signed messages can then be used as inputs to the encryption process

Hang on, how about the keys?

- PGP needs a public and private (`secret`) keypair

The private key is a secret

The private key should be kept secret. Only the public key should be shared!

Hang on, how about the keys?

- PGP needs a public and private (`secret`) keypair

The private key is a secret

The private key should be kept secret. Only the public key should be shared!

- When encrypting to someone, you need their public key

Hang on, how about the keys?

- PGP needs a public and private (`secret`) keypair

The private key is a secret

The private key should be kept secret. Only the public key should be shared!

- When encrypting to someone, you need their public key
- GPG uses a 'web of trust' - you need to sign a key yourself (or trust someone else who has signed the key)

Hang on, how about the keys?

- PGP needs a public and private (`secret`) keypair

The private key is a secret

The private key should be kept secret. Only the public key should be shared!

- When encrypting to someone, you need their public key
- GPG uses a 'web of trust' - you need to sign a key yourself (or trust someone else who has signed the key)
- This is what keysigning involves

Identifying and signing keys

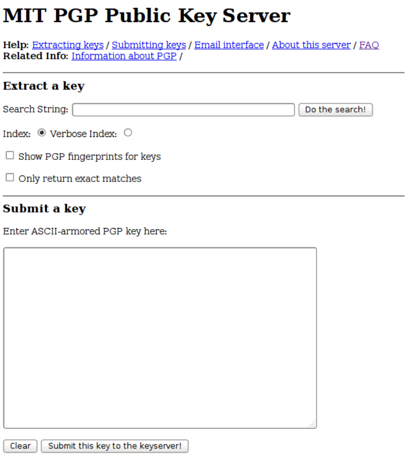
`Michael Howe (Sysdev) <michael.howe@it.ox.ac.uk>`

Short ID: `0x6853C4FA`

Long ID: `0x3B8BC9316853C4FA`

Fingerprint: `810A 24B4 83E8 B097 E7B0 4EA1 3B8B C931 6853 C4FA`

Sharing keys



MIT PGP Public Key Server

Help: Extracting keys / Submitting keys / Email interface / About this server / FAQ
Related Info: Information about PGP /

Extract a key

Search String:

Index: ● Verbose Index: ○

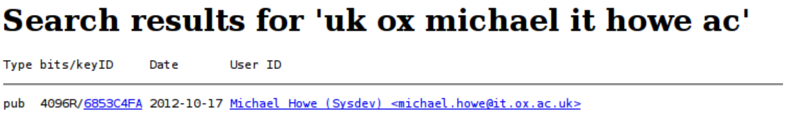
□ Show PGP fingerprints for keys

□ Only return exact matches

Submit a key

Enter ASCII-armored PGP key here:

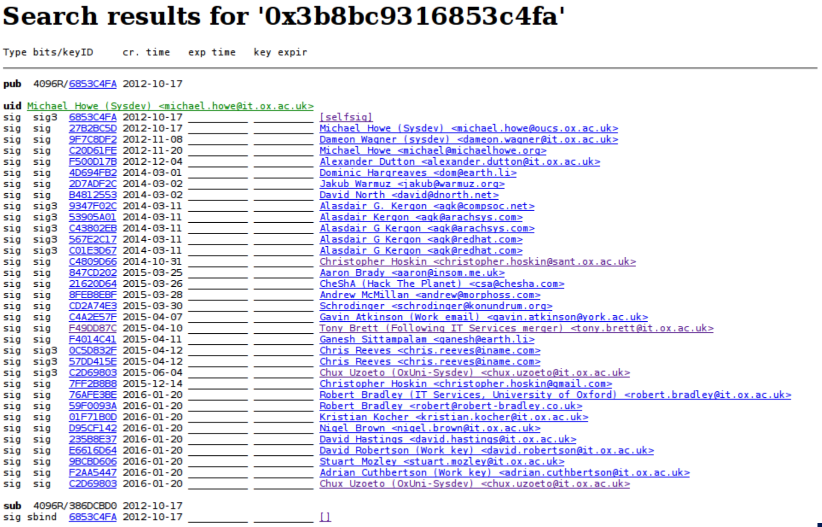
Sharing keys



Search results for 'uk ox michael it howe ac'

```
Type bits/keyID     Date       User ID
pub  4096R/6853C4FA 2012-10-17 Michael Howe (Sysdev) <michael.howe@it.ox.ac.uk>
```

Sharing keys



Search results for '0x3b8bc9316853c4fa'

```
Type bits/keyID     cr. time   exp time   key expir
-------------------------------------------------------------------------------

pub  4096R/6853C4FA 2012-10-17

uid Michael Howe (Sysdev) <michael.howe@it.ox.ac.uk>
sig  sig3  6853C4FA 2012-10-17 __________ __________ [selfsig]
sig  sig   27B2BC5D 2012-10-17 __________ __________ Michael Howe (Sysdev) <michael.howe@oucs.ox.ac.uk>
sig  sig   9F7C8DF2 2012-11-08 __________ __________ Dameon Wagner (sysdev) <dameon.wagner@it.ox.ac.uk>
sig  sig   C20D61FE 2012-11-20 __________ __________ Michael Howe <michael@michaelhowe.org>
sig  sig   F500D17B 2012-12-04 __________ __________ Alexander Dutton <alexander.dutton@it.ox.ac.uk>
sig  sig   4D694FB2 2014-03-01 __________ __________ Dominic Hargreaves <dom@earth.li>
sig  sig   2D7ADF2C 2014-03-02 __________ __________ Jakub Warmuz <jakub@warmuz.org>
sig  sig   B4812553 2014-03-02 __________ __________ David North <david@dnorth.net>
sig  sig3  9347F02C 2014-03-11 __________ __________ Alasdair G. Kergon <agk@compsoc.net>
sig  sig3  53905A01 2014-03-11 __________ __________ Alasdair Kergon <agk@arachsys.com>
sig  sig3  C43802EB 2014-03-11 __________ __________ Alasdair G Kergon <agk@arachsys.com>
sig  sig3  567E2C17 2014-03-11 __________ __________ Alasdair G Kergon <agk@redhat.com>
sig  sig3  C01E3D67 2014-03-11 __________ __________ Alasdair G Kergon <agk@redhat.com>
sig  sig   C4809D66 2014-10-31 __________ __________ Christopher Hoskin <christopher.hoskin@sant.ox.ac.uk>
sig  sig   847CD202 2015-03-25 __________ __________ Aaron Brady <aaron@insom.me.uk>
sig  sig   21620D64 2015-03-26 __________ __________ CheShA (Hack The Planet) <csa@chesha.com>
sig  sig   8FEB8EBF 2015-03-28 __________ __________ Andrew McMillan <andrew@morphoss.com>
sig  sig   CD2A74E3 2015-03-30 __________ __________ Schrodinger <schrodinger@konundrum.org>
sig  sig   C4A2E57F 2015-04-07 __________ __________ Gavin Atkinson (Work email) <gavin.atkinson@york.ac.uk>
sig  sig   F49DD87C 2015-04-10 __________ __________ Tony Brett (Following IT Services merger) <tony.brett@it.ox.ac.uk>
sig  sig   F4014C41 2015-04-11 __________ __________ Ganesh Sittampalam <ganesh@earth.li>
sig  sig3  0C5D832F 2015-04-12 __________ __________ Chris Reeves <chris.reeves@iname.com>
sig  sig3  57DD415E 2015-04-12 __________ __________ Chris Reeves <chris.reeves@iname.com>
sig  sig3  C2D69803 2015-06-04 __________ __________ Chux Uzoeto (OxUni-Sysdev) <chux.uzoeto@it.ox.ac.uk>
sig  sig   7FF2B8B8 2015-12-14 __________ __________ Christopher Hoskin <christopher.hoskin@gmail.com>
sig  sig   76AFE3BE 2016-01-20 __________ __________ Robert Bradley (IT Services, University of Oxford) <robert.bradley@it.ox.ac.uk>
sig  sig   59F0093A 2016-01-20 __________ __________ Robert Bradley <robert@robert-bradley.co.uk>
sig  sig   01F71B0D 2016-01-20 __________ __________ Kristian Kocher <kristian.kocher@it.ox.ac.uk>
sig  sig   D95CF142 2016-01-20 __________ __________ Nigel Brown <nigel.brown@it.ox.ac.uk>
sig  sig   235B8E37 2016-01-20 __________ __________ David Hastings <david.hastings@it.ox.ac.uk>
sig  sig   E6616D64 2016-01-20 __________ __________ David Robertson (Work key) <david.robertson@it.ox.ac.uk>
sig  sig   9BCBD606 2016-01-20 __________ __________ Stuart Mozley <stuart.mozley@it.ox.ac.uk>
sig  sig   F2AA5447 2016-01-20 __________ __________ Adrian Cuthbertson (Work key) <adrian.cuthbertson@it.ox.ac.uk>
sig  sig   C2D69803 2016-01-20 __________ __________ Chux Uzoeto (OxUni-Sysdev) <chux.uzoeto@it.ox.ac.uk>

sub  4096R/386DCBD0 2012-10-17
sig sbind  6853C4FA 2012-10-17 __________ __________ []
```

Why might you use it?

- Encryption
- Signing

Why might you use it?

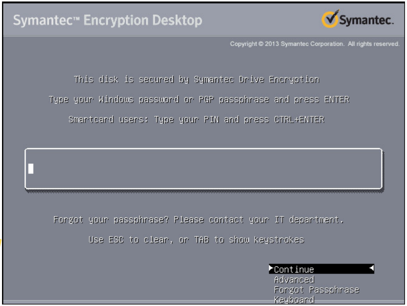
- Encryption
- Signing

Why might you use it?

- Encryption
- Signing

Why might you use it?

- Encryption
- Signing



Symantec™ Encryption Desktop

Copyright © 2013 Symantec Corporation. All rights reserved.

This disk is secured by Symantec Drive Encryption

Type your Windows password or PGP passphrase and press ENTER

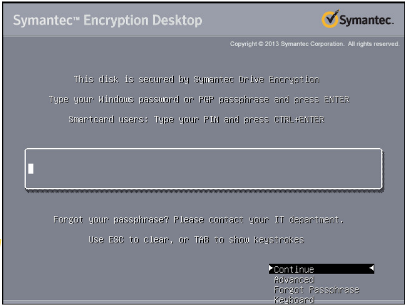
Smartcard users: Type your PIN and press CTRL+ENTER

Forgot your passphrase? Please contact your IT department.

Use ESC to clear, or TAB to show keystrokes

Why might you use it?

- Encryption
- Signing



Symantec™ Encryption Desktop

Symantec.

Copyright © 2013 Symantec Corporation. All rights reserved.

This disk is secured by Symantec Drive Encryption

Type your Windows password or PGP passphrase and press ENTER

Smartcard users: Type your PIN and press CTRL+ENTER

Forgot your passphrase? Please contact your IT department.

Use ESC to clear, or TAB to show keystrokes

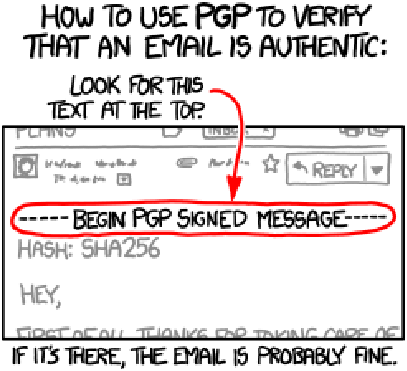
...

Things I use it for

A non-exhaustive list:

- Signing mails
- Signing SSL certificate signing requests
- Signing team-internal Debian packages
- Storing passwords with `pass` (`https://www.passwordstore.org`)
- Sharing passwords with members of my team
- Validating CSRs and Shibboleth metadata requests

How might you use it?


HOW TO USE PGP TO VERIFY THAT AN EMAIL IS AUTHENTIC:
LOOK FOR THIS TEXT AT THE TOP:
PLANS
INBOX
REPLY
-----BEGIN PGP SIGNED MESSAGE-----
HASH: SHA256
HEY,
FIRST OF ALL, THANKS FOR TAKING CARE OF
IF IT'S THERE, THE EMAIL IS PROBABLY FINE.

If you want to be extra safe, check that there's a big block of jumbled characters at the bottom.

http://xkcd.com/1181/

Don't panic!

Despite all that, don't give up yet!

How might you use it?

- Work out what you want to do
  - Encrypt files in transit (eg Oxfile)
  - Assert your identity when communicating with, eg, IT Services

How might you use it?

- Work out what you want to do
  - Encrypt files in transit (eg Oxfile)
  - Assert your identity when communicating with, eg, IT Services
- Start small

How might you use it?

- Work out what you want to do
  - Encrypt files in transit (eg Oxfile)
  - Assert your identity when communicating with, eg, IT Services
- Start small
- Find a friend

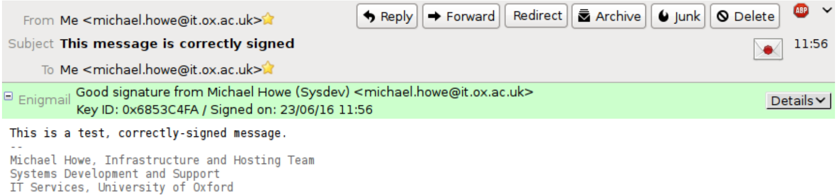
How might you use it?

- Work out what you want to do
  - Encrypt files in transit (eg Oxfile)
  - Assert your identity when communicating with, eg, IT Services
- Start small
- Find a friend
- Know what you're doing before involving non-technical people

Some demonstrations

Here's one I partially prepared earlier...

Using Thunderbird and Enigmail



From Me <michael.howe@it.ox.ac.uk>

Subject **This message is correctly signed**

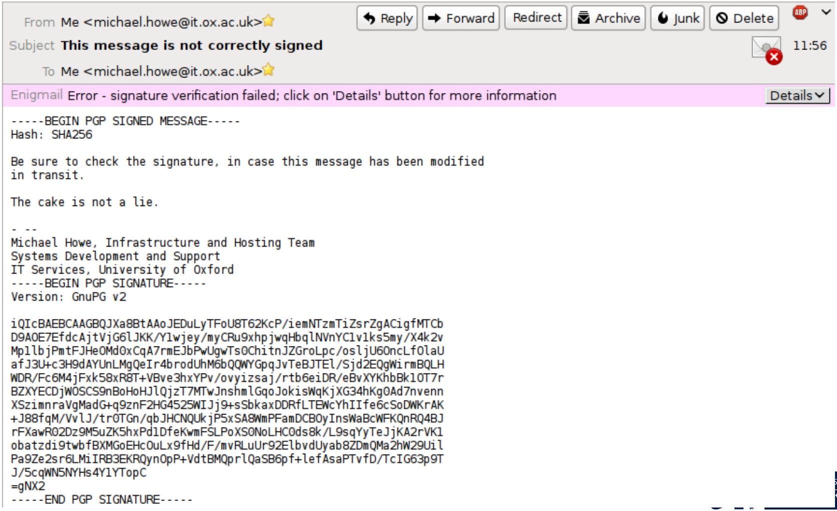
To Me <michael.howe@it.ox.ac.uk>

11:56

Enigmail Good signature from Michael Howe (Sysdev) <michael.howe@it.ox.ac.uk>
Key ID: 0x6853C4FA / Signed on: 23/06/16 11:56

```
This is a test, correctly-signed message.
--
Michael Howe, Infrastructure and Hosting Team
Systems Development and Support
IT Services, University of Oxford
```

Using Thunderbird and Enigmail



From Me <michael.howe@it.ox.ac.uk>

Subject **This message is not correctly signed**

11:56

To Me <michael.howe@it.ox.ac.uk>

Enigmail Error - signature verification failed; click on 'Details' button for more information

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Be sure to check the signature, in case this message has been modified
in transit.

The cake is not a lie.

- --
Michael Howe, Infrastructure and Hosting Team
Systems Development and Support
IT Services, University of Oxford
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iQIcBAEBCAAGBQJXa8BtAAoJEDuLyTFoU8T62KcP/iemNTzmTiZsrZgACigfMTCb
D9AOE7EfdcAjtVjG6lJKK/Y1wjey/myCRu9xhpjwqHbqlNVnYC1v1ks5my/X4k2v
Mp1lbjPmtFJHeOMd0xCqA7rmEJbPwUgwTs0ChitnJZGroLpc/osljU6OncLf0laU
afJ3U+c3H9dAYUnLMgQeIr4brodUhM6bQQWYGpqJvTeBJTEl/Sjd2EQgWirmBQLH
WDR/Fc6M4jFxk58xR8T+VBve3hxYPv/ovyizsaj/rtb6eiDR/eBvXYKhbBk1OT7r
BZXYECDjWOSCS9nBoHoHJlQjzT7MTwJnshmlGqoJokisWqKjXG34hKg0Ad7nvenn
XSzimnraVgMadG+q9znF2HG4525WIJj9+sSbkaxDDRfLTEWcYhIIfe6cSoDWKrAK
+J88fqM/VvlJ/tr0TGn/qbJHCNQUkjP5xSA8WmPFamDCBOyInsWaBcWFKQnRQ4BJ
rFXawR02Dz9M5uZK5hxPd1DfeKwmFSLPoXS0NoLHC0ds8k/L9sqYyTeJjKA2rVK1
obatzdi9twbfBXMGoEHcOuLx9fHd/F/mvRLuUr92ElbvdUyab8ZDmQMa2hW29Uil
Pa9Ze2sr6LMiIRB3EKRQynOpP+VdtBMQprlQaSB6pf+lefAsaPTvfD/TcIG63p9T
J/5cqWN5NYHs4Y1YTopC
=gNX2
-----END PGP SIGNATURE-----
```

Phew!

A whistlestop tour:

- How PGP came to be
- How it works
- How and why it's used, and you might consider using it

Phew!

A whistlestop tour:

- How PGP came to be
- How it works
- How and why it's used, and you might consider using it
- Anyone interested in keysigning?

Useful resources

Applications

GnuPG: `https://www.gnupg.org/`
GPG4Win: `https://www.gpg4win.org/`
Enigmail: `https://www.enigmail.net/`

Tutorials

GPG on Windows:
`https://ssd.eff.org/en/module/how-use-pgp-windows`
GPG on Linux: `https://help.ubuntu.com/community/GnuPrivacyGuardHowto`

Papers

Why Johnny Can't Encrypt:
`http://dl.acm.org/citation.cfm?id=1251435`
Why Johnny Still, Still Can't Encrypt:
`https://arxiv.org/abs/1510.08555`

Me

michael.howe@it.ox.ac.uk

Questions?

Any questions?

Useful resources

Applications

GnuPG: https://www.gnupg.org/
GPG4Win: https://www.gpg4win.org/
Enigmail: https://www.enigmail.net/

Tutorials

GPG on Windows:
https://ssd.eff.org/en/module/how-use-pgp-windows
GPG on Linux: https://help.ubuntu.com/community/GnuPrivacyGuardHowto

Papers

Why Johnny Can't Encrypt:
http://dl.acm.org/citation.cfm?id=1251435
Why Johnny Still, Still Can't Encrypt:
https://arxiv.org/abs/1510.08555

Me

michael.howe@it.ox.ac.uk