Basics of public key cryptography

Big numbers are nice for public key cryptography

Basics of public key cryptography

Big numbers are nice for public key cryptography

- based on number theory and the fact that factoring a product of two primes is an *np* problem

Basics of public key cryptography

Big numbers are nice for public key cryptography

- based on number theory and the fact that factoring a product of two primes is an *np* problem
- given two primes p, q, the time required to factorise pq is proportional to e^N, where N is number of digits in pq (roughly $N = \log pq$).

Basics of public key cryptography

Big numbers are nice for public key cryptography

- based on number theory and the fact that factoring a product of two primes is an *np* problem
- given two primes p, q, the time required to factorise pq is proportional to e^N, where N is number of digits in pq (roughly $N = \log pq$).
- in public key cryptography, we generate primes p and q, who constitute our secret key, the public key is pq

Basics of public key cryptography

Big numbers are nice for public key cryptography

- based on number theory and the fact that factoring a product of two primes is an *np* problem
- given two primes p, q, the time required to factorise pq is proportional to e^N, where N is number of digits in pq (roughly $N = \log pq$).
- in public key cryptography, we generate primes p and q, who constitute our secret key, the public key is pq
- public key can be freely distributed - it takes too long to figure out p and q from it

Basics of public key cryptography

Big numbers are nice for public key cryptography

- based on number theory and the fact that factoring a product of two primes is an *np* problem
- given two primes p, q, the time required to factorise pq is proportional to e^N, where N is number of digits in pq (roughly $N = \log pq$).
- in public key cryptography, we generate primes p and q, who constitute our secret key, the public key is pq
- public key can be freely distributed - it takes too long to figure out p and q from it
- secret key still needs to be kept secret but it is never distributed like secret keys of shared secret systems (i.e. ordinary single key crypto)

Key security

Secret key is you

Key security

Secret key is you

- secret key must be kept safe from others

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable
  - hard disc failures

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable
  - hard disc failures
  - fires

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable
  - hard disc failures
  - fires
  - keep a copy safe

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable
  - hard disc failures
  - fires
  - keep a copy safe
- key exchange only concerns publicly available data - no need to fear eavesdroppers

Key security

Secret key is you

- secret key must be kept safe from others
  - preferably NOT on a machine someone else has root access to
  - some work-related keys an exception to this
  - even personal work-related keys may be exceptions
- secret key must not be lost - it is irreplaceable
  - hard disc failures
  - fires
  - keep a copy safe
- key exchange only concerns publicly available data - no need to fear eavesdroppers
- eavesdroppers replaced by men-in-the-middle

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- everyone’s probably seen this

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- everyone's probably seen this

```
The authenticity of host 'purnukka.ath.cx (82.203.129.58)' can't be established.
DSA key fingerprint is 39:fb:35:c5:b4:87:35:d8:01:e7:2d:71:e6:5a:98:73.
Are you sure you want to continue connecting (yes/no)?
```

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- everyone's probably seen this

```
The authenticity of host 'purnukka.ath.cx (82.203.129.58)' can't be established.
DSA key fingerprint is 39:fb:35:c5:b4:87:35:d8:01:e7:2d:71:e6:5a:98:73.
Are you sure you want to continue connecting (yes/no)?
```

and this

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- everyone's probably seen this

```
The authenticity of host 'purnukka.ath.cx (82.203.129.58)' can't be established.
DSA key fingerprint is 39:fb:35:c5:b4:87:35:d8:01:e7:2d:71:e6:5a:98:73.
Are you sure you want to continue connecting (yes/no)?
```

and this

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA1 host key has just been changed.
The fingerprint for the RSA1 key sent by the remote host is
28:7f:f3:68:6a:84:8e:61:28:e9:55:8d:cc:a5:12:8b.
Please contact your system administrator.
```

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- you never answer yes to these

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- you never answer yes to these
- if you do, you have no security what so ever

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- you never answer yes to these
- if you do, you have no security what so ever
- it is easy to spoof host authenticity if you answer “yes” to either of these

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

ssh

- ssh uses public key cryptography to ensure host identity
- you never answer yes to these
- if you do, you have no security what so ever
- it is easy to spoof host authenticity if you answer “yes” to either of these
- using public key cryptography to authenticate the user as well will alleviate the problem

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

GnuPG

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

GnuPG

- GnuPG uses public key cryptography to digitally sign or encrypt messages

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

GnuPG

- GnuPG uses public key cryptography to digitally sign or encrypt messages
- key trust and authenticity must be maintained

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

GnuPG

- GnuPG uses public key cryptography to digitally sign or encrypt messages
- key trust and authenticity must be maintained
- getting key from a keyserver is not trustworthy

Men in the middle - key exchange scenarios

These are **the point of failure** in public key systems

GnuPG

- GnuPG uses public key cryptography to digitally sign or encrypt messages
- key trust and authenticity must be maintained
- getting key from a keyserver is not trustworthy
- the key you get is not necessarily authentic either

Web of Trust - or How to defend against the Man in the Middle

foo

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host
2. build a Web of Trust

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host
2. build a Web of Trust
   - obtain **signed** keys from trusted parties

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host
2. build a Web of Trust
   - obtain **signed** keys from trusted parties
   - assign different trust levels for diffrent purposes, keys, persons

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host
2. build a Web of Trust
   - obtain **signed** keys from trusted parties
   - assign different trust levels for diffrent purposes, keys, persons
     - at least GnuPG can do this

Web of Trust - or How to defend against the Man in the Middle

foo

1. only obtain public keys first-hand
   - directly from the person
   - directly from the console-screen of the sshd host
2. build a Web of Trust
   - obtain **signed** keys from trusted parties
   - assign different trust levels for diffrent purposes, keys, persons
     - at least GnuPG can do this
   - always check these signatures

GnuPG use

Main commands in GnuPG

GnuPG use

Main commands in GnuPG

- create your own private-public -keypair:

  ```
  gpg -gen-key
  ```

GnuPG use

Main commands in GnuPG

- create your own private-public -keypair:

  ```
  gpg -gen-key
  ```

- sign a message file:

  ```
  gpg -sign [filename]
  ```

GnuPG use

Main commands in GnuPG

- create your own private-public -keypair:

  ```
  gpg -gen-key
  ```

- sign a message file:

  ```
  gpg -sign [filename]
  ```

- verify the signature:

  ```
  gpg -verify [filename]
  ```

GnuPG use

Main commands in GnuPG

- create your own private-public -keypair:

  `gpg -gen-key`

- sign a message file:

  `gpg -sign [filename]`

- verify the signature:

  `gpg -verify [filename]`

- encrypt a message file:

  `gpg -encrypt [filename]`

  Note: this only encrypts the symmetric-cryptosystem key using public-key cryptography. The actual file is crypted using a single-key-system. This is simply a performance issue: using integers of the size of 2^{2048} tends to be a bit slow.

GnuPG use

Main commands in GnuPG

- decrypt a message file:

  ```
  gpg -decrypt [filename]
  ```

GnuPG use

Main commands in GnuPG

- decrypt a message file:

  ```
  gpg -decrypt [filename]
  ```

- import keys from file:

  ```
  gpg -import [filename]
  ```

GnuPG use

Main commands in GnuPG

- decrypt a message file:

  ```
  gpg -decrypt [filename]
  ```

- import keys from file:

  ```
  gpg -import [filename]
  ```

- import keys from default keyserver:

  ```
  gpg -recv-keys key-ids
  ```

GnuPG use

Main commands in GnuPG

- decrypt a message file:

  ```
  gpg -decrypt [filename]
  ```

- import keys from file:

  ```
  gpg -import [filename]
  ```

- import keys from default keyserver:

  ```
  gpg -recv-keys key-ids
  ```

- list your secret key(s) fingerprints:

  ```
  gpg -fingerprint -list-secret-keys
  ```

I am not being paranoid

foo

While I have not met a forged PGP key or seen a Man in the Middle attack with PGP systems, I have seen an attempted Man in the Middle attack with ssh servers.
I even staged my own once to demonstrate! Took about 10 minutes to build the system.