

A talk in 32 parts

Onion

Misc

~~Examples~~

1: Peeling the crypto Onion

Solving problems to find ... more problems

Need to keep secrets

- Secrecy \sim Privacy
- Not just for bad-guys
- But ... need to share them as well

Whispering

- Alice meets Bob, whispers the secret
- But ... inconvenient

Trusted courier

- James Bond, or Postman Pat
- But ... *trusted* vs. *trustworthy*
- ... slow

Encode

- Pre-arrange messages, assign code-words
- Allows use of un-trusted couriers (like Radio)
- But ... limited vocabulary

Encrypt

- Symmetric cipher:
plaintext + key → ciphertext
ciphertext + key → plaintext

- Kerckhoffs's Principle:

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

- DES, IBM, NSA and Paranoia for Beginners
- But ... key management

Key Management 101

- Easy: Military
 - Hierarchy, predictable channels
 - Pre-existing relationship
- Difficult: Internet
 - Many-to-many
 - Unpredictable requirements
- RSA hack – made worse because copies of secret keys were kept

Asymmetric Cryptography

- Key pair

What one key of the pair encrypts, only the matching key can decrypt (and visa versa)

- Encryption / Decryption

plaintext + public → ciphertext
ciphertext + private → plaintext

- Authentication

message → digest
digest + private → signature
signature + public → digest?

- But ... Identity, Authenticity?

Certificate

- Identification (FQDN)
- Public Key
- Signed by trusted third party
- But ... who do you trust?

Certification authority

- List of CA's decided by someone else
- Pre-installed in your browser
- Mistakes (Verisign)
- Race to the bottom
- A chain is only as strong as the weakest link
- PROFITABLE

Web of Trust

- Security decisions in the hands of the users
- LCA key-signing parties
- Too much like hard work...

2: Misc

- Cryptographic primitives
- Key Size
- Hybrid Cryptography
- Block ciphers and Stream ciphers
- Crypto Attacks
- What does **HTTPS** actually mean?
- How NOT to design a crypt-system
- Snakeoil
- Legal

Primitives

- Symmetric Block Cipher
DES, AES, Blowfish
- Asymmetric Cipher
RSA, ECC
- Digest
- Random Number Generator

Keys. Size matters....

- “a large key is no guarantee of security, but a small key is a guarantee of insecurity”
- Symmetric cipher
256 currently sufficient (*)
add 1 bit, make it twice as difficult
- Asymmetric
4096 currently sufficient (*)

Hybrid Crypto

- Asymmetric to generate session key
- Symmetric to handle bulk transfer
- Periodically re-key

Block → Stream

- Practically all crypto done on computers by block ciphers
- Practically all requirements are for a stream cipher
- Convert block → stream
- Needs Initialization Vector (MS Office)

Crypto Attacks

- Recovering plaintext is not always the aim
- Cipher
- Key Size
- Entropy
- Known/chosen plaintext
- Side Channel

Side Channel

- TEMPEST
- Efficiency is the enemy of security
 - Power
 - Time
 - Cache
- Environmental
 - Power
 - Time
 - Temperature

HTTPS

- HTTPS secure iff:
 - Good primitives (esp RNG (Debian))
 - Good implementation
 - Right URL
 - *Trustworthy* CA
 - HTML/XSS
 - Data-in-motion vs. data-at-rest

How NOT to ...

- CSS
- WEP
- DRM in general

Snakeoil

- Revolutionary Secret Algorithm
- One-Time-Pad
- Ridiculous key-size
- Competition to break
- Jargon Overload

Legal

IANAL!

- US – export restrictions, Clipper, DMCA
- France – illegal
- UK – RIPA
- AU – RIPA-*lite*, DMCA-by-proxy
- Customs
- Fighting fire with ... staples?

~~3: Examples~~

```
$ gpg < /dev/null
$ echo Hello World | tee plaintext \
  | gpg --symmetric --armour \
    --passphrase bob \
  | tee ciphertext
$ gpg < ciphertext --passphrase bob \
  | tee recovered
$ gpg SHA256SUMS.gpg
$ sha256sum --check SHA256SUMS
```

~~Examples cont.~~

```
$ gpg --gen-key
```

```
$ gpg --export --armour >  
AlicePublicKey.gpg
```

```
$ gpg --import < BobPublicKey.gpg
```

```
$ gpg --sign-key robert
```

```
$ gpg --recv-keys FBB75451
```

```
$ gpg --sign-key FBB75451
```

```
$ gpg SHA256SUMS.gpg
```

~~Examples cont..~~

```
$ echo Eve is listening \
| gpg --encrypt --armour \
  --recipient robert \
| tee message.gpg
```

```
$ gpg < message.gpg
```

```
$ echo Eve is listening \
| gpg --encrypt --armour \
  --recipient robert --sign \
| tee signed-message.gpg
```

```
$ gpg < signed-message.gpg
```

Thank You

thomas.sprinkmeier@gmail.com

<http://en.wikipedia.org/wiki/Portal:Cryptography>

One Geek Per Classroom

<http://ogpc.dns.id.au>

Attribution-NonCommercial-ShareAlike 3.0 Unported
(CC BY-NC-SA 3.0)

<http://creativecommons.org/licenses/by-nc-sa/3.0/>