

# Last Time

- Internet Application Security and Privacy
  - Transport-layer security and privacy: Tor
  - Application-layer security and privacy: SSH, remailers

# This time

[Stinson, Shmatikov-Boneh]

- Internet Application Security and Privacy
  - Transport-layer security and privacy: Tor
  - Application-layer security and privacy: remailers, PGP/gpg

# Anonymity for email: remailers

- Tor allows you to anonymously communicate over the Internet in real time
  - What about (non-interactive) email?
  - This is actually an easier problem, and was implemented much earlier than Tor
- Anonymous remailers allow you to send email without revealing your own email address
  - Of course, it's hard to have a conversation that way
  - Pseudonymity is useful in the context of email

# Simple remailers

- In the 1990s, there were very simple (“type 0”) remailing services, the best known being anon.penet.fi
- How it worked:
  - Send email to anon.penet.fi
  - It is forwarded to your intended recipient
  - Your “From” address is changed to anon43567@anon.penet.fi (but your original address is stored in a table)
  - Replies to the anon address get mapped back to your real address and delivered to you

# anon.penet.fi

- This works, as long as:
  - No one's watching the net connections to or from anon.penet.fi
  - The operator of anon.penet.fi and the machine itself remain trustworthy and uncompromised
  - The mapping of anon addresses to real addresses is kept secret
- Unfortunately, a lawsuit forced Julf (the operator) to turn over parts of the list, and he shut down the whole thing, since he could no longer legally protect it

# Cypherpunk remailers

- Cypherpunk (type I) remailers removed the central point of trust
- Messages are now sent through a “chain” of several remailers, with dozens to choose from
- Each step in the chain is encrypted to avoid observers following the messages through the chain; remailers also delay and reorder messages
- Support for pseudonymity is dropped: no replies!

# Mixmaster remailers

- Mixmaster (type II) remailers appeared in the late 1990s
- Constant-length messages to provide protection against traffic analysis
- Protections against replay attacks
- Improved message reordering
- Requires a special email client to construct the message fragments

# Pretty Good Privacy

- The first popular implementation of **public-key cryptography**.
- **Originally made by Phil Zimmerman in 1991**
  - He got in a lot of trouble for it, since cryptography was highly controlled at the time.
- **Today, there are many (more-or-less) compatible programs**
  - **GNU Privacy Guard (gpg), Hushmail, etc.**



# Pretty Good Privacy

- What does it do?
  - Its primary use is to protect the contents of email messages
- How does it work?
  - Uses public-key cryptography to provide:
    - Encryption of email messages
    - Digital signatures on email messages

# Recall

- In order to use public-key encryption and digital signatures, Alice and Bob must each have:
  - A public **encryption key**
  - **A private decryption key**
  - **A private signature key**
  - **A public verification key**

# Sending a message

- To send a message to Bob, Alice will:
  - Write a message
  - Sign it with her own signature key
  - Encrypt both the message and the signature with Bob's public encryption key
- Bob receives this, and:
  - Decrypts it using his private decryption key to yield the message and the signature
  - Uses Alice's verification key to check the signature

# Back to PGP

- PGP's main functions:
  - Create these four kinds of keys
    - encryption, decryption, signature, verification
  - Encrypt messages using someone else's encryption key
  - Decrypt messages using your own decryption key
  - Sign messages using your own signature key
  - Verify signatures using someone else's verification key
  - **Sign other people's *keys using your own signature key***

# Obtaining keys

- Earlier, we said that Alice needs to get a copy of Bob's public key in order to send him an encrypted message.
- How does she do this?
  - In a secure way?
- Bob could put a copy of his public key on his webpage, but this isn't good enough to be really secure!
  - Why?

# Verifying public keys

- If Alice knows Bob personally, she could:
  - Download the key from Bob's web page
  - Phone up Bob, and verify she's got the right key
  - Problem: keys are big and unwieldy!

```
mQGIBDi5qEURBADitpDzvzvzW+9lj/zYgK78G3D76hvvtIT6gpTIlwg6WIJNLKJat
01yNpMIYNvpwi7EUd/lSNl6t1/A022p7s7bDbE4T5NJda0IOAgWeOZ/plIJC4+o2
tD2RNUskwDQcxzm8KUNZOJla4LvgRkm/oUubxyeY5omus7hcfNrB0wjC1wCg4Jnt
m7s3eNfMu72Cv+6FzBgFog8EANirkNdC1Q8oSMDihWjlogiWbBz4s6HMxzAaqNf/
rCJ9qoK5SLFeoB/r5ksRWty9QKV4VdhhCIy1U2B9tSTlEPYXJHQPZ3mwCxUnJpGD
8UgFM5uKXaEq2pwpArTm367k0tTpMQgXAN2HwiZv//ahQXH4ov30kBBVL5VFxMUL
UJ+yA/4r5HLTpP2SbbqtPWdeW7uDwhe2dTqffAGuf0kuCpHwCTAhr83ivXzT/7OM
```

# Fingerprints

- Luckily, there's a better way!
- A **fingerprint** is a cryptographic hash **of a key**.
- **This, of course, is *much shorter*:**
  - ***B117 2656 DFF9 83C3 042B C699 EB5A 896A 2898 8BF5***
- ***Remember: there's no (known) way to make two different keys that have the same fingerprint.***

# Fingerprints

- So now we can try this:
  - Alice downloads Bob's key from his webpage
  - Alice's software calculates the fingerprint
  - Alice phones up Bob, and asks him to read his key's actual fingerprint to her
  - If they match, Alice knows she's got an authentic copy of Bob's key
- That's great for Alice, but what about Carol, who doesn't know Bob
  - At least not well enough to phone him



# Signing keys

- Once Alice has verified Bob's key, she uses her signature key to sign Bob's key.
- This is effectively the same as Alice signing a message which says "I have verified that the key with fingerprint B117 2656 DFF9 83C3 042B C699 EB5A 896A 2898 8BF5 really belongs to Bob."
- Bob can attach Alice's signature to the key on his webpage.

# Web of Trust

- Now Alice can act as an **introducer for Bob.**
- **If Carol doesn't know Bob, but does know Alice (and has already verified Alice's key, and trusts her to introduce other people):**
  - she downloads Bob's key from his website
  - she sees Alice's signature on it
  - she is able to use Bob's key without having to check with Bob personally
- **This is called the Web of Trust, and the PGP software handles it mostly automatically.**

# So, great!

- So if Alice and Bob want to have a private conversation by email:
  - They each create their sets of keys
  - They exchange public encryption keys and verification keys
  - They send signed and encrypted messages back and forth
- Pretty Good 😊

# Plot Twist

- **Bob's computer is stolen by “bad guys”**
  - **Criminals**
  - **Competitors**
  - **Subpoenaed by the RCMP**
- **Or just broken into**
  - **Virus, trojan, spyware**
- **All of Bob's key material, including his private key is discovered**

# The Bad Guys Can...

- Decrypt past messages
- Learn their content
- Learn that Alice sent them
- And have a **proof** they can show to anyone else!
- How private is that?

# What Really Went Wrong

- PGP creates lots of incriminating records:
  - Key material that decrypts data sent over the public Internet
  - Signatures with proofs of who said what
- Alice privacy depends on Bob's actions!

# Recap

- Internet Application Security and Privacy
  - Transport-layer security and privacy: Tor
  - Application-layer security and privacy: remailers, PGP/gpg

# Next time

- Database Security and Privacy
  - Introduction to Databases
  - Security Requirements
  - Integrity
  - Auditability, Access Control, and Availability