Last time

- Internet Application Security and Privacy
  - Transport-layer security and privacy: TLS / SSL, Tor
    - The Nymity Slider
  - Application-layer security and privacy: ssh

This time

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

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
    - Nymity Slider

Type 0 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

Type I 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!

Type II remailers

- Mixmaster (type II) remailers appeared in the late 1990s
- Constant-length messages to avoid an observer watching “that big file” travel through the network
- Protections against replay attacks
- Improved message reordering
- But! Requires a special email client to construct the message fragments
    - premail (a drop-in wrapper for sendmail) makes it easy

Nym servers

- Recovering pseudonymity: “nym servers” mapped pseudonyms to “reply blocks” that contained a nested encrypted chain of type I remailers. Attaching your message to the end of one of these reply blocks would cause it to be sent through the chain, eventually being delivered to the nym owner
- But remember that there were significant privacy issues with the type I remailer system
- Easier recipient anonymity: alt.anonymous.messages

Type III remailers

- Type II remailers were the state of the art until recently
- Mixminion (type III) remailer
  - Native (and much improved) support for pseudonymity
    - No longer reliant on type I reply blocks
  - Improved protection against replay and key compromise attacks
- But it's not very well deployed or mature
  - “You shouldn't trust Mixminion with your anonymity yet”

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.
  - But that's a whole 'nother story. :-)
- 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!


```
mQGiBDi5qEURBADitpDzvvzW+9lj/zYgK78G3D76hvvvIT6gpTIlwg6WIJNLKJat
01yNpMIYNvpwi7EUd/lSNl6t1/A022p7s7bDbE4T5NJda0IOAgWeOZ/plIJC4+o2
tD2RNuSkwDQcxzm8KUNZOJla4LvgRkm/oUubxyeY5omus7hcfNrBOwjC1wCg4Jnt
m7s3eNfMu72Cv+6FzBgFog8EANirkNdC1Q8oSMDihWj1ogiWbBz4s6HMxzAaqNf/
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, no?

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 is discovered
  - Oh, no!

The Bad Guys Can...

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

- How private is that?

What went wrong?

- Bob's computer got stolen?
- How many of you have never...
  - Left your laptop unattended?
  - Not installed the latest patches?
  - Run software with a remotely exploitable bug?
- What about your friends?

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 had better watch what she says!
  - Her privacy depends on Bob's actions

Casual Conversations

- Alice and Bob talk in a room
- No one else can hear
  - Unless being recorded
- No one else knows what they say
  - Unless Alice or Bob tells them
- No one can **prove** what was said
  - Not even Alice or Bob
- These conversations are “off-the-record”

We Like Off-the-Record Conversations

- Legal support for having them
  - Illegal to record conversations without notification
- We can have them over the phone
  - Illegal to tap phone lines
- But what about over the Internet?

Crypto Tools

- We have the tools to do this
  - We've just been using the wrong ones
  - (when we've been using crypto at all)

- We want **perfect forward secrecy**

- We want **deniable authentication**

Perfect Forward Secrecy

- Future key compromises should not reveal past communication
- Use a short-lived encryption key
- Discard it after use
  - Securely erase it from memory
- Use long-term keys to help distribute and authenticate the short-lived key
- Q: Why do these new long-term keys not have the very same forward secrecy problem?

Recap

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

Next time

- Finish OTR
- Database Security
  - Introduction to Databases
  - Security Requirements
  - Integrity
  - Auditability, Access Control, and Availability
  - Data Inference
  - Statistical Inference
  - Controls against Inference