27 Encryption

Basic idea: store and transmit information in an encoded form that doesn't make any sense.

The basic mechanism:

- Start with text to be protected. Initial readable text is called *clear text.*
- Encrypt the clear text so that it doesn't make any sense at all. The nonsense text is called *cipher text.* The encryption is controlled by a secret password or number; this is called the *encryption key.*
- The encrypted text can be stored in a readable file, or transmitted over unprotected channels.
- To make sense of the cipher text, it must be *decrypted* back into clear text. This is done with some other algorithm that uses another secret password or number, called the *decryption key.*

All of this only works under three conditions:

- The encryption function cannot easily be inverted (cannot get back to clear text unless you know the decryption key; cannot compute key even when given cleartext and ciphertext).
- The encryption and decryption must be done in some safe place so the clear text can't be stolen.
- The keys must be protected. In *secret key* systems, can compute one key from the other (usually the encryption and decryption keys are identical), so can't afford to let either key leak out.

Federal encryption standard (DES). Can be implemented efficiently in hardware and used to be considered safe. Today, no longer considered safe enough (due to increased compute power). Triple DES encryption still relatively safe. There are chips that encrypt/decrypt hundreds of Megabits per second.

Problem: how to establish secure channel (i.e. distribute keys) in the first place?

Public key encryption: new mechanism for encryption where knowing the encryption key doesn't help you to find decryption key, or vice versa.

- User provides a single password, system uses it to generate two keys (use a one-way function, so can't derive password from either key).
- In these systems, keys are inverses of each other: could just as easily encrypt with decryption key and then use encryption key to recover clear text.
- Each user keeps one key secret, publicizes the other. Can't derive private key from public. Public keys are made available to everyone, in a phone book for example.

Safe mail:

- Use public key of destination user to encrypt mail.
- Anybody can encrypt mail for this user and be certain that only the user will be able to decipher it.

It's a nice scheme because the user only has to remember one key, and all senders can use the same key. However, how does receiver know for sure who it's getting mail from?

Positive identification: can also use public keys to certify identity:

- To certify your identity, use your private key to encrypt a text message, e.g. "I agree to pay Mary Wallace $100 per year for the duration of life."
- You can give the encrypted message to anybody, and they can certify that it came from you by seeing if it decrypts with your public key. Anything that decrypts into readable text with your public key *must* have come from you! This can be made legally binding as a form of electronic signature.

These two forms of encryption can be combined together. To identify sender in secure mail, encrypt first with your private key, then with receiver's public key.

Encryption appears to be a great way to thwart listeners. It doesn't help with Trojan Horses, though.

General problem: how do we know that an encryption mechanism is safe? It's extremely hard to prove. This is a hot topic for research: theorists are trying to find provably hard problems, and use them for proving safety of encryption.