CS 321 Lecture Notes

Orion Sky Lawlor
Department of Computer Science, University of Alaska at Fairbanks
http://lawlor.cs.uaf.edu/ olawlor olawlor@acm.org

1 2005/04/27 Lecture: Cryptography for System Security

Crypto is discussed in detail on Page 576 of the Silberschatz textbook.

Cryptographic algorithms all basically take a piece of regular data called the "plaintext" P, and encrypt it using an encryption algorithm and encryption key E. This results in a piece of "cyphertext" C, which should not look at all similar to P. With the decryption key D, the intended recipient can then run a decryption algorithm to recover the plaintext P.

That is, the sender turns their message P into cypher text C using the encryption key E.

$$P =>_E C$$

And the receiver takes the cyphertext C and turns it into plaintext P using the decryption key D.

$$C =>_D P$$

For a "symmetric cypher", $E = D$. The oldest symmetric cypher is Caeser's cypher, where the encryption and decryption algorithms both compute $C = (P + 13) \, MOD \, 26$. So ACK becomes NPX. Caeser's cyper, or any similar "substitution cypher" is painfully insecure– if you can work out one letter (e.g., based on formatting, frequency analysis, or just repetition), you can recognize it anytime that letter appears again. Useful examples of symmetric cyphers are the older DES (56-bit key, 64-bits of data at a time), and the more modern AES (128-bit to 256-bit key, 128 bits at a time). Both of these are "block cyphers", where they take the input one block (64 or 128 bits) at a time, mix it up thoroughly, spit it out, keep the mixed copy, and move to the next block. This means changing one bit of the text changes its block and all subsequent blocks, which makes these cyphers more robust to certain attacks.

The main weakness of symmetric cyphers is making sure both sides have the key, and nobody else does. To solve this key distribution problem, there are a variety of "asymmetric cyphers", where $E! = D$. The most common by far is the RSA Algorithm (Mathematics and Example Code), which boils down to exponentiation and modulo operations on very large numbers. In RSA, one can easily generate (e.g., with "ssh-keygen" on any UNIX machine) a public key E and private key D. Then you can publish E (or hand it out on demand), keep D secret, and then people can encrypt messages that only you can decrypt. This is used for SSL (https), where the encryption key E is in the "certificate" the server uses during communication set up; and by SSH, where the encryption key E is sent by the server. This is also used in reverse when publishing software over the net–here, the encryption key E is kept private, at the software manufacturer, and the decryption key D is published and included in every machine. This way when a machine receives an update, they can

decrypt it using the decryption key, and know that only somebody with the encryption key could have created the message.

Asymmetric cypers are hence quite useful, but they're also quite slow. So most of the time we use the asymmetric cypher to exchange a new (random) shared key, and then use a fast symmetric cypher for the bulk of the communication. This is used by both SSL and SSH.

There's a final sort of algorithm cryptographers concern themselves with, which is a hash function. A hash is just an encryption method where there isn't a corresponding decryption method–the hash has scrambled the data forever. Hashes are used check passwords by hashing the proported password, and comparing its hash against the stored hashcode of the old password. This allows you to check passwords without ever actually storing them. Hashing also allows you to determine if a file has been tampered with, because the hash code will change, but you've got to somehow store the hash codes where they can't be tampered with as well!

Encryption is used in computer security for:

- Checking passwords (via their hash code) without ever storing them.
- Detecting changed files (via changed hash codes).
- Communicating over an insecure channel, such as sending passwords over a wireless network.
- Protecting data from physical compromise, such as encrypting a filesystem.
- Verifying the identity of a possibly forged message (network packet, file, email).

A small encrypted filesystem can be generated in Linux 2.6 via:

```
modprobe cryptoloop
modprobe aes
dd if=/dev/urandom of=my.sto bs=1024K count=1
losetup -e AES128 /dev/loop0 my.sto
mke2fs -m 0 /dev/loop0
mkdir my
mount /dev/loop0 my
```

Once you've build the filesystem, you can repeat these steps without the "dd" and without the "mke2fs" to remount the data. Of course, it's easy to write a script to do this, which should ideally also turn off swap space ("swapoff -a") to prevent writing sensitive data to the swap file, turn off network services to prevent remote access, and other paranoia.