Administrivia

We now have (informal) project proposals from everyone. Expect to present these the last week of the semester, May 1 and May 3. The format is pretty flexible: computer demonstration, talk, etc. You should be prepared for questions.

PGP

PGP, or *Pretty Good Privacy*, was developed by Phil Zimmermann as an alternative crypto system. Zimmermann has continued to develop and expand the system (his latest target is VoIP telephone security), and the PGP Corporation has a fancy website with a a long list of corporate customers (they claim that 95% of Fortune 100 companies use PGP, for example).

PGP uses a blend of encryption technologies; this alone would make it worth studying. More intriguing is the PGP “web of trust” for key authentication. This imposes a social aspect on PGP analysis, in contrast with the centralized authority of a system like Kerberos.

Workings of PGP

The basic outline of PGP encryption has remained stable, although over the years different algorithms have been, and probably will be, used to accomplish the various steps.

**PGP Authentication**

1. Create a message $M$;
2. Generate a hash of the message, $H$;
3. Encrypt $H$ using the sender’s private key, prepended to message;
4. Receiver uses public key to recover hash code;
5. Receiver compares recovered hash code to code derived from message $M$.

**PGP Confidentiality**

1. Sender generates message $M$ and random key $K$;
2. Encrypt $M$ using $K$;
3. Encrypt $K$ using receiver’s public key;
4. Receiver recovers $K$ using own private key;
5. Receiver uses $K$ to recover $M$.

**PGP Confidentiality and Authentication**

1. Create a message $M$ and random key $K$;
2. Generate a hash of the message, $H$;
3. Encrypt $H$ using the sender’s private key, prepended to message;
4. Encrypt $E(H) || M$ using $K$;
5. Encrypt $K$ using receiver’s public key;
6. Receiver recovers $K$ using own private key;
7. Receiver uses $K$ to recover $E(H) || M$;
8. Receiver uses sender’s public key to recover $H$;
9. Receiver compares recovered hash code to code derived from message $M$.

The missing details are the key generation algorithm, the hash algorithm, the public key encryption algorithm, and the private key encryption algorithm.

PGP also uses compression, which has become less important with the advent of greater bandwidth and cheaper storage. Compression occurs after applying the signature, but before encryption. It is a little better to sign an uncompressed message, because then only the message and the signature need be stored. If a compressed message is signed, then one must either recompress to determine the signature, or store both the compressed and uncompressed versions. This scheme also allows some flexibility in the choice of compression algorithm.

Encryption naturally follows compression, because compression removes much of the redundancy from the original message, thus increasing strength against ciphertext attacks. One might say that no encryption algorithm should be used on an uncompressed message.

Random numbers, compression, hashing, public key crypto, private key crypto, PGP has it all.

---

**PGP algorithms**

So, what algorithms does PGP use? The current version is 8; version 9 is in the works.

**Hash.** While PGP originally used MD5, this is now considered insecure, and SHA–1 is used.

**Public Key Encryption.** PGP uses RSA, and has always used RSA.

**Compression.** Good old ZIP. ZIP uses a variation of the Lempel-Ziv compression algorithm. Roughly speaking, ZIP looks for redundancy in the message and replaces a redundant block with a pointer to the first occurrence of the block.

ZIP (or its gnu-variant, gzip), is quite effective:

```
jimwolper$ dir Lecture23.tex
998153 16 -rw-r--r-- 1 jimwolpe jimwolpe 4521 Apr 9 10:57 Lecture23.tex
jimwolper$ gzip Lecture23.tex
jimwolper$ dir Lecture23.tex.gz
998280 8 -rw-r--r-- 1 jimwolpe jimwolpe 1853 Apr 9 10:57 Lecture23.tex.gz
```

**Private Key Encryption.** Zimmermann was quite adamant from the beginning that the 56-bit key in DES was inadequate. Early versions of PGP used IDEA (International Data
Encryption Algorithm), a 128-bit key block cipher invented by James Massey, a pioneer in
coding and information theory (he also developed SAFER (Secure And Fast Encryption
Routine), which Schneier considers tainted by NSA. Both IDEA and SAFER use higher
arithmetic, along the lines of, but less esoteric than, AES. (Version 9 of PGP will support
AES.)

**Random numbers.**

PGP uses genuine random numbers as well as pseudorandom numbers (PRNs). The ran-
dom numbers are used to generate RSA keys, to seed the PRN generator, and as input to
the PRN algorithm. PRNs are used as session keys ($K$ in the outline above) and in cipher
initialization.

The random numbers come from keyboard latency and the ANSI X9.17 standard. The
ANSI X9.17 standard “reverses” the usual order, by using crypto to generate random
numbers. (This has some kind of bizarre information theory self-reference which we will
not go into now.) The basic operation is

$$R = E(E(T) \oplus V) \text{ and } V = E(E(T) \oplus R);$$

where $E$ is encryption, $T$ is a timestamp, and $V$ is an intialization vector (the first value is
secret, the later values are generated as indicated above). PGP keeps a buffer of “random”
bits which is refreshed whenever a keystroke is expected.

Following Stallings, PRn generation uses a 24-octet seed, and produces a 16-octet PRN
along with an 8-octet initialization vector and a new seed. The message hash is used as a
key to encrypt the output. Then new random octets are generated, and the resulting key
is encrypted using another random key. (There is ne decryption here.)