San Skulrattanakulchai
Gustavus Adolphus College

Cryptography

DES (Data Encryption Standard) and **AES (Advanced Encryption Standard)**
DES was developed in the mid 70's, approved by The National Institute of Standards and Technology (NIST), and became the most popular digital encryption algorithm e.g. UNIX password files, e-banking transactions. Today, DES is no longer considered secure due to its vulnerability to brute-force attack. A newer standard successor to DES is AES (Advanced Encryption Standard), providing keys of up to 256 bits. Both DES and AES are symmetric key algorithms, i.e., the same key is used for encryption and decryption. AES is used today in internet banking, wireless communications, and to encrypt data on the hard disks.

Discrete Logarithm Problem
Given a, c, m, find b such that $a^b \bmod m = c$. This is a hard problem, at least as hard as FACTORING. Many cryptographic protocols are based on the hardness of the discrete logarithm problem.

Definition: A *generator mod p* (where p is prime) is an integer g such that

$$\{\, g^i \bmod p : i = 1, 2, ..., p-1\} = \{1, 2, ..., p-1\}.$$

Examples: 2 and 3 are generators mod 5; 3 and 5 are generators mod 7.
Every prime p has some generator. The discrete log problem always has a solution if a is a generator modulo p, and $c \neq 0 \pmod{p}$.

Diffie-Hellman Key Exchange
Purpose: Alice & Bob generate a secret key S known only to them (e.g. S can be used as the symmetric key for AES).
Setup: Public values prime p, and g, a generator modulo p
Protocol:

1. Alice chooses a random integer $a < p$ and sends Bob $g^a \bmod p$.

2. Bob chooses a random integer $b < p$ and sends Alice $g^b \bmod p$.

3. Alice computes $S_a = (g^b \bmod p)^a \bmod p$.

4. Bob computes $S_b = (g^a \bmod p)^b \bmod p$.

Now Alice and Bob knows $S = S_a = S_b$.
An eavesdropper Eve might have intercepted the values $g^a \bmod p$, and $g^b \bmod p$ but the only way Eve can find S is to solve the discrete log problem for $g^a \bmod p$ or $g^b \bmod p$.

Public key systems
In public-key cryptography each person Alice has a public key P_A and a secret key S_A.
Everyone has access to all the public keys, e.g., Bob can look up Alice's public key P_A.
No one knows the secret key except the owner, e.g., Alice is the only one knowing S_A.
Bob sends a message M to Alice by encrypting M into the cyphertext C using P_A and transmitting C to Alice.
Alice decrypts C using S_A to recover message M.
Alice can also sign a message in a secure manner: she writes the message M, encrypts it into C using S_A and send the signed document (M, C).
Anyone can verify that C decrypts to M using P_A and so is convinced that Alice wrote the document.

RSA system is a public key cryptography protocol.
Key generation: Alice's keys are generated from the following: 2 large primes p, q (chosen by Alice, should be between 1,024–4,096 bits each).
Let $n = pq$.
Let $e =$ a number relatively prime to $(p-1)(q-1)$ (chosen by Alice, usually a small prime).
Let $d =$ multiplicative inverse of $e \bmod (p-1)(q-1)$, i.e., $de \equiv 1 \pmod{(p-1)(q-1)}$.
$P_A = (e, n)$ is the public key.
$S_A = (d, n)$ is the secret key.
Protocol for Bob to send message to Alice (assume $M < n$):

1. Bob encrypts a message $M < n$ as the value $C = M^e \bmod n$ and sends to Alice.

2. Alice decrypts the cipher text by raising to the dth power modulo n.

Because of Fermat's Little Theorem & the Chinese Remainder Theorem, $C^d \bmod n = M$.

El Gamal cryptosystem is another public-key cryptography protocol.
x secret key
(p, g, e) public key, where p is prime, g generator modulo p, $e = g^x \bmod p$.
To encrypt $M < p$, follow these steps:

1. Choose random r relatively prime to $p - 1$.
2. Send $(C_1, C_2) = (g^r \bmod p,\ Me^r \bmod p)$.

To decrypt, follow these steps:

1. Find multiplicative inverse i of C_i modulo p.
2. Compute $M = C_2 i^x \bmod p$.

1-way hash functions ("message digests", "fingerprints", etc)
A 1-way hash function h maps an arbitrarily long string M into a fixed length string $h(M)$, e.g. MD5 uses 128 bits, SHA-1 uses 160 bits.
In addition, $h(M)$ is easy to compute and it's hard to find a string with a given hash value, or 2 strings with the same hash value.
Here are some uses of 1-way hash functions:

1. Economizing on the amount of encryption, e.g., Alice can sign a message M as $S_A(h(M))$ rather than $S_A(M)$.
2. Timestamping:
   (a) To timestamp a message M Alice sends $h(M)$ to a trusted authority Trent.
   (b) Trent provides the timestamp t by signming the document $(h(M), t)$.

   Note that Trent doesn't learn M because of the 1-way hash function.
3. Alice & Bob, at remote locations, can agree on a random coin flip by this protocol:
   (a) Alice generates a number x and sends $h(x)$ to Bob.
   (b) Bob guesses if x is odd or even.

(c) Alice declares heads if Bob is right and tails if he's wrong.

(d) Alice sends x to Bob, who verifies $h(x)$ is the original value.

Alice could cheat if she could find x & x' with the same hash value but opposite parity.