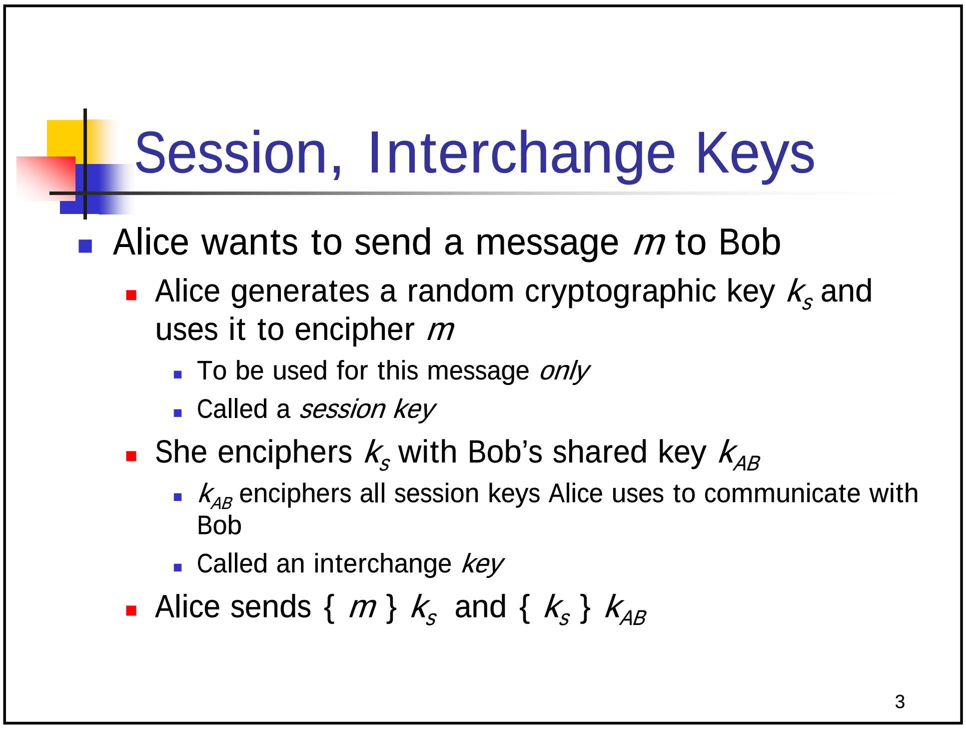
Key Management

Classic (symmetric) Key exchange

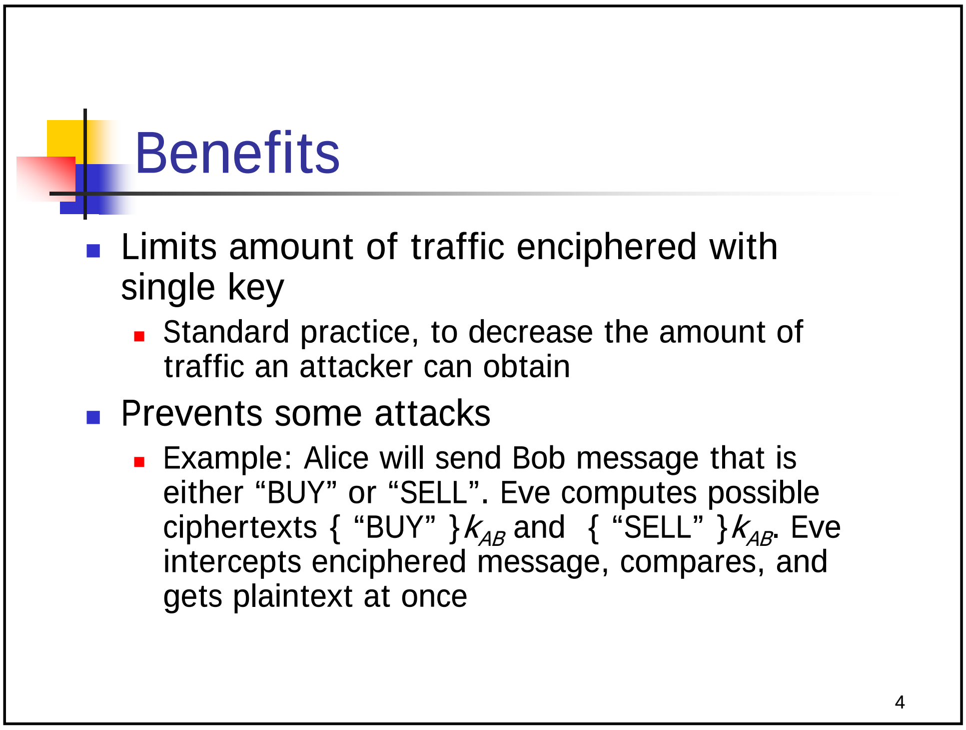
Key Establishment problem

- Securing communication requires that the data is encrypted before being transmitted.
- Associated with encryption and decryption are keys that must be shared by the participants.
- The problem of securing the data then becomes the problem of securing the establishment of keys.
- Task: If the participants do not physically meet, then how do the participants establish a shared key?
- Two types of key establishment:
  - Key Agreement
  - Key Distribution



Session, Interchange Keys

- Alice wants to send a message m to Bob
  - Alice generates a random cryptographic key k_s and uses it to encipher m
    - To be used for this message *only*
    - Called a *session key*
  - She enciphers k_s with Bob's shared key k_{AB}
    - k_{AB} enciphers all session keys Alice uses to communicate with Bob
    - Called an interchange *key*
  - Alice sends $\{ m \} k_s$ and $\{ k_s \} k_{AB}$



Benefits

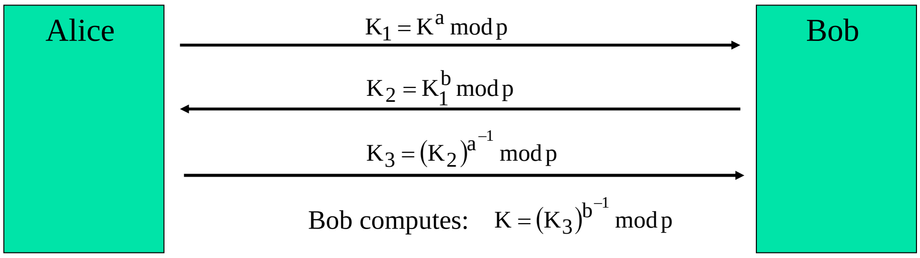
- Limits amount of traffic enciphered with single key
  - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
  - Example: Alice will send Bob message that is either "BUY" or "SELL". Eve computes possible ciphertexts { "BUY" }k_{AB} and { "SELL" }k_{AB}. Eve intercepts enciphered message, compares, and gets plaintext at once

Key Exchange Algorithms

- Goal: Alice, Bob get shared key
  - Key cannot be sent in clear
    - Attacker can listen in
    - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
  - All cryptosystems, protocols publicly known
    - Only secret data is the keys, or information known only to Alice and Bob needed to derive keys
    - Anything transmitted is assumed known to attacker

Key Distribution

- Key Agreement protocols: the key is not determined until after the protocol is performed.
- Key Distribution protocols: one party generates the key and distributes it to Bob and/or Alice (Shamir's 3pass, Kerberos).
- Shamir's Three-Pass Protocol:
  - Alice generates $a \in Z_p^*$ and Bob generates $b \in Z_p^*$.
  - a key K is distributed/generated by:



$K_1 = K^a \bmod p$

$K_2 = K_1^b \bmod p$

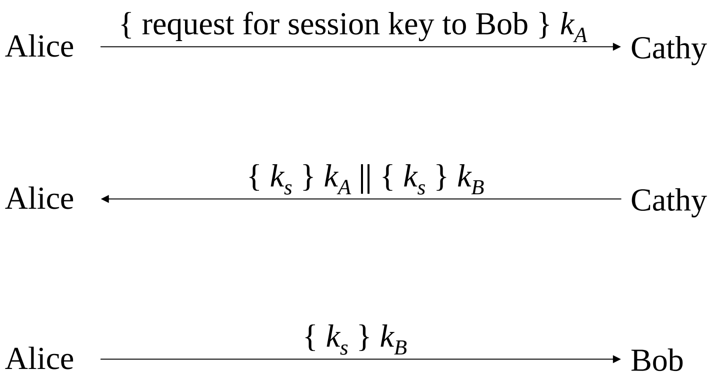
$K_3 = (K_2)^{a^{-1}} \bmod p$

Bob computes: $K = (K_3)^{b^{-1}} \bmod p$

Classical Key Exchange

- Bootstrap problem: how do Alice and Bob begin since Alice cannot send it to Bob in the clear
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key k_A
  - Bob and Cathy share secret key k_B
- Use this to exchange shared key k_s

Simple Protocol



Alice $\xrightarrow{\{\text{ request for session key to Bob }\}\, k_A}$ Cathy

Alice $\xleftarrow{\{\, k_s \,\}\, k_A \,\|\, \{\, k_s \,\}\, k_B}$ Cathy

Alice $\xrightarrow{\{\, k_s \,\}\, k_B}$ Bob

Problems

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- Protocols must provide authentication and defense against replay

Needham-Schroeder

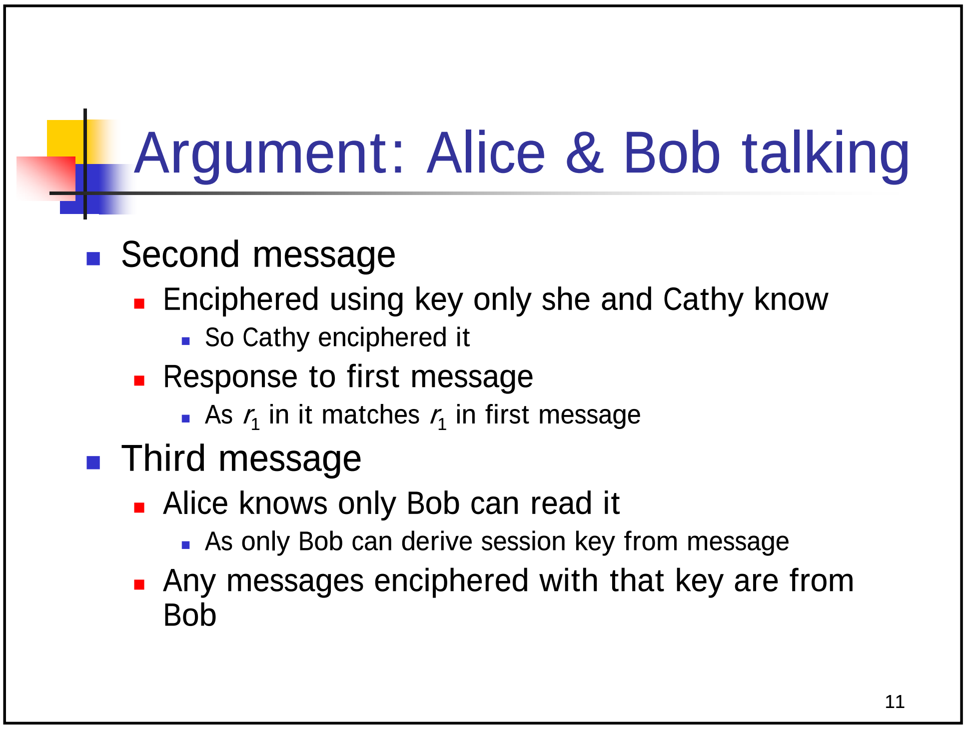
$$\text{Alice} \xrightarrow{\text{Alice} \mid\mid \text{Bob} \mid\mid r_1} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{\ \text{Alice} \mid\mid \text{Bob} \mid\mid r_1 \mid\mid k_s \mid\mid \{\ \text{Alice} \mid\mid k_s\ \}\ k_B\ \}\ k_A} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{\ \text{Alice} \mid\mid k_s\ \}\ k_B} \text{Bob}$$

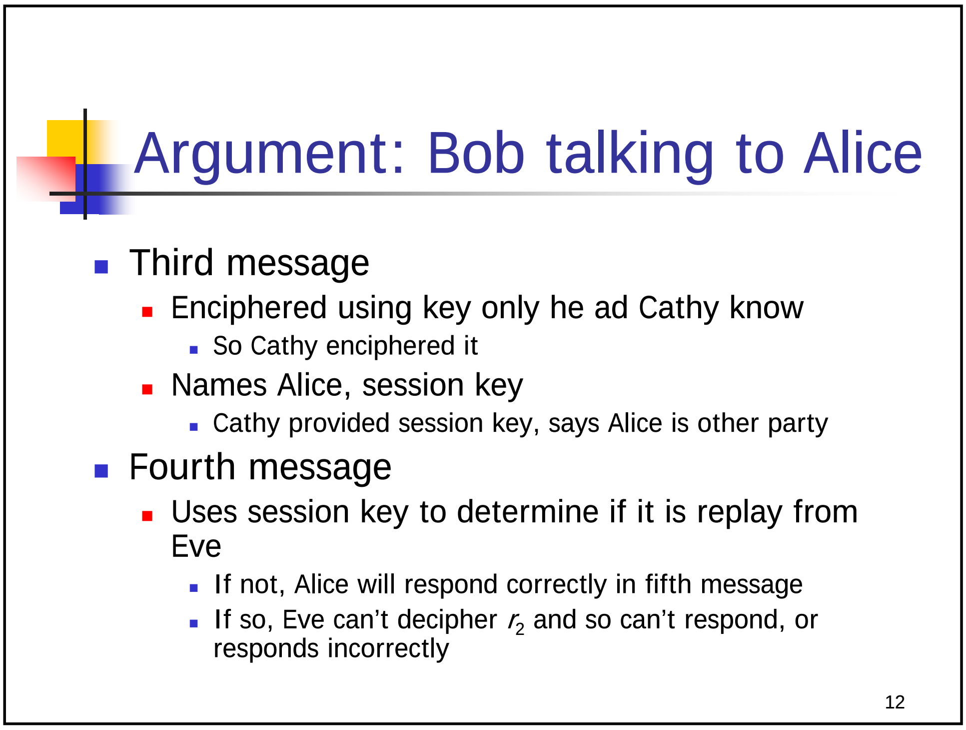
$$\text{Alice} \xleftarrow{\{\ r_2\ \}\ k_s} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{\ r_2 - 1\ \}\ k_s} \text{Bob}$$



Argument: Alice & Bob talking

- Second message
  - Enciphered using key only she and Cathy know
    - So Cathy enciphered it
  - Response to first message
    - As r_1 in it matches r_1 in first message
- Third message
  - Alice knows only Bob can read it
    - As only Bob can derive session key from message
  - Any messages enciphered with that key are from Bob

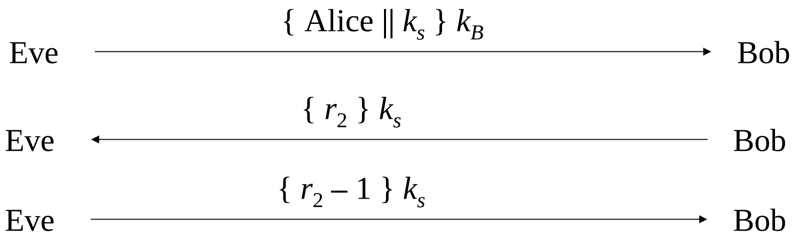


Argument: Bob talking to Alice

- Third message
  - Enciphered using key only he ad Cathy know
    - So Cathy enciphered it
  - Names Alice, session key
    - Cathy provided session key, says Alice is other party
- Fourth message
  - Uses session key to determine if it is replay from Eve
    - If not, Alice will respond correctly in fifth message
    - If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly

Denning-Sacco Modification

- **Assumption**: all keys are secret
- **Question**: suppose Eve can obtain session key. How does that affect protocol?
  - In what follows, Eve knows k_s



Eve $\xrightarrow{\{\text{ Alice } \| \; k_s \} \, k_B}$ Bob

Eve $\xleftarrow{\{ r_2 \} \, k_s}$ Bob

Eve $\xrightarrow{\{ r_2 - 1 \} \, k_s}$ Bob

Solution

- In protocol above, Eve impersonates Alice
- Problem: replay in third step
  - First in previous slide
- Solution: use time stamp T to detect replay
- Weakness: if clocks not synchronized, may either reject valid messages or accept replays
  - Parties with either slow or fast clocks vulnerable to replay
  - Resetting clock does *not* eliminate vulnerability

Needham-Schroeder with Denning-Sacco Modification

$$\text{Alice} \xrightarrow{\text{Alice} \,\|\, \text{Bob} \,\|\, r_1} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{\, \text{Alice} \,\|\, \text{Bob} \,\|\, r_1 \,\|\, k_s \,\|\, \{\, \text{Alice} \,\|\, T \,\|\, k_s \,\}\, k_B \,\}\, k_A} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{\, \text{Alice} \,\|\, T \,\|\, k_s \,\}\, k_B} \text{Bob}$$

$$\text{Alice} \xleftarrow{\{\, r_2 \,\}\, k_s} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{\, r_2 - 1 \,\}\, k_s} \text{Bob}$$

Key Distribution Solutions

Centralized Approach: the trusted third party acts as a **Certificate Authority**:

- Has a known (by the parties) symmetric key, so that clients can be sure that they are talking to the CA

(there is a public key version of the Kerberos protocol using certificates)

Storing Keys

- Multi-user or networked systems: attackers may defeat access control mechanisms
  - Encipher file containing key
    - Attacker can monitor keystrokes to decipher files
    - Key will be resident in memory that attacker may be able to read
  - Use physical devices like “smart card”
    - Key never enters system
    - Card can be stolen, so have 2 devices combine bits to make single key

Key secrecy

- There are problems with truly secret keys:
  - What if someone loses or forgets a key?
  - What if the holder of the key resigns or is killed?
  - What if the user is a criminal?
- On the other hand simply divulging the key to anybody (even - or perhaps especially! - the government) is very insecure
- Encryption Dilemma
  - Public’s need for secure communication
  - Government’s need for lawful access to information

Key Escrow

- A proposed solution is *Key Escrow:*
  - The private key is broken into pieces, which can be verified to be correct
  - Each piece is given to some authority
  - The whole key can only be reconstructed if all the authorities agree
- This is the basis of the US Clipper Chip
  - http://www.cosc.georgetown.edu/~denning
  - http://www.cpsr.org/program/clipper/clipper.html

Secret Splitting

- Usage: you want to keep “components” of some secret in several locations, so that the compromise of one location will not compromise the entire secret.
- This is a very “strong” method, in that it is not vulnerable to guessing IFF it is used correctly
- This method is vulnerable to destruction of one of the “secret” locations

Simple Technique!

- Frank has a message M to protect
- Frank generates a random message R with as many bits in it as message M
- Frank uses XOR of M and R to generate $P = M \oplus R$
- Frank gives P to Alice and R to Bob and destroys M
- If Frank wishes to reconstruct the original message:
    - Frank gets P from Alice and R from Bob
    - Frank XORs them together; the result is $M = P \oplus R$
- As long as Frank does not reuse R, brute force guessing will not tell an enemy whether he/she has the right M if only one of P or R is compromised

Quick Illustration

- Suppose that the message is 1101
- Frank chooses random number R=0101

$$P = M \oplus R = (1101) \oplus (0101)$$

Bitwise: $(1 \oplus 0,\ 1 \oplus 1,\ 0 \oplus 0,\ 1 \oplus 1)$

$$P = 1000$$

- Reversing:

$$P \oplus R = (1000) \oplus (0101)$$

Bitwise: $(1 \oplus 0,\ 0 \oplus 1,\ 0 \oplus 0,\ 0 \oplus 1)$

$$M = 1101$$

We can extend this:

- For splitting the secret M amongst n individuals rather than two, Frank must generate a sequence of random strings:

  $R_1 \ldots R_{n-1}$
- Computing P is then:
  - $P = M \oplus R_1 \oplus \ldots \oplus R_{n-1}$
- Reconstruction involves XOR-ing **all** of the R_i plus P.

Secret Sharing

- Note that secret splitting was vulnerable to the loss of just one site
- To balance a desire to preserve a message with the need to keep the message secret, a “secret sharing” technique is more appropriate
  - Among several, discuss one involving Polynomials!
  - This example is scaled down for ease of typing
- Threshold Scheme
  - (m,n) Threshold Scheme: A secret is divided into n pieces (called the shadows), such that combining any m of the shadows will reconstruct the original secret.
  - Our (scaled down!) example uses Shamir’s LaGrange Interpolating Polynomial Scheme

Shamir’s (m,n) Scheme

- Choose a (public) large prime p bigger than
  - the possible number of shadows
  - the size of the secret
  - other requirements for strength
  - all arithmetic will be “mod p”
- Generate an arbitrary polynomial of degree m-1
- Evaluate the polynomial at n different points to obtain the shadows ki
- Distribute the shadows and destroy M and all the polynomial coefficients

---

Example poly: (3,n) threshold

- Choose an arbitrary polynomial
- m=3 so polynomial is degree 2
  - $F(x) = ax^2 + bx + M \quad (\text{mod } P)$
- We must decide on a size for n - this is the number of shadows. The number of shadows is independent of the degree of the polynomial

(3,5) threshold with M=11

- Suppose we want a (3,5) scheme - that means we will have 5 shadows - for hiding the message “11” (eleven)
- We choose a prime number (for example 13) larger than 5 and 11
- Our polynomial must be of degree m-1=2. Select the coefficients a, b at random:

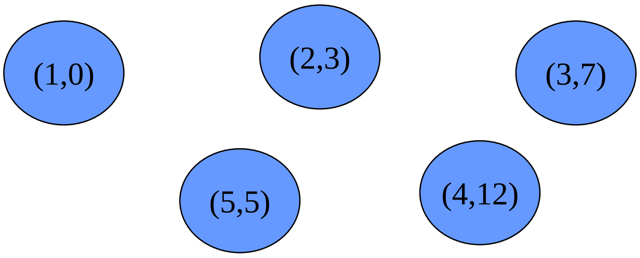
  $F(x) = 7x^2 + 8x + 11 \pmod{13}$

Continuing …

- Now generate five shadows, evaluating the polynomial at points 1,2,3,4,5 (for example)
  - $F(x) = 7x^2 + 8x + 11 \pmod{13}$
  - $k1 = F(x1=1) = 7+8+11 = 0$
  - $k2 = F(x2=2) = \ldots = 3$
  - $k3 = F(x3=3) = \ldots = 7$
  - $k4 = F(x4=4) = \ldots = 12$
  - $k5 = F(x5=5) = \ldots = 5$

Distribution!

- then put the shadows (the ki) somewhere, keeping the selected value for x k_{AB} with the shadow or with the “coordinator”. Discard M, a, b.



(1,0)

(2,3)

(3,7)

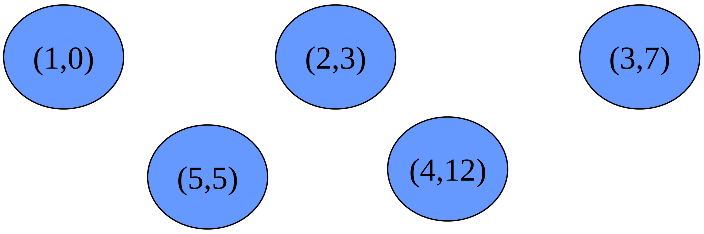
(5,5)

(4,12)

How to get the message back

- We know that this is a (3,5) scheme, so the polynomial is known to be of degree 2:

$$F(x) = Ax^2 + Bx + M$$



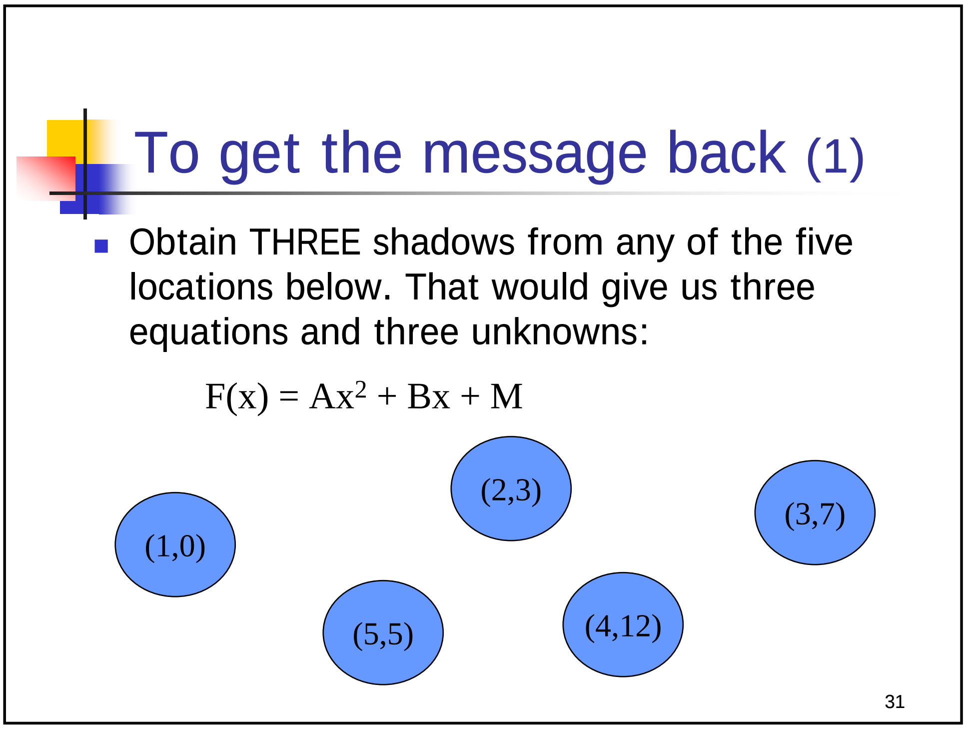
(1,0)

(2,3)

(3,7)

(5,5)

(4,12)



To get the message back (1)

- Obtain THREE shadows from any of the five locations below. That would give us three equations and three unknowns:

$$F(x) = Ax^2 + Bx + M$$

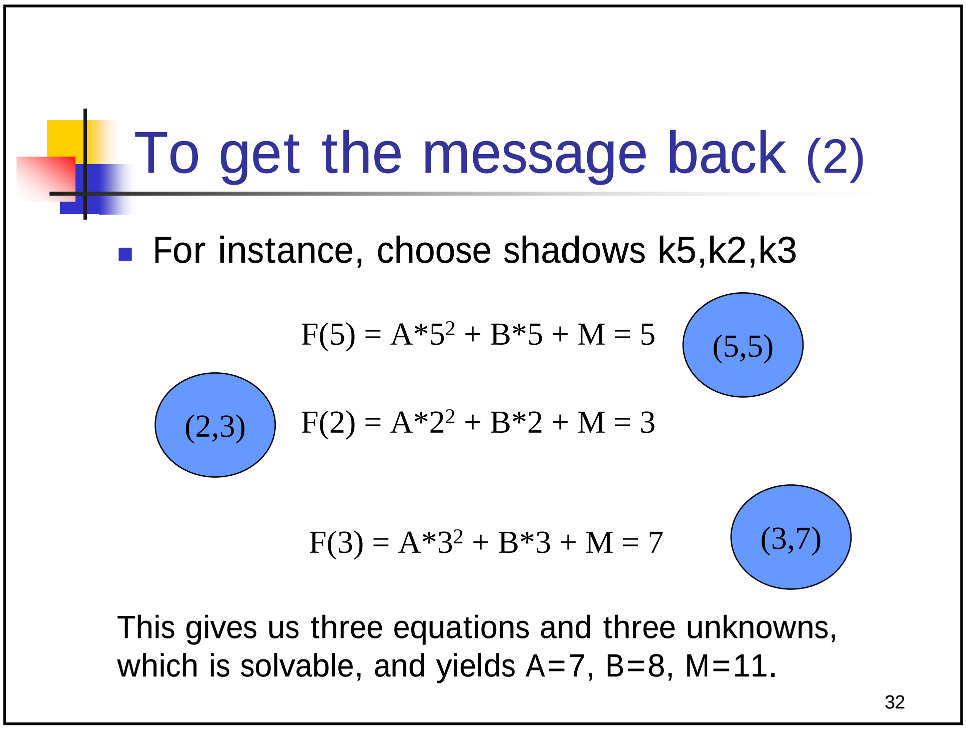
(2,3)

(3,7)

(1,0)

(5,5)

(4,12)



To get the message back (2)

- For instance, choose shadows k5,k2,k3

$$F(5) = A*5^2 + B*5 + M = 5$$

(5,5)

(2,3)

$$F(2) = A*2^2 + B*2 + M = 3$$

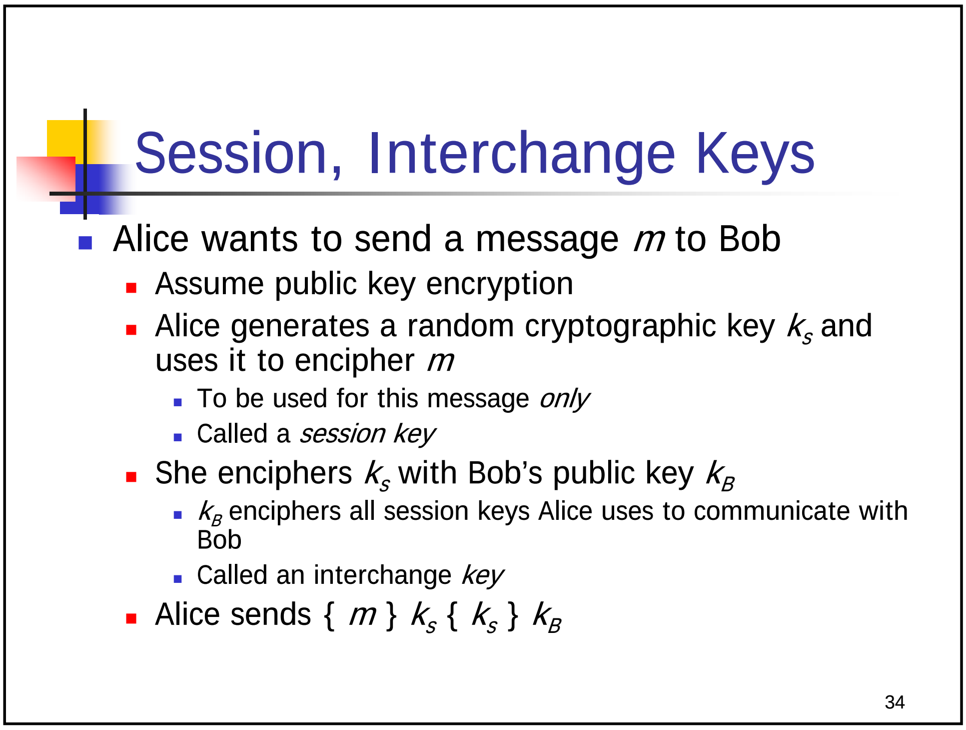
$$F(3) = A*3^2 + B*3 + M = 7$$

(3,7)

This gives us three equations and three unknowns, which is solvable, and yields A=7, B=8, M=11.

Key Management

Public Key exchange



Session, Interchange Keys

- Alice wants to send a message *m* to Bob
  - Assume public key encryption
  - Alice generates a random cryptographic key k_s and uses it to encipher *m*
    - To be used for this message *only*
    - Called a *session key*
  - She enciphers k_s with Bob's public key k_B
    - k_B enciphers all session keys Alice uses to communicate with Bob
    - Called an interchange *key*
  - Alice sends $\{ m \} k_s \{ k_s \} k_B$

Benefits

- Limits amount of traffic enciphered with single key
  - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
  - Example: Alice will send Bob message that is either “BUY” or “SELL”. Eve computes possible ciphertexts { “BUY” } k_B and { “SELL” } k_B. Eve intercepts enciphered message, compares, and gets plaintext at once

Key Generation

- Goal: generate keys that are difficult to guess
- Problem statement: given a set of K potential keys, choose one randomly
  - Equivalent to selecting a random number between 0 and K–1 inclusive
- Why is this hard: generating random numbers
  - Actually, numbers are usually *pseudo-random*, that is, generated by an algorithm

Best Pseudorandom Numbers

- *Strong mixing function*: function of 2 or more inputs with each bit of output depending on some nonlinear function of all input bits
  - Examples: DES, MD5, SHA-1
  - In UNIX-based multiuser systems, the list of all information about all processes on system

Public-Key Key Exchange

- Here interchange keys known
  - e_A, e_B Alice and Bob's public keys known to all
  - d_A, d_B Alice and Bob's private keys known only to owner
- Simple protocol
  - k_s is desired session key

$$\text{Alice} \xrightarrow{\{ k_s \}\, e_B} \text{Bob}$$

Problem and Solution

- Vulnerable to forgery or replay
  - Because e_B known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
  - k_s is desired session key

$$\text{Alice} \xrightarrow{\{\,\{\,k_s\,\}\,d_A\,\}\,e_B} \text{Bob}$$

Notes

- Can include message enciphered with k_s
- Assumes Bob has Alice's public key, and *vice versa*
  - If not, each must get it from public server
  - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle* attack
    - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

Crypto Key Infrastructure

- Goal: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Key Distribution Solutions

Certificate Authority:

Centralized Approach

- Has a very well publicized public key, so that clients can be sure that they're talking to the CA
- This is the public key version of the Kerberos protocol

Digital Certificates

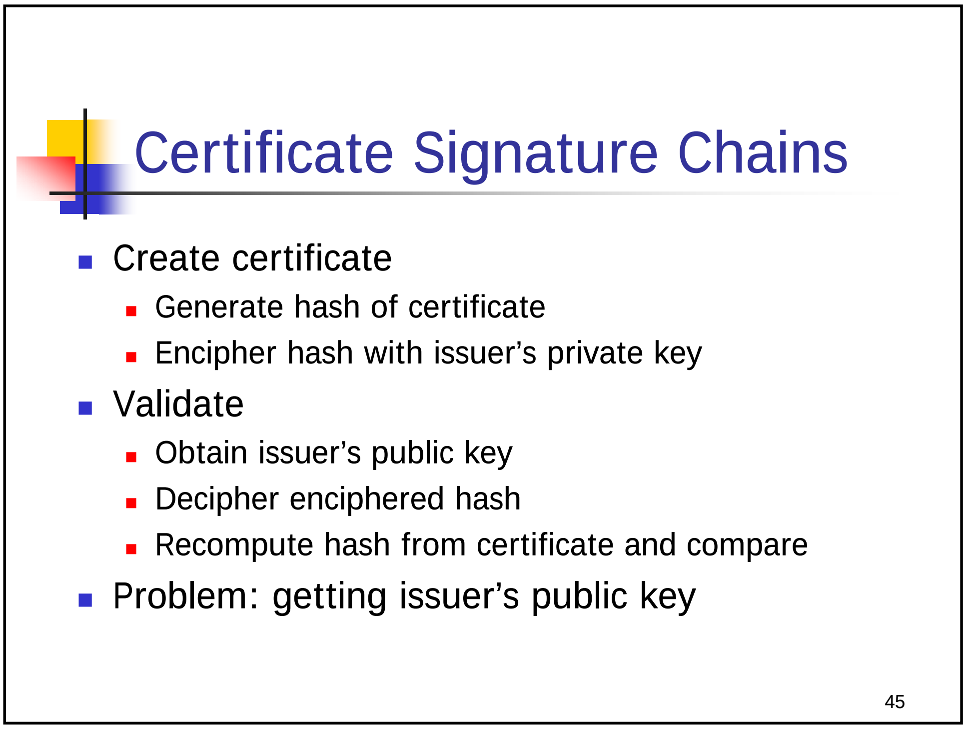
- A digital certificate is an assertion
  - Digitally signed by a “certificate authority”, “famous” and with known public key
- An assertion
  - Typically an identity assertion, sometimes a list of authorizations
- Create token (message) containing
  - Identity of principal (here, Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)

signed by trusted authority (here, Cathy)

$$C_A = \{ e_A \,||\, \text{Alice} \,||\, T \}\, d_C$$

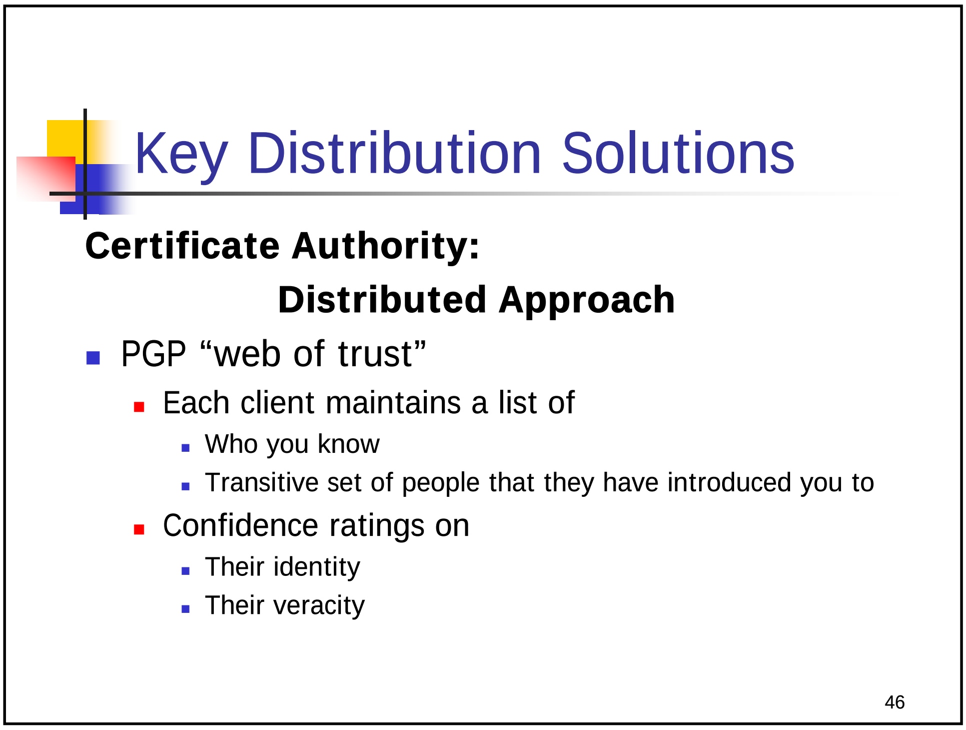
Use

- Bob gets Alice’s certificate
  - If he knows Cathy’s public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice’s public key
- Problem: Bob needs Cathy’s public key to validate certificate
  - Problem pushed “up” a level
  - Two approaches: Merkle’s tree, signature chains



Certificate Signature Chains

- Create certificate
  - Generate hash of certificate
  - Encipher hash with issuer’s private key
- Validate
  - Obtain issuer’s public key
  - Decipher enciphered hash
  - Recompute hash from certificate and compare
- Problem: getting issuer’s public key



Key Distribution Solutions

Certificate Authority:

Distributed Approach

- PGP “web of trust”
  - Each client maintains a list of
    - Who you know
    - Transitive set of people that they have introduced you to
  - Confidence ratings on
    - Their identity
    - Their veracity

Key Revocation

- Certificates invalidated *before* expiration
  - Usually due to compromised key
  - May be due to change in circumstance (*e.g.,* someone leaving company)
- Problems
  - Entity revoking certificate authorized to do so
  - Revocation information circulates to everyone fast enough
    - Network delays, infrastructure problems may delay information

Key secrecy

- There are problems with truly secret keys:
  - What if someone loses or forgets a key?
  - What if the holder of the key resigns or is killed?
  - What if the user is a criminal?
- On the other hand simply divulging the key to anybody (even - or perhaps especially! - the government) is very insecure
- Encryption Dilemma
  - Public's need for secure communication
  - Government's need for lawful access to information