UnitelmaSapienza
Università degli Studi di Roma

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Key Management

Classic (symmetric) and Public (asymmetric) Key exchange

Francesco Parisi Presicce

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** 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).

Key Establishment Algorithm

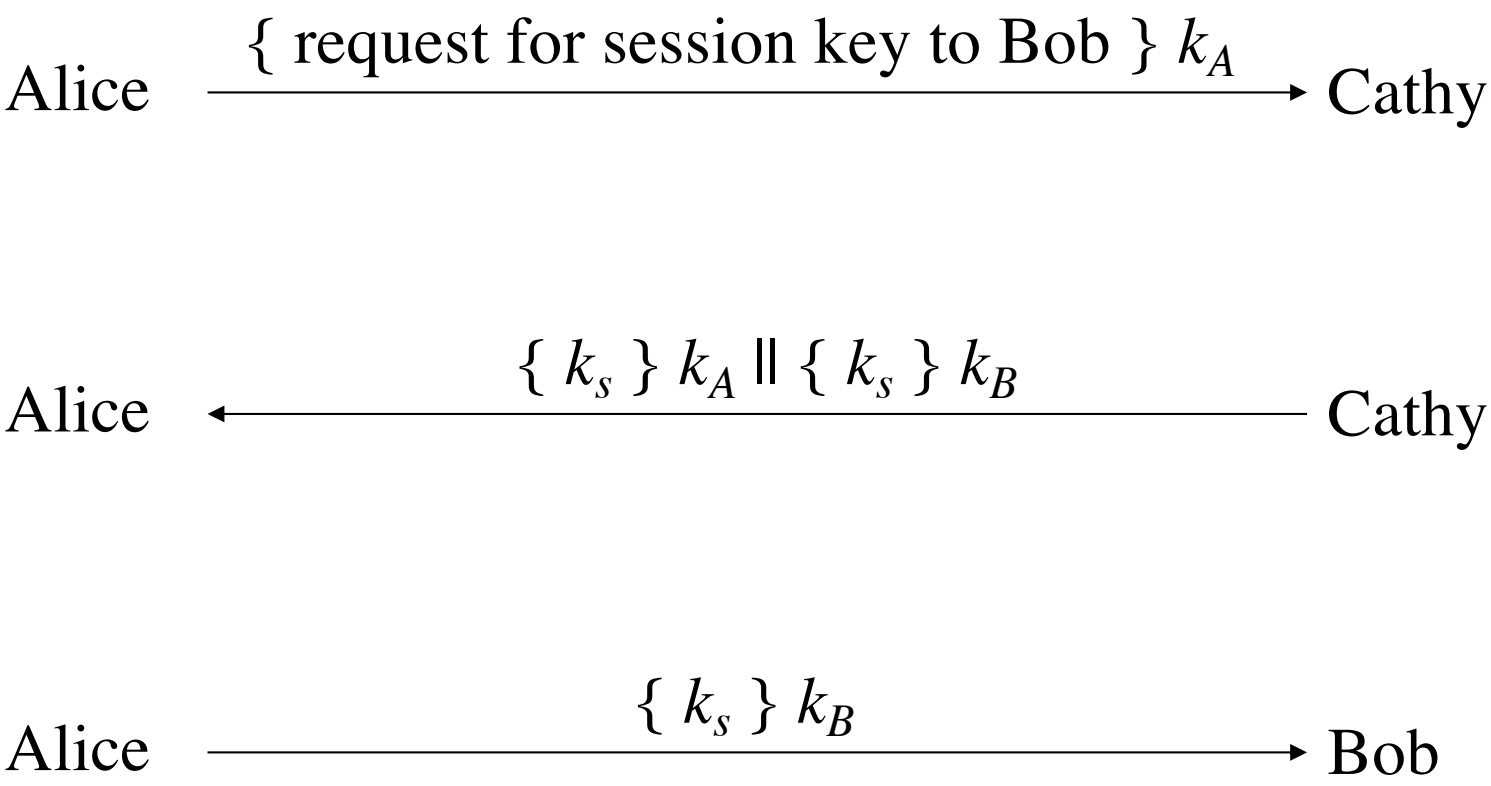
Goal: Alice, Bob get shared key

- 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

Classical Key Exchange

- Bootstrap problem: how do Alice and Bob begin since Alice cannot send it to Bob in the clear (Alice may not know Bob directly)
- Assume a trusted third party, Cathy
  - Alice and Cathy share a secret key k_A
  - Bob and Cathy share a secret key k_B
- Use this to exchange a shared key (generated by Cathy) k_s

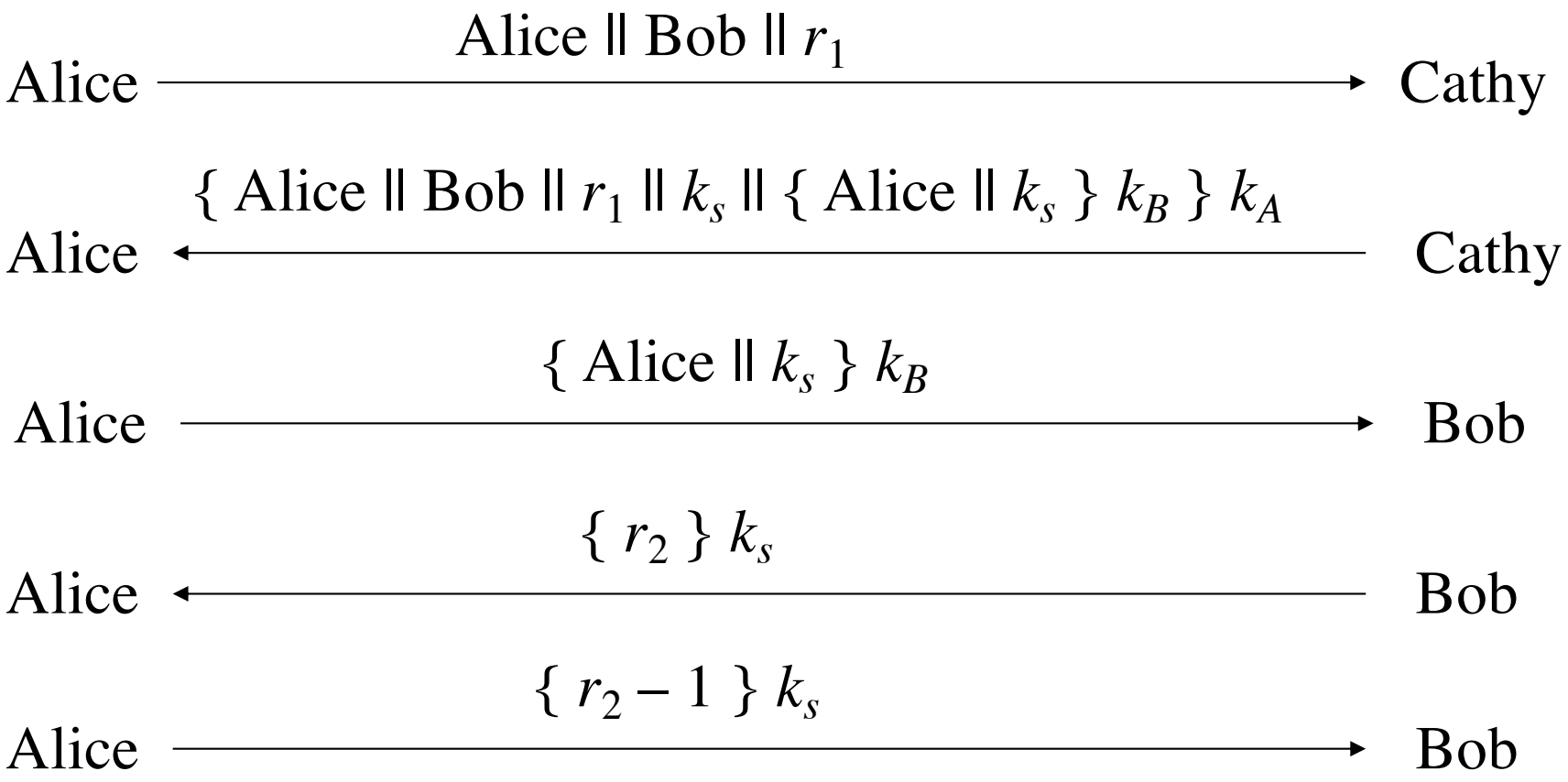
Simple Protocol



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



description

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

Fourth 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

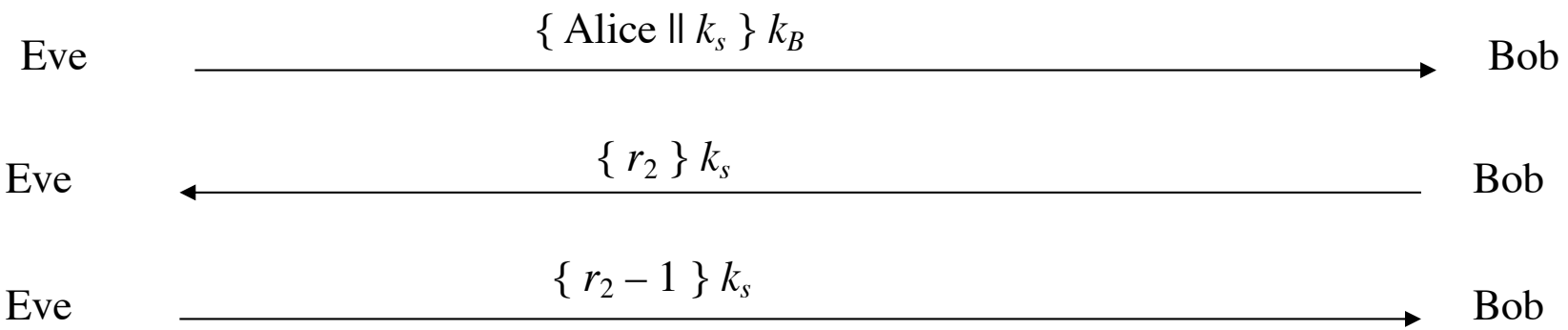
Fifth message
- Uses session key to determine if it is replay from Eve
  - If not, Alice will respond correctly in fifth message
  - If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly

Problem !!

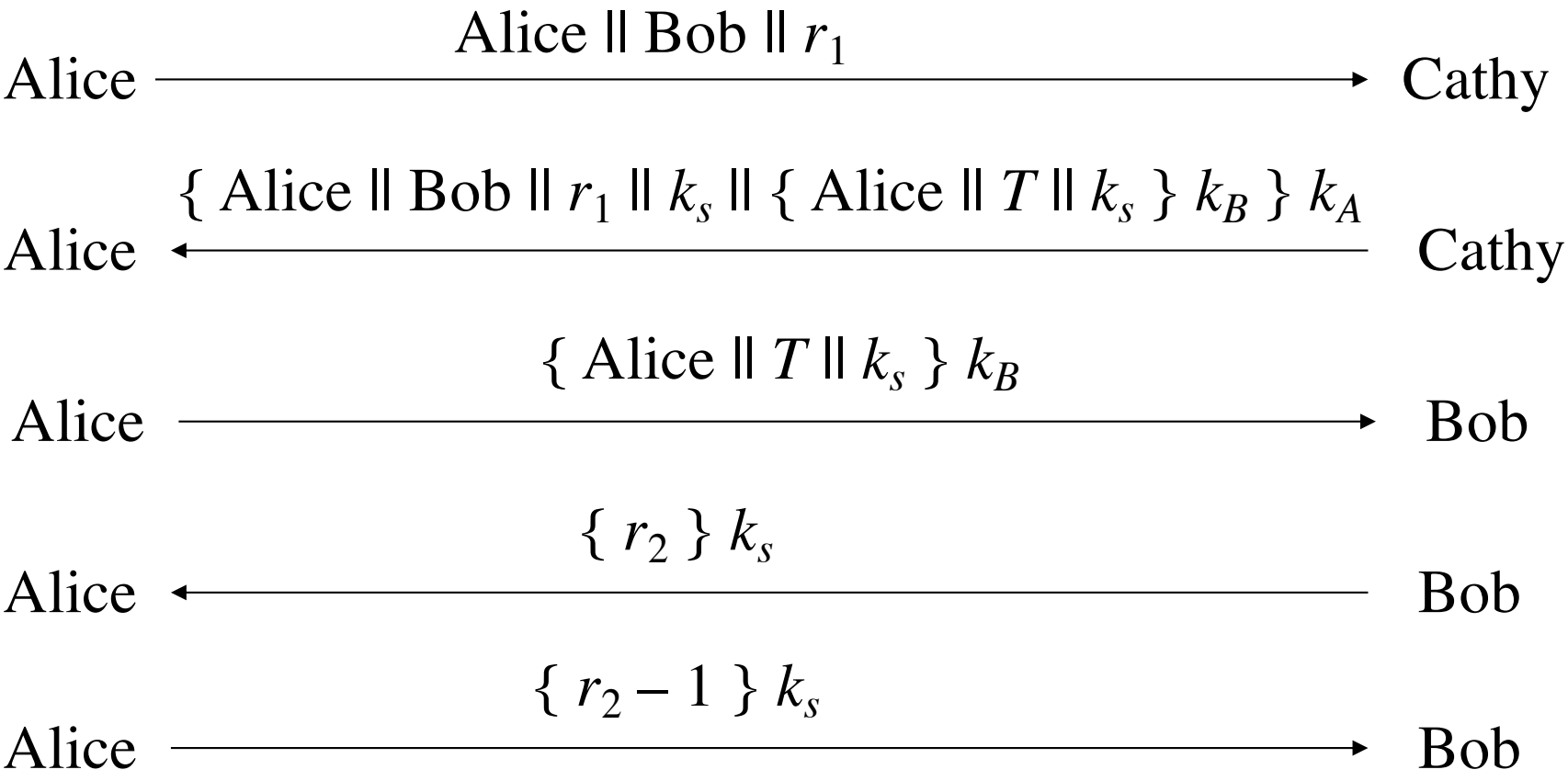
Question: suppose Eve can obtain (with time) the session key. How does that affect protocol?

Eve can impersonate Alice

- Problem: replay in third step
- 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



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 this protocol using certificates)

Key Points

In Classical (symmetric) cryptosystems, enciphering and deciphering use different algorithms but the same key

- Or one key is easily derived from the other
- Difficult distribution of (shared) key

In Public key (asymmetric) cryptosystems enciphering and deciphering use different keys

- Computationally infeasible to derive one (private) from the other (public)
- “Easier” distribution of (public) key

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*
  - k_s is called a *session key* to be used for this message *only*
- She enciphers k_s with Bob's public key k_B
  - k_B (called an interchange *key*) enciphers all session keys Alice uses to communicate with Bob
- 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*, generated by an algorithm

Strong mixing function: function of 2 or more inputs with each bit of output depending on some nonlinear function of all input bits

- (old) Examples: DES, MD5, SHA-1
- In UNIX-based multiuser systems, the list of all information about all processes on system

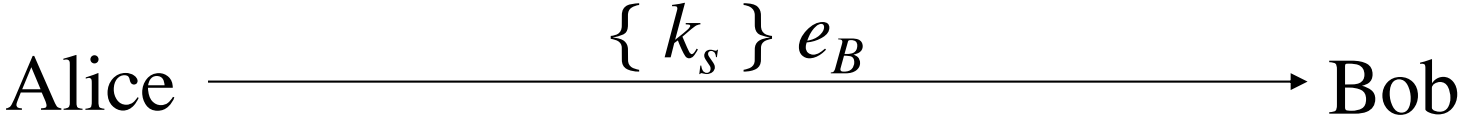
Public-key Key Exchange

Here interchange keys known

- e_A , e_B Alice's and Bob's public key known to all
- d_A, d_B Alice's and Bob's private key known only to the owner

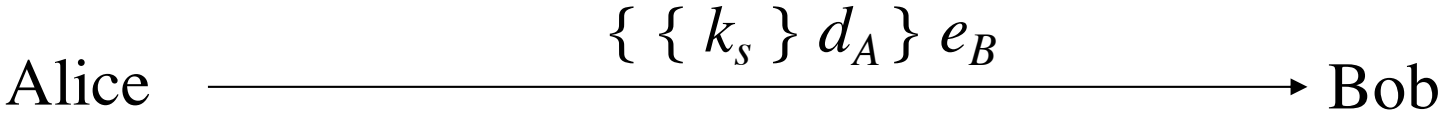
Simple protocol

- k_s is the desired session key



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



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 within public key infrastructure (PKI)

Public-Key Infrastructure

Goal: bind identity to public key

- Crucial as people 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 token 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 verify the validity of the certificate
    - When was the 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

Storing private 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 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

Key Escrow

A proposed solution is *Key Escrow*:

- The 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

More general: you want to keep “components” of some secret in several locations, so that the compromise of one location will not compromise the entire secret: **secret splitting**

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

$$M = 1101$$

Easily extended

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. So, this approach is vulnerable to the losso of just ONE site

Secret Sharing

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 of degree m-1
m=3 so polynomial is degree 2

- $F(x) = ax^2 + bx + M \pmod P$

Decide on a size for n, the number of shadows, which is independent of the degree of the polynomial

- In a (3,5) scheme, we need 5 shadows to hide the message “11” (eleven)
- Choose a prime number (for example 13) larger than 5 and 11.
- The polynomial must be of degree m-1=2. Select the coefficients a, b at random:

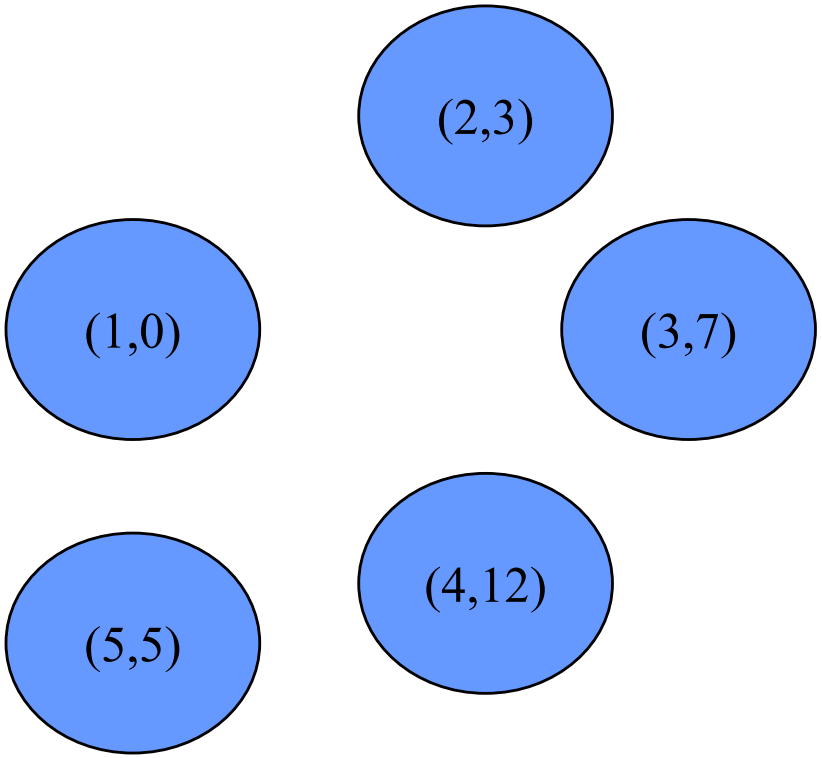
$$F(x) = 7x^2 + 8x + 11 \pmod{13}$$

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) = … = 3
- k3 = F(x3=3) = … = 7
- k4 = F(x4=4) = … = 12
- k5 = F(x5=5) = … = 5

Discard a, b and M

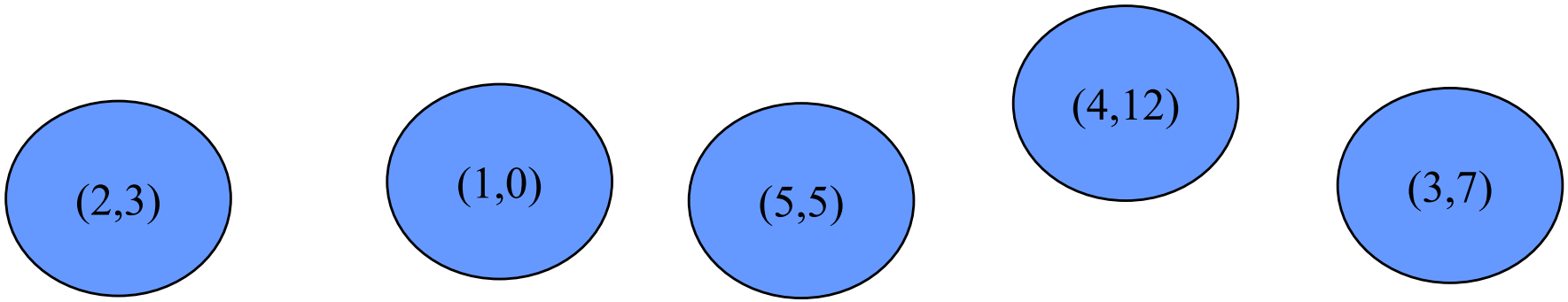


To get the message back

Knowing that it is a (3,5) scheme, the polynomial is known to be of degree 2:

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

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



To get the message back cont.

- 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.

Observations (theorems)

- Given m points (xi, yi) with all xi distinct, there exists a unique polynomial F of degree <m such that yi=F(xi)
  - Hence the secret can be reconstructed with any subset of m of the n shadows
- Any subset of less than (m-1) shadows does not leak any information about the secret
- The reconstruction (Lagrange's Interpolation) requires $O(m*\log^2 m)$ steps
- Shares can be added (increase n) or destroyed without affecting existing shares
- Shares can be replaced without affecting the secret
- Some parties can be given more than one share