CIS 551 / TCOM 401

Computer and Network Security

Spring 2008
Lecture 18

Announcements

- Project 3 available on the web.
  - Get the handout in class today.
  - Project 3 is due April 4th
  - It is easier than project 1 or 2, but *don't wait to start*

- Midterm 2 is *next Tuesday.*
  - Tuesday: April 1st.
  - Will cover all material since the last midterm.

General Principles

- Don’t do anything more than necessary until confidence is built.
  - Initiator should prove identity before the responder does any “expensive” action (like encryption)
- Embed the intended recipient of the message in the message itself
- Principal that generates a nonce is the one that verifies it
- Before encrypting an untrusted message, add “salt” (i.e. a nonce) to prevent chosen plaintext attacks
- Use asymmetric message formats (either in “shape” or by using asymmetric keys) to make it harder for roles to be switched
- Use keys only for one purpose (e.g. authentication but not digital signatures)

Physical Signatures

- Consider a paper check used to transfer money from one person to another
- Signature confirms authenticity
  - Only legitimate signer can produce signature
- In case of alleged forgery
  - 3rd party can verify authenticity
- Checks are cancelled
  - So they can't be reused
- Checks are not alterable
  - Or alterations are easily detected

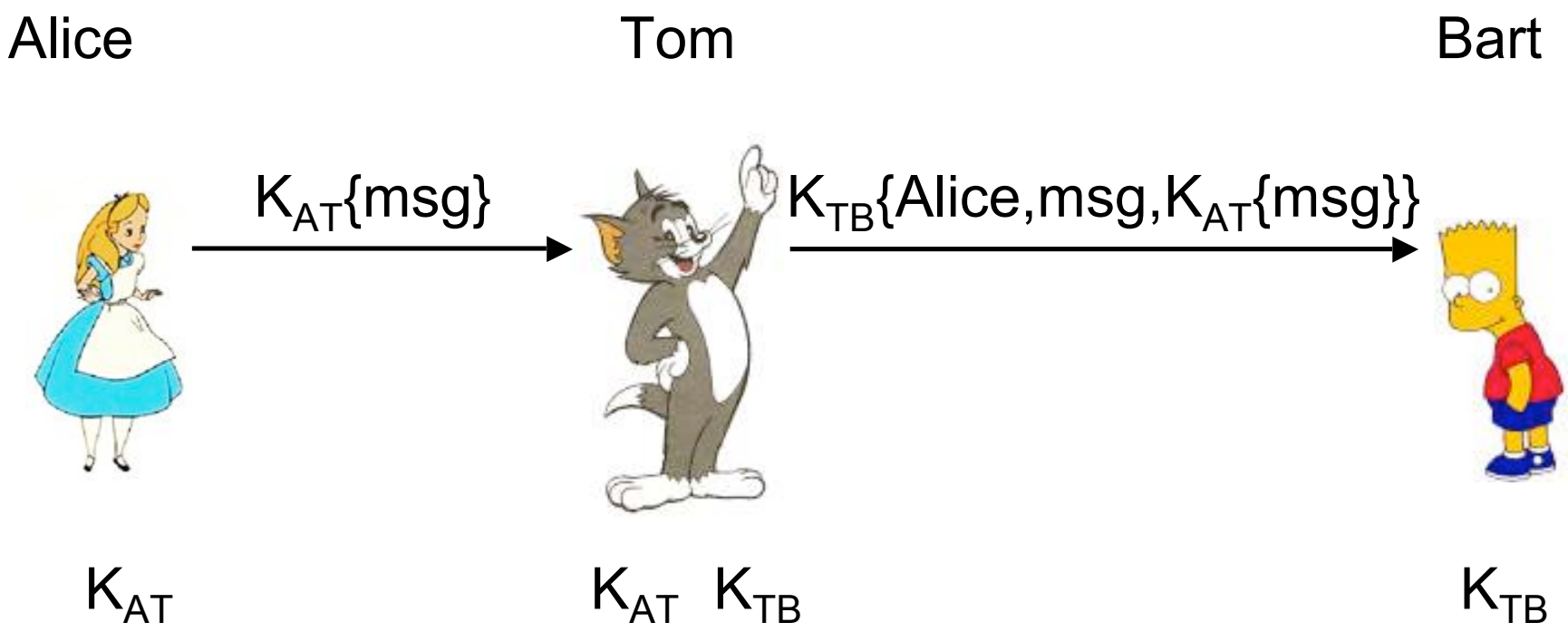
Digital Signatures: Requirements I

- A mark that only one principal can make, but others can easily recognize
- Unforgeable
  - If P signs a message M with signature $S_P\{M\}$ it is impossible for any other principal to produce the pair (M, $S_P\{M\}$).
- Authentic
  - If R receives the pair (M, $S_P\{M\}$) purportedly from P, R can check that the signature really is from P.

Digital Signatures: Requirements II

- Not alterable
  - After being transmitted, $(M,S_P\{M\})$ cannot be changed by P, R, or an interceptor.
- Not reusable
  - A duplicate message will be detected by the recipient.
- Nonrepudiation:
  - P should not be able to claim they didn't sign something when in fact they did.
  - (Related to unforgeability: If P can show that someone else could have forged P's signature, they can repudiate ("refuse to acknowledge") the validity of the signature.)

Digital Signatures with Shared Keys



Tom is a trusted 3rd party (or arbiter).
Authenticity: Tom verifies Alice's message, Bart trusts Tom.
No Forgery: Bart can keep msg, $K_{AT}\{msg\}$, which only Alice (or Tom, but he's trusted not to) could produce

Preventing Reuse and Alteration

- To prevent reuse of the signature
  - Incorporate a *timestamp* (or sequence number)
- Alteration
  - If a block cipher is used, recipient could splice-together new messages from individual blocks.
- To prevent alteration
  - Timestamp must be part of each block
  - Or… use *cipher block chaining*

Digital Signatures with Public Keys

- Assumes the algorithm is *commutative*:
  - D(E(M, K), k) = E(D(M, k), K)
- Let K_A be Alice's public key
- Let k_A be her private key
- To sign msg, Alice sends D(msg, k_A)
- Bart can verify the message with Alice's public key

- Works! RSA: $(m^e)^d = m^{ed} = (m^d)^e$

Digital Signatures with Public Keys

Alice

Bart

$k_A\{msg\}$

k_A, K_A, K_B

k_B, K_B, K_A

- No trusted 3rd party.
- Simpler algorithm.
- More expensive
- No confidentiality

Variations on Public Key Signatures

- Timestamps again (to prevent replay)
  - Signed certificate valid for only some time.
- Add an extra layer of encryption to guarantee confidentiality
  - Alice sends $K_B\{k_A\{msg\}\}$ to Bart
- Combined with hashes:
  - Send (msg, $k_A\{MD5(msg)\}$)

Key Establishment

- Establishing a "session key"
  - A shared key used for encrypting communications for a short duration -- a session
  - Need to authenticate first

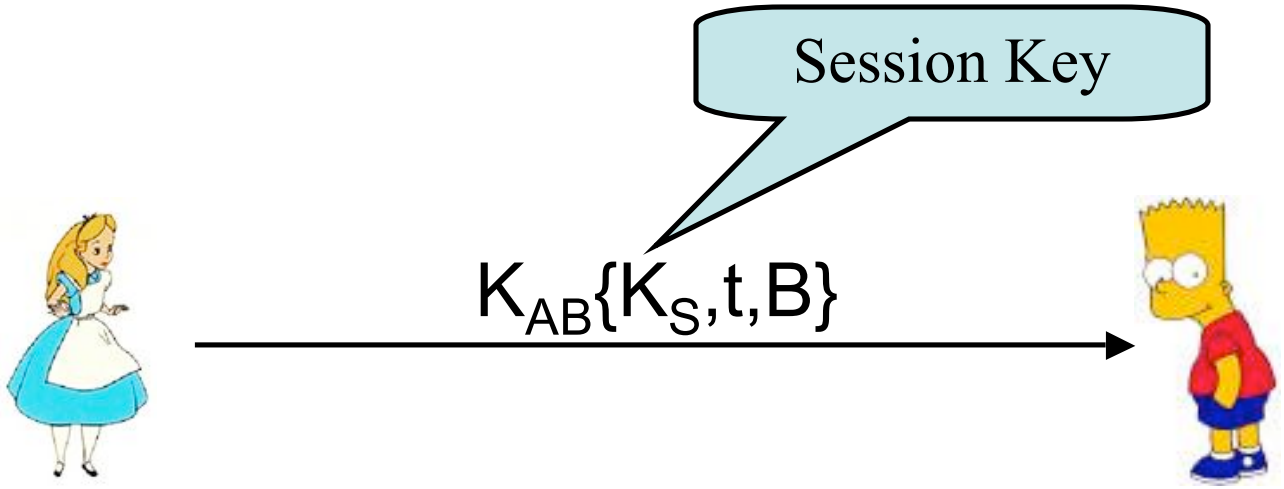
- Symmetric keys.
  - Point-to-Point.
  - Needham-Schroeder.
  - Kerberos.

Symmetric Keys

- Key establishment using only symmetric keys requires use of pre-distribution keys to get things going.

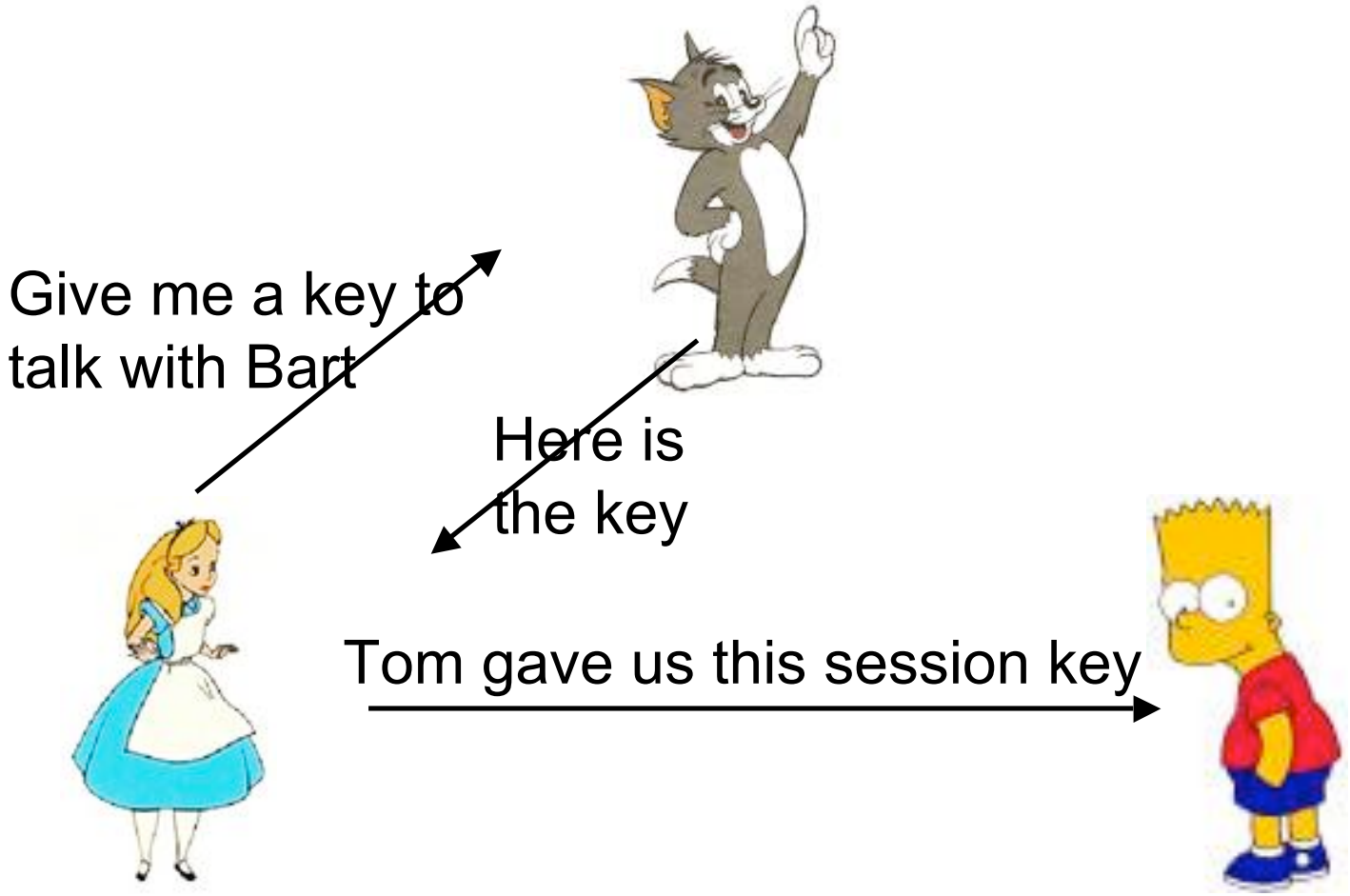
- Then protocol can be based on:
  - Point to point distribution, or
  - Key Distribution Center (KDC).

Point-to-Point



- Should also use timestamps & nonces.
- Session key should include a validity duration.
- Could also use public key cryptography to
  - Authenticate
  - Exchange symmetric shared key

Key Distribution Centers



Distribution Center Setup

- A wishes to communicate with B.
- T (trusted 3rd party) provides session keys.
- T has a key K_{AT} in common with A and a key K_{BT} in common with B.
- A authenticates T using a nonce n_A and obtains a session key from T.
- A authenticates to B and transports the session key securely.

Needham-Schroeder Protocol

1. $A \rightarrow T : \quad A, B, n_A$
2. $T \rightarrow A : \quad K_{AT}\{K_S, n_A, B, K_{BT}\{K_S, A\}\}$

   A decrypts with K_{AT} and checks n_A and B. Holds K_S for future correspondence with B.
3. $A \rightarrow B : \quad K_{BT}\{K_S, A\}$

   B decrypts with K_{BT}.
4. $B \rightarrow A : \quad K_S\{n_B\}$

   A decrypts with K_S.
5. $A \rightarrow B : \quad K_S\{n_B - 1\}$

   B checks n_B-1.

Attack Scenario 1

1. $A \rightarrow T$: A, B, n_A
2. $T \rightarrow C\ (A)$: $K_{AT}\{k, n_A, B, K_{BT}\{K_S, A\}\}$

   C is unable to decrypt the message to A; passing it along unchanged does no harm. Any change will be detected by A.

Attack Scenario 2

1. $A \rightarrow C\ (T)$: A, B, n_A
2. $C\ (A) \rightarrow T$: A, C, n_A
3. $T \rightarrow A$: $K_{AT}\{K_S, n_A, C, K_{CT}\{K_S, A\}\}$

Rejected by A because the message contains C rather than B.

Attack Scenario 3

1. $A \rightarrow C\ (T) :$ A, B, n_A
2. $C \rightarrow T : C, B, n_A$
3. $T \rightarrow C : K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C\}\}$
4. $C\ (T) \rightarrow A :$ $K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C\}\}$

A is unable to decrypt the message.

Attack Scenario 4

1. $C \rightarrow T : C, B, n_A$
2. $T \rightarrow C : K_{CT}\{K_S, n_A, B, K_{BT}\{K_S, C\}\}$
3. $C (A) \rightarrow B : K_{BT}\{K_S, C\}$

B will see that the purported origin (A) does not match the identity indicated by the distribution center.

Valid Attack

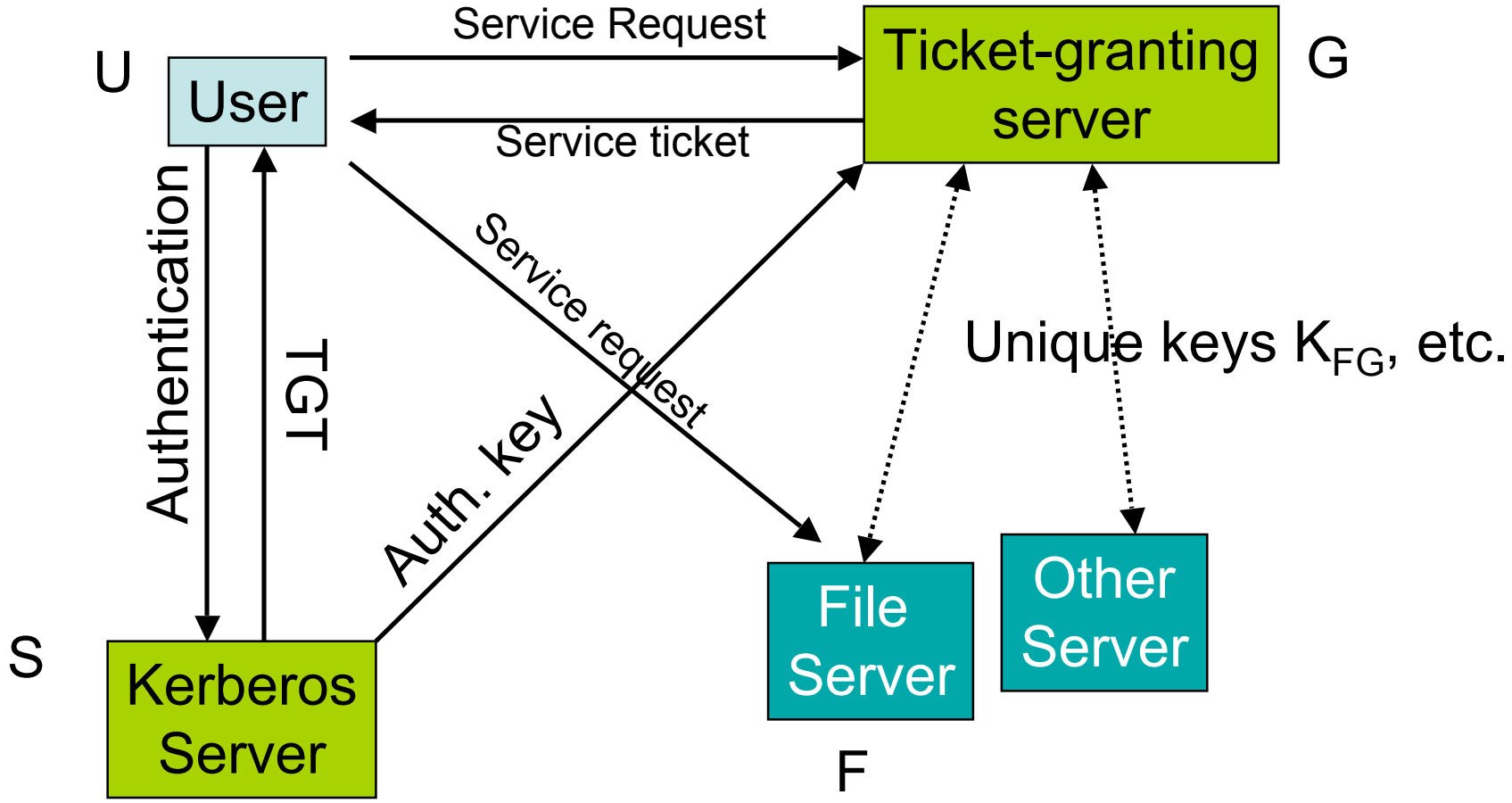
- The attacker records the messages on the network
  - in particular, the messages sent in step 3
- Consider an attacker that manages to get an old session key K_S.
- That attacker can then masquerade as Alice:
  - Replay starting from step 3 of the protocol, but using the message corresponding to K_S.

- Could be prevented with time stamps.

Kerberos

- Key exchange protocol developed at MIT in the late 1980's
- Central server provides "tickets"
- *Tickets* – (also known as *capabilities*):
  - Unforgeable
  - Nonreplayable
  - Authenticated
  - Represent authority
- Designed to work with NFS (network file system)
- Also saves on authenticating for each service
  - e.g. with ssh.

Kerberos


Service Request
U
User
Ticket-granting server
G
Service ticket
Authentication
TGT
Service request
Auth. key
Unique keys K_{FG}, etc.
S
Kerberos Server
File Server
Other Server
F

Kerberos Login

- U = User's machine
- S = Kerberos Server
  - Has a database of user "passwords": userID → k_{pwd}
- G = Ticket granting server

- U → S : userID, G, n_U
- S → U : $k_{pwd}\{n_U, K_{UG}\}$, $K_{SG}\{T(U,G)\}$
- S → G : $K_{SG}\{K_{UG}$, userID$\}$

Kerberos ticket granting ticket

- T(X,Y) = X, Y, L, K_{XY}

Session key

Ticket lifetime

Kerberos Service Request

- Requesting a service from server F
- $U \rightarrow G$: $K_{UG}\{userID,timestamp\}, K_{SG}\{T(U,G)\}, req(F), n'_U$
- $G \rightarrow U$: $K_{UG}\{K_{UF},n'_U\}, K_{FG}\{T(U,F)\}$
- $U \rightarrow F$: $K_{UF}\{userID,timestamp\}, K_{FG}\{T(U,F)\}$

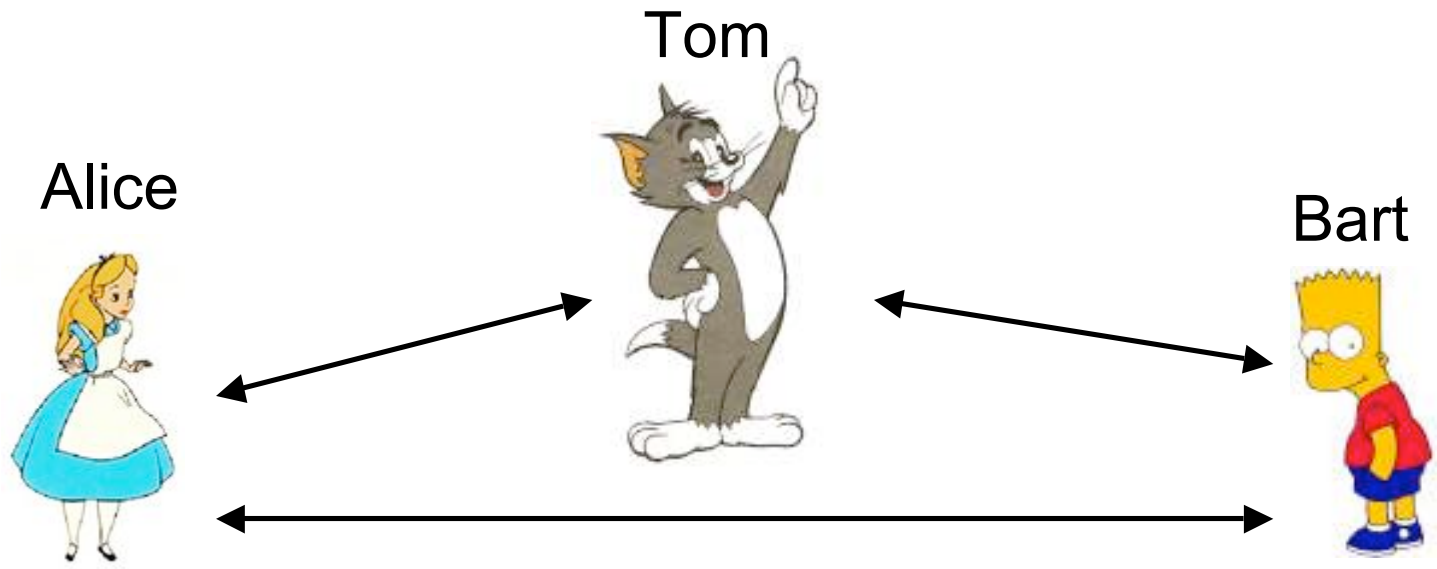
Kerberos Benefits

- Distributed access control
  - No passwords communicated over the network
- Cryptographic protection against spoofing
  - All accesses mediated by G (ticket granting server)
- Limited period of validity
  - Servers check timestamps against ticket validity
  - Limits window of vulnerability
- Timestamps prevent replay attacks
  - Servers check timestamps against their own clocks to ensure “fresh” requests
- Mutual authentication
  - User sends nonce challenges

Kerberos Drawbacks

- Requires available ticket granting server
  - Could become a bottleneck
  - Must be reliable
- All servers must trust G, G must trust servers
  - They share unique keys
- Kerberos requires synchronized clocks
  - Replay can occur during validity period
  - Not easy to synchronize clocks
- User's machine could save & replay passwords
  - Password is a weak spot
- Kerberos does not scale well
  - Hard to replicate authentication server and ticket granting server
  - Duplicating keys is bad, extra keys = more management

Arbitrated Protocols

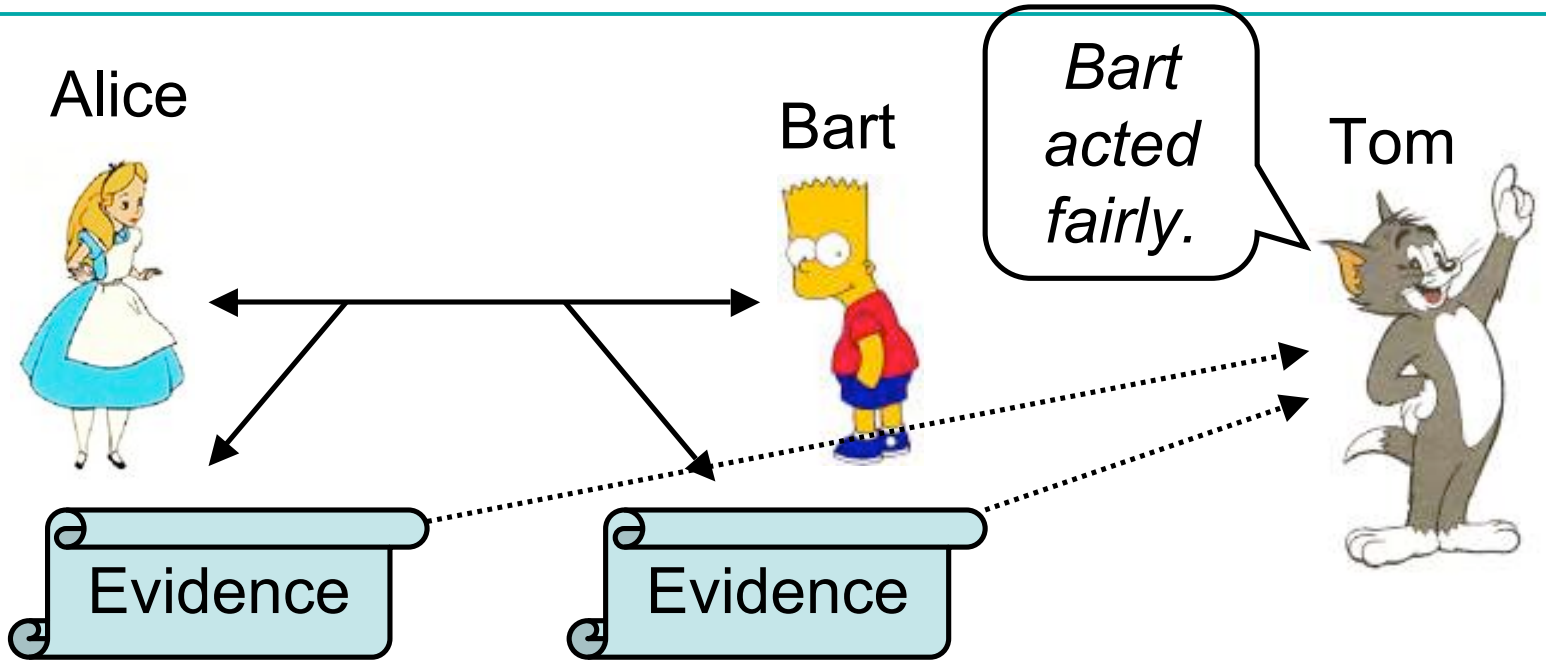


- Tom is an *arbiter*
  - Disinterested in the outcome (doesn't play favorites)
  - Trusted by the participants (Trusted 3rd party)
  - Protocol can't continue without T's participation

Arbitrated Protocols (Continued)

- Real-world examples:
  - Lawyers, Bankers, Notary Public
- Issues:
  - Finding a trusted 3rd party
  - Additional resources needed for the arbitrator
  - Delay (introduced by arbitration)
  - Arbitrator might become a bottleneck
  - Single point of vulnerability: attack the arbitrator!

Adjudicated Protocols



- Alice and Bard record an *audit log*
- Only in exceptional circumstances to they contact a trusted 3rd party. (3rd party is not always needed.)
- Tom as the *adjudicator* can inspect the evidence and determine whether the protocol was carried out fairly

Self-Enforcing Protocols

- No trusted 3rd party involved.
- Participants can determine whether other parties cheat.
- Protocol is constructed so that there are no possible disputes of the outcome.