CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 19

Announcements

- Plan for Today:
  - Key establishment
- Project 3 is due 6 April 2009 at 11:59 pm
- Midterm 2 is this Thursday in class
  - Covers material since the first midterm
- Final exam has been scheduled:
  Friday, May 8, 2009
  9:00am – 11:00am, Moore 216
- TALK: Alan Mislove "Leveraging Social Networks in Information Systems" 3:00 today in Wu & Chen

Key Establishment

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

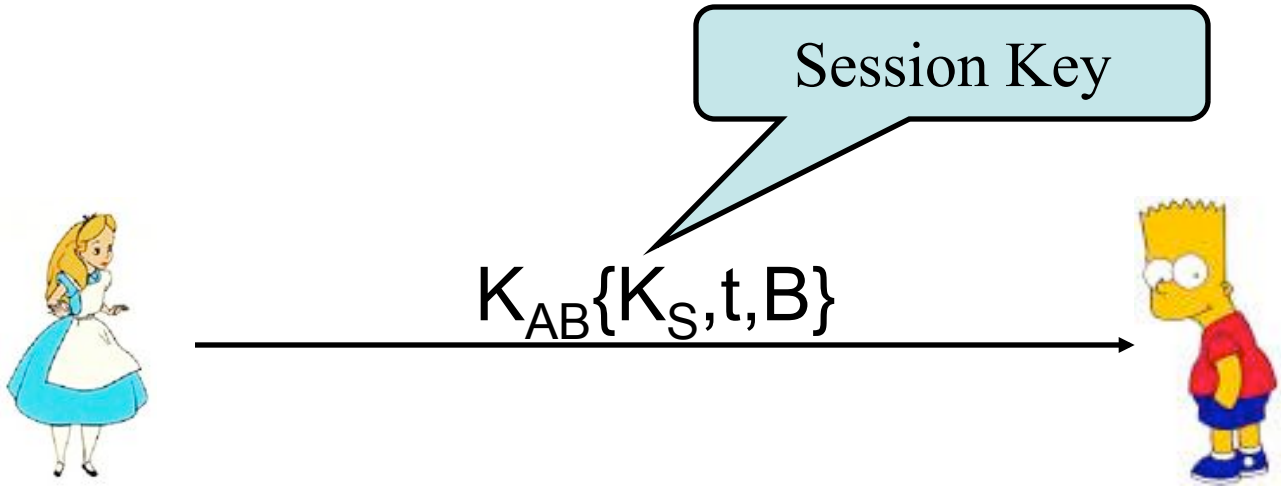
- Symmetric keys.
  - Point-to-Point.
  - Needham-Schroeder.
  - Kerberos.

Symmetric Keys

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

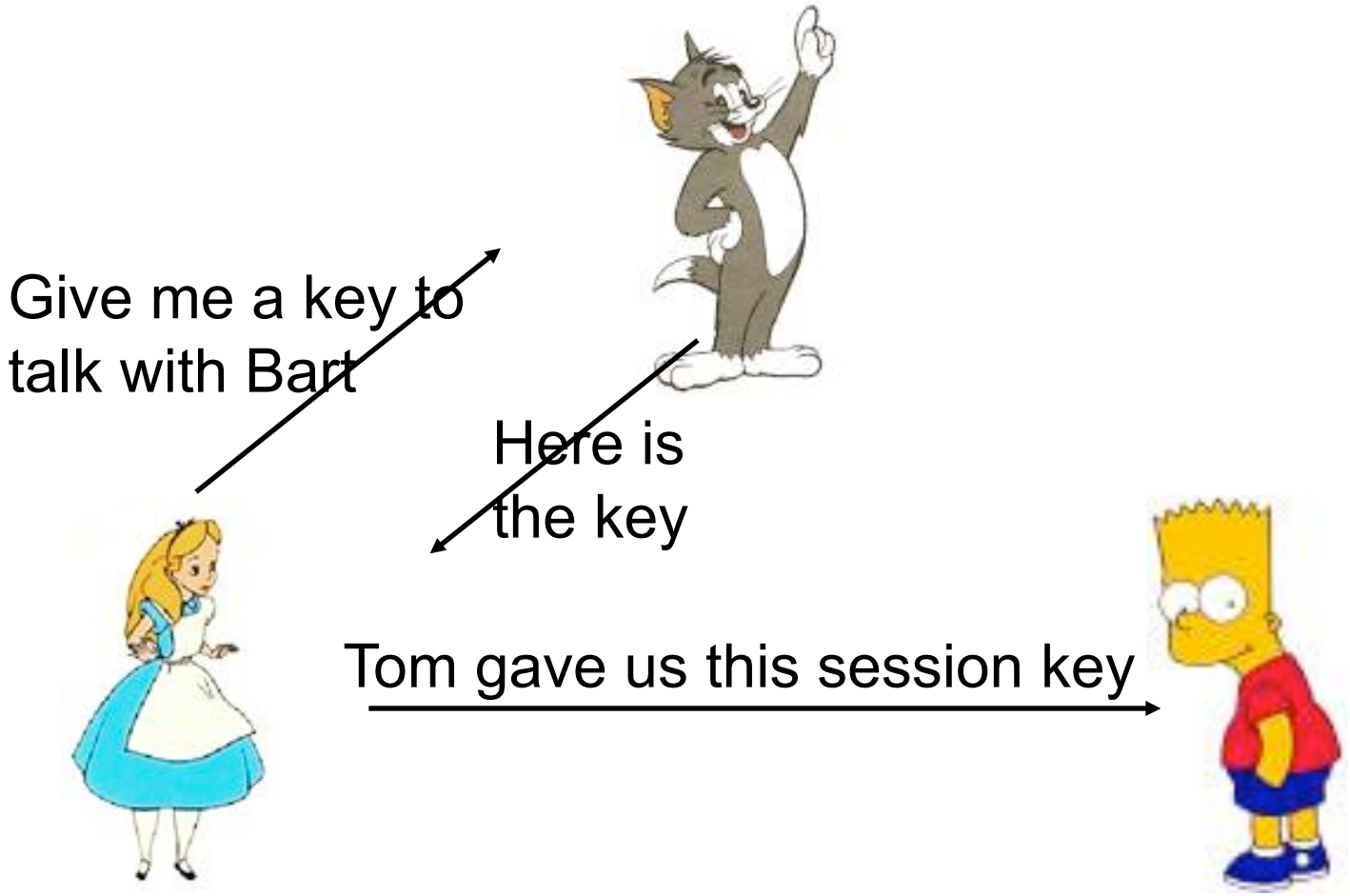
- Then protocol can be based on:
  - Point to point distribution, or
  - Key Distribution Center (KDC).

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

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

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

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

   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. $C\ (A) \rightarrow B : \quad K_{BT}\{K_S, C\}$

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

Valid Attack

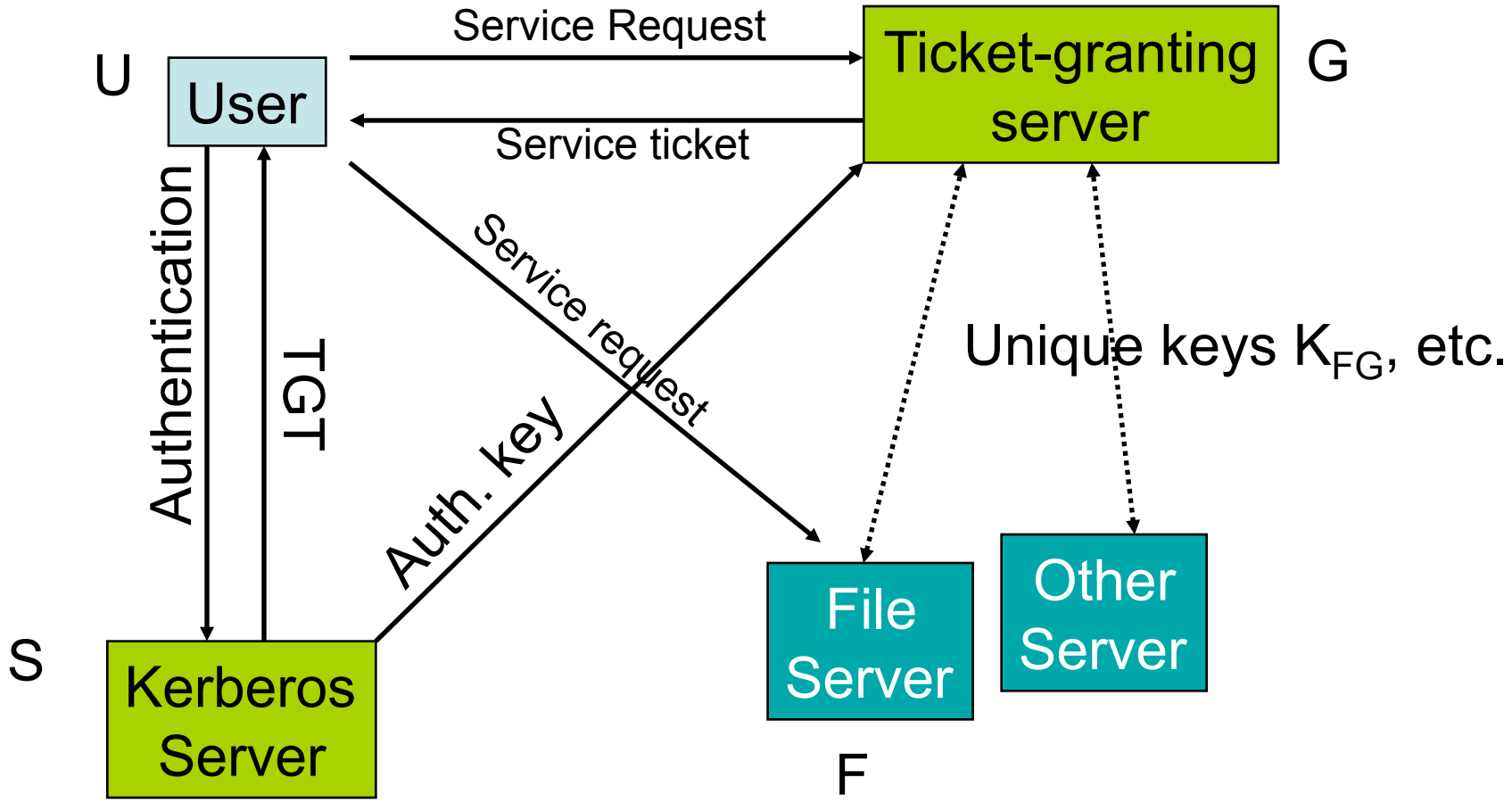
- The attacker records the messages on the network
  - in particular, the messages sent in step 3
- Consider an attacker that manages to get an old session key K_S.
- That attacker can then masquerade as Alice:
  - Replay starting from step 3 of the protocol, but using the message corresponding to K_S.

- 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
Ticket-granting server
G
U
User
Service ticket
Authentication
TGT
Service request
Auth. key
Unique keys K_{FG}, etc.
File Server
Other Server
S
Kerberos 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 : $K_{SG}\{K_{UG}, \text{userID}\}$

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

Public Key Infrastructure (PKI)

- *Public key infrastructure* (PKI)
  - PKI is the set of services needed to create, manage, store, distribute and revoke digital certificates based on public-key cryptography.

- Certification Authorities (CAs)
  - A trusted third party that issues certificates and (often) certificate revocation lists.
  - Each certificate is (roughly) of the form M, $k_{CA}\{H(M)\}$
    where M = Name, K_{Name}, L
    Name = identifier of a principal (e.g. a URL)
    K_{Name} = the public key of the principal
    L = lifetime of the certificate
- Example: Verisign
  - Issues credentials

X.509 Certificate Standard

- Issued in 1988 by the PKIX working group of the IETF
- Message format that specifies how certificates should be shared:

```
Certificate
        Version, Serial Number, Algorithm ID
        Issuer, Validity (Not Before, Not After)
        Subject, Subject Public Key Info (Algorithm, Key)
        Issuer Unique Identifier (Optional)
        Subject Unique Identifier (Optional)
        Extensions (Optional)
 Certificate Signature Algorithm
 Certificate Signature
```

Example X.509 certificate

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   [...]
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
      [...]
```

Top-level Certificates

- To check an X.509 certificate, one needs to have the public key of the issuer.
- Such certificates can be “self-signed” by top-level, trusted CAs
- In practice, companies like Verisign pay web browser developers to include such certificates in the browser releases.

Certificate Chains

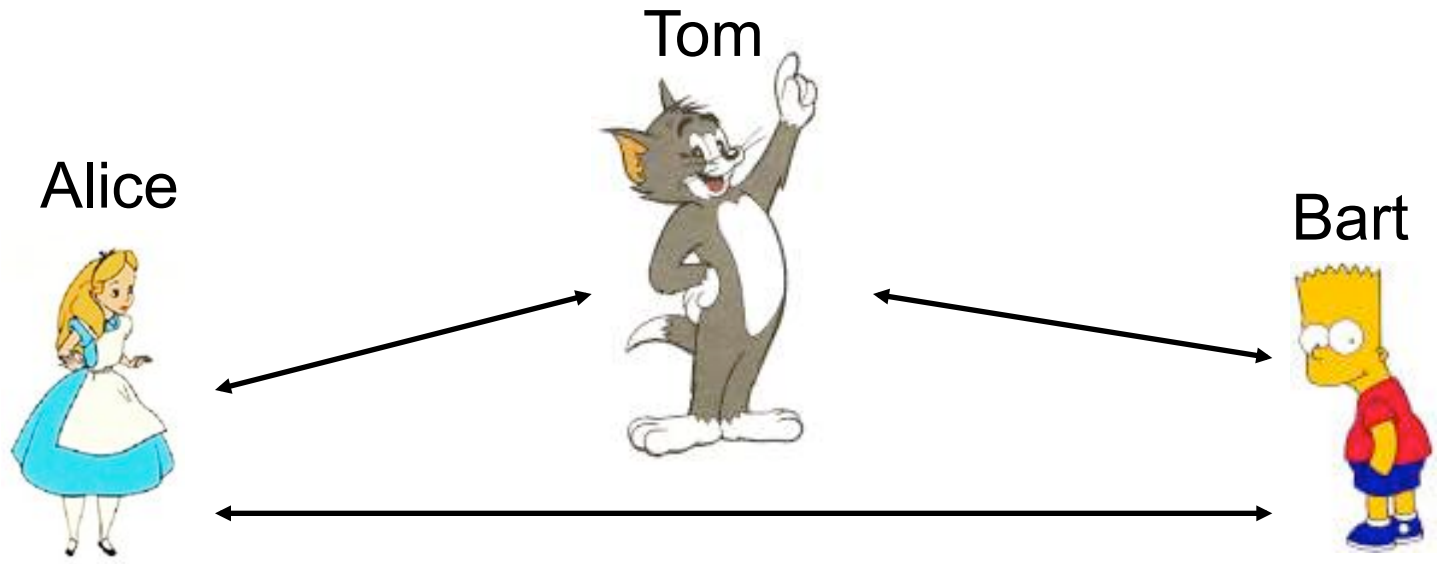
- Notation: Y <<X>> means the certificate of principal X issued by authority Y.
- One can create *certificate chains* to delegate authentication duties among principals:
- Example:

  Y <<X>>, X <<Z>>

  – These two certificates together allow a principal who trusts Y to verify the authenticity of the identity of Z.

- Chains can be arbitrarily long.
  – CAs can attest to each other's identities via peering agreements

Arbitrated Protocols

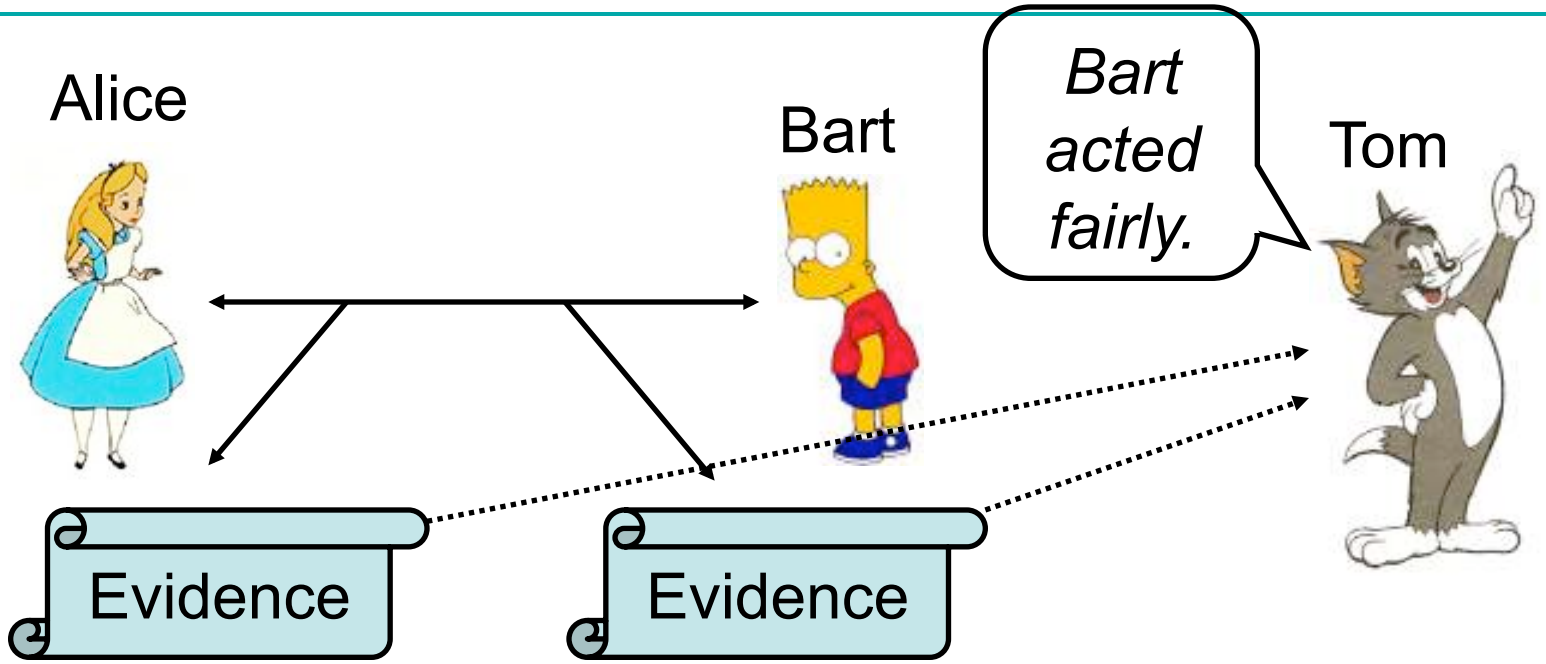


- Tom is an *arbiter*
  - Disinterested in the outcome (doesn't play favorites)
  - Trusted by the participants (Trusted 3rd party)
  - Protocol can't continue without T's participation

Arbitrated Protocols (Continued)

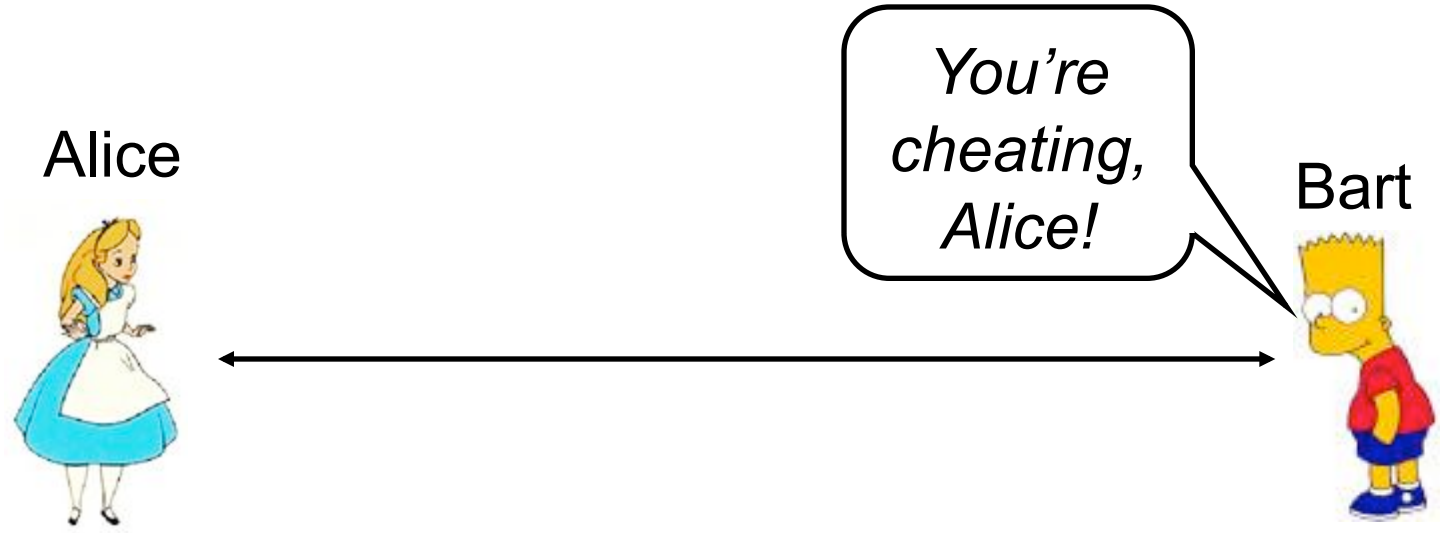
- Real-world examples:
  - Lawyers, Bankers, Notary Public
- Issues:
  - Finding a trusted 3rd party
  - Additional resources needed for the arbitrator
  - Delay (introduced by arbitration)
  - Arbitrator might become a bottleneck
  - Single point of vulnerability: attack the arbitrator!

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.