CIS 551 / TCOM 401

Computer and Network Security

Spring 2006

Lecture 11

Announcements

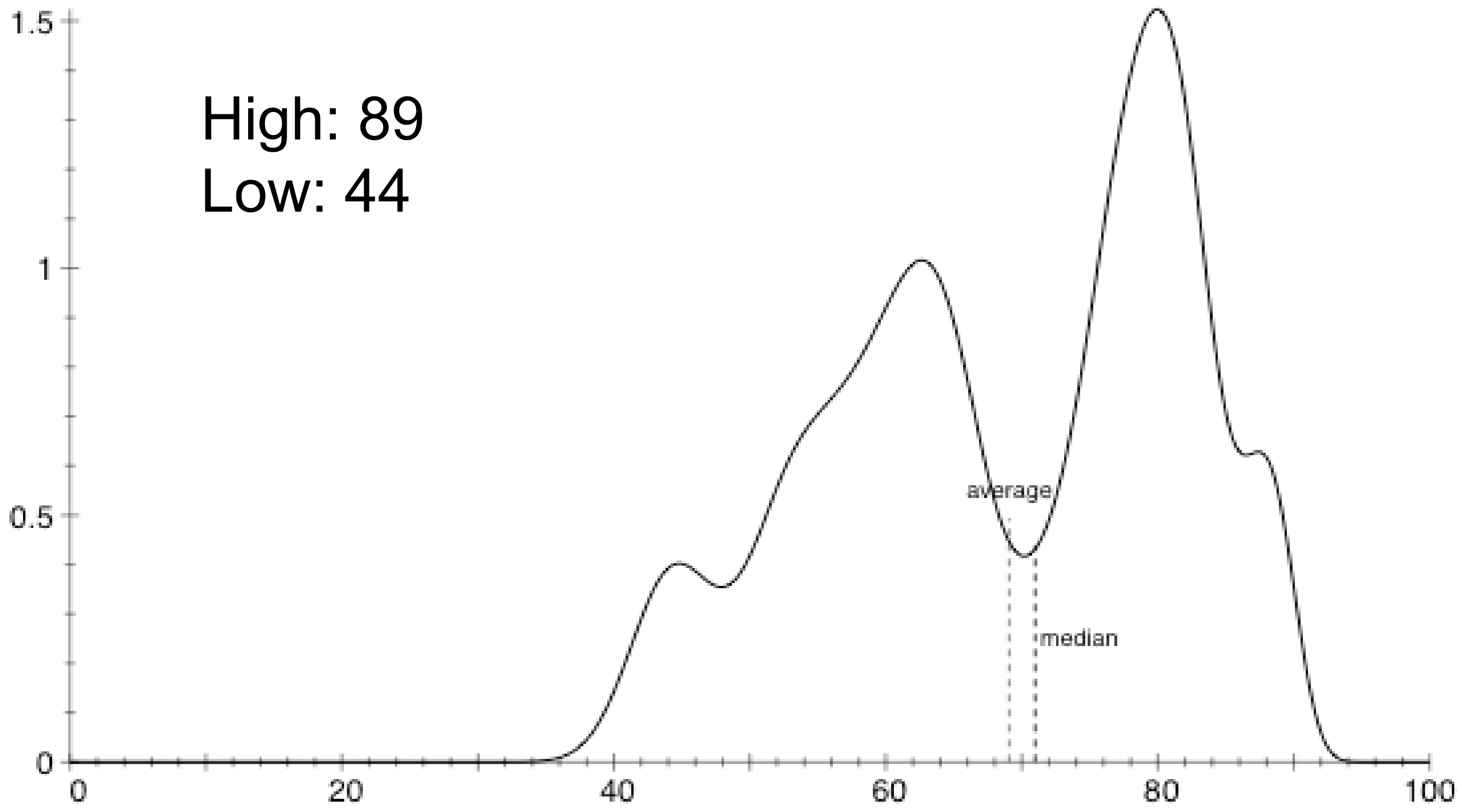
- Project 2 is available on the web.
  - Due: March 14, 2006

- Project 1 has been graded
  - You should have received e-mail.
  - We will be putting up the grading guidelines on the web shortly

- Midterm 1 has been graded

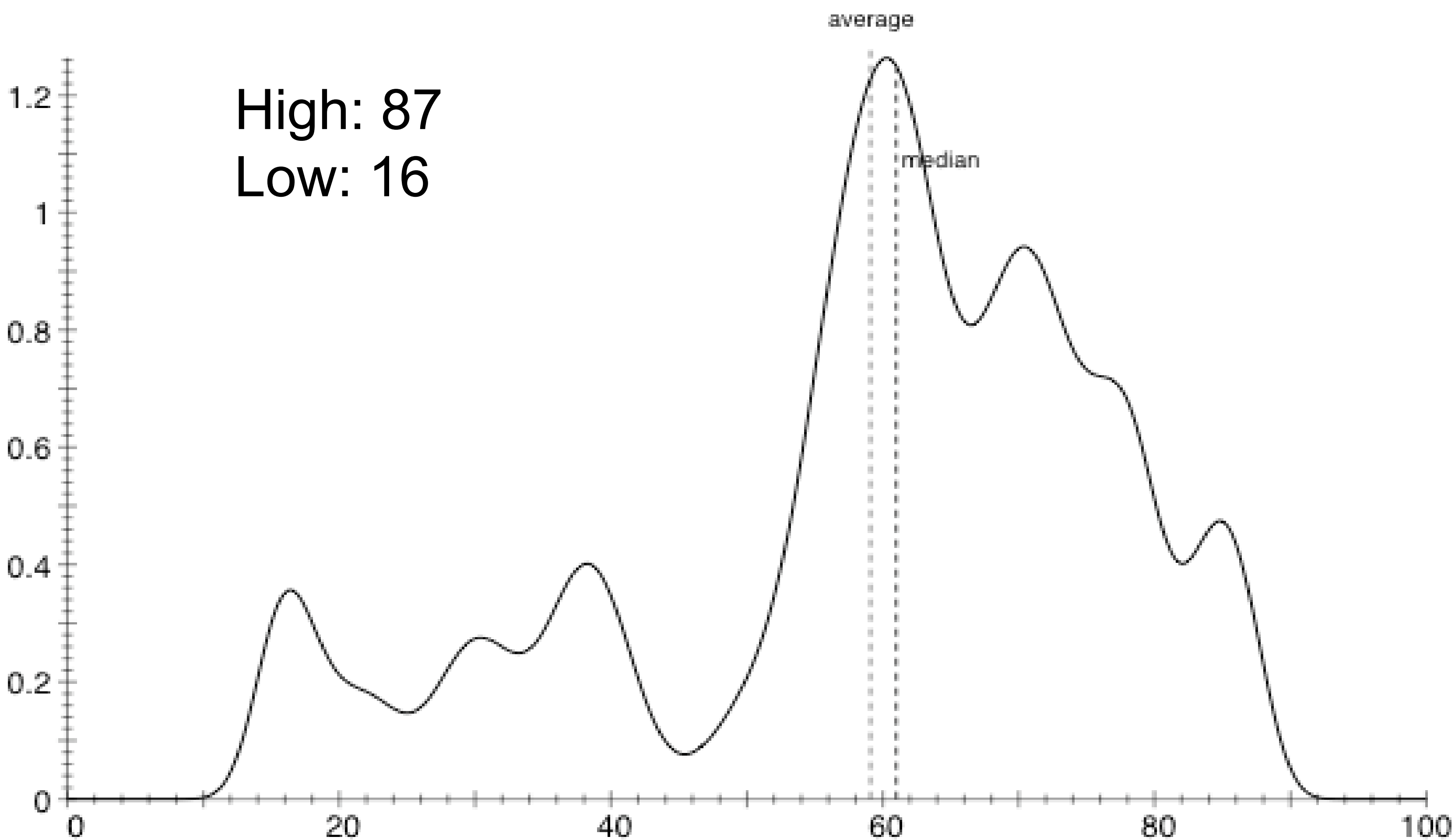
Midterm 1 distribution

High: 89
Low: 44



Project 1 Distribution

High: 87
Low: 16



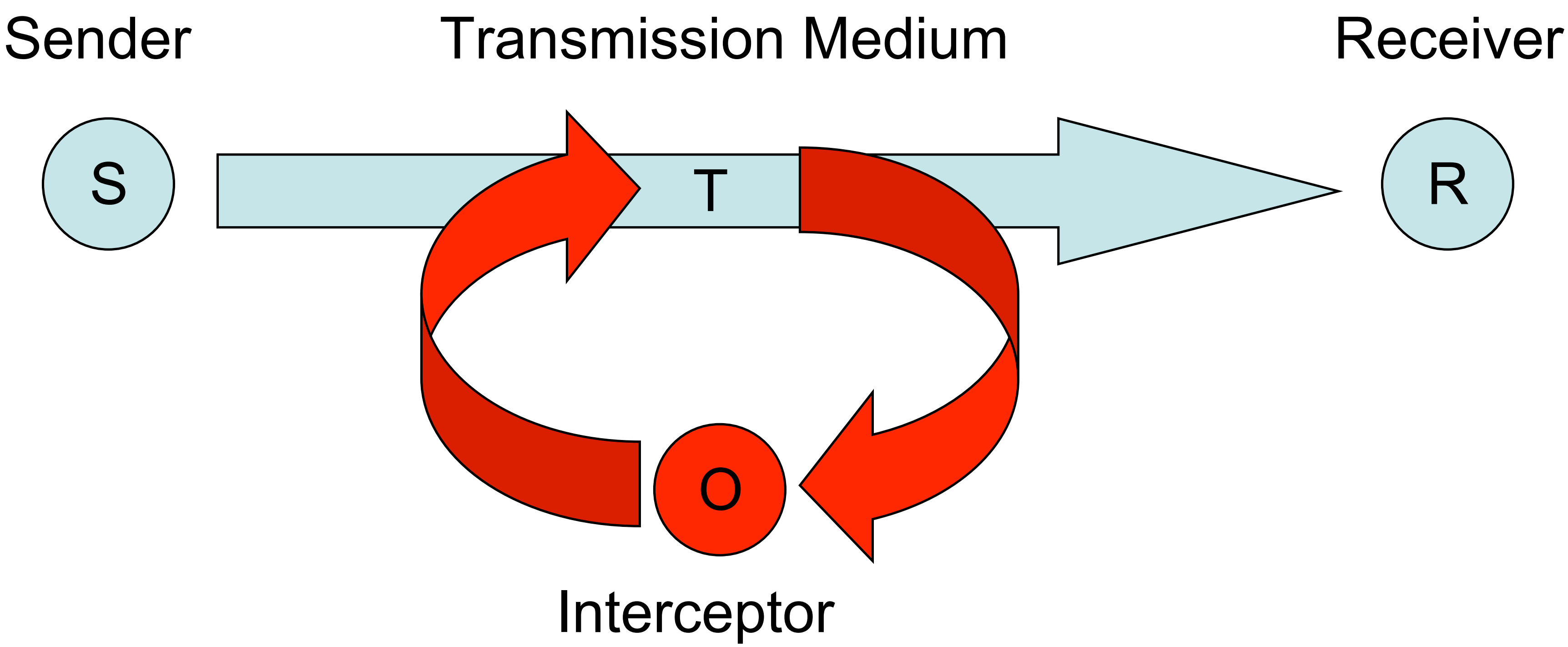
General Definition of "Protocol"

- A *protocol* is a multi-party algorithm
  - A sequence of steps that precisely specify the actions required of the parties in order to achieve a specified objective.

- Important that there are multiple participants
- Typically a situation of heterogeneous trust
  - Alice may not trust Bart
  - Bart may not trust the network

Cryptographic Protocols

- Consider communication over a network…
- What is the threat model?
  - What are the vulnerabilities?



What Can the Attacker Do?

- Intercept them (confidentiality)
- Modify them (integrity)
- Fabricate other messages (integrity)
- Replay them (integrity)

- Block the messages (availability)
- Delay the messages (availability)
- Cut the wire (availability)
- Flood the network (availability)

Dolev-Yao Model

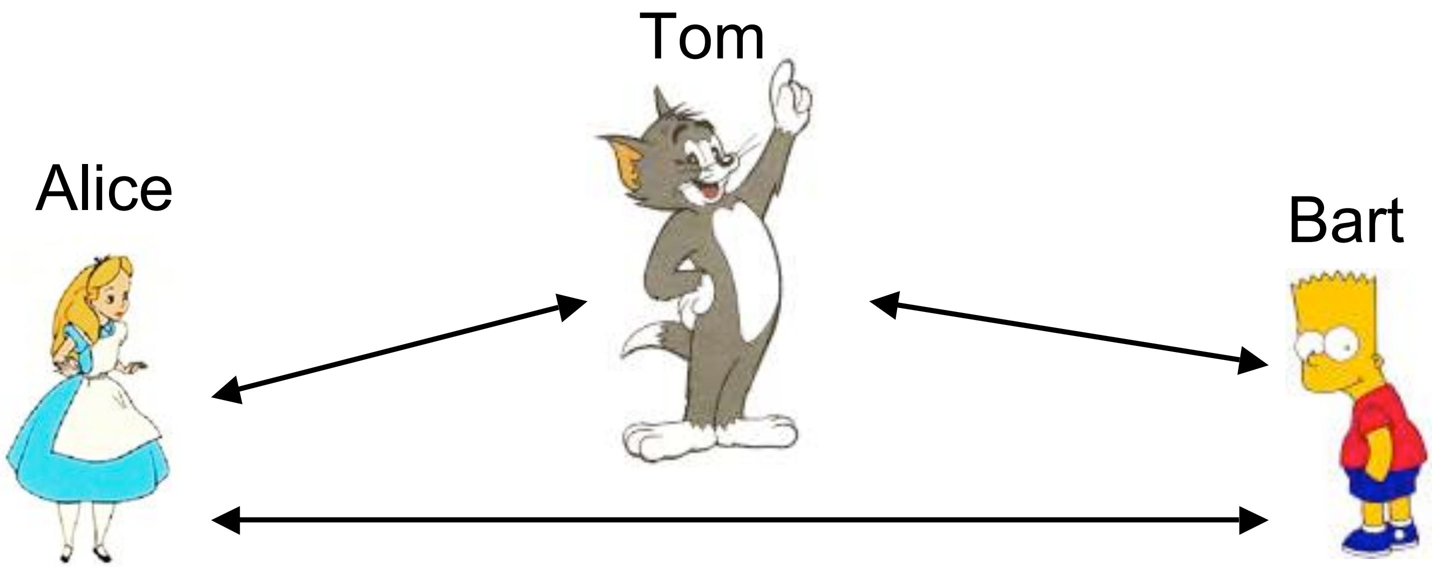
- Treat cryptographic operations as "black box"
- Simplifies reasoning about protocols (doesn't require reduction to computational complexity)

- Given a message M = (c1,c2,c3,…) attacker can deconstruct message into components c1 c2 c3
- Given a collection of components c1, c2, c3, attacker can forge message (c1,c2,c3)
- Given an encrypted object K{c}, attacker can learn c only if attacker knows decryption key corresponding to K
- Attacker can encrypt components by using:
  - fresh keys, or
  - keys they have learned during the attack

Characteristics of Protocols

- Every participant must know the protocol and the steps in advance.
- Every participant must agree to follow the protocol
  - *Honest participants*
- Big problem: How to deal with bad participants?
  - 3 basic kinds of protocols

Arbitrated Protocols



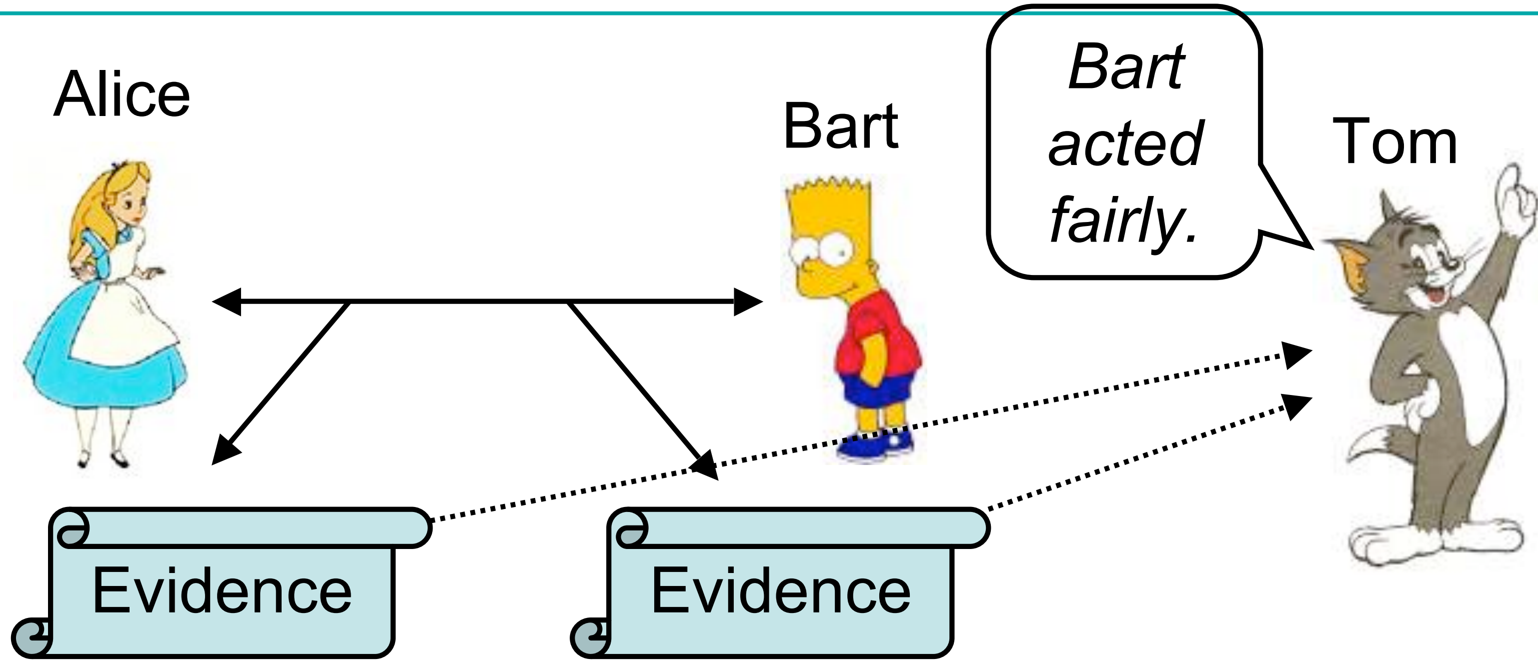
- Tom is an *arbiter*
  - Disinterested in the outcome (doesn’t play favorites)
  - Trusted by the participants (Trusted 3rd party)
  - Protocol can’t continue without T’s participation

Arbitrated Protocols (Continued)

- Real-world examples:
  - Lawyers, Bankers, Notary Public

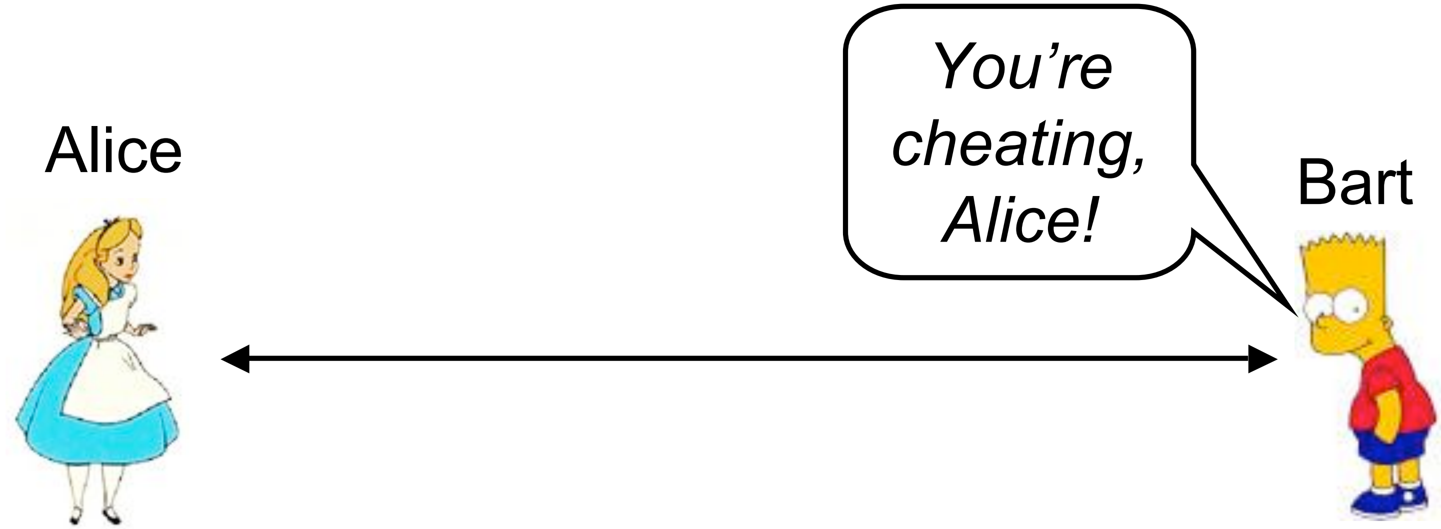
- Issues:
  - Finding a trusted 3^{rd} party
  - Additional resources needed for the arbitrator
  - Delay (introduced by arbitration)
  - Arbitrator might become a bottleneck
  - Single point of vulnerability: attack the arbitrator!

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.

Authentication

- For honest parties, the claimant A is able to authenticate itself to the verifier B. That is, B will complete the protocol having accepted A’s identity.

Shared-Key Authentication

Alice

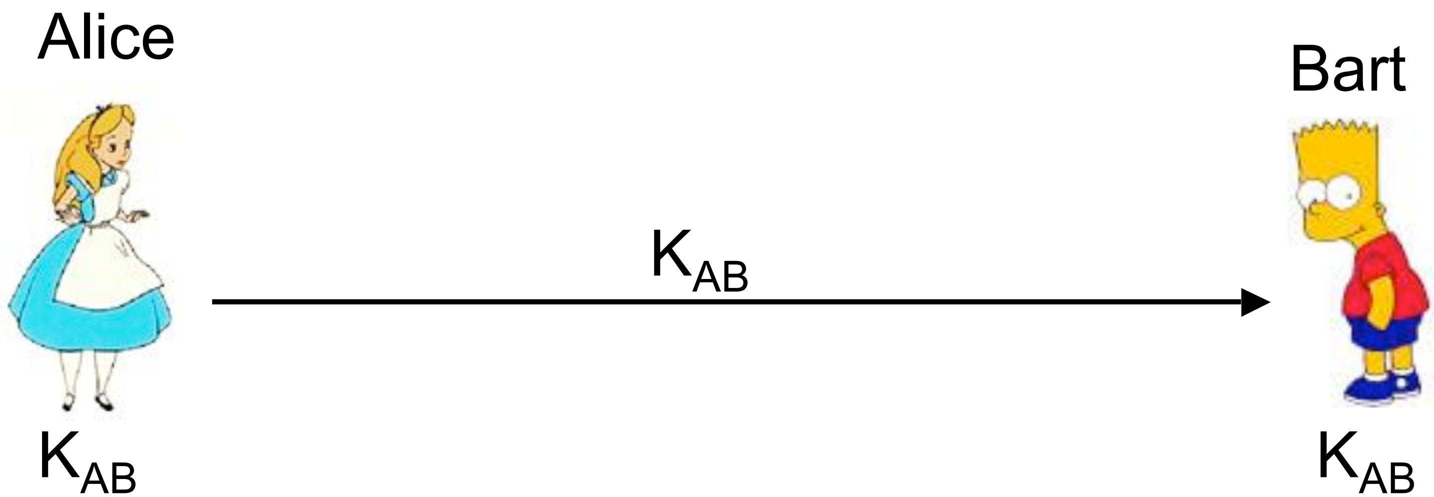
K_{AB}

Bart

K_{AB}

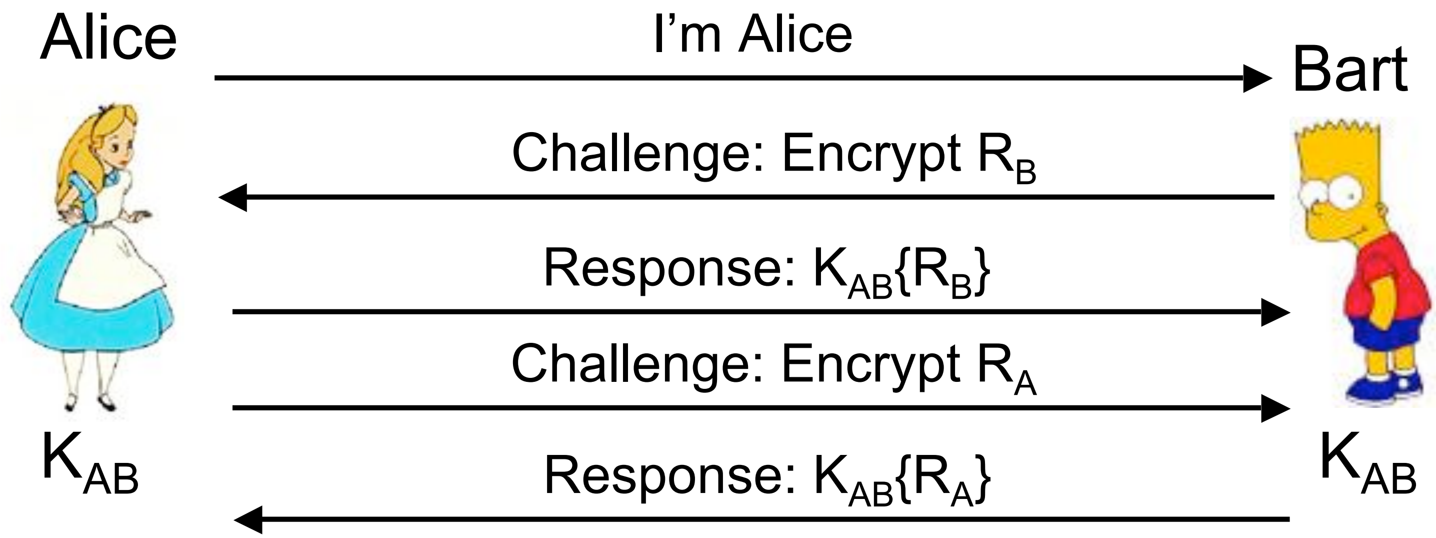
- Assume Alice & Bart already share a key K_{AB}.
  - The key might have been decided upon in person or obtained from a trusted 3rd party.
- Alice & Bart now want to communicate over a network, but first wish to authenticate to each other

Solution 1: Weak Authentication



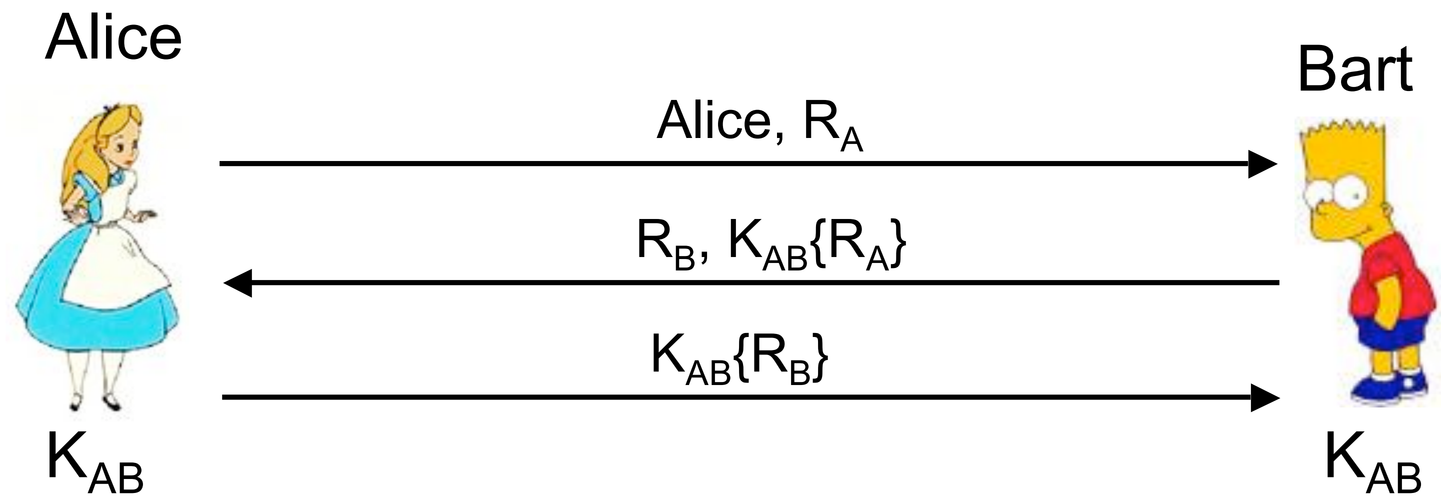
- Alice sends Bart $K_{AB.}$
  - K_{AB} acts as a password.
- The secret (key) is revealed to passive observers.
- Only works one-way.
  - Alice doesn't know she's talking to Bart.

Solution 2: Strong Authentication



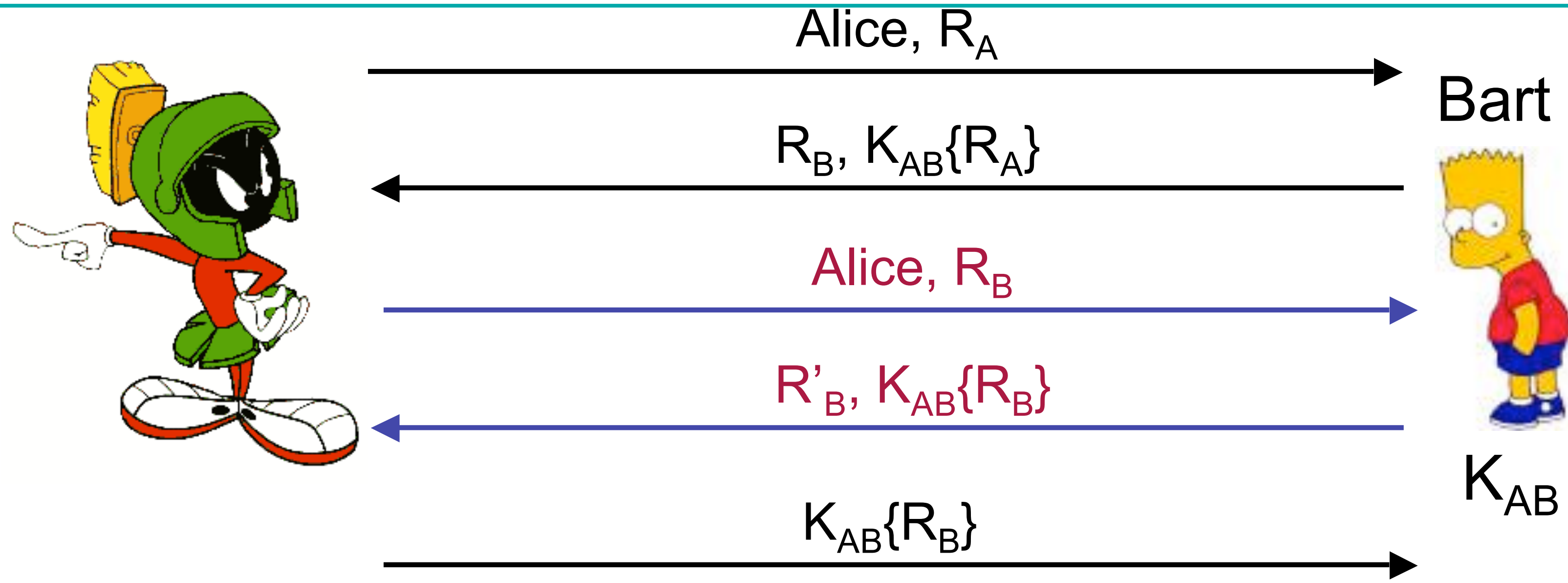
- Protocol doesn't reveal the secret.
- *Challenge/Response*
  - Bart requests proof that Alice knows the secret
  - Alice requires proof from Bart
  - R_A and R_B are randomly generated numbers

(Flawed) Optimized Version



- Why not send more information in each message?
- This seems like a simple optimization.
- But, it’s broken… how?

Attack: Marvin can Masquerade as Alice



- Marvin pretends to take the role of Alice in two runs of the protocol.
  - Tricks Bart into doing Alice’s part of the challenge!
  - Interleaves two instances of the same protocol.

Lessons

- Protocol design is tricky and subtle
  - “Optimizations” aren’t necessarily good
- Need to worry about:
  - Multiple instances of the same protocol running in parallel
  - Intruders that play by the rules, mostly
- General principle:
  - Don’t do anything more than necessary until confidence is built.
  - Initiator should prove identity *before* responder takes action (like encryption)