CIS 551 / TCOM 401

Computer and Network Security

Spring 2007

Lecture 16

Announcements

- Project 2 is due on Thursday at 11:59 pm.

- Midterm II is coming up soon.
  - Originally Scheduled for next Tuesday.
  - Postpone to Thursday instead?

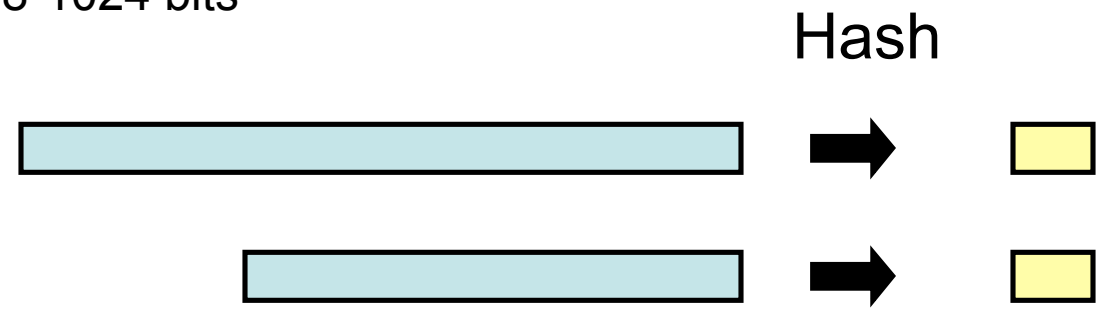
Recap

- Before Spring Break:
  - Cryptography
  - DES / AES (shared key cryptography)
  - RSA (public key cryptography)
  - Diffie-Hellman Key Exchange

- Today:
  - Cryptographic Hashes
  - Dolev-Yao model
  - Authentication Protocols

Hash Algorithms

- Take a variable length string
- Produce a fixed length digest
  - Typically 128-1024 bits



- (Noncryptographic) Examples:
  - Parity (or byte-wise XOR)
  - CRC (cyclic redundancy check) used in communications
  - Ad hoc hashes used for hash tables
- Realistic Example
  - The NIST Secure Hash Algorithm (SHA) takes a message of less than 2^{64} bits and produces a digest of 160 bits

Cryptographic Hashes

- Create a hard-to-invert summary of input data
- Useful for integrity properties
  - Sender computes the hash of the data, transmits data and hash
  - Receiver uses the same hash algorithm, checks the result
- Like a check-sum or error detection code
  - Uses a cryptographic algorithm internally
  - More expensive to compute
- Sometimes called a Message Digest
- History:
  - Message Digest (MD4 -- invented by Rivest, MD5)
  - Secure Hash Algorithm - 1993 - (SHA-0)
  - Secure Hash Algorithm (SHA-1)
  - SHA-2 (actually a family of hash algorithms with varying output sizes)

- Attacks have been found against both SHA-0 and SHA-1

Uses of Hash Algorithms

- Hashes are used to protect *integrity* of data
  - Virus Scanners
  - Program fingerprinting in general
  - Modification Detection Codes (MDC)
- Message Authenticity Code (MAC)
  - Includes a cryptographic component
  - Send (msg, hash(msg, key))
  - Attacker who doesn't know the key can't modify msg (or the hash)
  - Receiver who knows key can verify origin of message
- Make digital signatures more efficient (we'll see this later)

Desirable Properties

- The probability that a randomly chosen message maps to an n-bit hash should ideally be $(½)^n$.
  - Attacker must spend a lot of effort to be able to modify the source message without altering the hash value

- Hash functions *h* for cryptographic use as MDC's fall in one or both of the following classes.
  - *Collision Resistant Hash Function:* It should be computationally infeasible to find two distinct inputs that hash to a common value (ie. *h(x) = h(y)*).
  - *One Way Hash Function:* Given a specific hash value *y*, it should be computationally infeasible to find an input x such that *h(x)=y*.

Secure Hash Algorithm (SHA)

- Pad message so it can be divided into 512-bit blocks, including a 64 bit value giving the length of the original message.
- Process each block as 16 32-bit words called *W(t)* for *t* from 0 to 15.
- Expand from these 16 words to 80 words by defining as follows for each t from 16 to 79:
  - W(t) := W(t-3) ⊕ W(t-8) ⊕ W(t-14) ⊕ W(t-16)
- Constants H0, …, H5 are initialized to special constants
- Result is final contents of H0, … , H5

```
for each 16-word block begin
    A := H0; B := H1; C := H2; D := H3; E := H4
    for I := 0 to 19 begin
        TEMP := S(5,A) + ((B ∧ C) ∨ (¬ B ∧ D)) + E + W(I) + 5A827999;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    for I := 20 to 39 begin
        TEMP := S(5,A) + (B ⊕ C ⊕ D) + E + W(I) + 6ED9EBA1;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    for I := 40 to 59 begin
        TEMP := S(5,A) + ((B ∧ C) ∨ (B ∧ D) ∨ (C ∧ D)) + E + W(I) + 8F1BBCDC;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    for I := 60 to 79 begin
        TEMP := S(5,A) + (B ⊕ C ⊕ D) + E + W(I) + CA62C1D6;
        E := D; D := C; C := S(30,B); B := A; A := TEMP
    end
    H0 := H0+A; H1 := H1+B; H2 := H2+C; H3 := H3+D; H4 := H4+E
end
```

Chaining Variables

Shift A left 5 bits

Attacks against SHA-1

- In early 2005, Rijmen and Oswald published an attack on a reduced version of SHA-1 (53 out of 80 rounds) which finds collisions with a complexity of fewer than 2^{80} operations.
- In February 2005, an attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu was announced. The attacks can find collisions in the full version of SHA-1, requiring fewer than 2^{69} operations (brute force would require 2^{80}.)
- In August 2005, same group lowered the threshold to 2^{63}.

- May lead to more attacks…

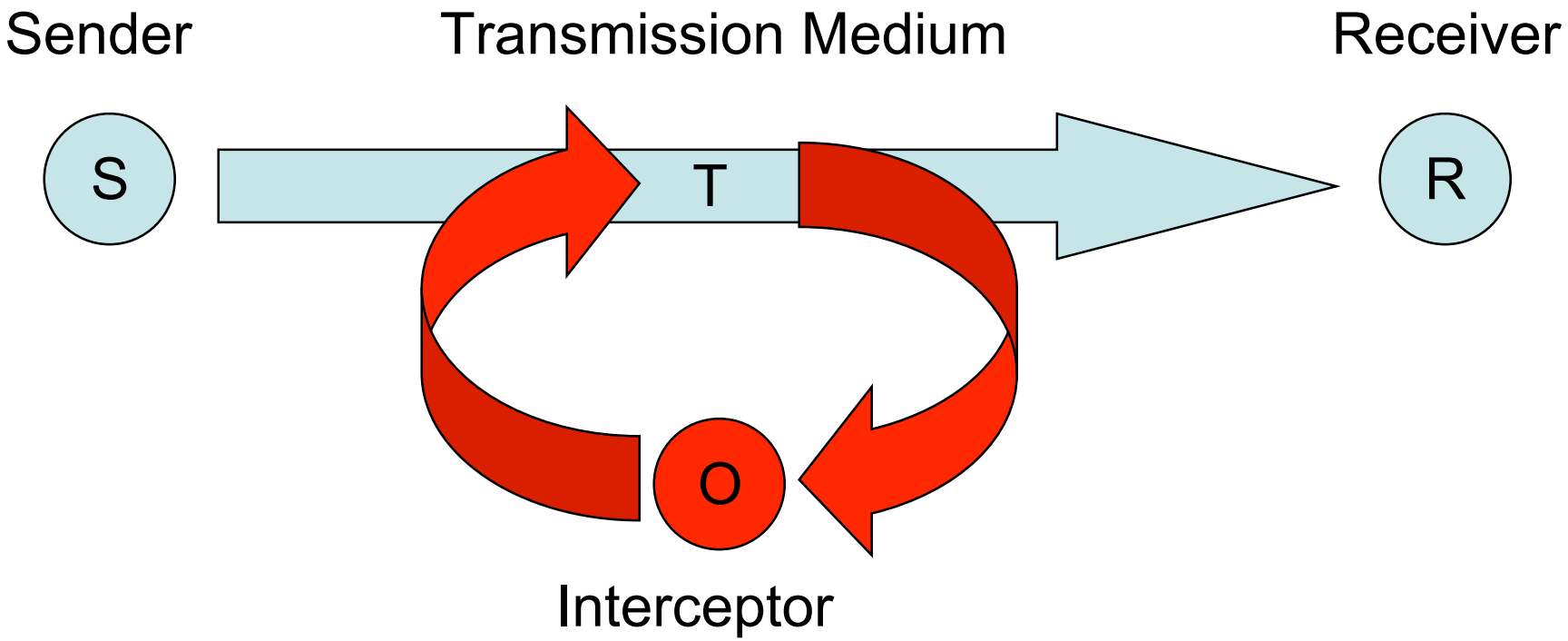
General Definition of “Protocol”

- 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)

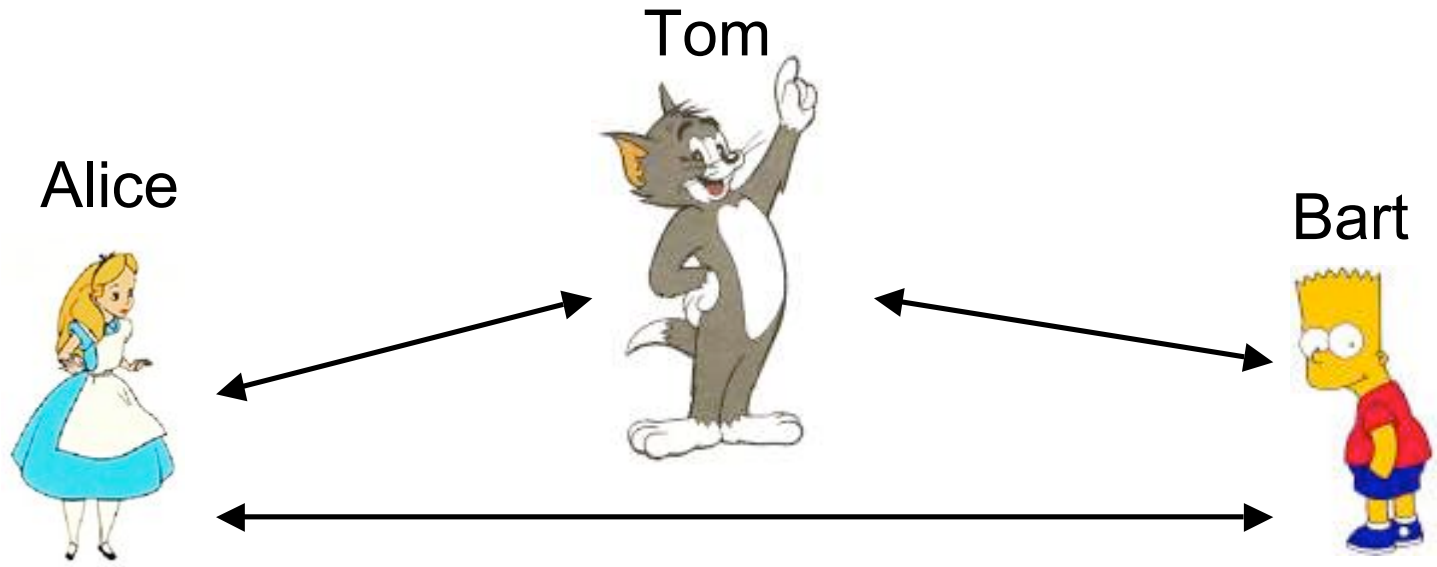
- Given a message M = (c1,c2,c3,…) attacker can deconstruct message into components c1 c2 c3
- Given a collection of components c1, c2, c3, attacker can forge message (c1,c2,c3)
- Given an encrypted object K{c}, attacker can learn c only if attacker knows decryption key corresponding to K
- Attacker 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

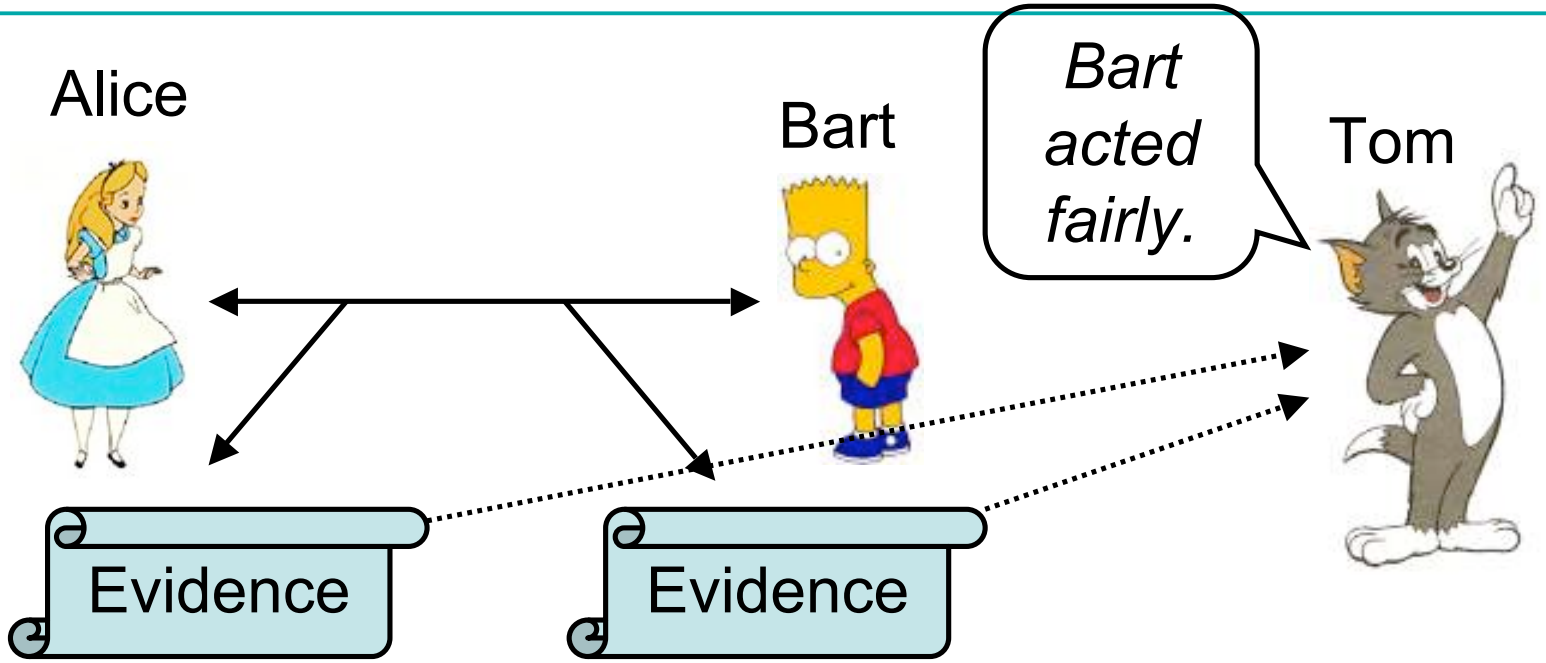


- Tom is an *arbiter*
  - Disinterested in the outcome (doesn't play favorites)
  - Trusted by the participants (Trusted 3rd party)
  - Protocol can't continue without T's participation

Arbitrated Protocols (Continued)

- Real-world examples:
  - Lawyers, Bankers, Notary Public
- Issues:
  - Finding a trusted 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



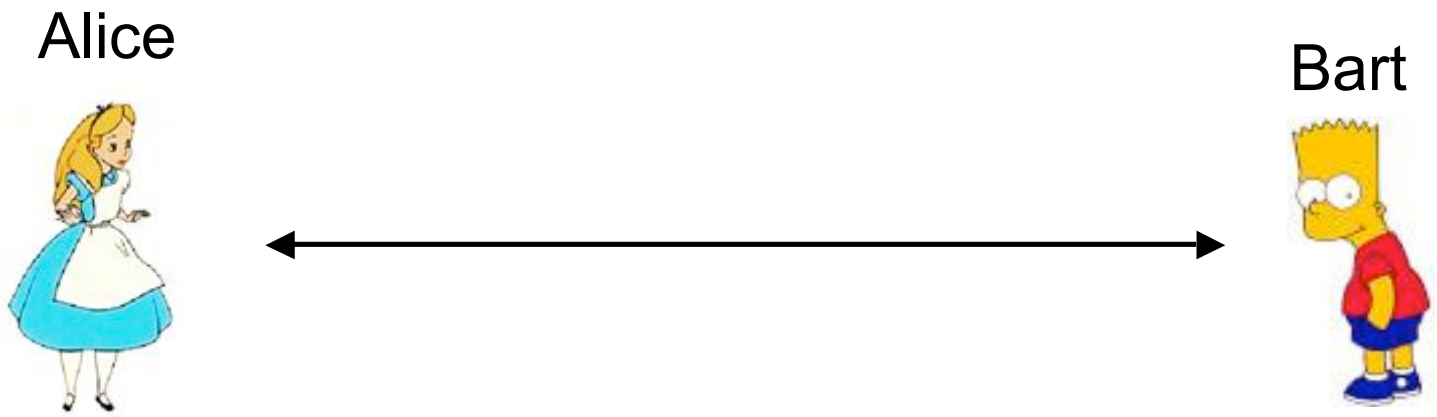
- Alice and Bard record an *audit log*
- Only in exceptional circumstances to they contact a trusted 3rd party. (3rd party is not always needed.)
- Tom as the *adjudicator* can inspect the evidence and determine whether the protocol was carried out fairly

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.

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

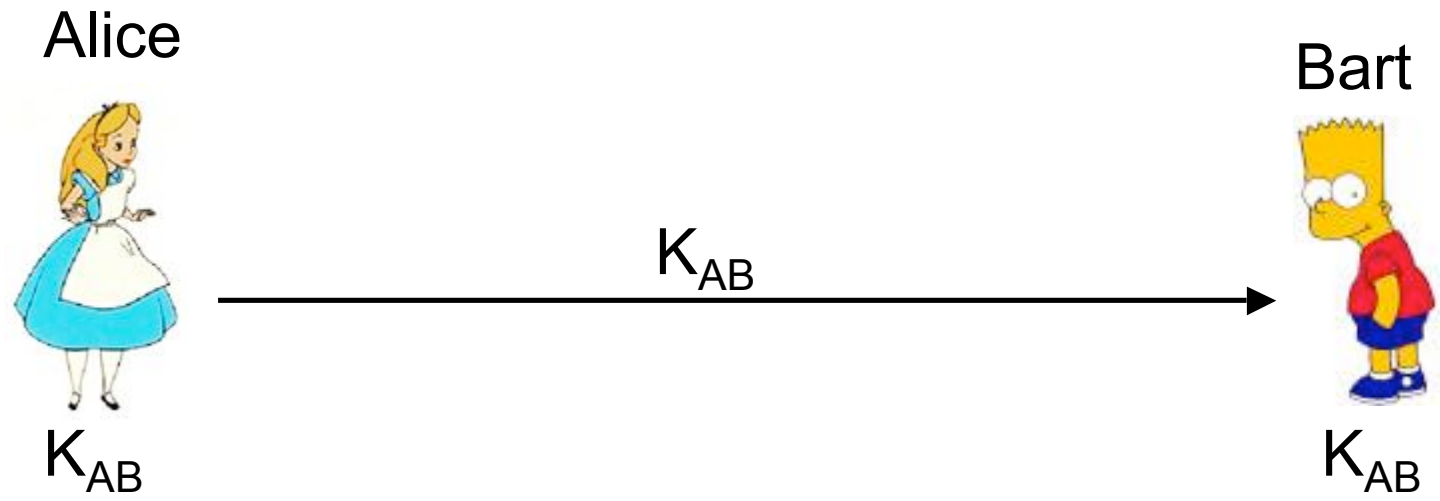
K_{AB}

Bart

K_{AB}

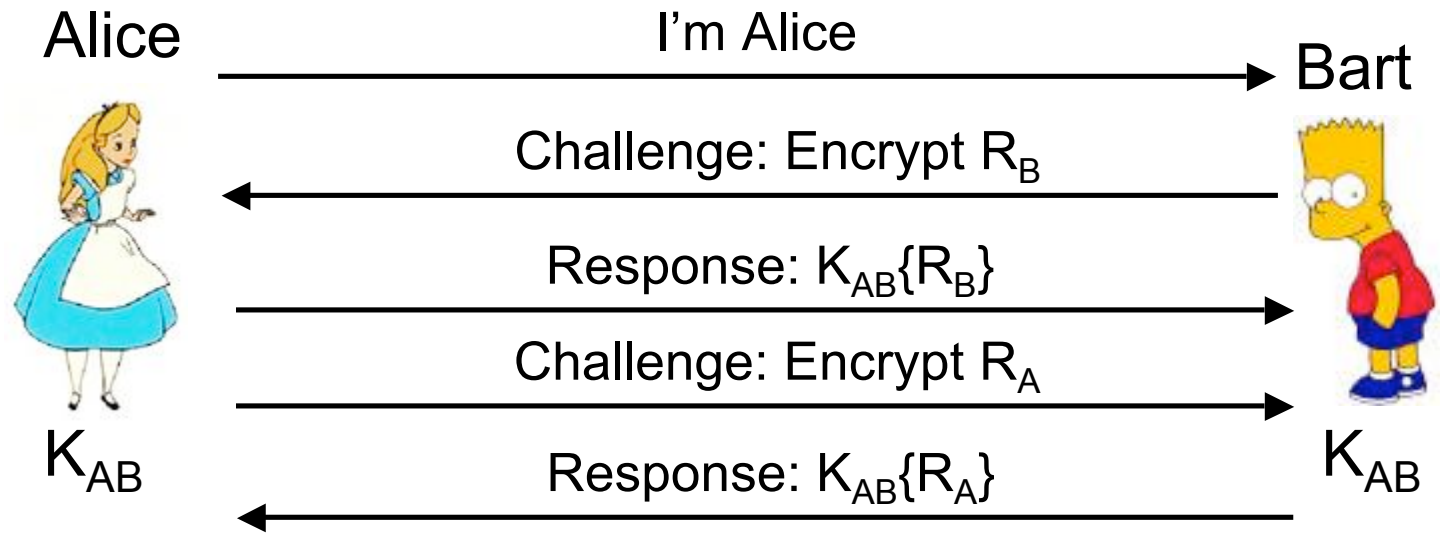
- Assume Alice & Bart already share a key K_{AB}.
  - The key might have been decided upon in person or obtained from a trusted 3rd party.
- Alice & Bart now want to communicate over a network, but first wish to authenticate to each other

Solution 1: Weak Authentication



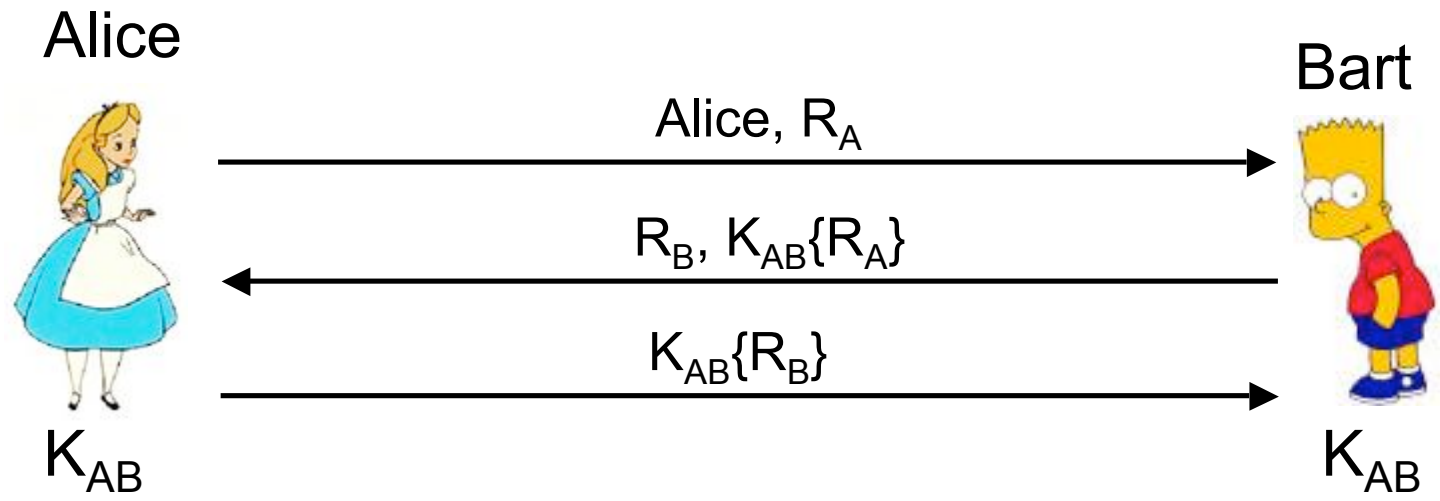
- Alice sends Bart $K_{AB.}$
  - K_{AB} acts as a password.
- The secret (key) is revealed to passive observers.
- Only works one-way.
  - Alice doesn't know she's talking to Bart.

Solution 2: Strong Authentication



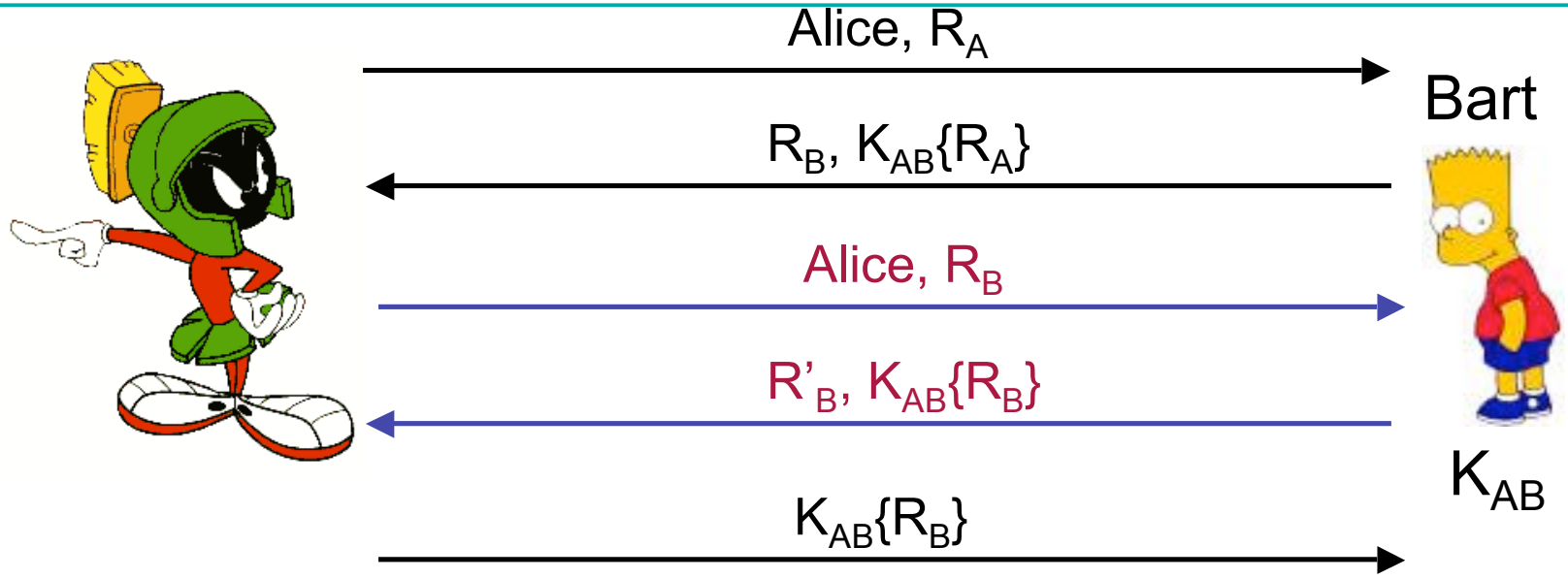
- Protocol doesn't reveal the secret.
- *Challenge/Response*
  - Bart requests proof that Alice knows the secret
  - Alice requires proof from Bart
  - R_A and R_B are randomly generated numbers

(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)