CIS 551 / TCOM 401

# Computer and Network Security

Spring 2006

Lecture 12

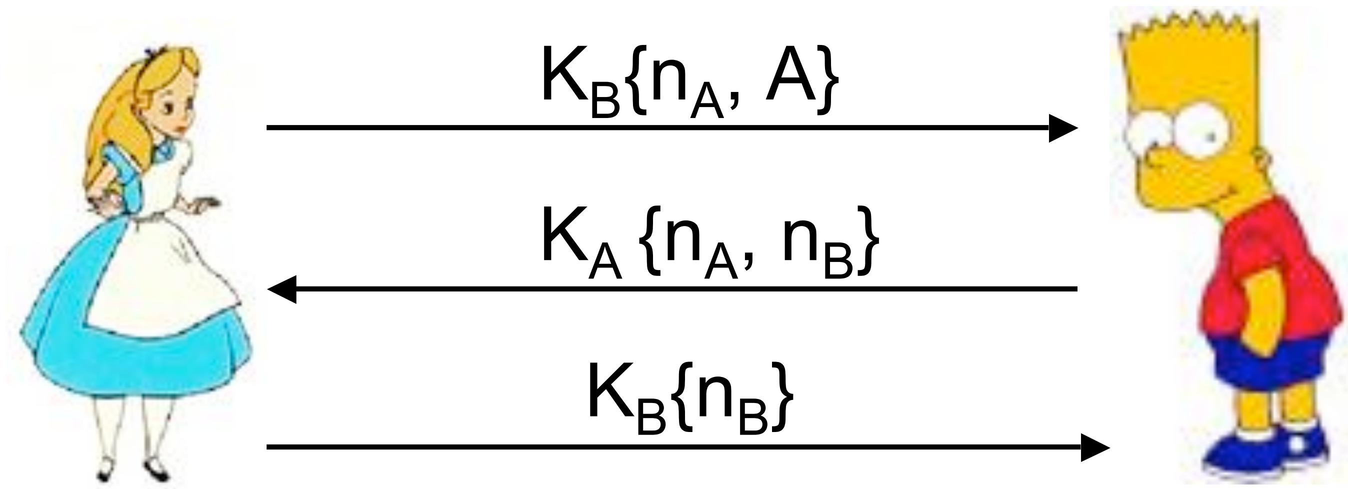
# Recap

- Last time:
  - Protocols in general
  - Authentication protocols with shared keys
  - Problem with interleaved protocol sessions

- Today:
  - Authentication protocol with public keys
  - Digital Signatures
  - Key distribution

# Mutual Authentication: Public Keys

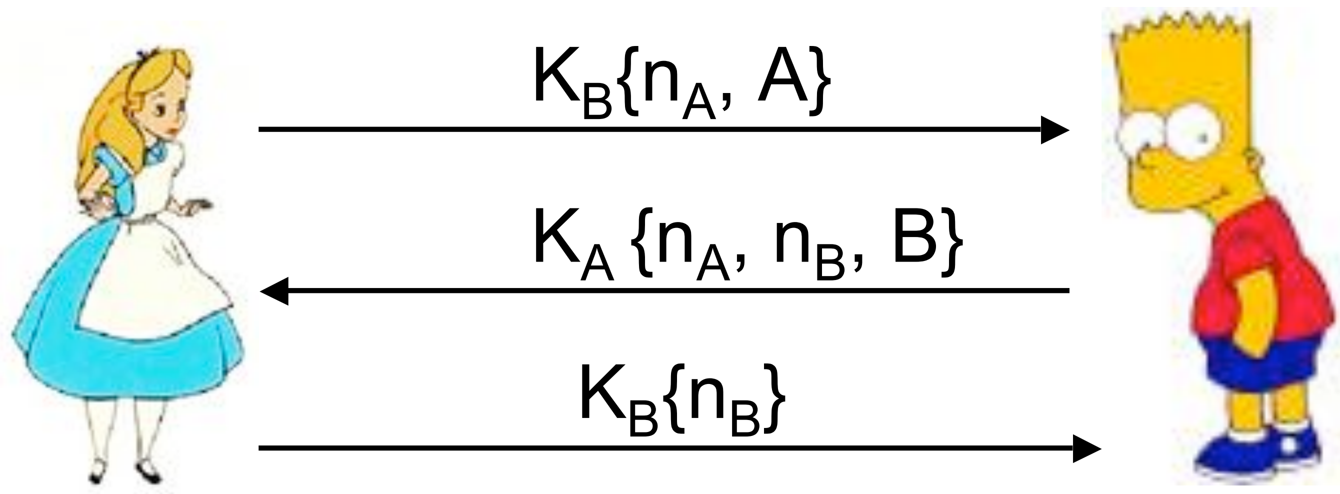
- Needham-Schroeder Public Key Authentication (1978)
- Consists of two stages:
  - 1st stage: use a trusted third party to exchange public keys.
  - 2nd stage: use the public keys to authenticate




- Flawed!

# Lowe's Fix

- Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR (1996!)



# 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
  - 3$^{rd}$ 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

Alice Tom Bart

$K_{AT}\{msg\}$ →

$K_{TB}\{Alice, msg, K_{AT}\{msg\}\}$ →

$K_{AT}$ $K_{AT}$ $K_{TB}$ $K_{TB}$

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 $(msg, k_A\{MD5(msg)\})$

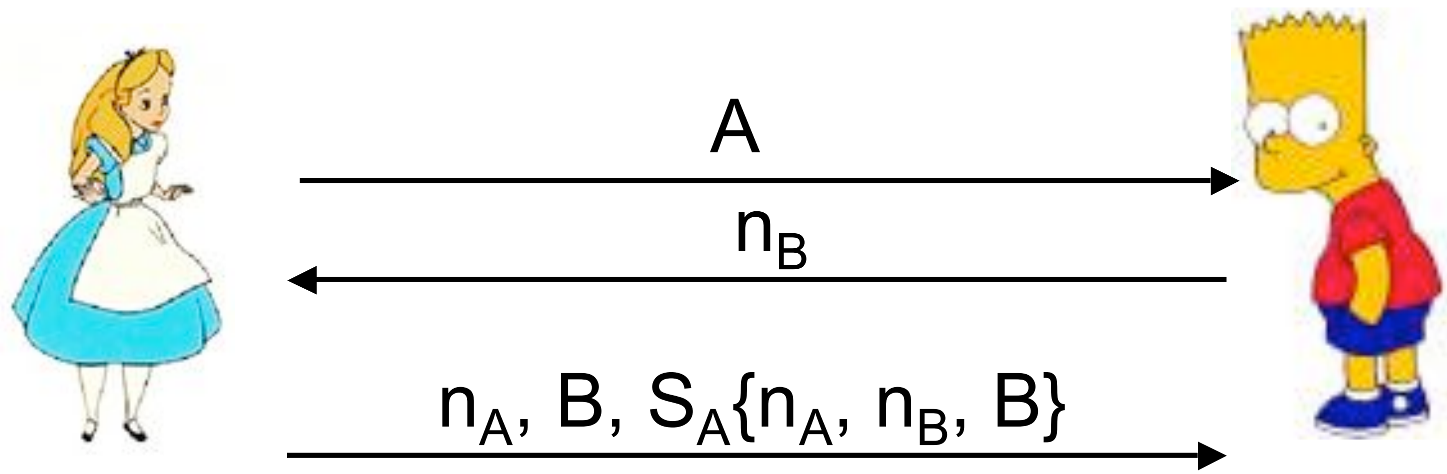
# Examples We've Seen

- Arbitrated Protocol
  - Shared key digital signature algorithm
  - Trusted 3rd party provided authenticity

- Adjudicated Protocol
  - Public key digital signature algorithm
  - Bart can keep Alice's digitally signed message
    - Trusted 3rd party provided non-repudiation

# Unilateral Authentication: Signatures

- $S_A\{M\}$ is A's signature on message M.
- Unilateral authentication with nonces:




The $n_A$ prevents chosen plaintext attacks.

# Primary Attacks

- Replay.
- Interleaving.
- Reflection.
- Forced delay.
- Chosen plaintext.

# Primary Controls

- Replay:
  - use of challenge-response techniques
  - embed target identity in response.
- Interleaving
  - link messages in a session with chained nonces.
- Reflection:
  - embed identifier of target party in challenge response
  - use asymmetric message formats
  - use asymmetric keys.

# Primary Controls, continued

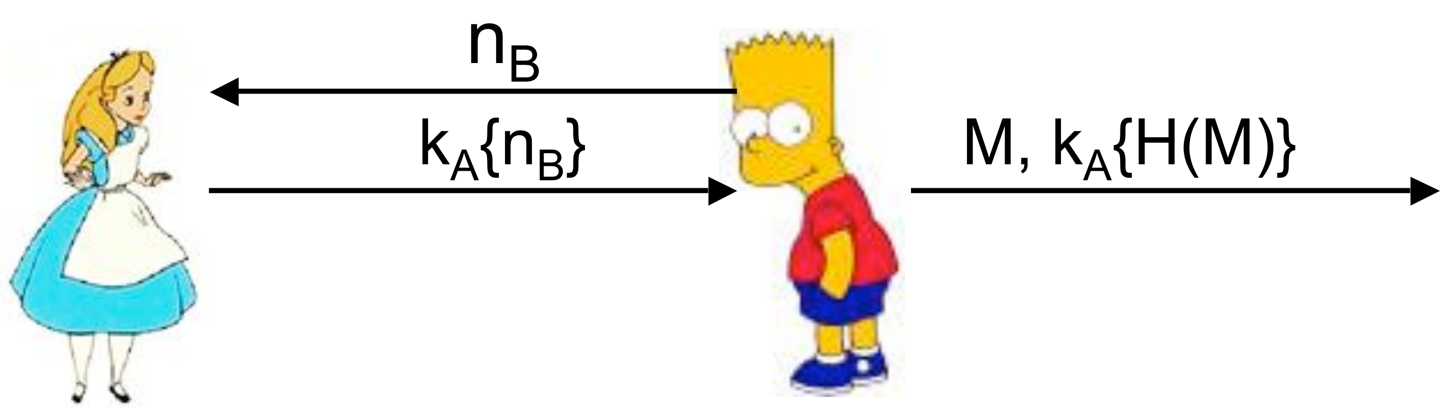
- Chosen text:
  - embed self-chosen random numbers (“confounders”) in responses
  - use “zero knowledge” techniques.
- Forced delays:
  - use nonces with short timeouts
  - use timestamps in addition to other techniques.

# 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

# Multiple Use of Keys

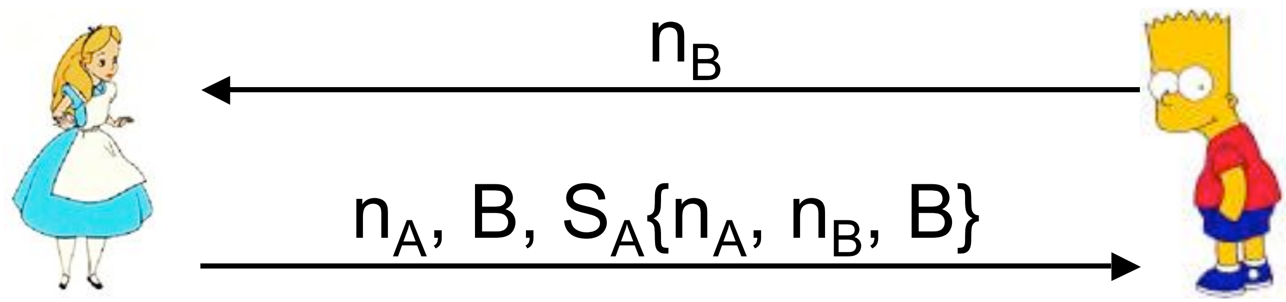
- Risky to use keys for multiple purposes.
- Using an RSA key for both authentication and signatures may allow a chosen-text attack.
- B attacker/verifier, $n_B$=H(M) for some message M.




B, pretending to be A

# Effective Control

- Notice how the protocol described earlier foils this. Here's the protocol:




- Here's what happens:
  - $B \rightarrow A: n_B$
  - $A \rightarrow B: n_A, B, k_A\{n_A, n_B, B\}$
  - $B(A) \rightarrow C: M, k_A\{n_A, H(M), B\}$
  - C finds that $k_A\{n_A, H(M), B\} \neq k_A\{H(M)\}$ and rejects the signature.

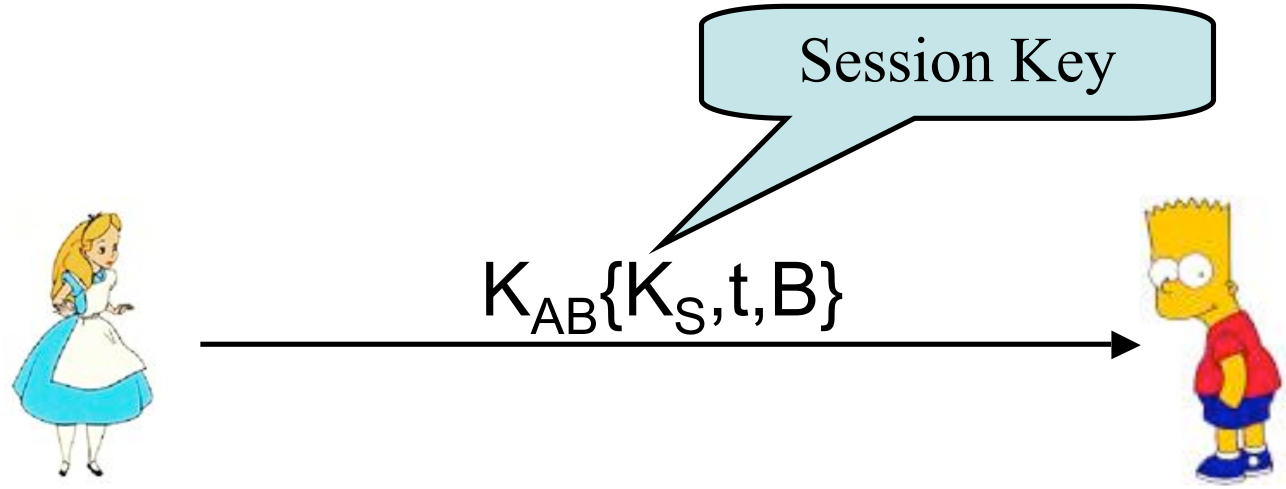
# Additional Controls

- Appropriate software engineering practices can rule out of these attacks.
- Many of the attacks contain "type confusion flaws"
  - A nonce is treated as a key (or vice versa)
- Actual implementations must "marshal" the values to be sent over the network
  - Marshal (or "Serialize"): convert to a sequence of bytes
  - Concretely in Java: Objects that implement "Serializable" interface can be safely written as a bytestream
  - The serialized version includes type information
- Therefore, appropriate use of type information (e.g. "Nonce" vs. "Key") can be used to prevent attacks.

# Key Establishment

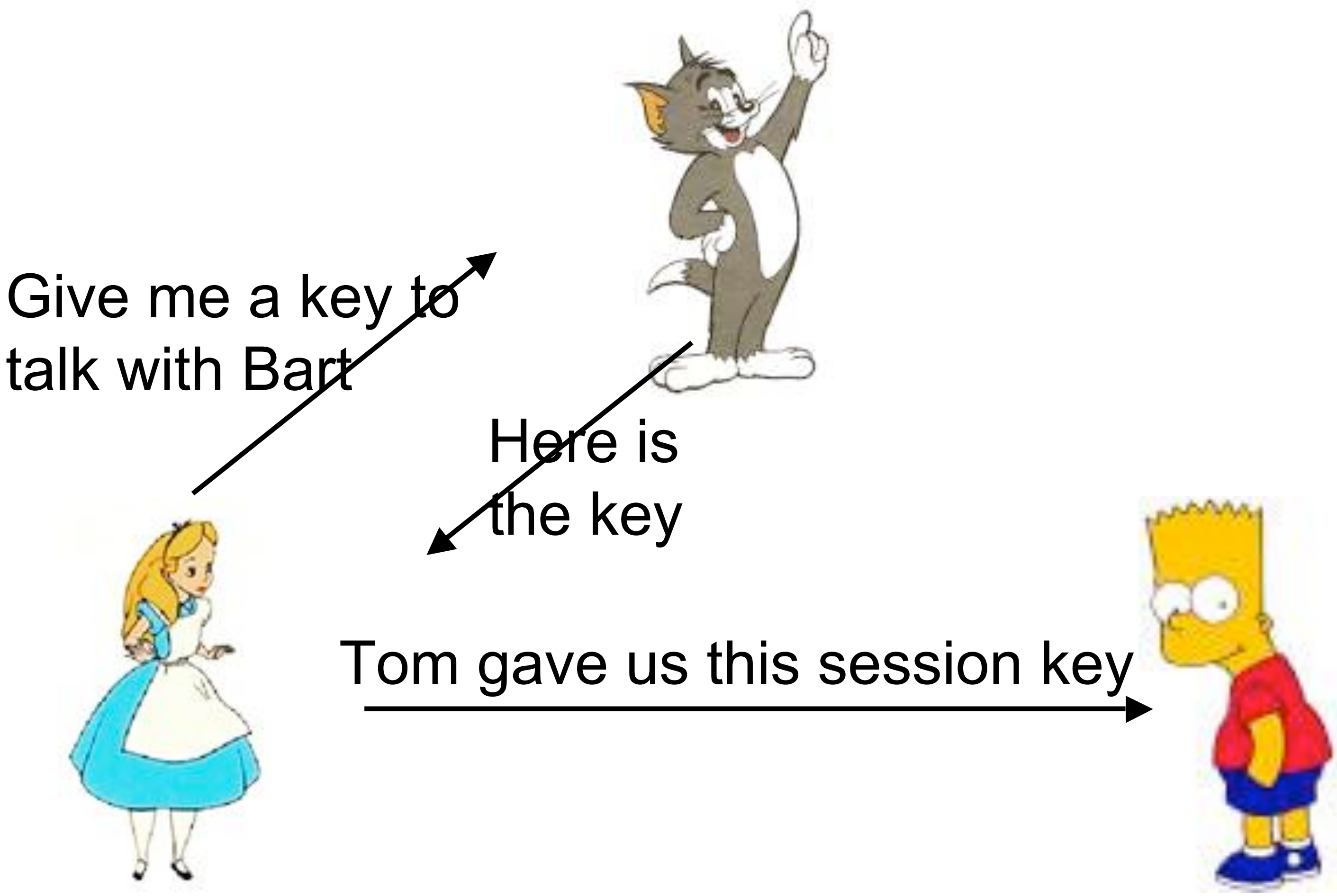
- Symmetric keys.
  - Point-to-Point.
  - Needham-Schroeder.
  - Kerberos.

# Point-to-Point




- Should also use timestamps & nonces.
- Session key should include a validity duration.

# 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 Key Distribution 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 : $K_{BT}\{K_S, C\}$

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.