
Protocol Attacks and Defences

27 April 2011
Lecture 8

Slide Credits: Steve Zdancewic (UPenn)

Topics for Today

- Primary Attacks and Controls
 - Usage Examples
- Other protocol classes

Primary Attacks

- Replay.
 - Reusing messages (or parts of messages) inappropriately
- Interleaving.
 - Mixing messages from different runs of the protocol.
- Reflection.
 - Sending a message intended for destination A to B instead.
- Chosen plaintext.
 - Choosing the data to be encrypted
- Forced delay.
 - Denial of service attack -- taking a long time to respond
 - Not captured by Dolev Yao model

Primary Controls

- Against Replay:
 - use of challenge-response techniques
 - embed target identity in response.
- Against Interleaving
 - link messages in a session with chained nonces.
- Against 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.

Replay

- *Replay*: the threat in which a transmission is observed by an eavesdropper who subsequently reuses it as part of a protocol, possibly to impersonate the original sender.
 - Example: Monitor the first part of a telnet session to obtain a sequence of transmissions sufficient to get a log-in.
- Three strategies for defeating replay attacks
 - Nonces
 - Timestamps
 - Sequence numbers.

Nonces: Random Numbers

- *Nonce*: A number chosen at random from a range of possible values.
 - Each generated nonce is valid only once.
- In a challenge-response protocol nonces are used as follows.
 - The verifier chooses a (new) random number and provides it to the claimant.
 - The claimant performs an operation on it showing knowledge of a secret.
 - This information is bound inseparably to the random number and returned to the verifier for examination.
 - A timeout period is used to ensure “freshness”.

Time Stamps

- The claimant sends a message with a timestamp.
- The verifier checks that it falls within an acceptance window of time.
- The last timestamp received is held, and identification requests with older timestamps are ignored.
- Good only if clock synchronization is close enough for acceptance window.

Sequence Numbers

- Sequence numbers provide a sequential or monotonic counter on messages.
- If a message is replayed and the original message was received, the replay will have an old or too-small sequence number and be discarded.
- Cannot detect forced delay.
- Difficult to maintain when there are system failures.

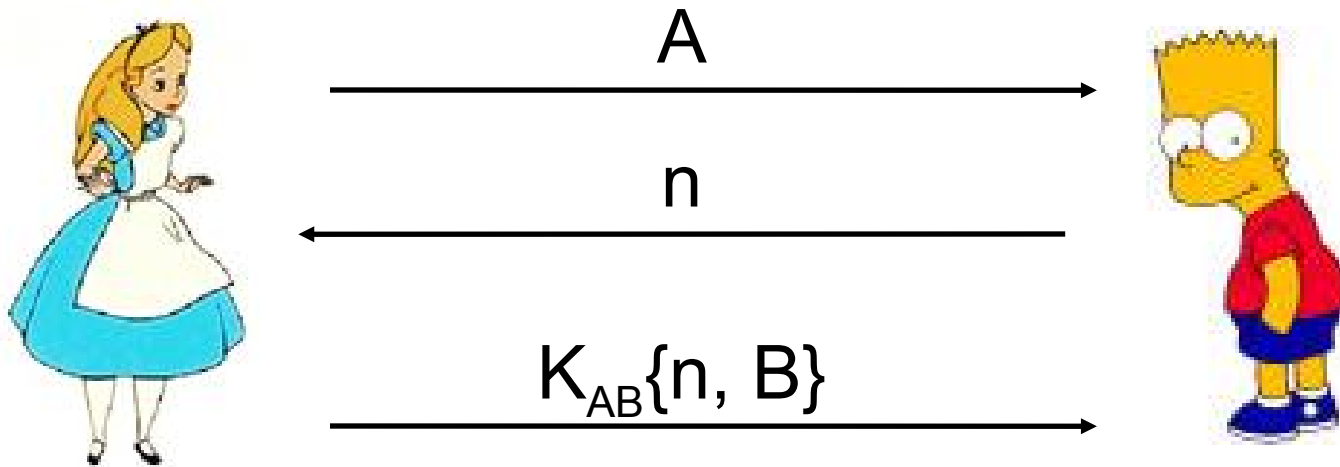
So Far

- Primary Attacks and Controls
 - Usage Examples
- Other protocol classes

How do we use all of these tools?

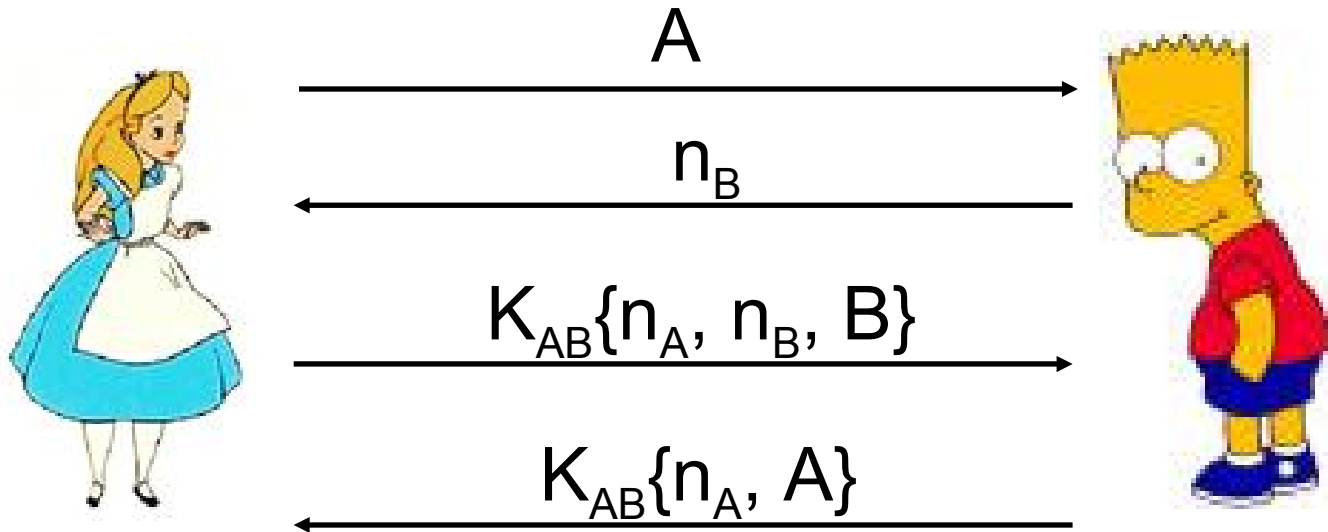
Unilateral Symmetric Key

- Unilateral = one way authentication
- Unilateral authentication with nonce.



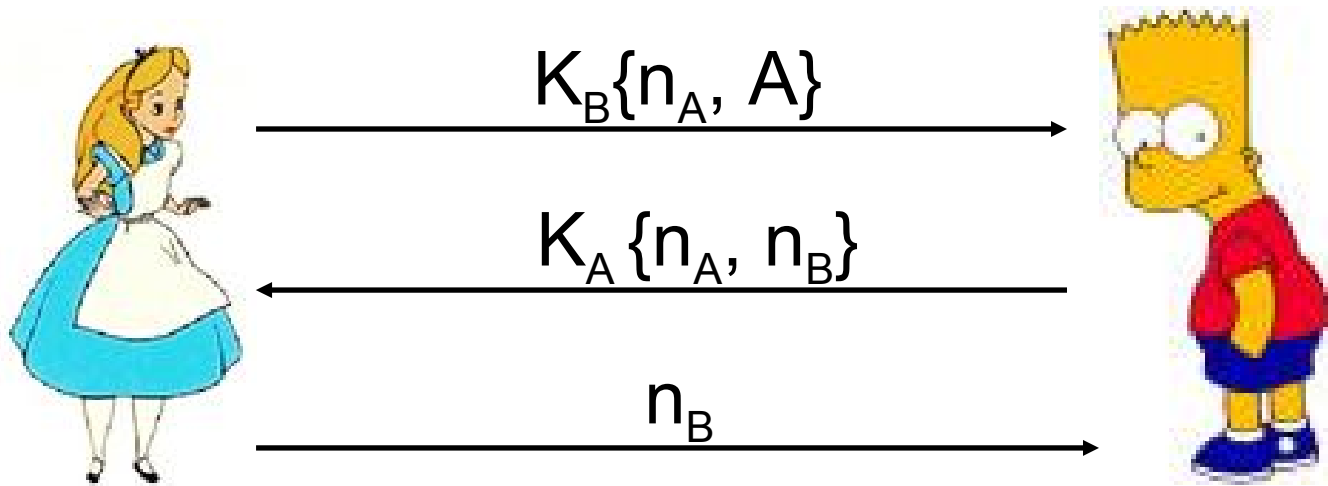
Mutual Symmetric Key

- Mutual = two way authentication
- Using Nonces:



Mutual Public Key Decryption

- Exchange nonces



Usurpation Attacks

- Identification protocols corroborate the identity of an entity only at a given instant in time.
 - An attacker could "hijack" a session after authentication.
- Techniques to assure ongoing authenticity:
 - Periodic re-identification.
 - Tying identification to an ongoing integrity service. For example: key establishment and encryption.

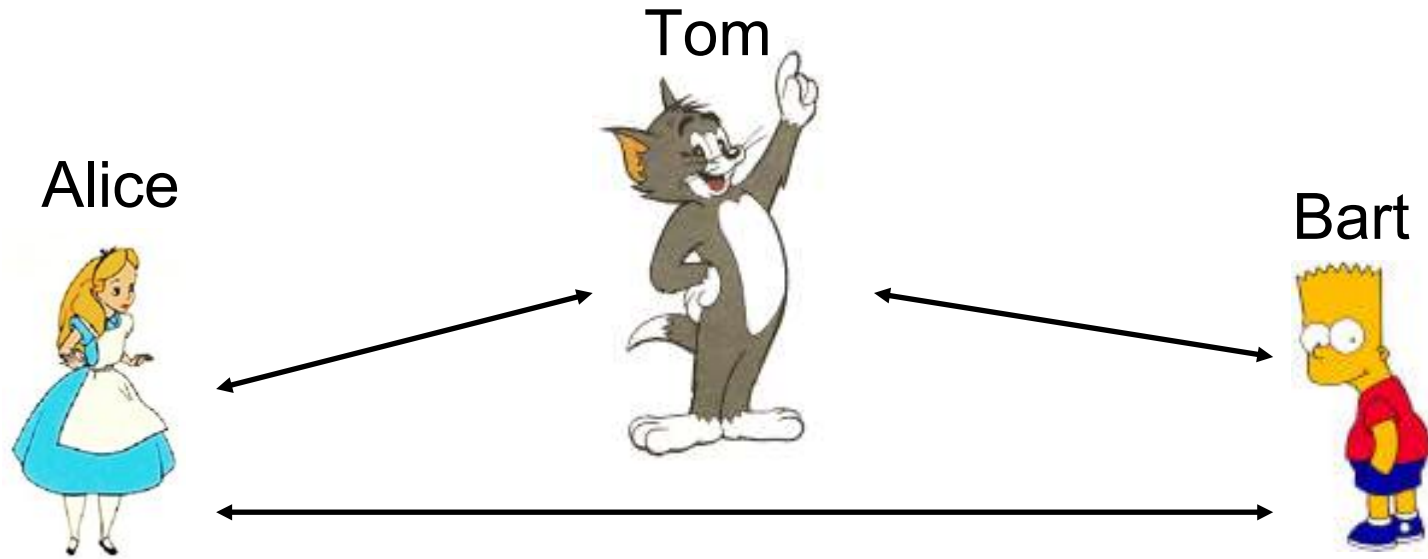
General Principles

- 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

So Far

- Primary Attacks and Controls
 - Usage Examples
- Other protocol classes

Arbitrated Protocols

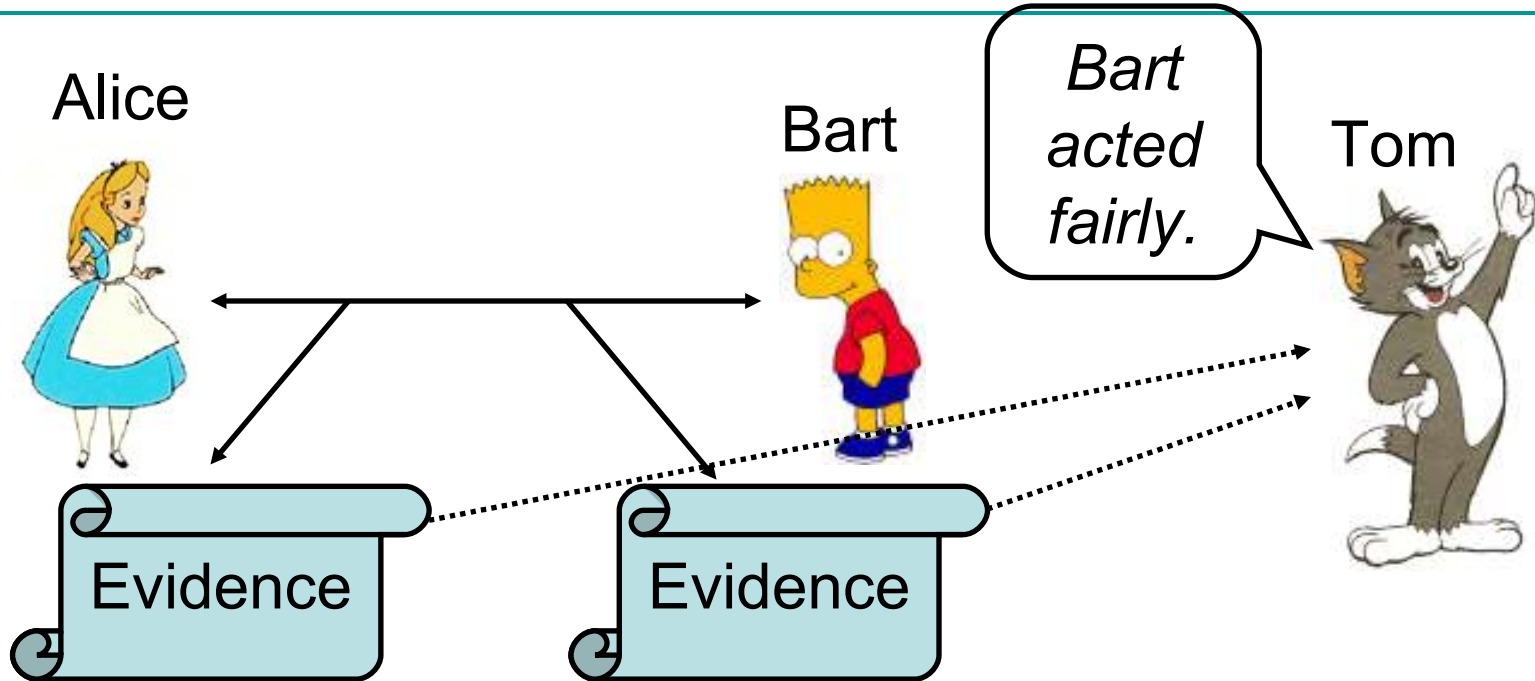


- Tom is an *arbiter*
 - Disinterested in the outcome (doesn't play favorites)
 - Trusted by the participants (Trusted 3rd party)
 - Protocol can't continue without T's participation

Arbitrated Protocols (Continued)

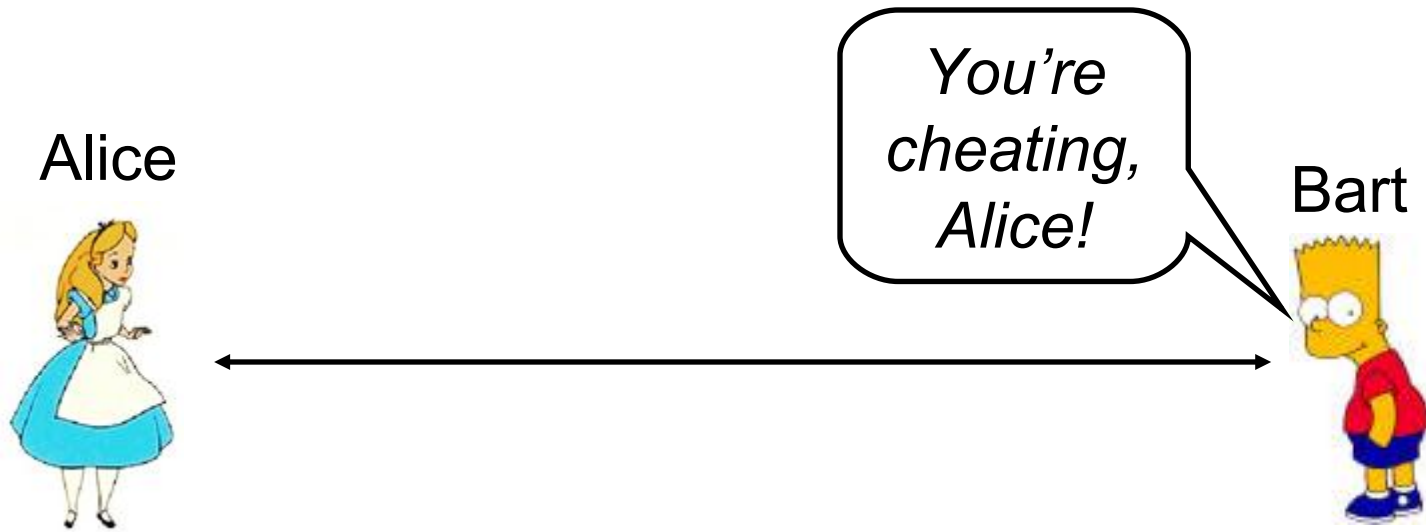
- Real-world examples:
 - Lawyers, Bankers, Notary Public
- Issues:
 - Finding a trusted 3rd party
 - Additional resources needed for the arbitrator
 - Delay (introduced by arbitration)
 - Arbitrator might become a bottleneck
 - Single point of vulnerability: attack the arbitrator!

Adjudicated Protocols



- Alice and Bart record an *audit log*
- Only in exceptional circumstances do 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.

Conclusion

- Primary Attacks and Controls
 - Usage Examples
- Other protocol classes