Security Protocol

2019. 4. 30

Terms

- Protocol
  - a sequence of steps(primitives) precisely specifying the actions required of two or more entities to achieve a specific security objectives
- Algorithms (primitives)
  - Specifying steps followed by a single entity(e.g., cryptographic algorithms, hash functions, digital signatures, and random number generation)
- Mechanism
  - A more general term encompassing protocols, algorithms, and non-cryptographic techniques(e.g., hardware protection and procedure controls) to achieve specific security objectives

Protocols

- Human protocols — the rules followed in human interactions
  - Example: asking a question in class, conversation over the phone
- Networking protocols — rules followed in networked communication systems
  - Examples: HTTP, FTP, etc.
- Security protocols — the (communication) rules followed in a security application
  - Examples: SSL, IPSec, Kerberos, etc.

Security protocol

- Protocol flaws can be very subtle, so not easy to detect
- Several well-known security protocols have serious flaws
  - Including IPSec, GSM and WEP
- Common to find implementation errors
  - Such as IE implementation of SSL
- Difficult to get protocols right

Protocol Design building blocks

- entities(parties, principals) involved
  - e.g., hosts, users, services, processes
- Secret information
- Authentication (authentic) information
  - e.g. passwords, public keys
- Basic cryptographic primitives
  - RSA, block cipher, stream cipher, hash function, MAC, Diffie-Hellman
- Trusted entities
- Proofs of freshness
  - nonces and timestamps

"Ideal" Protocol Wish List

- Satisfies security requirements
  - Requirements must be plain and precise
- Efficiency
  - Minimize computational overhead
  - Minimize delays/bandwidth overhead
- Easy to use and implement, flexible
- Robust
  - Must work when attacker tries to break it
  - Works even if environment changes

- As little trust as necessary
- As few assumptions as necessary
  - Idealized encryption???
  - Synchronized clocks?
  - Synchronized sequence numbers?
  - Randomly selected nonces and IVs?
  - Security of crypto primitives?
  - Authenticity or secrecy of keys?
- Very difficult to satisfy all of these

Possible Attacks

- Eavesdrop on communication
- replay messages at a later time
- Inject messages into the network
  - Modified (forged) from previous messages or new fabricated messages
- Alter or delete messages
- Hijack (take control of) a session or device
  - Initiate multiple parallel protocol sessions: MITM attack
  - Generate new messages on his own
- Run dictionary attack on passwords
- Run exhaustive attack on low-entropy nonce

Basic Security Protocols

- Entity(user) authentication protocols
  - Prove identity to each other
- Key establishment/agreement protocols
  - Establish a session key between two parties
  - Usually used to set up trusted communication channel providing secrecy and authenticity
- Other protocols
  - secure e-commerce, e-voting, time synchronization, etc.
- We use our basic cryptographic primitives as building blocks to design higher-level security measures

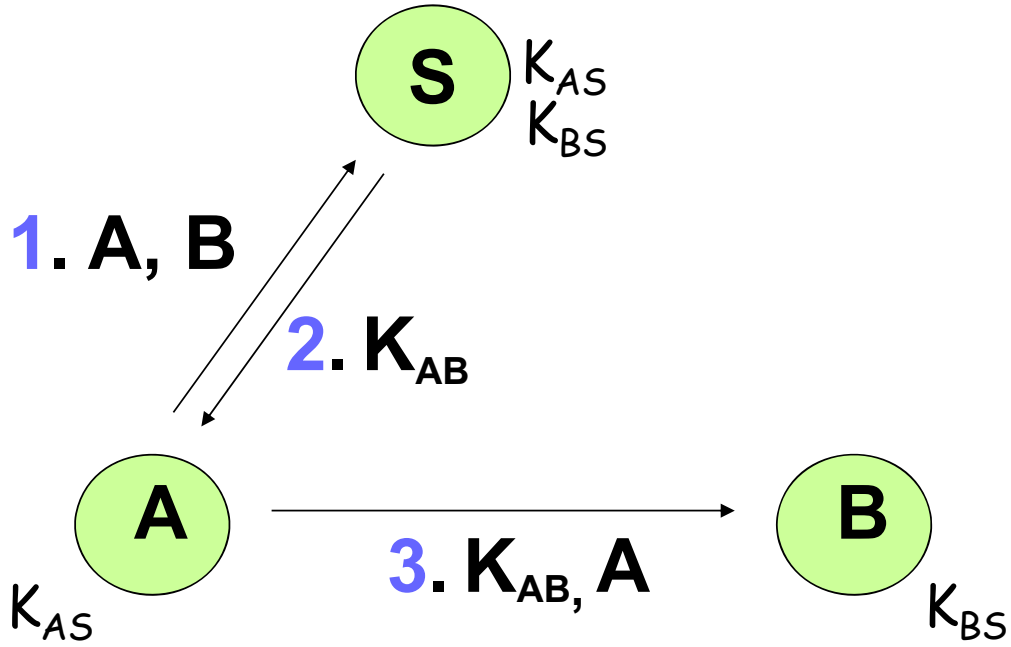
Key establishment protocol

- Protocol by which a shared secret key becomes available to two or more parties
- How the key is generated:
  - Key transport protocol
    - one of the parties generates a key.
  - Key agreement protocol
    - Key is a function of inputs by all parties.
    - ex, Diffie-Hellman
- How many users a protocol is designed to serve:
  - Normally two entities are users.
  - Mulicast users
  - Broadcast users
- Which keys are assumed to be established:

Design for secure protocols

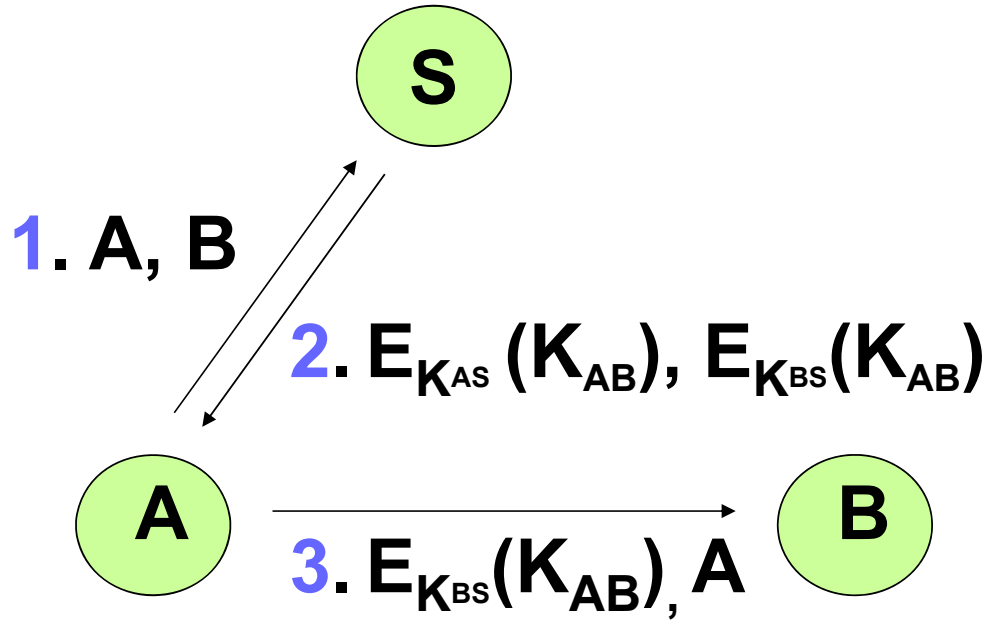
- Let's take an example of the key establishment protocol to illustrate the problems to design secure protocols.
  - Goal: A and B want to establish a key K_{AB}.
  - Assumption
    - A shares a key K_{AS} with a trusted server S.
    - B shares a key K_{BS} with a trusted server S.

Key establishment: 1st protocol



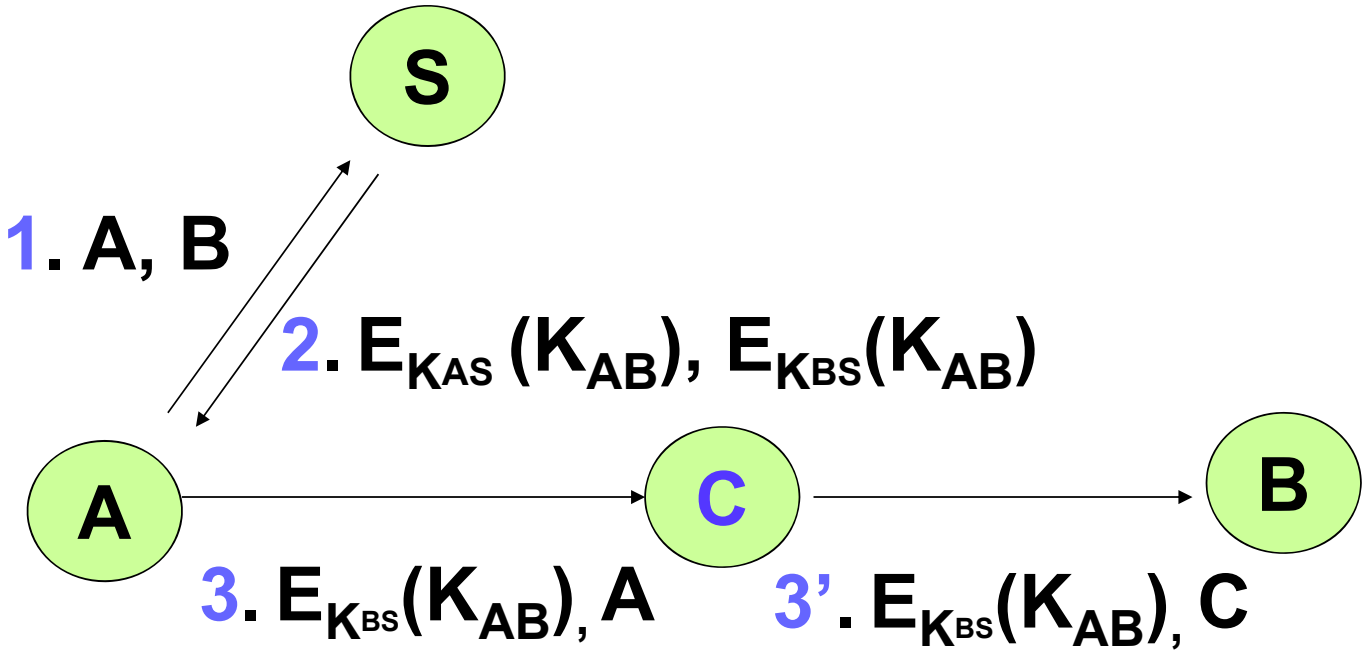
Problem? **The adversary can eavesdrop on all messages.**

Key establishment: 2nd protocol



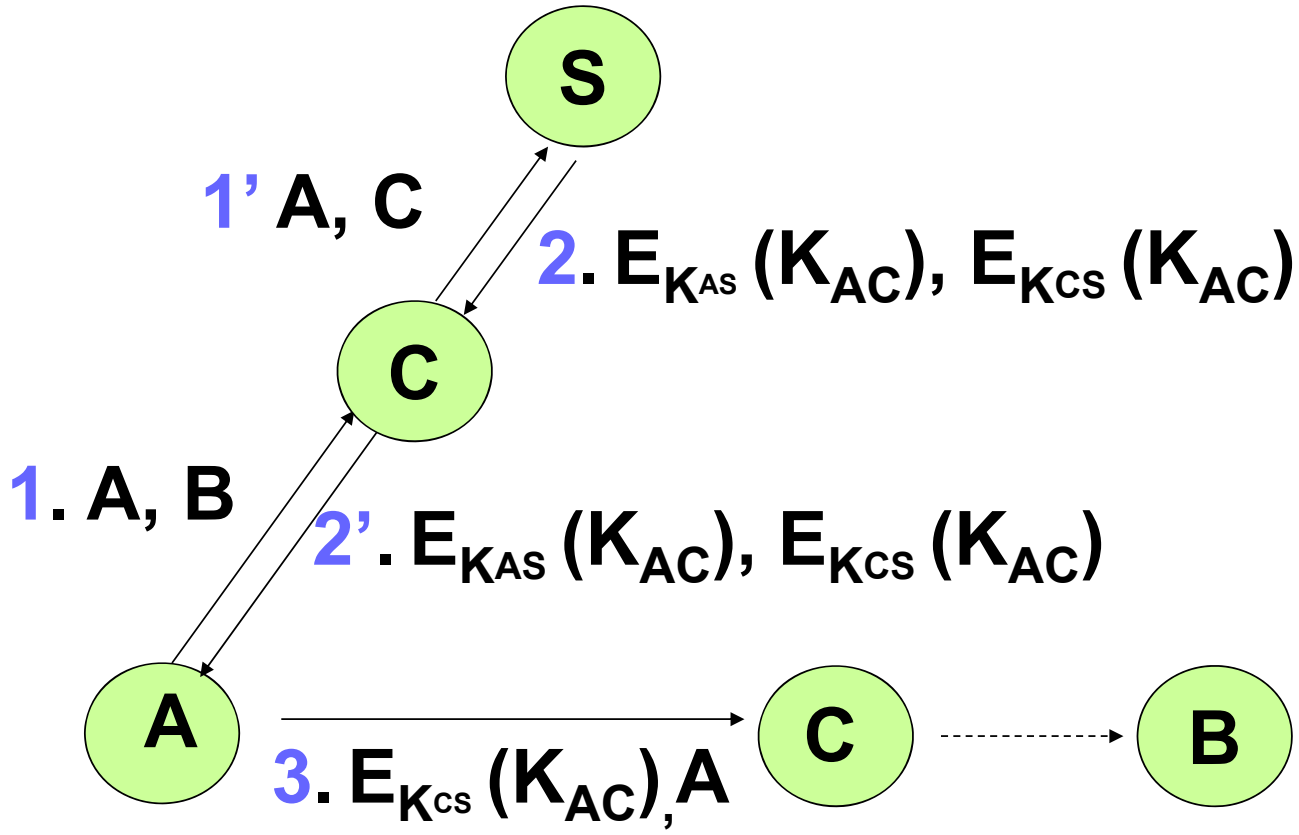
The adversary can alter all messages sent in a protocol using any information available.

Attack on 2nd protocol



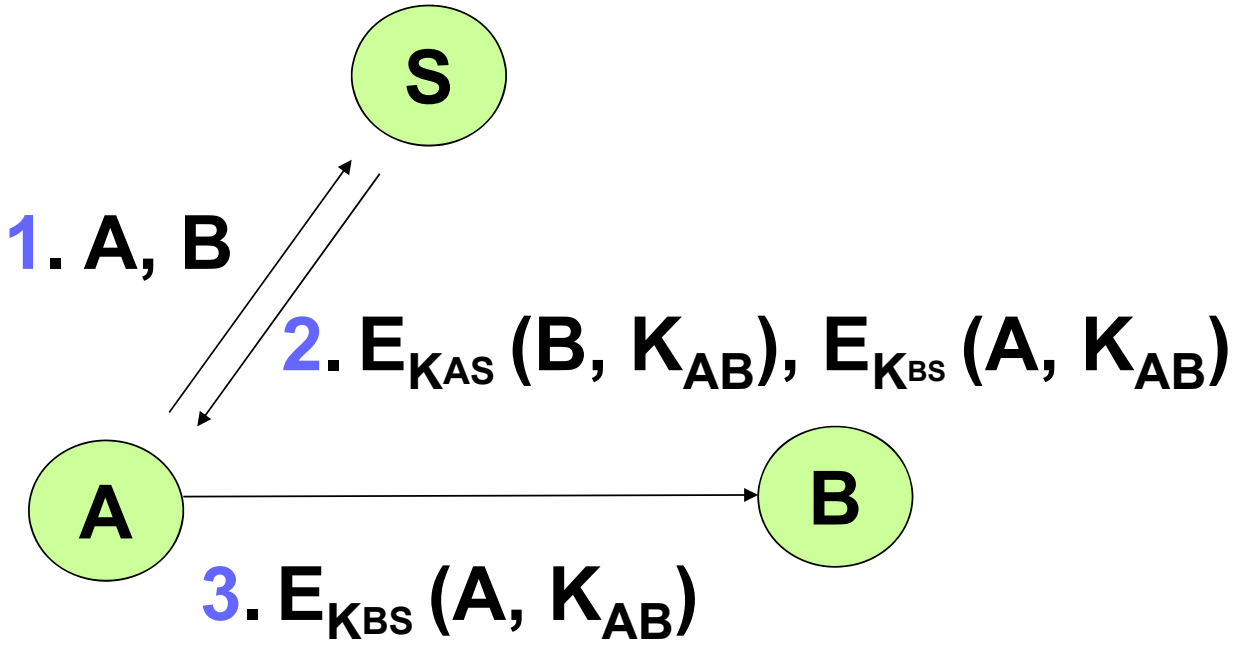
The adversary C intercepts the message and substitute C's ID for A's, so B believes that he is sharing the key with C.

Another attack on 2nd protocol

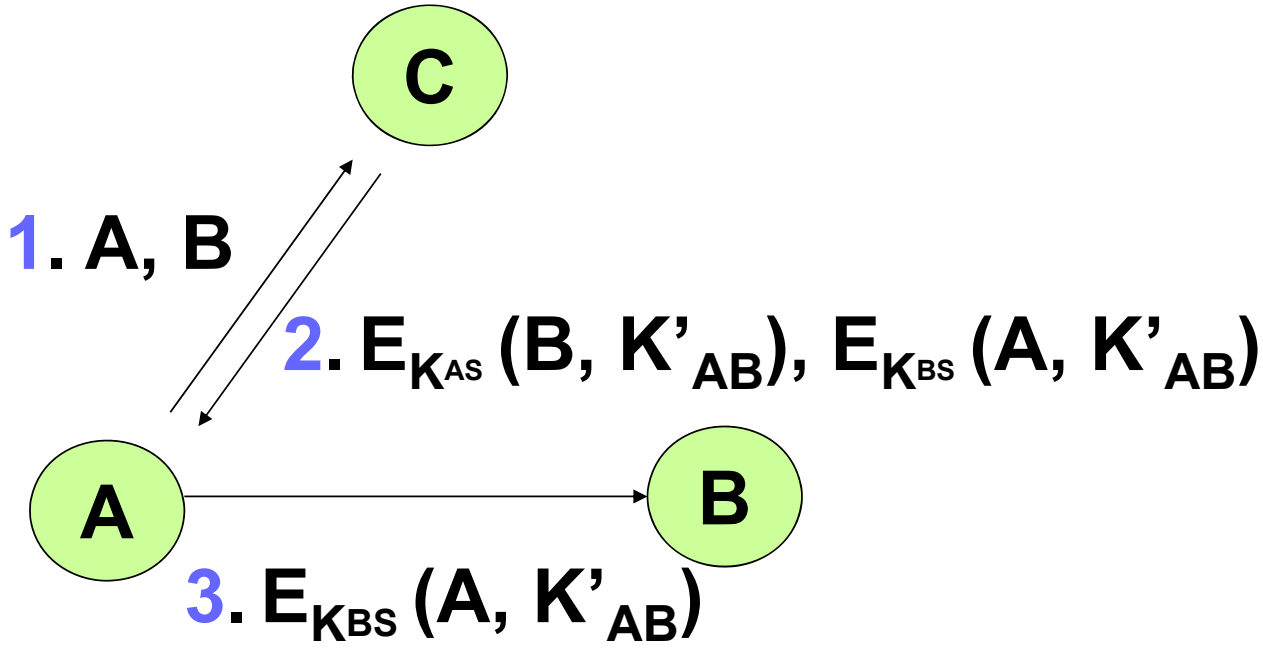


The adversary C impersonates B.

Key establishment: 3rd protocol

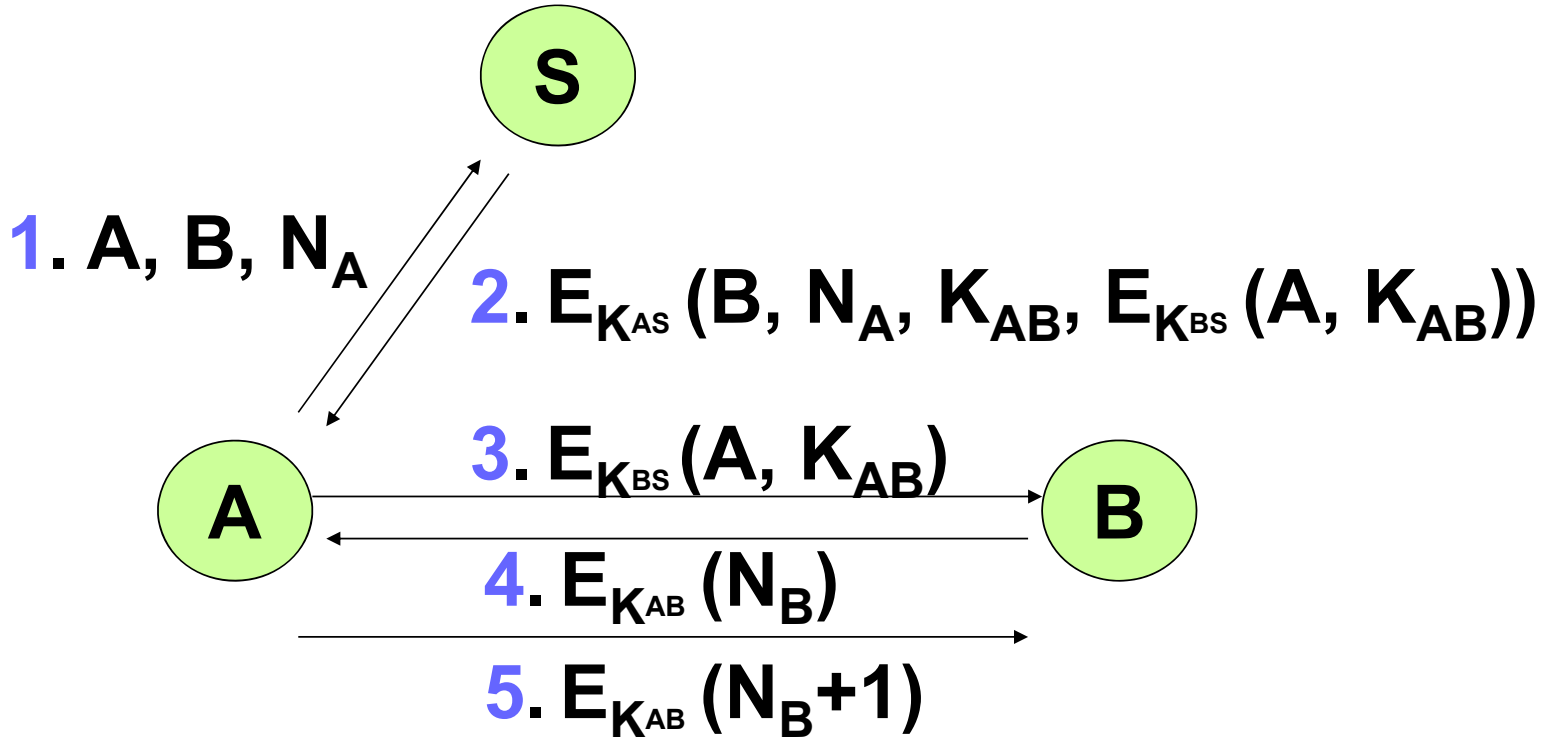


Attack on 3rd protocol

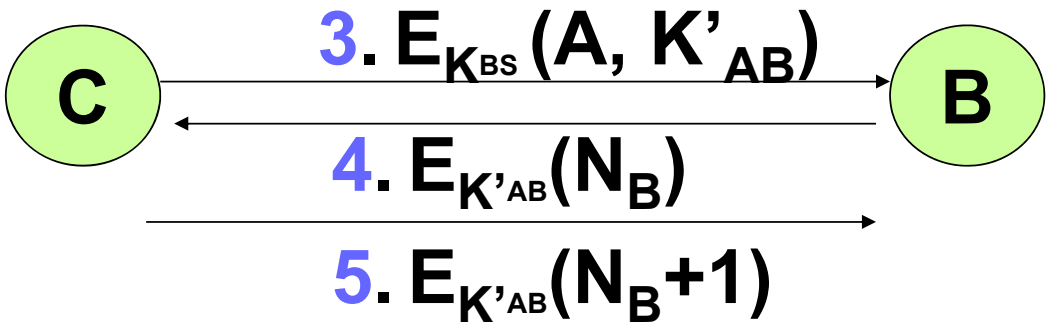


The key K'$_{AB}$ is an old key used by A and B.
The adversary replays the old key which he might know.

4th protocol (Needham-Shroeder)

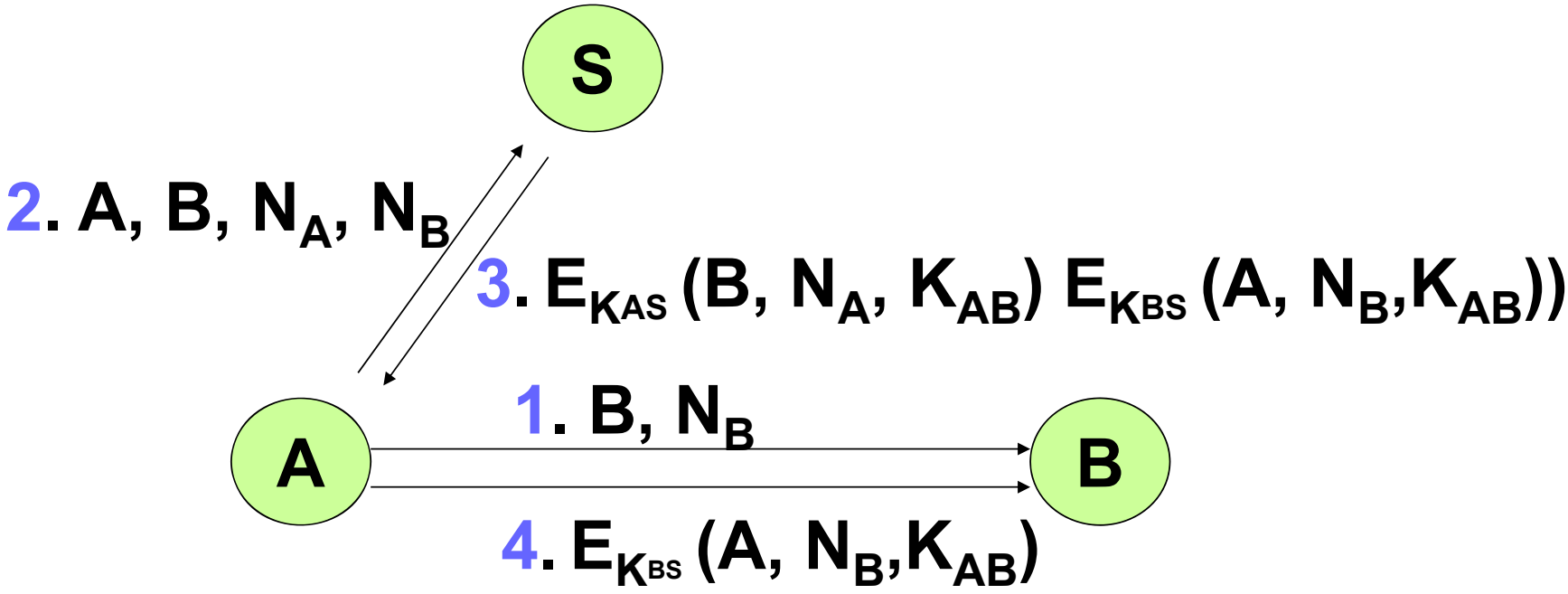


Attack on 4th protocol



Still, the adversary can know the old key.
C can masquerades as A.

Key establishment: 5th protocol



Design principles for secure protocol

- Guidelines for constructing secure protocols:
  - Abadi and Needham: “Prudent Engineering Practice for Cryptographic Protocols”

Principle 1: Explicit Communication

- Every message should say what it means. The interpretation of the message should depend only on its content.
- It should be possible to write down an English sentence describing the content.
- This principle reduces ambiguity
  - that messages are not used out of context.
  - prevent replay attacks, and intermixing of messages from concurrent sessions.

Example 1

- The core of the Denning-Sacco protocol is:

  $A \rightarrow B : E_{K_B+}(E_{K_A-}(K_{AB}, T))$

  A tells B that K_{AB} is a secure key for A and B at time T

- An attack goes:

  $A \rightarrow C : E_{K_C+}(E_{K_A-}(K_{AC}, T))$

  $C \rightarrow B : E_{K_B+}(E_{K_A-}(K_{AC}, T))$

  so C may have the key that B believes is shared with A

- Then: $C(A) \rightarrow B : \{data, T\}_{K_{AC}}$ and B would believe that A sent data
- Thus, C can impersonate A to B, B will believe it communicates with A, but it is communicating with C

What's wrong?

- Optimistic use of encryption
- Names are missing
  - It is not possible to interpret the message into the statement that represents its meaning
- Solution
  - A → B : $E_{K_B^+}(E_{K_A^-}(A, B, K_{AB}, T))$
  - or any other unambiguous encoding of the meaning of the message

Principle 2

- The conditions for a message to be acted upon should be clearly set out so that someone reviewing a design may see whether they are acceptable or not.
- That is, clearly state your assumptions!
  - Be clear on how encryption is used, and the meaning of encryption
  - Be clear on how the timeliness of messages is proved, and on the meaning of temporal information in messages

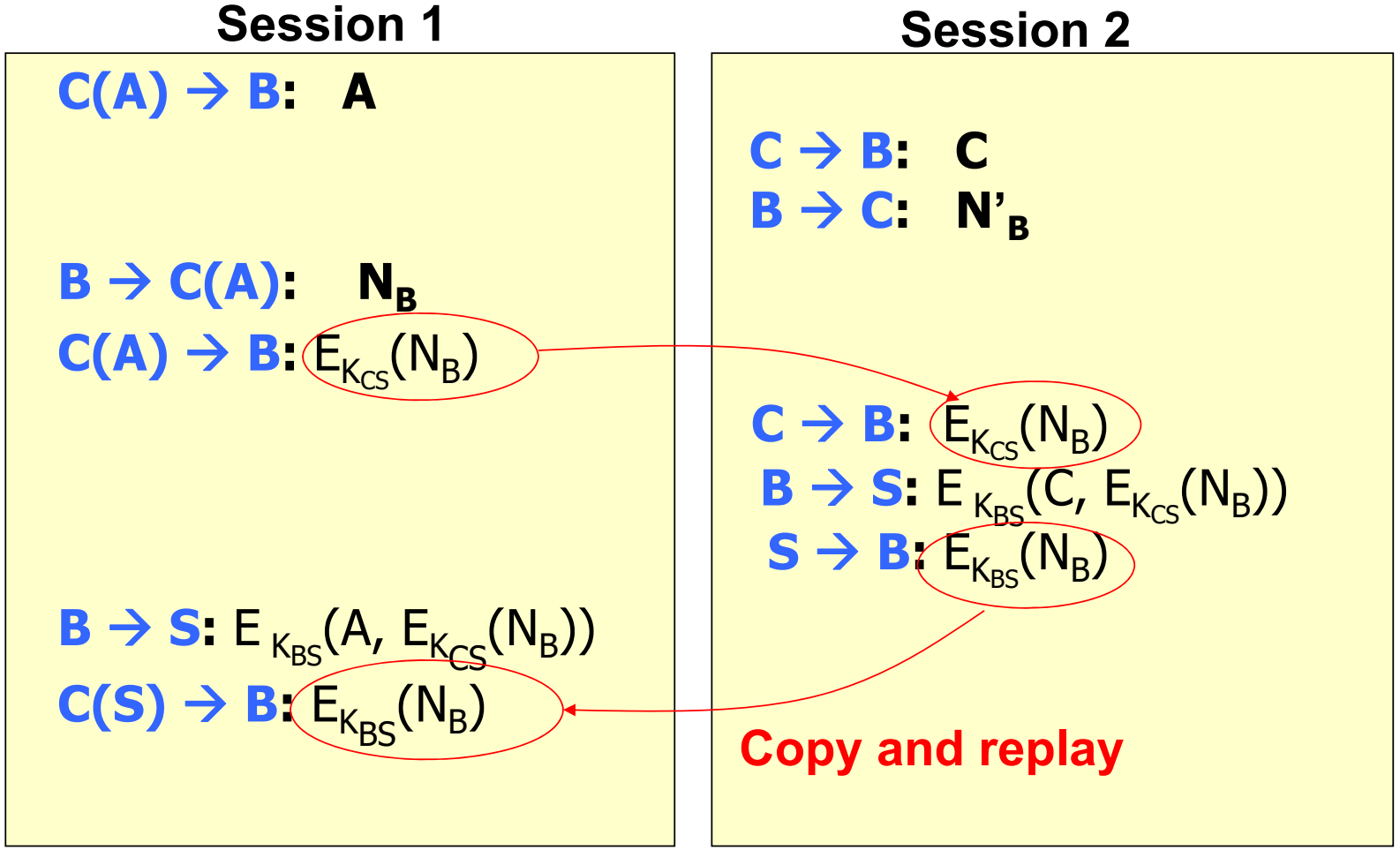
Principle 3: Naming

- If the identity of a principal is important for the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

The Woo Lam Protocol

- A → B: A
- B → A: N_B
- A → B: $E_{K_{AS}}(N_B)$
- B → S: $E_{K_{BS}}(A, E_{K_{AS}}(N_B))$
- S → B: $E_{K_{BS}}(N_B)$
- Goals
  - A wants to authenticate itself to B
  - B sends a random challenge N_B
  - A responds with N_B encrypted with key K_{AS} shared by A and the server S
  - B queries S

Attack: B believes that C is A.



What's wrong?

- It is not possible to parse S's message into the statement that represents its meaning: A's name is missing
- Nonces are good for ensuring freshness but not always for association
- Double encryption is no cause for optimism
- How would you fix this protocol?

Solution

- A → B: A
- B → A: N_B
- A → B: $E_{K_{AS}}(N_B)$
- B → S: A, $E_{K_{AS}}(N_B)$
- S → B: $E_{K_{BS}}$(A said N_B)
- S is explicit, expresses result
- Nonce is only used for freshness
- No double encryption

Principle 4

- Be clear as to why encryption is being done. Encryption is not synonymous with security, and its improper use can lead to errors.

Principle 5: signing encrypted data

- When signing already encrypted data, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for secrecy knows the content of the message.

Uses of Encryption

- Secrecy
- Authenticity: an entity proves ownership of a key by encrypting a known message with the key. Is there any problem?
- Example: entity authentication protocol
  - A → B: A, N_A
  - B → A: $\{ N_A , N_B , A \}_{K_{AB}}$
  - A → B: N_B

Principle 6: Timeliness

- Be clear as to what properties you assume of nonces. What may do for ensuring timeliness may not do for ensuring association; and perhaps association is best established by other means.
- For freshness and hence not a replay of an old one
  - Two types of Nonces (Number used only ONCE)
    - Counter: unique (non-repeating) but predictable, may use a time stamp for this purpose
    - Random number: unique and unpredictable
  - Timestamp

Principle 7

- The use of a predictable quantity (such as time or the value of a counter) can serve in guaranteeing freshness. But if a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay a response.

Predictability of Nonces

- Simple clock synchronization protocol:
  A → S: A, N_A
  S → A: $E_{K_{AS}}(T_S , N_A)$
- Is this protocol secure if N_A is a predictable nonce
- How to fix the protocol if N_A predictable?

- When NA is predictable, it should be protected:

  A → S: A, $E_{K_{AS}}(N_A)$

  S → A: $E_{K_{AS}}(T_S , E_{K_{AS}}(N_A))$

Principle 8

- If timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be much less than the allowable age of a message deemed to be valid. Furthermore, the time maintenance mechanism everywhere becomes part of the trusted computing base.

Principle 9: what is fresh

- A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised. Recent use does not make the key look any better than it would otherwise.

Principle 10

- If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

Principle 11

- The protocol designer should know which trust relations his protocol depends on, and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit though they will be founded on judgement and policy rather than on logic.