Introduction to Crypto Protocols : Features and Attacks

Source materials

1. Previous departmental course materials
2. Protocols for Authentication and Key Establishment (Boyd-Mathuria)
3. Handbook of Applied Cryptography (Menezes, Van Oorschot and Vanstone)
   Will be referred as HAC
4. Computer Security and the Internet: Tools and Jewels
   "PVO book"
5. Aalto University (Andrew Paverd and Tuomas Aura)
6. Academic papers and tech reports
7. Online resources

What's in this slide deck?

- Definition
- Designing a protocol by an example
- Types of attacks on a protocol
- Security properties of a protocol
- Types of crypto protocols & goals

Cryptographic protocols

- A protocol is a set of rules or conventions that govern the exchange of information between two or more principals (computers, hosts, humans)

- Cryptographic protocols are a subclass of protocols that use cryptographic techniques to achieve security objectives
  - Authentication, key establishment, anonymity….

Steps in cryptographic protocols

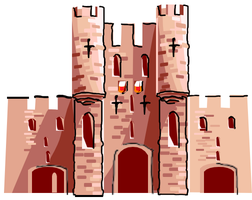
- Steps in a protocol:
  - A communication step transfers a message from one principal to another
    - Sender and receiver
    - Multiple parties might be involved
  - A computation step updates a principal's internal state
    - The computation results may be used only by the principal, or sent to other parties

Examples of deployed protocols

- Examples of deployed security protocols:
  - SSL (Secure Sockets Layer)
  - TLS (Transport Layer Security)
  - IPSec
  - PGP
  - Blockchain-based protocols
  - E-voting
  - Kerberos

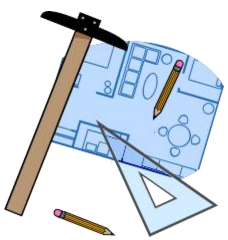
- Important assumptions:
  - The protocol used is NOT a secret
  - All specifications/parameters are publicly known

Relationship between system/protocol/ crypto primitives



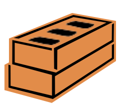
Systems

Firewalls, intrusion detection…



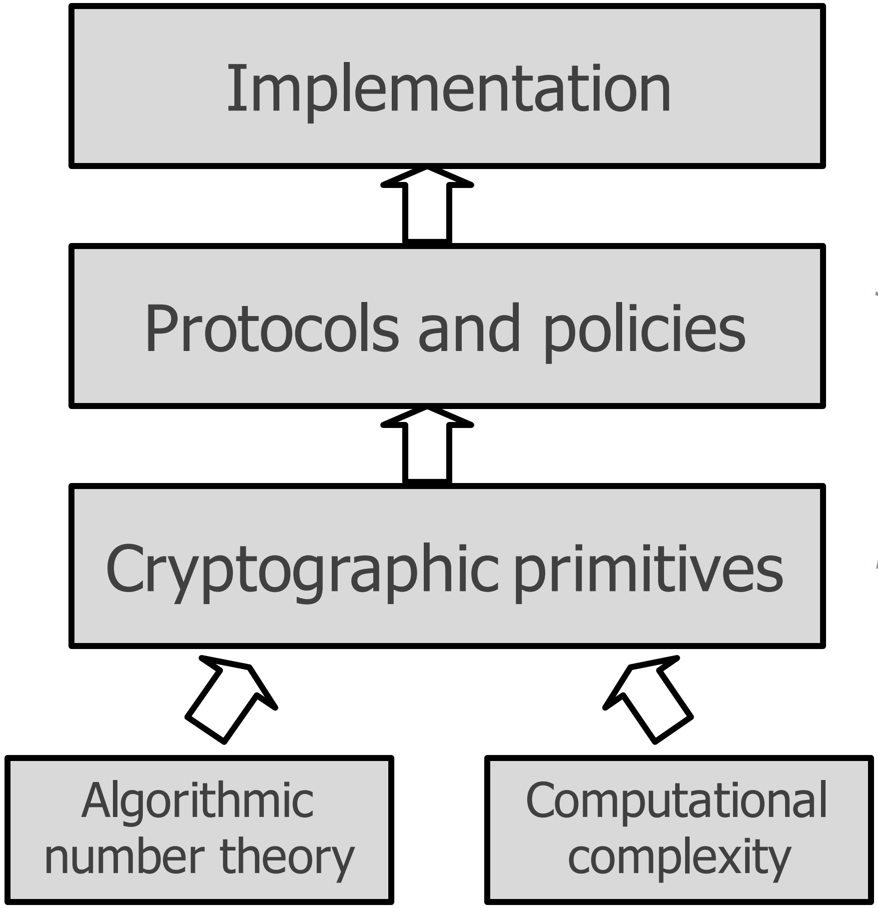
Security Protocols

SSL, IPSec, Kerberos, …



Building blocks

RSA, DES, SHA-1,…



[Shmatikhov]

Difficulties in security protocol design/analysis

- Properties/goals are sometimes more subtle than apparent
- Capturing assumptions / threat model is non-trivial
- Hostile operating environment
- Concurrent runs

Adversary attributes

1. *objectives*—assets requiring special protection
2. *methods*—anticipated attack techniques
3. *capabilities*—computing resources (CPU, storage, bandwidth), skills, knowledge, personnel, opportunity (e.g., physical access to target machines)
4. *funding level*—this influences attacker determination, methods and capabilities
5. *outsider vs. insider*—an attack launched without any prior special access to the target network is an *outsider attack*. In contrast, *insiders* and *insider attacks* originate from parties having some starting advantage, e.g., employees with physical access or network credentials as legitimate users.

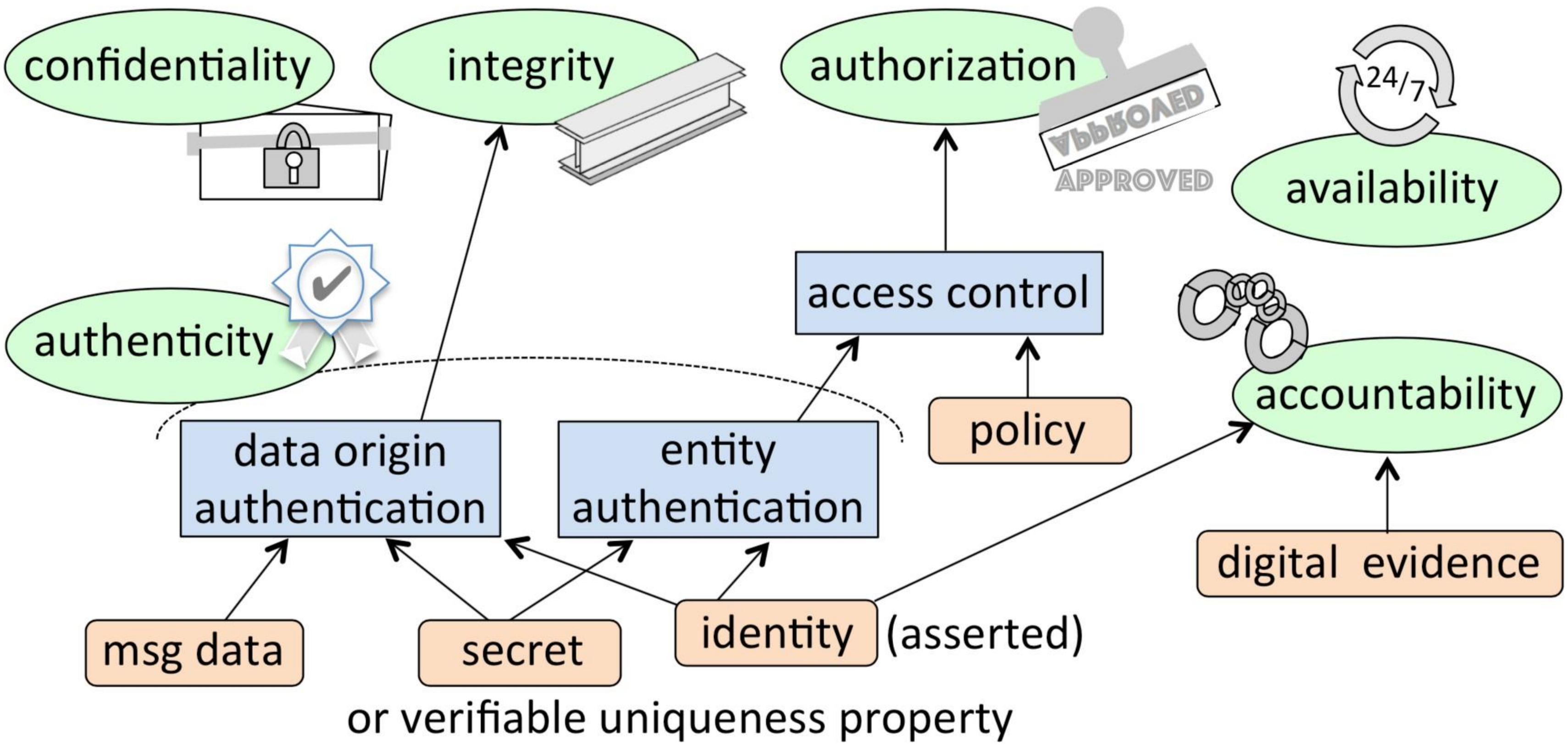
Adversary groups

	Named Groups of Adversaries
1	foreign intelligence (including government-funded agencies)
2	cyber-terrorists or politically-motivated adversaries
3	industrial espionage agents (perhaps funded by competitors)
4	organized crime (groups)
5	lesser criminals and *crackers* (i.e., individuals who break into computers)
6	malicious *insiders* (including disgruntled employees)
7	non-malicious employees (often security-unaware)

Adversary goals

- Learn confidential information
  - E.g. intercept passwords during login
- Modify data in transit
  - E.g. change recipient account in bank transfer
- Impersonate other participants
  - E.g. send a phishing email
- Block certain communication
  - E.g. prevent a Bitcoin transaction

Security goals (from PVO)



Adversary vs. security goals

- Learn confidential information
  - – Threat to data confidentiality
- Modify data in transit
  - – Threat to data integrity
- Impersonate other participants
  - – Threat to data origin/entity authentication
- Block certain communication
  - – Threat to data/service availability

Designing a security protocol

How to design security protocols

- Let's learn by an example

- Let's build a **key establishment protocol**
  - We want to exchange a session key (K_{AB}) between two users: Alice & Bob
  - Parties in a protocol are also termed as "Principals"
  - Assume we have three parties: Alice (A), Bob (B) and a trusted server (S)
  - S will generate K_{AB} and distribute it to A & B

Protocol goals

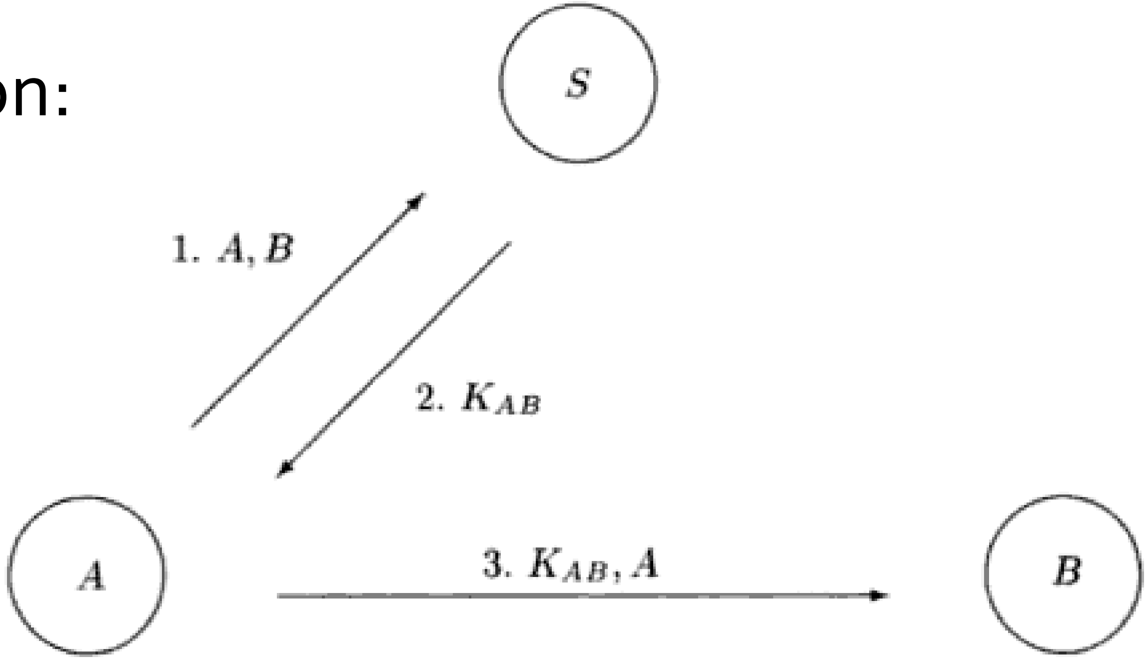
At the end of the protocol run, we should achieve:

1. G1: A & B must know K_{AB}
2. G2: K_{AB} must not be known to anyone else except (A, B, S)
3. G3: A & B must know that K_{AB} is a fresh key

   Newly generated for the current session only

Protocol 1.1

- Compact form:
  1. A $\rightarrow$ S: A, B
  2. S $\rightarrow$ A: K_{AB}
  3. A $\rightarrow$ B: K_{AB}
- As an illustration:



Goals & notation

- Question:
  - What goals did we achieve with Protocol 1.1?
  - G1, G2, G3?
- Notation used:

$E_A(M)$	Public key encryption of message M, with entity A's public key
$\{M\}_K$	Symmetric key encryption with shared key K
$MAC_K(M)$	MAC of M under shared key K
$Sig_A(M)$	Digital signature of M, generated by A

Threat model

- One important fact we did not set out before we initiate the design

- Assumptions about the operating environment
  - i.e.: "Rules of the game"
  - What powers the attacker has?
  - What are the limitations of the attacker?
  - What resources you need for a safe run?

Assumption - Confidentiality

- Attackers have access to all protocol messages exchanged in the protocol, in all protocol runs

- Now, let's check the goals again

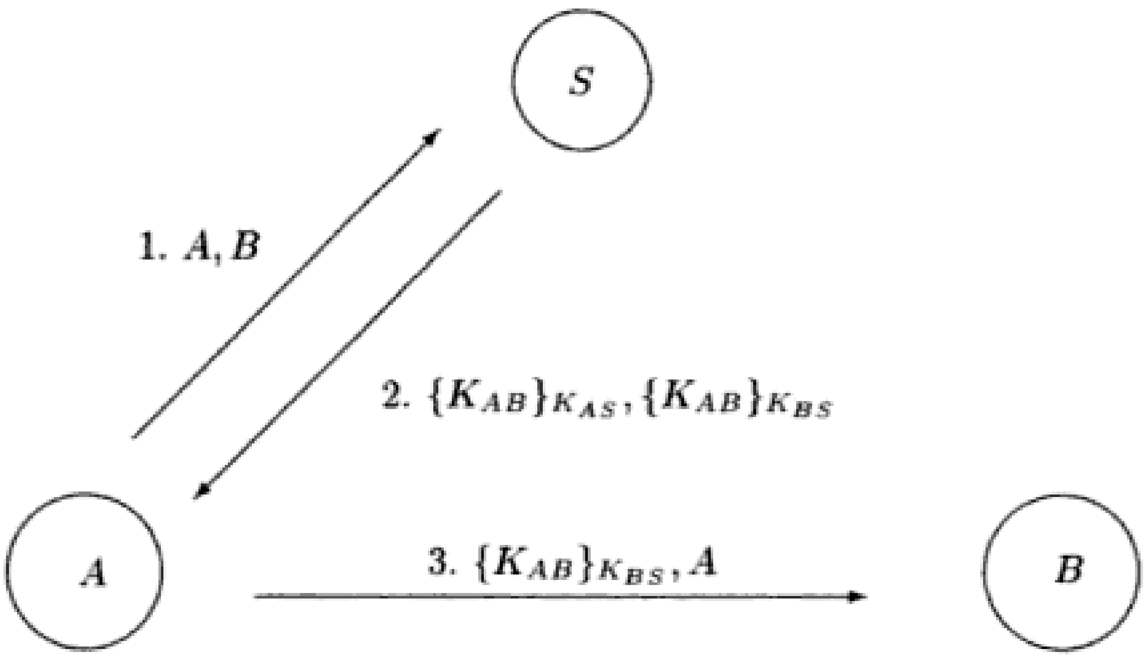
  G2 is not achieved

- Assume we have long-terms keys between parties
  - K_{AS} (between A & S)
  - K_{BS} (between B & S)
  - Note the differences between long-term and session keys

- Let's redesign the protocol

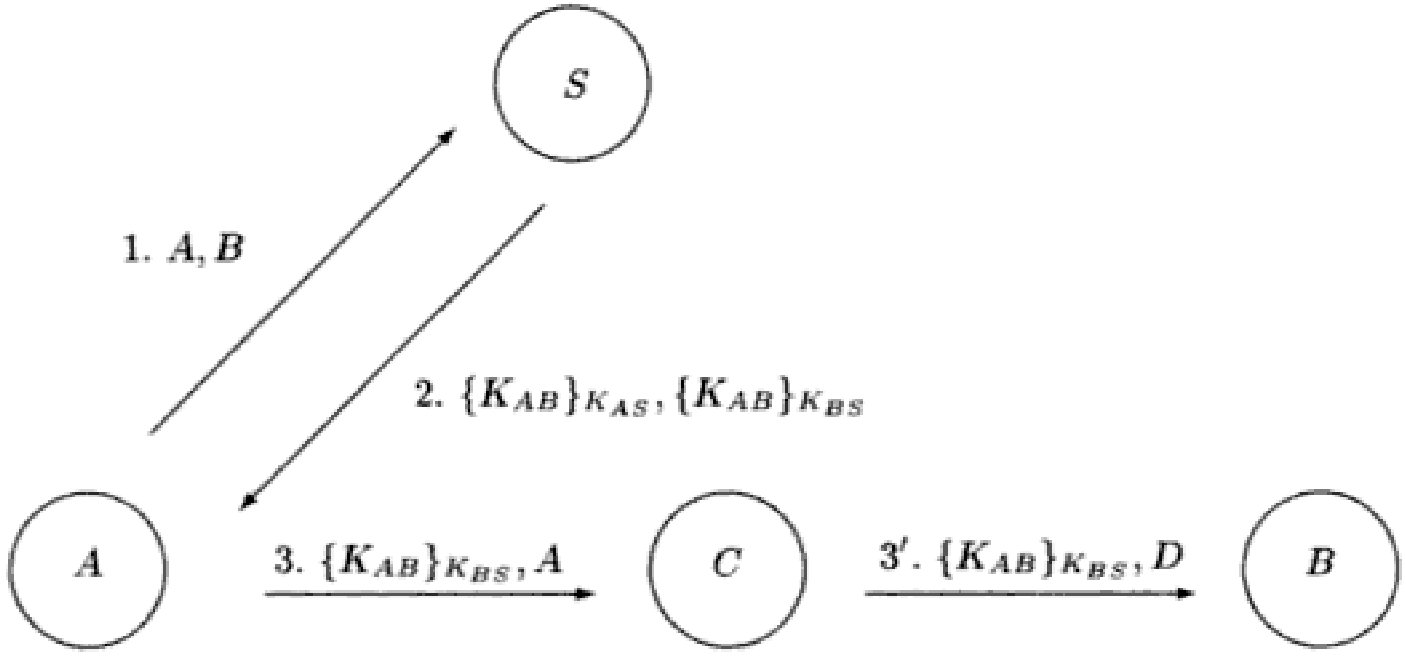
Protocol 1.2

- What goals do we achieve now?
- Who else may have access to K_{AB}?



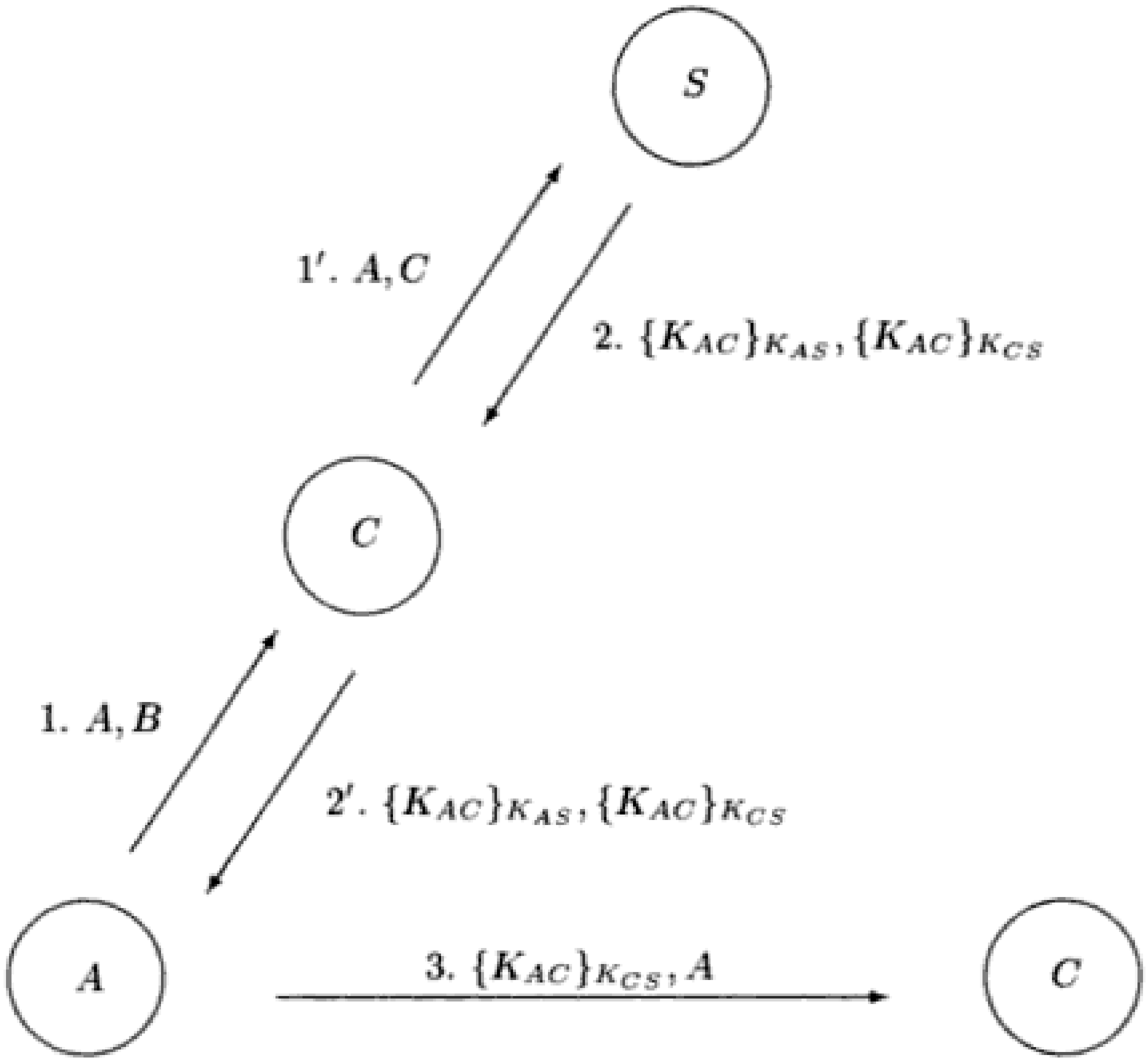
Attacks on Protocol 1.2

- Let's consider more security assumptions
  - "Attackers can alter & re-route protocol messages"
  - "Alter" = insert, remove, modify
- Attack: altering a principal



Attacks on Protocol 1.2

- Exposed session key (C must be an existing user):

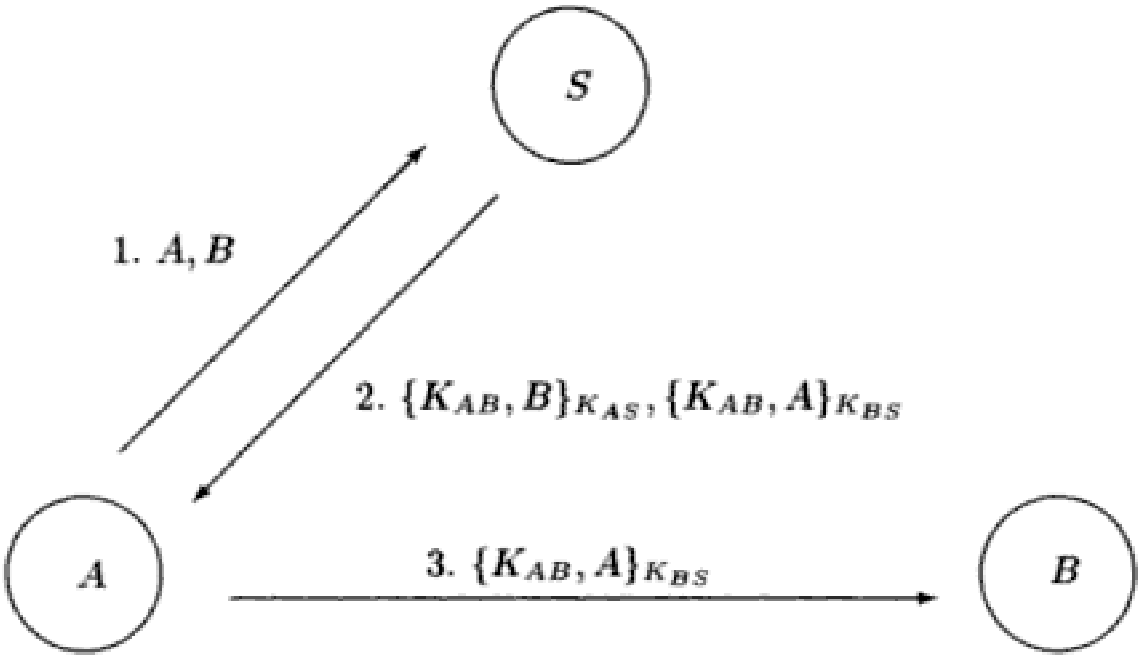


Why did the previous attack work?

- The session key is exposed because:
  - A, B received a legitimate session key generated by S – but not the one meant for them!
- Let's redesign…

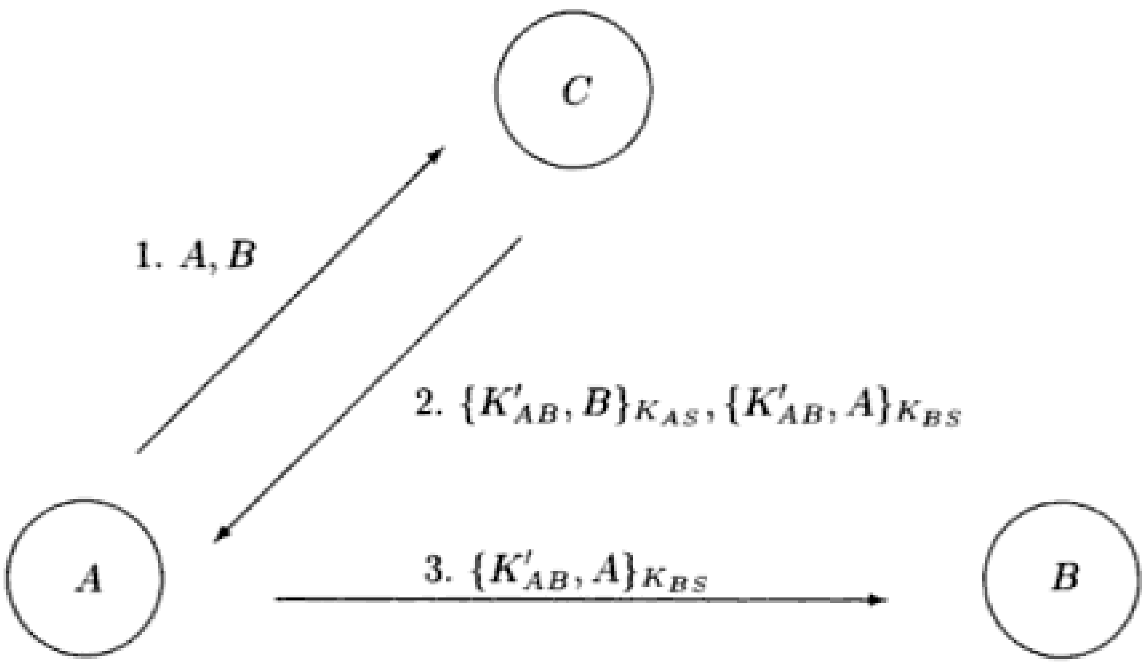
Protocol 1.3

- Let's add participants' names (i.e., unique identifiers)



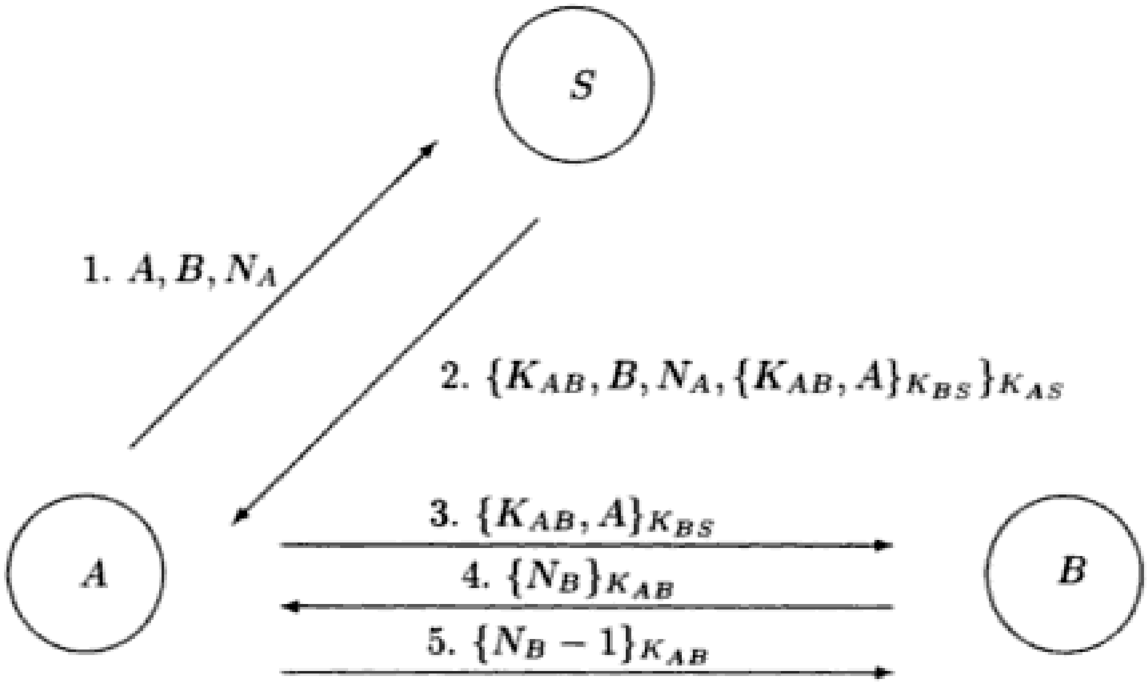
Attack on Protocol 1.3

- Assumption: past session keys may be leaked (How?)
- Replay attack:



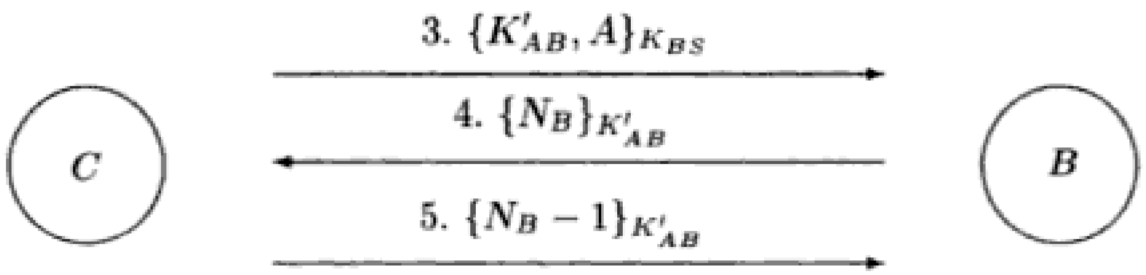
Protocol 1.4

- Let's add nonces



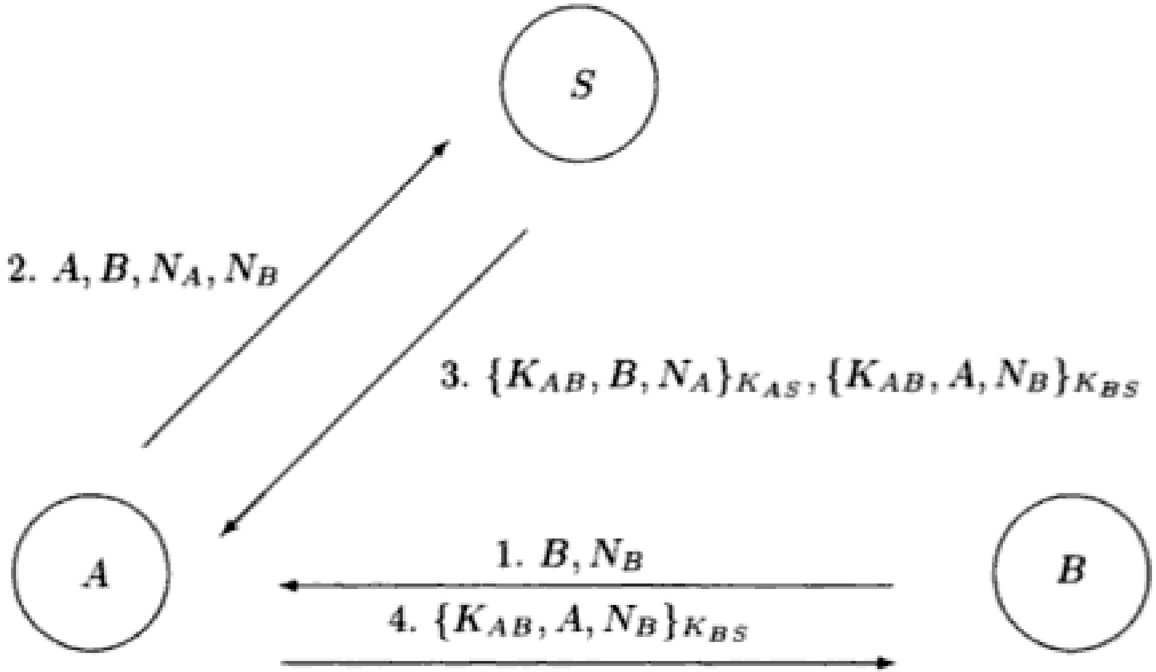
Attack on Protocol 1.4

- Again – think about an exposed session key



Protocol 1.5

- Another version…



Are we there yet?

- How do we know that Protocol 1.5 is safe?
  - Formal techniques
  - Tools (e.g., ProVerif, Tamarin, AVISPA)
  - Attack analysis

- But none of these techniques are perfect
  - Rely more on public review, test of time

- Many past protocols are found to be flawed

- Discovering a flaw helps us design better protocol

How to achieve freshness guarantee?

- Freshness is critical to prevent replays

- Two ways to get freshness guarantee of a value
  1. User takes part in choosing the value
     a) $K_{AB} = f(N_A, N_B)$
     b) f() must ensure that old K_{AB} cannot be generated even if N_A or N_B is known
     c) What choices we have for f()?

Freshness techniques (2)

2. User relies on something received with the value that is known to be fresh
   a) Timestamps
   b) Nonces (random challenges)
      i. $A \rightarrow B$: N_A
      ii. $B \rightarrow A$: $f(N_A, \ldots)$
   c) Counters