CNT4406/5412 Network Security

Real-time Communication Security

Zhi Wang

Florida State University

Fall 2014

Introduction

A **real-time protocol** negotiates interactively to authenticate each other and establish a session key

Introduction

A **real-time protocol** negotiates interactively to authenticate each other and establish a session key

- e.g., IPsec, SSL/TLS, and SSH

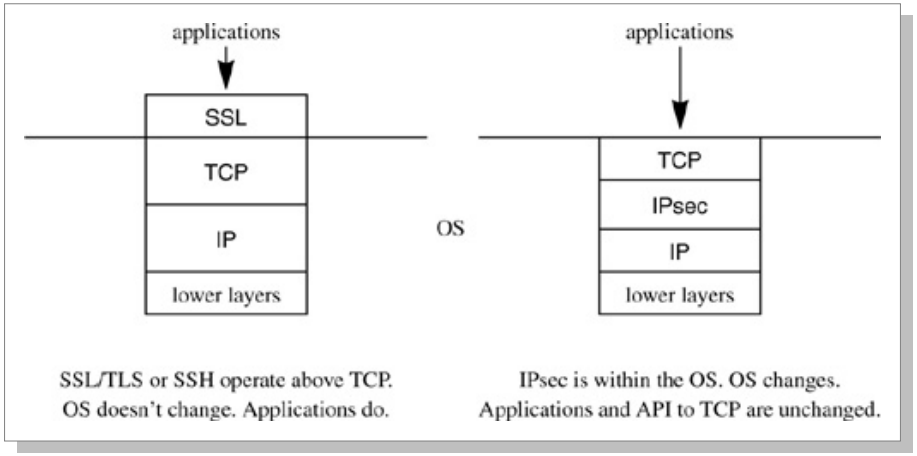
Introduction

A **real-time protocol** negotiates interactively to authenticate each other and establish a session key

- e.g., IPsec, SSL/TLS, and SSH
  - What layer?
  - Perfect forward secrecy
  - Denial-of-service protection
  - Endpoint identifier hiding
  - Live partner reassurance
  - Session resumption
  - Plausible deniability

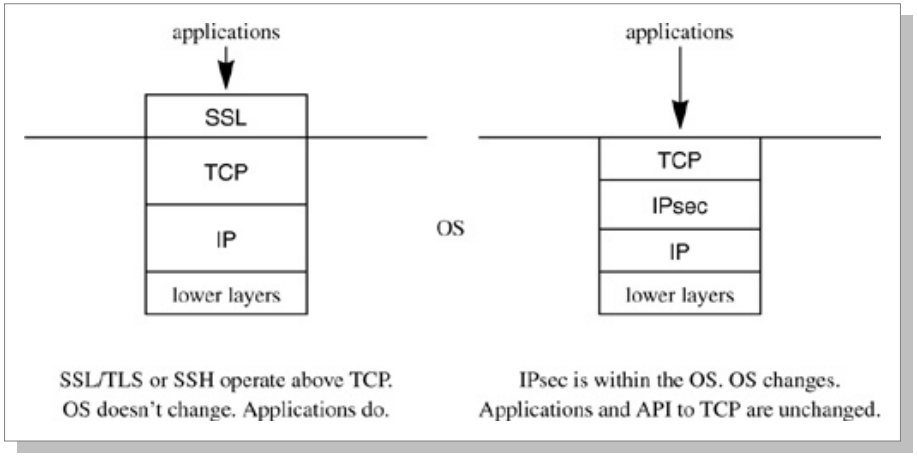
What Layer?

- IP stacks (i.e., TCP/UDP/IP...) are implemented in the OS kernel



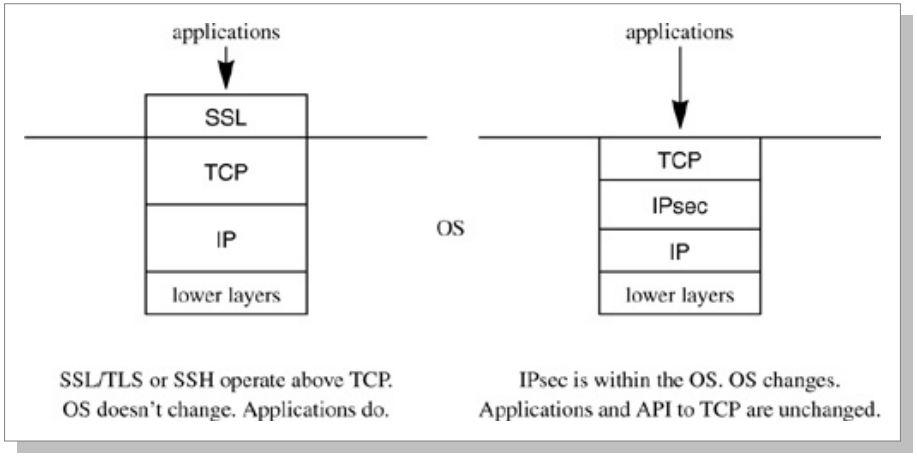
What Layer?

- IP stacks (i.e., TCP/UDP/IP...) are implemented in the OS kernel
- SSL and SSH are built above TCP in the user space
  - ➠ easy to deploy, but applications have to be (minimally) modified



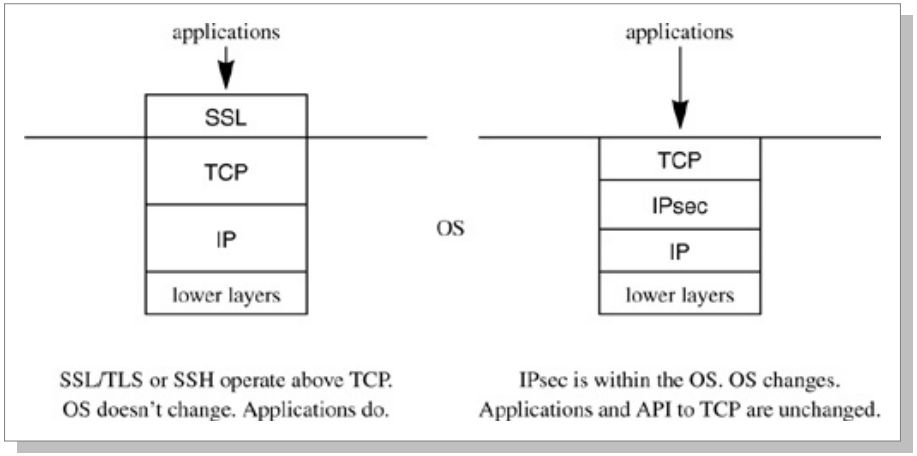
What Layer?

- IP stacks (i.e., TCP/UDP/IP...) are implemented in the OS kernel
- SSL and SSH are built above TCP in the user space
  - ➠ easy to deploy, but applications have to be (minimally) modified
- IPSec is implemented inside the OS
  - ➠ OS needs to be changed, applications may remain unchanged



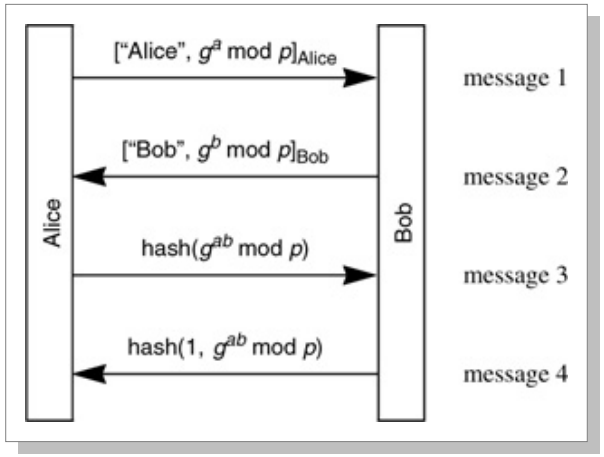
What Layer?

- IP stacks (i.e., TCP/UDP/IP...) are implemented in the OS kernel
- SSL and SSH are built above TCP in the user space
  - ➠ easy to deploy, but applications have to be (minimally) modified
- IPSec is implemented inside the OS
  - ➠ OS needs to be changed, applications may remain unchanged
  - ➠ changes are necessary to take full advantage of IPSec
    e.g., to authenticate a user instead of the IP address



Perfect Forward Secrecy

Perfect forward secrecy: it is impossible for Trudy to decrypt a recorded conversation even she subsequently steals all parties' long-term secrets

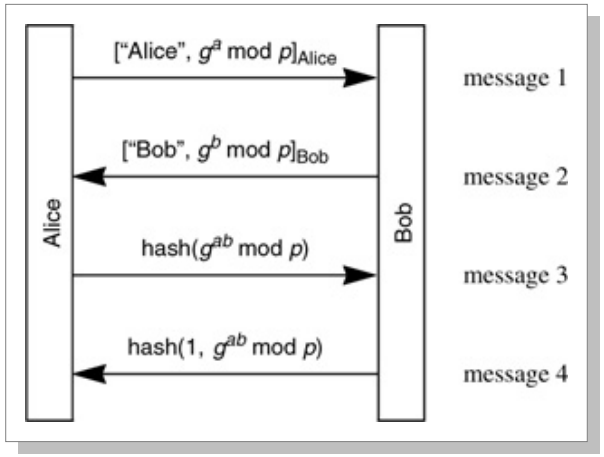


Perfect Forward Secrecy

Perfect forward secrecy: it is impossible for Trudy to decrypt a recorded conversation even she subsequently steals all parties' long-term secrets

To achieve PFS:

- generate a session key not derivable from stored information and cleartext data of the session

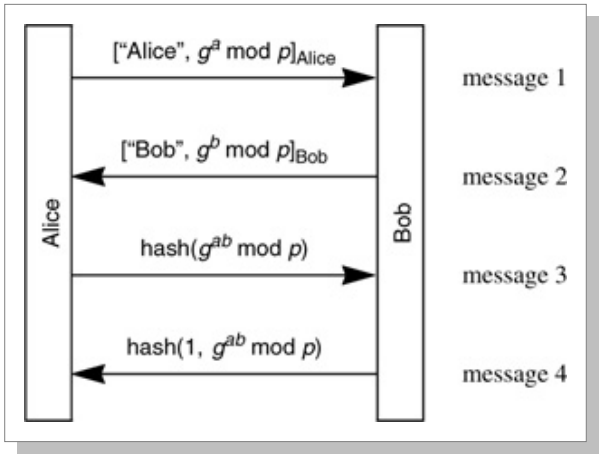


Perfect Forward Secrecy

Perfect forward secrecy: it is impossible for Trudy to decrypt a recorded conversation even she subsequently steals all parties' long-term secrets

To achieve PFS:

- generate a session key not derivable from stored information and cleartext data of the session
- forget the key after the session concludes



Perfect Forward Secrecy...

Do they have PFS?:

- Alice and Bob exchange messages encrypted with the peer's public key

Perfect Forward Secrecy...

Do they have PFS?:

- Alice and Bob exchange messages encrypted with the peer's public key
- Alice and Bob communicate using the session key issued by a KDC

Perfect Forward Secrecy...

Do they have PFS?:

- Alice and Bob exchange messages encrypted with the peer's public key
- Alice and Bob communicate using the session key issued by a KDC
- Alice picks a session key, encrypts it with Bob's public key, then signs and sends it to Bob

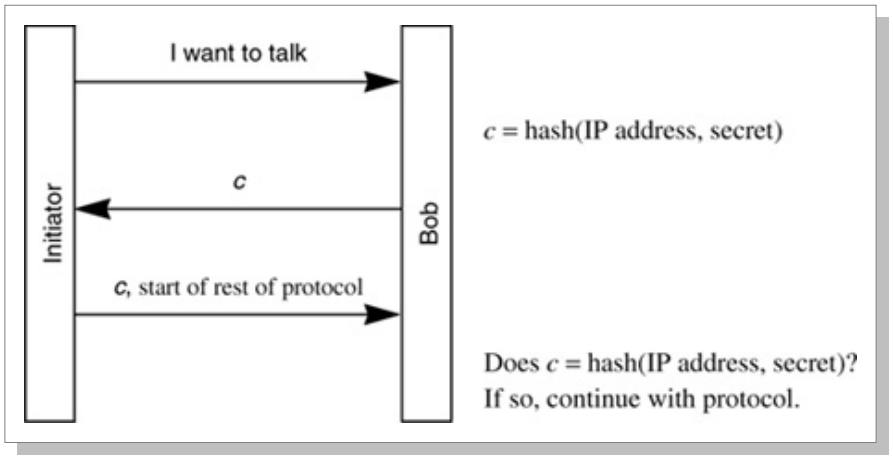
Denial-of-service Protection

To protect against DoS, avoid significant computation or saving states until the initiator is proved to be reachable

Denial-of-service Protection

To protect against DoS, avoid significant computation or saving states until the initiator is proved to be reachable

Stateless cookie

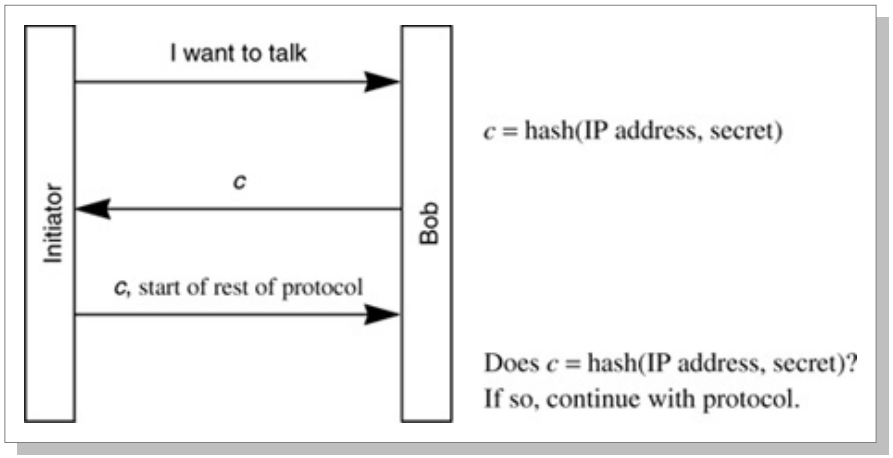


Denial-of-service Protection

To protect against DoS, avoid significant computation or saving states until the initiator is proved to be reachable

Stateless cookie

- Bob derives an unpredictable number from the connection

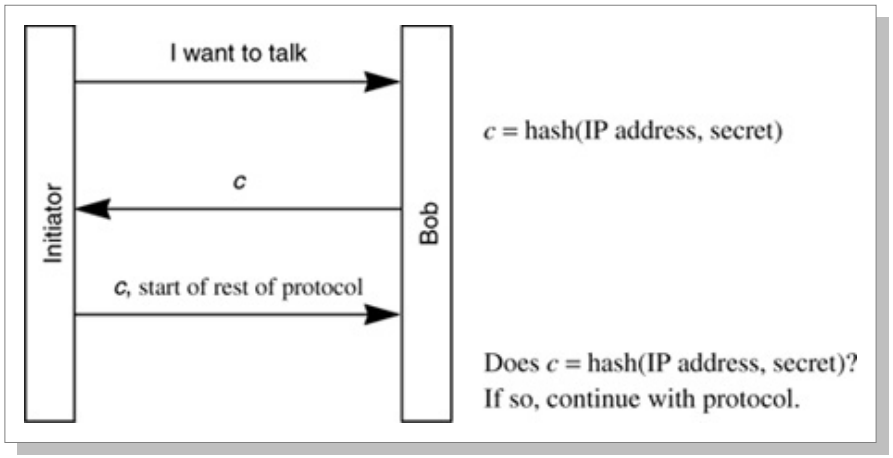


Denial-of-service Protection

To protect against DoS, avoid significant computation or saving states until the initiator is proved to be reachable

Stateless cookie

- Bob derives an unpredictable number from the connection
- Bob sends the number to Alice in the clear, Alice just return it

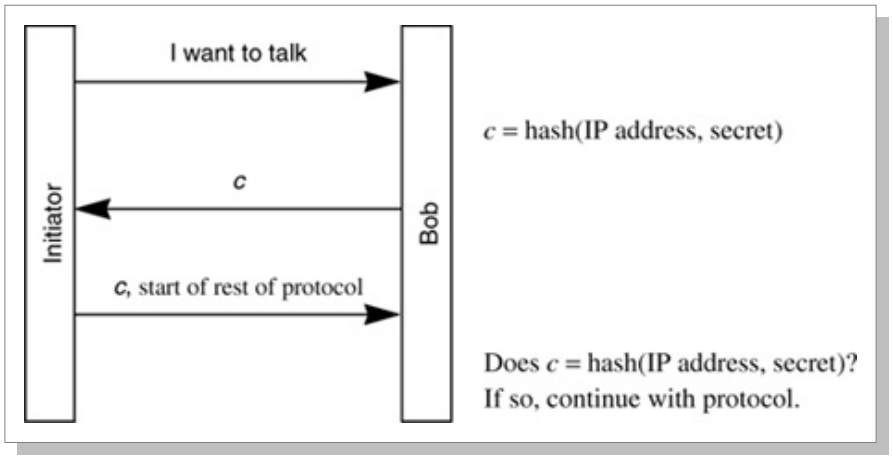


Denial-of-service Protection

To protect against DoS, avoid significant computation or saving states until the initiator is proved to be reachable

Stateless cookie

- Bob derives an unpredictable number from the connection
- Bob sends the number to Alice in the clear, Alice just return it
- Bob verifies the cookie by recomputing it ➠ no need to save states



Denial-of-service Protection: Puzzles

- Bob asks Alice to do more computation in order to connect
  - e.g., "what 27-bit number has a hash of x?"

Denial-of-service Protection: Puzzles

- Bob asks Alice to do more computation in order to connect
  - e.g., "what 27-bit number has a hash of x?"
  - verifying a puzzle should be fast and simple
- Problems:

Denial-of-service Protection: Puzzles

- Bob asks Alice to do more computation in order to connect
  - e.g., "what 27-bit number has a hash of x?"
  - verifying a puzzle should be fast and simple
- Problems:
  - difference in the computational powers for the clients

Denial-of-service Protection: Puzzles

- Bob asks Alice to do more computation in order to connect
  - e.g., "what 27-bit number has a hash of x?"
  - verifying a puzzle should be fast and simple
- Problems:
  - difference in the computational powers for the clients
  - not effective against distributed DoS (e.g., botnet)

Endpoint Identifier Hiding

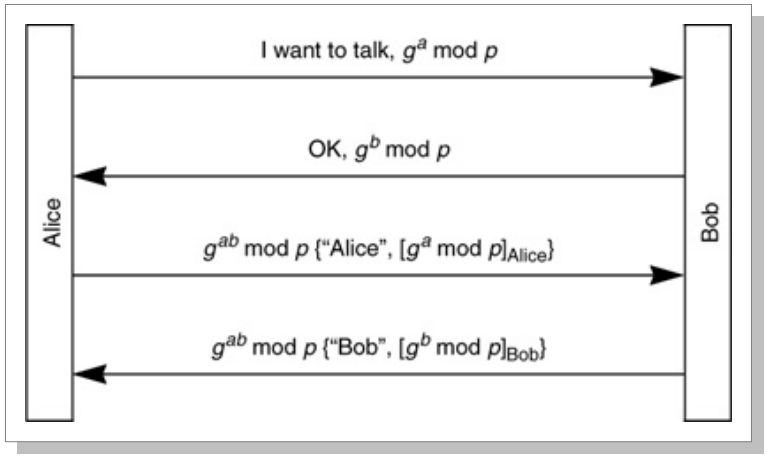
Endpoint identifier hiding: to hide the identities of the communicating parties from eavesdroppers

Endpoint Identifier Hiding

Endpoint identifier hiding: to hide the identities of the communicating parties from eavesdroppers

To achieve EIH:

- Do a Diffie-Hellman exchange to establish an encrypted tunnel

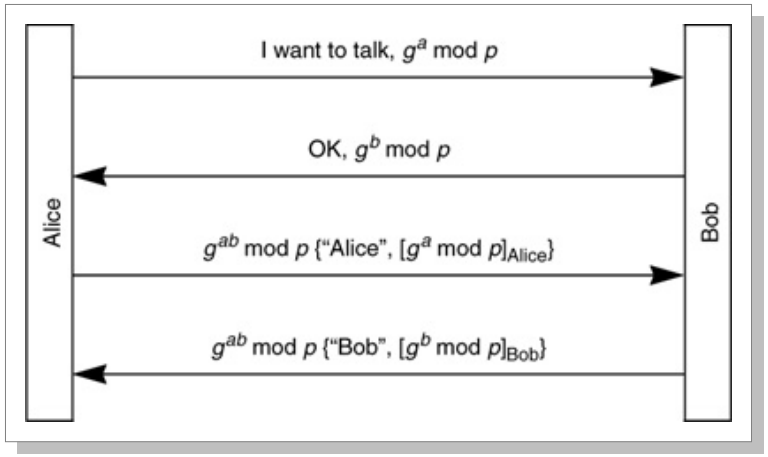


Endpoint Identifier Hiding

Endpoint identifier hiding: to hide the identities of the communicating parties from eavesdroppers

To achieve EIH:

- Do a Diffie-Hellman exchange to establish an encrypted tunnel
  - ➠ passive attackers cannot learn their identities
  - ➠ active attackers doing MITM can learn one or both identities

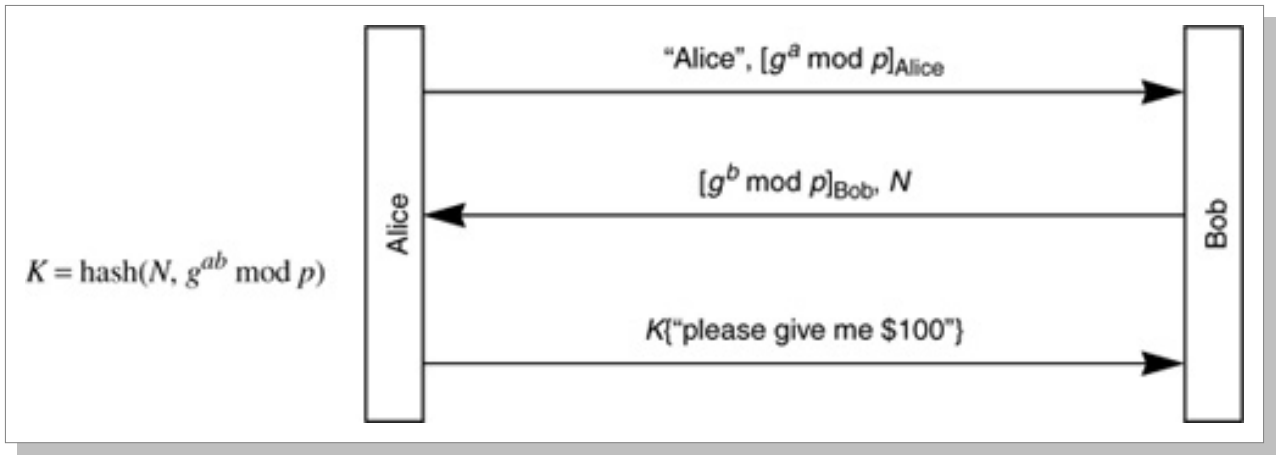


Live Partner Reassurance

- **Replay attack**: Trudy replays messages from previous conversations
  - ➠ replayed message may cause Bob to repeat some actions

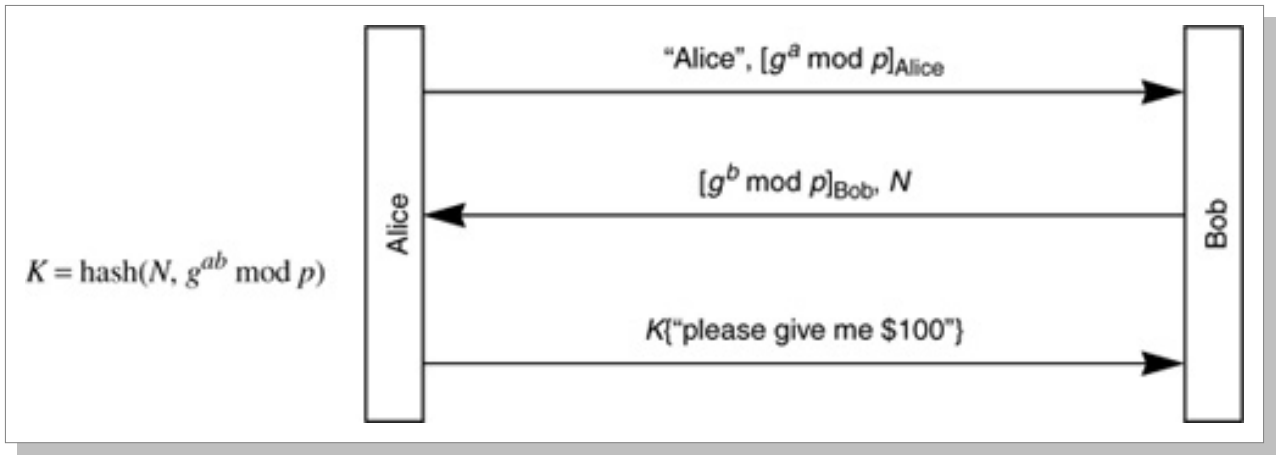
Live Partner Reassurance

- **Replay attack**: Trudy replays messages from previous conversations
  - ➠ replayed message may cause Bob to repeat some actions
- Use a random nounce for each connection to prevent replay attacks
  - ➠ e.g., set session key to $K = hash(N, g^{ab} \mod p)$



Live Partner Reassurance

- **Replay attack**: Trudy replays messages from previous conversations
  - replayed message may cause Bob to repeat some actions
- Use a random nounce for each connection to prevent replay attacks
  - e.g., set session key to $K = hash(N, g^{ab} \mod p)$
  - what is the difference from a cookie?



Session Resumption

Session resumption: bypass the initial public key authentication if Bob has recently authenticated Alice and established a session key

Session Resumption

Session resumption: bypass the initial public key authentication if Bob has recently authenticated Alice and established a session key

Example:

- Alice sends Bob $X_{ab} = [\{S\}_{Bob}]_{Alice}$, encrypted and signed session key

Session Resumption

Session resumption: bypass the initial public key authentication if Bob has recently authenticated Alice and established a session key

Example:

- Alice sends Bob $X_{ab} = [\{S\}_{Bob}]_{Alice}$, encrypted and signed session key
  - ➠ if Bob saves Alice's last X'_{ab} and $X_{ab} = X'_{ab}$, use the last S

Session Resumption

Session resumption: bypass the initial public key authentication if Bob has recently authenticated Alice and established a session key

Example:

- Alice sends Bob $X_{ab} = [\{S\}_{Bob}]_{Alice}$, encrypted and signed session key
  - if Bob saves Alice's last X'_{ab} and $X_{ab} = X'_{ab}$, use the last S
  - otherwise, Bob verifies and decrypts X_{ab} to get S

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?
  - ➠ yes, the session can be forged (solely) by Alice or Bob

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?
  ➠ yes, the session can be forged (solely) by Alice or Bob
- Alice and Bob authenticate using public encryption key?

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?
  ➠ yes, the session can be forged (solely) by Alice or Bob
- Alice and Bob authenticate using public encryption key?
  ➠ yes, anyone can make up a conversation

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?
  ➠ yes, the session can be forged (solely) by Alice or Bob
- Alice and Bob authenticate using public encryption key?
  ➠ yes, anyone can make up a conversation
- Alice and Bob authenticate by signing the other's identity?

Plausible Deniability

Plausible deniability: to design the protocol so there is a lack of evidence proving that two parties talked

Does the following protocol have plausible deniability?

- Alice and Bob authenticate with a shared secret?
  ➠ yes, the session can be forged (solely) by Alice or Bob
- Alice and Bob authenticate using public encryption key?
  ➠ yes, anyone can make up a conversation
- Alice and Bob authenticate by signing the other's identity?
  ➠ no, only Alice or Bob can access its signature key

Summary

- What layer?
- Perfect forward secrecy
- Denial-of-service protection
- Endpoint identifier hiding
- Live partner reassurance
- Session resumption
- Plausible deniability

- Next lecture: IPSec