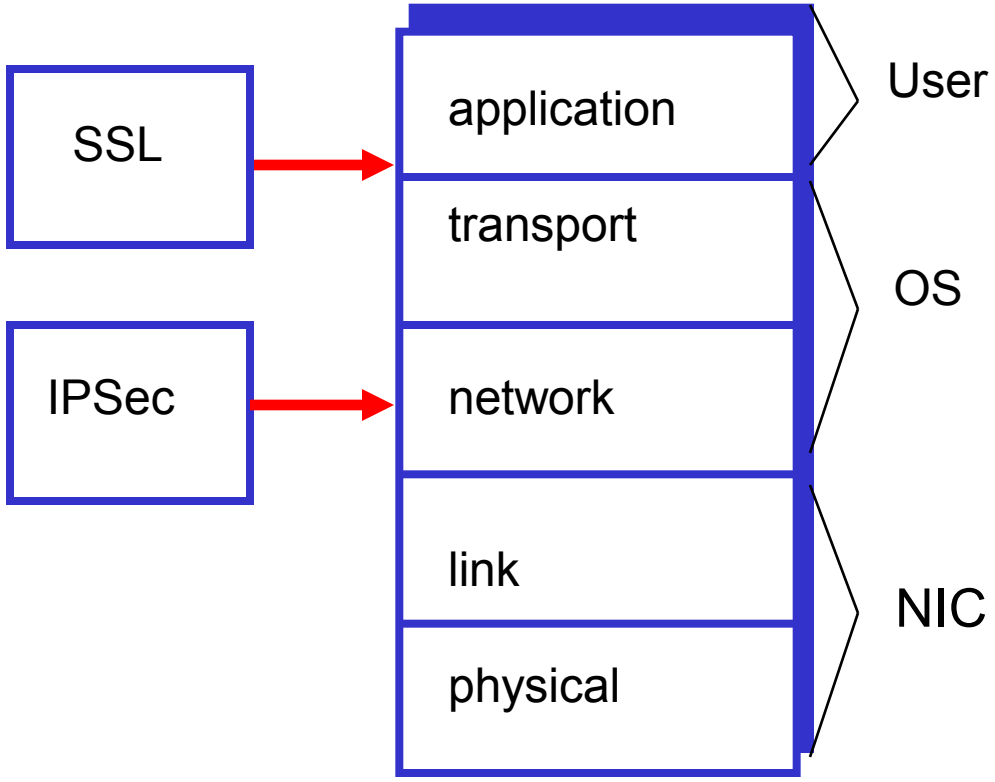
SSL and TLS

- Readings
  - Chapter 19

Internet Security Protocols

- IPSec and SSL
- Socket layer lives between application and transport layers
- SSL usually lives between HTTP and TCP

- IPSec lives at the network layer



What is SSL (Secure Socket Layer)?

- SSL is the protocol used for most secure transactions over the Internet
- For example, if you want to buy a book at amazon.com…
  - You want to be sure you are dealing with Amazon (**authentication**)
  - Your credit card information must be protected in transit (**confidentiality** and/or **integrity**)
  - As long as you have money, Amazon doesn't care who you are (authentication need not be mutual)

Secure Socket Layer

- Originally developed by Netscape
- SSL runs above layer 4 (the transport control layer), It is sometimes said to be at layer 4
- In practice, SSL uses TCP sockets
  - The underlying TCP implementation handles robustness of communication, such as replay of lost packets, buffering packets to re-order them correctly, etc.
- SSL extends the TCP interface (sockets API) by adding security features
- Versions 2 & 3, with SSLv3 most commonly deployed.
  - TLS (Transport Layer Security) is a variation of v3
  - Internet standard

What does It Entail?

- To use SSL, applications must change.
  - They have to use the SSL API (application programming interface) and use SSL calls instead of TCP socket calls. Applications' networking code must change.
  - See for example OpenSSL, based on the SSLeay Library.
- SSL runs above TCP/IP (transport/network layer) and below high-level application protocols such as HTTP, LDAP and IMAP.
- SSL may be deployed without making changes to the underlying Operating System, because it does not alter the implementation of the TCP protocol.

Capabilities of SSL

- SSL server authentication
  - User can confirm a server's identity
  - SSL client software checks server's certificate and public key, and that they have been issued by a CA in the clients list of trusted CAs.
- SSL client authentication
  - Server can optionally check client identity
- Encrypted SSL Connection
  - Supports encryption and data integrity

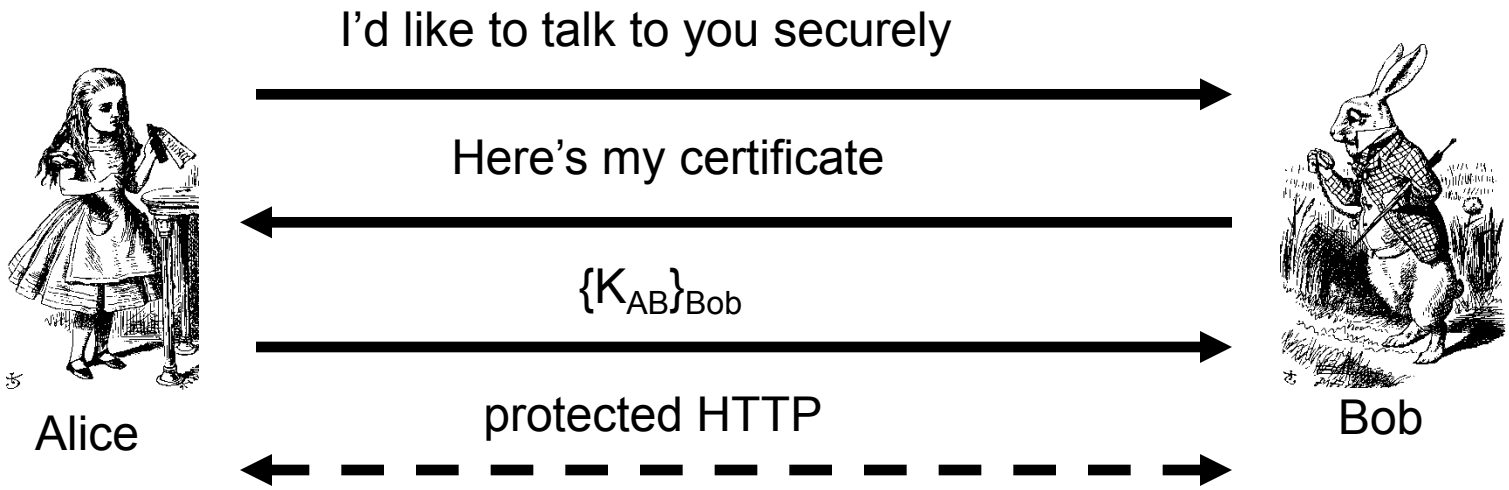
SSL/TLS: First Ingredients

- SSL supports several “cipher suites”:
  - Algorithm sets for public key encryption, symmetric key encryption, and authentication (MACs).
  - Flexibility was needed because of export restrictions.
  - Client and server must negotiate which algorithms are used in a session.
- Client and server agree on a common secret:
  - Negotiated using public key cryptography
  - Incorporates challenges (nonces) from both parties.
  - From this common secret the symmetric keys are derived.
- SSL’s focus is on real-time communication security
  - for applications such as those requiring authentication of web sites but not specifically the authentication of clients

SSL/TLS: Ingredients (2)

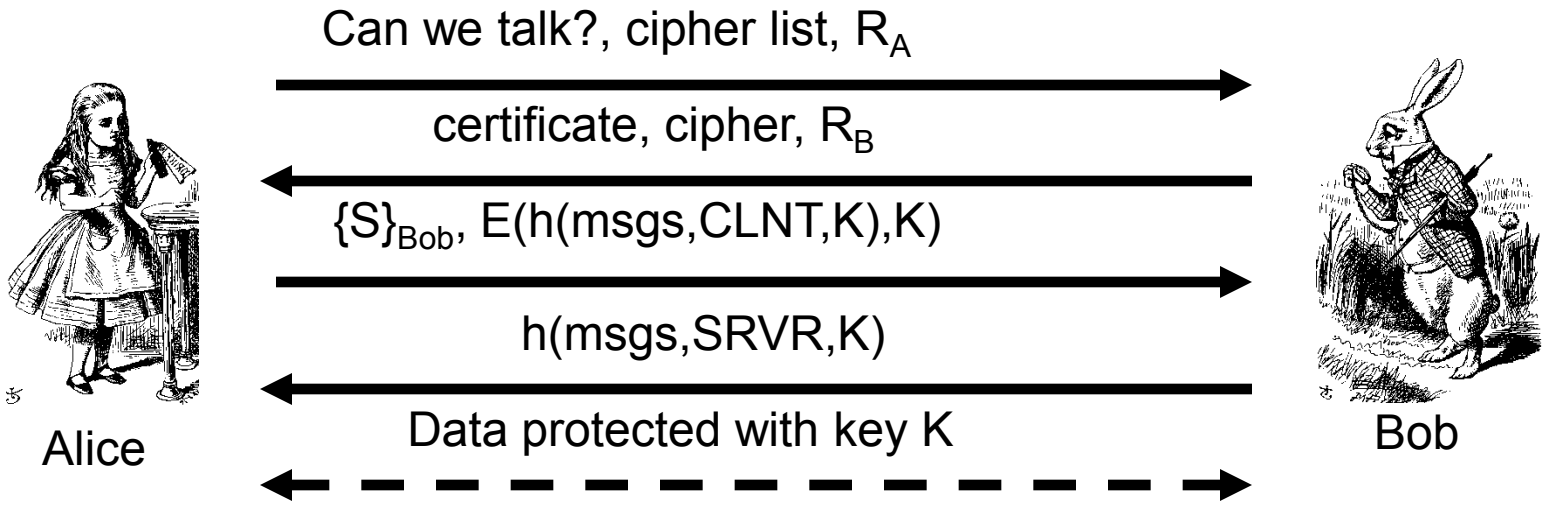
- SSL uses *directional* symmetric keys. After agreeing on a common secret (master key K):
  - Client and server derive from it two IVs, *encrypt* and *decrypt* keys, as well as *authentication* and *verification* keys. A total of six secrets are derived from the agreed secret (pre-key).
  - Client read keys = Server write keys
    - Server IV = part of Server encryption parameters = part of client decryption parameters
    - Server encryption key = Client decryption key
    - Server authentication key = Client verification key (also called integrity keys or MAC keys)
  - Client write keys = Server read keys
    - Client IV = part of Client encryption parameters = part of server decryption parameters
    - Client encryption key = Server decryption key
    - Client authentication key = Server verification key

Simple SSL-like Protocol



- Is Alice sure she's talking to Bob?
- Is Bob sure he's talking to Alice?

Simplified SSL Protocol



- S is **pre-master secret,** chosen by Alice
- $K = h(S,R_A,R_B)$
- msgs = all previous messages
- CLNT and SRVR are constants

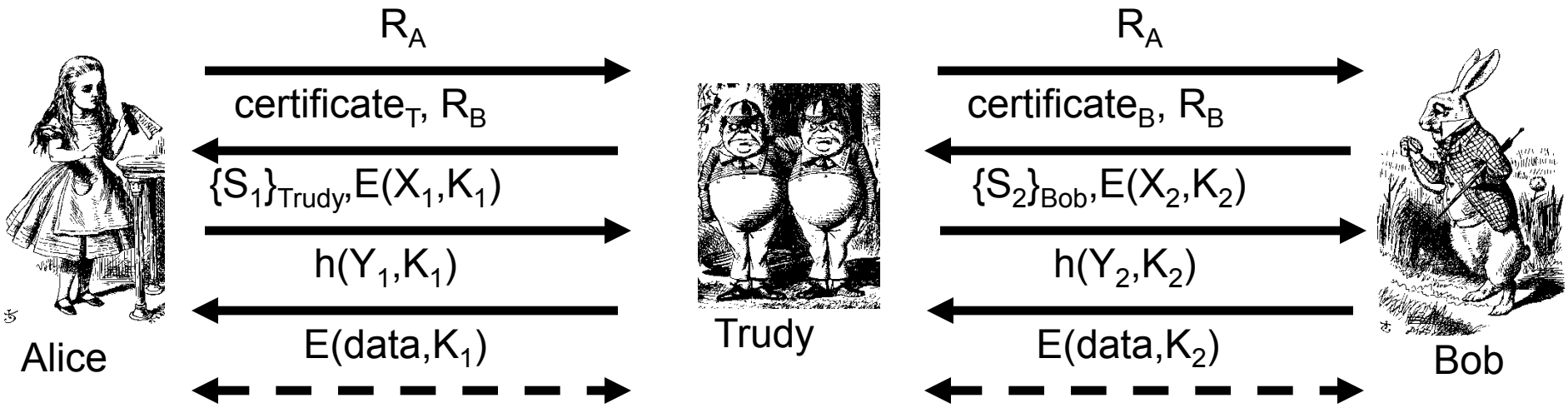
SSL Keys

- 6 “keys” derived from master key K = hash(S,R_A,R_B)
  - 2 encryption keys: send and receive
  - 2 integrity keys: send and receive
  - 2 IVs: send and receive (client and server side IVs)
  - Why different keys in each direction?
- **Q:** Why is h(msgs,CLNT,K) encrypted (and integrity protected)?
- **A:** It adds no security…

SSL Authentication

- Alice authenticates Bob, not vice-versa
  - How does client authenticate server?
  - Why does server not authenticate client?
- Mutual authentication is possible: Bob sends **certificate request** in message 2
  - This requires client to have certificate
  - If server wants to authenticate client, server could instead require (encrypted) password

SSL MiM Attack

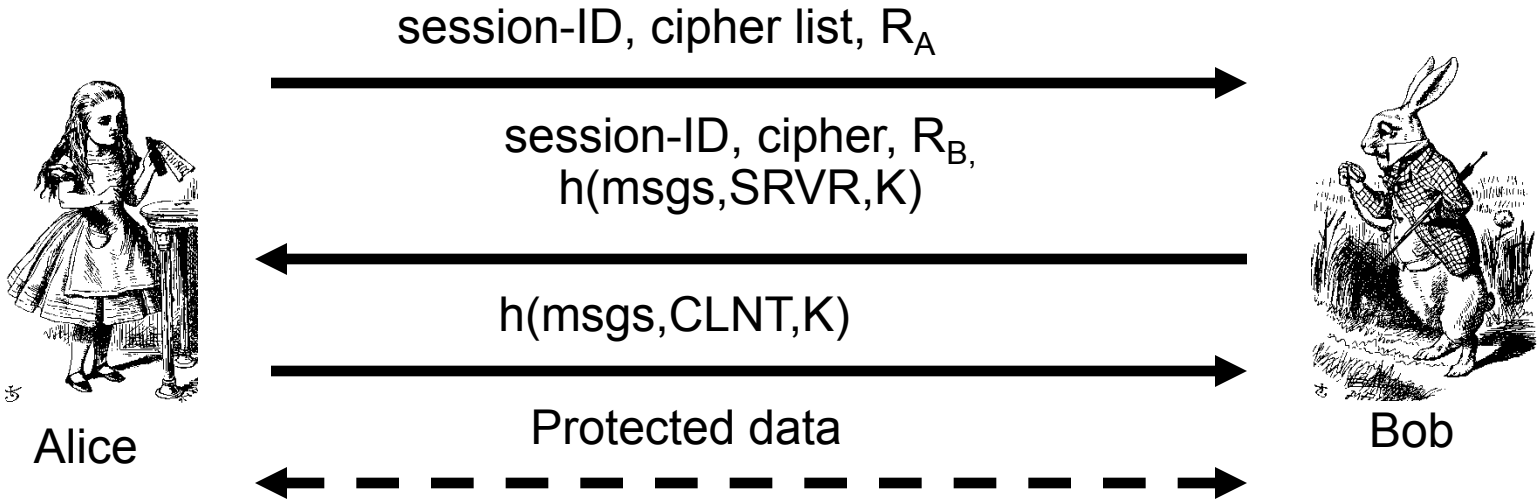


- **Q:** What prevents this MiM attack?
- **A:** Bob's certificate must be signed by a certificate authority (such as Verisign)
- What does Web browser do if sig. not valid?
- What does user do if signature is not valid?

SSL Sessions vs Connections

- SSL **session** is established as shown on previous slides
- SSL designed for use with HTTP 1.0
- HTTP 1.0 usually opens multiple simultaneous (parallel) **connections**
- SSL session establishment is costly
  - Due to public key operations
- SSL has an efficient protocol for opening new connections given an existing session

SSL Connection



- Assuming SSL **session** exists
- So S is already known to Alice and Bob
- Both sides must remember session-ID
- Again, $K = h(S,R_A,R_B)$
  - No public key operations! (relies on known S)

HTTPS

- Hypertext Transfer Protocol Secure
  - Different from S-HTTP (Secure HTTP)
- Not a new single protocol
  - A combination of HTTP and SSL/TLS
  - On default port number 443 instead of 80

Reading Assignment

- Chapter 17