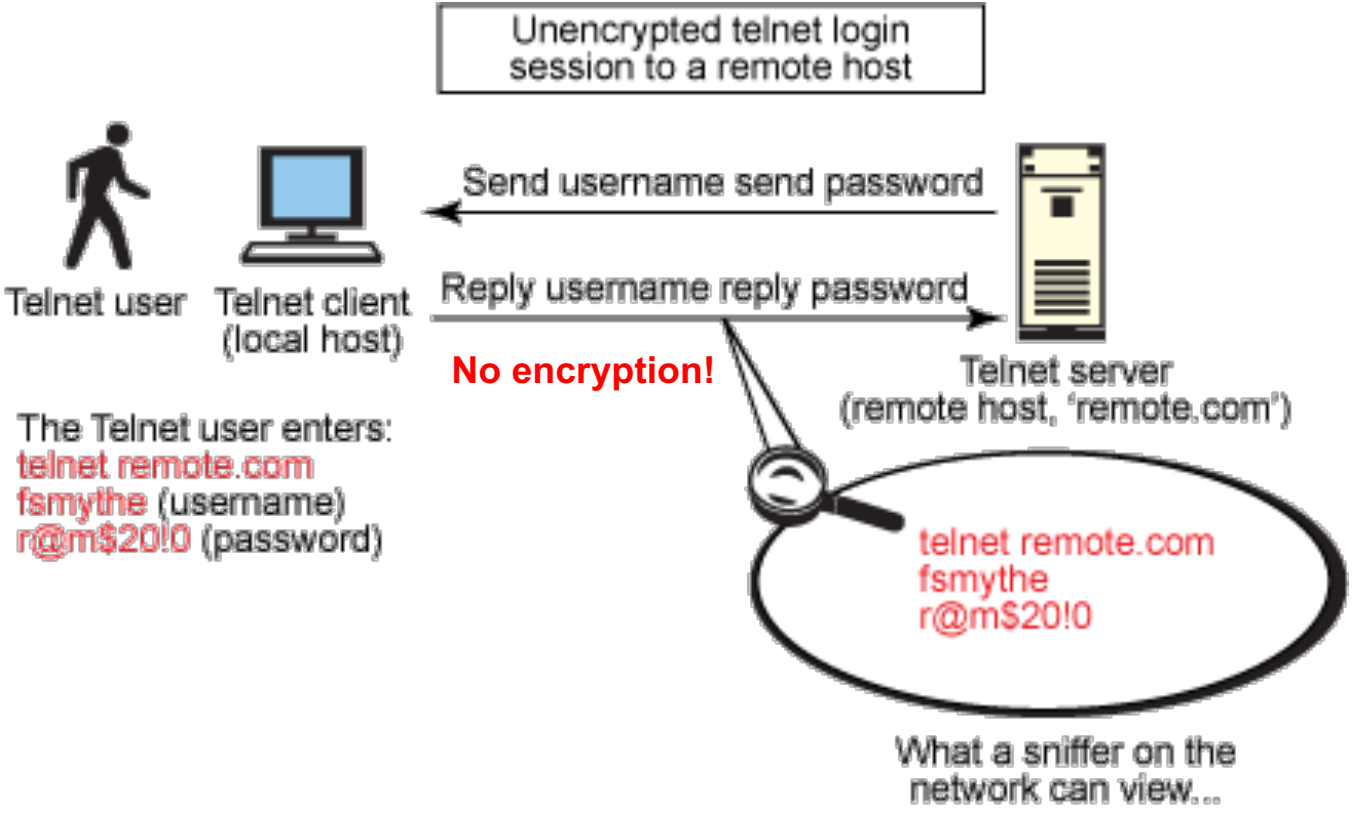
SSH and SSL/TLS

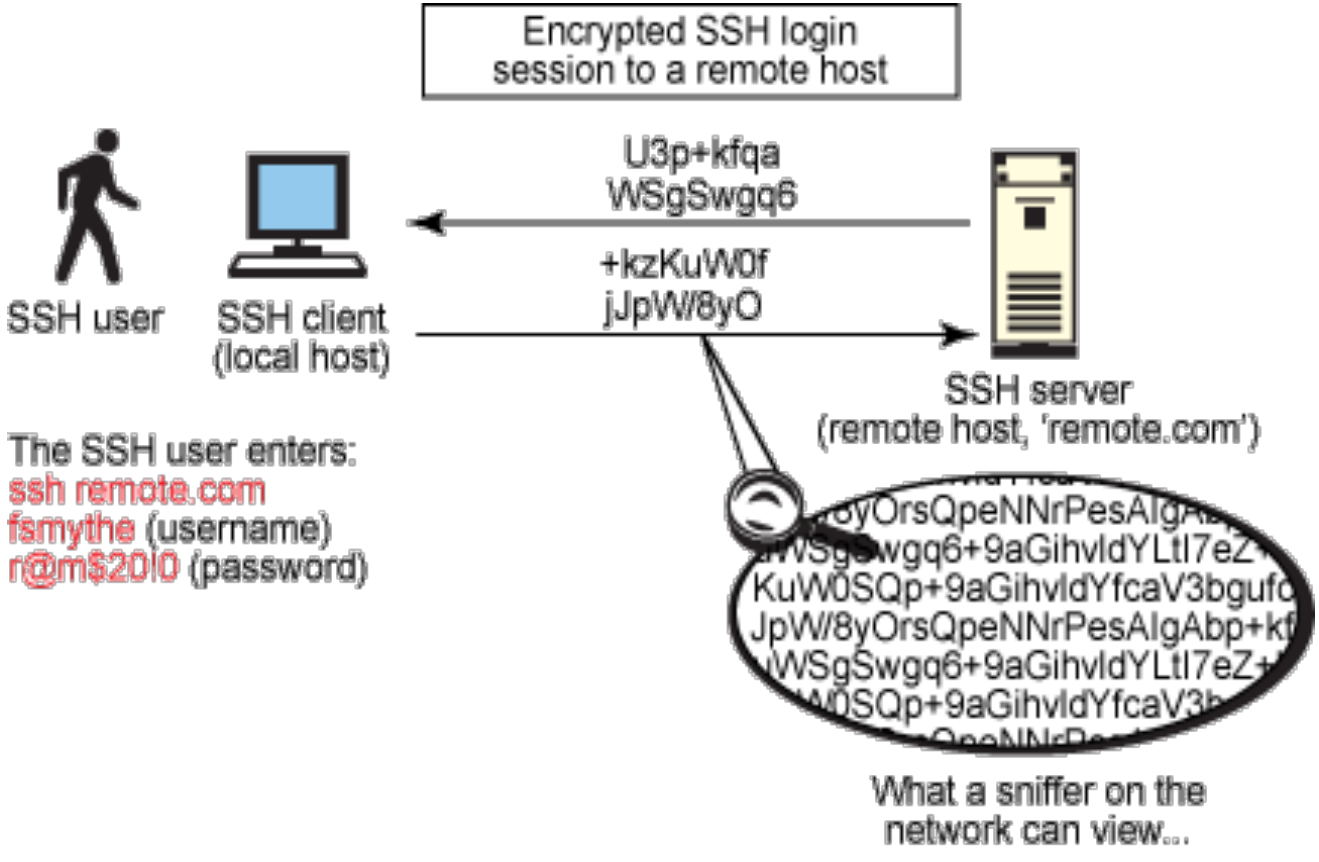
Real-World Protocols

- Next, we look at real protocols
  - **SSH** — a simple & useful security protocol
  - **SSL** — practical security on the Web
  - **IPSec** — security at the IP layer
  - **Kerberos** — symmetric key, single sign-on
  - **WEP** — "Swiss cheese" of security protocols
  - **GSM** — mobile phone (in)security

Telnet – Insecure Network protocol



SSH – Secure Network Protocol



SSH

- Creates a “secure tunnel”
- Insecure commands sent through SSH tunnel are then secure
- SSH used with things like `rlogin`
  - Why is rlogin insecure without SSH?
    - Password is sent in the clear
    - Attacker can observe
  - Why is rlogin secure with SSH?
    - Any commands in a SSH session is secured(encrypted)
- SSH is a relatively simple protocol

SSH related commands

- Run: **`ssh xzhang@remote.cs.binghamton.edu`**

- Run: **`ssh-keyscan -t rsa remote.cs.binghamton.edu`**

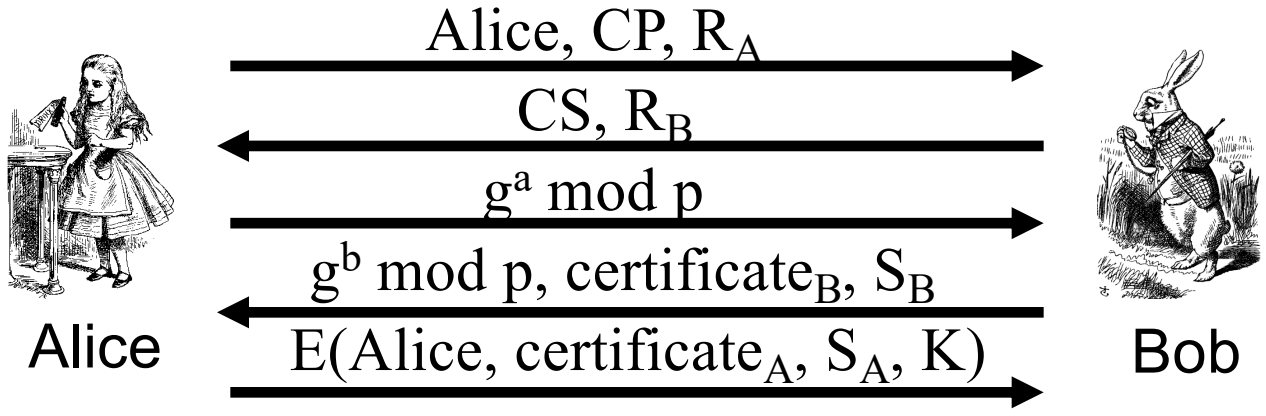
```
# remote.cs.binghamton.edu SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze5
remote.cs.binghamton.edu ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAtD0z6ezSt8vAL/ZL+/yInr/uY20JWSx5jspCi4TRrCC3Xvjq
O6Fvkuns+EYeMb1JYBh1+nxp2RMfRlVPdHuuKCEAbFPvyiLUNS89A0kU8Svurck7oBUlnU17o
Wtjty177Ewicy3eAMSLXDWl7BfbwNVs6roGcxQbnQ+70h6wZtM=
```

- Run: **`ssh-keygen`**
  - Version 1: create RSA keys
  - Version 2: create DSA, EC-DSA, or RSA (default) keys

SSH – How it works

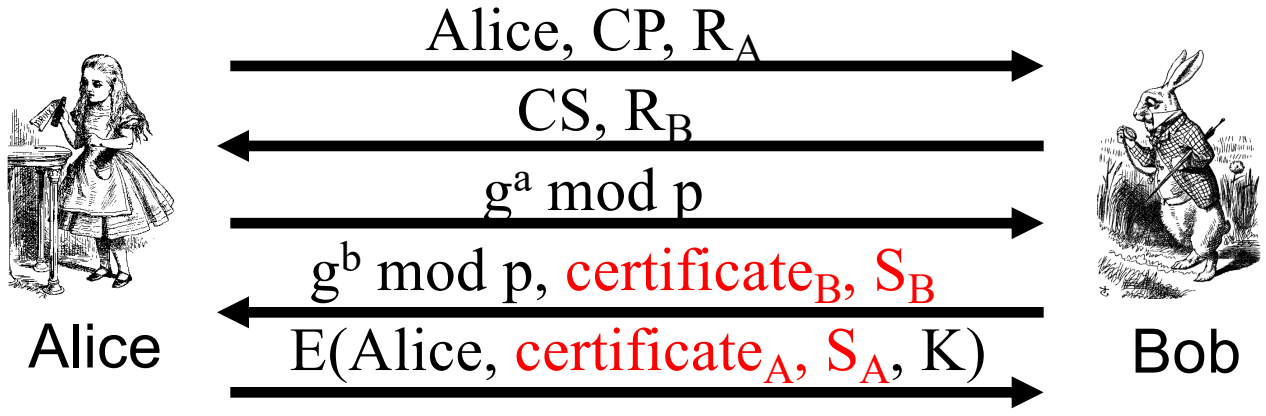
- **Connection Setup**
  - When a SSH client connect to an SSH server, they perform a version exchange to identify what version of protocol to use
- **Server authentication**
  - Client verifies the server's identity by checking its public key
- **User authentication**
  - User can authenticate using a password, an SSH key pair, certificates
- **Session Encryption**
  - Once both authenticated, all future commands are encrypted
- Here, we consider the **certificate mode** authentication
- And using a slightly simple SSH as an example

Simplified SSH



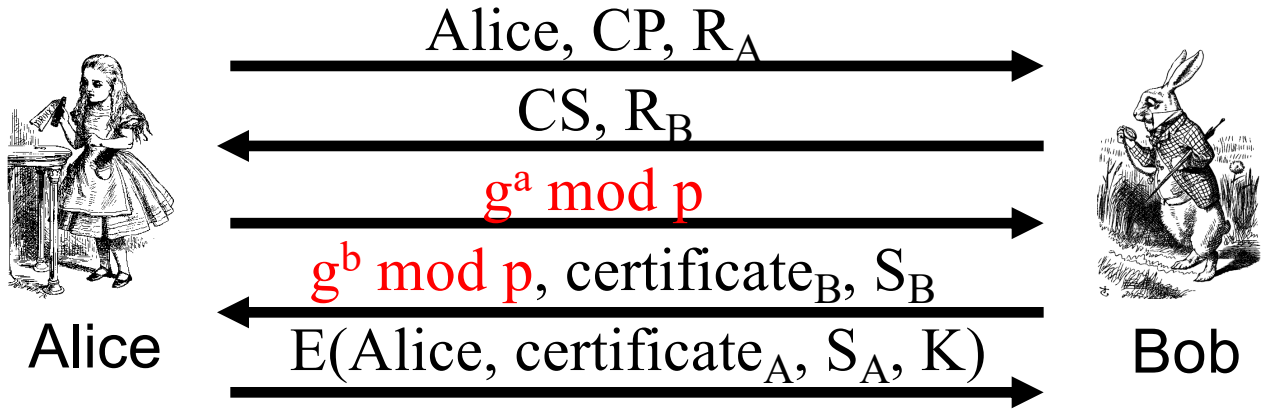
- **CP** = "crypto proposed", and **CS** = "crypto selected"
- **H** = h(Alice,Bob,CP,CS,$\mathbf{R_A}$,$\mathbf{R_B}$,g^a mod p,g^b mod p,g^{ab} mod p)
- $\mathbf{S_B}$ = $[H]_{Bob}$
- $\mathbf{S_A}$ = $[H, Alice, certificate_A]_{Alice}$
- **K** = $\underline{g^{ab} \text{ mod } p}$

How is Alice/Bob authenticated?



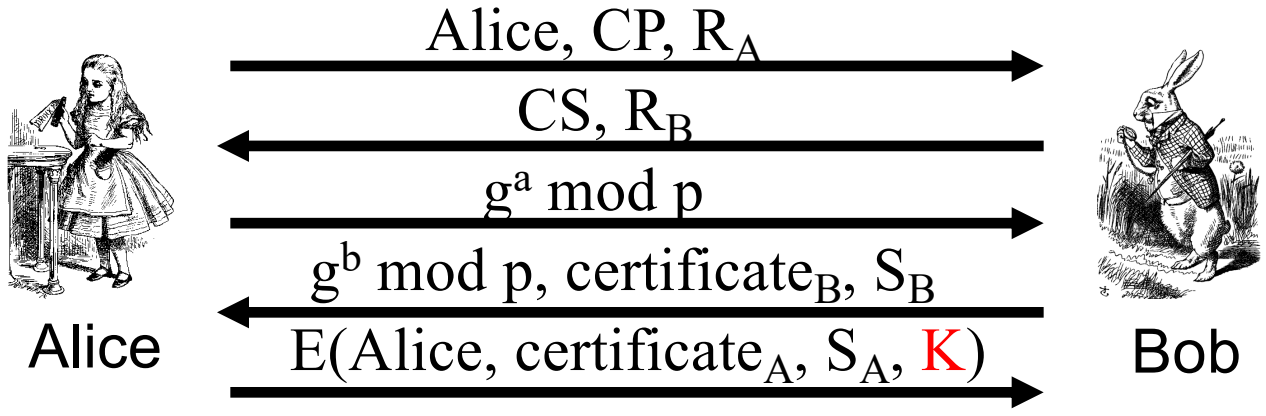
- **CP** = "crypto proposed", and **CS** = "crypto selected"
- **H** = h(Alice,Bob,CP,CS,R_A,R_B,g^a mod p,g^b mod p,g^{ab} mod p)
- $\mathbf{S_B} = [H]_{Bob}$
- $\mathbf{S_A} = [H, Alice, certificate_A]_{Alice}$
- **K** = g^{ab} mod p

If Trudy is a passive attacker, can she get K?



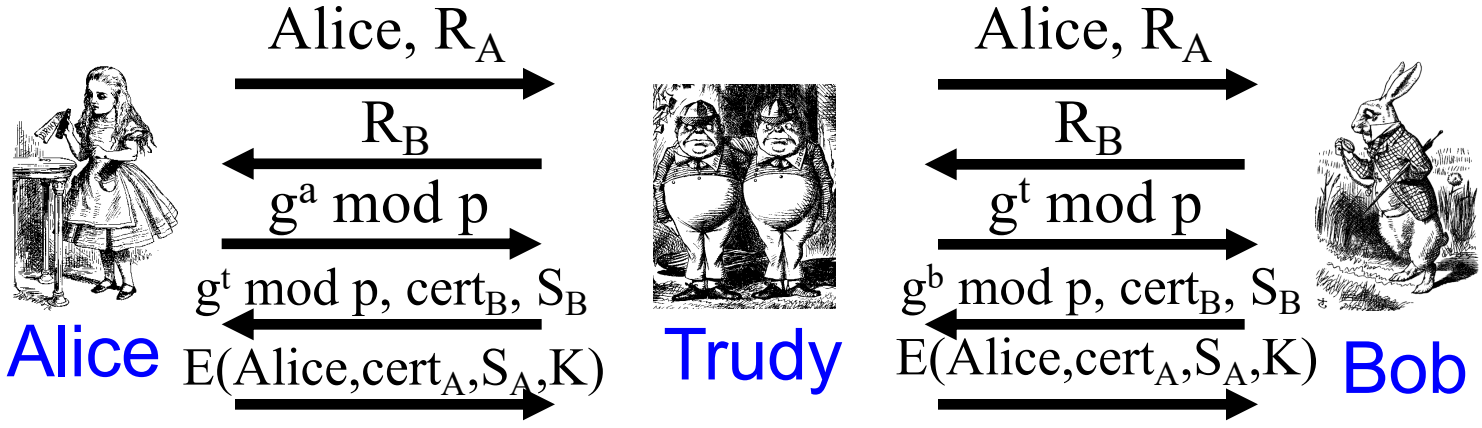
- **CP** = "crypto proposed", and **CS** = "crypto selected"
- **H** = h(Alice,Bob,CP,CS,$\mathbf{R_A}$,$\mathbf{R_B}$,g^a mod p,g^b mod p,g^{ab} mod p)
- $\mathbf{S_B}$ = $[H]_{Bob}$
- $\mathbf{S_A}$ = $[H, Alice, certificate_A]_{Alice}$
- **K** = $\underline{g^{ab} \text{ mod } p}$

What's the purpose of encryption with K?



- **CP** = "crypto proposed", and **CS** = "crypto selected"
- **H** = h(Alice,Bob,CP,CS,$\mathbf{R_A}$,$\mathbf{R_B}$,g^a mod p,g^b mod p,g^{ab} mod p)
- $\mathbf{S_B}$ = $[H]_{Bob}$
- $\mathbf{S_A}$ = $[H, Alice, certificate_A]_{Alice}$
- **K** = $\underline{g^{ab} \text{ mod } p}$

MiM Attack on SSH?

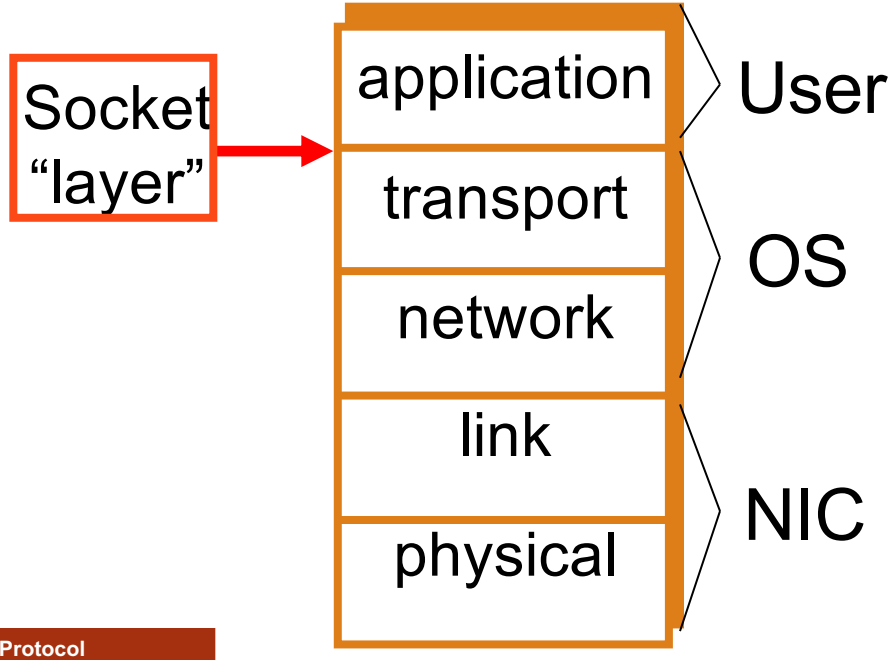


- Where does this attack fail?
- Alice computes:
  - H_a = h(Alice,Bob,CP,CS,R_A,R_B,**g^a mod p**,g^t mod p,**g^{at} mod p**)
- But Bob signs:
  - H_b = h(Alice,Bob,CP,CS,R_A,R_B,g^t mod p,**g^b mod p**,**g^{bt} mod p**)

Secure Socket Layer

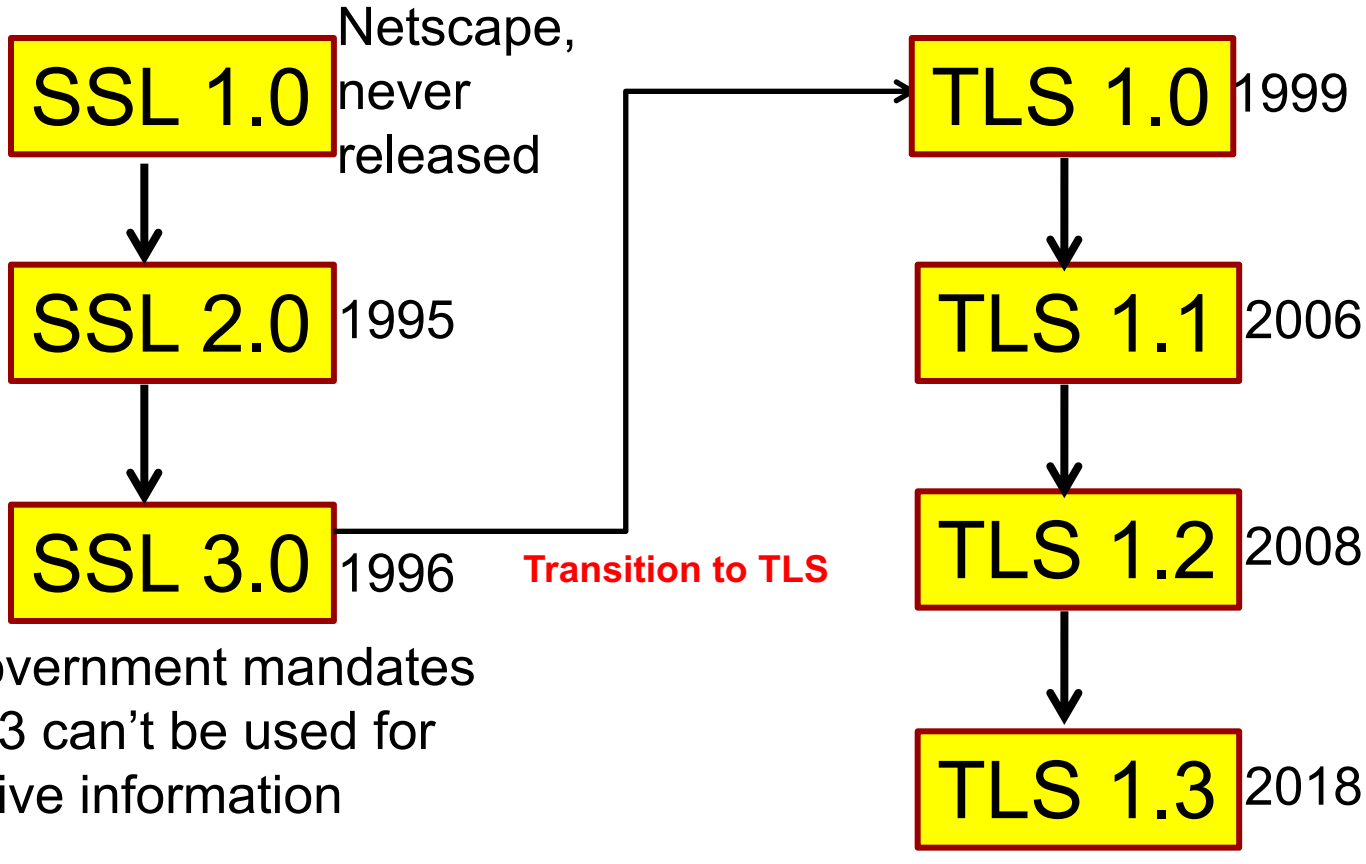
Socket layer

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



Protocol	Secure Protocol
HTTP: 80	HTTPS: 443
IMAP: 143	Secure IMAP: 993
POP3: 110	Secure POP3: 995
SMTP: 25	Secure SMTP: 465

History of SSL and TLS



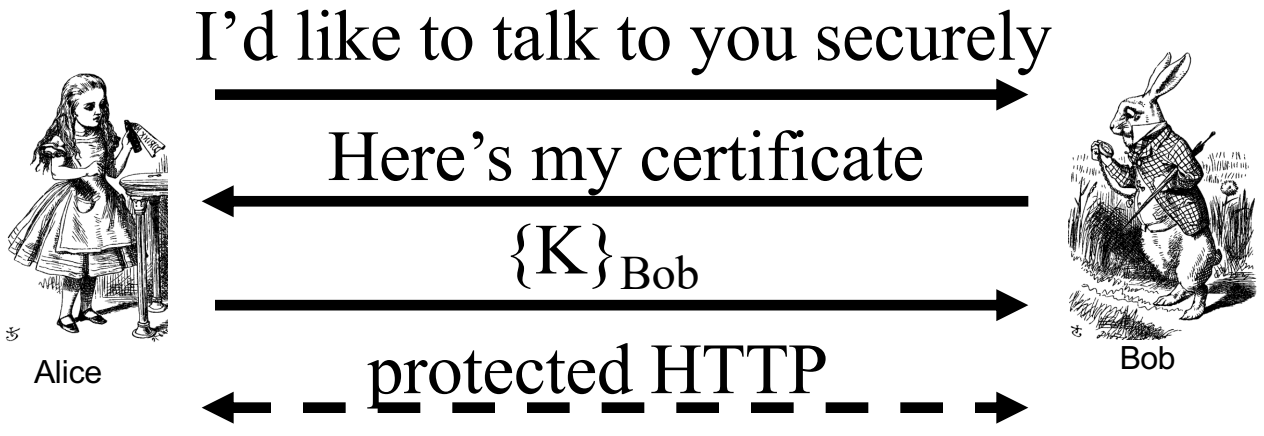
US government mandates SSL v3 can't be used for sensitive information

POODLE attack, Sept. 2014

What is SSL?

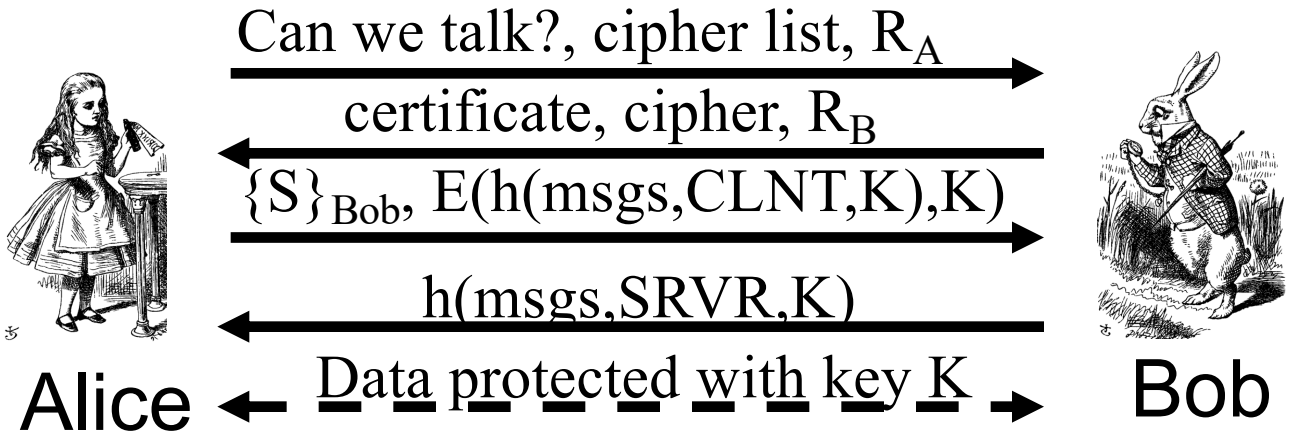
- SSL is the protocol used for majority of secure transactions on the Internet
- For example, if you want to buy a book at amazon.com…
  - You want to be sure you are dealing with Amazon (**authentication**)
  - Your credit card information must be protected in transit (**confidentiality** and/or **integrity**)
  - As long as you have money, Amazon does not care who you are
  - So, **no need for mutual authentication**

Simple SSL-like Protocol



- Is Alice sure she's talking to Bob?
  - Only through the encryption/decryption
  - Anyone can have Bob's certificate
- Is Bob sure he's talking to Alice?
  - No. But again, Bob(server) don't care

Simplified SSL Protocol



- S is known as **pre-master secret,** randomly generated
- $K = h(S, R_A, R_B)$
- "msgs" means all previous messages
- CLNT and SRVR are constants string

Simplified SSL Protocol

Can we talk?, cipher list, R_A

certificate, cipher, R_B

$\{S\}_{Bob}$, E(h(msgs,CLNT,K),K)

h(msgs,SRVR,K)

Data protected with key K

Alice

Bob

- S is known as **pre-master secret,** randomly generated
- $K = h(S, R_A, R_B)$
- "msgs" means all previous messages
- CLNT and SRVR are constants

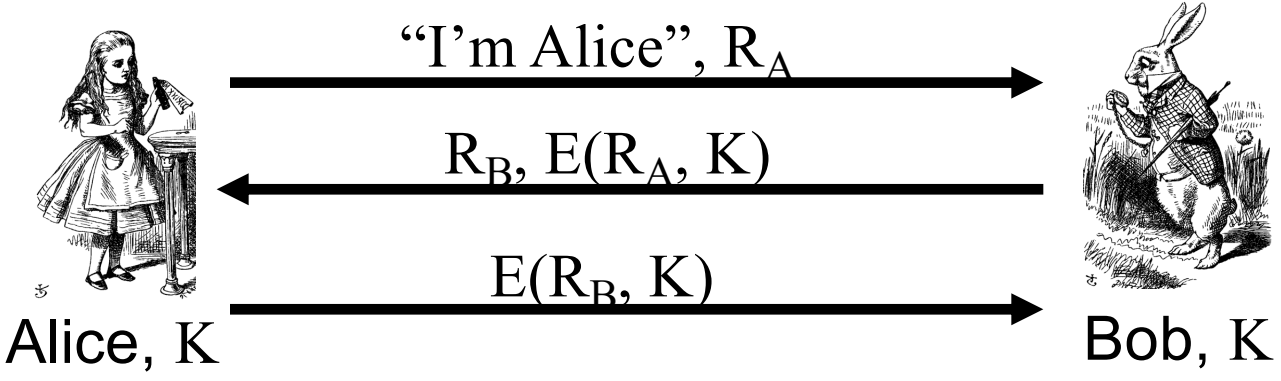
Q: Why is h(msgs, CLNT, K) encrypted?

A: Apparently, it adds no security…

SSL Keys

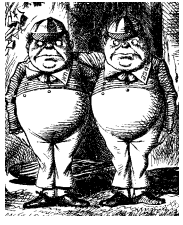
- **6 "keys" derived from $K = h(S, R_A, R_B)$**
  - 2 encryption keys: send and receive
  - 2 integrity keys: send and receive
  - 2 IVs: send and receive
  - **Why different keys in each direction?**

Recall: Mutual Authentication



- This provides mutual authentication…
- …or does it? See the next slide

Recall: Mutual Authentication Attack



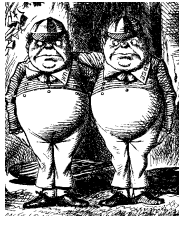
Trudy

1. "I'm Alice", R_A

2. R_B, $E(R_A, K)$

5. $E(R_B, K)$

Bob, K



Trudy

3. "I'm Alice", R_B

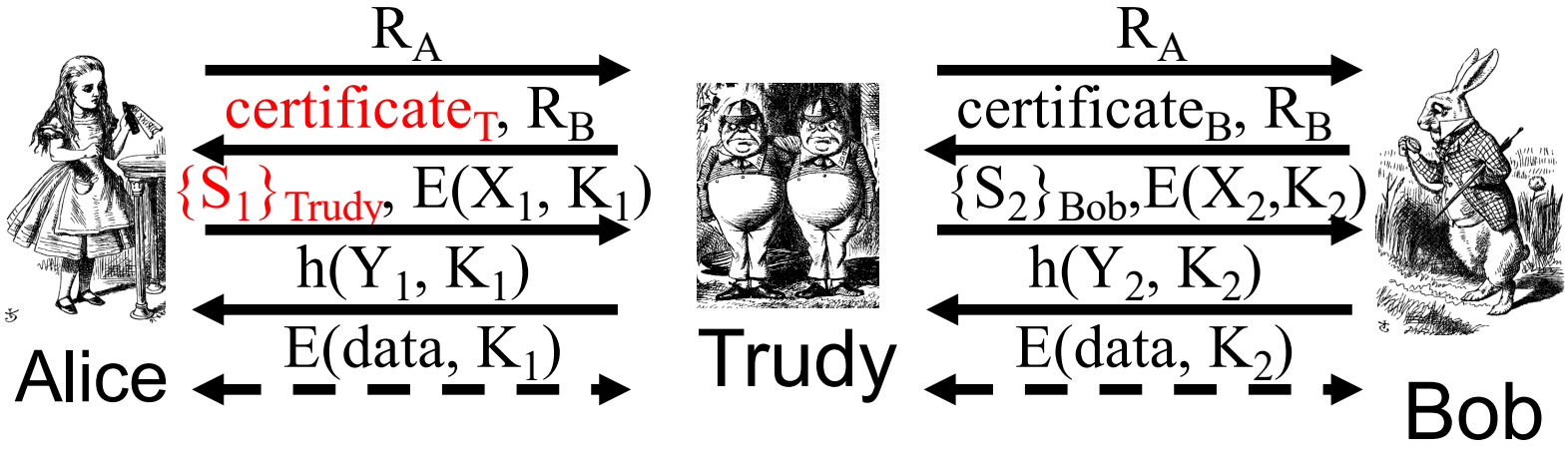
4. R_C, $E(R_B, K)$

Bob, K

SSL Authentication

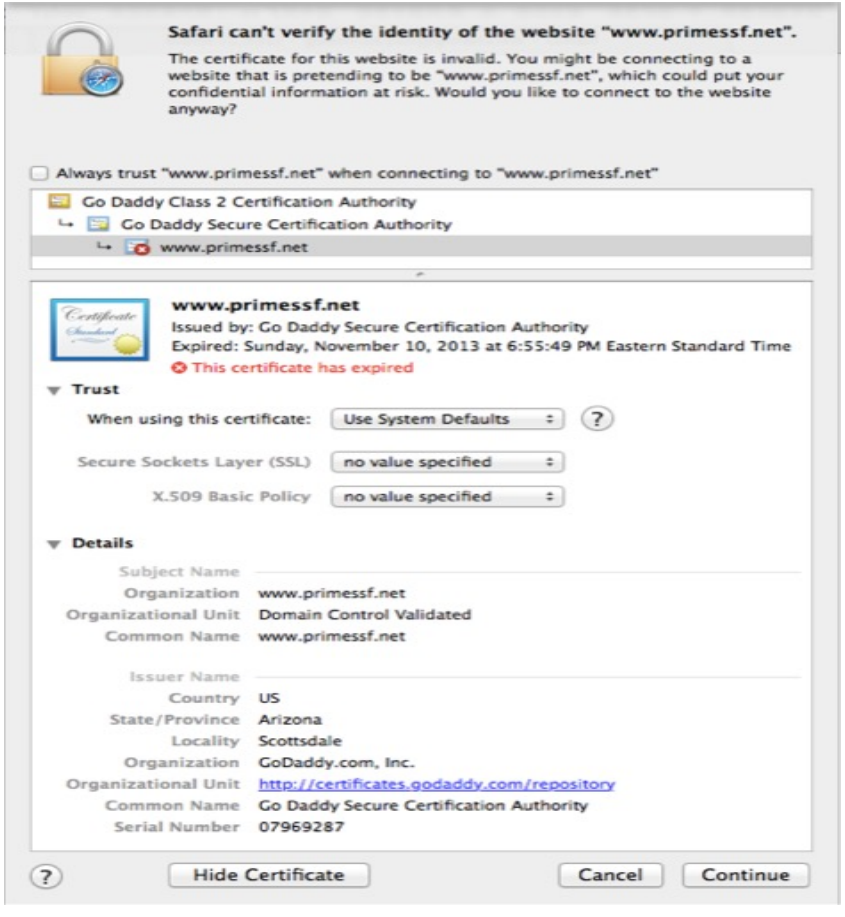
- Alice authenticates Bob, not vice-versa
  - How does client authenticate server?
    - Use public key pairs, certificate
  - Why would server not authenticate client?
- Mutual authentication is possible: Bob sends **certificate request** in message 2
  - Then client must have a valid certificate
  - But, if server wants to authenticate client, server could instead require password, authenticate outside the scope of SSL

SSL MiM Attack?



- **Q:** What prevents this MiM "attack"?
- **A:** Bob's certificate must be signed by a certificate authority (CA)
- What does browser do if signature not valid?
- What does user do when browser complains?

What happens with an invalid certificate?



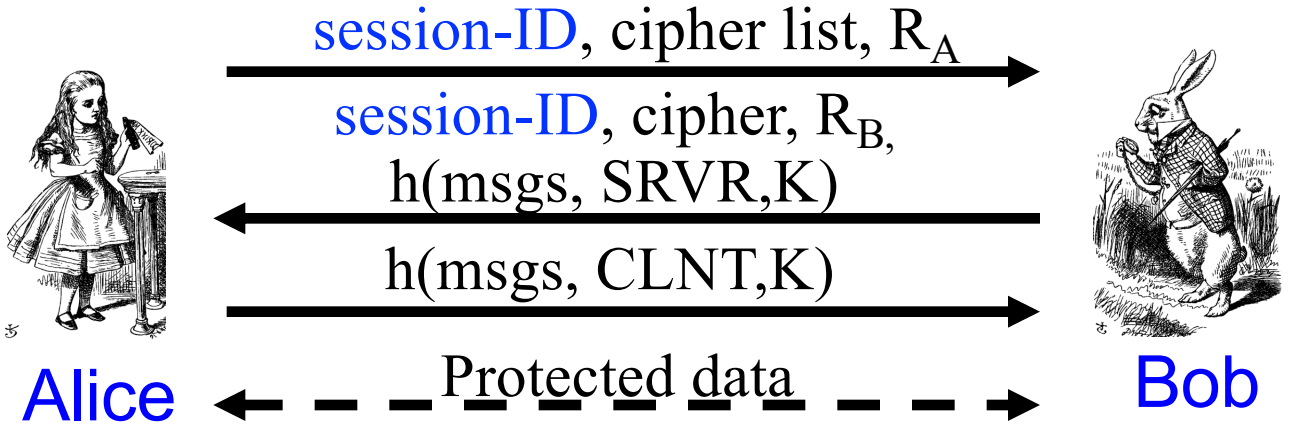
Browser will check the certificate.
Provides a warning.
But, most user ignore the warning.
Opens the door of MiMA!

https://www.primessf.net

SSL Sessions vs Connections

- SSL **session** is established as shown on previous slides
- SSL designed for use with HTTP 1.0
- HTTP 1.0 often opens multiple simultaneous (parallel) **connections**
  - Multiple connections per session
  - Improve performance
- SSL session is costly, public key operations
  - Encryption/decryption needed
- SSL has an **efficient** protocol for opening new connections ***given an existing session***

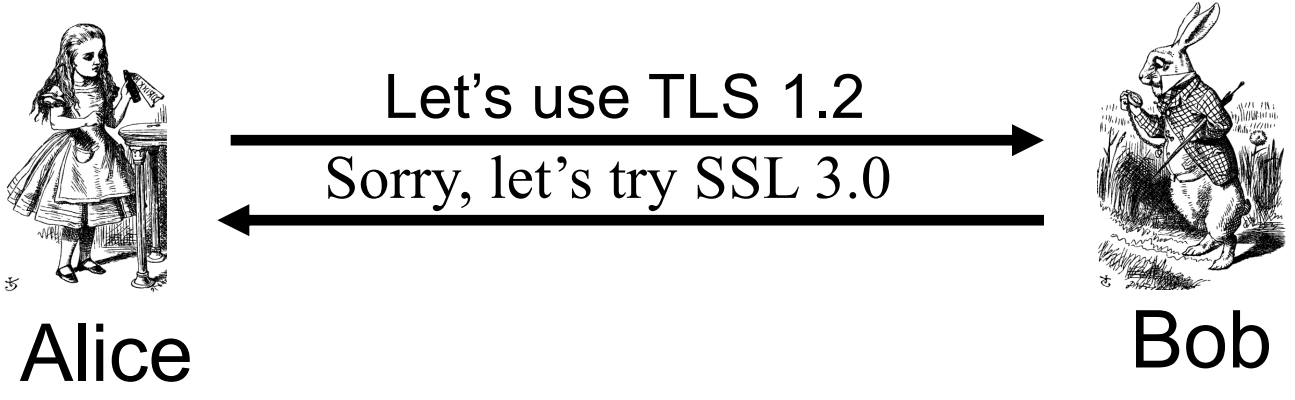
SSL Connection

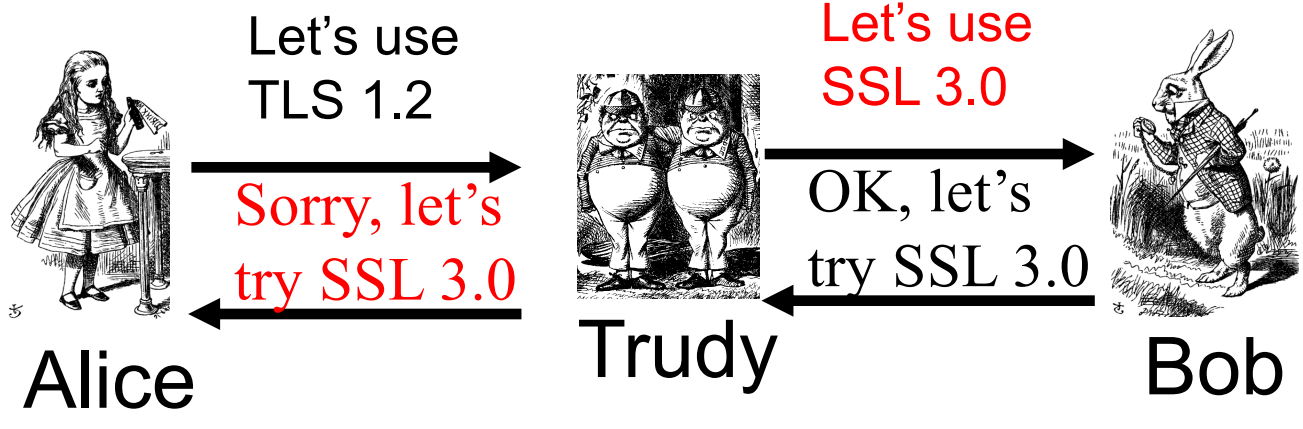


- Assuming SSL **session** exists
- So, S is already known to Alice and Bob
- Both sides must remember **session-ID**
- Again, $K = h(S, R_A, R_B)$

- No public key operations! (relies on known S)

TLS downgrade dance





SSL 3.0 POODLE Attack

- A severe problem of the CBC encryption in SSL 3.0 is that its block cipher padding is not deterministic and not covered by the MAC (Message Authentication Code)
- In a web setting, the weakness can be exploited by a man-in-the-middle attacker to decrypt “secure” HTTP cookie

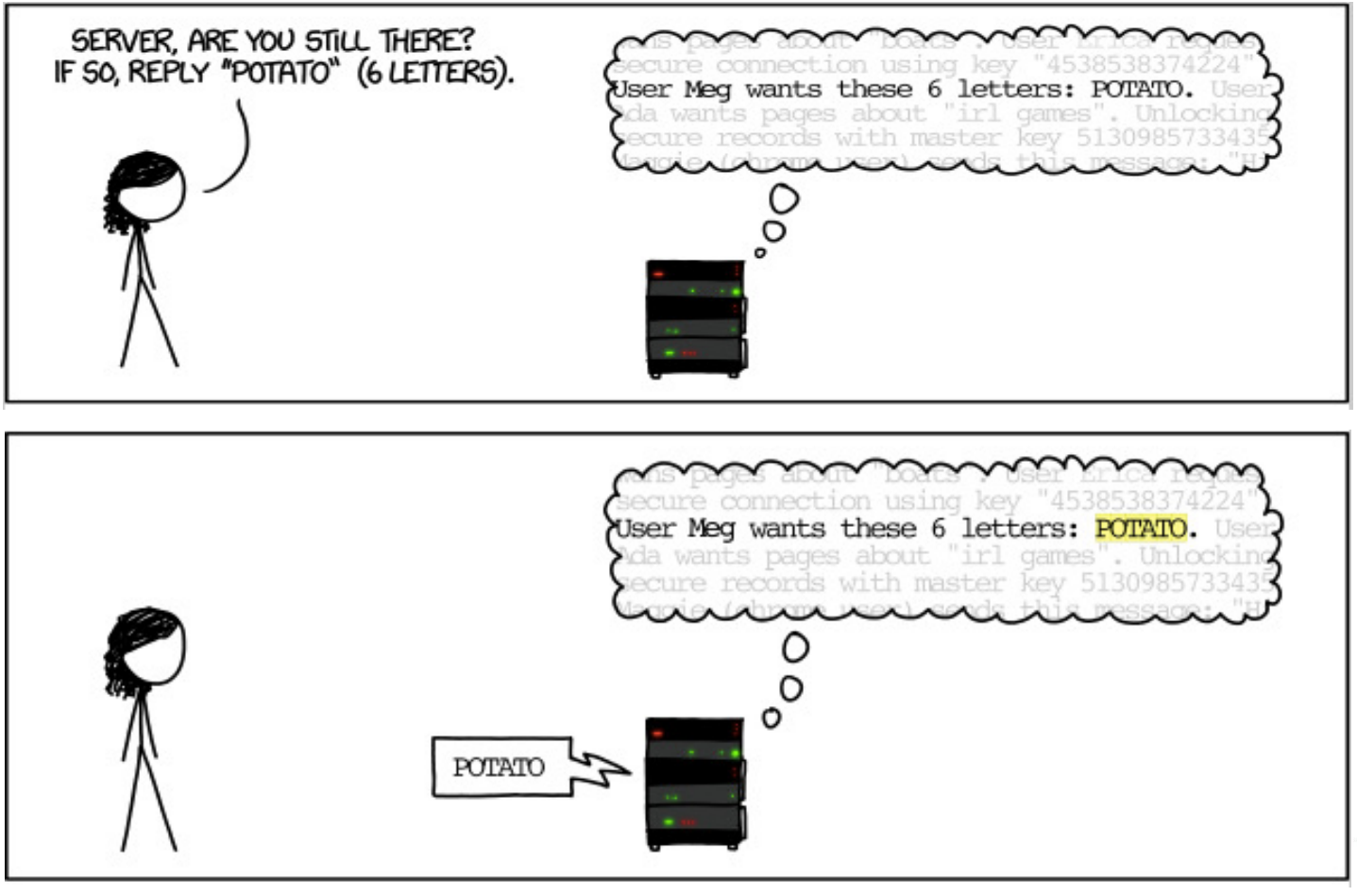
- **Downgrade to 3.0**
- **Intercepting the data**
- **Manipulating the cipher text**
  - Modifies the ciphertext block, especially the last few bytes that include padding
- **Analyzing the response**
  - Server accept -> Last padding guessed right
  - Use this information to deduce the value of the adjacent encrypted bytes
- **Repeat** till decrypt the the encrypted data
- Poodle attack is a **protocol flaw**

Heartbleed

Heartbeat Extension for TLS

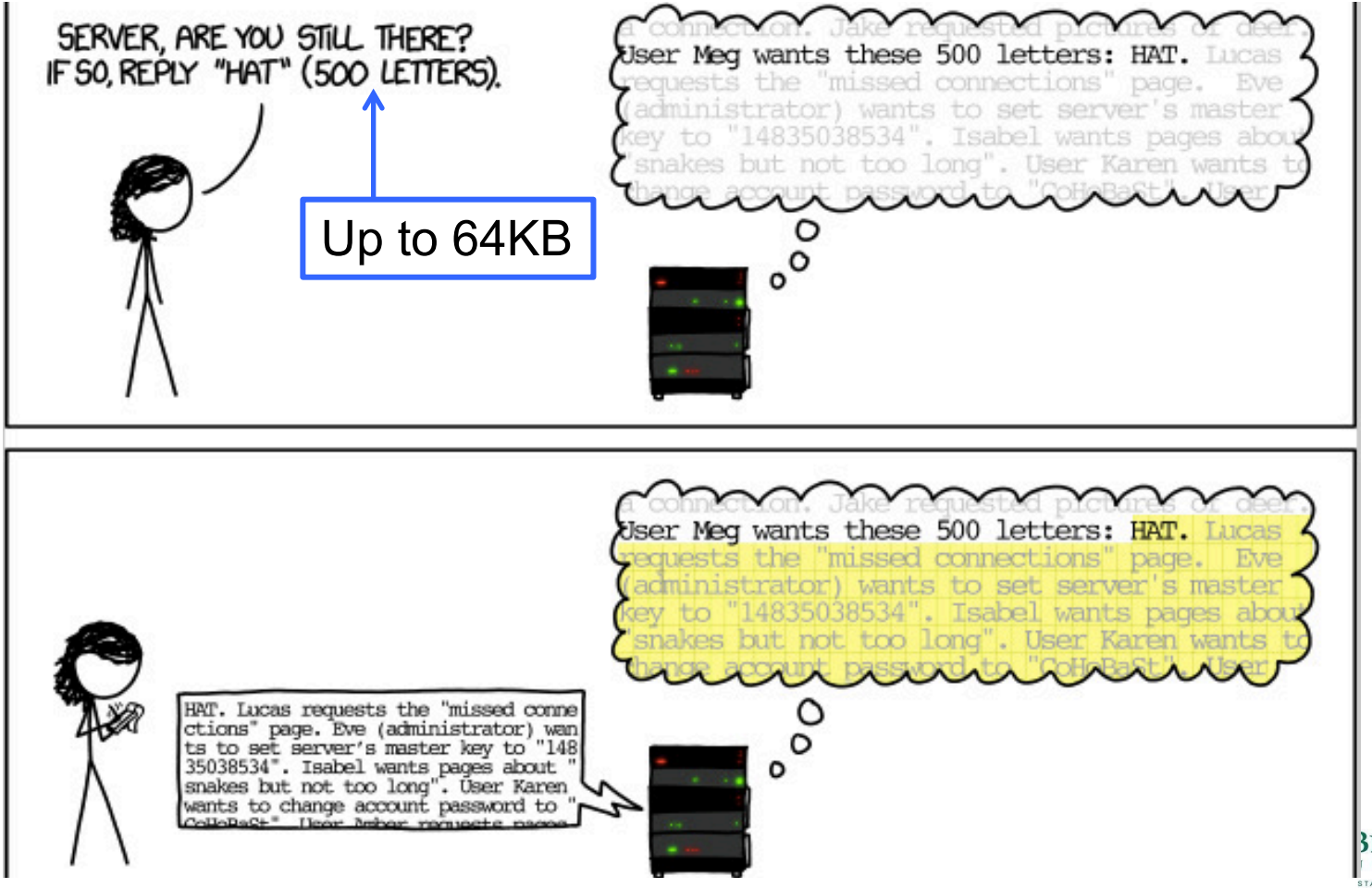
- Used to test and keep alive secure communication links without the need to renegotiate the connection each time
- Proposed as a standard in February 2012
- Implemented in OpenSSL by Robin Seggelmann
- Code reviewed by Stephen N. Henson, one of OpenSSL's four core developers, and introduced into OpenSSL's source code repository on December 31, 2011

How does Heartbeat work?



How does Heartbeat work?

How does Heartbleed work?



Is Heartbleed a virus?

- Absolutely NO, It's not a virus
- The Heartbleed **bug** is a **vulnerability** resided in TLS heartbeat mechanism built into certain versions of the popular open source encryption standard OpenSSL, a popular version of the Transport Layer Security (TLS) protocol

Heartbleed Bug

```
/* Code from OpenSSL */
    …
    unsigned int payload;
    n2s(p, payload); // payload value provided by client
    …
    if (hbtype == TLS1_HB_REQUEST)
    {
        …
        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
        …
        s2n(payload, bp);
        …
        memcpy(buffer, pl, payload); // The critical bug. No check on payload value
        …
        /* Send it back to the client */
        dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
    }
    /* ... additional code ... */
}
```

Specific systems affected

- Cisco systems has identified 78 of its products as vulnerable, including IP phone systems and video conferencing systems

- Many websites and online services, such as Yahoo!, Stack Overflow, Amazon, Akamai, Github, Reddit, Pinterest, SourceForge, Tumblr, Wikipedia, …

Is it a client side or server side vulnerability?

- TLS heartbeats can be sent by either side of a TLS connection, so it can be used to attack clients as well as servers

- An attacker can obtain **up to 64K memory** from the server or client as well that uses an OpenSSL implementation vulnerable to Heartbleed (*CVE-2014-0160*).

Heartbleed relies on man-in-the-middle attack?

- No, it has nothing to deal with a Man-in-the-Middle (MitM) attack. But using Heartbleed attack, one can manage to **obtain the private encryption key** for an SSL/TLS certificate and could set up a fake website that passes the security verification

- An attacker could also decrypt the traffic passing between a client and a server, i.e., perfect man-in-the-middle attack on HTTPS connection

How Heartbleed affect smartphone?

- All versions of Android OS include outdated versions of OpenSSL library, but only Android 4.1.1 Jelly Bean has the vulnerable heartbeat feature enabled by default. Blackberry also confirmed that some of its products were vulnerable to Heartbleed bug, whereas Apple's iOS devices were not affected by OpenSSL flaw.

- IP phones, Routers, Medical devices, Smart TV sets, embedded devices and millions of other devices that rely on the OpenSSL to provide secure communications could also be vulnerable to Heartbleed bug.