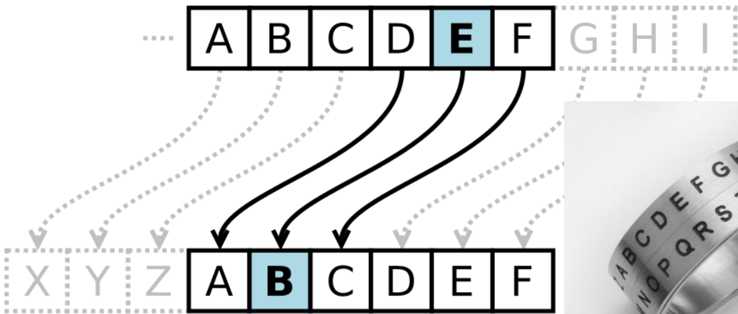
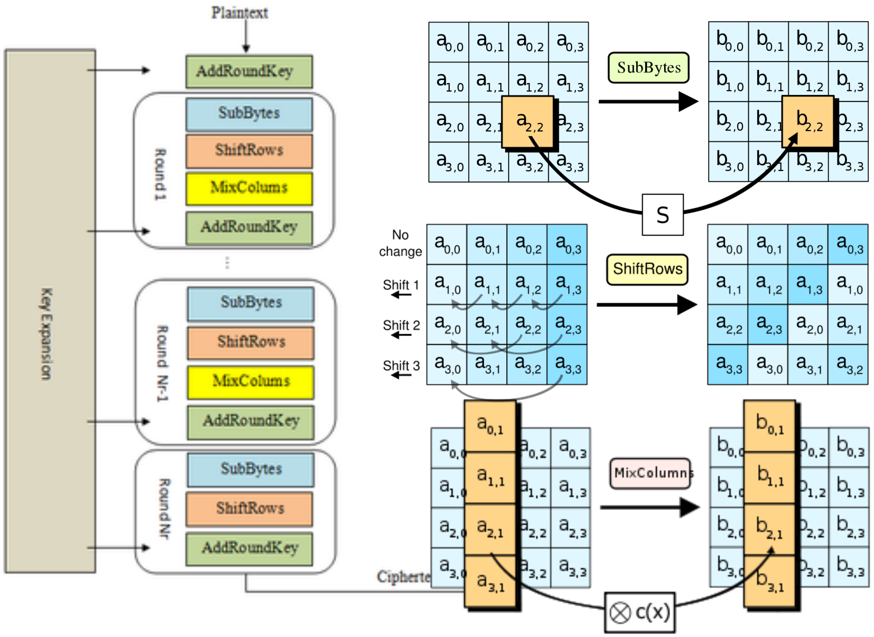
Lecture 9: Secure Channels

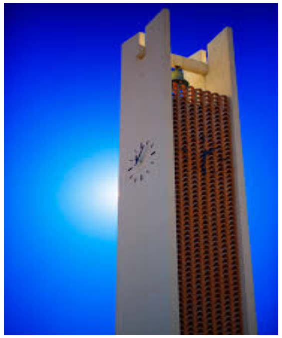
CS 181S

Fall 2020

Crypto Thus Far…



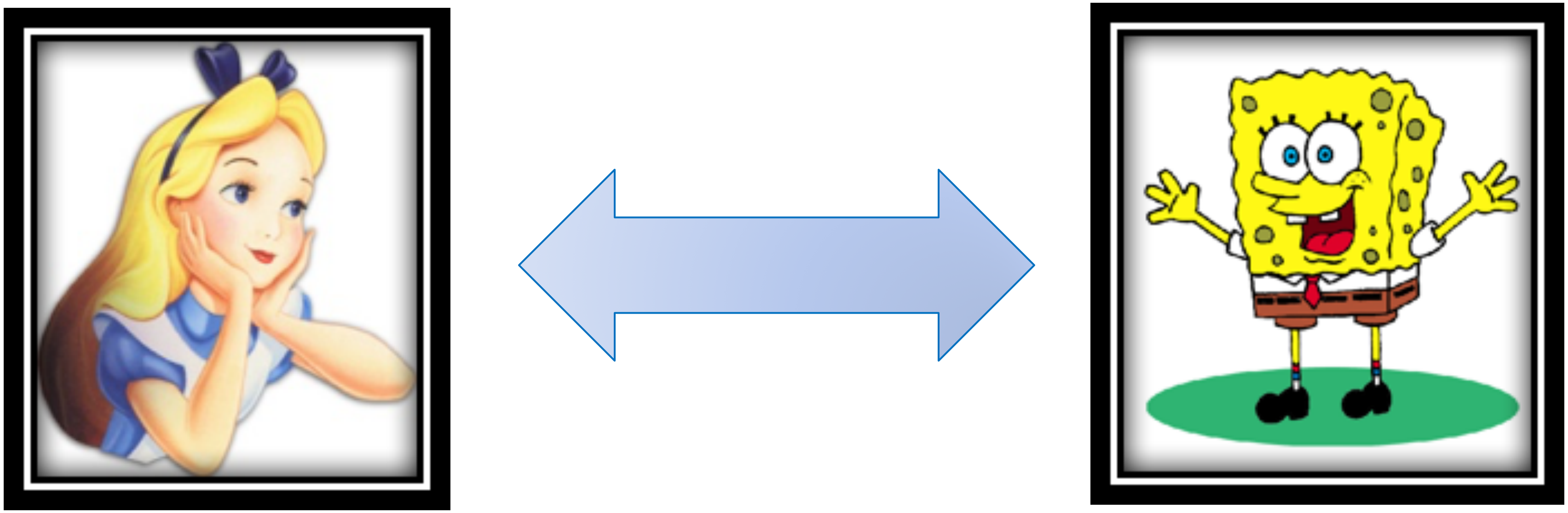




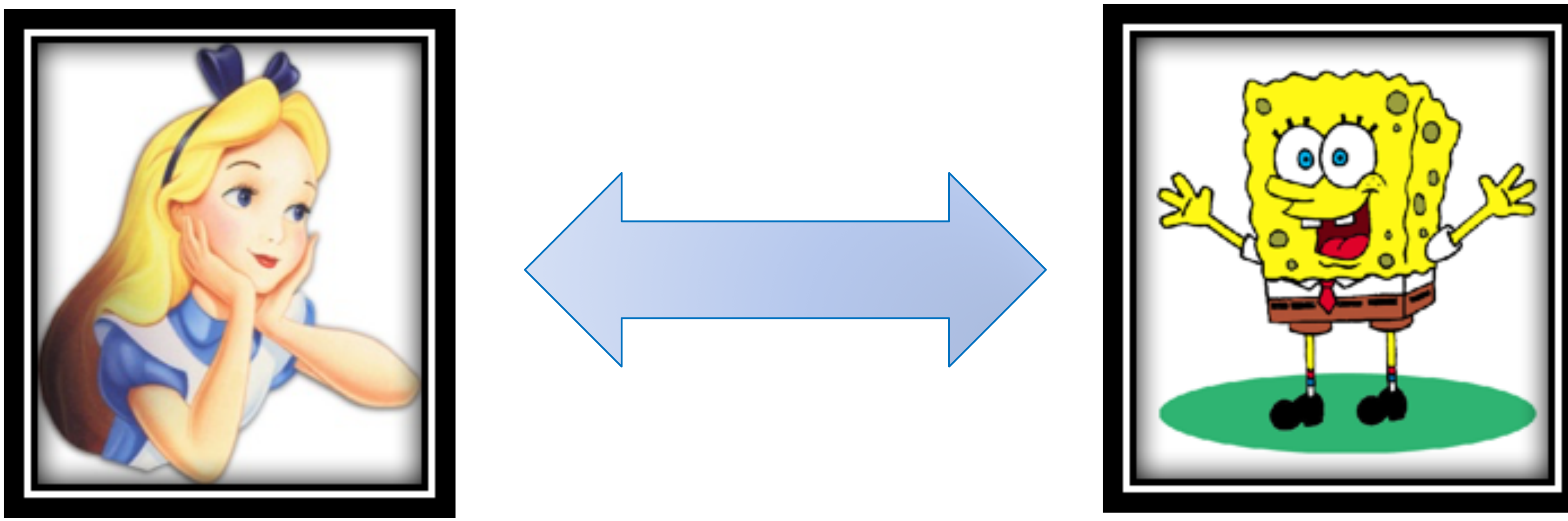
Today: Secure Channels

- **Threat:** attacker who controls the network
  - Dolev-Yao model: attacker can read, modify, delete messages
- **Vulnerability:** communication channel between sender and receiver can be controlled by other principals
- **Harm:** conversation can be learned (violating confidentiality) or changed (violating integrity) by attacker
- **Countermeasure:** all the crypto…

Today: Secure Channels



Today: Secure Channels



Requirements:

1) Channel must provide both confidentiality and integrity
2) A and B must agree on session key(s)
3) A and B must agree on cipher suite (crypto protocols, encryption mode, key lengths)
4) Must detecting missing messages & repeated messages
5) Must maintain connection (and be able to end it)

Authenticated encryption

- Traditionally: MAC-then-encrypt
- Now: block cipher modes designed to provide confidentiality and integrity (e.g., GCM)

Today: Secure Channels

Requirements:

1) Channel must provide both confidentiality and integrity
2) A and B must agree on session key(s)
3) A and B must agree on cipher suite (crypto protocols, encryption mode, key lengths)
4) Must detecting missing messages & repeated messages
5) Must maintain connection (and be able to end it)

Agreeing on a session key

Hybrid Encryption (RSA)

Diffie-Hellman

- `A -> B: g, p, g^a mod p`
- `B -> A: g^b mod p`
- `B: k_s := (g^a)^b mod p`
- `A: k_s := (g^b)^a mod p`

- DH, ECDH

Exercise 1: DH Key Agreement

- Assume that Alice chooses a=13 and sends Bob the message (5, 47, 43)
- Assume that Bob then chooses b=21 and sends Alice the message 15

1. What secret key will be generated by Bob?
2. What secret key will be generated by Alice?

Exercise 1: DH Key Agreement

- Assume that Alice chooses a=13 and sends Bob the message (5, 47, 43)
- Assume that Bob then chooses b=21 and sends Alice the message 15

1. What secret key will be generated by Bob?

   $$\mathbf{43^{21} \bmod 47 = 44}$$

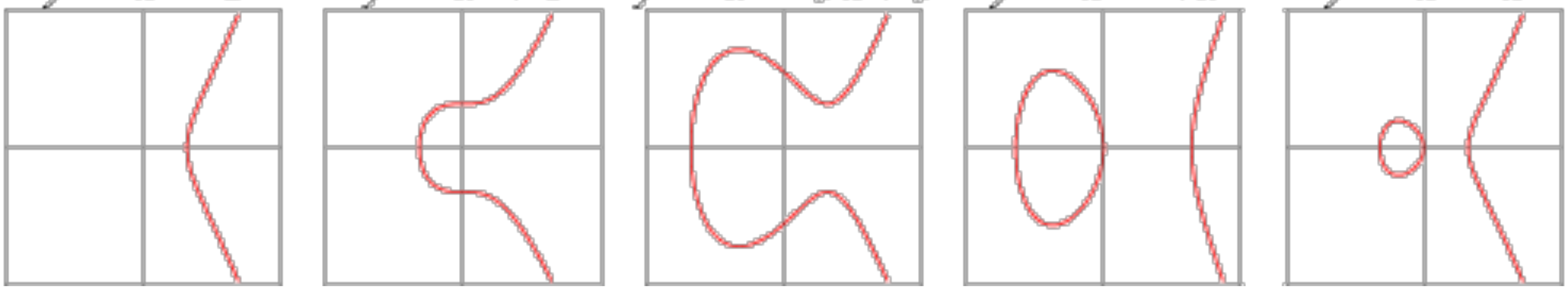
2. What secret key will be generated by Alice?

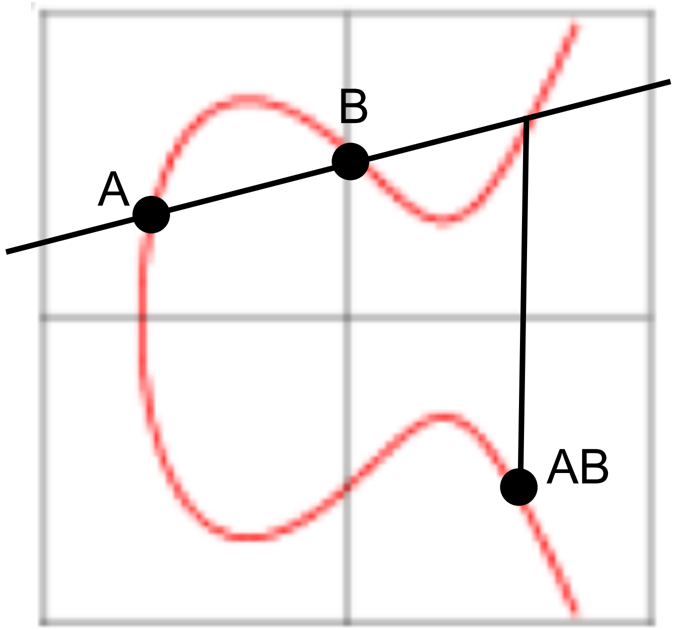
   $$\mathbf{15^{13} \bmod 47 = 44}$$

Elliptic Curves

- $y^2 = x^3 + ax + b$

$y^2 = x^3 - 1$ | $y^2 = x^3 + 1$ | $y^2 = x^3 - 3x + 3$ | $y^2 = x^3 - 4x$ | $y^2 = x^3 - x$





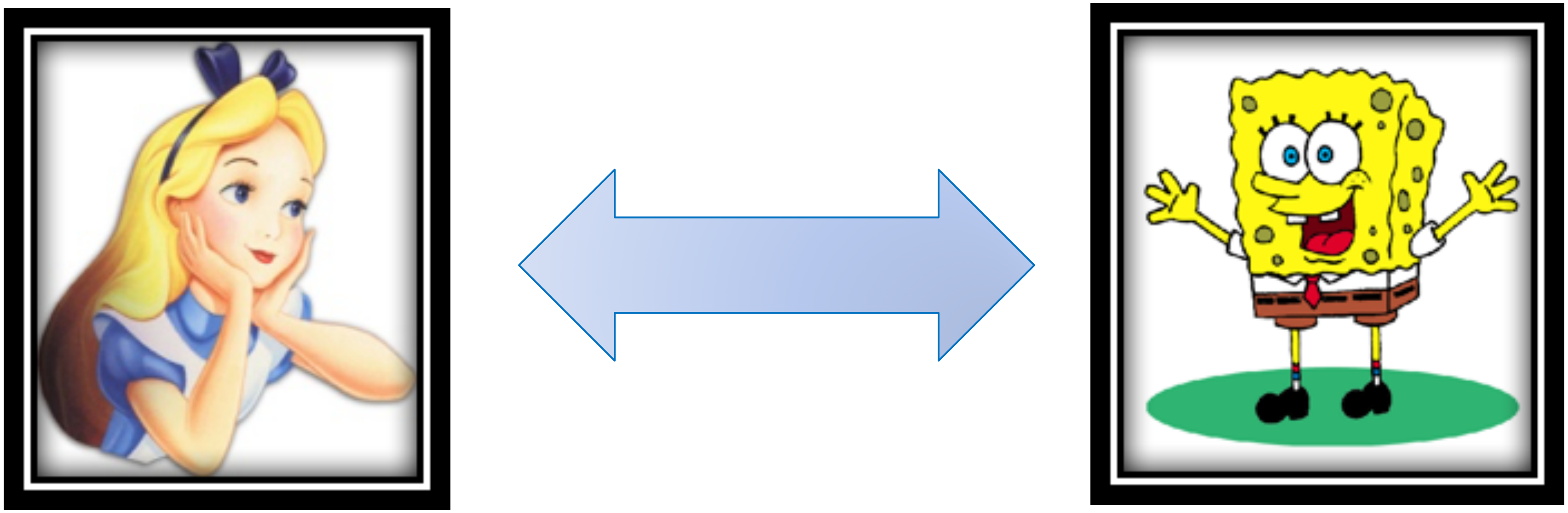
Key reuse

- **Principle:** every key in system should have unique purpose

- generate a fresh session key for every connection (**ephemeral**)

- Have one key: k_s, Need 2-4 keys:
- How to get many out of one: use a cryptographic hash function H to derive keys...
  1. kea = H(k, "Enc Alice to Bob")
  2. keb = H(k, "Enc Bob to Alice")
  3. kma = H(k, "MAC Alice to Bob")
  4. kmb = H(k, "MAC Bob to Alice")

Today: Secure Channels



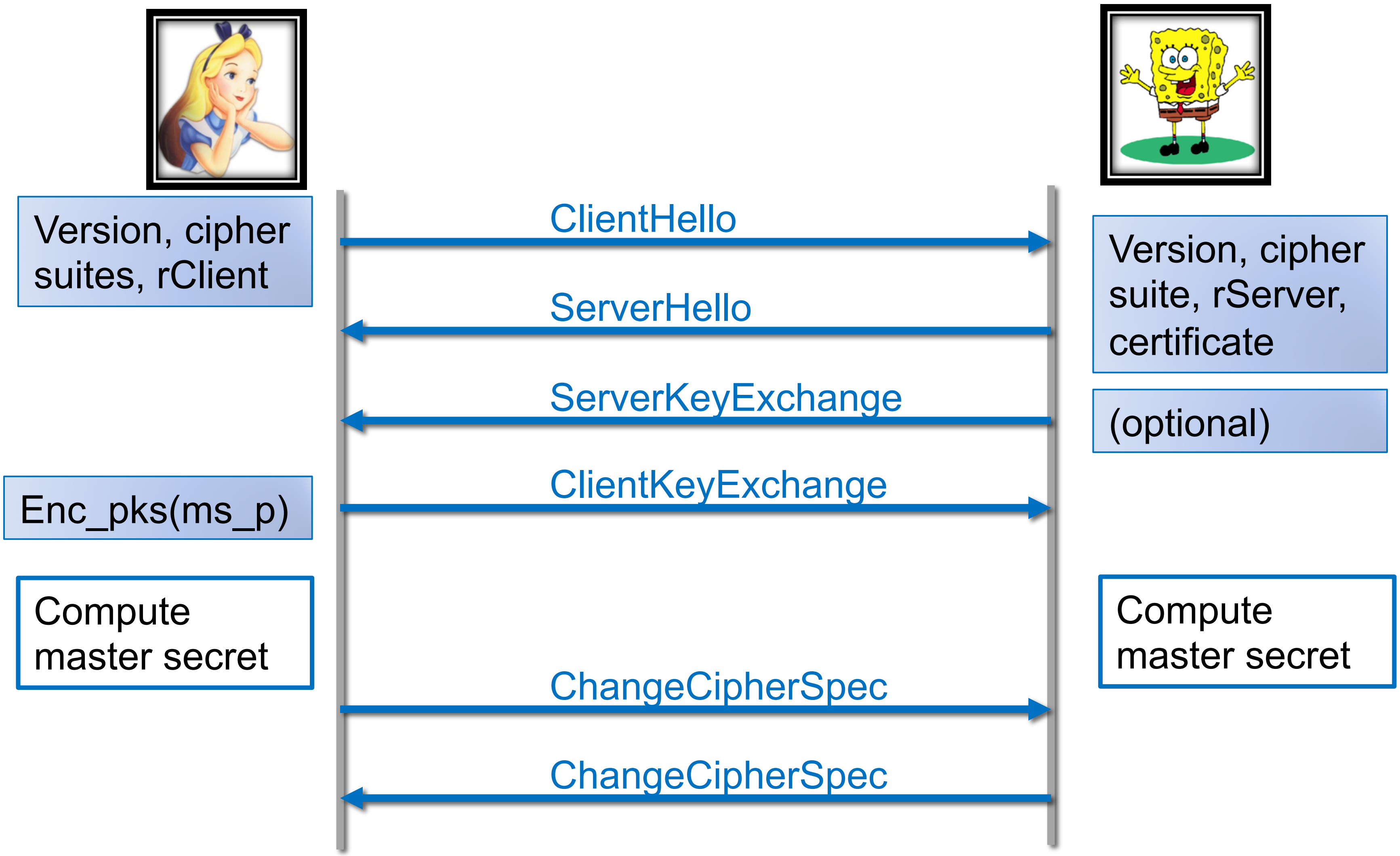
Requirements:

1) Channel must provide both confidentiality and integrity
2) A and B must agree on session key(s)
3) A and B must agree on cipher suite (crypto protocols, encryption mode, key lengths)
4) Must detecting missing messages & repeated messages
5) Must maintain connection (and be able to end it)

Secure Socket Layer (SSL)

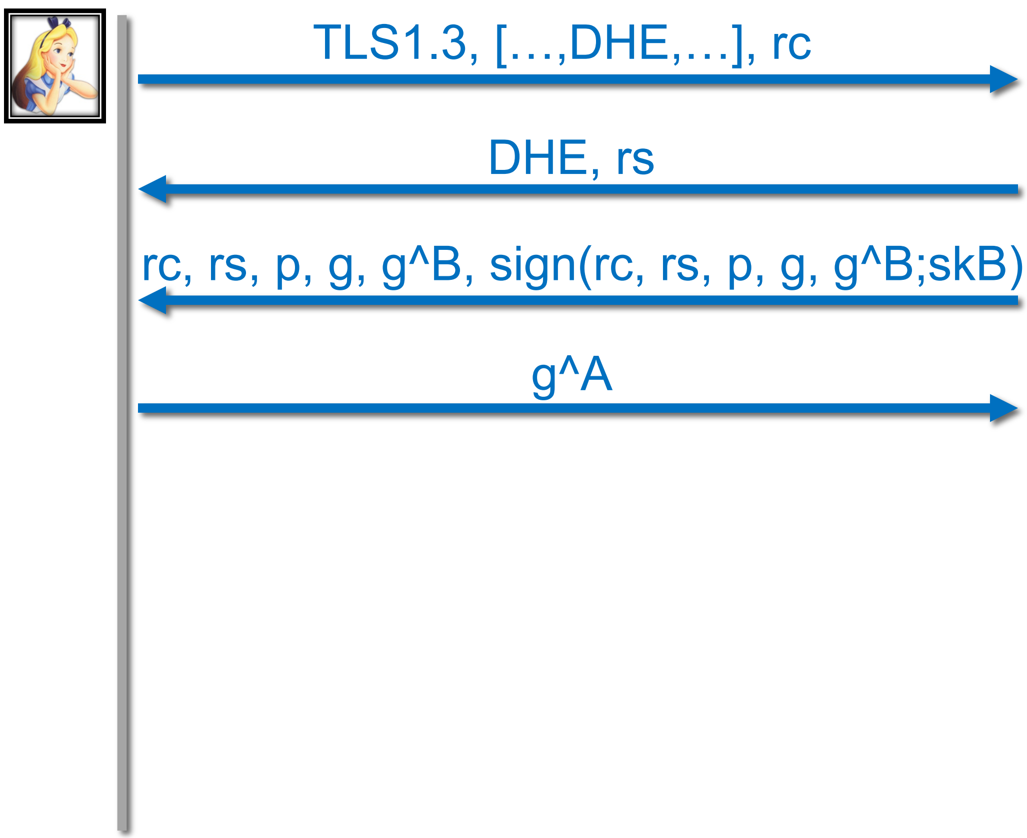
- SSL 2.0 (1995): designed by Netscape, contains a number of security flaws, prohibited since 2011
- SSL 3.0 (1996): complete re-design, all accepted cipher suites now have known vulnerabilities, prohibited since 2015
- TLS 1.0 (1999): contains known vulnerabilities, suggested migration by June 2018
- TLS 1.1 (2006): update with significant changes in how IVs/padding are handled to prevent known attacks
- TLS 1.2 (2008): update with modern cipher suites
- TLS 1.3 (2018): drops insecure features and introduces additional cipher suites

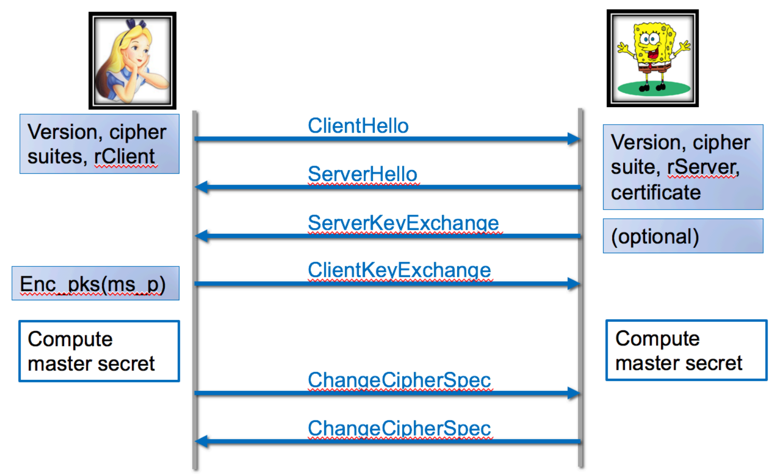
SSL/TLS Handshake



Exercise 2: TLS Handshake

- What messages would be exchanged in the initial three-way handshake if the principals elected to use DH instead of hybrid encryption to agree on a message?





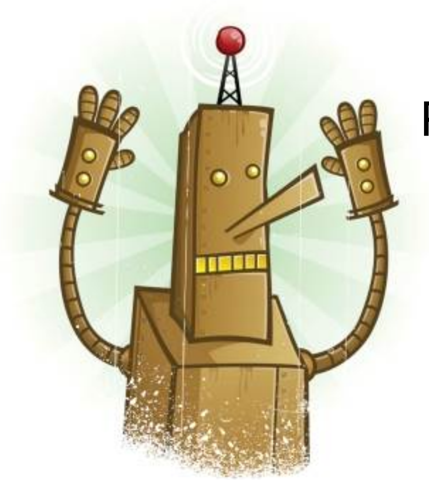
Supported Cipher Suites

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes
DH-DSS	No	Yes	Yes	Yes	Yes	No
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[42]
ECDH-ECDSA	No	No	Yes	Yes	Yes	No
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

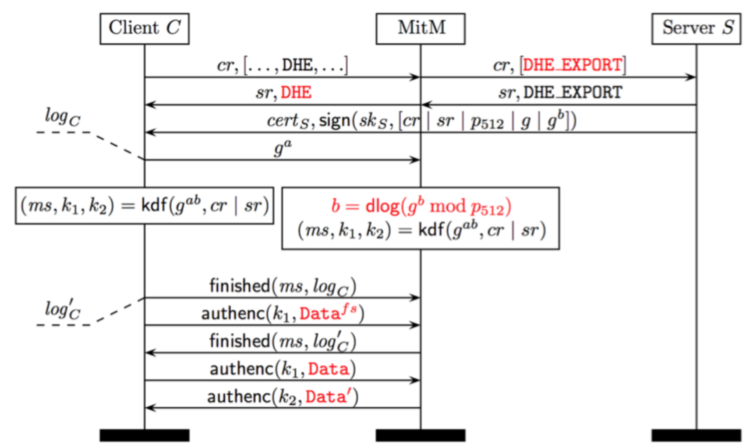
Cipher			Protocol version					
Type	Algorithm	Nominal strength (bits)	SSL 2.0	SSL 3.0 [n 1][n 2][n 3][n 4]	TLS 1.0 [n 1][n 3]	TLS 1.1 [n 1]	TLS 1.2 [n 1]	TLS 1.3
Block cipher with mode of operation	AES GCM[44][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	Secure
	AES CCM[45][n 5]		N/A	N/A	N/A	N/A	Secure	Secure
	AES CBC[n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	Camellia GCM[46][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A
	Camellia CBC[47][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	ARIA GCM[48][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A
	ARIA CBC[48][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	SEED CBC[49][n 6]	128	N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	N/A
	GOST 28147-89 CNT[43][n 7]	256	N/A	N/A	Insecure	Insecure	Insecure	N/A
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	N/A	N/A
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A
Stream cipher	ChaCha20-Poly1305[54][n 5]	256	N/A	N/A	N/A	N/A	Secure	Secure
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A

Attacks on Cipher Negotiation

Padding Oracle On Downgraded Legacy Encryption (POODLE)

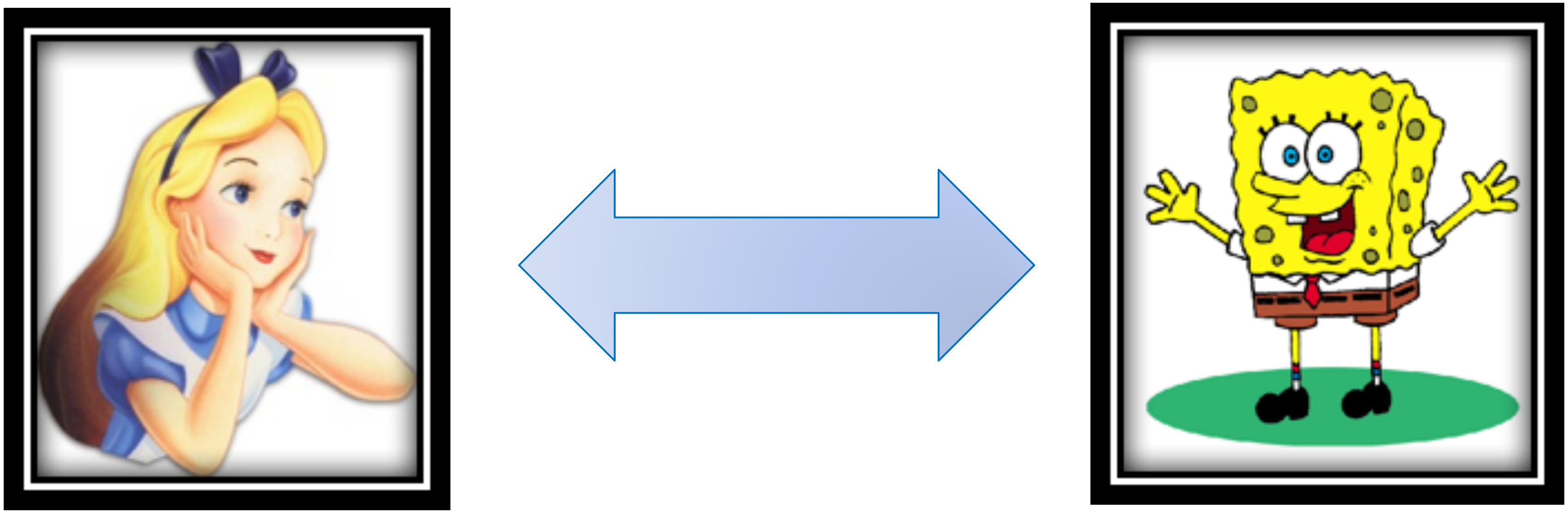


Return of Beichenbacher's Oracle Threat (ROBOT)



Logjam

Today: Secure Channels



Requirements:

1) Channel must provide both confidentiality and integrity
2) A and B must agree on session key(s)
3) A and B must agree on cipher suite (crypto protocols, encryption mode, key lengths)
4) Must detecting missing messages & repeated messages
5) Must maintain connection (and be able to end it)

Message numbers

- Every message that Alice sends is numbered
  - 1, 2, 3, ...
  - numbers increase monotonically
  - never reuse a number
- Bob keeps state to remember last message number he received
- Bob accepts only increasing message numbers
- And ditto all the above, for Bob sending to Alice
  - so each principal keeps two independent counters: messages sent, messages received

Message numbers

What if Bob detects a gap? e.g. 1, 2, 5

- Maybe Mallory deleted messages 3 and 4 from network
- Maybe Mallory detectably changed 3 and 4, causing Bob to discard them
- In either case, channel is under active attack
  - Absent availability goals, time to PANIC: abort protocol, produce appropriate information for later auditing, shut down channel

What if network non-maliciously dropped messages or will deliver them later?

- Let's assume underlying transport protocol guarantees that won't happen (e.g. TCP)

Message numbers

- Message number usually implemented as a fixed-size unsigned integer, e.g., 32 or 48 or 64 bits
- What if that `int` overflows and wraps back around to 0?
  - Message number **must** be unique within conversation to prevent Mallory from replaying old conversation
  - So conversation **must** stop at that point
  - Can start a new conversation with a new session key

Today: Secure Channels

Requirements:

1) Channel must provide both confidentiality and integrity
2) A and B must agree on session key(s)
3) A and B must agree on cipher suite (crypto protocols, encryption mode, key lengths)
4) Must detecting missing messages & repeated messages
5) Must maintain connection (and be able to end it)

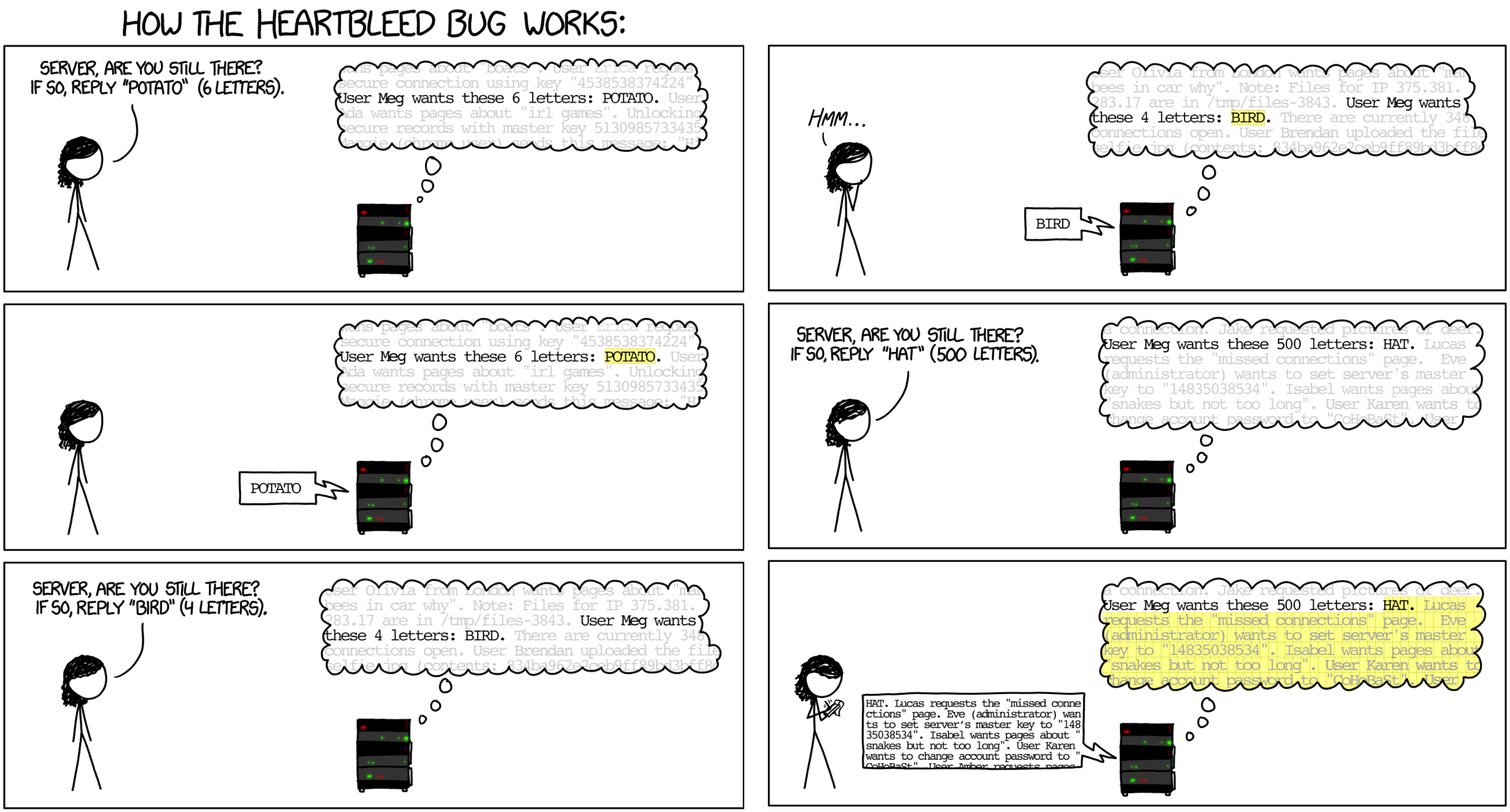
TLS record

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	Content type			
Bytes 1..4	Version		Length	
	(Major)	*(Minor)*	*(bits 15..8)*	*(bits 7..0)*
Bytes 5..(*m*–1)	Protocol message(s)			
Bytes *m*..(*p*–1)	MAC (optional)			
Bytes *p*..(*q*–1)	Padding (block ciphers only)			

Hex	Dec	Type
0x14	20	ChangeCipherSpec
0x15	21	Alert
0x16	22	Handshake
0x17	23	Application
0x18	24	Heartbeat

Heartbleed

Heartbeat



HOW THE HEARTBLEED BUG WORKS:
SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).
User Meg wants these 6 letters: POTATO.
POTATO
SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Meg wants these 4 letters: BIRD.
HMM...
BIRD
SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT.
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "

Truncation Attack

Exercise 3: Feedback

1. Rate how well you think this recorded lecture worked
    1. Better than an in-person class
    2. About as well as an in-person class
    3. Less well than an in-person class, but you still learned something
    4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have particular questions you would like me to address in this week's problem session?

4. Do you have any other comments or feedback?