Network Tunneling

Shadowsocks, AEAD, and Obfuscating

Problem Statement

- Alice and Bob want to send messages to each other

Problem Statement

- Alice and Bob want to send messages to each other
- Mallory ships messages for them

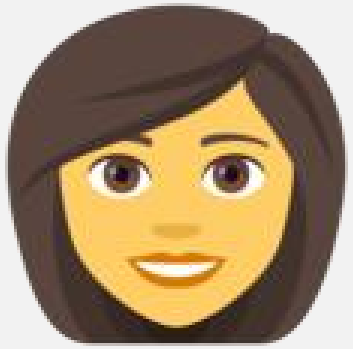
Problem Statement

- Alice and Bob want to send messages to each other
- Mallory ships messages for them
- But if Mallory can recognize what Alice and Bob are talking, he may cut the connection off

Mallory

- An active attacker
    - Eve = A passive attacker
- Man-in-the-middle
    - He can change/drop messages
    - He will send malicious messages
- Mallory as a “firewall”

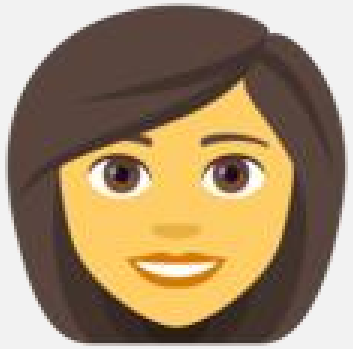
!@#$%^



Random Bytes

!@#$%^

Hello!

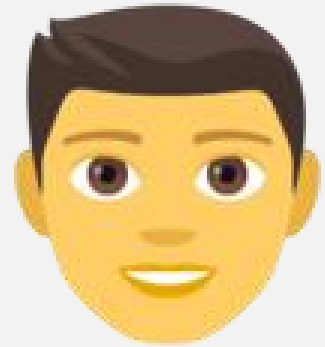


Recognizable Data

Hello!

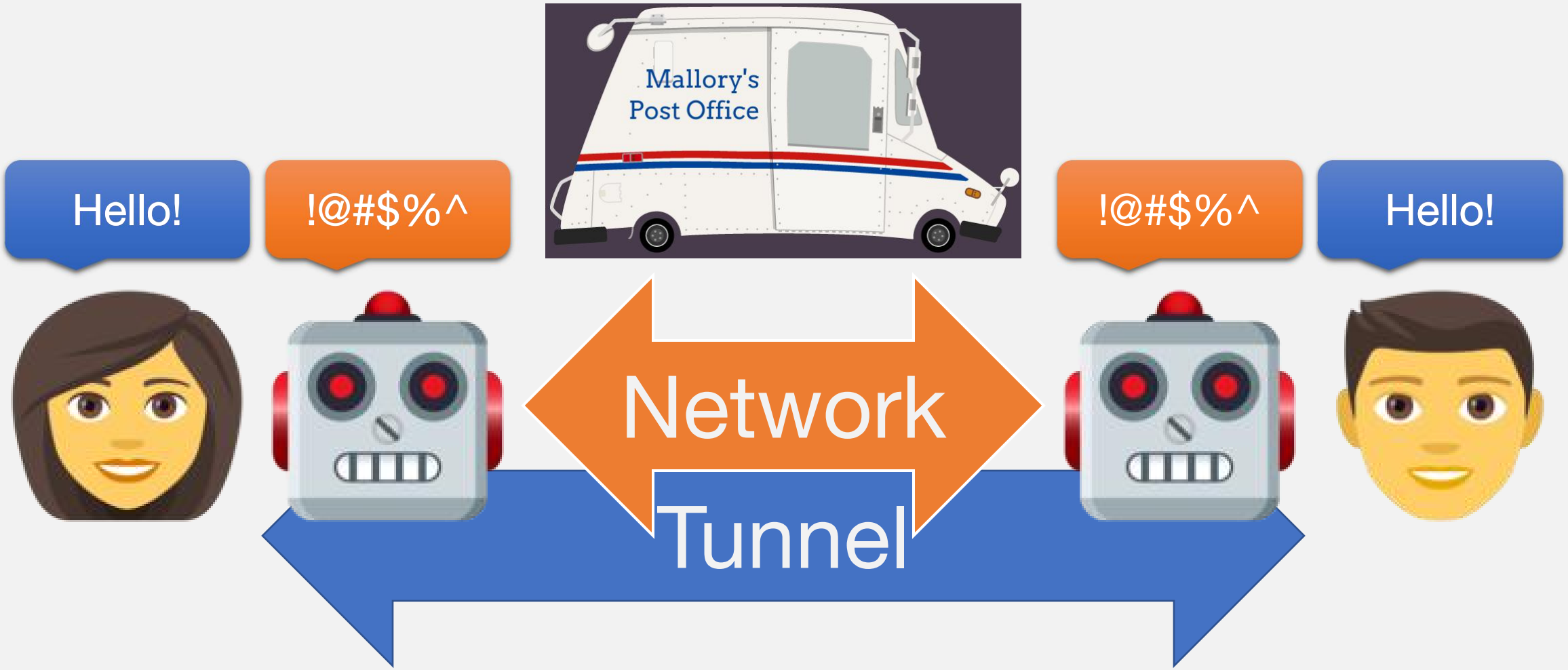
Hello!

Attack Data



Basic Solution

- Establish a tunnel
    - Capture all the messages
    - Send them with special protection
    - Restore the messages
- Encrypt the messages
    - Use pre-shared keys (PSK)


Mallory's
Post Office
Hello!
!@#$%^
!@#$%^
Hello!
Network
Tunnel

Shadowsocks

- An open-source encrypted proxy
    - Multiple implementations
    - Cross-platform
- Created by @clowwindy, 2012
    - Maintained by the community now
- Aimed at bypassing censorship

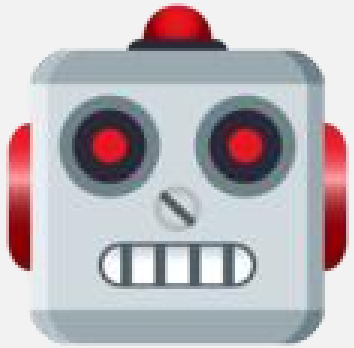
Shadowsocks

- Tunneled SOCKS5
    - Local SOCKS5 proxy server
    - Translating to SS protocol
    - Remote SS outbound server
- Stateless C/S architecture
    - Single user, multiple connections

What If...

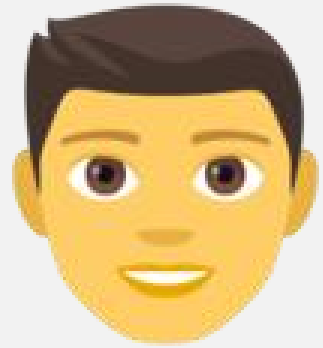
^%$#@!

^%$#@!



It's
Invalid

&*_+=\



But, What If...

Attack Scenario

Alice: “What would you like today?”
Bob: “2 donuts please.”

Encrypted:

Alice: “AAA BBB CCC DDD EEE”
Bob: “FFF GGG HHH”

Attack Scenario

Bob sends: “2 donuts please.”
Encrypted: “FFF GGG HHH”

Mallory ships: “FFF GGG HHH”
Alice receives: “2 donuts please.”

Attack Scenario

Bob sends: “2 donuts please.”
Encrypted: “FFF GGG HHH”

But, Mallory changes the message.

Attack Scenario

Bob sends: “2 donuts please.”
Encrypted: “FFF GGG HHH”

Mallory ships: “JJJ GGG HHH”
Alice receives: “1890236749 donuts please.”

The Weakness of Stream Cipher

Changing a byte in an encrypted message, the corresponding byte in the decrypted message will change. Although Mallory does not know the key, if he knows the byte structure of it, he can still construct an attack message (CCA).

Shadowsocks Classical

- Mallory himself can send messages
- Mallory knows the header format
    - There is a flag byte
    - The valid values are 1, 3, 4
- Mallory tries different values on the position of the flag byte
    - If 253/256 closed, it is SS!

Shadowsocks One-Time Auth (OTA)

- Alice sends a special header
- Bob decrypt the header and verify it
    - If it is correct, go ahead
    - Otherwise, cut off the connection

Shadowsocks One-Time Auth (OTA)

- Alice sends a special header
- Bob decrypt the header and verify it
    - If it is correct, go ahead
    - Otherwise, no response

Shadowsocks One-Time Auth (OTA)

- Alice sends a special header
- Bob decrypt the header and verify it
    - If it is correct, go ahead
    - Otherwise, no response
- But, the attack is still possible
    - It is similar to the donuts example

Shadowsocks One-Time Auth (OTA)

- Mallory can modify the message
- The structure of the header is known
    - There will be a “message length”
    - Certainly, Mallory can change it
- It will cause a detectable server error

Shadowsocks AEAD

- Alice hash each message
- Structure:
    - Message length
    - Hash of the message length
    - Message body
    - Hash of the message body

Shadowsocks AEAD

- Alice hash each message
- Structure:
    - Message length // Trusted now
    - Hash of the message length
    - Message body
    - Hash of the message body

Shadowsocks AEAD

- Alice hash each message
- Structure:
    - Message length // Trusted now
    - Hash of the message length
    - Message body // Trusted now
    - Hash of the message body

Obfuscating (ss-obfs)

- “Always send random bytes” can be detectable

Obfuscating (ss-obfs)

- “Always send random bytes” can be detectable
- Alice and Bob may encapsulate the messages into some form (HTTP, WebSocket, etc.)

Obfuscating (ss-obfs)

- “Always send random bytes” can be detectable
- Alice and Bob may encapsulate the messages into some form (HTTP, WebSocket, etc.)
- Two sides: Obfs itself may leave data feature