# Last Time

- Internet Application Security and Privacy
  - Network-layer security: VPN, IPSec
  - Transport-layer security and privacy: TLS / SSL

# This time

[Stinson, Shmatikov-Boneh]

- Internet Application Security and Privacy
  - Transport-layer security and privacy: Tor
  - Application-layer security and privacy: SSH, remailers, PGP/gpg, OTR

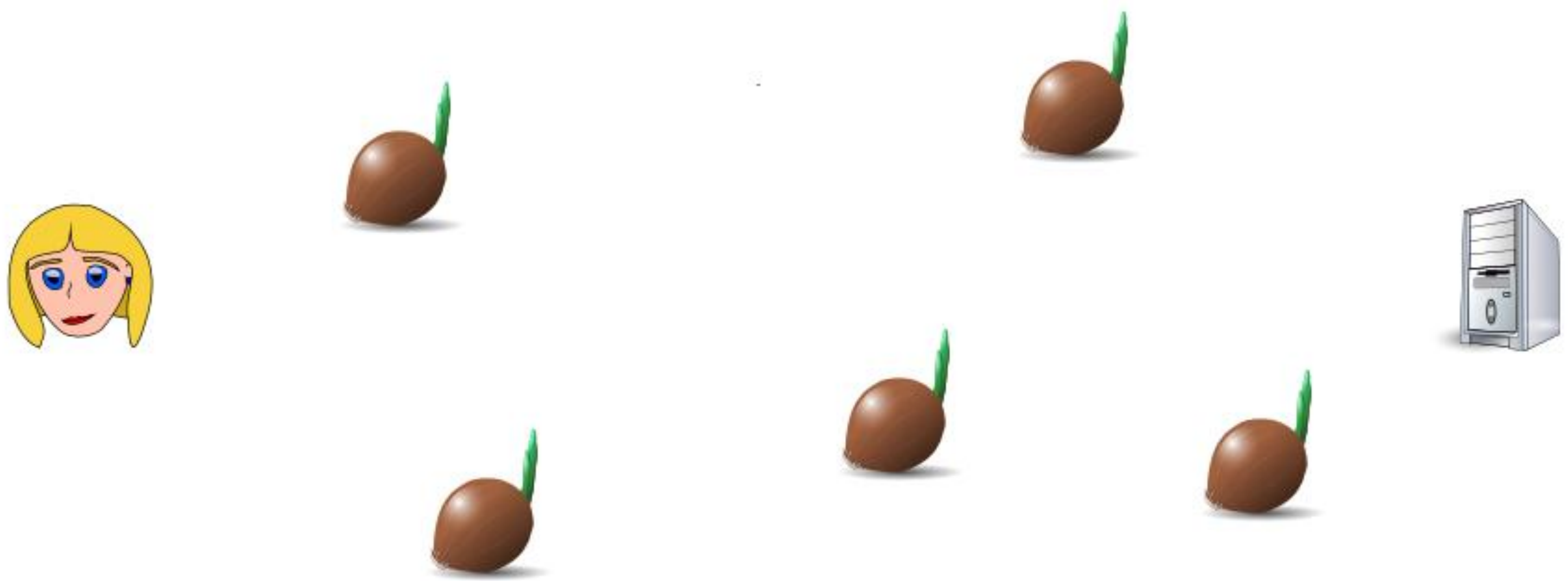
# Tor

- Tor is a successful privacy enhancing technology that works at the transport layer
  - Hundreds of thousands of users
- Normally, any TCP connection you make on the Internet automatically reveals your IP address
  - Why?
- Tor allows you to make TCP connections without revealing your IP address
- It's most commonly used for HTTP (web) connections

- Client connects to server, indicates it wants to speak TLS, and which cipher suites it knows
- Server sends its certificate to client, which contains:
  - Its host name
  - Its public key
  - Some other administrative information
  - A signature from a Certificate Authority
- Server also chooses which cipher suite to use
- Client sends symmetric encryption key *K, encrypted with server's public key*
- *Communication now proceeds using K and the chosen ciphersuite*

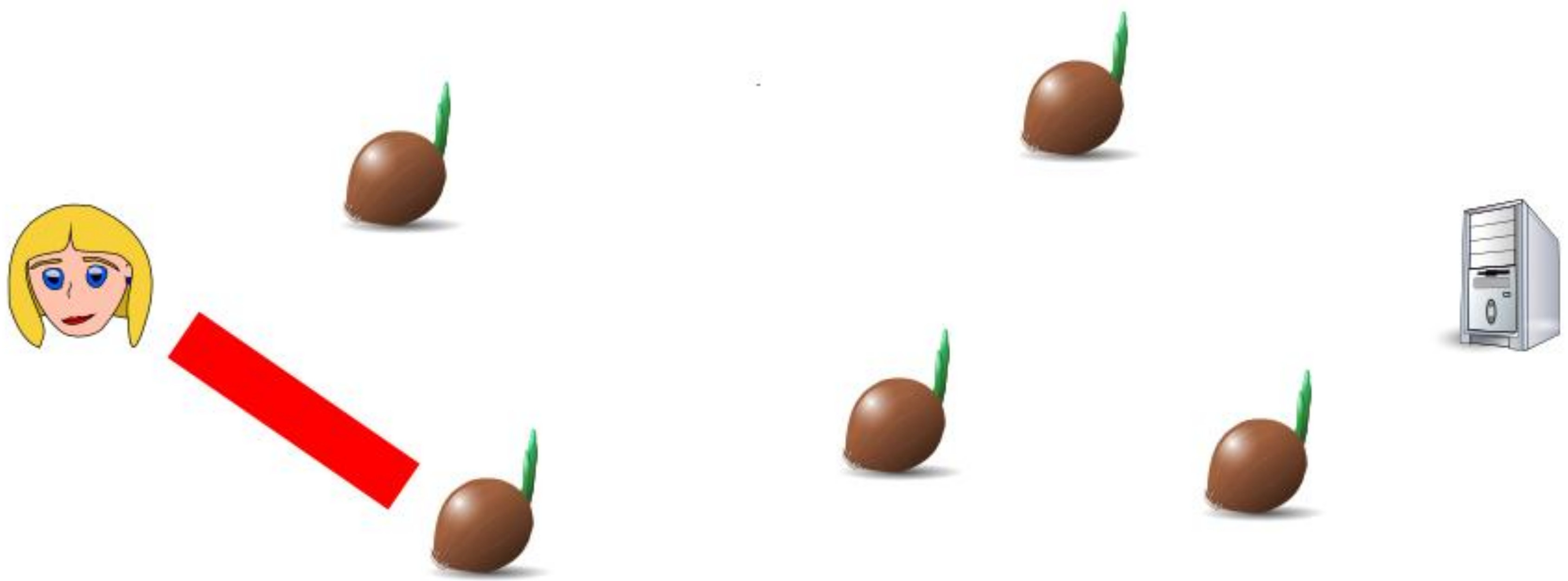
# How Tor works

- Scattered around the Internet are about 1000 Tor nodes, also called Onion Routers
- Alice wants to connect to a web server without revealing her IP address



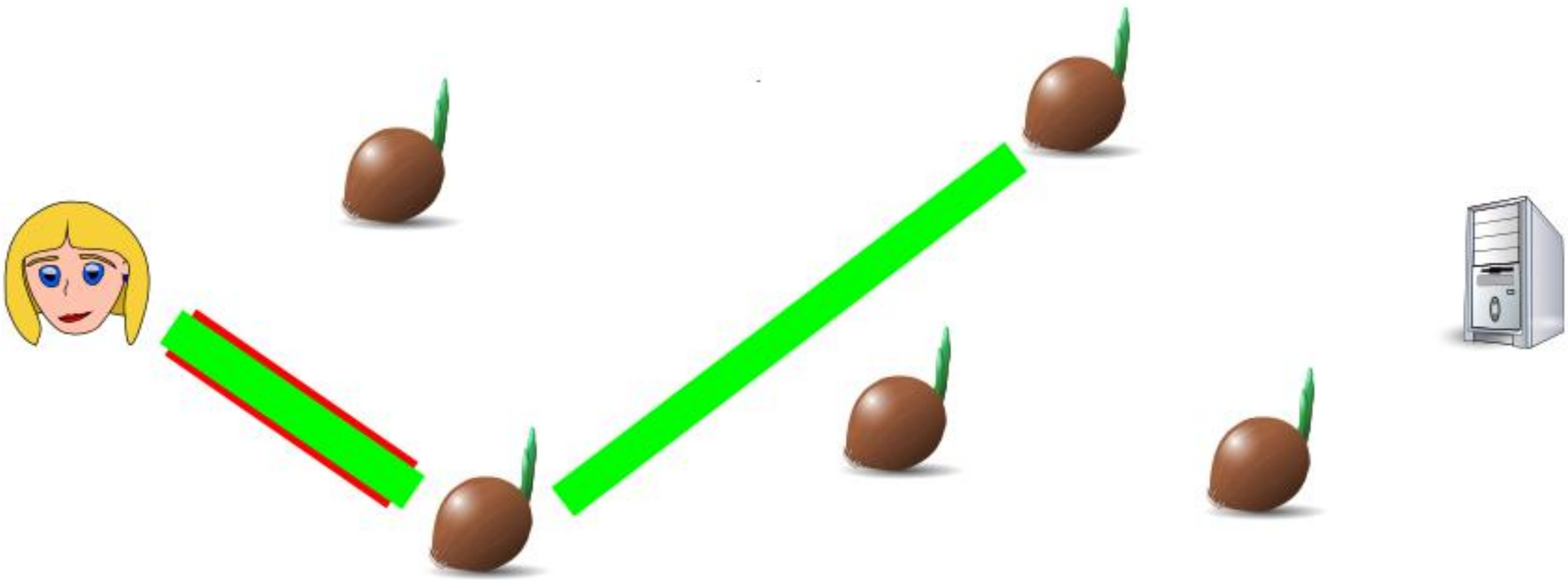
# How Tor works

- Alice picks one of the Tor nodes (n1) and uses public-key cryptography to establish an encrypted communication channel to it (much like TLS)



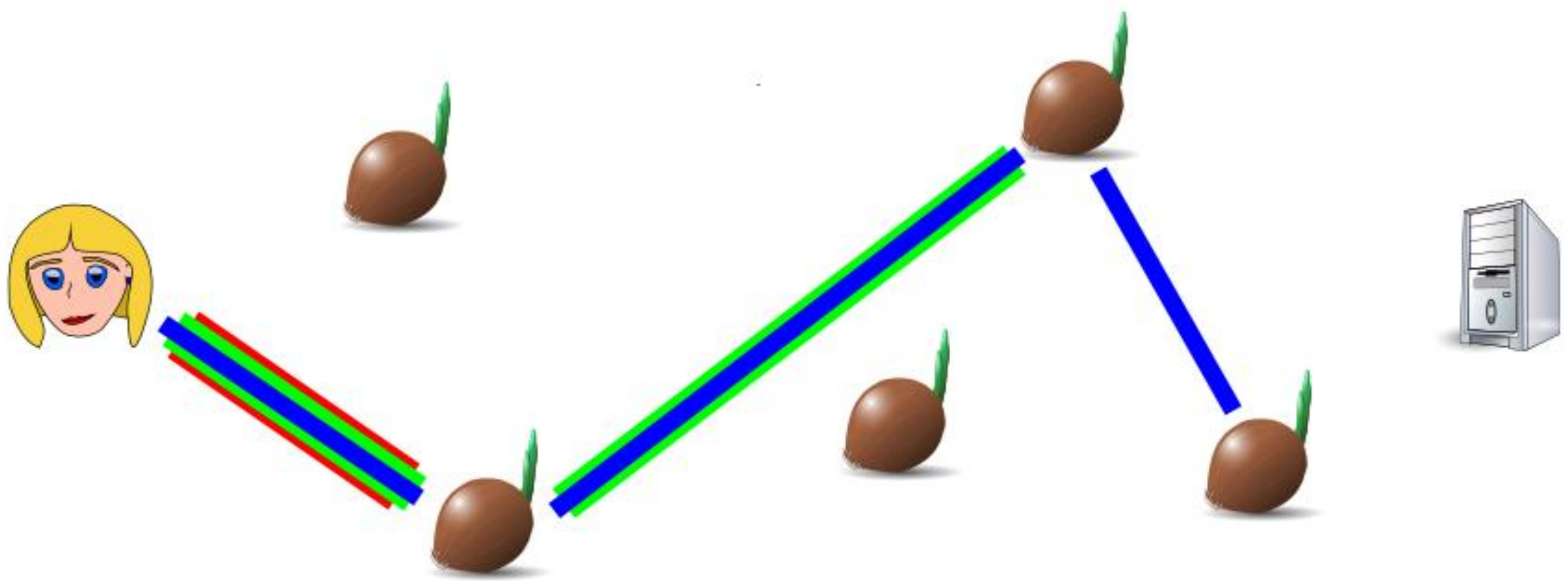
# How Tor works

- Alice tells n1 to contact a second node (n2), and establishes a new encrypted communication channel to n2, tunnelled within the previous one to n1



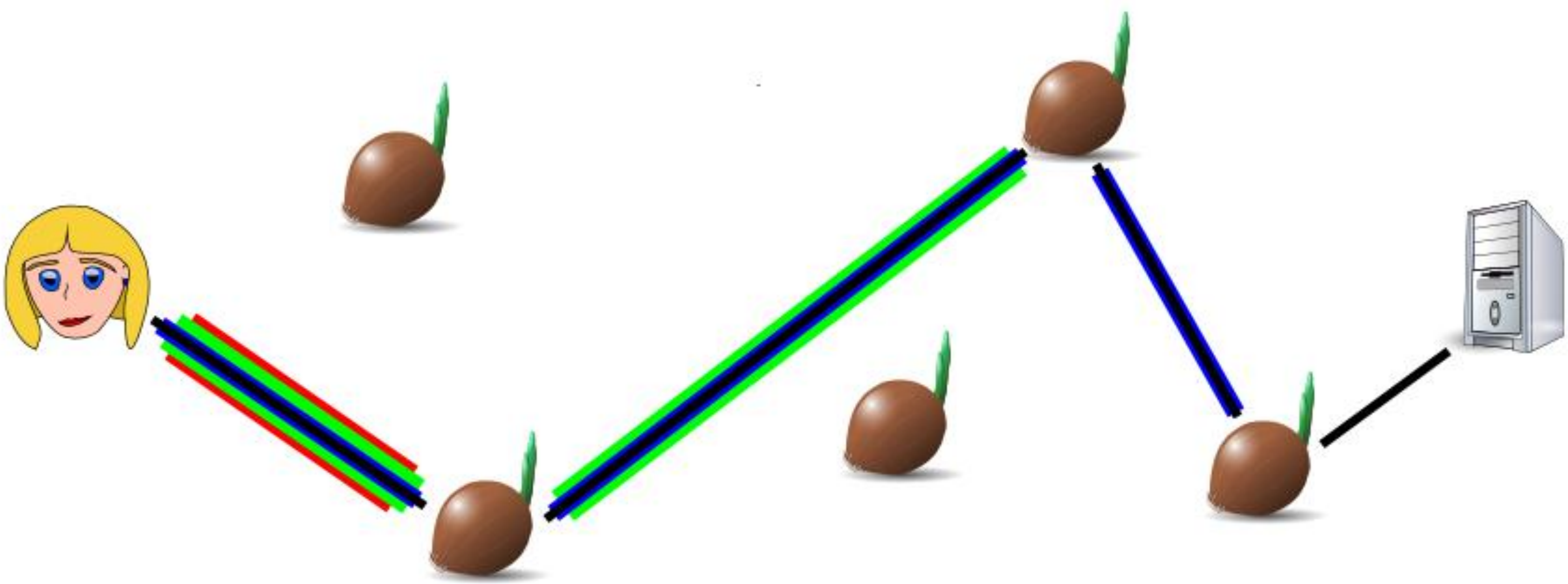
# How Tor works

- Alice tells n2 to contact a third node (n3), and establishes a new encrypted communication channel to n3, tunnelled within the previous one to n2



# How Tor works

- And so on, for as many steps as she likes (usually 3)
- Alice tells the last node (within the layers of tunnels) to connect to the website



# Sending messages with Tor

- Alice now shares three symmetric keys:
  - *K1* with n1
  - *K2* with n2
  - *K3* with n3
- When Alice wants to send a message M, she actually sends $E_{K1}(E_{K2}(E_{K3}(M)))$
- Node n1 uses *K1 to decrypt the outer layer, and passes the result $E_{K2}(E_{K3}(M))$ to n2*
- *Node n2 uses K2 to decrypt the next layer, and passes the result $E_{K3}(M)$ to n3*
- *Node n3 uses K3 to decrypt the final layer, and sends M to the website*

# Replies in Tor

- When the website replies with message R, it will send it to node n3
  - Why?
- Node n3 will encrypt R with *K3 and send $E_{K3}(R)$ to n2*
- *Node n2 will encrypt that with K2 and send $E_{K2}(E_{K3}(R))$ to n1*
- *Node n1 will encrypt that with K1 and send $E_{K1}(E_{K2}(E_{K3}(R)))$ to Alice*
- *Alice will use K1, K2, and K3 to decrypt the layers of the reply and recover R*

# Who knows what?

- Notice that node n1 knows that Alice is using Tor, and that her next node is n2, but does not know which website Alice is visiting
- Node n3 knows some Tor user (with previous node n2) is using a particular website, but doesn't know who
- The website itself only knows that it got a connection from Tor node n3
- Note: the connection between n3 and the website is not encrypted! If you want encryption as well as the benefits of Tor, you should use encryption in addition (HTTPS)

# Anonymity vs. pseudonymity

- Tor provides for anonymity in TCP connections over the Internet, both unlinkably (long-term) and linkably (short-term)
- What does this mean?
  - There's no long-term identifier for a Tor user
  - If a web server gets a connection from Tor today, and another one tomorrow, it won't be able to tell whether those are from the same person
  - But two connections in quick succession from the same Tor node are more likely to in fact be from the same person

# Application-layer security and privacy

- TLS can provide for encryption at the TCP socket level
  - “End-to-end” in the sense of a network connection
  - Is this good enough? Consider SMTPS (SMTP/email over TLS)

- Many applications would like true end-to-end security
- We'll look at three particular applications:
  - Remote login, email, instant messaging

# Secure remote login (ssh)

- You're already familiar with this tool for securely logging in to a remote machine
- Usual usage (simplified):
  - Client connects to server
  - Server sends its public key
    - The client should verify that this is the correct key
  - Client picks a random session key, encrypts it with server's public key, sends to server
    - All communication from here on in is encrypted and MACd with the session key
  - Client authenticates to server
  - Server accepts authentication, login proceeds (under encryption and MAC)

# Authentication with ssh

- There are two main ways to authenticate with ssh:
  - Send a password over the encrypted channel
    - The server needs to know (a hash of) your password
  - Sign a challenge with your private signature key
    - The server needs to know your public key
- Which is better?  Why?

[http://www.debian-administration.org/articles/530]

# Anonymity for email: remailers

- Tor allows you to anonymously communicate over the Internet in real time
  - What about (non-interactive) email?
  - This is actually an easier problem, and was implemented much earlier than Tor
- Anonymous remailers allow you to send email without revealing your own email address
  - Of course, it's hard to have a conversation that way
  - Pseudonymity is useful in the context of email

# Recap

- Internet Application Security and Privacy
  - Transport-layer security and privacy: Tor
  - Application-layer security and privacy: SSH, remailers

# Next time

- Internet Application Security and Privacy
  - Application-layer security and privacy: remailers, PGP/gpg