COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Onion Routing and Tor

Information Privacy with Applications

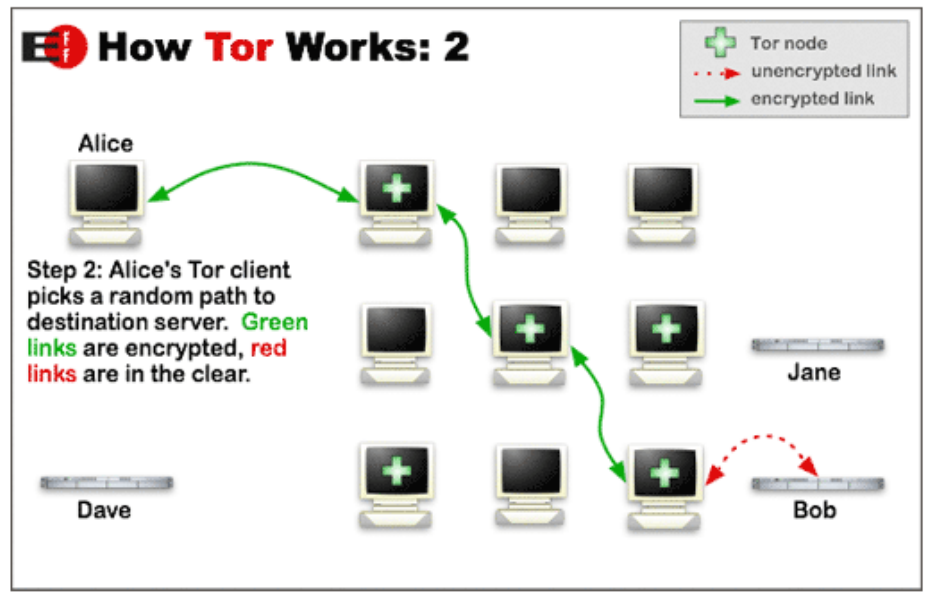
David Sidi (dsidi@email.arizona.edu)

Administrative Items

- Final project proposal assignment posted
- More on the anonymity assignment

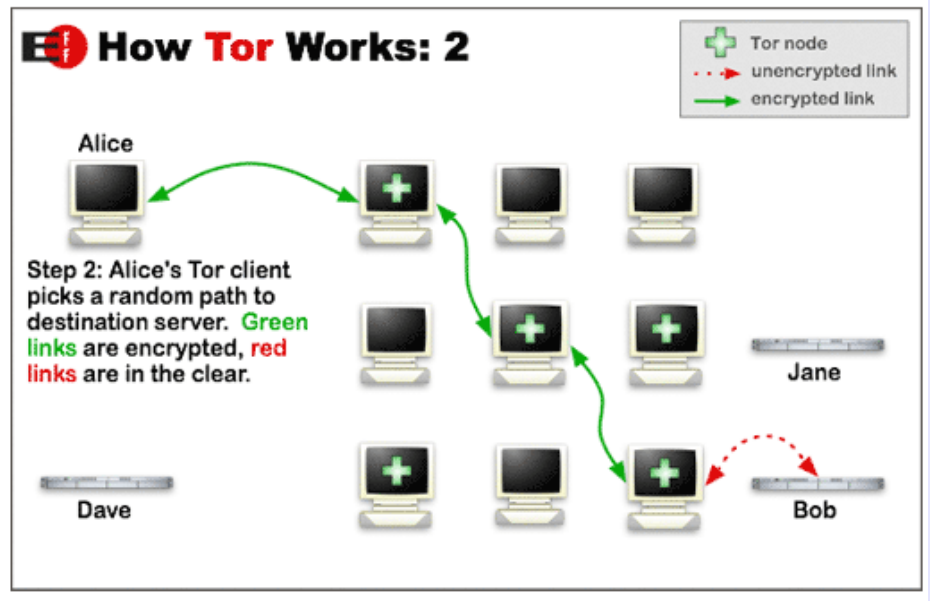
Onion Routing

- First was from the US Naval Laboratory, 1996
  - pure peering at this stage, loafers!
- Freedom Network was an independent onion routing network from Zero Knowledge Systems
- Tor is a third-gen. onion routing network



Tor is an onion routing system

- Example: simplified, slightly out-of-date Tor (*link*)
- Distributed TCP overlay network
- Sets up a “virtual circuit” as a cascade of three onion relays (OR) from the initial client onion proxy (OP)
- guard (from “helper nodes”), relay, and exit nodes
  - each node only knows its immediate predecessor and successor



Tor is an onion routing system

- originally, onion routing systems sent an initial onion message that was “just layers” to set up the circuit; Tor does it in stages (“telescoping”)
- Next hop in the circuit is determined by unwrapping an “extend” relay cell with a symmetric key, which causes the OR to send its own “create” control cell

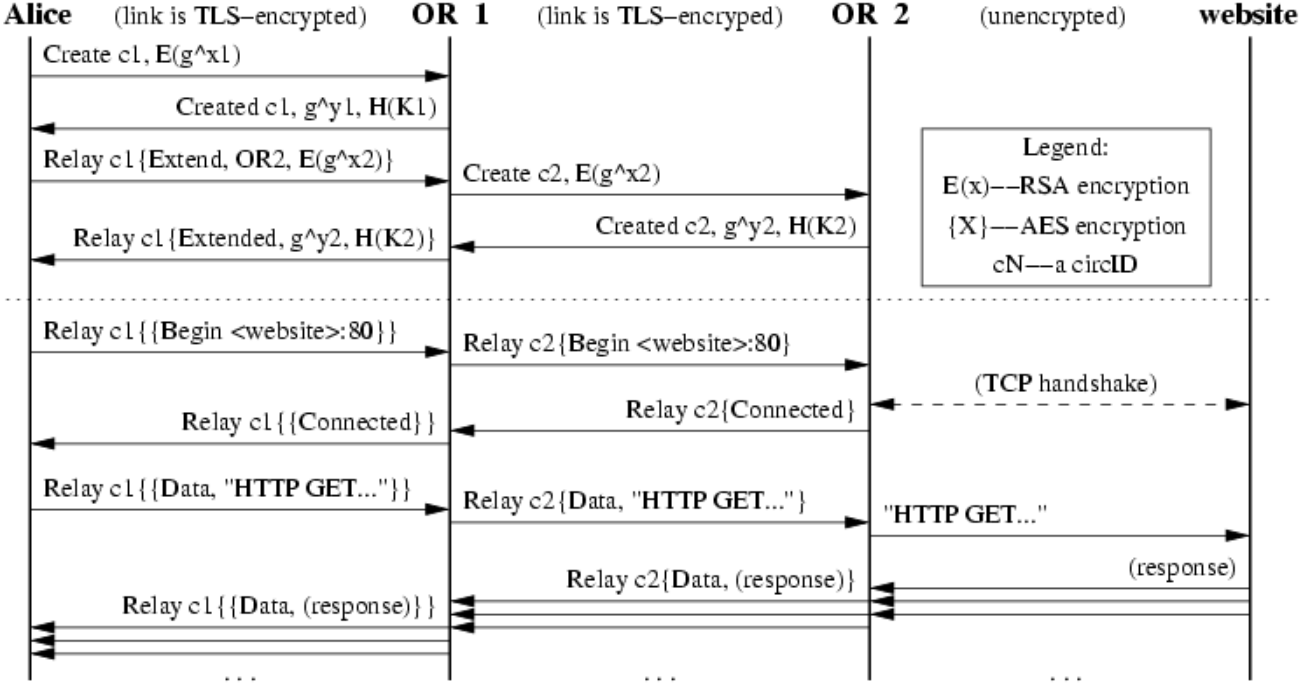


Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

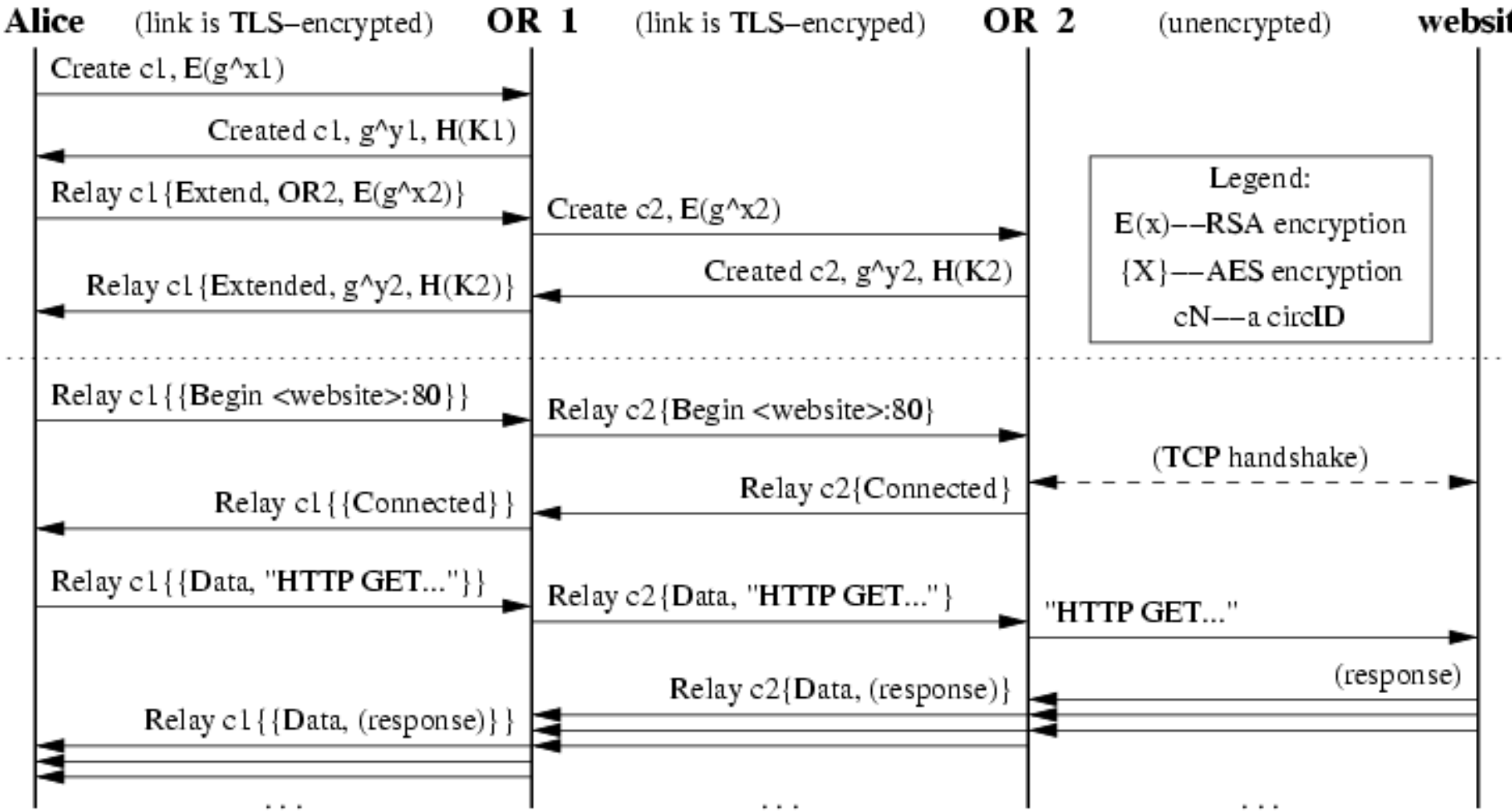


Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

OP picks the route

- First picks the exit node E such that E's exit policy includes at least one pending stream that needs a circuit
- Choose N-1 distinct nodes (default is three), with some order
- Open a connection to the first (guard) node, negotiate session keys
- extend the circuit incrementally over the remaining N-1 nodes

Tor uses PKC to protect negotiation of a session key

- One hop at a time over an encrypted and authenticated channel
  - TLS: use *identity keys* to sign certs
- Use public-key cryptography (PKC) over this channel to set up an ephemeral session key
  - PKC is RSA (legacy) or Curve25519: use short-term *onion keys*
  - symmetric is AES, set up with DHE (legacy) or ECDHE
- Once ephemeral keys are set up OP layers them, and ORs unwrap them

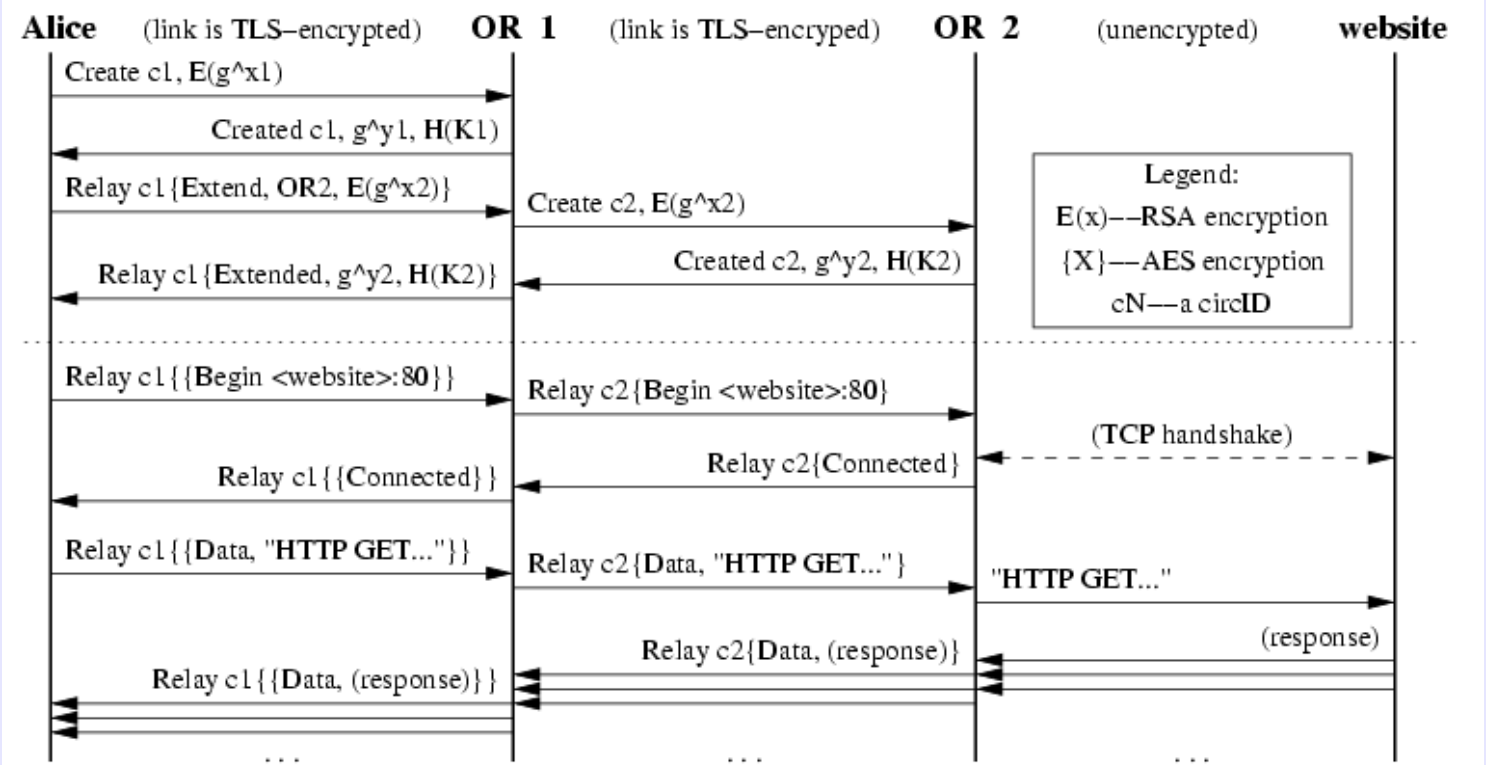


Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

Discussion

- We have a public key for the guard. What reason did I give to not just use PKC to encrypt communications?

Session keys are negotiated using Diffie-Hellman Key Exchange

- First published in 1976; still around
- Alice and Bob want to share a secret key for use in a symmetric cipher. Every piece of information that they exchange is observed by their adversary Eve. How is it possible for Alice and Bob to agree on a key without making it available to Eve?

Diffie-Hellman

Publicly choose:

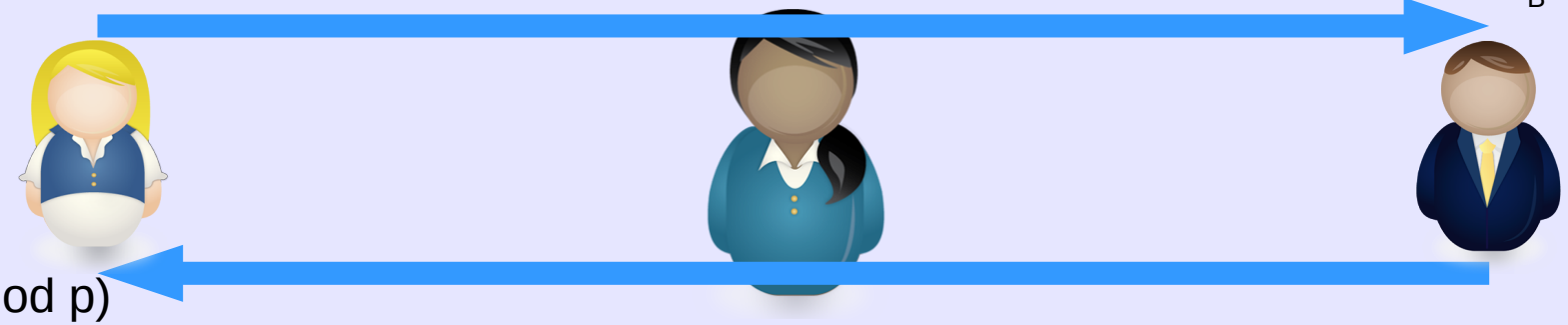
- a safe large prime *p* (e.g. Tor docs use rfc2409 section 6.2. But see Logjam)
- g, a primitive root mod p, with *2 <= g <= p-2*

Secretly generate:

- Alice and Bob randomly choose secret integers 1 <= *x, y <= p-2* respectively

$A := g^x \pmod p$

$K_B := A^y \pmod p$

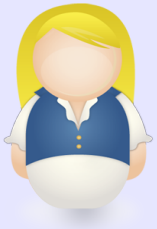


$K_A := B^x \pmod p$

$B := g^y \pmod p$

$$K_A = (g^y)^x = (g^x)^y = K_B \text{ is the key}$$

Diffie-Hellman



$x = \log_g A \pmod p$

$y = \log_g B \pmod p$

Diffie-Hellman

$x = \log_g A \pmod p$

$y = \log_g B \pmod p$

Discrete Log Problem is in NP

Diffie Hellman Problem is no harder than DL problem; there is no proof of the converse

Question

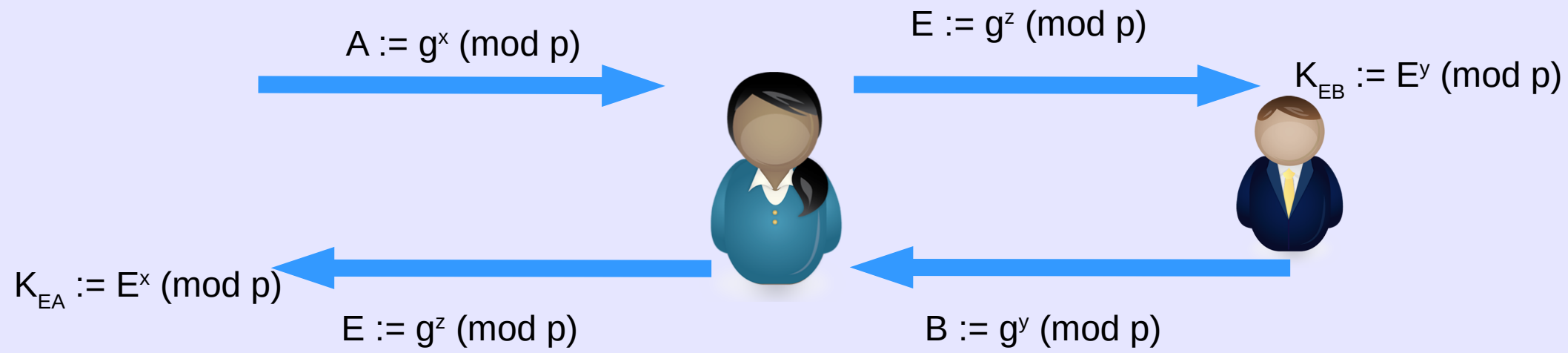
- Ian Goldberg remarked that a good way to fight mass surveillance by a global passive adversary would be to “do a quick Diffie-Hellman” by default when setting up otherwise unprotected connections. He notes that this won’t help against an active attack. Can you guess what he means by an active attack?

MiTM Diffie-Hellman

Publicly choose:
a secure large prime *p*
g, a primitive root mod p, with *2 <= g <= p-2*

Secretly generate:

- Alice and Bob choose secret integers 0 <= *x, y <= p-2* respectively
- Eve picks her own secret integer, *z*



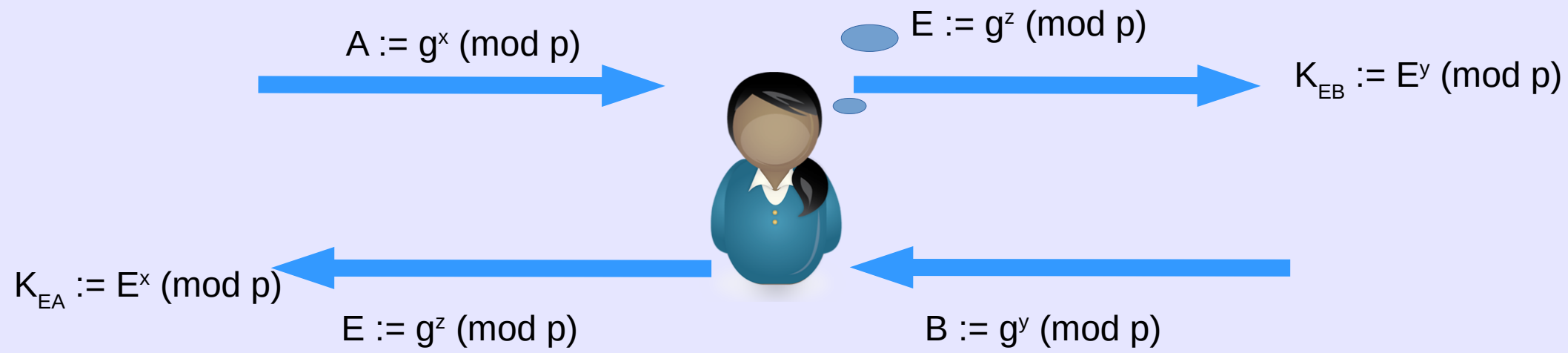
MiTM Diffie-Hellman

Publicly choose:
a secure large prime *p*
g, a primitive root mod p, with *2 <= g <*

$K_{EA} := g^{xz} \pmod p$
$K_{EB} := g^{yz} \pmod p$

Secretly generate:

- Alice and Bob choose secret integers 0 <= *x, y <= p-2* respectively
- Eve picks her own secret integer, *z*



Tor protocol, cont'd

- once session keys are agreed upon with DH, encrypt for the exit node, then encrypt the result for the relay node, and finally encrypt the result for the guard node
- send the layered result to the guard node
- guard node decrypts, gets the next hop and sends it on, then the middle key decrypts, ...
- on way back, each node uses the session key agreed on with the client OP, and passes to its neighbor in the circuit

Tor strengths and weaknesses

- Strengths
  - widespread adoption
  - low latency
  - easy to run nodes, easy to use as a client: adds to security
  - bridges, pluggable transports for censorship circumvention
  - fingerprint resistance (Tor Browser)
  - Applications ecosystem (SecureDrop, Briar, Ricochet, OnionShare, Tails, ...)
- Weaknesses
  - traffic analysis by a global/pervasive passive adversary
  - end-to-end timing attacks
  - content is revealed to exit node
  - blockable exit nodes

Who runs exit nodes?

- Universities
  - MIT+, Michigan, CMU, UNC, Karlsruhrer IT, Stanford, Clarkson, U. Washington, Utah+, Caltech, RIT+, Bowdoin, Northeastern+, Princeton
- Bad people too! (Why might they do that?)
- Not Arizona :-(
  - yet :-)

Who runs hidden services?

- Propublica, Duckduckgo, Facebook, Scihub, Riseup, Protonmail, Debian, Whonix, The Intercept, Wikileaks, Securedrops for The Freedom of the Press Foundation, The Guardian, The Associated Press, NY Times, USA Today, Washington Post, etc., TORCH (these are all onion links)
- A bunch of illegal stuff
- Hidden services are easy to set up (demo)
  - even inside firewalled networks