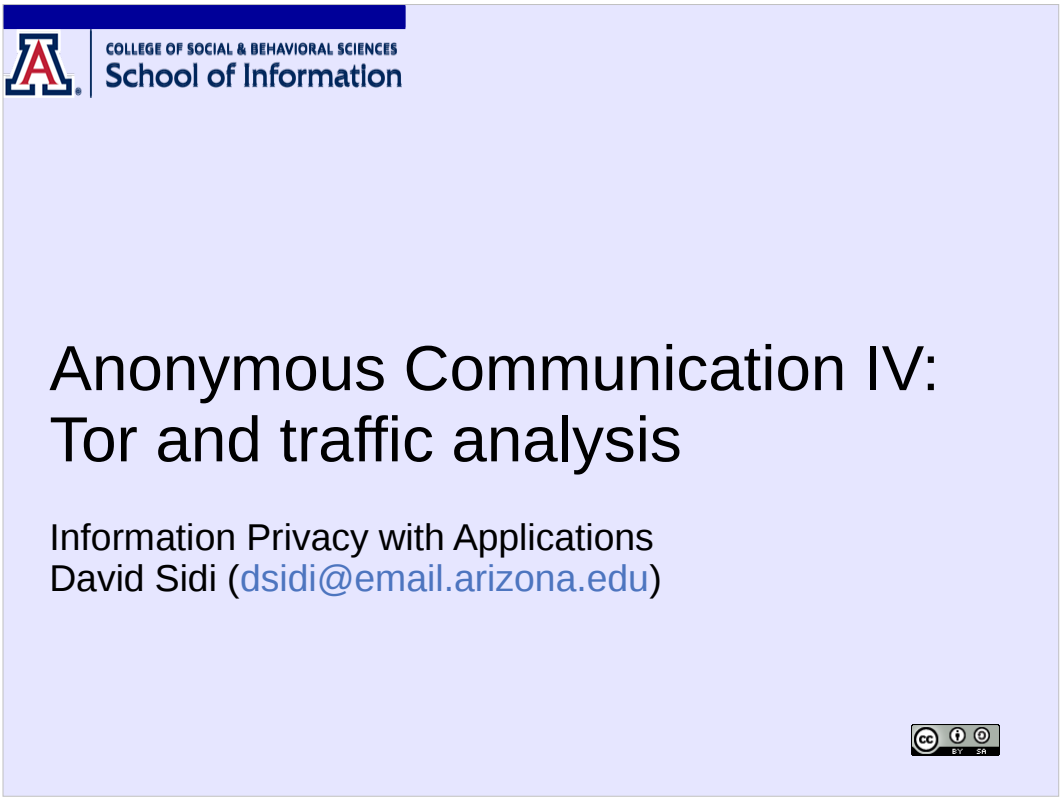


We’re going to finish talking about Tor, discuss Diffie-Hellman key exchange, TLS, and do some hands on with mitmproxy

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Small mention of interesting things

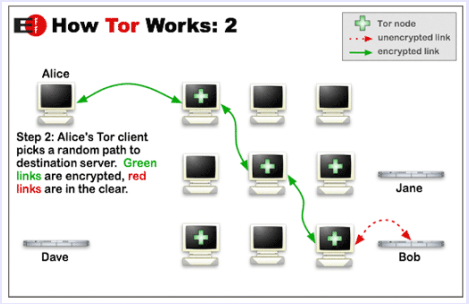
- Google offered former Stasi headquarters for its campus in the Licthenberg district in Berlin

recall from last time that DNS leaks information about your browsing.

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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- Example: simplified, slightly out-of-date Tor (*link*)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



CC-SA License by David Sidi

each time a user creates a circuit, there is a small chance that the circuit will be compromised. However, most users create a large number of Tor circuits, so with the original path selection algorithm, these small chances would build up into a potentially large chance that at least one of their circuits will be compromised.

For users who have good guard nodes, the situation is much better, and for users with bad guard nodes the situation is not much worse than before.

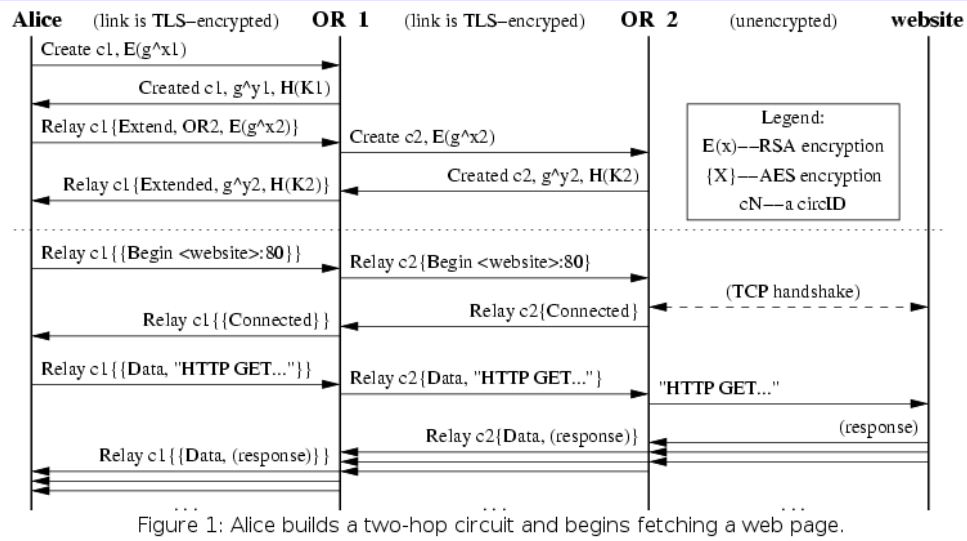
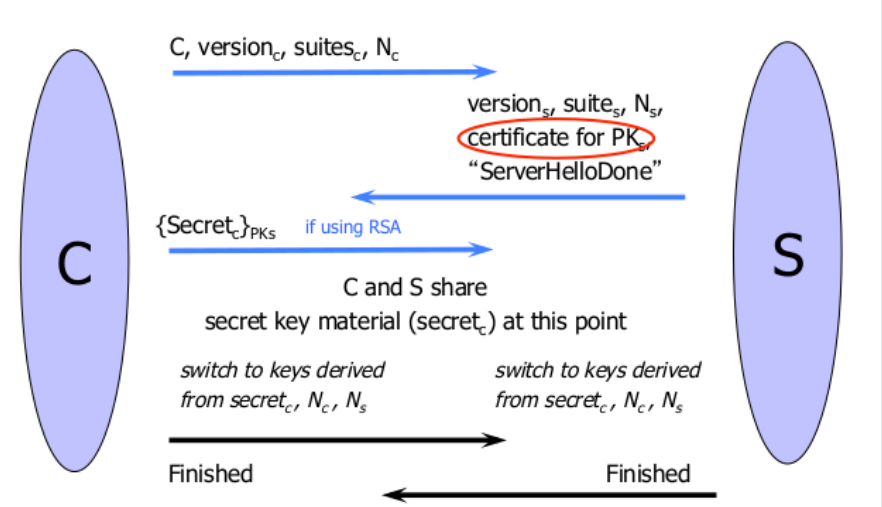


Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

CC-SA License by David Sidi

- 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
- walk through diagram

Excursus on TLS



credit: Vitaly Shmatikov

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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 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

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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, router descriptors
- 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

CC-SA License by David Sidi

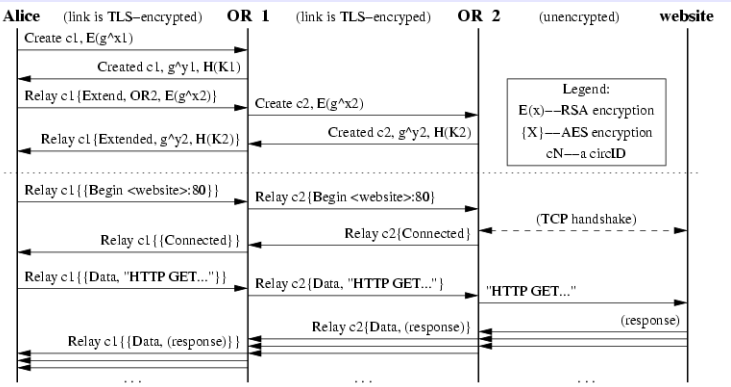


Figure 1: Alice builds a two-hop circuit and begins fetching a web page.

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Discussion

- We have a public key for the guard. What reason did I give to not just use PKC to encrypt communications?
- Why encrypt the first half of the DH handshake?

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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?

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

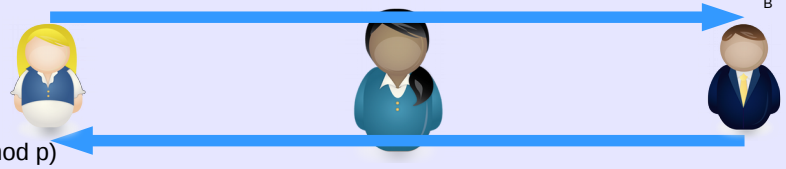
Diffie-Hellman

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

$A := g^x \pmod p$

$K_B := A^y \pmod p$



$K_A := B^x \pmod p$

$B := g^y \pmod p$

$K_A = (g^y)^x = (g^x)^y = K_B$ is the key

CC-SA License by David Sidi

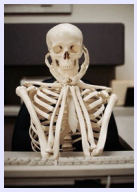
COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Diffie-Hellman

$x = \log_g A \pmod p$
$y = \log_g B \pmod p$

CC-SA License by David Sidi

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 (do you see why not?); there is no proof of the converse

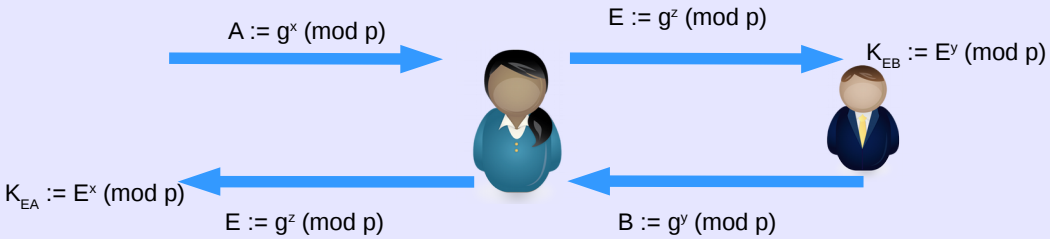
Question

- Ian Goldberg remarked that a good way to fight mass surveillance by a global passive adversary with minimum fuss 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*



CC-SA License by David Sidi

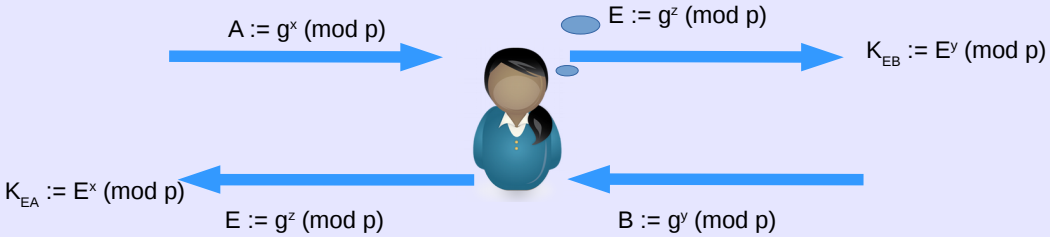
COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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*



CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Tor strengths and weaknesses

- Strengths
  - faster than mixnets
  - perfect forward secrecy
  - easy to run nodes, easy to use as a client: adds to security
  - bridges, pluggable transports for censorship circumvention
  - sandboxing
- Weaknesses
  - traffic analysis by a pervasive passive adversary
  - end-to-end timing attacks
  - content is revealed to exit node
  - blockable exit nodes

CC-SA License by David Sidi

COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

Who runs exit nodes?

- Universities (as of Oct 2017) (link)
  - 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 :-)

CC-SA License by David Sidi

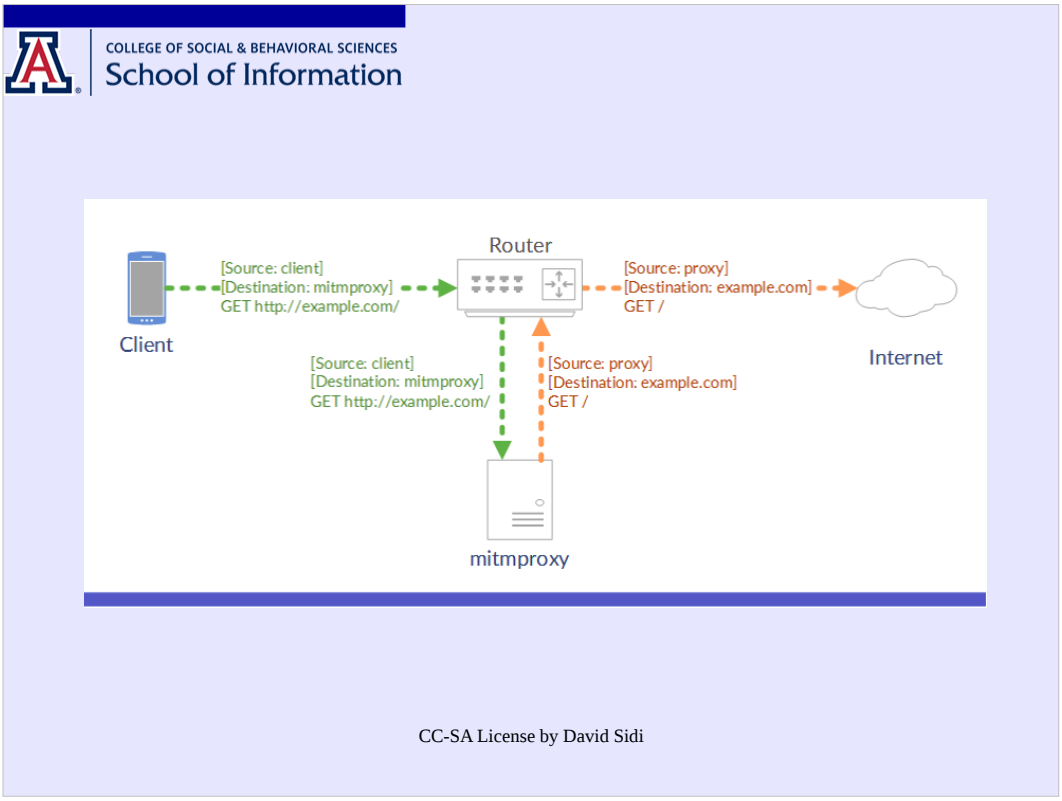
COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

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

CC-SA License by David Sidi

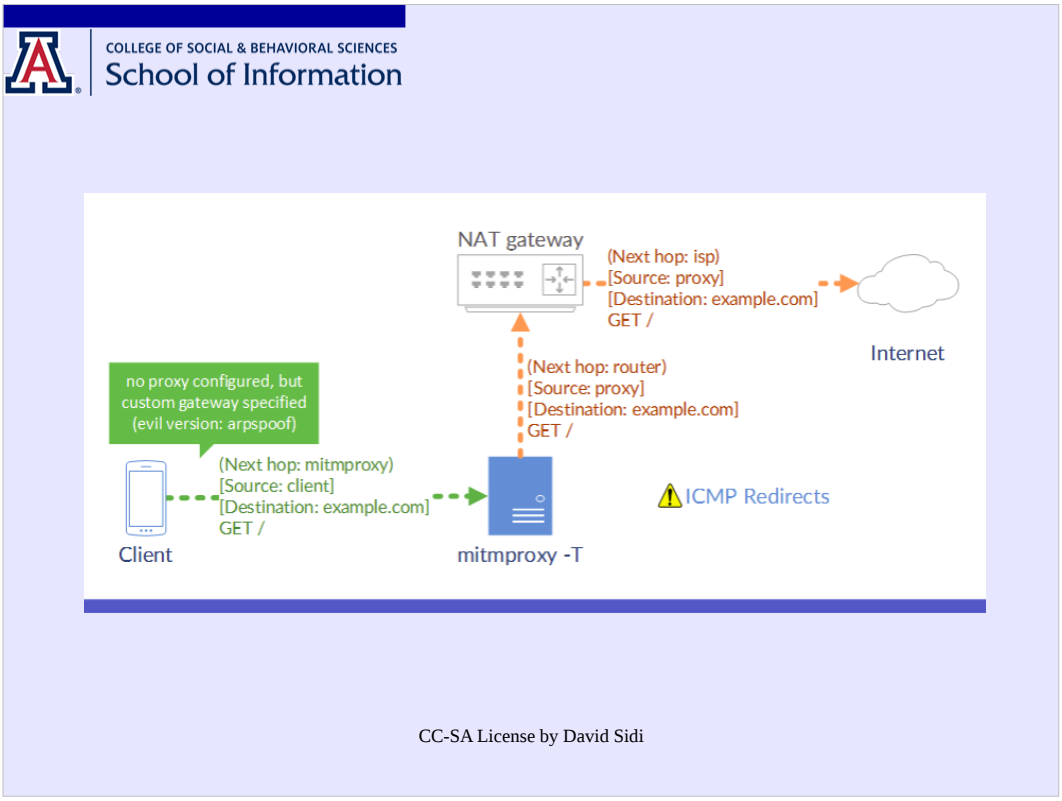

D.C.'09



transparent vs. nontransparent proxying

nontransparent, remote proxy case

what is the local proxy case? Let's set up local nontransparent proxying on ourselves. (1) install conda, pip, and python 3.x if you don't have these already (2) create and activate a new environment called "mitm_sandbox" (3) install requisites for mitmproxy (4) install mitmproxy (5) set up proxy settings on your browser (6) run mitmproxy (8) access httpS://webauth.arizona.edu

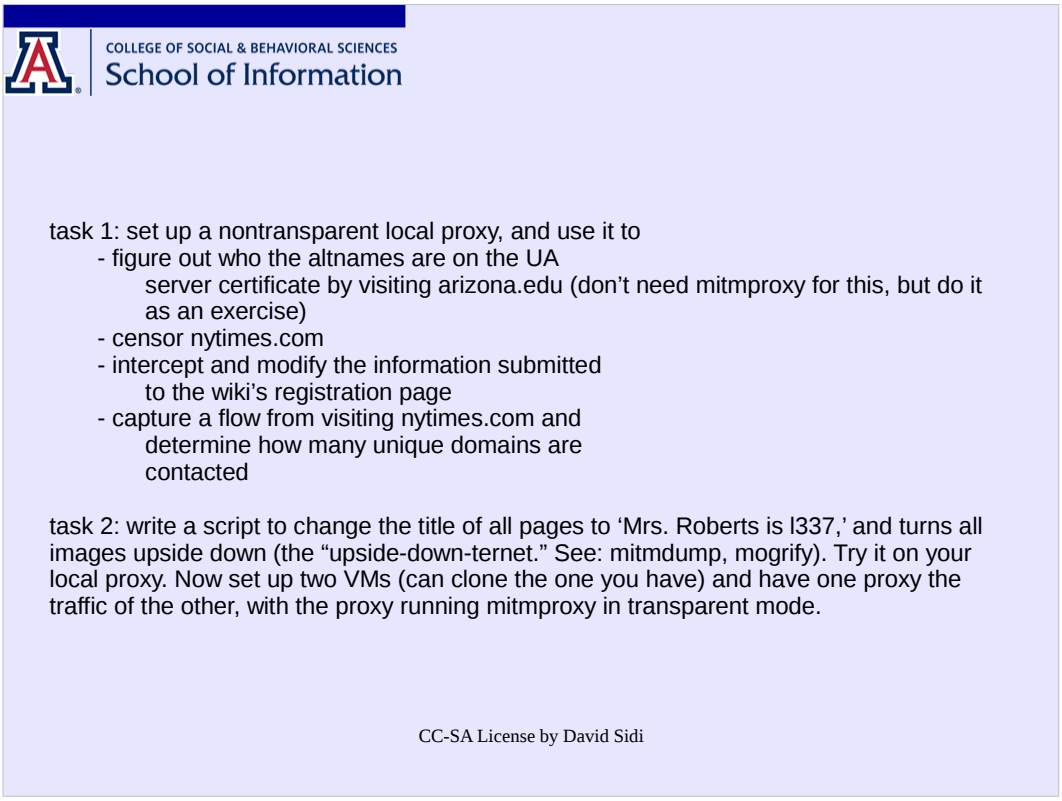


transparent case (note: -T): what does it mean to say traffic is “directed at the network layer” (what in TCP/IP is called the internet layer)?

mentioned in the docs without comment: “arp spoofing”

What is arp (from last time)? So arp spoofing (AKA arp cache poisoning)?

Simple defense, whipped up in an hour or so: antipineapple



set up:

install VM (optional, but recommended for windows)

install conda

install mitmproxy (pip)