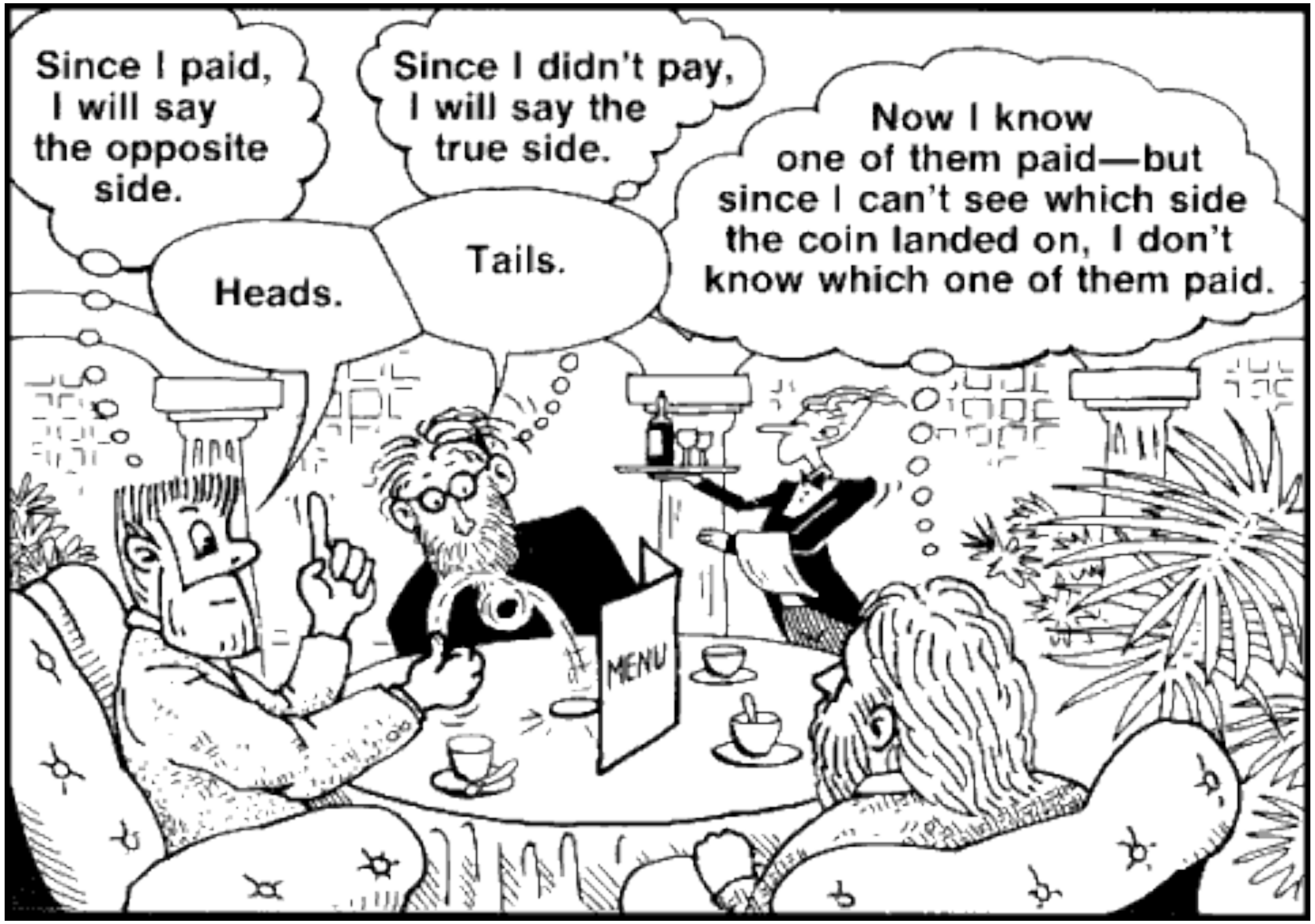
Anonymous Communication: DC-nets, Crowds, Onion Routing

Simone Fischer-Hübner

PETs PhD course

Spring 2012

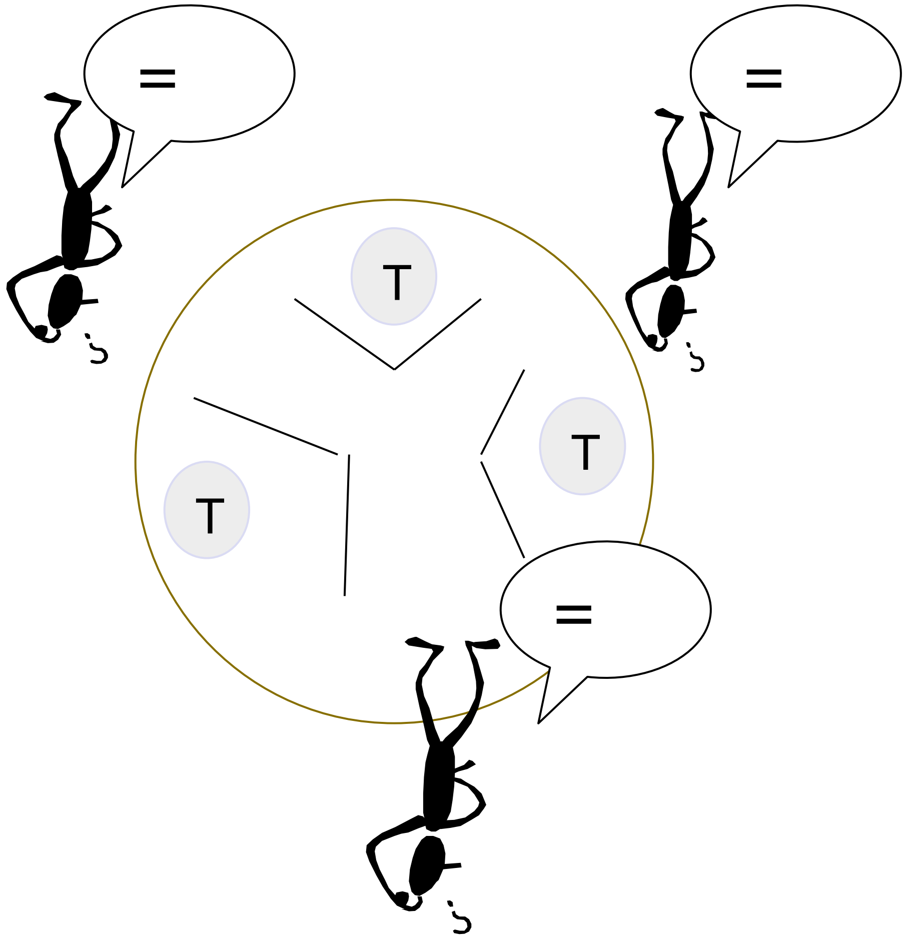
DC (Dining Cryptographers) nets [Chaum 1988]

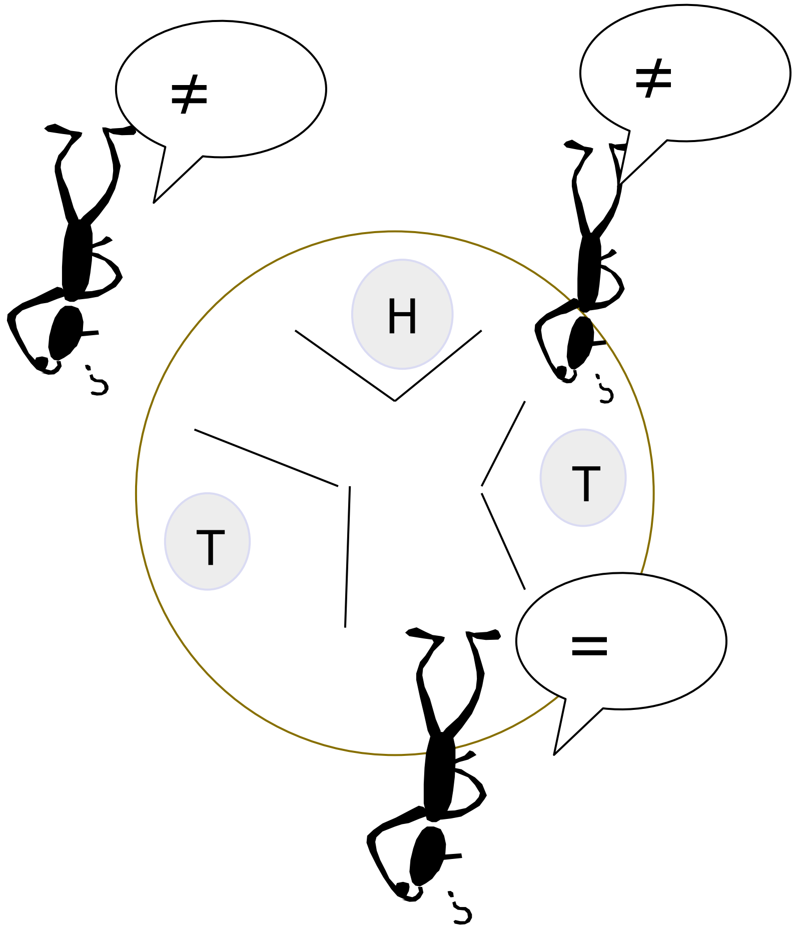

Since I paid, I will say the opposite side.
Since I didn't pay, I will say the true side.
Now I know one of them paid—but since I can't see which side the coin landed on, I don't know which one of them paid.
Heads.
Tails.
MENU

Chaum, CACM **28**(10), October 1985

Who paid for the Dinner (anonymously)? (I)

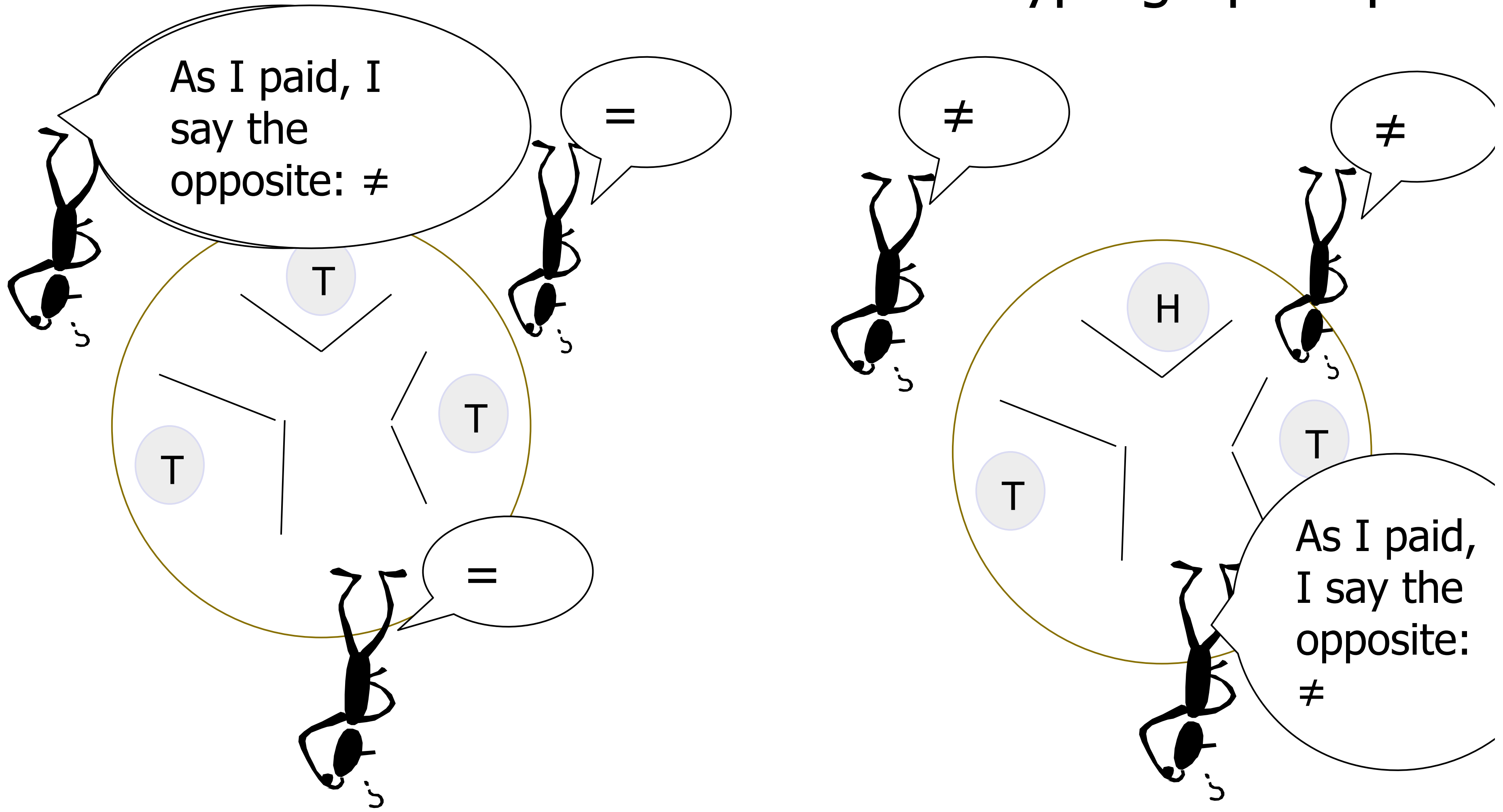
- Equal number of differences ⇔ NSA paid





Who paid for the Dinner (anonymously)? (II.a)

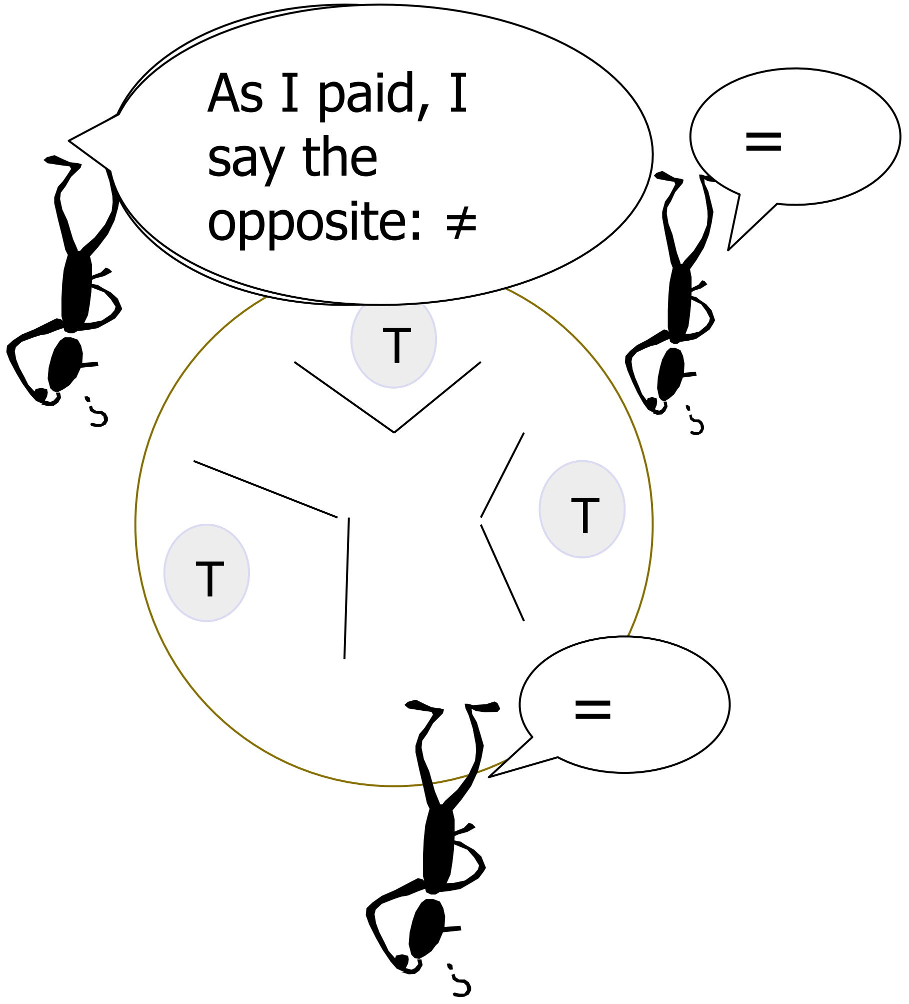
- Odd number of differences⇔ one cryptographer paid

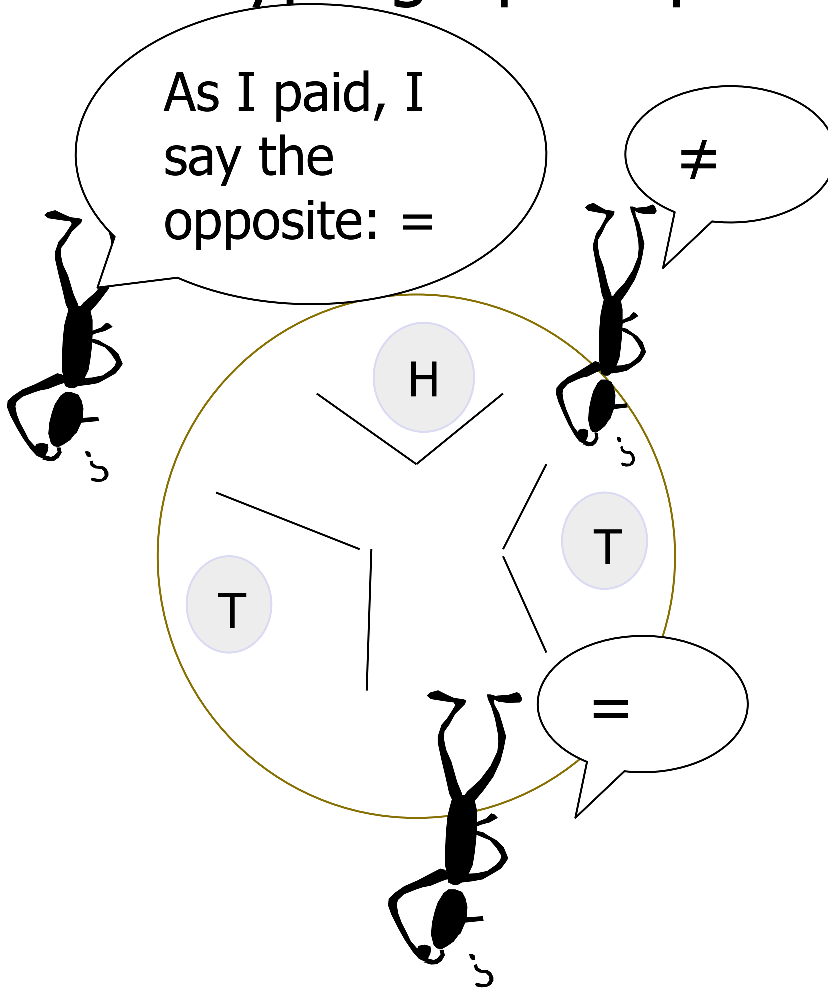


As I paid, I say the opposite: ≠
=
=
T
T
T
≠
≠
H
T
T
As I paid, I say the opposite: ≠

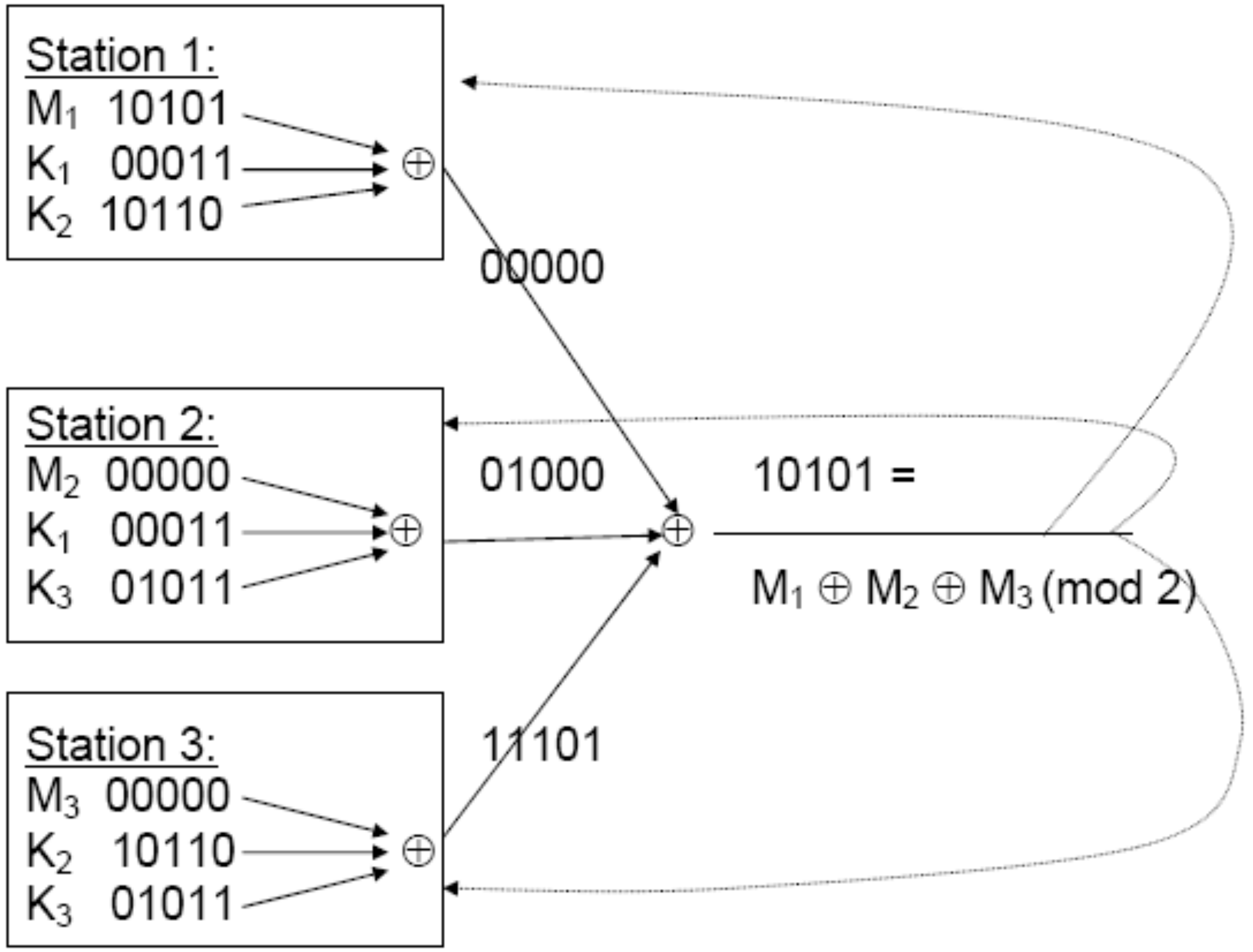
Who paid for the Dinner (anonymously)? (II.b)

- Odd number of differences⇔ one cryptographer paid





DC-nets: Perfect sender anonymity through Binary superposed sending and broadcast



Anonymity preserving multi-access protocols

Message: fixed number c of characters (containing also an implicit address)

slot: c consecutive rounds, in which a message is transferred

a.) <u>Slotted ALOHA:</u>

- If P_i wants to send a message, he does so in the next slot

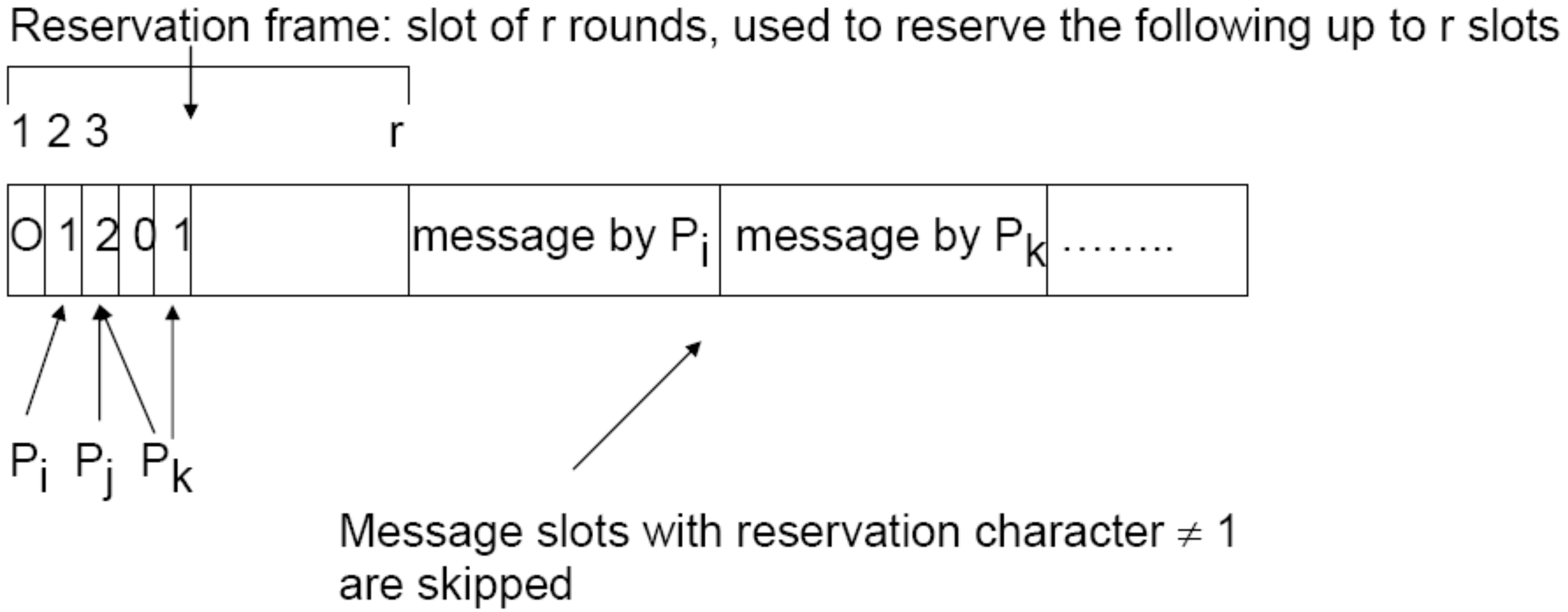
- If P_j also sends a message
  ➔ P_i (as well as P_j) detects collision ($S \neq M_i$).
  P_i retransmits his message after a random number of slots.

Retransmitted messages should be differently end-to-end encrypted. Otherwise:

If attacker monitors output messages X+Y, X, Y
  ➔ attacker assumes: X and Y were sent by different participants

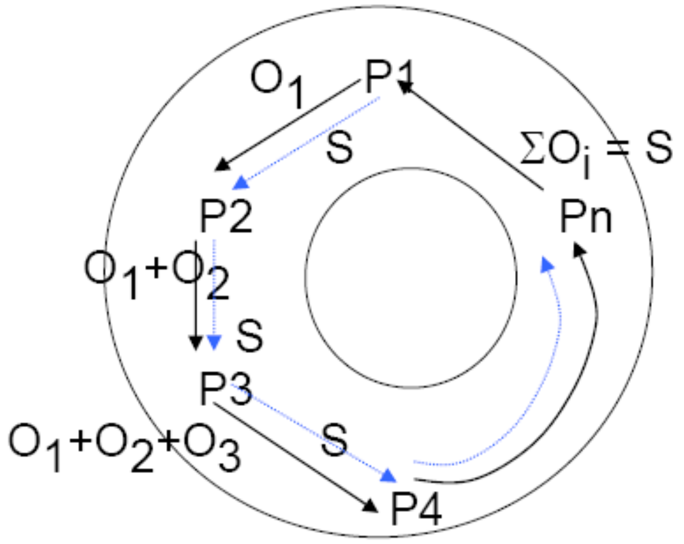
Anonymity preserving multi-access protocols (cont.)

b.) <u>Reservation map technique:</u>



Reservation frame: slot of r rounds, used to reserve the following up to r slots
1 2 3
r
O
1
2
0
1
message by P_i
message by P_k
....
P_i
P_j
P_k
Message slots with reservation character ≠ 1 are skipped

Implementation-Example: Local-Area Ring Networks



<u>First cycle:</u> $\longrightarrow$
Each participant adds his own output to the input he receives, forwards sum to the next participant

<u>Second Cycle:</u> $\dashrightarrow$
Result (global sum) is sent around to each participant

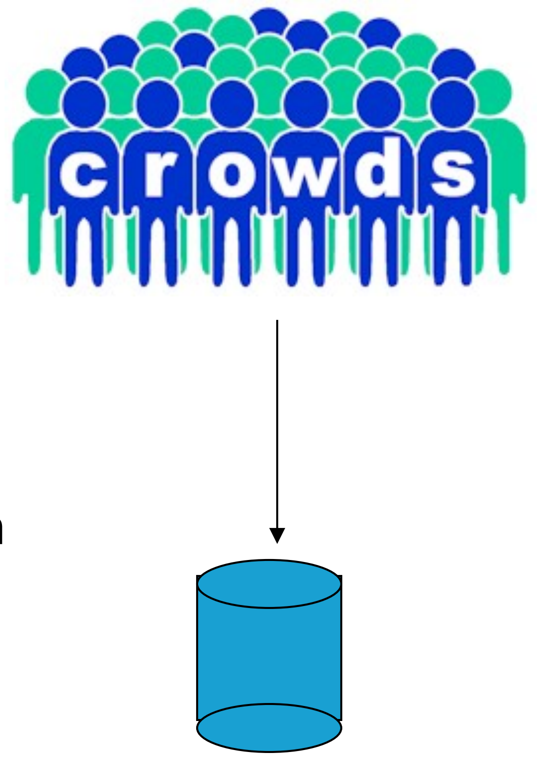
DC nets - Review

- Protection properties:
  - Perfect sender anonymity through superposed sending (message bits are hidden by one-time pad encryption)
  - Message secrecy through encryption
  - Recipient anonymity through broadcast and implicit addresses (addressee is user who can successfully decrypt message)
- Problems:
  - Denial of Service attacks by DC-net participants (Defense: trap protocols)
  - Random key string distribution

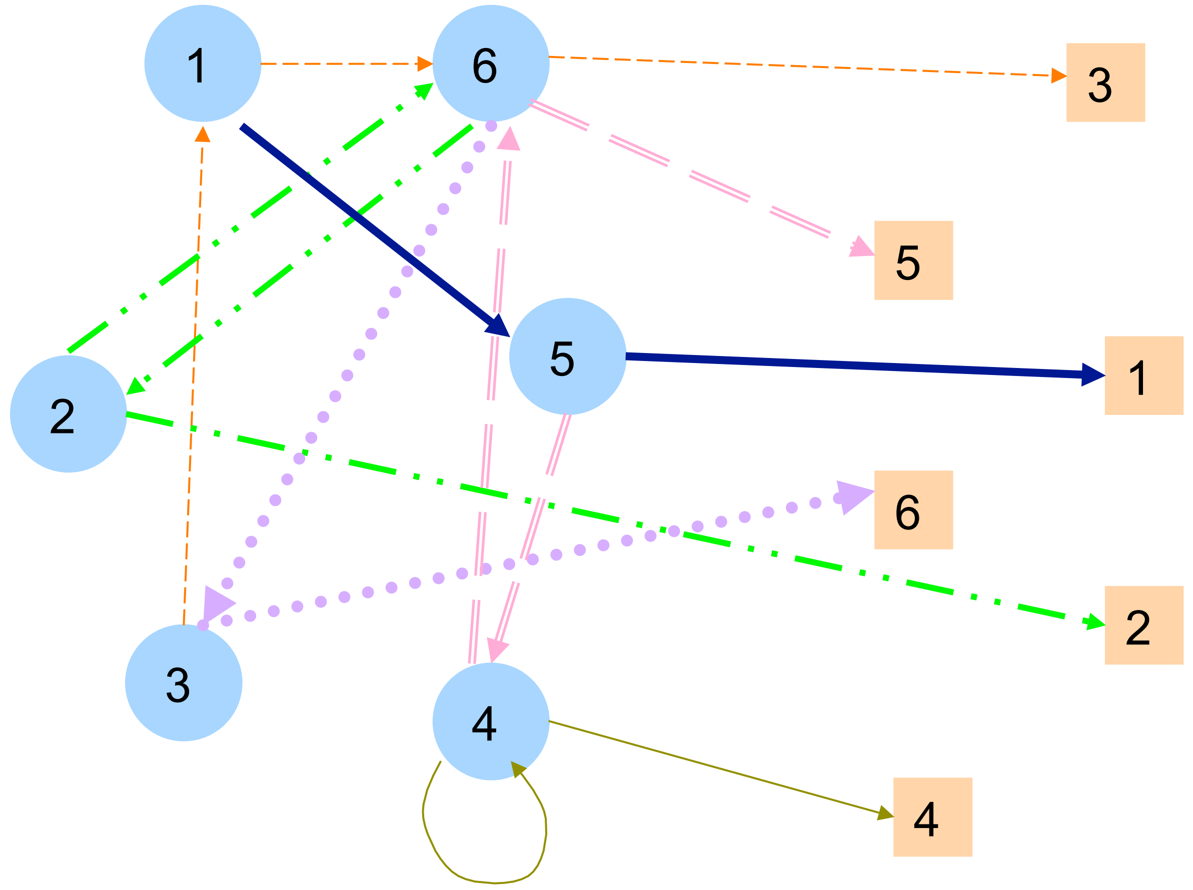
Crowds for anonymous Web-Transactions

1. User first joins a "crowd" of other users, where he is represented by a "jondo" process on his local machine
2. User configures his browser to employ the local jondo as a proxy for all new services
3. User´s request is passed by the jondo to a random member of the crowd
4. That member can either submit the request directly to the web server or forward it to another randomly (with pf> 1/2) chosen user.

-> Request is eventually submitted by a random member



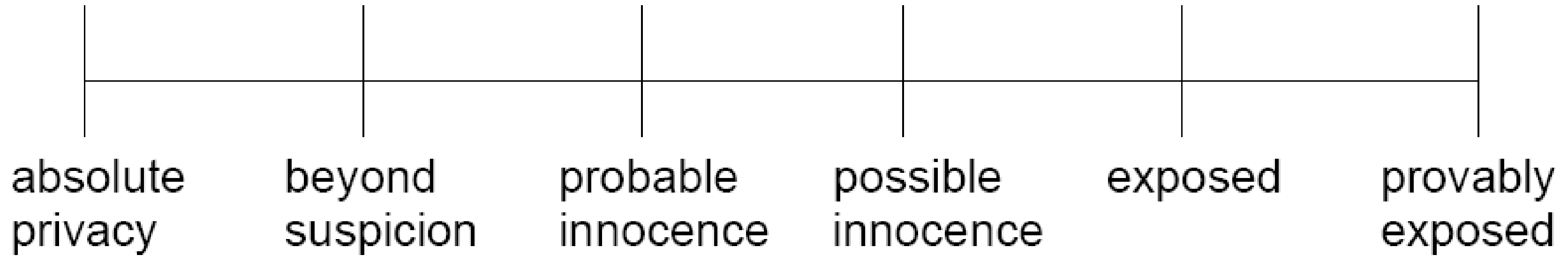
Communication Paths in Crowds



Communications between jondos is encrypted with keys shared between jondos

Anonymity degrees in Crowds

- **Absolute Privacy:** The attacker cannot distinguish the situations in which a potential sender sent a message and those in which he did not
- **Beyond suspicion:** sender appears no more likely to be originator of a message than any other potential sender in the system
- **Probably innocense:** sender appears no more likely to be originator than not to be the originator
- **Possible innocense:** There is a non-trival possibility that the sender is someone else



Anonymity Properties in Crowds

Attacker	Sender anonymity	Receiver anonymity
local eavesdropper	Exposed	P(beyond suspicion) $\xrightarrow[n \to \infty]{}$ 1
c collaborating members, n > [pf/ (pf- 1/2)] * (c+1)	probable innocence, P(absolute privacy) $\xrightarrow[n \to \infty]{}$ 1	P(absolute privacy) $\xrightarrow[n \to \infty]{}$ 1
end servers	beyond suspicion	N/A

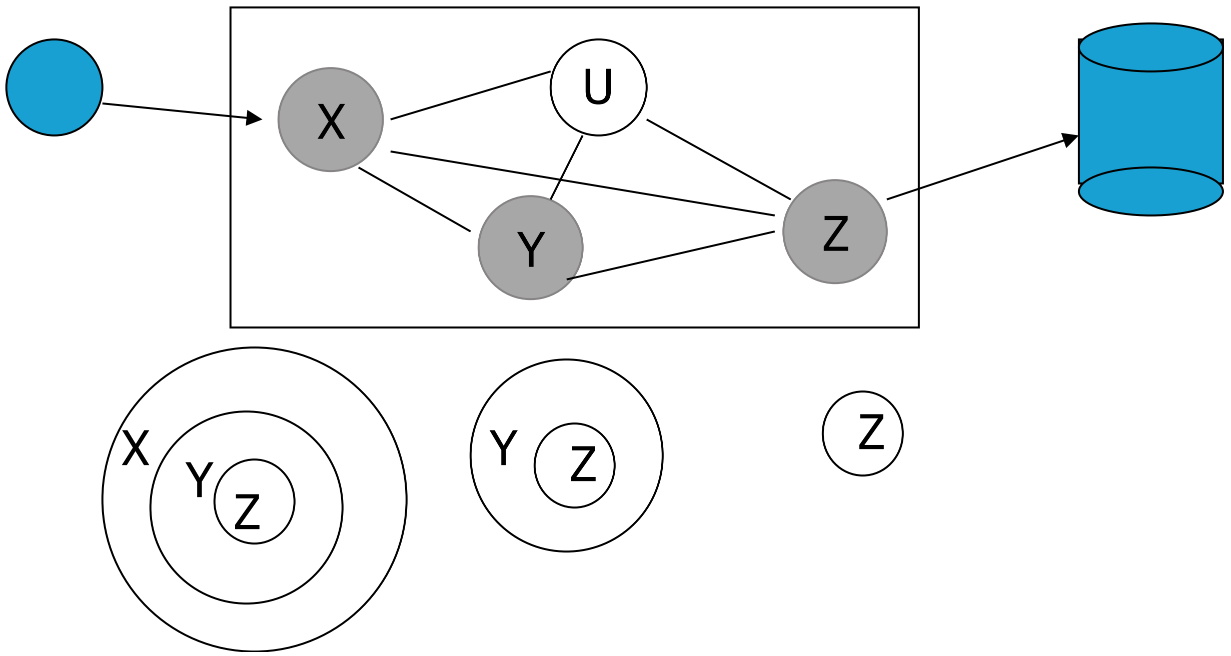
n: Number of Crowds members

Crowds -Review

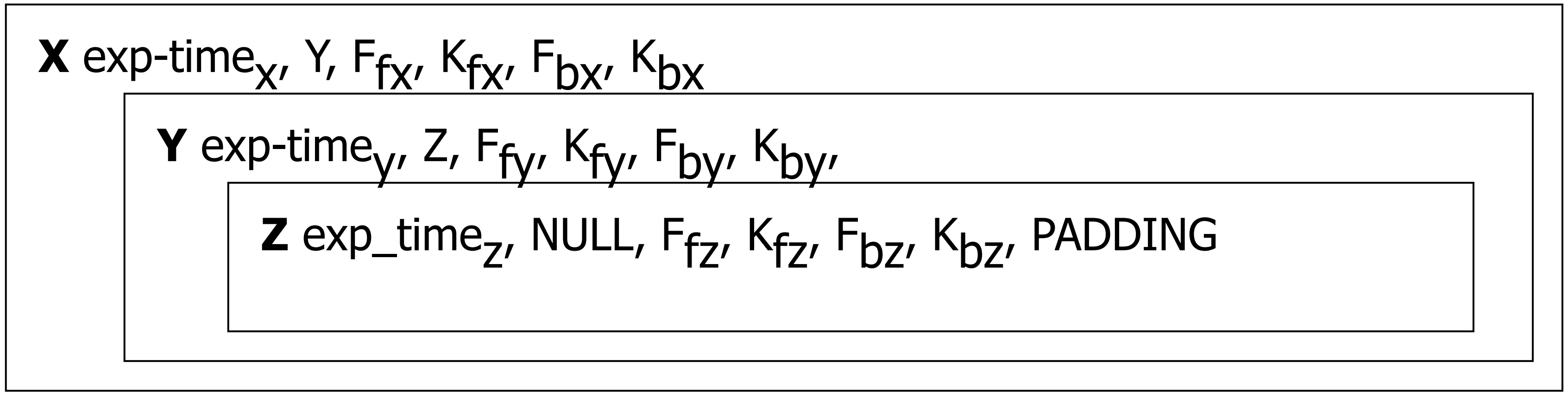
- **Sender anonymity against:**
  - end web servers
  - other Crowd members
  - eavesdroppers
- **Limitations:**
  - No protection against “global” attackers, timing/message length correlation attacks
  - Web server´s log may record submitting jondo´s IP address as the request originator´s address
  - Request contents are exposed to jondos on the path
  - Anonymising service can be circumvented by Java Applets, Active X controls
  - Performance overhead (increased retrieval time, network traffic and load on jondo machines)
  - No defend against DoS-attacks by malicious crowd members

Onion Routing

- Onion = Object with layers of public key encryption to produce anonymous bi-directional virtual circuit between communication partners and to distribute symmetric keys
- Initiator's proxy constructs "forward onion" which encapsulates a route to the responder
- (Faster) symmetric encryption for data communication via the circuit



Forward Onion for route W-X-Y-Z:


X exp-time$_x$, Y, F_{fx}, K_{fx}, F_{bx}, K_{bx}
Y exp-time$_y$, Z, F_{fy}, K_{fy}, F_{by}, K_{by},
Z exp_time$_z$, NULL, F_{fz}, K_{fz}, F_{bz}, K_{bz}, PADDING

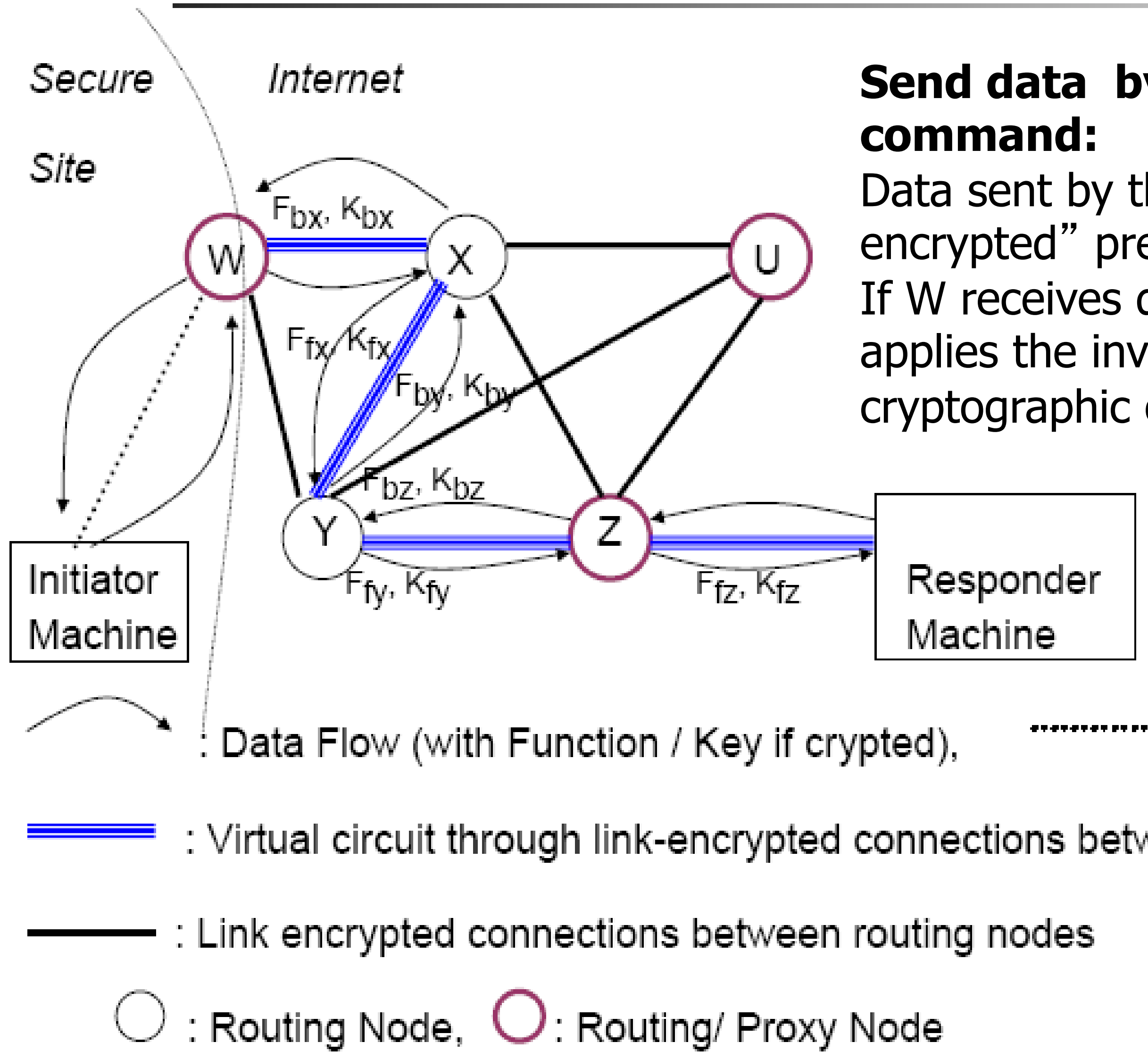
Each node N receives (PK_N = public key of node N):

- {exp-time, next-hop, F_f, K_f, F_b, K_b, payload} PK_N
- exp-time: expiration time
- next_hop: next routing node
- (F_f, K_f) : function / key pair for symmetric encryption of data moving forward in the virtual circuit
- (F_b, K_b) : function/key pair for symmetric encryption of data moving backwards in the virtual circuit
- payload: another onion (or null for responder´s proxy)

Virtual circuit creation and communication

- **Create command accompanies an Onion:** If node receives onion, it peels off one layer, keeps forward/ backward encryption keys, it chooses a virtual circuit (vc) identifier and sends create command+ vc identifier + (rest of) onion to next hop.
- It stores the vc identifier it receives and the one that it sent out as a pair.
- Until circuit is destroyed -> whenever it receives data on one connection, it sends it off to the other
- Forward encryption is applied to data moving in the forward direction, backward encryption is applied in the backward direction

Example: Virtual Circuit with Onion Routing



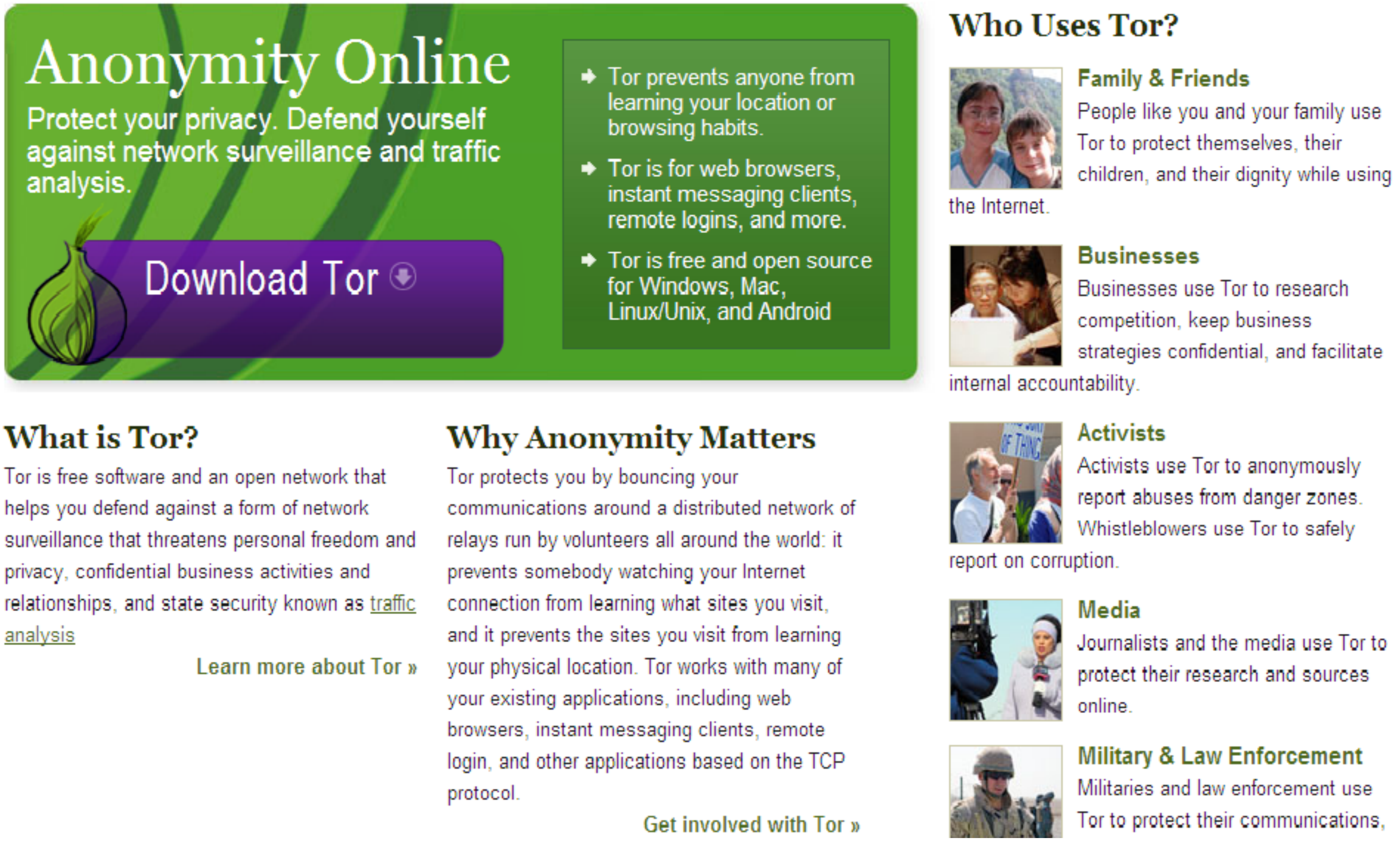
Secure
Site
Internet
W
X
U
Y
Z
F_{bx}, K_{bx}
F_{fx}, K_{fx}
F_{by}, K_{by}
F_{bz}, K_{bz}
F_{fy}, K_{fy}
F_{fz}, K_{fz}
Initiator Machine
Responder Machine
: Data Flow (with Function / Key if crypted),
: Unsecured Socket connection
: Virtual circuit through link-encrypted connections between routing nodes
: Link encrypted connections between routing nodes
: Routing Node,
: Routing/ Proxy Node

Send data by the use of send command:
Data sent by the initiator is ”pre-encrypted” prepeatedly by his proxy.
If W receives data sent back by Z, it applies the inverse of the backward cryptographic operations (outermost first).

Onion Routing - Review

- Functionality:
  - Hiding of routing information in connection oriented communication relations
  - Nested public key encryption for building up virtual circuit
  - Expiration_time field reduces costs of replay detection
  - Dummy traffic between Mixes (Onion Routers)
- Limitations:
  - First/Last-Hop Attacks by
    - Timing correlations
    - Message length (No. of cells sent over circuit)

TOR (2nd Generation Onion Router – www.torproject.org)



Anonymity Online

Protect your privacy. Defend yourself against network surveillance and traffic analysis.

Download Tor

- Tor prevents anyone from learning your location or browsing habits.
- Tor is for web browsers, instant messaging clients, remote logins, and more.
- Tor is free and open source for Windows, Mac, Linux/Unix, and Android

What is Tor?

Tor is free software and an open network that helps you defend against a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security known as traffic analysis

Learn more about Tor »

Why Anonymity Matters

Tor protects you by bouncing your communications around a distributed network of relays run by volunteers all around the world: it prevents somebody watching your Internet connection from learning what sites you visit, and it prevents the sites you visit from learning your physical location. Tor works with many of your existing applications, including web browsers, instant messaging clients, remote login, and other applications based on the TCP protocol.

Get involved with Tor »

Who Uses Tor?

Family & Friends
People like you and your family use Tor to protect themselves, their children, and their dignity while using the Internet.

Businesses
Businesses use Tor to research competition, keep business strategies confidential, and facilitate internal accountability.

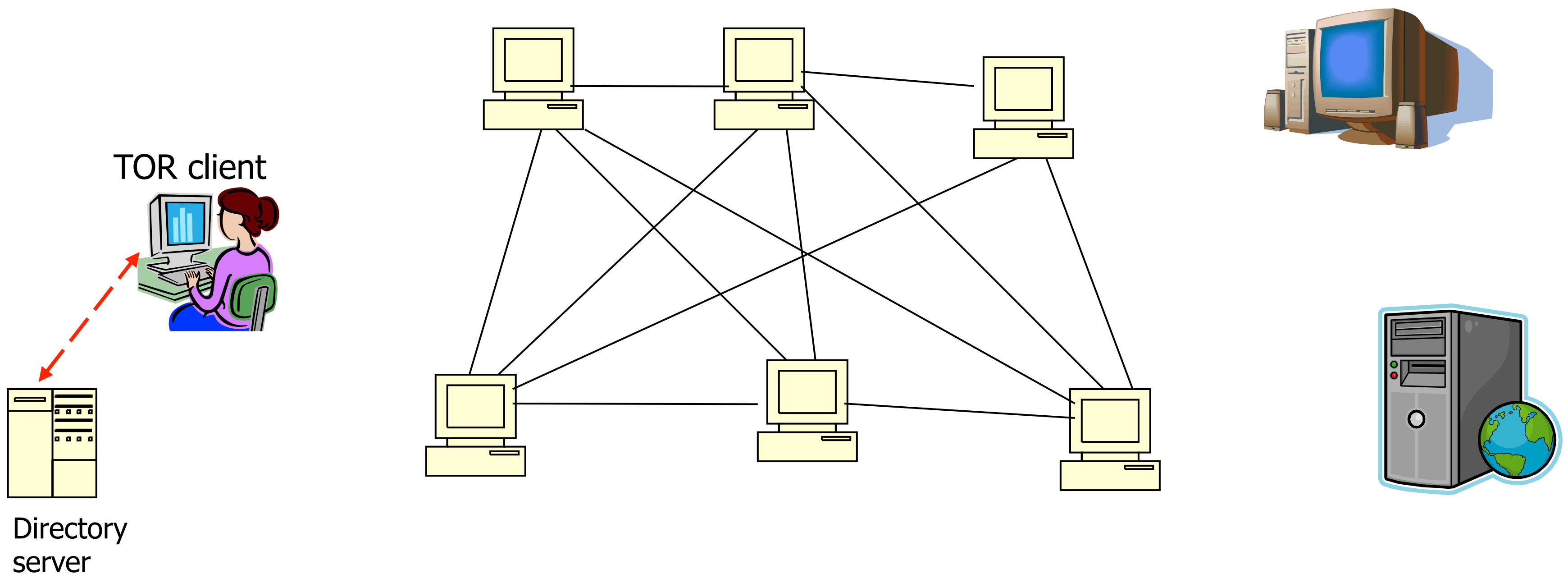
Activists
Activists use Tor to anonymously report abuses from danger zones. Whistleblowers use Tor to safely report on corruption.

Media
Journalists and the media use Tor to protect their research and sources online.

Military & Law Enforcement
Militaries and law enforcement use Tor to protect their communications,

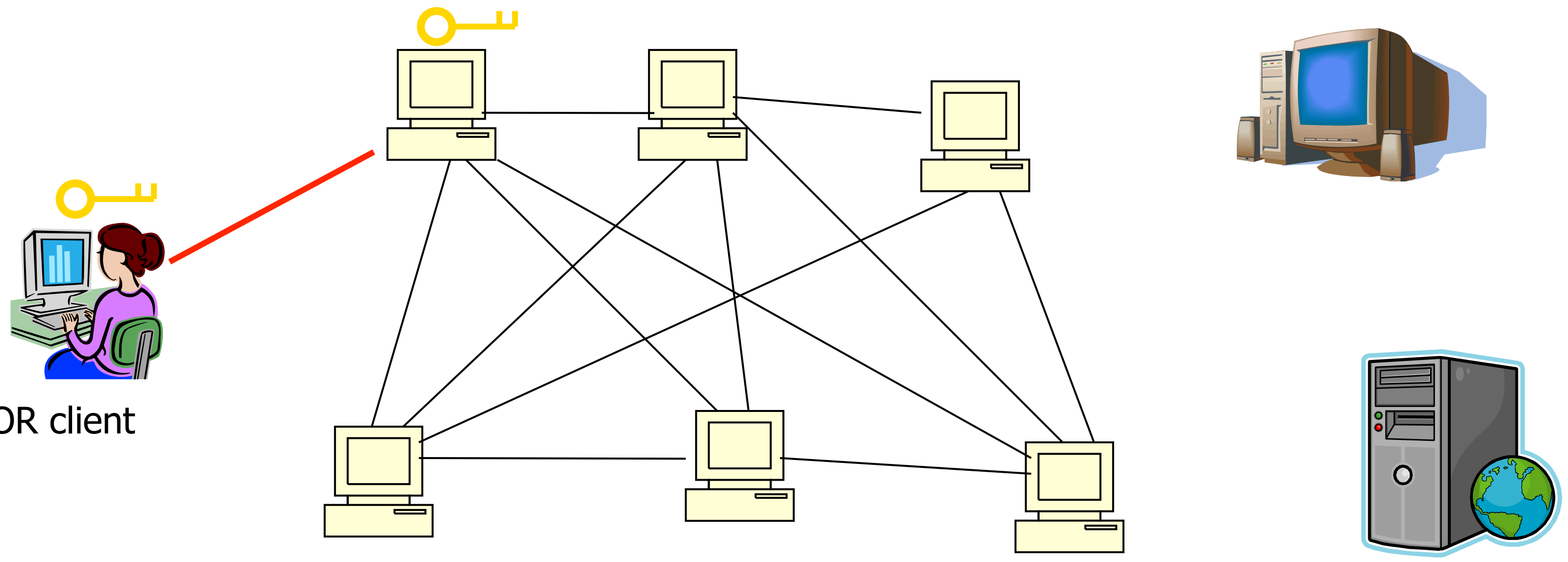
First Step

- TOR client obtains a list of TOR nodes from a directory server
- Directory servers maintain list of which onion routers are up, their locations, current keys, exit policies, etc.



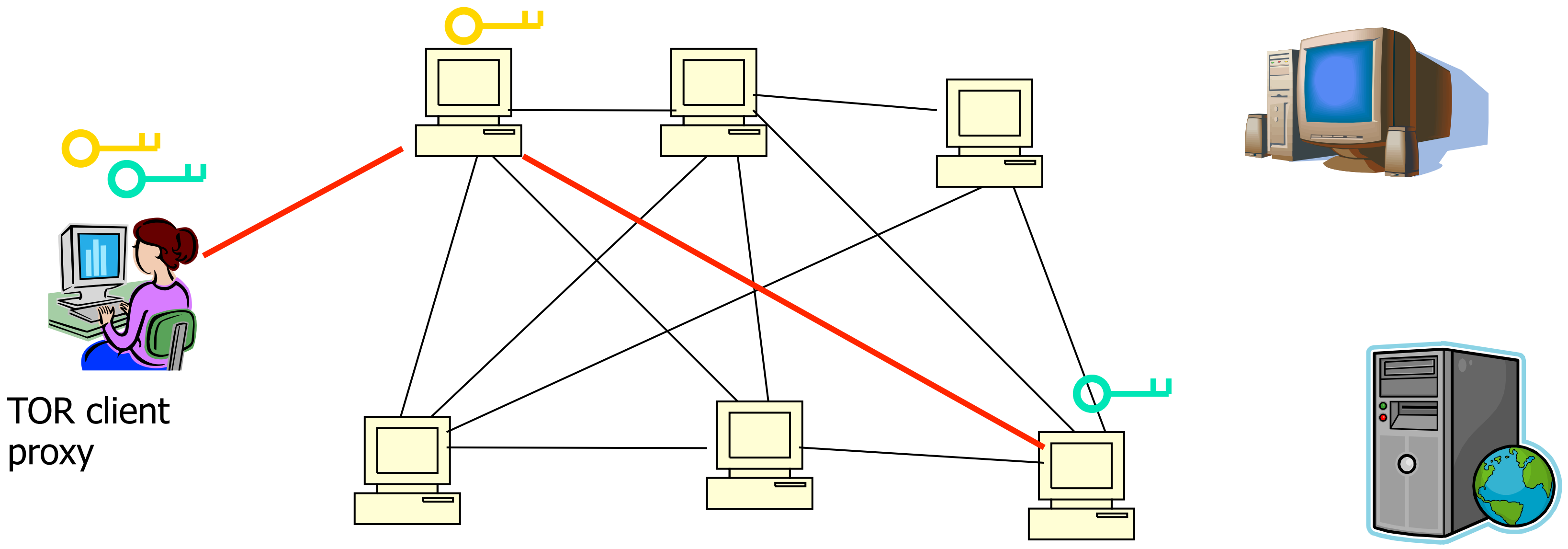
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1



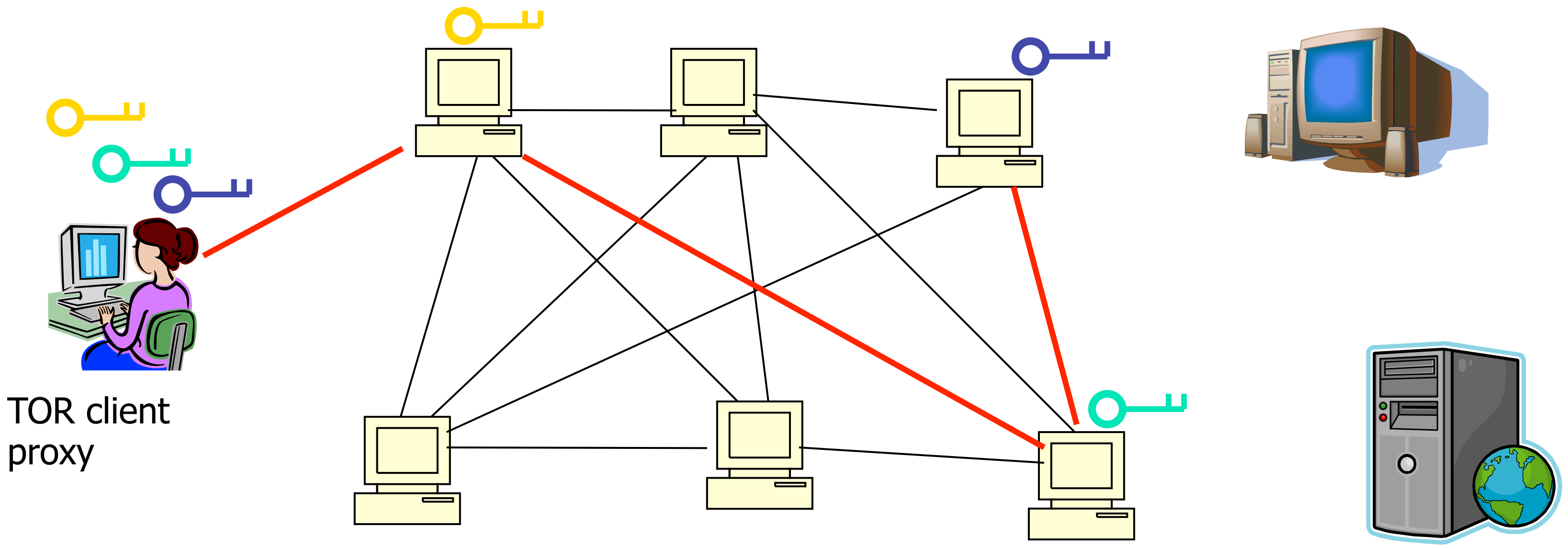
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2



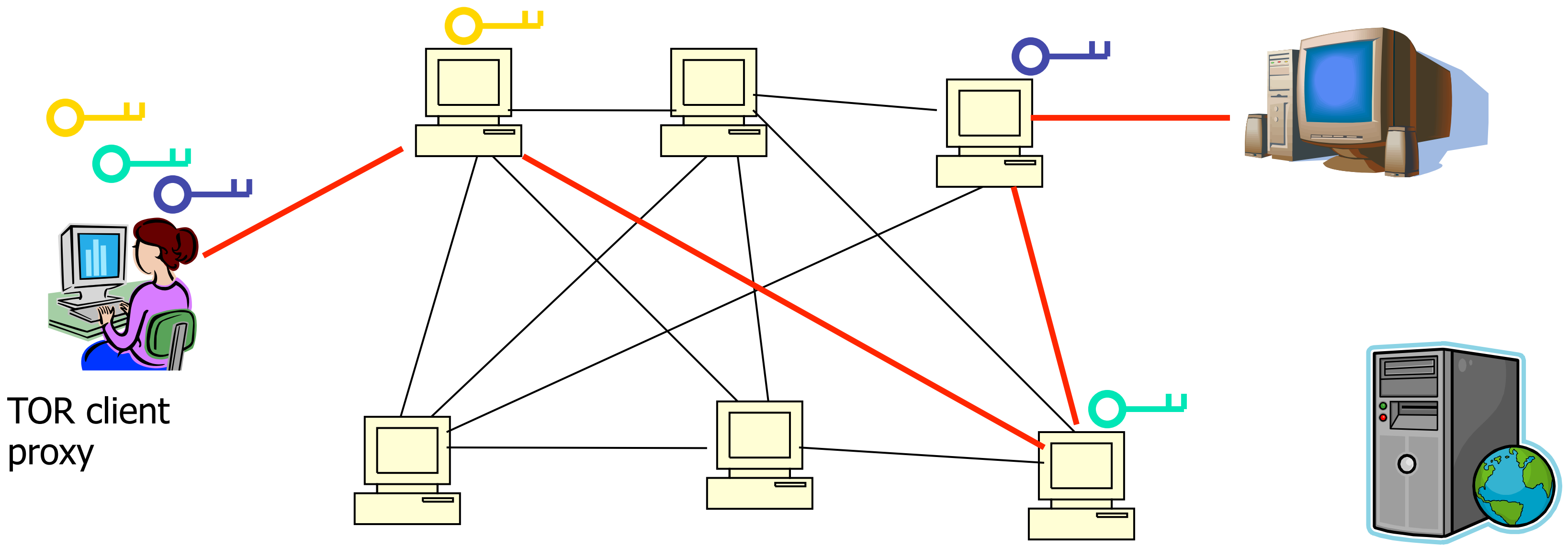
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.



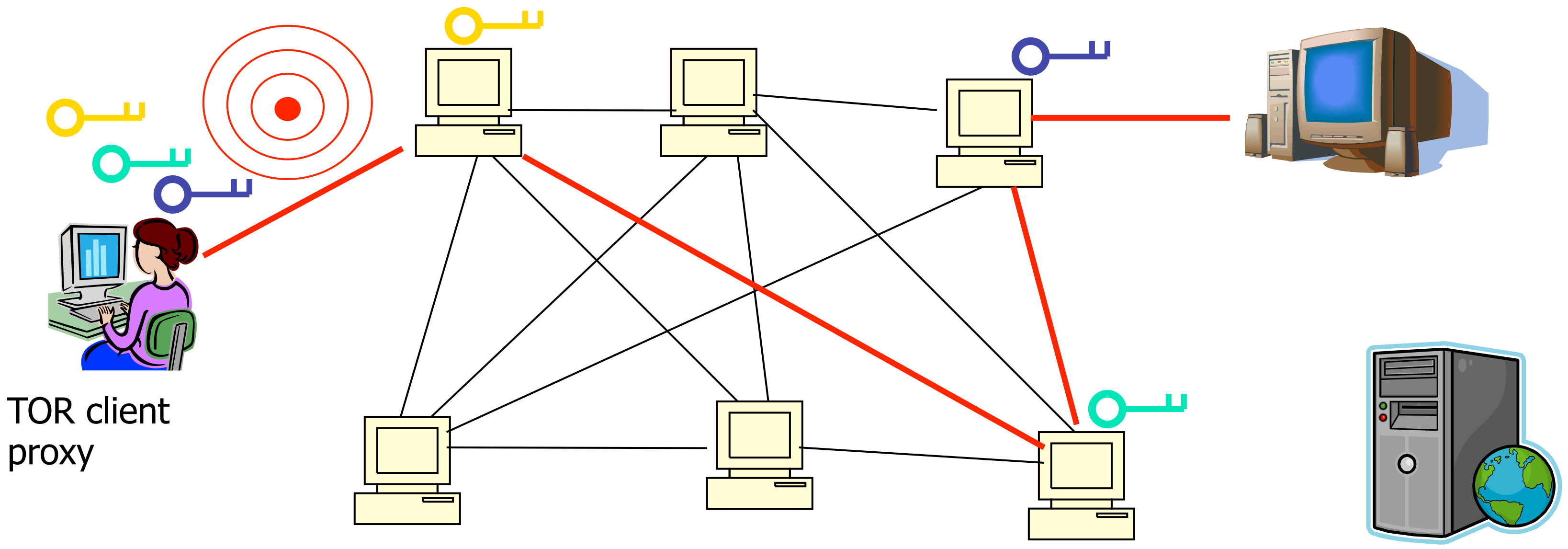
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



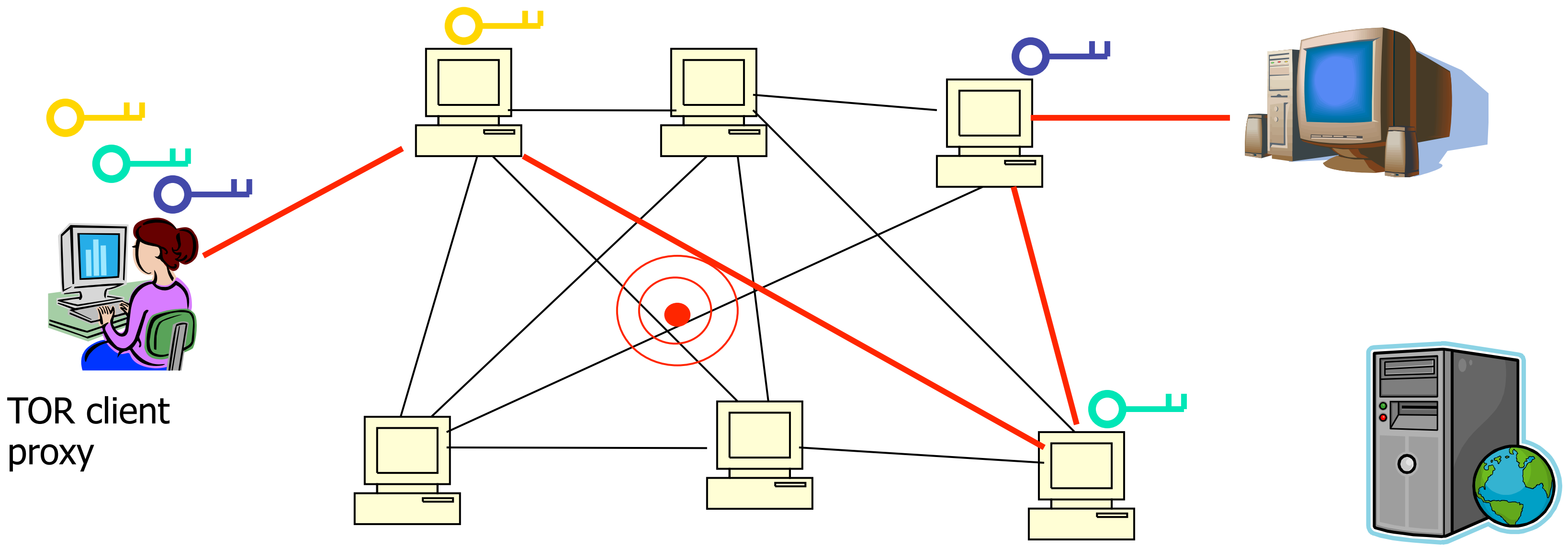
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



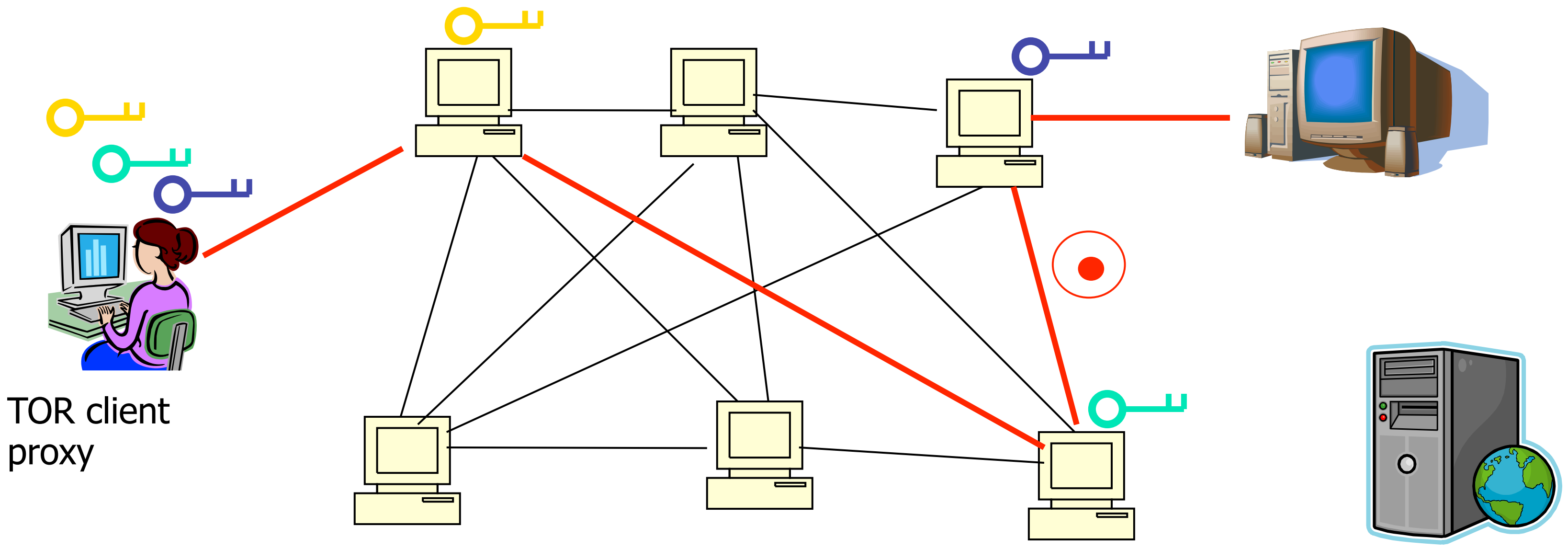
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



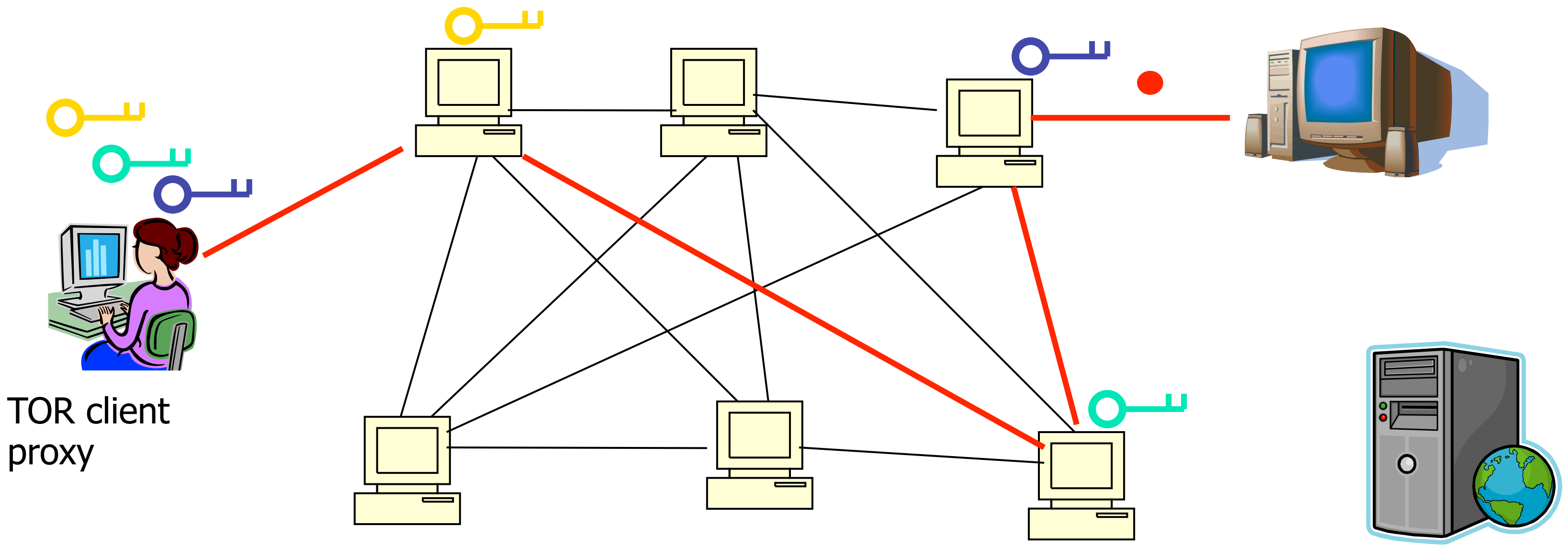
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



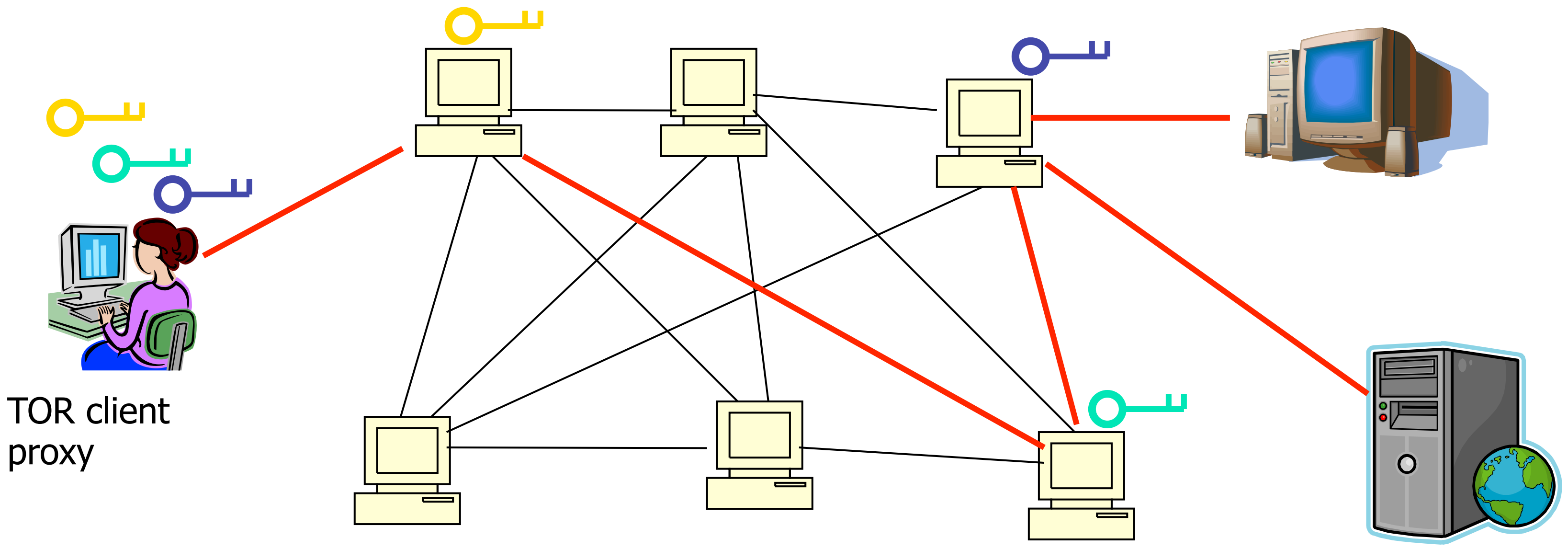
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



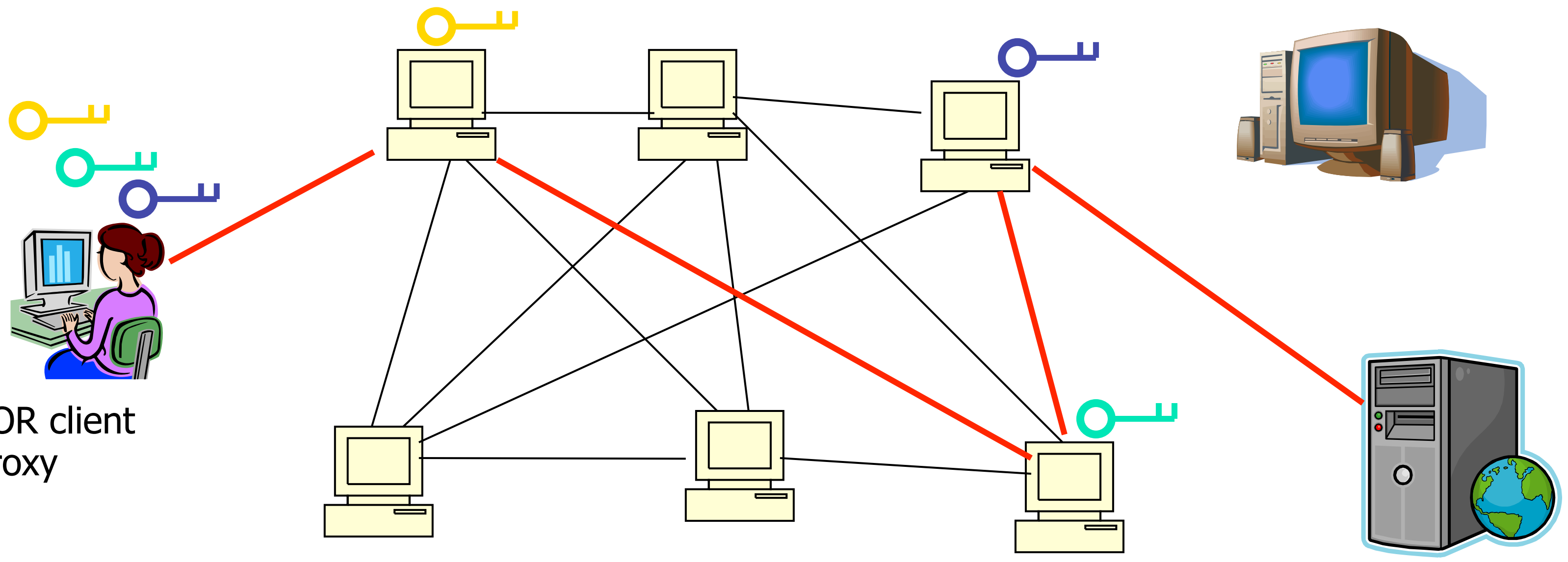
TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit

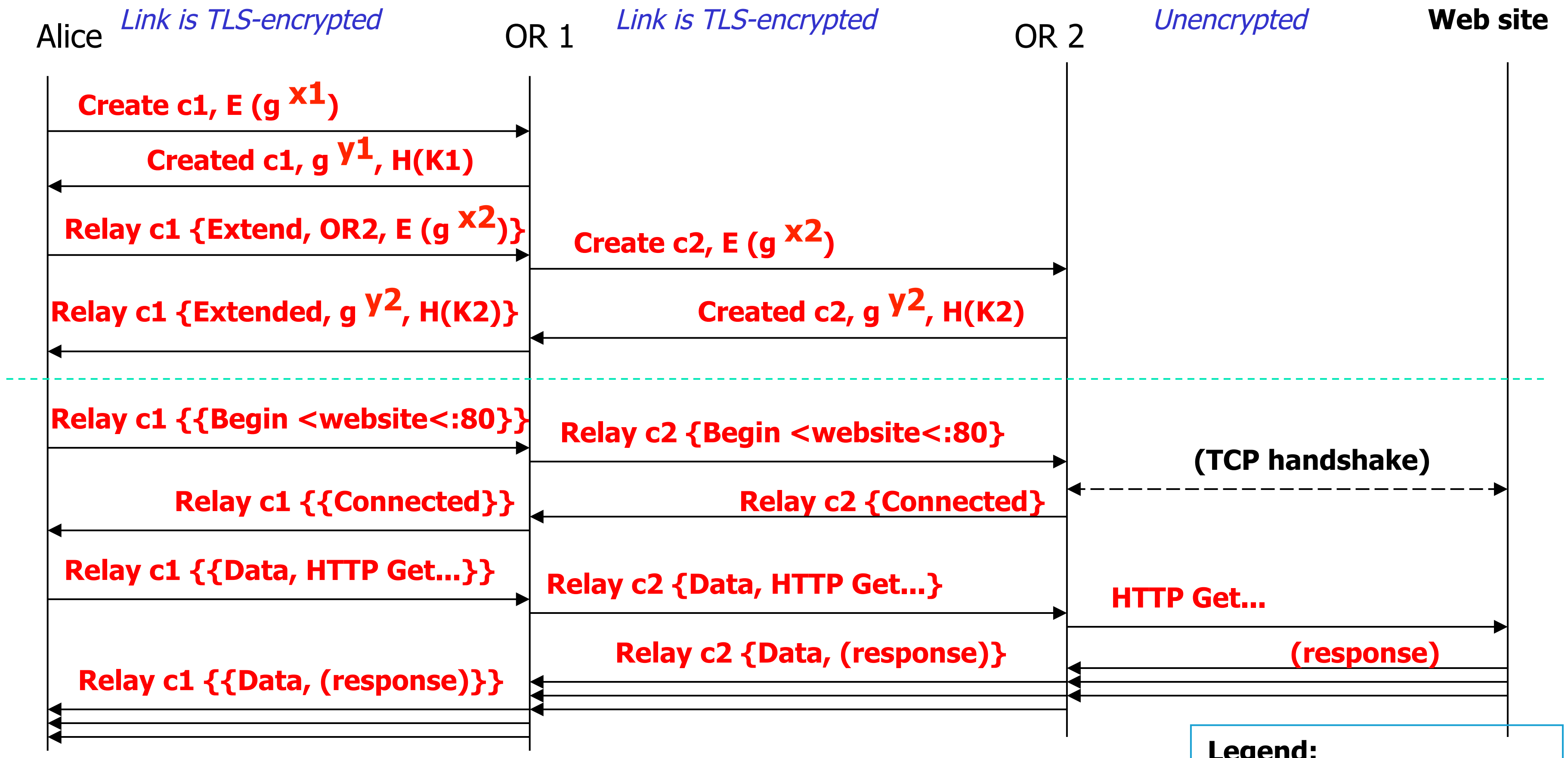


TOR circuit setup

- Client proxy establishes key + circuit with Onion Router 1
- Proxy tunnels through that circuit to extend to Onion Router 2
- Etc.
- Client applications connect and communicate over TOR circuit



TOR: Building up a two-hop circuit and fetching a web page



Legend:
E(x): RSA encryption
{X}: AES encryption
cN: a circuit ID

TOR - Review

- Some improvemnets in comparision with Onion Routing:
  - Perfect forward secrecy
  - Resistant to replay attacks
  - Many TCP streams can share one circuit
  - Seperation of ”protocol cleaning” from anonymity:
    - Standard SOCKS proxy interface (instead of having a seperate application proxy for each application)
    - Content filtering via Privoxy
  - Directory servers
  - Variable exit policies
  - End-to-end integrity checking
  - Hidden services
- Still vulnerable to end-to-end timing and size correlations

Further reading

- Andreas Pfitzmann, Marit Hansen, Anonymity. Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology, Version v0.31,Feb. 15, 2008. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.doc#_Toc64643839.

- Andreas Pfitzmann et al. "Communication Privacy", in: Aquisti et al. (Eds.), Digital Privacy – Theory, Technologies, and Practices, Auerbach Publications, 2008

- D.Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", Communications of the ACM, 24 (2). 1981, pp. 84-88, http://world.std.com/~franl/crypto/chaum-acm-1981.html

- P. Syverson, D. Goldschlag, M. Reed, "Anonymous Connections and Onion Routing", Proceedings of the 1997 Symposium on Security and Privacy, Oakland, 1997, http://www.itd.nrl.navy.mil/ITD/5540/projects/onion-routing/OAKLAND_97.ps , http://www.onion-router.net/Publications.html

- Roger Dingledine and Nick Mathewson, The Free Haven Project; Paul Syverson, Naval Research Lab, "Tor: The Second-Generation Onion Router", 13th USENIX Security Symposium, 2004, http://static.usenix.org/event/sec04/tech/full_papers/dingledine/dingledine.pdf

- M.Reiter, A.Rubin, "Anonymous Web Transactions with Crowds", Communications of the ACM, Vol.42, No.2, February 1999, pp. 32-38.

- , Simone Fischer-Hübner, "IT-Security and Privacy - *Design and Use of Privacy-Enhancing Security Mechanisms*", Springer Scientific Publishers, Lecture Notes of Computer Science, LNCS 1958, May 2001, ISBN 3-540-42142-4 (chapter 4)

Repetition: Diffie-Hellman Key exchange

<u>Global Public Elements:</u>

q: prime number

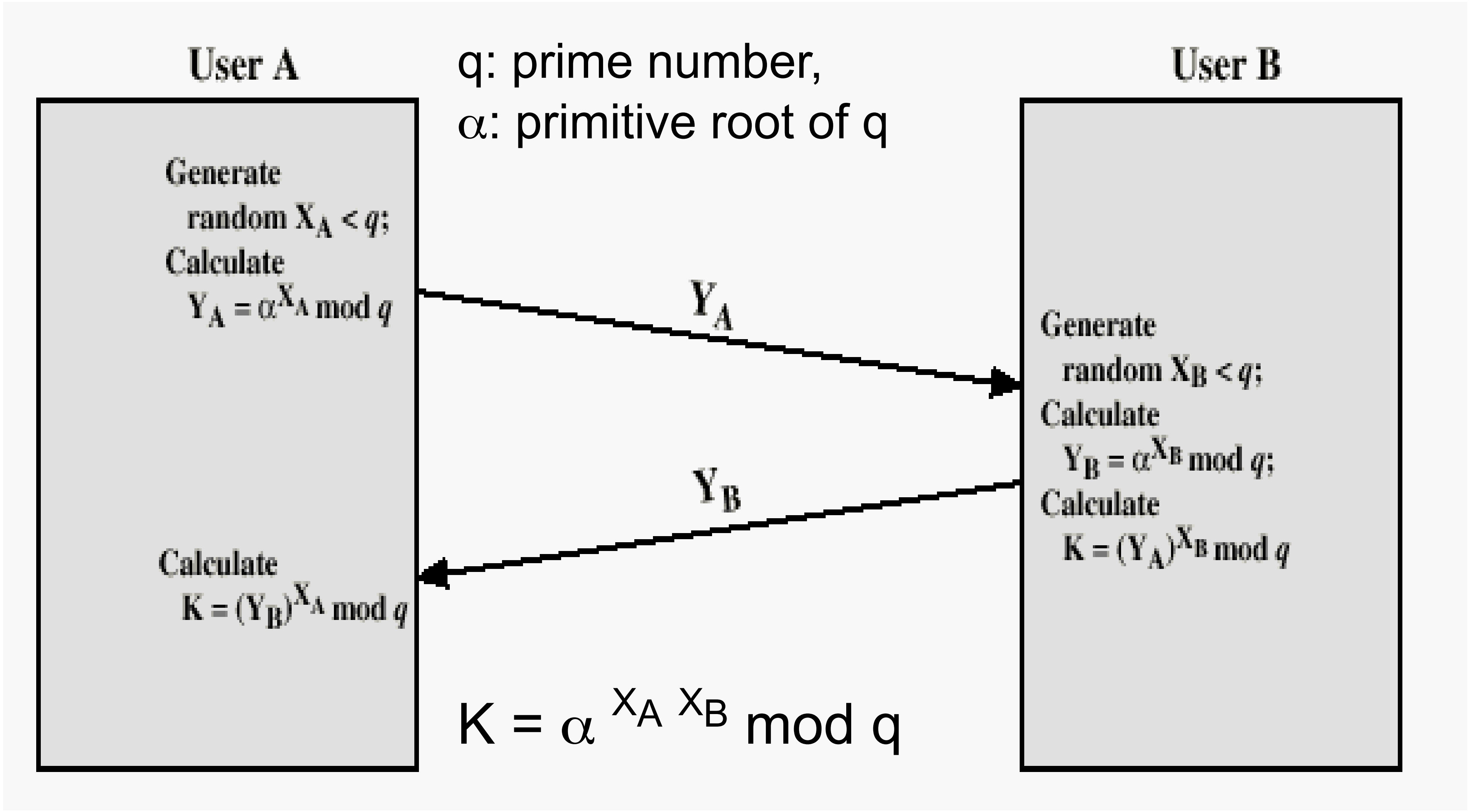
α: $\alpha < q$ and α is a primitive root of q

[If α is a primitive root of prime number p, then the numbers:
α mod p, α^2 mod p,…, α^{p-1} mod p
are distinct and are a permutation of {1..p-1}.

For any integer b<p, primitive root α of prime number p, one can find unique exponent i (discrete logarithm),
such that b= α^i mod p, $0 \leq i \leq (p-1)$

For larger primes, calculating discrete logarithms is considered as practically infeasible]

Diffie-Hellman Key Exchange



User A
q: prime number,
α: primitive root of q
User B
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate $K = (Y_B)^{X_A} \bmod q$

$$K = \alpha^{X_A X_B} \bmod q$$