cybersecurity centre

GCHQ Academic Centre of Excellence EPSRC

Cybersecurity Southampton

CyberSecuritySoton.org [w]

@CybSecSoton [fb & tw]

UNIVERSITY OF Southampton

COMP 6224 (2016)
week 4: Tor, Onion Routing & Anonymity Networks

Vladimiro Sassone

Cyber Security Centre

University of Southampton

Part I

Anonymity

motivations

- Protecting ***sensitive*** and ***confidential*** data is one of the fundamental issues in cyber security and social applications.

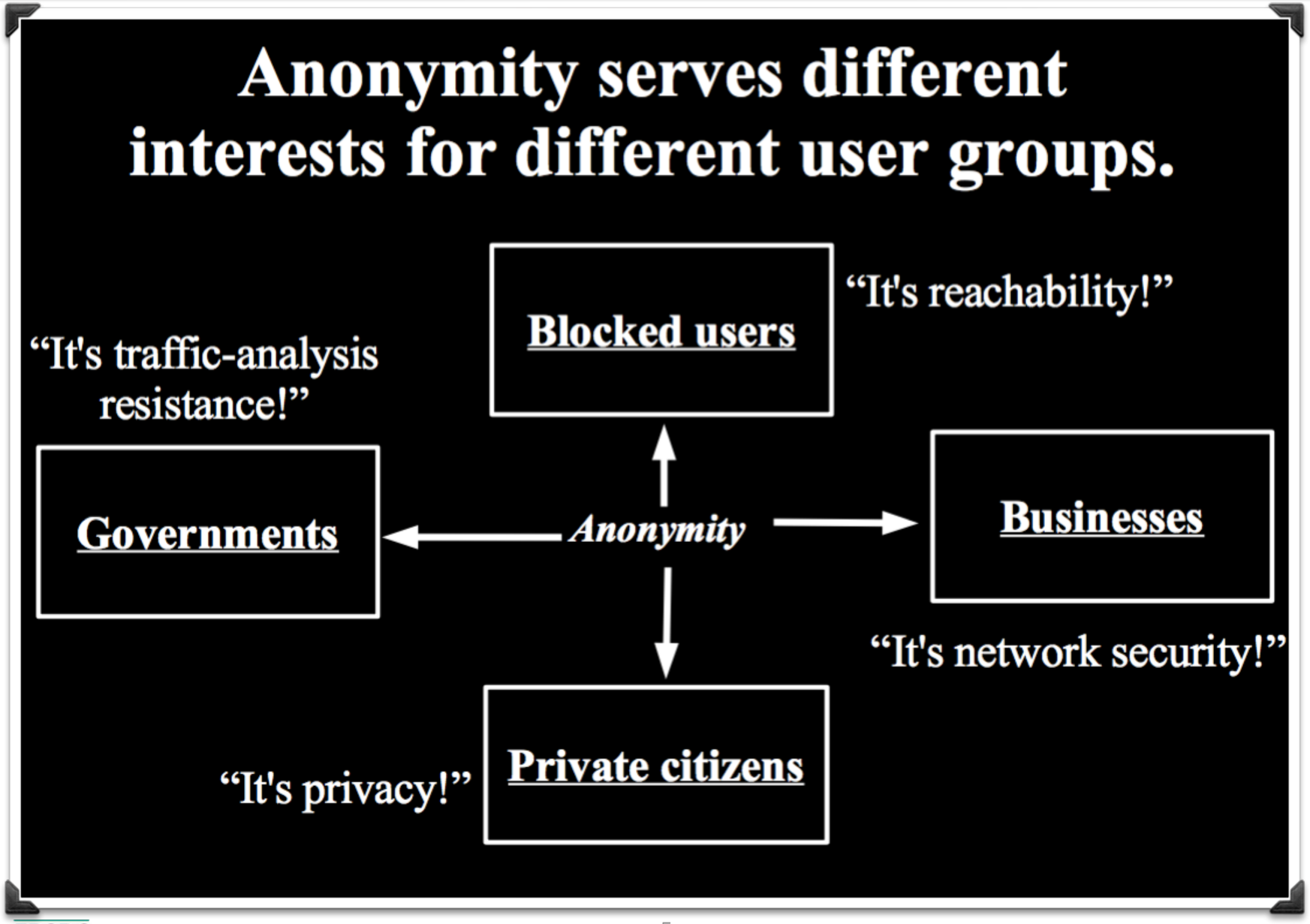
- Becoming increasingly important in many fields of human activities:
  - electronic communication, auction, payment, voting, etc…

- Many systems for protecting confidential information have been proposed in the literature.

motivations

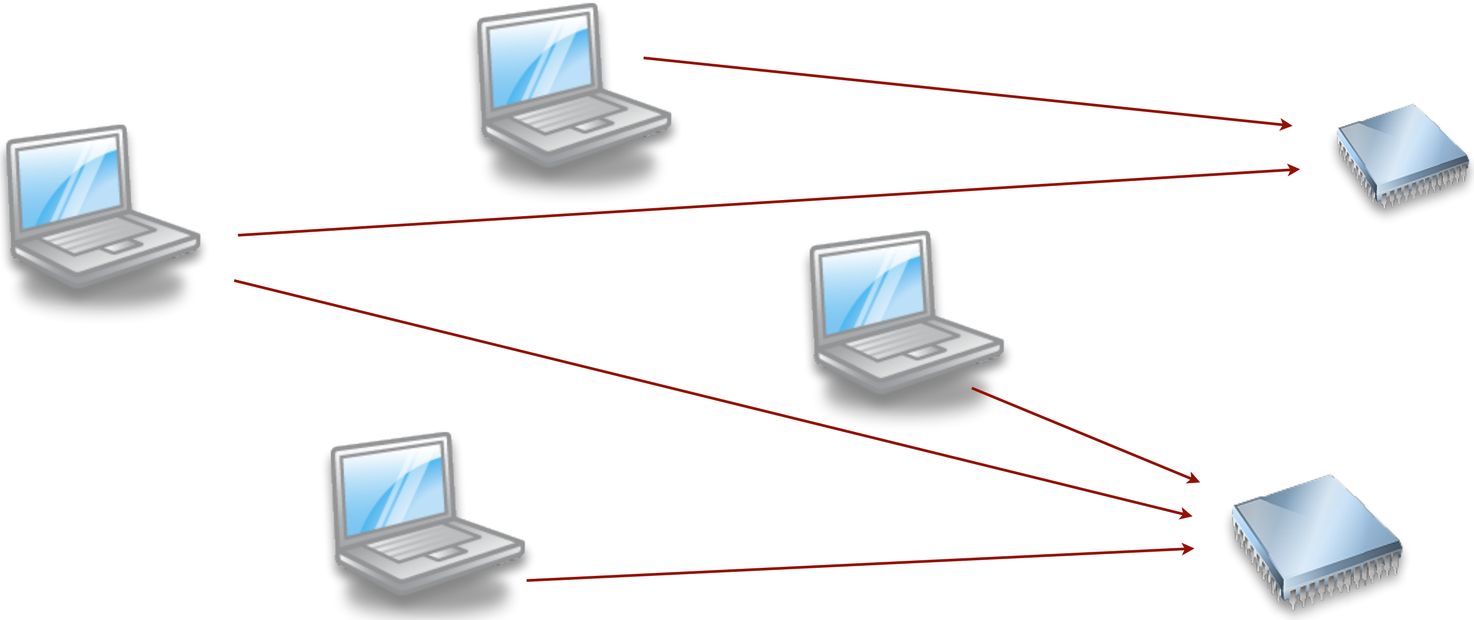
- Aims at obfuscating the link between private input (anonymous actions) and public (observable) output
- Attacker tries to infer the hidden info from his observations of protocols and activities
- Much in the same way as: Information Flow; Hypothesis Testing; channels as in Information Theory; Untraceability. In fact such communities are converging over the methods
- Here mainly in the sense of privacy & information hiding

beneficiaries



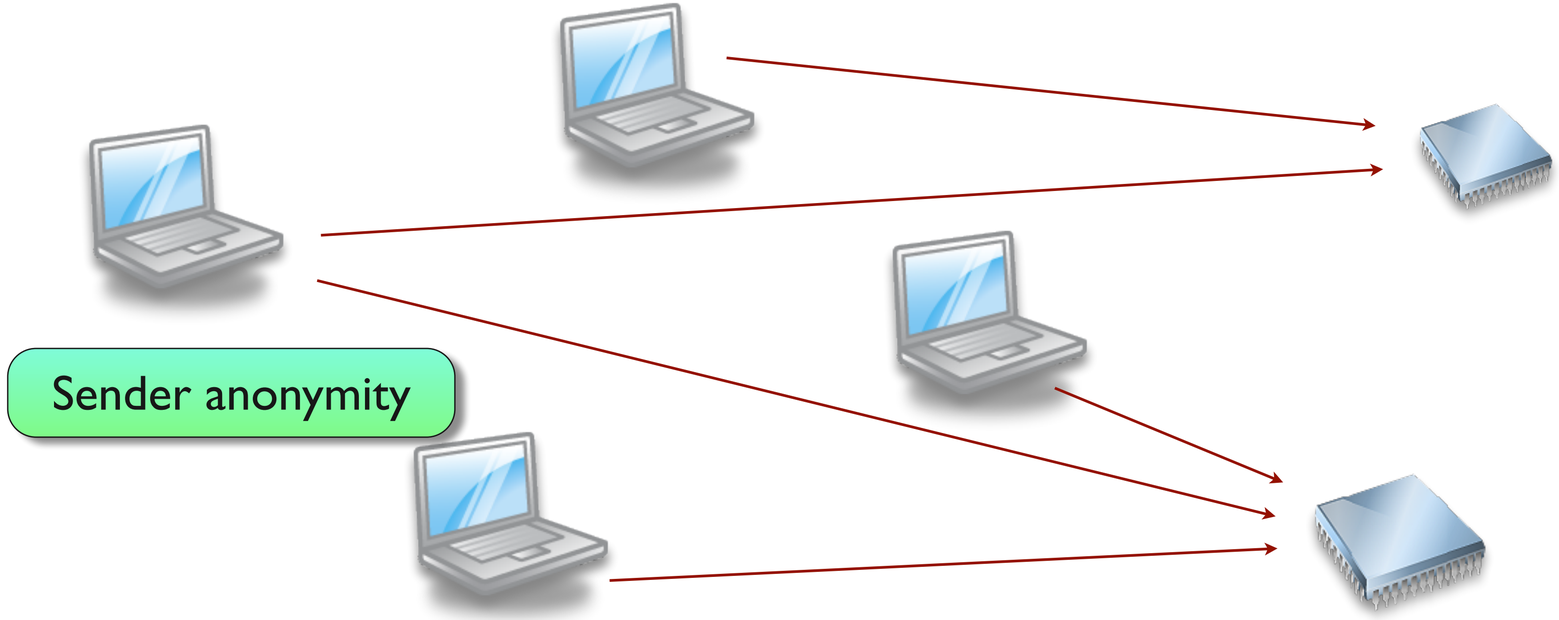
anonymity networks

networks of users that can hide information about users IDs and other parameters which might identify them.



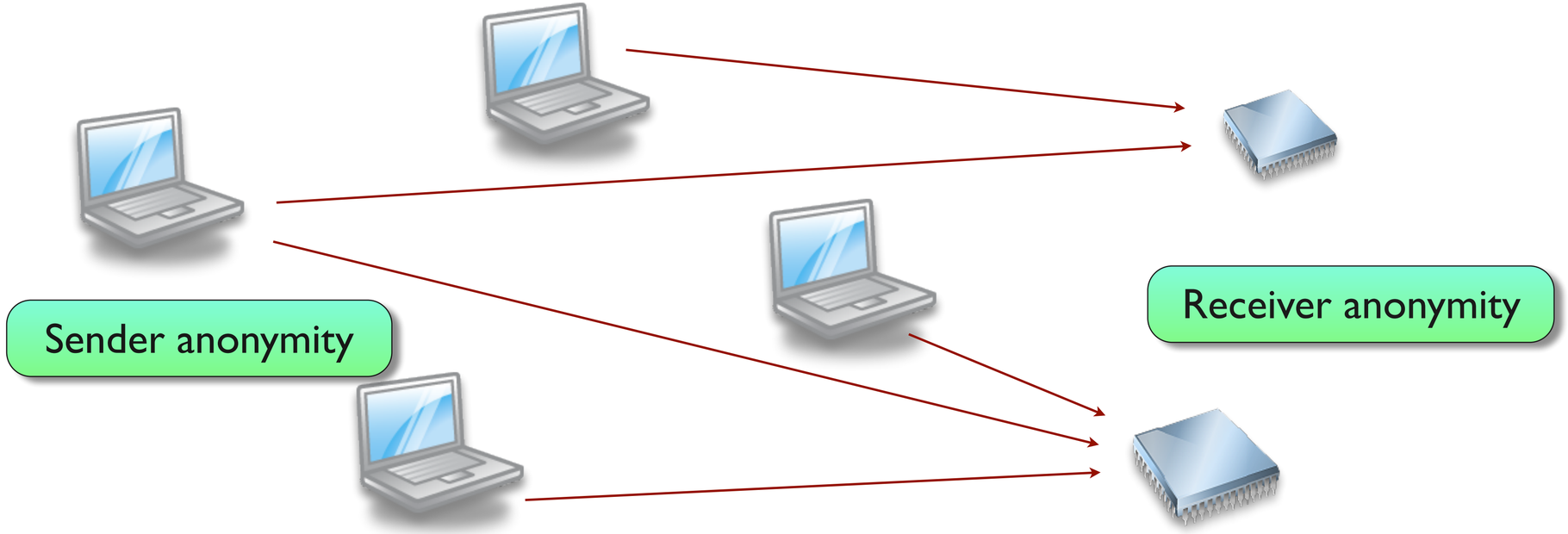
anonymity networks

networks of users that can hide information about users IDs and other parameters which might identify them.

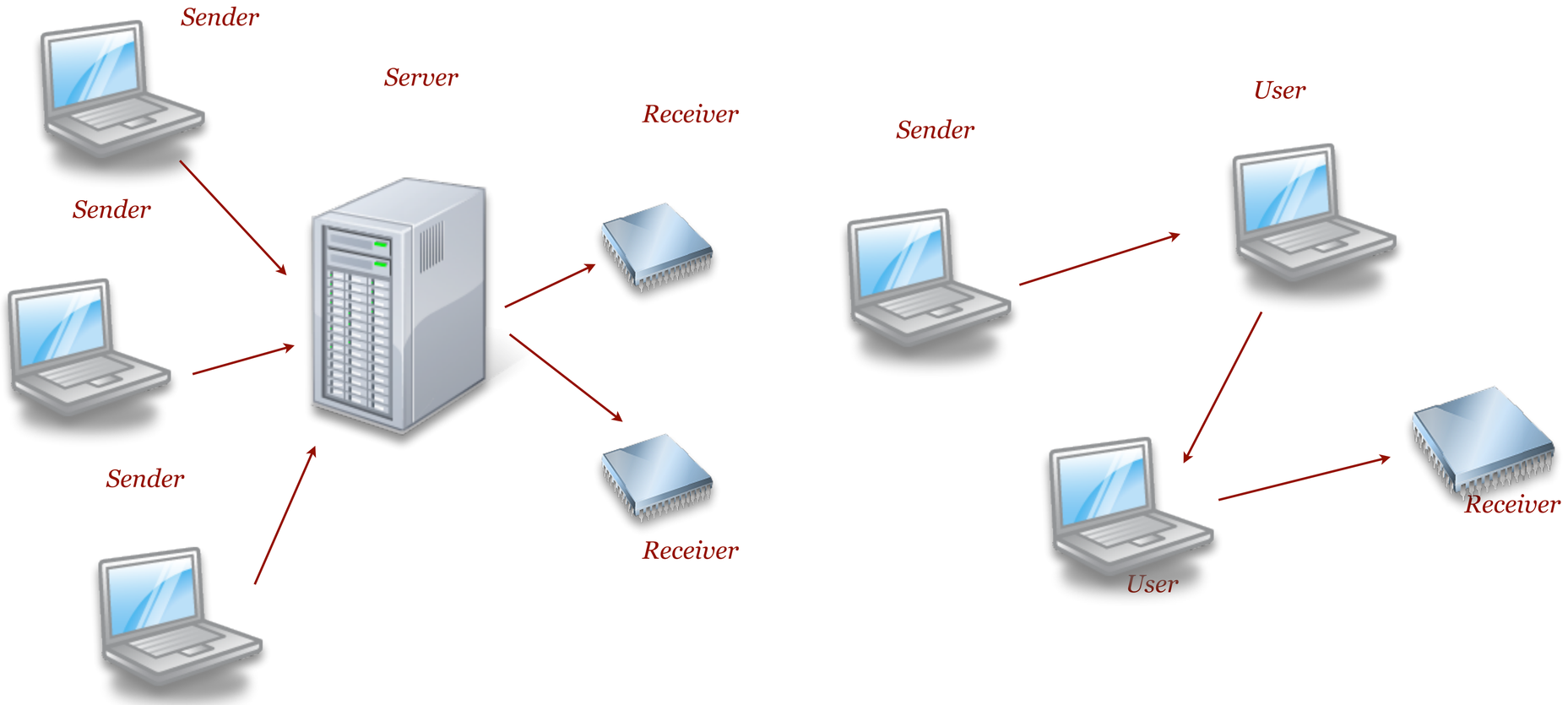


anonymity networks

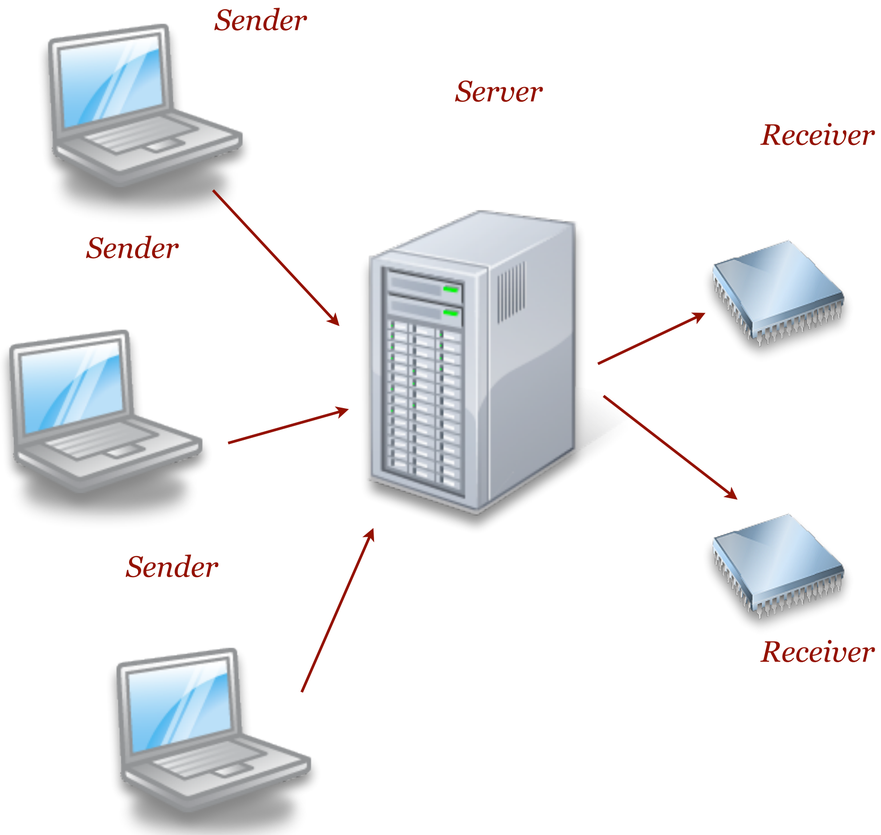
networks of users that can hide information about users IDs and other parameters which might identify them.

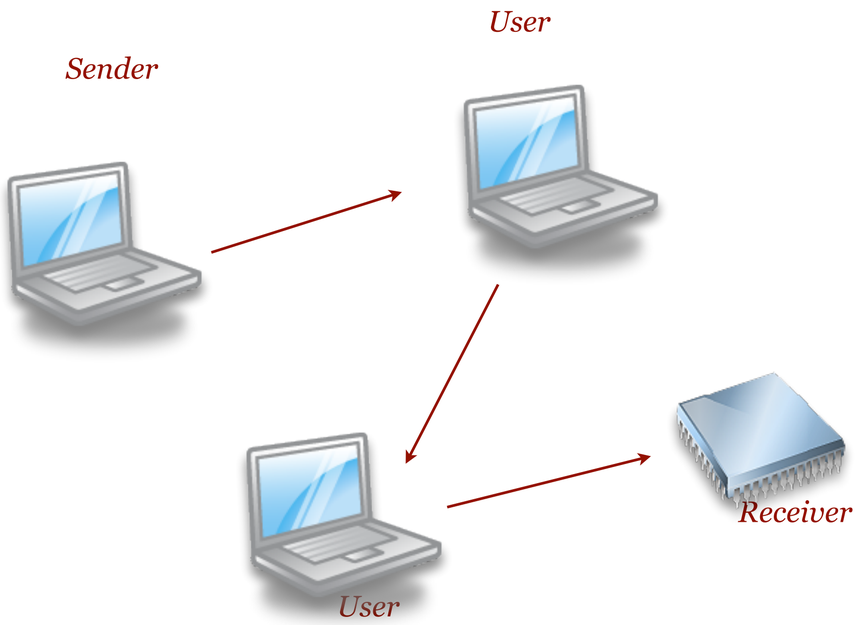


anonymity protocols



anonymity protocols





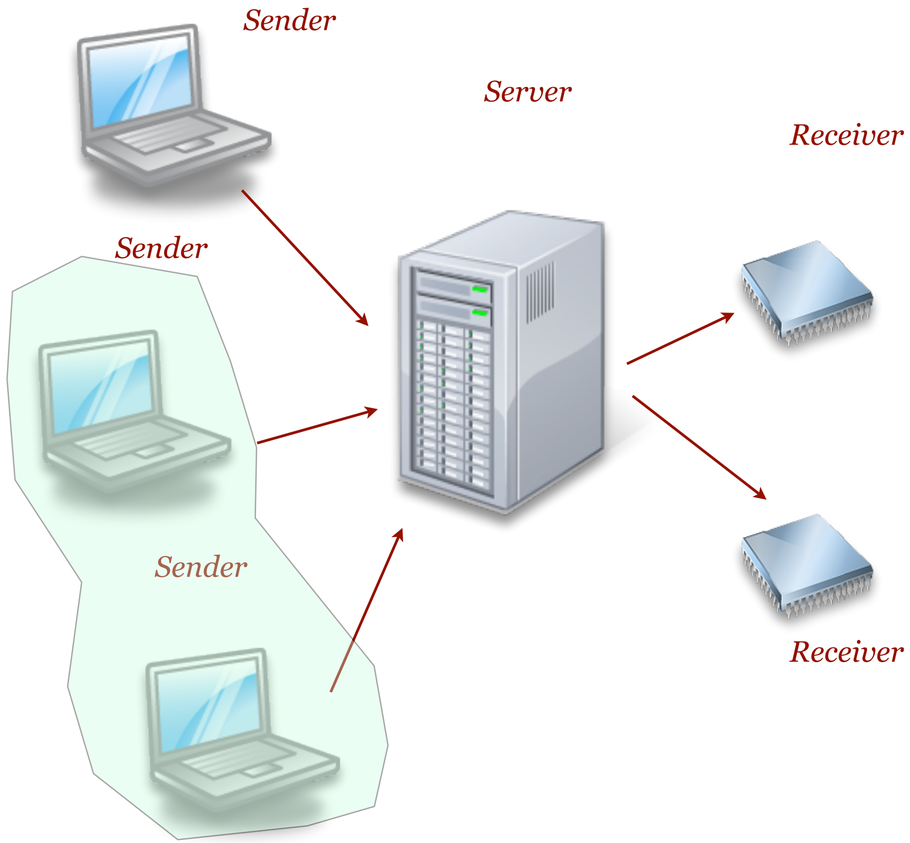
Mix network

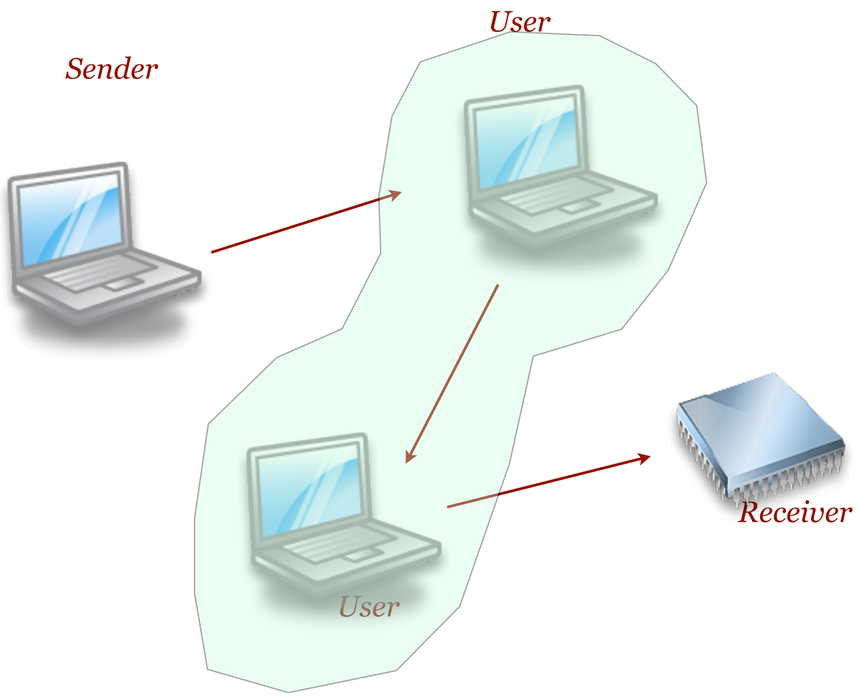
[chaum 1980]

Crowds

[reiter & rubin 1998]

anonymity protocols





Mix network

[chaum 1980]

Crowds

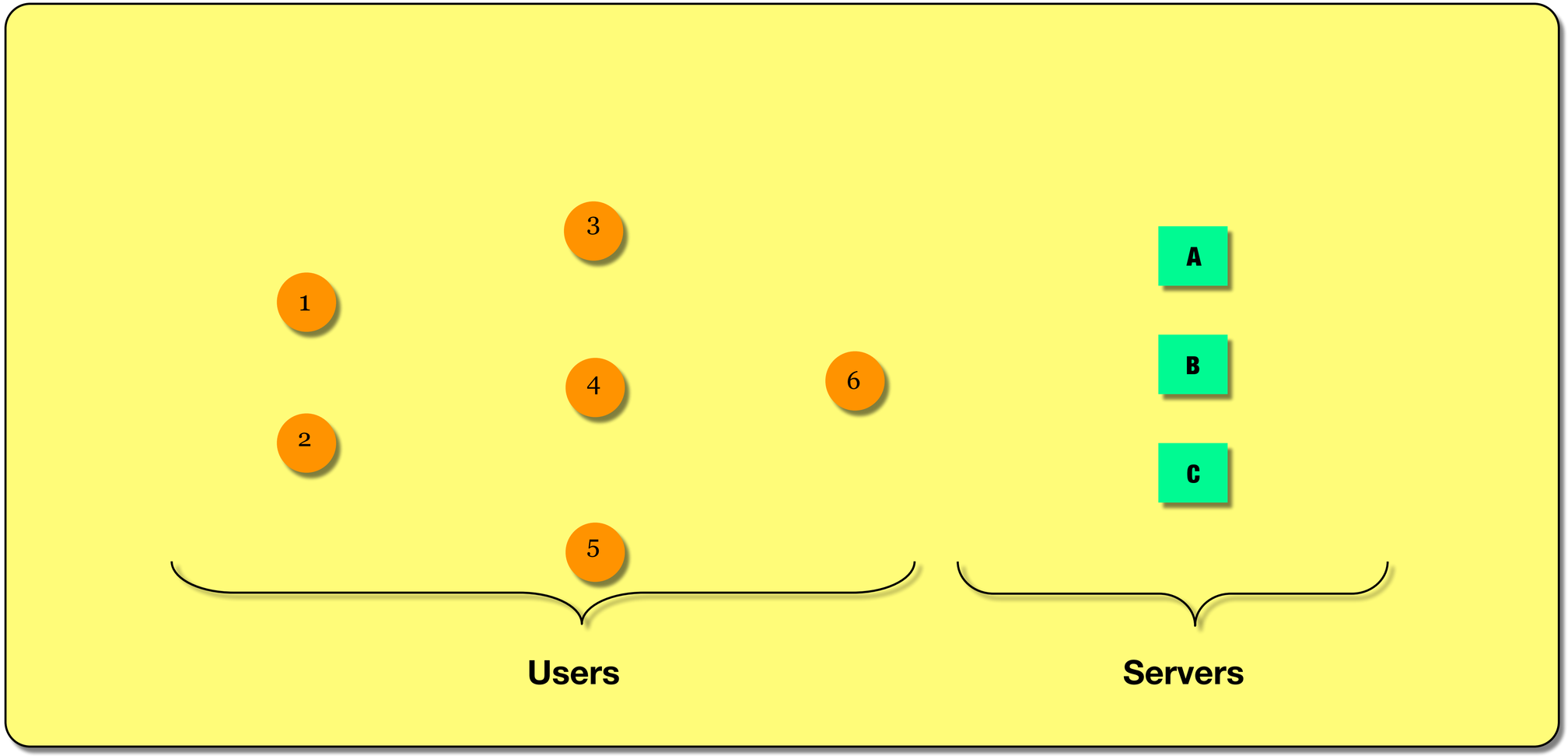
[reiter & rubin 1998]

crowds: the protocol

UNIVERSITY OF Southampton

Allows internet users to perform anonymous web transactions

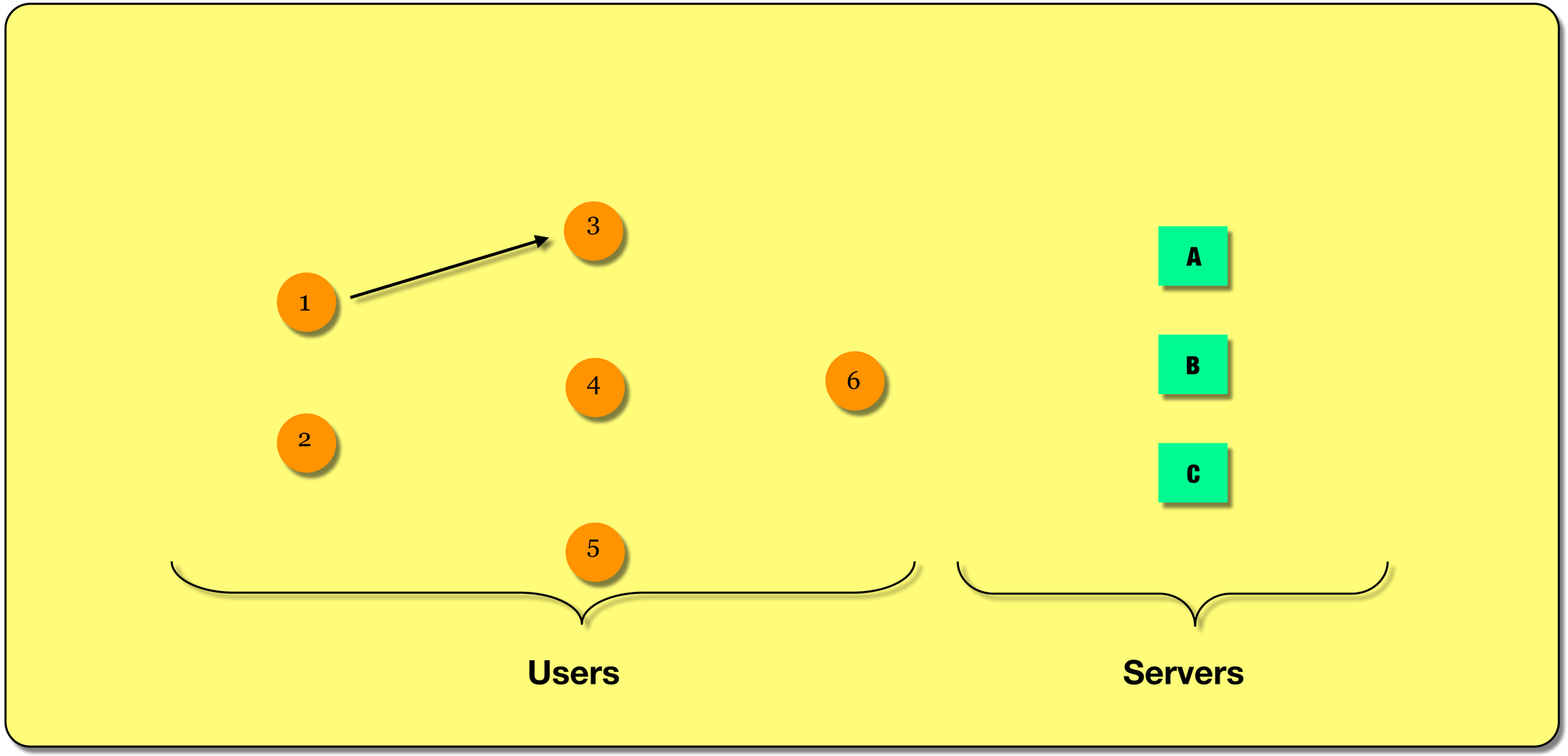
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

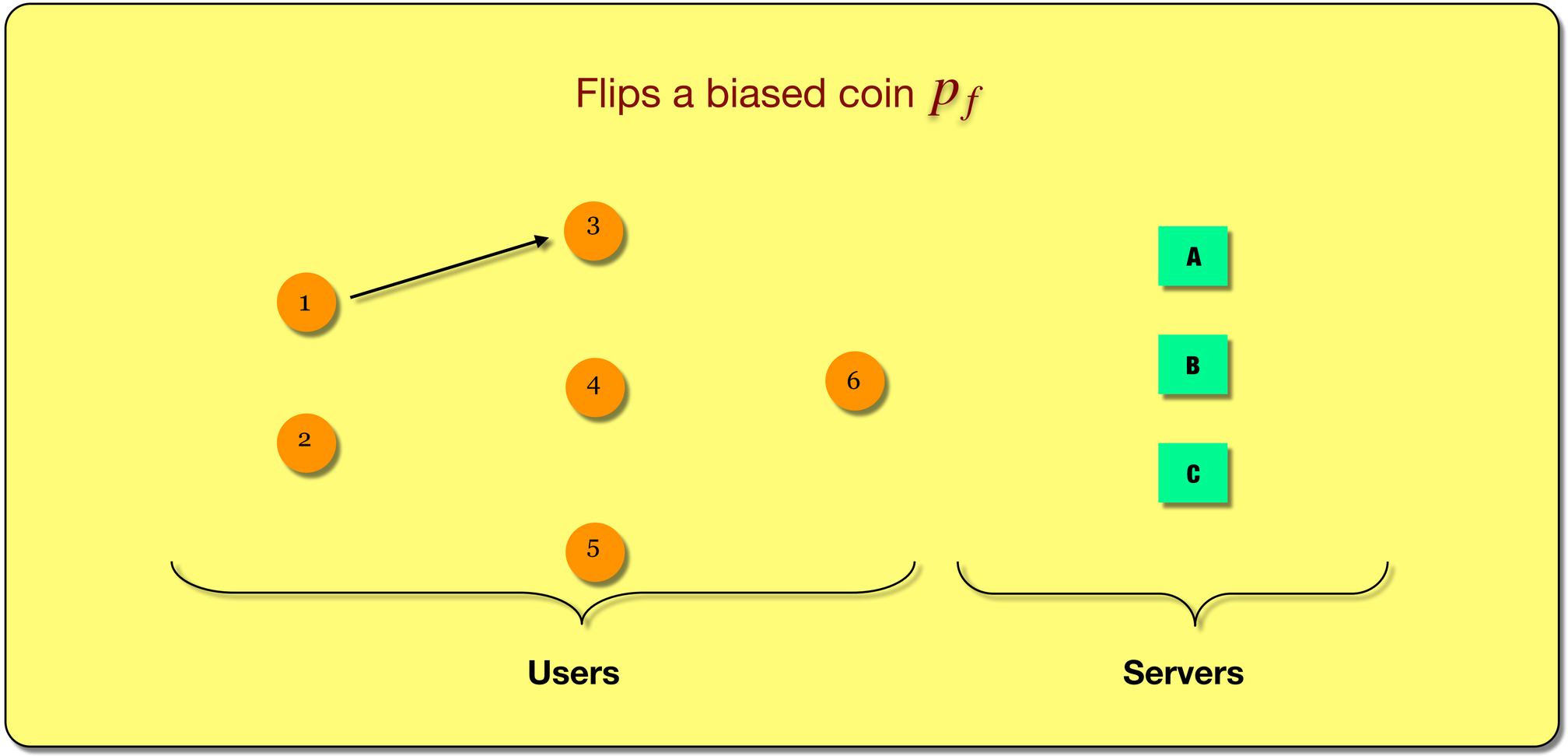
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

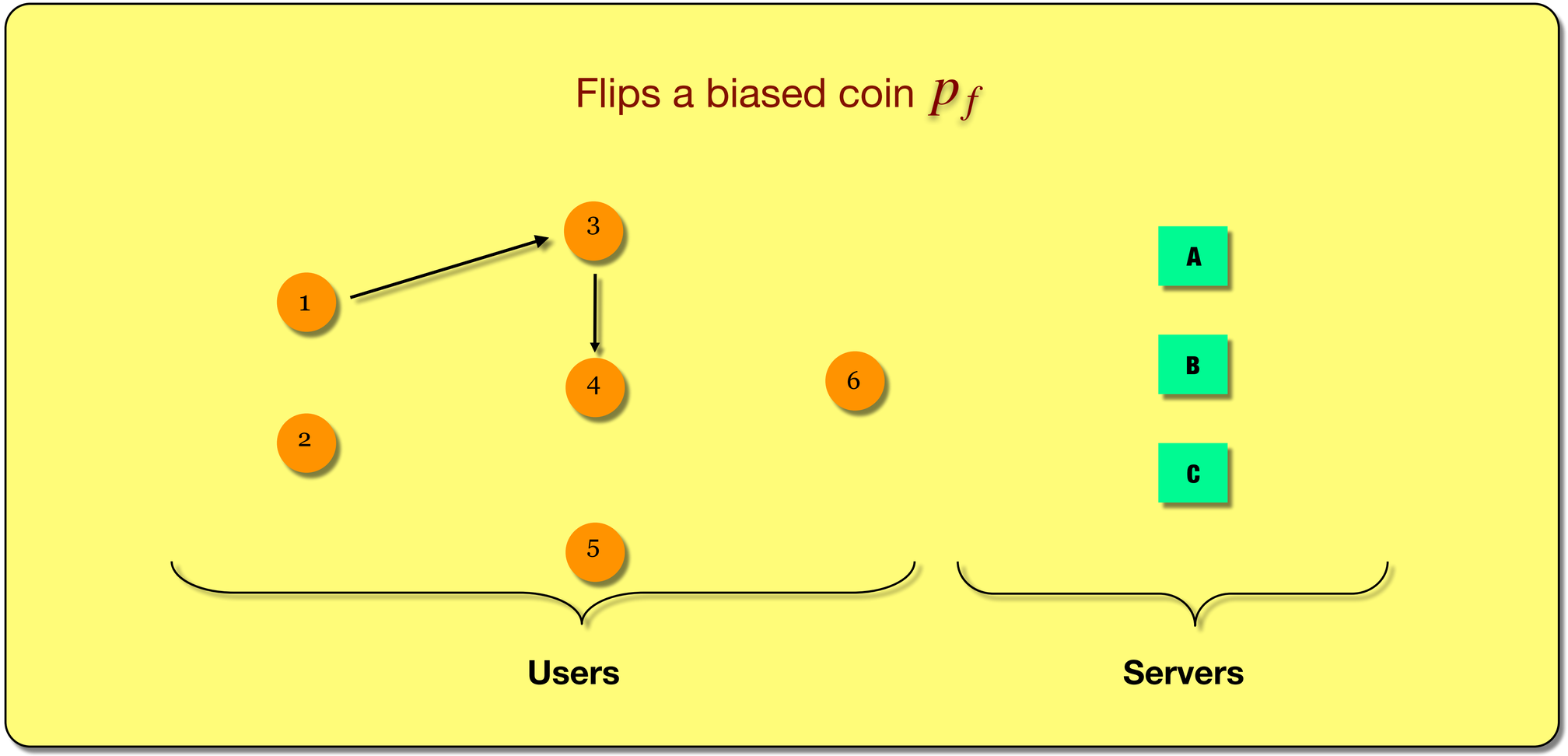
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

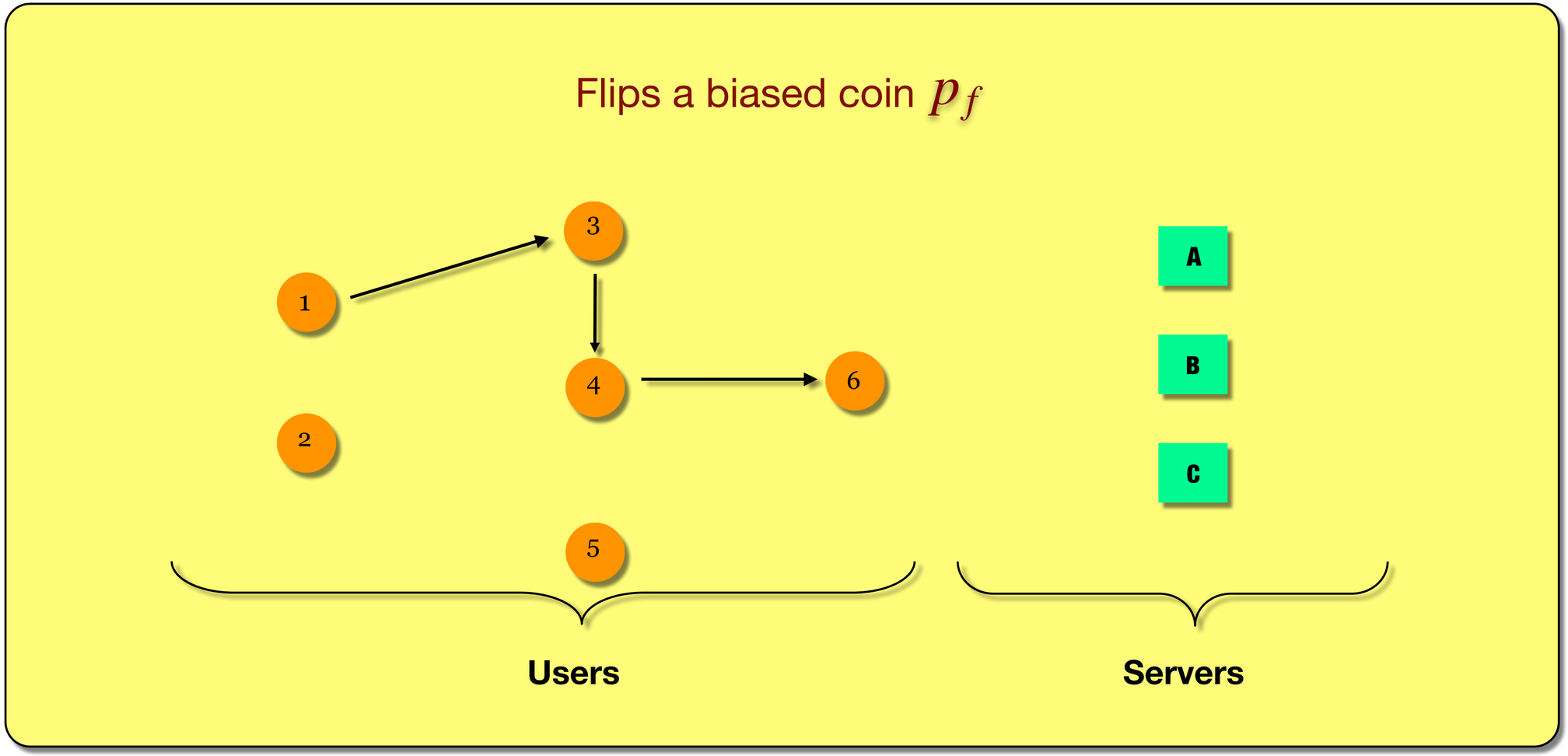
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

[reiter & rubin]

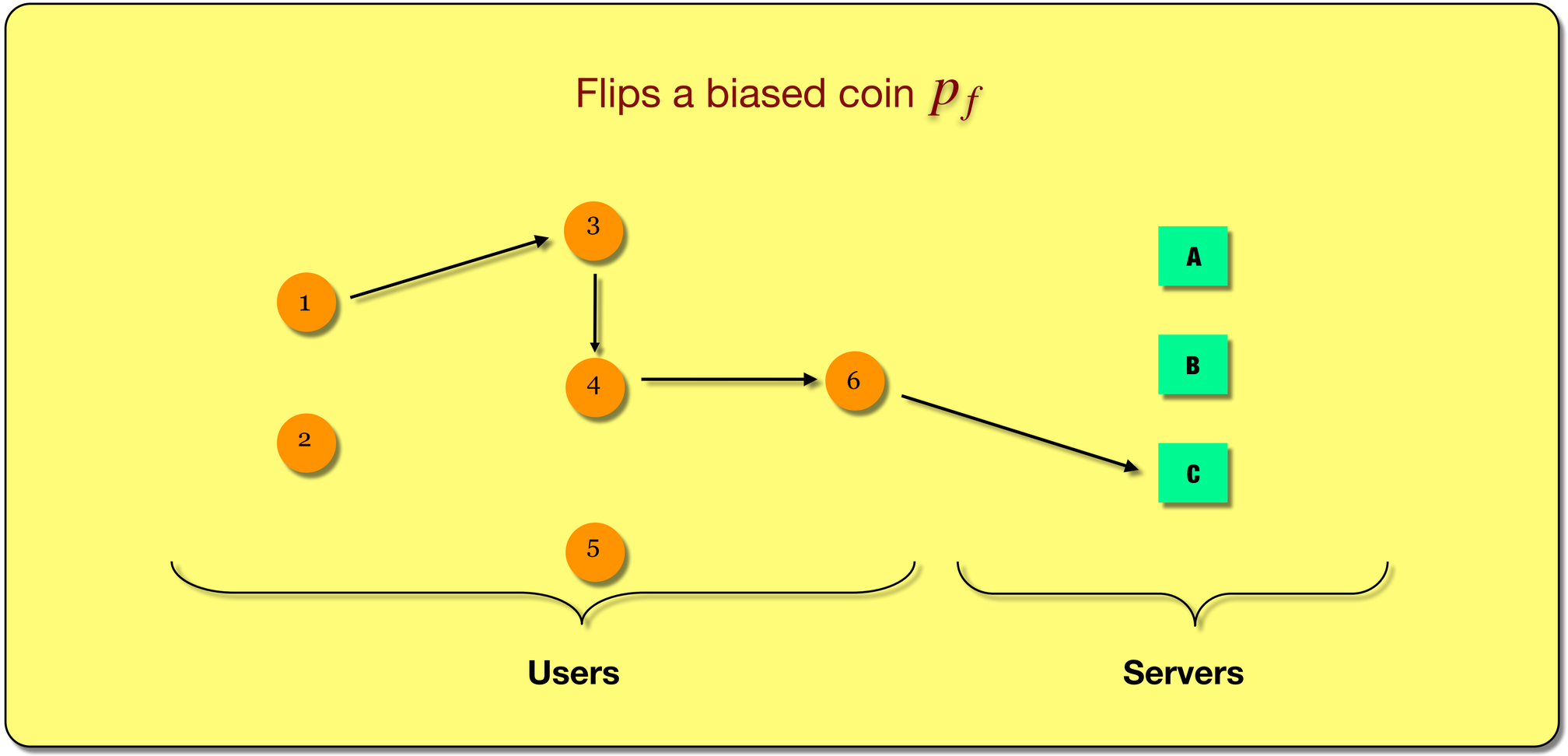


crowds: the protocol

UNIVERSITY OF Southampton

Allows internet users to perform anonymous web transactions

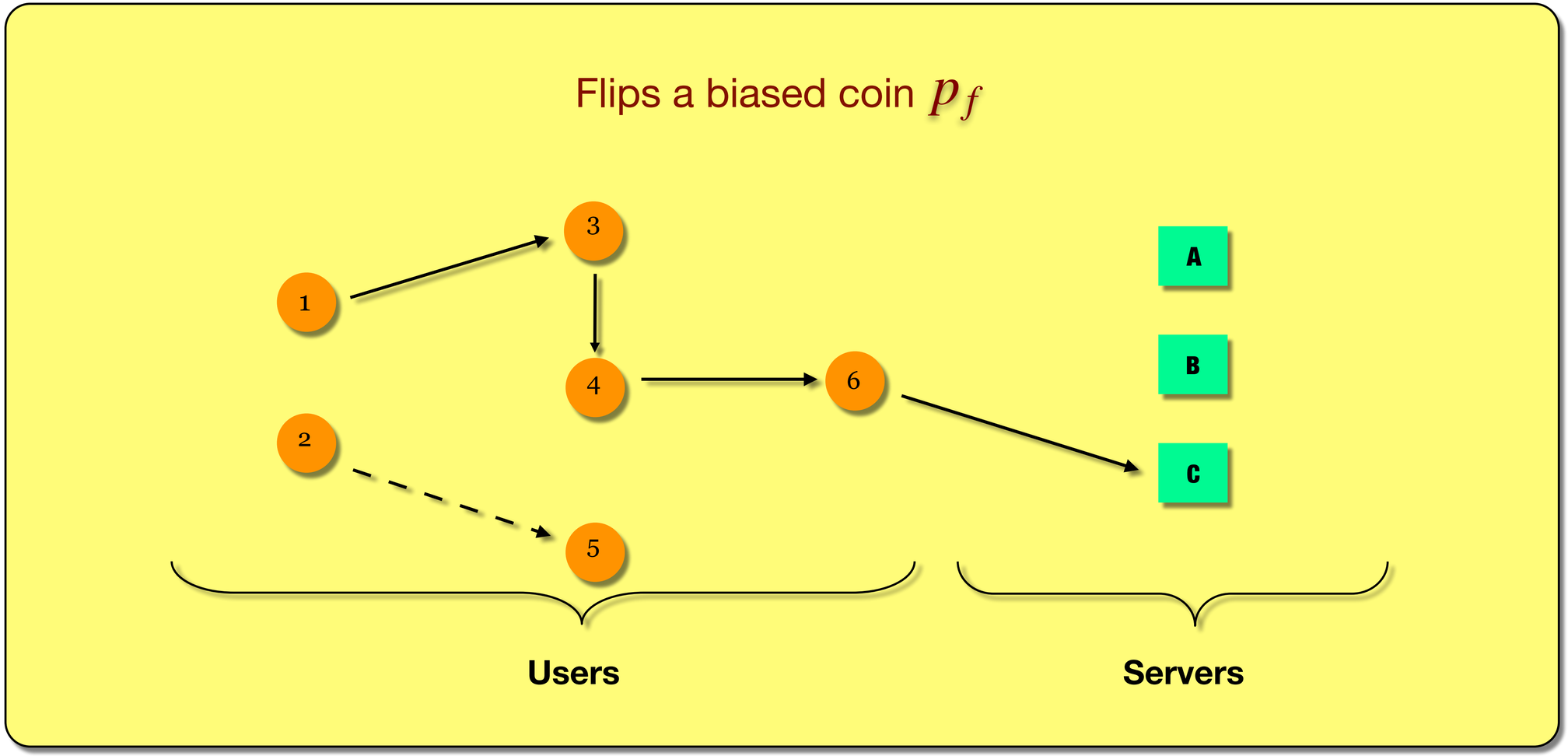
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

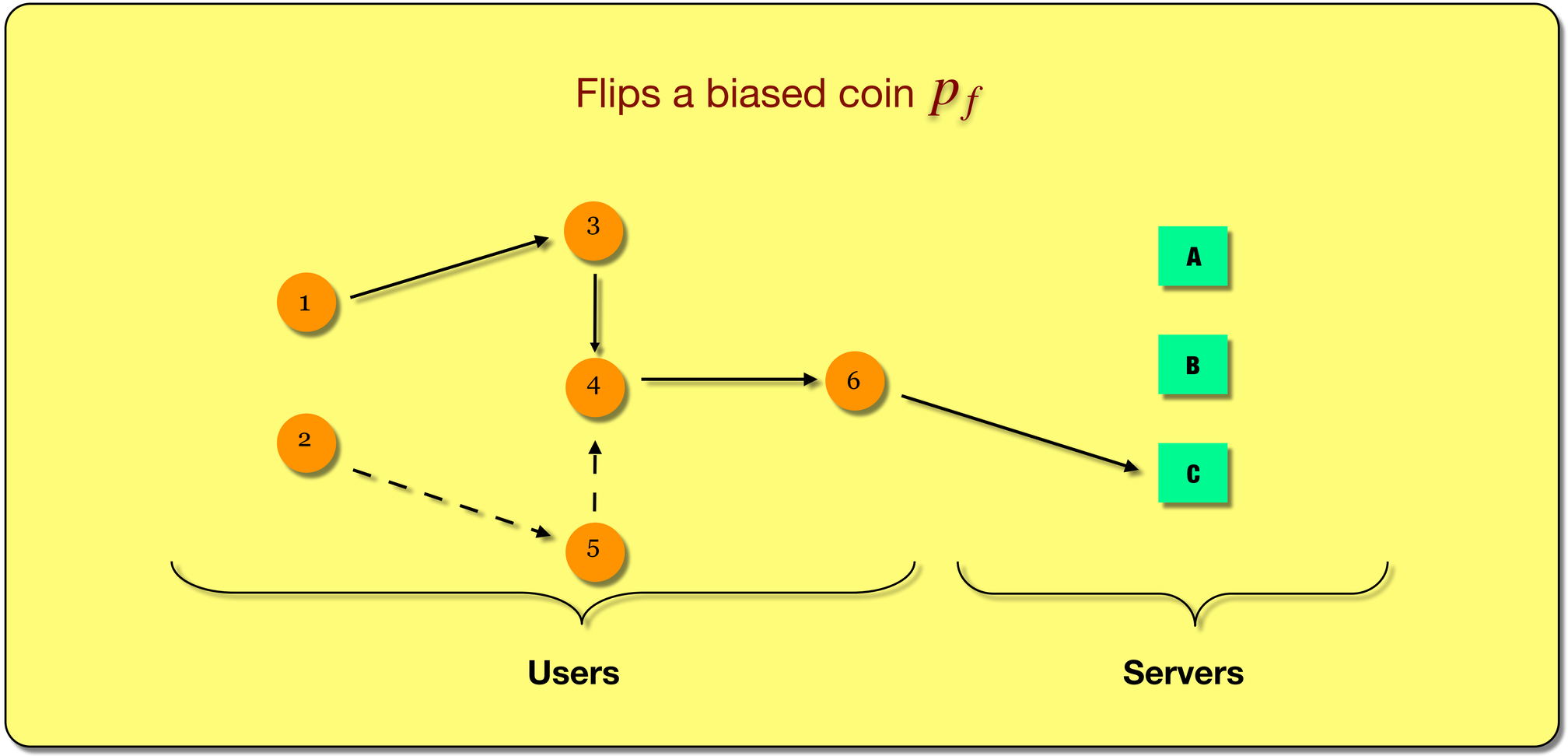
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

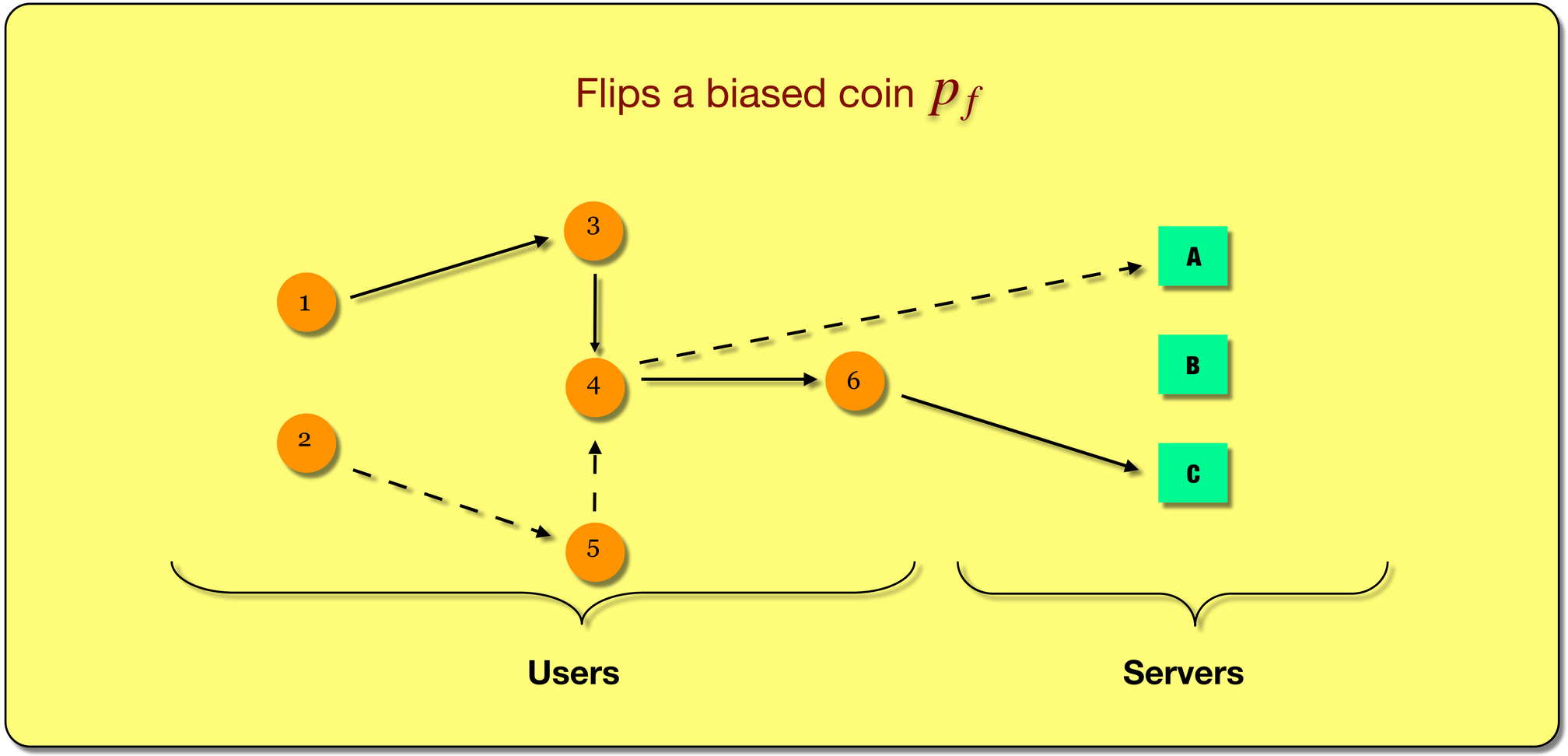
[reiter & rubin]



crowds: the protocol

Allows internet users to perform anonymous web transactions

[reiter & rubin]

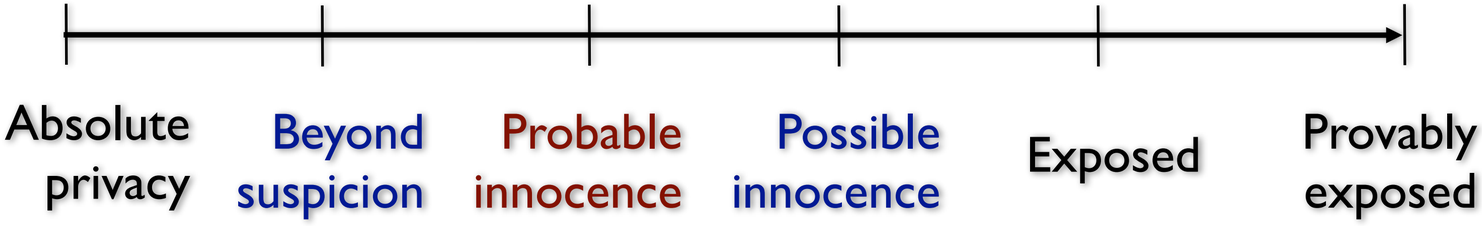


crowds: the protocol (algorithmically)

```
function aSend(M,D)
  begin
    j:= Random_Pick({1,...,n})
    {Relay(M,D)}_{κ_j} ⟶ j
  end
```

```
function Relay(M,D)
  begin
    if(Flip_biased_coin(p_f))
      M ⟶ D
    else
      j:= Random_Pick({1,...,n})
      {Relay(M,D)}_{κ_j} ⟶ j
    endif
  end
```

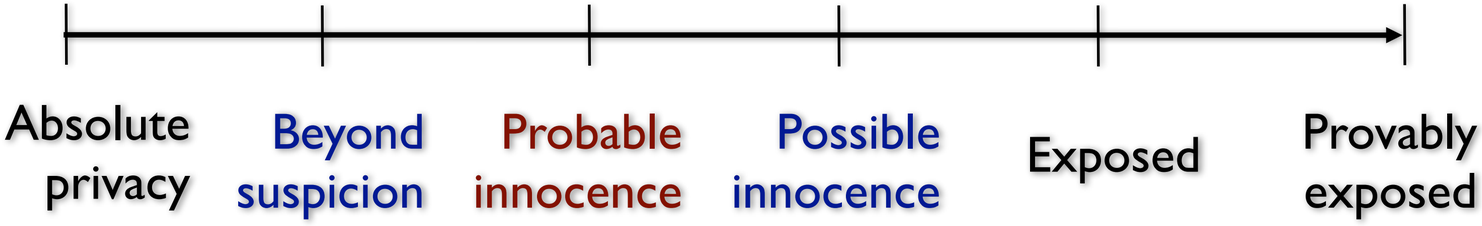
probable innocence: informal definition



"A sender is <u>probably innocent</u> if, from the attacker's point of view, the sender appears no more likely to be the originator than to not be the originator"

[*reiter & rubin*]

probable innocence: informal definition

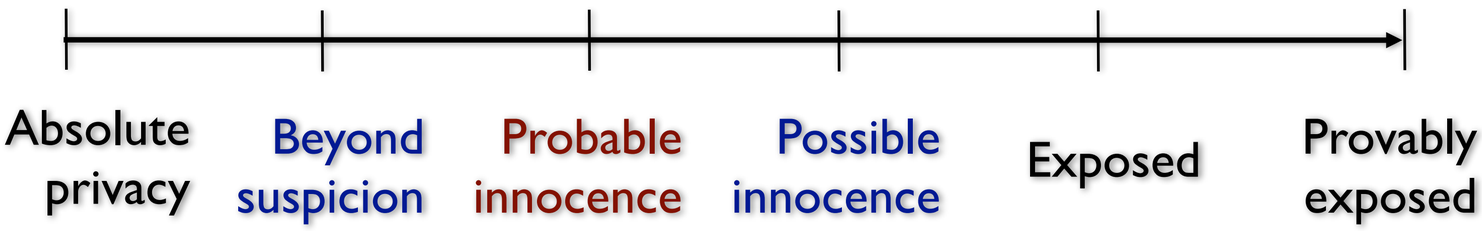


"A sender is probably innocent if, from the
e sender appears
ginator than to not
ator"

"A sender is possibly innocent if, from the attacker's point of view, is innocent with a non-null probability"

[reiter & rubin]

probable innocence: informal definition



"A sender is beyond suspicion if, from the attacker's point of view, is not more likely than anybody else to be the originator"

"A sender is probab

"A sender is possibly innocen
from the attacker's point of vie
innocent with a non-null probability"

[reiter & rubin]

crowds' anonymity guarantees

- Clearly Crowds cannot guarantee strong anonymity
- However, under simple conditions on the ratio of attackers and p_f, does provide useful anonymity against local observers
- from server's viewpoint: beyond suspicion
  - the first step, picks a user with probability $1/n$, therefore when the server receives a message it could come equally from any of the participants
- from the peers' viewpoint: probable innocence
  - the a forwarder has obvious good reasons to think that the preceding member in the path is more likely than anybody else to be the initiator; yet if there are enough honest users, the forwarder will know it is still more likely not to be

crowds' anonymity guarantees

- Clearly Crowds cannot guarantee strong anonymity

- However, under simple conditions on the ratio of attackers and p_f, does provide useful anonymity against local observers

Key:

there must sufficient user cooperation in the network

extreme case: all users selfish ➡ no anonymity at all

- from serve
  - the firs ver receives a messa

- from the peers' viewpoint: probable innocence
  - the a forwarder has obvious good reasons to think that the preceding member in the path is more likely than anybody else to be the initiator; yet if there are enough honest users, the forwarder will know it is still more likely not to be

Part II

Theoretical Analysis

probable innocence: formal definition

Members: n members participating in the protocol

n-c honest members

c corrupt members (collaborating attackers)

Anonymous events: a random variable A distributed over $\{ a_1, a_2, \ldots, a_{(n-c)} \}$, where a_i indicates that the honest user i is the initiator of the message.

Observable events: a random variable O distributed over $\{ o_1, o_2, \ldots, o_{(n-c)} \}$, where o_i indicates that user i is honest and forwards the message to a corrupted user.

In this case we say that user i is detected.

probable innocence: competing definitions

Definition [Reiter and Rubin, 98]: a protocol satisfies probable innocence if

$$\forall i.\ p(\, o_i \mid a_i \,) \leq 1/2$$

Definition [Halpern and O'Neill, 05]:

$$\forall i.\ p(\, a_i \mid o_i \,) \leq 1/2$$

probable innocence: competing definitions

Definition [Reiter and Rubin, 98]: a protocol satisfies probable innocence if

$$\forall i.\ p(\, o_i \mid a_i \,) \leq 1/2$$

based on detection

Definition [Halpern and O'Neill, 05]:

based users' actions

$$\forall i.\ p(\, a_i \mid o_i \,) \leq 1/2$$

probable innocence: competing definitions

UNIVERSITY OF Southampton

Definition [Reiter and Rubin, 98]: a protocol satisfies probable innocence if

$$\forall i.\ p(\, o_i \mid a_i \,) \leq 1/2$$

reported when guilty

based on detection

Definition [Halpern and O'Neill, 05]:

based users' actions

guilty when reported

$$\forall i.\ p(\, a_i \mid o_i \,) \leq 1/2$$

probable innocence: competing definitions

The symmetry of Crowds when picking relays makes the two definitions equivalent:

Proposition: if the a priori distribution is uniform then

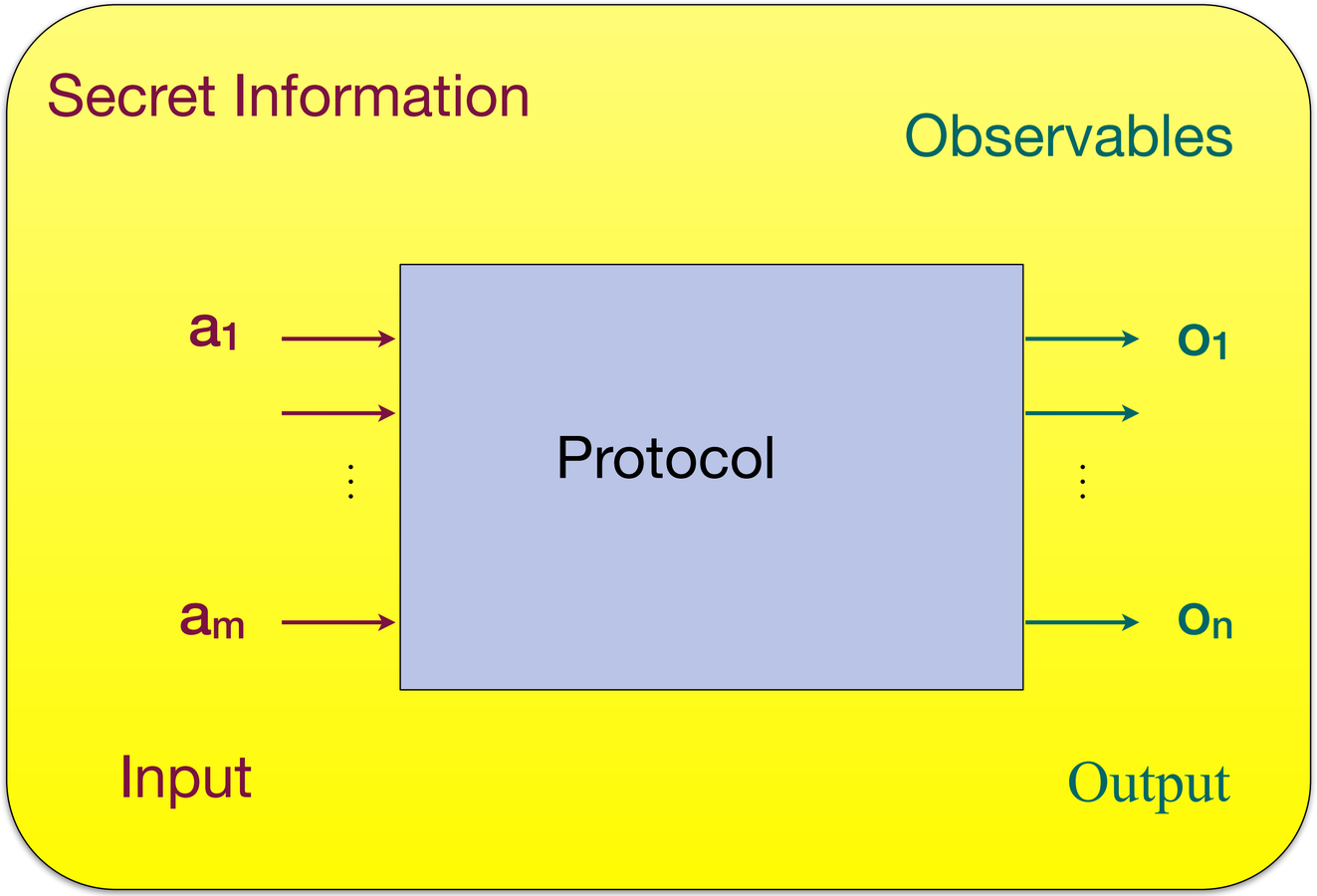
$$\forall i.\ p(\, a_i \,|\, o_i \,) = \ p(\, o_i \,|\, a_i \,)$$

In fact, $p(\, a_i \,) = p(\, o_j \,) = (n - c)^{-1}$ for all i, j

This however cease to be the case when the protocol is modified.

quantitative measures

Current consensus is both RR's and HON's metrics to be rejected for quantitative ones, similarly to Information Flow analysis



[*palamidessi et al*]

Information-theoretic measures

The entropy $H(A)$ measures the uncertainty about the anonymous events:

$$H(A) = -\sum_{a \in \mathcal{A}} p(a) \log p(a)$$

It is the attacker's initial uncertainty.

[*diaz et al*]

The conditional entropy $H(A \mid O)$ measures the uncertainty about A after making observation O

$$H(A \mid O) = -\sum_{o \in \mathcal{O}} p(o) \sum_{a \in \mathcal{A}} p(a \mid o) \log p(a \mid o)$$

It is the information leaked by the observation.

The mutual information $I(A\,;\,O) = H(A) - H(A \mid O)$ measures the attacker's remaining uncertainty after the observation.

[*malacaria et al*]

Information-theoretic measures

The entropy $H(A)$ measures the uncertainty about the anonymous events:

$$H(A) = -\sum_{a \in \mathcal{A}} p(a) \log p(a)$$

It is the attacker's initial uncertainty.

[diaz et al]

Intuitive slogan:

remaining uncertainty =

initial uncertainty - information leaked

The conditional entrop er making observation O

$$H(A \mid O) = -\sum_{o \in \mathcal{O}} p(o) \sum_{a \in \mathcal{A}} p(a \mid o) \log p(a \mid o)$$

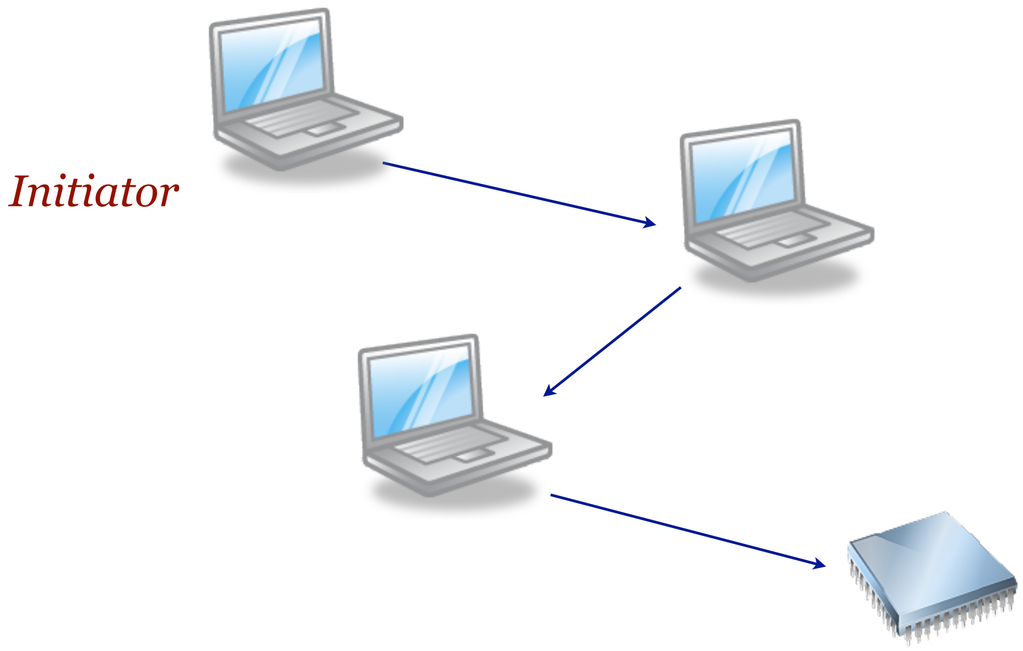
It is the information leaked by the observation.

The mutual information $I(A\,;\,O) = H(A) - H(A \mid O)$ measures the attacker's remaining uncertainty after the observation.

[malacaria et al]

the role of cooperation

During path creation...

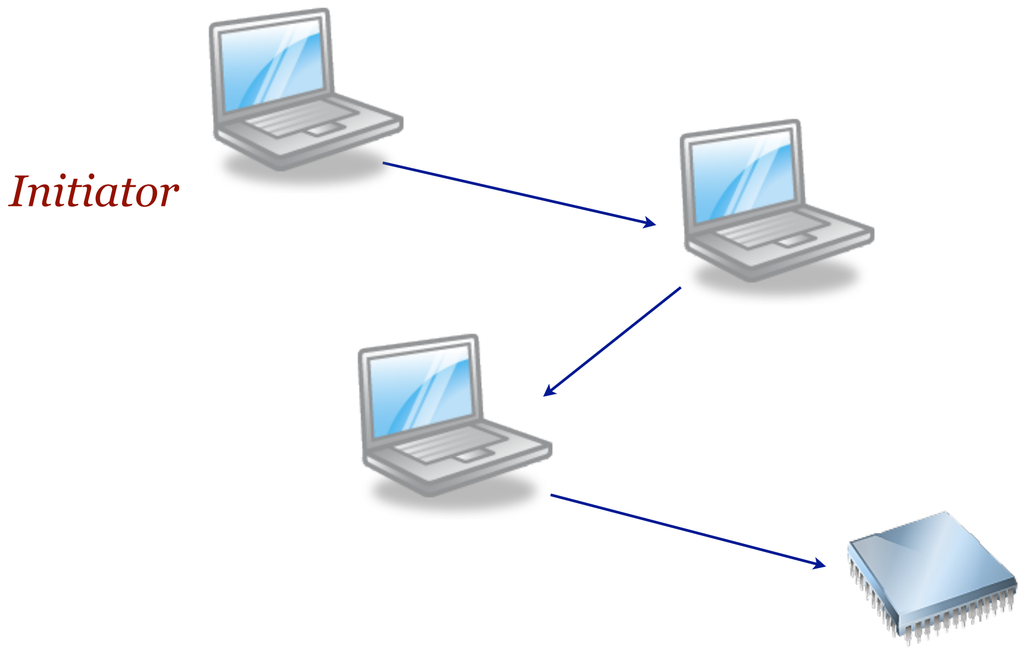


…fewer cooperative users will degrade anonymity...

the role of cooperation

UNIVERSITY OF Southampton

During path creation...



$$p(o_i \mid a_i) \;=\; 1 - \frac{n - c - 1}{n} p_f$$

o_i: the event that i is detected
a_i: the event that i is initiator
n: no of cooperative honest users
c: no of malicious users
p_f: forwarding probability

if n decreases, then $p(o_i \mid a_i)$ increases

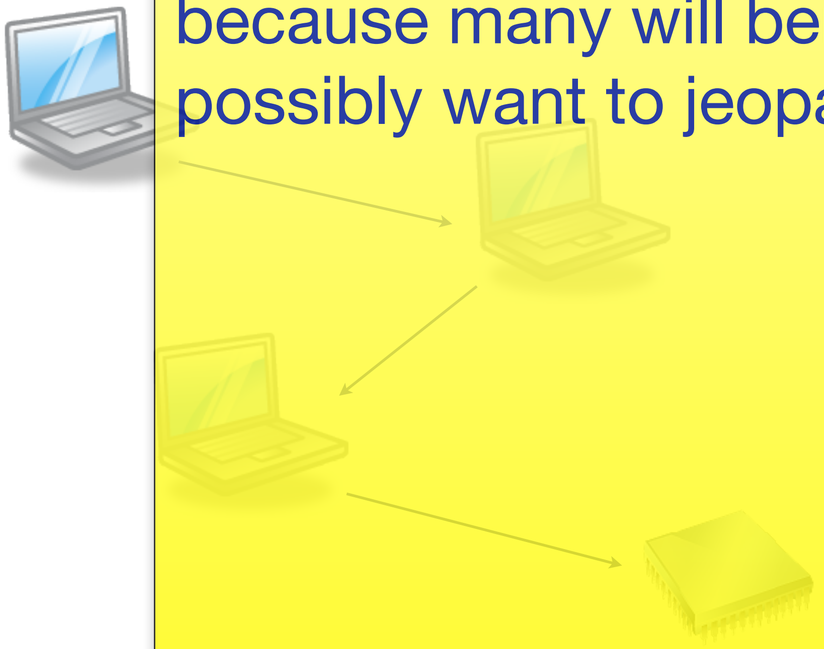
…fewer cooperative users will degrade anonymity...

the role of cooperation

UNIVERSITY OF Southampton

During path creation...

Sounds like this should work fine because many will benefit. Who may possibly want to jeopardise it ?



$$p(o_i \mid a_i) = 1 - \frac{n - c - 1}{n} p_f$$

o_i: the event that i is detected
a_i: the event that i is initiator
n: no of cooperative honest users
c: no of malicious users
p_f: forwarding probability

if n decreases, then $p(o_i \mid a_i)$ increases

...fewer cooperative users will degrade anonymity...

the role of cooperation

UNIVERSITY OF Southampton

During path creation...

Initiator

$$p(o_i \mid a_i) = 1 - \frac{n - c - 1}{n} p_f$$

o_i: the event that i is detected

a_i: the event that i is initiator

n: no of cooperative honest users

c: no of malicious users

p_f: forwarding probability

if n decreases, then $p(o_i \mid a_i)$ increases

Sounds like this should work fine because many will benefit. Who may possibly want to jeopardise it ?

Well, cooperation has costs, e.g. bandwidth. Users may be in the network only to serve themselves. Or might be trying to avoid consequences

...fewer cooperative users will degrade anonymity...

the role of cooperation

UNIVERSITY OF Southampton

During path creation...

Initiator

$$p(o_i \mid a_i) = 1 - \frac{n - c - 1}{n} p_f$$

o_i: the event that i is detected

a_i: the event that i is initiator

n: no of cooperative honest users

c: no of malicious users

p_f: forwarding probability

if n decreases, then $p(o_i \mid a_i)$ increases

Sounds like this should work fine because many will benefit. Who may possibly want to jeopardise it ?

Well, cooperation has costs, e.g. bandwidth. Users may be in the network only to serve themselves. Or might be trying to avoid consequences

How will this affect the network ??

...fewer cooperative users will degrade anonymity...

incentives to cooperation

UNIVERSITY OF Southampton

Social Mechanisms:

Build a community of volunteer forwarders: **Tor**

Reputation-based Mechanisms:

users with higher reputation will get higher-quality service (e.g., Gold-star mechanism, Crowds extended)

Payment-based Mechanisms:

charge for anonymity services and/or reward cooperative user behaviours (e.g., Freedom system)

gold-star mechanism

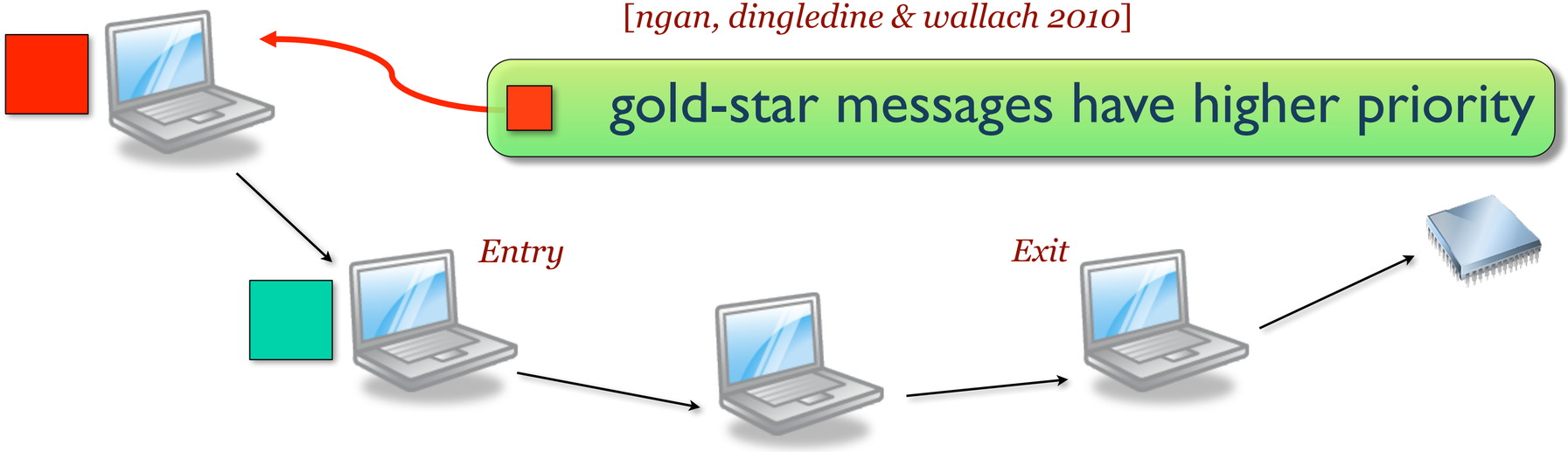
[ngan, dingledine & wallach 2010]

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star mechanism

[ngan, dingledine & wallach 2010]



gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star mechanism

[ngan, dingledine & wallach 2010]

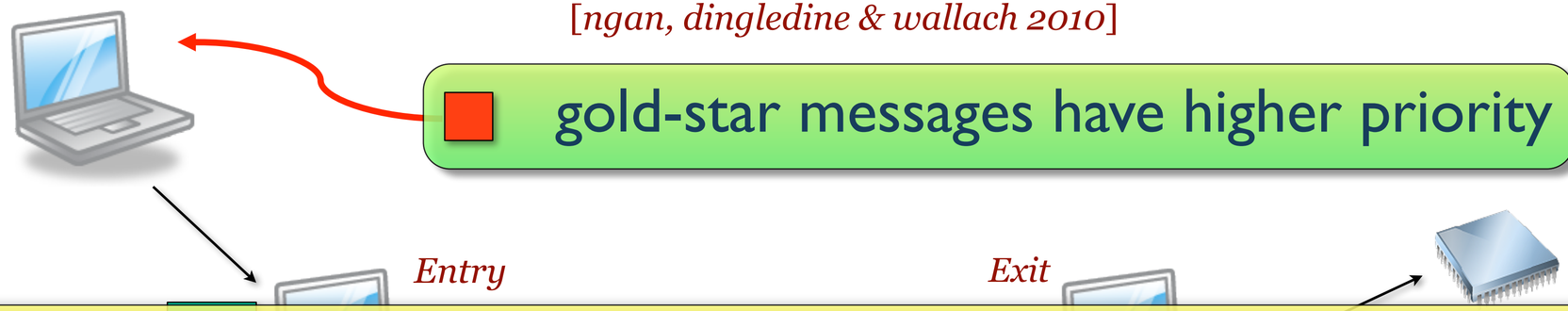
gold-star messages have higher priority

Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star messages have higher priority



Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

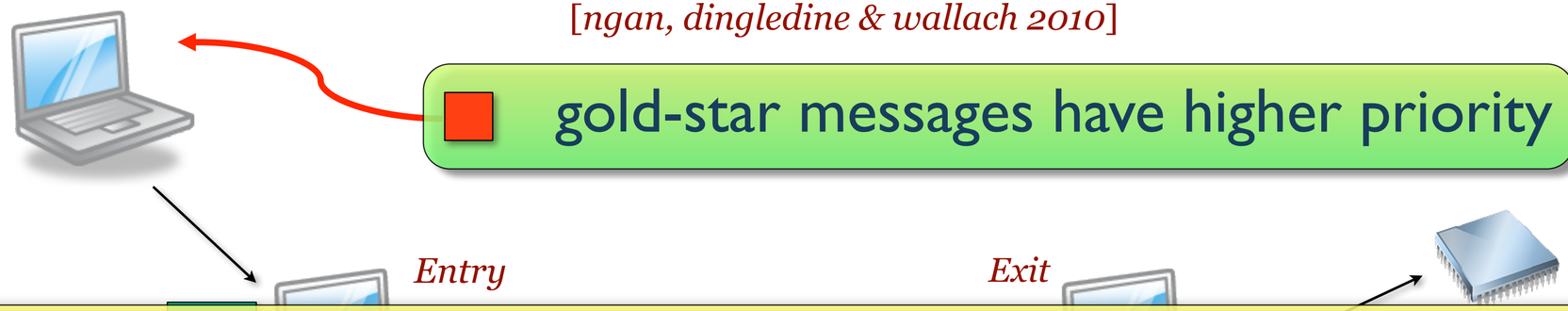
Cooperative:

bandwidth

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star messages have higher priority



Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

Cooperative:

Selfish:

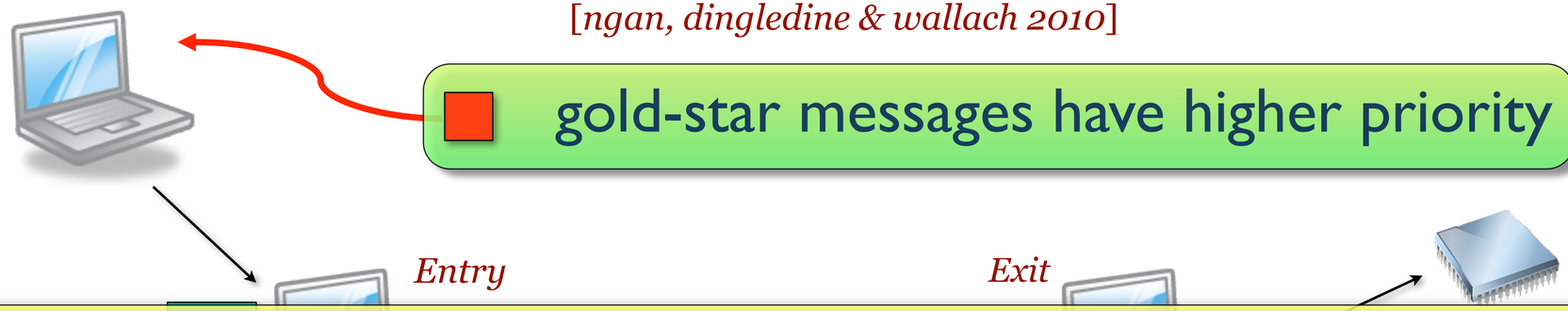
bandwidth

only routes own messages

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star messages have higher priority



Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

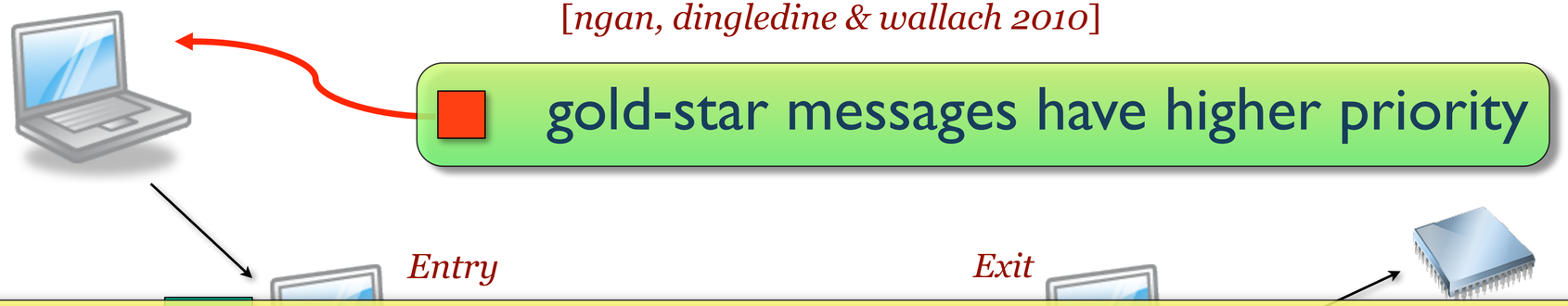
Cooperative: bandwidth

Selfish: only routes own messages

Cooperative Slow:

gold-star mechanism

[ngan, dingledine & wallach 2010]



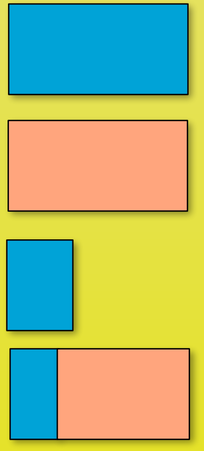
Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

Cooperative:

Selfish:

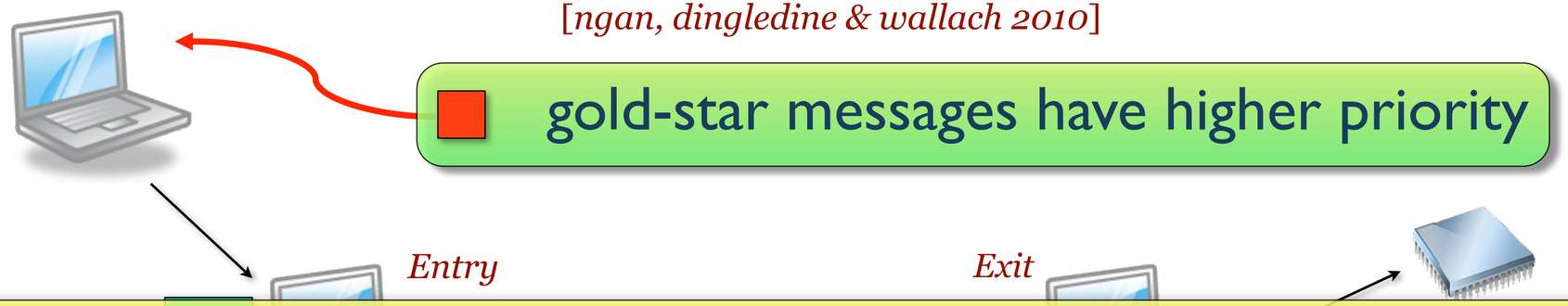
Cooperative Slow:

Cooperative Reserve:



gold-star mechanism

[ngan, dingledine & wallach 2010]



gold-star messages have higher priority

Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

Cooperative: bandwidth

Selfish: only routes own messages

Cooperative Slow:

Cooperative Reserve:

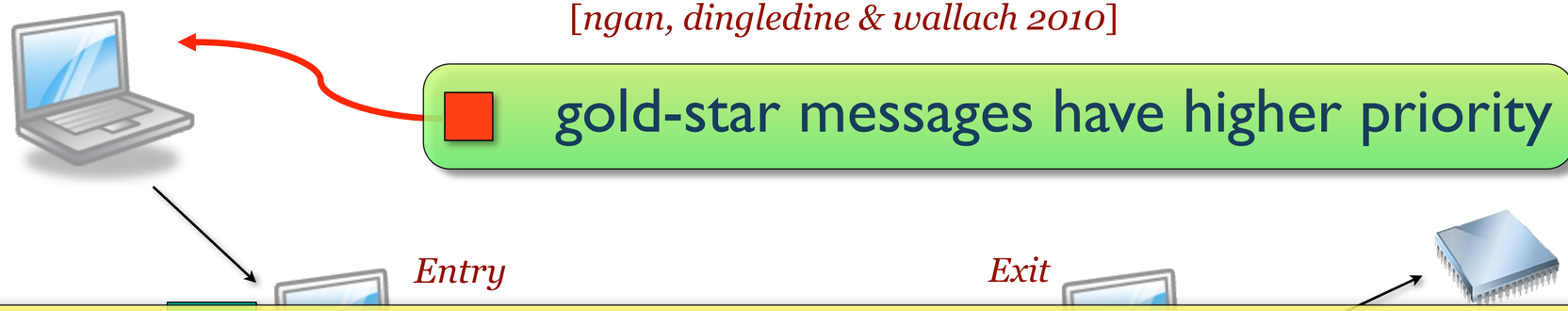
Adaptive: behave like until you get gold-star

behave like afterwards

gold-star mechanism

[ngan, dingledine & wallach 2010]

gold-star messages have higher priority



Roles: $S_i = \{\mathcal{C}, \mathcal{L}, \mathcal{R}, \mathcal{S}, \mathcal{A}\}$

gold-star if $\mathcal{C}$ k in each n+1 times

Cooperative: bandwidth

Selfish: only routes own messages

Cooperative Slow:

Cooperative Reserve:

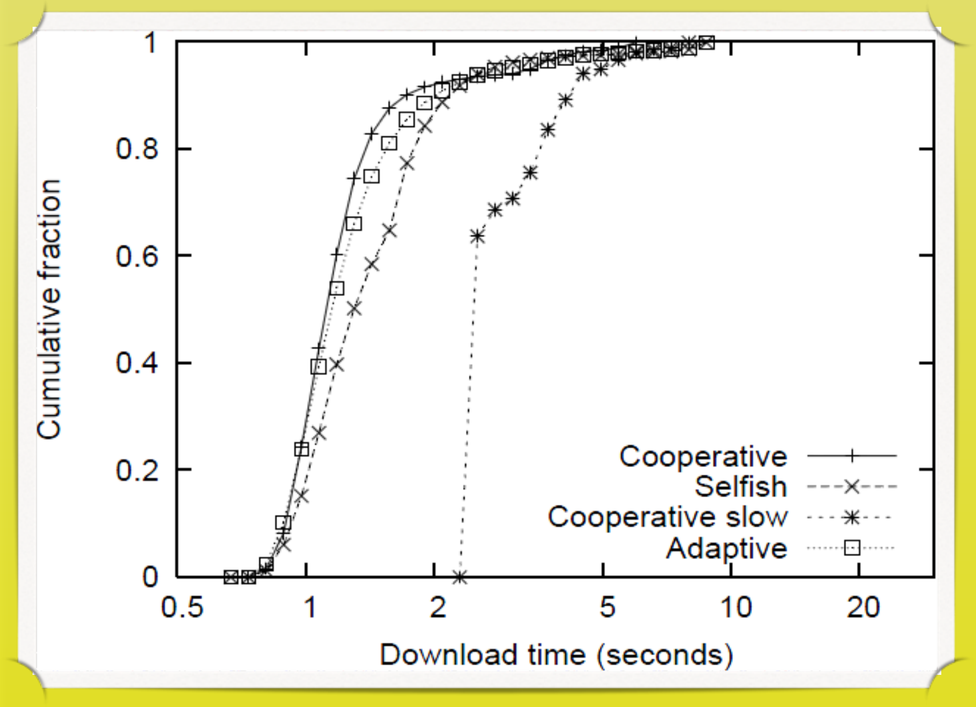
Adaptive: behave like until you get gold-star

behave like afterwards

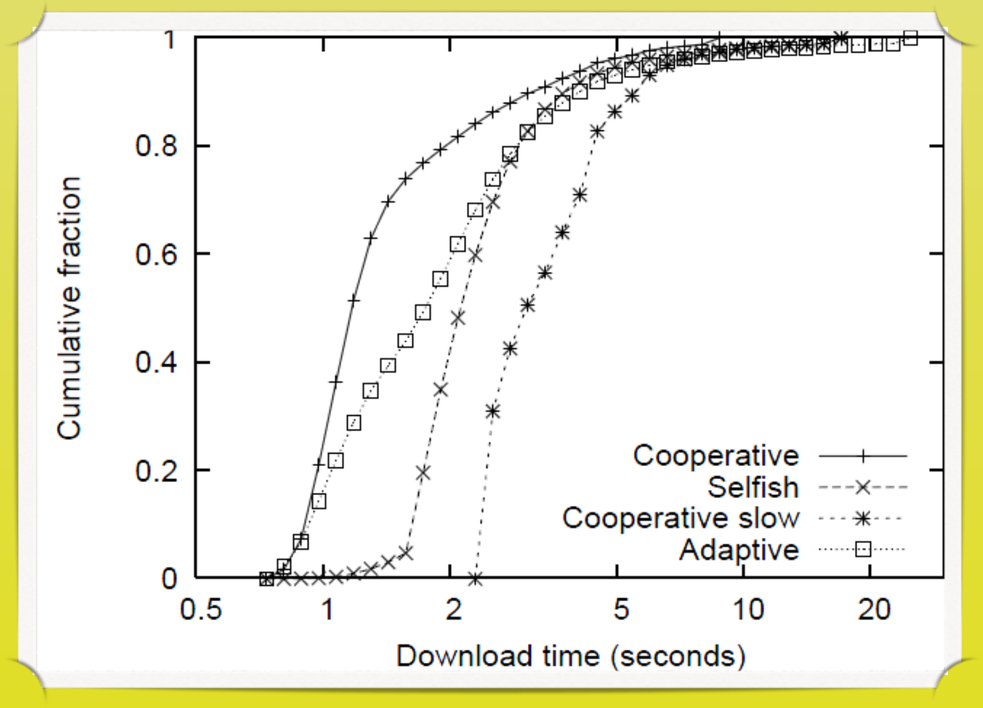
evaluation of the mechanism

[ngan, dingledine & wallach 2010]

Cumulative fraction: $p(t_i \leq t')$



Traffic is light



Traffic is heavy

Part III

Onion routing (aka Tor)

shortcomings of crowds

- message is not encrypted
  - could it be explicitly encrypted?
  - anyway, the destination is visible for the entire path
  - this creates potential for de-anonymisation

- sender has no control on routing parameters
  - pick forwarders they trust
  - pick length of path

- anyway, Crowds is only a theoretical model:
  - too many details are too loose for a real implementation, better start to seek for a better protocol altogether

shortcomings of crowds

- message is not encrypted
  - could it be explicitly encrypted? *hmm, not really*
  - anyway, the destination is visible for the entire path
  - this creates potential for de-anonymisation

- sender has no control on routing parameters
  - pick forwarders they trust
  - pick length of path

- anyway, Crowds is only a theoretical model:
  - too many details are too loose for a real implementation, better start to seek for a better protocol altogether

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[*syverson et al*]

$data \times B$

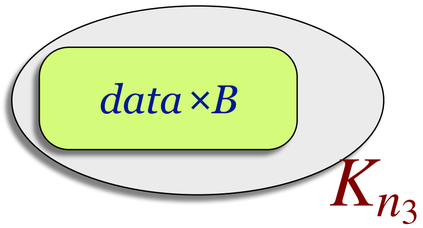
A wants to send data to B

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[*syverson et al*]



A wants to send data to B

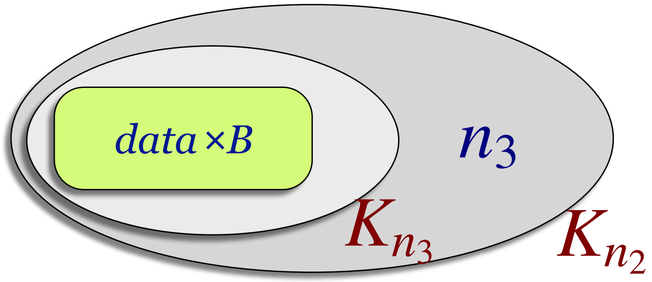
she wraps it with n_3's public key

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[*syverson et al*]



A wants to send data to B

she wraps it with n_3's public key

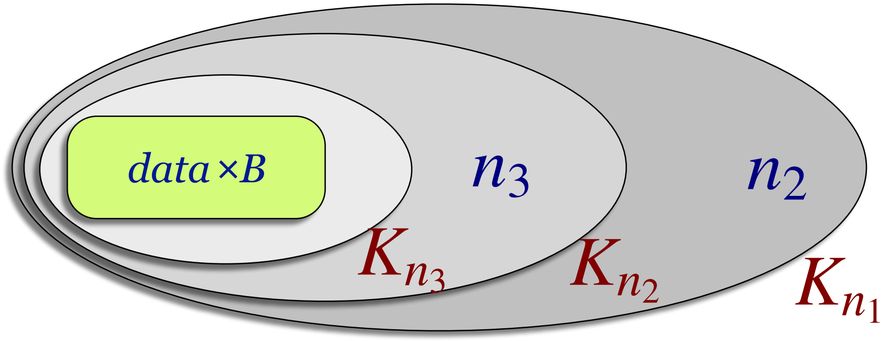
then adds n_3's id and wraps it with n_2's public key

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[syverson et al]



A wants to send data to B

she wraps it with n_3's public key

then adds n_3's id and wraps it with n_2's public key

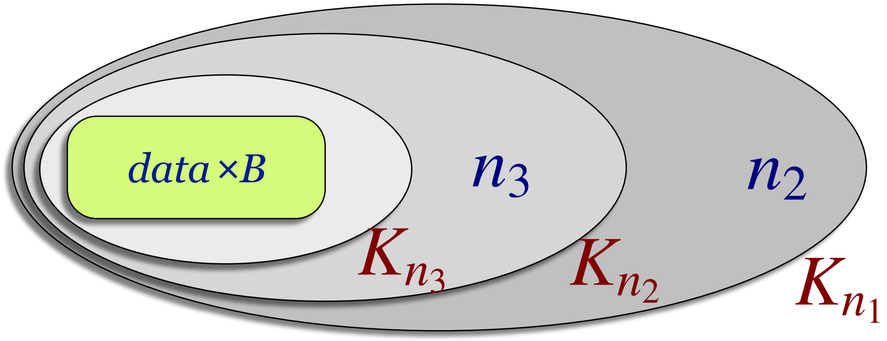
then adds n_2's id and wraps it with n_1's public key

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[syverson et al]



Observe that onions also contain session keys so that replies can be encrypted on the way back and PKe is only used on new paths

A wants to send data to B

she wraps it with n_3's public key

then adds n_3's id and wraps it with n_2's public key

then adds n_2's id and wraps it with n_1's public key

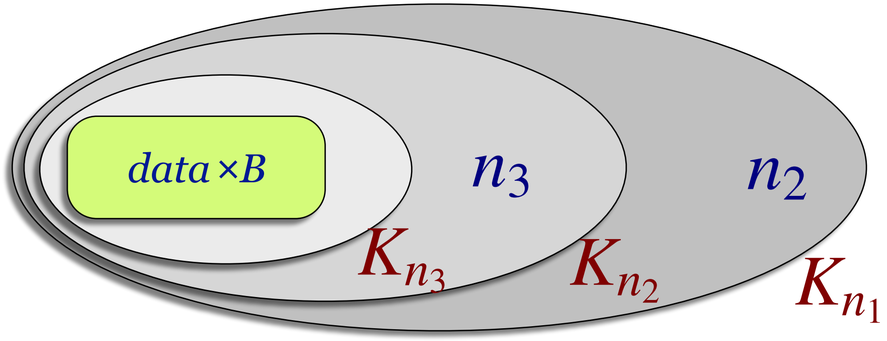
and sends the onion to n_1

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[syverson et al]



Whilst onions are built by A and peeled on their way to be, reply onions acquire a new layer at each station, until they and are peeled by A

upon reception:

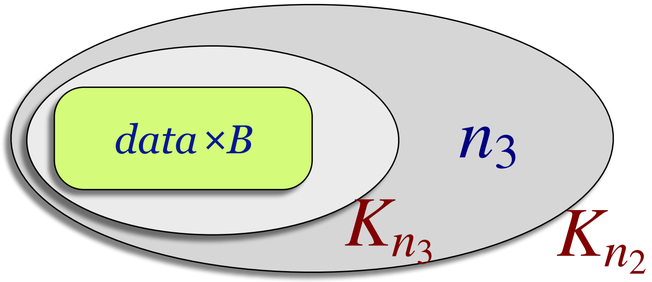
n_1 peels one onion's layer by decryption

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[*syverson et al*]



Whilst onions are built by A and peeled on their way to be, reply onions acquire a new layer at each station, until they and are peeled by A

upon reception:

n_1 peels one onion's layer by decryption

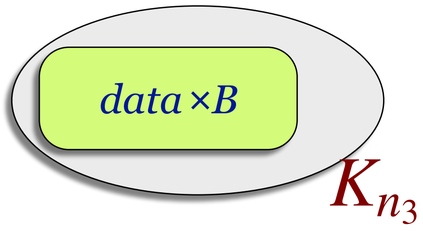
then discovers n_2's id, and forwards the remaining onion

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[syverson et al]



Whilst onions are built by A and peeled on their way to be, reply onions acquire a new layer at each station, until they and are peeled by A

upon reception:

n_1 peels one onion's layer by decryption

then discovers n_2's id, and forwards the remaining onion

similarly for n_2

Onion Routing

- Path established by initiator A, eg:

$$A \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow B$$

[syverson et al]

$data \times B$

Whilst onions are built by A and peeled on their way to be, reply onions acquire a new layer at each station, until they and are peeled by A

upon reception:

n_1 peels one onion's layer by decryption

then discovers n_2's id, and forwards the remaining onion

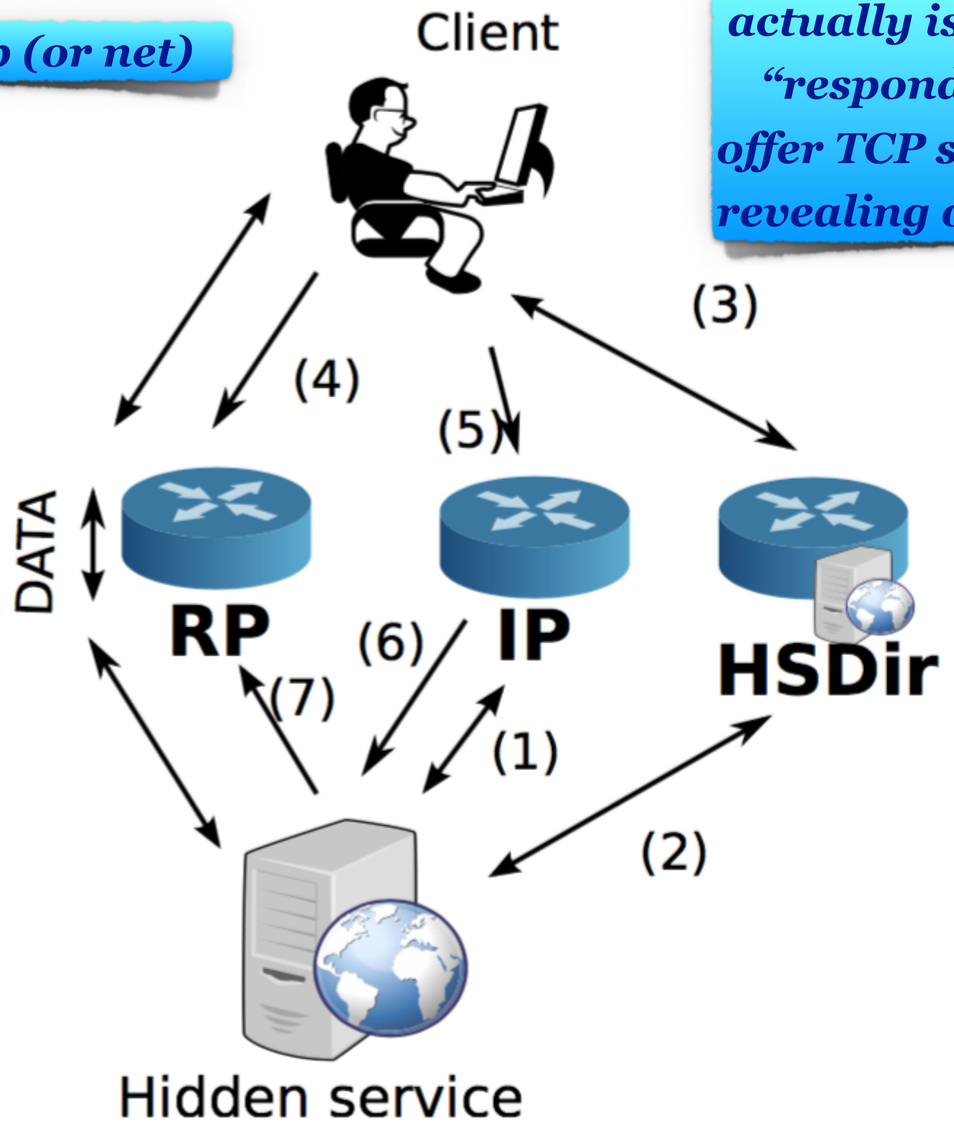
similarly for n_2

finally, n_3 decrypts and deliver cleartext data to B

OR technologies: hidden services

aka the dark web (or net)

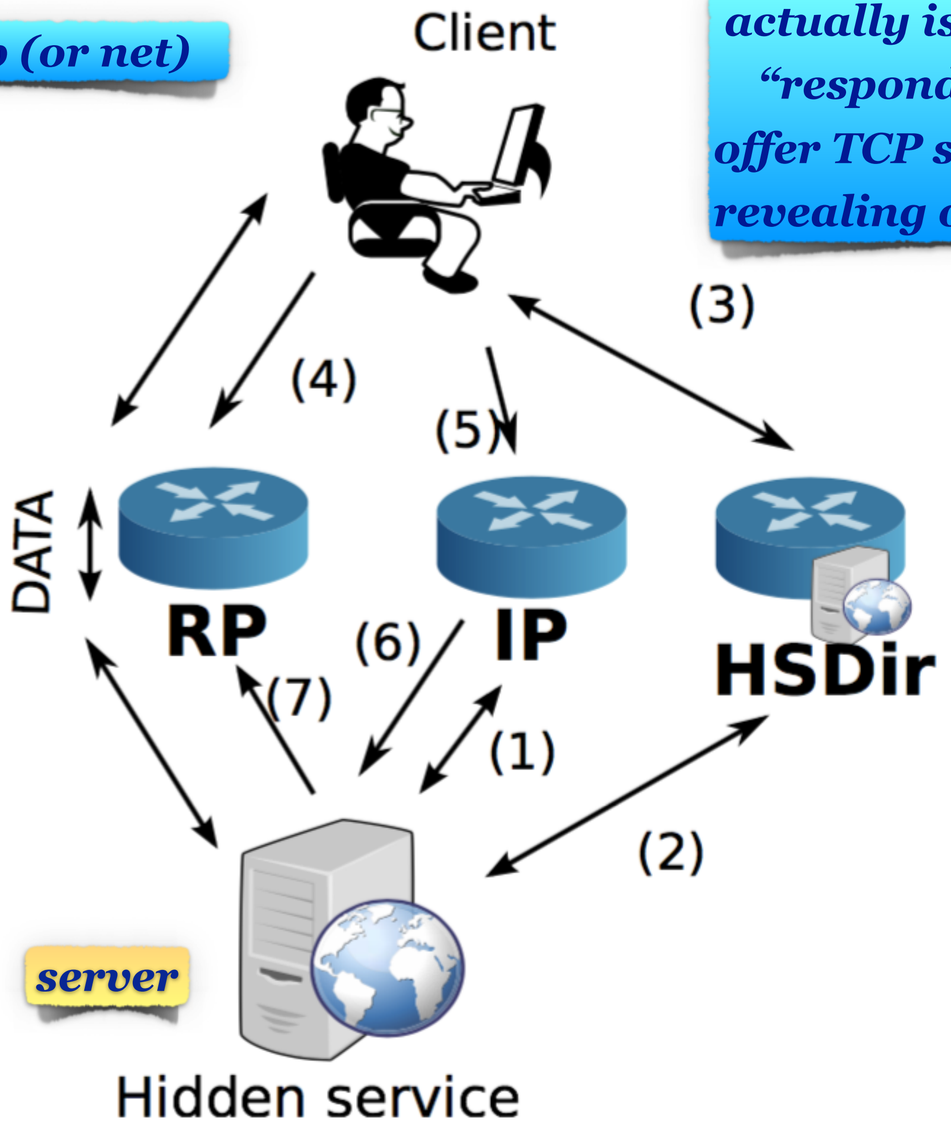
actually is
"responder anonymity"
offer TCP service without revealing own IP



OR technologies: hidden services

aka the dark web (or net)

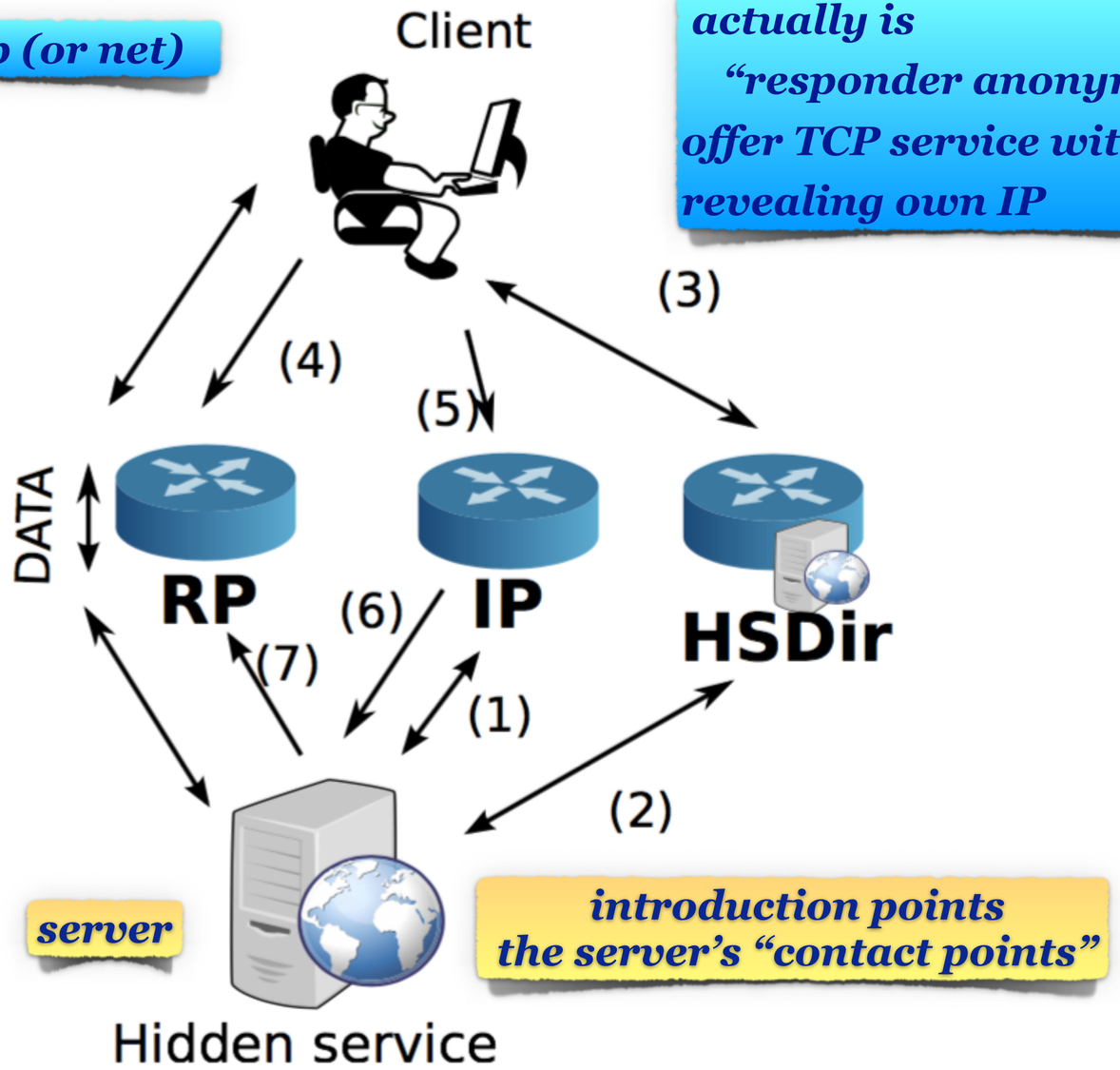
actually is
"responder anonymity"
offer TCP service without revealing own IP



OR technologies: hidden services

aka the dark web (or net)

actually is
"responder anonymity"
offer TCP service without revealing own IP



server

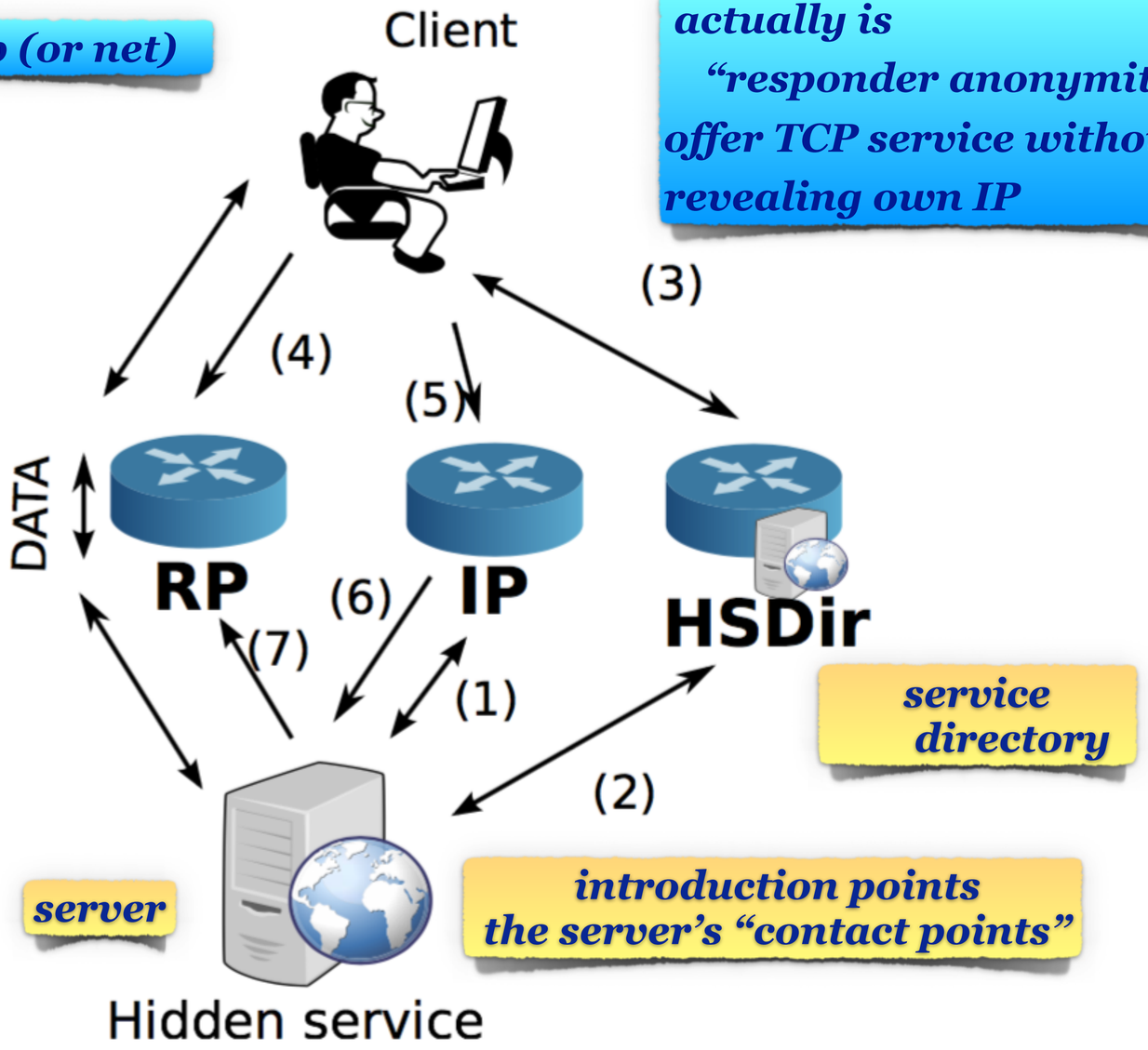
introduction points
the server's "contact points"

OR technologies: hidden services

UNIVERSITY OF Southampton

aka the dark web (or net)

actually is
"responder anonymity"
offer TCP service without revealing own IP



service directory

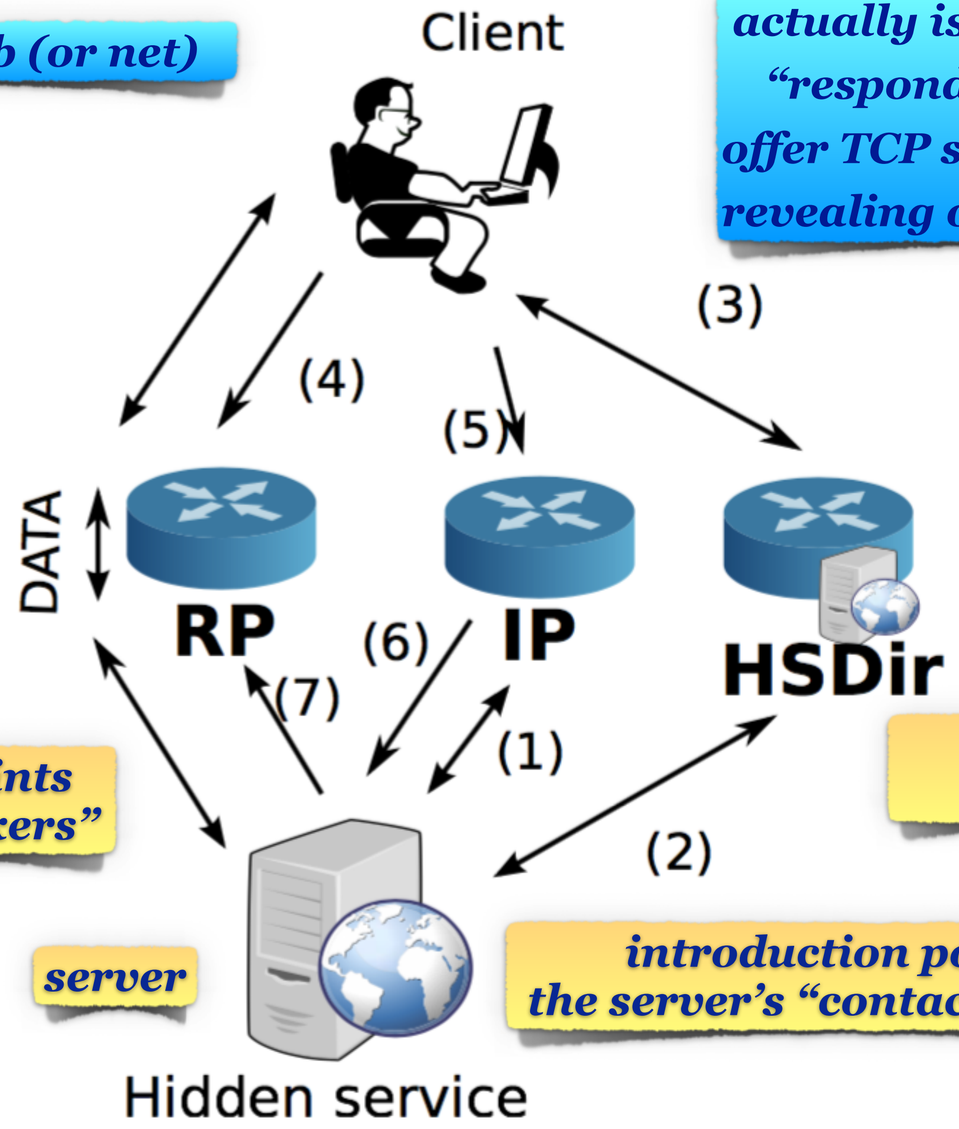
server

introduction points
the server's "contact points"

OR technologies: hidden services

aka the dark web (or net)

actually is
"responder anonymity"
offer TCP service without revealing own IP



rendezvous points
the client's "brokers"

service directory

server

introduction points
the server's "contact points"

Tor: The Second-Generation Onion Router

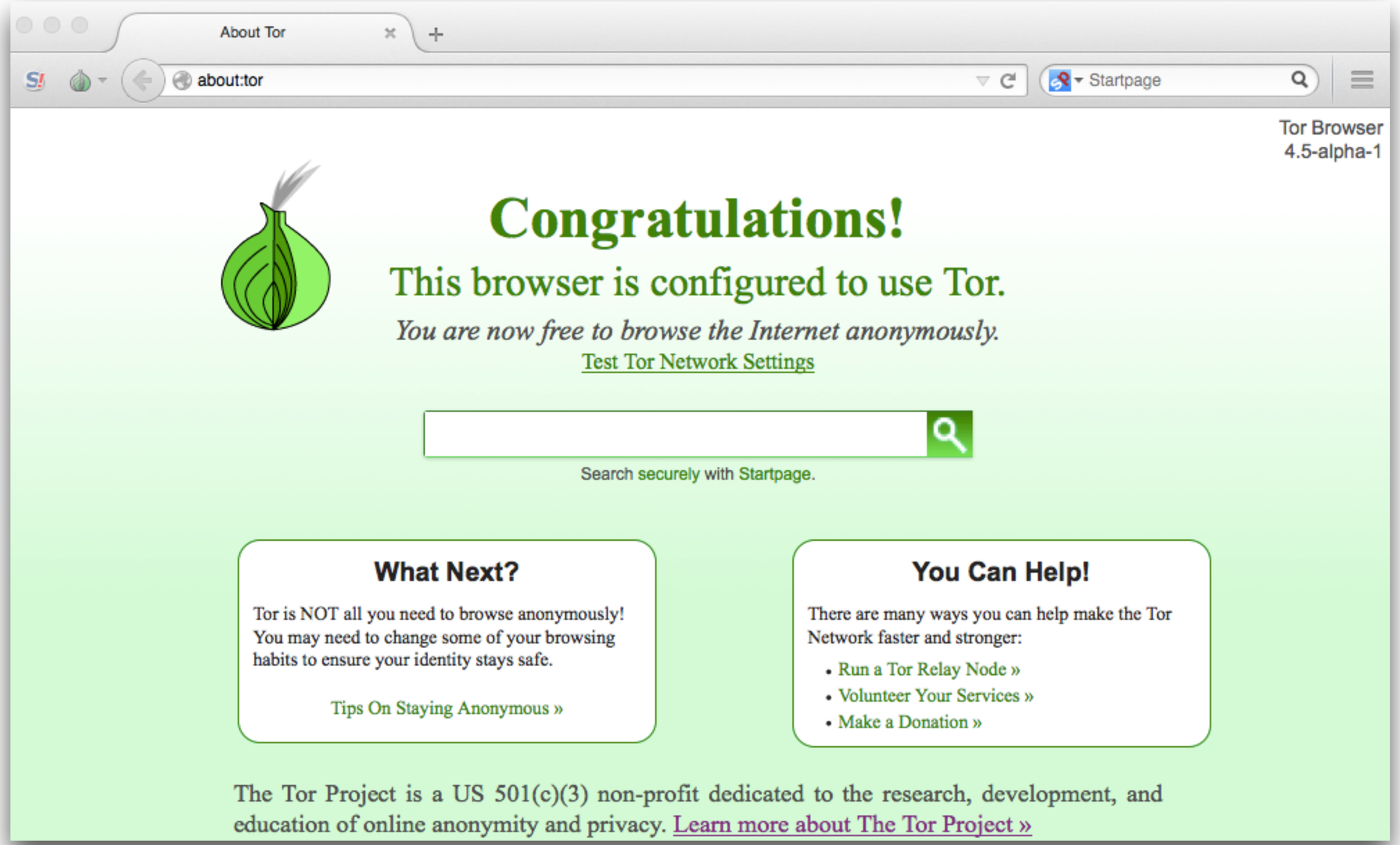
Roger Dingledine
The Free Haven Project
arma@freehaven.net

Nick Mathewson
The Free Haven Project
nickm@freehaven.net

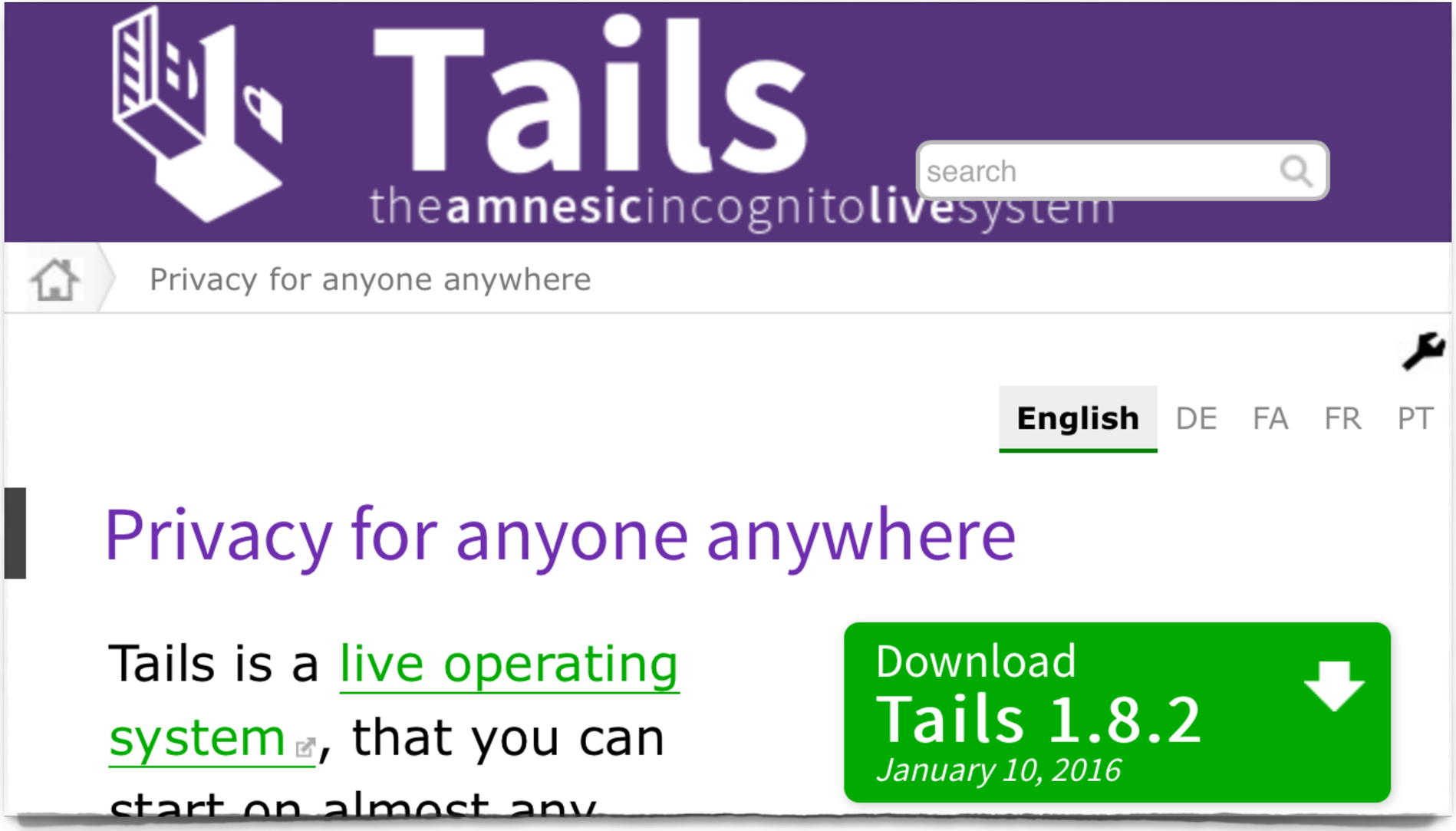
Paul Syverson
Naval Research Lab
syverson@itd.nrl.navy.mil

homework: *read and discuss this paper*

OR technologies: browser



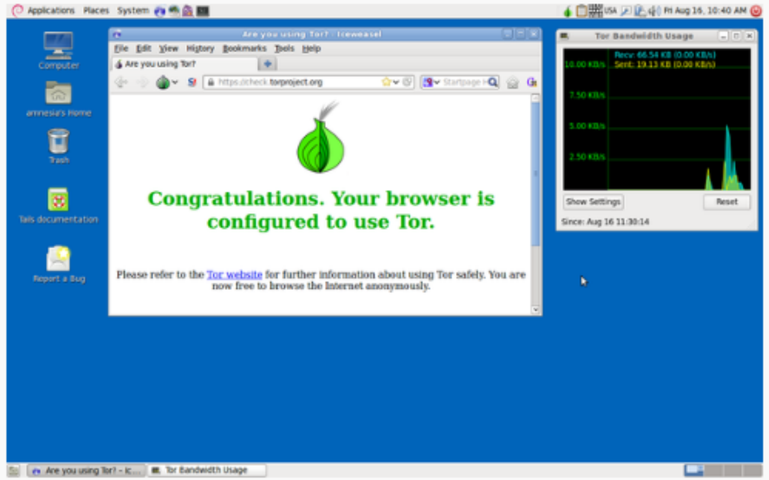
OR technologies: operating system



OR technologies: operating system

Tails or **The Amnesic Incognito Live System** is a security-focused Debian-based Linux distribution aimed at preserving privacy and anonymity.[3] All its outgoing connections are forced to go through Tor,[4] and non-anonymous connections are blocked. The system is designed to be booted as a live DVD or live USB, and will leave no digital footprint on the machine unless explicitly told to do so. The Tor Project has provided financial support for its development.[5]

Tails



OS family	Unix-like
Working state	Current
Source model	Open source
Initial release	June 23, 2009; 6 years ago
Latest release	1.8.2 / January 10, 2016; 6 days ago[1]

OR technologies: community

Tor

HOME | ARCHIVES | ABOUT TOR | DONATE

Blogs

Transparency, Openness, and our 2014 Financials

Posted January 14th, 2016 by arma in donations, financial statements, Form 990, Form PC

After completing the standard audit, our 2014 state and federal tax filings are available. We publish all of our related tax documents because we believe in transparency.

Tor's annual revenue in 2014 held steady at about $2.5 million. Tor's budget is modest considering the number of people involved and the impact we have. And it is dwarfed by the budgets that our adversaries are spending to make the world a more dangerous and less free place.

To achieve our goals, which include scaling our user base, we fund about 20 contractors and staff members (some part time, some full time) and rely on thousands of volunteers to do everything from systems administration to outreach. Our relay operators are also volunteers, and in 2014 we grew their number to almost 7,000 — helped along by the Electronic Frontier Foundation's wonderful Tor Challenge, which netted 1,635 relays. Our user base is up to several million people each day.

Transparency doesn't just mean that we show you our source code (though of course we do). The second layer to transparency is publishing specifications to explain what we thought we implemented in the source code. And the layer above that is publishing design documents and research papers to explain why we chose to build it that way, including analyzing the security implications and the tradeoffs of alternate designs. The reason for all these layers is to help people evaluate every level of our system: whether we chose the right design, whether we turned that design into a concrete plan that will keep people safe, and whether we correctly implemented this plan. Tor gets a huge amount of analysis and attention from professors and university research groups down to individual programmers around the world, and this consistent peer review is one of our core strengths over the past decade.

As we look toward the future, we are grateful for our institutional funding, but we want to expand and diversify our funding too. The recent donations campaign is a great example

Upcoming events

- No upcoming events available

full calendar

Recent blog posts

- Transparency, Openness, and our 2014 Financials
- This Is What a Tor Supporter Looks Like – Jacob Appelbaum
- Tor Browser 5.5a6-hardened is released
- Tor Browser 5.5a6 is released
- Tor Browser 5.0.7 is released
- This is What a Tor Supporter Looks Like: Ethan Zuckerman
- This is What a a Tor Supporter Looks Like: Susan Landau
- This is What a Tor Supporter Looks Like: Giordano Nanni
- This is What a Tor Supporter

the big picture

Freely available (Open Source), unencumbered.

Comes with a spec and full documentation

- Dresden and Aachen implemented compatible Java Tor clients; researchers use it to study anonymity.

1500 active relays, 200000+ active users, >1Gbit/s.

Official US 501(c)(3) nonprofit. Eight full-time developers (!), dozens more dedicated volunteers.

Funding from

- US DoD, Electronic Frontier Foundation, Voice of America, a French NGO, Google, NLnet, Human Rights Watch, …

Part IV

Tor

Tor attacker model

A *global passive adversary* is common to analyse theoretical anonymity designs.

Tor does *not* protect against such a strong adversary. Tor focusses on adversary whose typical goal is to observe initiator and responder, and who can:

- *observe* some fraction of network traffic;
- *generate*, *modify*, *delete*, or *delay* traffic;
- *operate* onion routers of his own;
- *compromise* some fraction of the onion routers or their keys.

Tor attacker model

Attackers can be either:

passive: confirm Alice is talking to Bob if timing and volume of the traffic are distinct enough;

active: induce timing signatures on traffic to *force* distinct patterns.

Typical active attacks also include selectively denying service to:

- trustworthy routers to move users to compromised routers;
- users to see if traffic elsewhere in the network stops.

The adversary can try to decrease the network's reliability by attacking nodes or by performing antisocial activities from reliable nodes and trying to get them taken down: making the network unreliable flushes users to other less anonymous and easier to attack systems.

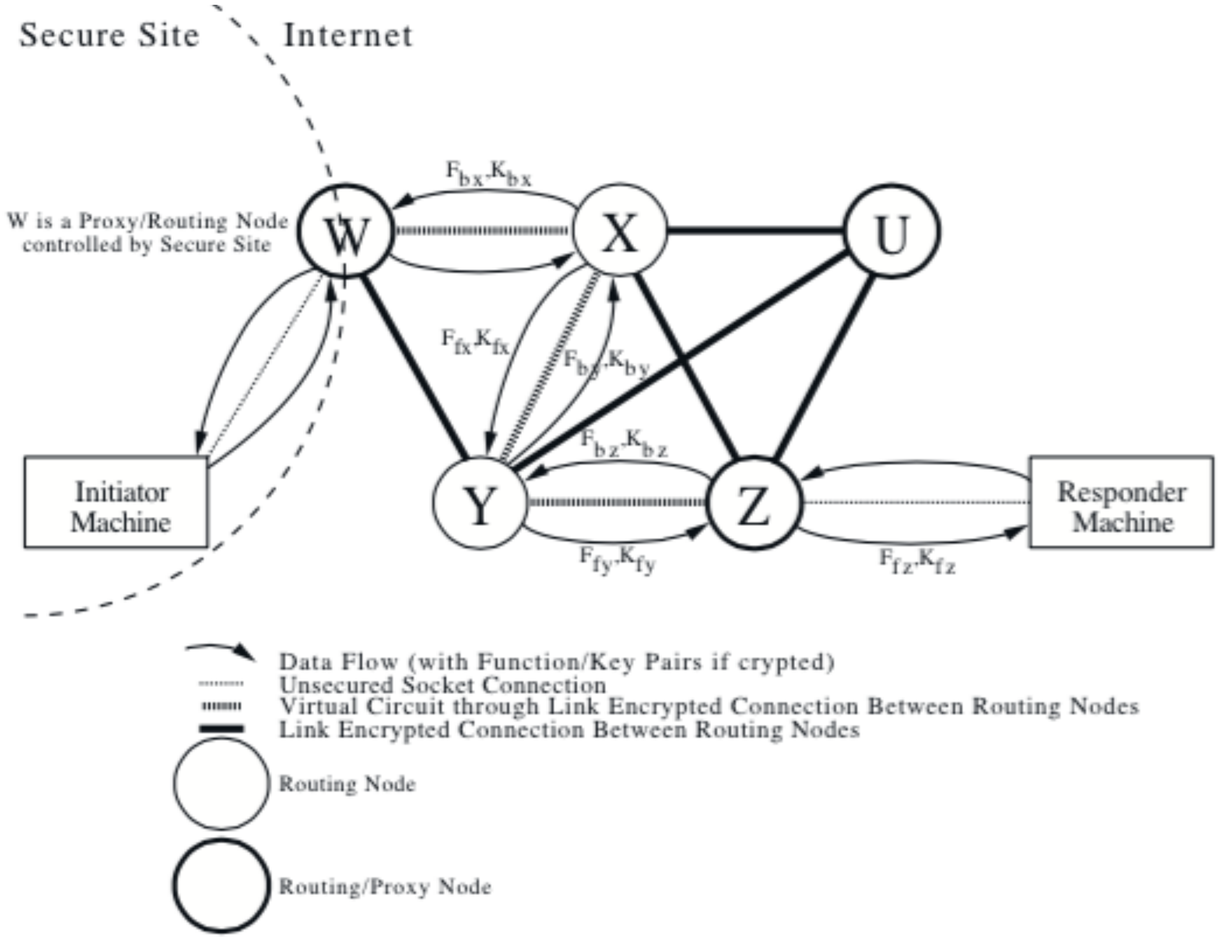
Tor: an overlay network

Onion routers (OR) run as normal user-level processes with no special privileges.

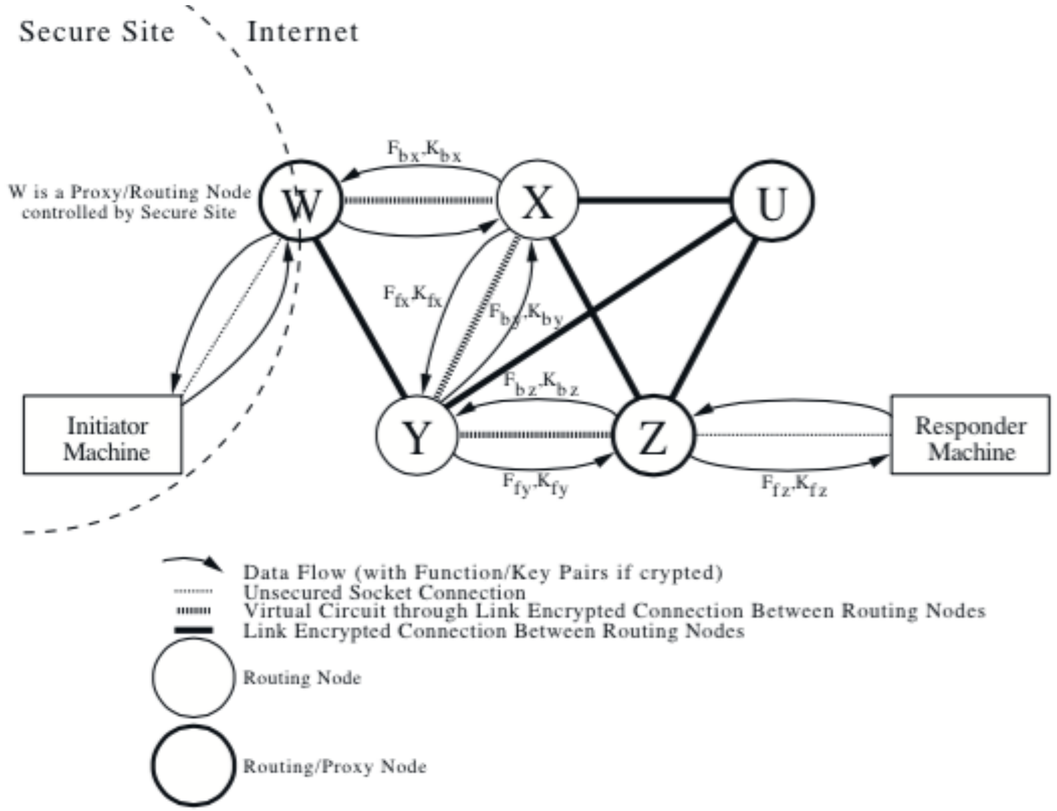
Users run a local software called *onion proxy* (OP) to:

fetch directories; establish/destroy circuits across the network; handle connections

Circuits connect users to the requested destinations and relay data.



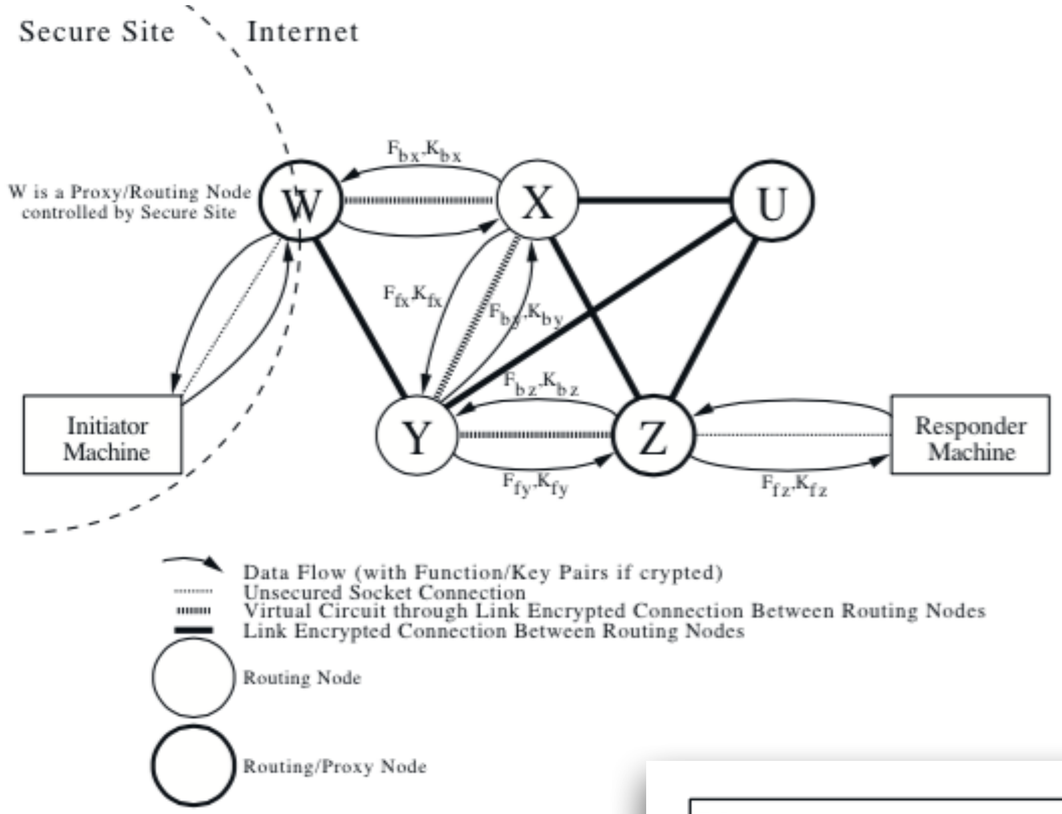
Tor: more details on the onion



node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

Tor: more details on the onion



node P_x receives looks like this

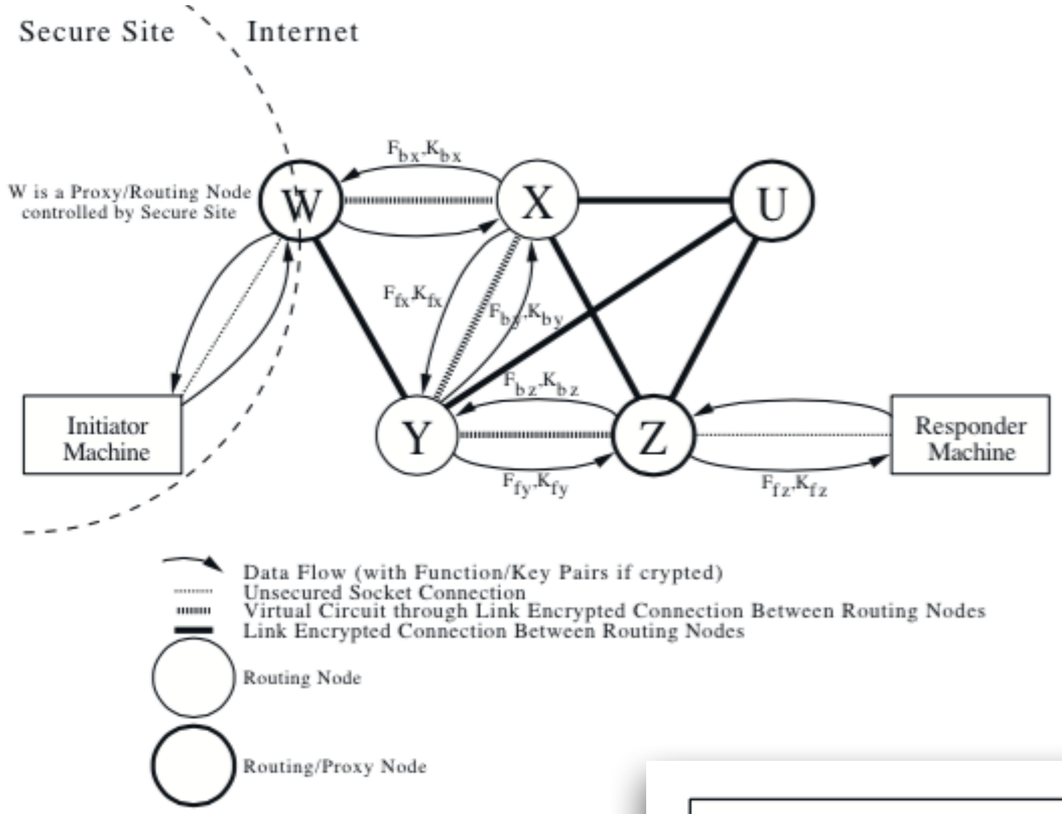
$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

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

Tor: more details on the onion



node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

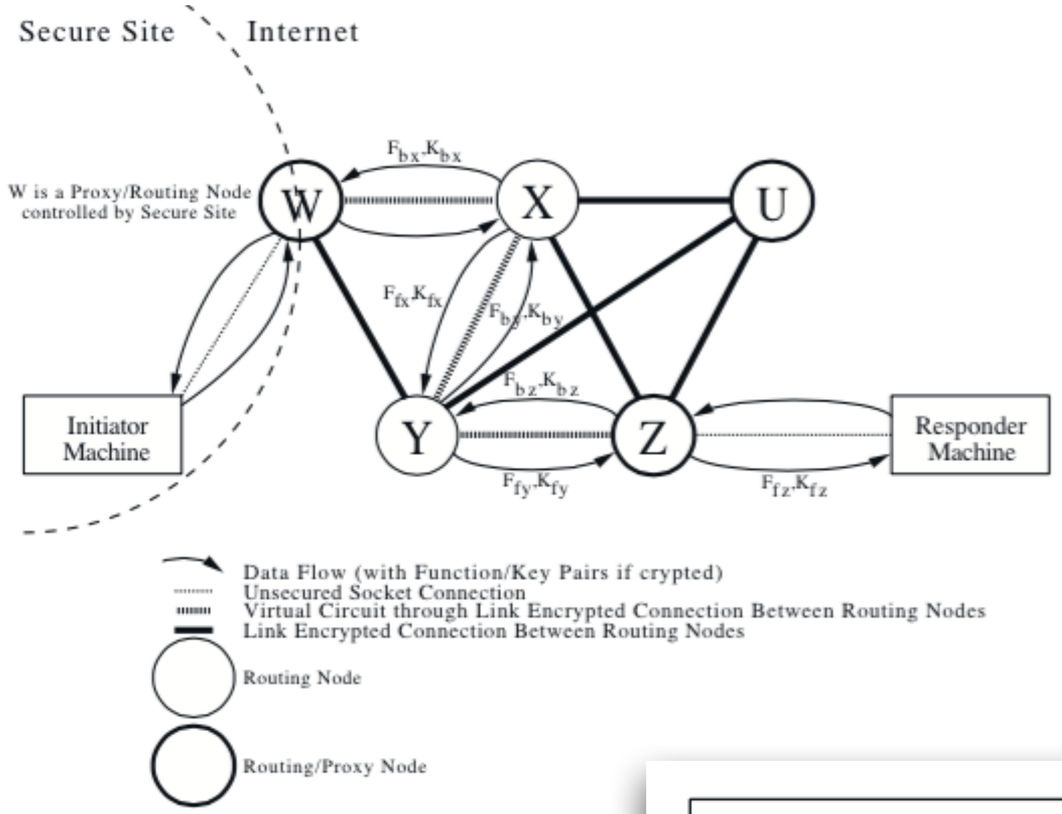
timeout

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},

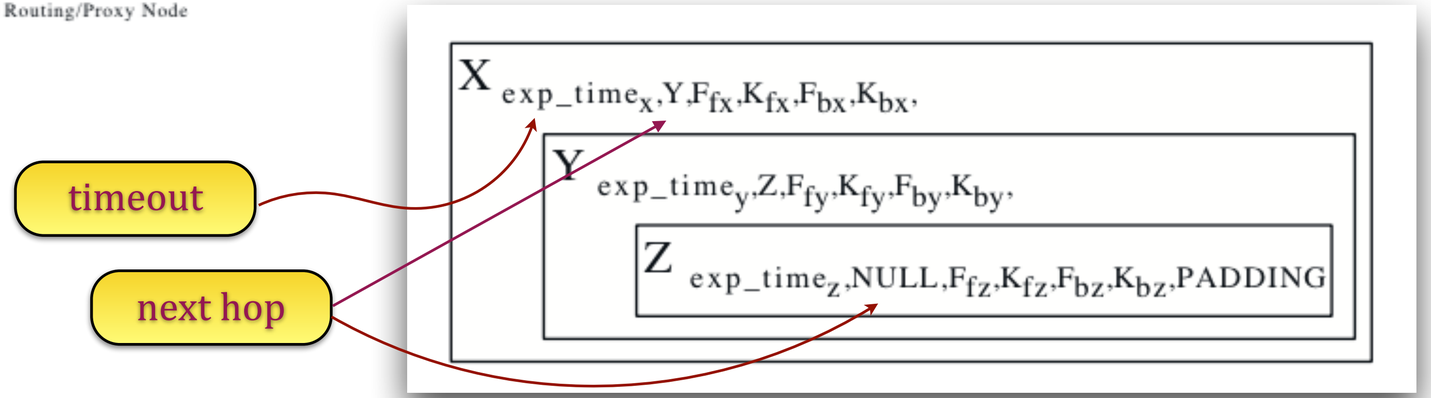
Z exp_time_z,NULL,F_{fz},K_{fz},F_{bz},K_{bz},PADDING

Tor: more details on the onion

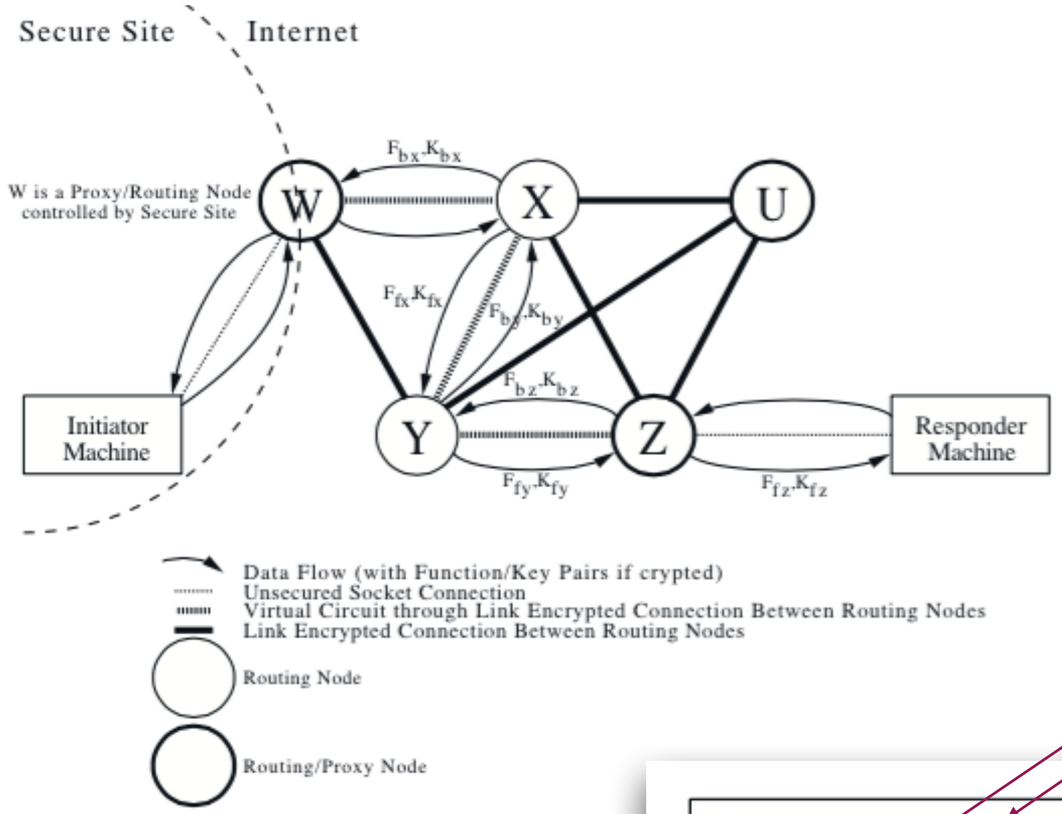


node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$



Tor: more details on the onion



node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

forward crypto function and key

timeout

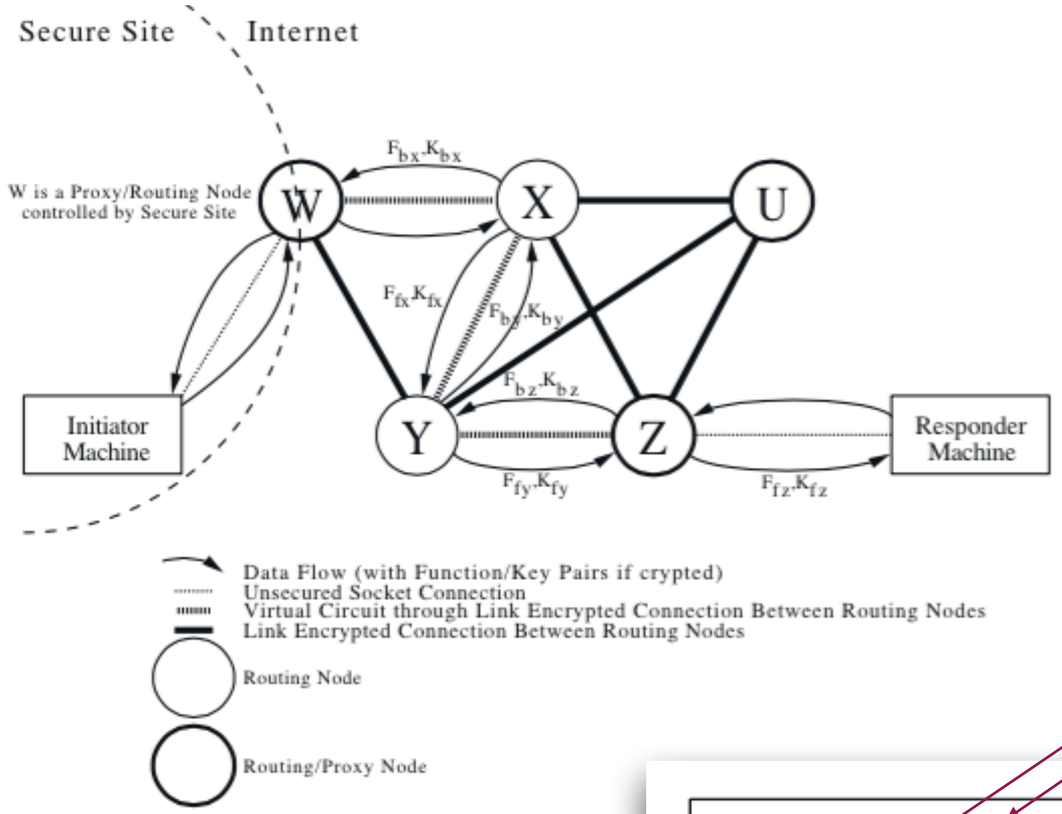
next hop

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

Tor: more details on the onion



node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

forward crypto function a

backward crypto function and key

timeout

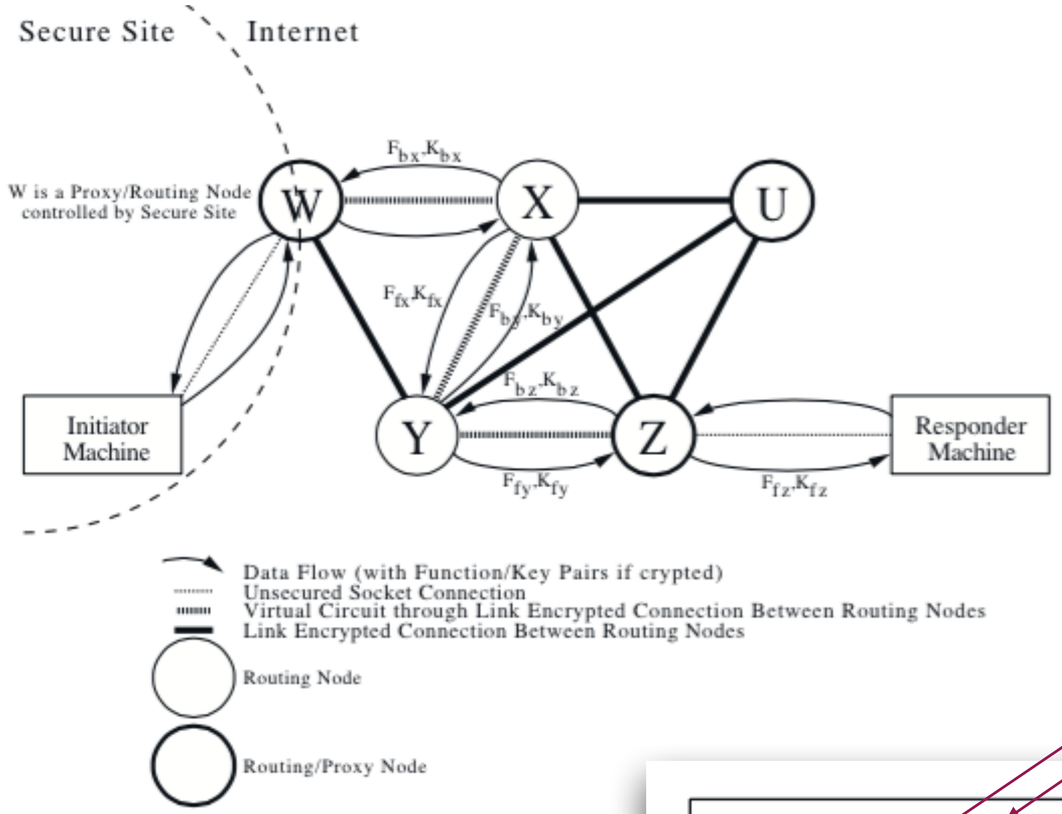
next hop

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

Tor: more details on the onion



node P_x receives looks like this

$$\{exp_time, next_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$$

forward crypto function a

backward crypto function and key

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

timeout

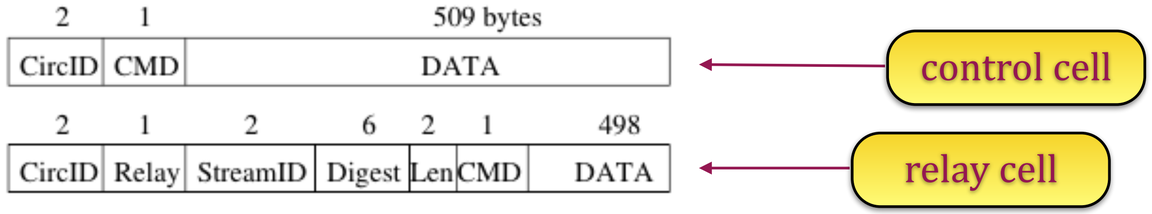
next hop

keep size constant

Tor: architecture

ORs communicate between them and with OPs via TLS connections to prevent eavesdropping, impersonation and tampering.

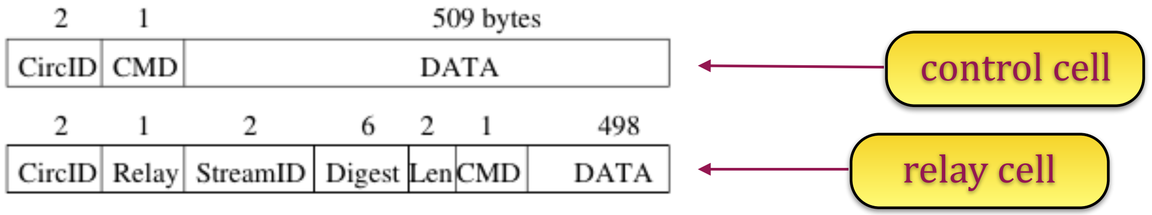
The unit of communication is a *cell* of fixed size: 512 bytes.



Circuit identifiers are connection-specific: the same circuit has a different circID on each OP/OR or OR/OR connection it traverses.

The control cell commands are: *padding* (ignored here); *create*/*created*; and *destroy*.

Tor: architecture



Relay cells have an additional header at the front of the payload:

- streamID (as many streams can be multiplexed over a circuit);
- end-to-end checksum for integrity checking;
- length of the relay payload (to identify the padding);
- relay command (*begin*, *end*, *extend*, …)

The entire contents of the relay header and the relay cell payload are encrypted or decrypted together as the relay cell moves along the circuit, using the 128-bit AES.

Tor: circuits and streams

Building circuits is expensive!

So each circuit is shared by many TCP streams.

Yet, reusing circuits opens to attacks that link requests together! So each circuit is replaced by a new circuit say once a minute.

In the real protocol A user's OP constructs circuits *incrementally*, negotiating a symmetric key with each OR on the circuit, *one hop at a time.*

Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.

Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.
2. Bob responds with a *created* cell containing g^y along with a hash of the negotiated key $K = g^{xy}$. Now they can send one another relay cells encrypted with the negotiated key!

Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.
2. Bob responds with a *created* cell containing g^y along with a hash of the negotiated key $K = g^{xy}$. Now they can send one another relay cells encrypted with the negotiated key!
3. Alice sends a *relay extend* cell to Bob, specifying the address of the next OR, Carol, and g^{x2} encrypted for her.

Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.
2. Bob responds with a *created* cell containing g^y along with a hash of the negotiated key $K = g^{xy}$. Now they can send one another relay cells encrypted with the negotiated key!
3. Alice sends a *relay extend* cell to Bob, specifying the address of the next OR, Carol, and g^{x2} encrypted for her.
4. Bob copies the half-handshake into a *create* cell, and passes it to Carol to extend the circuit. (Bob chooses a new circID CBC, which Alice never needs to know.)

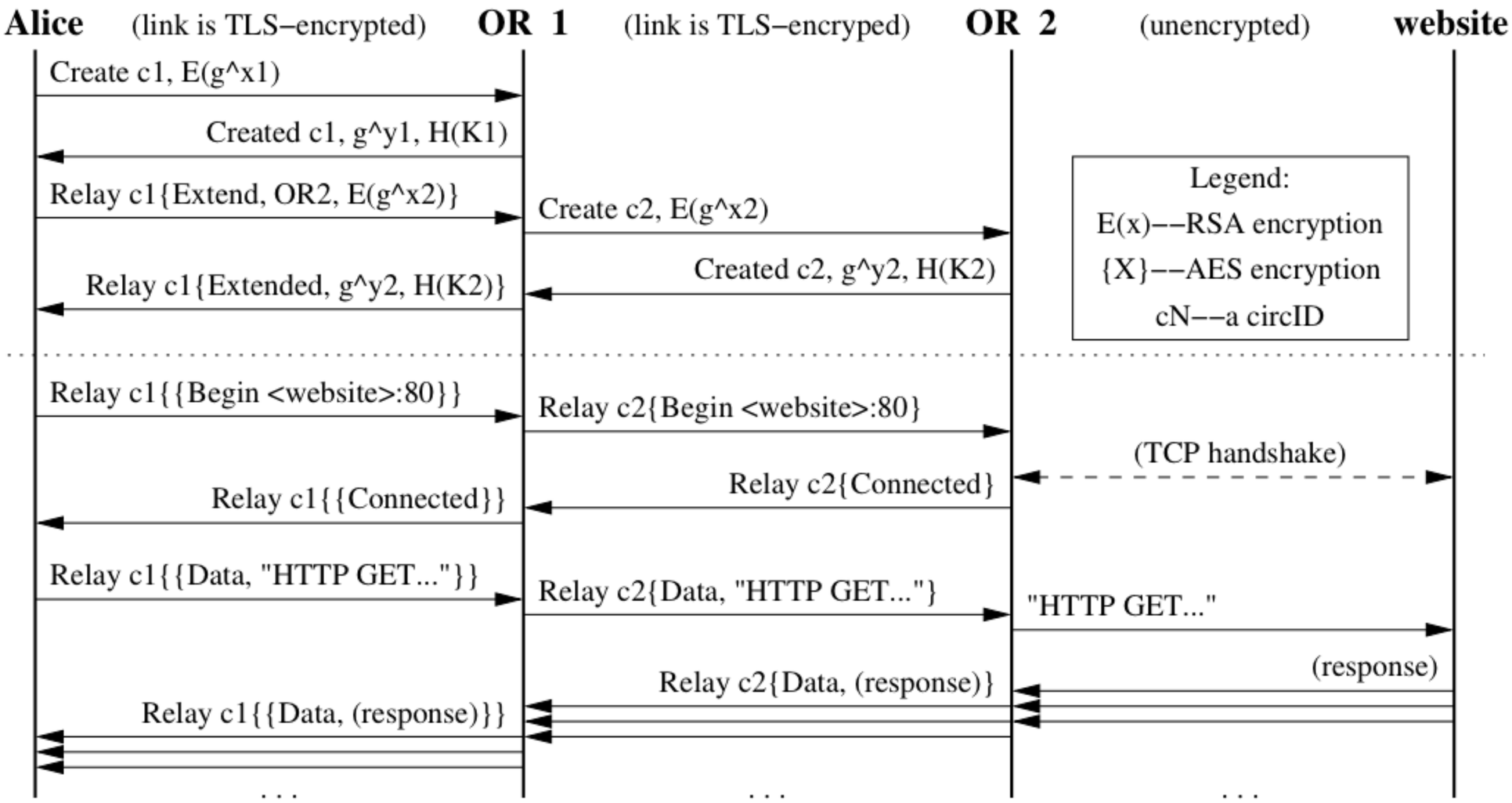
Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.
2. Bob responds with a *created* cell containing g^y along with a hash of the negotiated key $K = g^{xy}$. Now they can send one another relay cells encrypted with the negotiated key!
3. Alice sends a *relay extend* cell to Bob, specifying the address of the next OR, Carol, and g^{x2} encrypted for her.
4. Bob copies the half-handshake into a *create* cell, and passes it to Carol to extend the circuit. (Bob chooses a new circID CBC, which Alice never needs to know.)
5. When Carol responds with a *created* cell, Bob wraps the payload into a *relay extended* cell and passes it back to Alice. Now the circuit is extended to Carol, and Alice and Carol share a common key $K2 = g^{x2y2}$, which Bob does not know. Nor does Carol know Alice!

Tor: building circuits

1. Alice sends a *create* cell to the first node in her chosen path, Bob and chooses a new circID, CAB. The payload contains the first half of the Diffie-Hellman handshake g^x encrypted to Bob's onion key.
2. Bob responds with a *created* cell containing g^y along with a hash of the negotiated key $K = g^{xy}$. Now they can send one another relay cells encrypted with the negotiated key!
3. Alice sends a *relay extend* cell to Bob, specifying the address of the next OR, Carol, and g^{x2} encrypted for her.
4. Bob copies the half-handshake into a *create* cell, and passes it to Carol to extend the circuit. (Bob chooses a new circID CBC, which Alice never needs to know.)
5. When Carol responds with a *created* cell, Bob wraps the payload into a *relay extended* cell and passes it back to Alice. Now the circuit is extended to Carol, and Alice and Carol share a common key $K2 = g^{x2y2}$, which Bob does not know. Nor does Carol know Alice!
6. To extend the circuit to a third node or beyond, Alice proceeds similarly, always asking the last node in the circuit to extend one hop further.
   Question: what will Alice ask Bob now?

Alice builds a two-hop circuit and…



Exercise: draw the diagram to build a 3-hop circuit

Tor: relay cells

Data communication occur through relay cells. Upon receiving a relay cell an OR:

- looks up the corresponding circuit by circID to find the session key;
- decrypts the relay header and payload with the session key for that circuit;
- looks up the circID and OR corresponding to the next step in that circuit;
- replaces the circID in the cell as appropriate;
- sends the decrypted relay cell to the next OR.

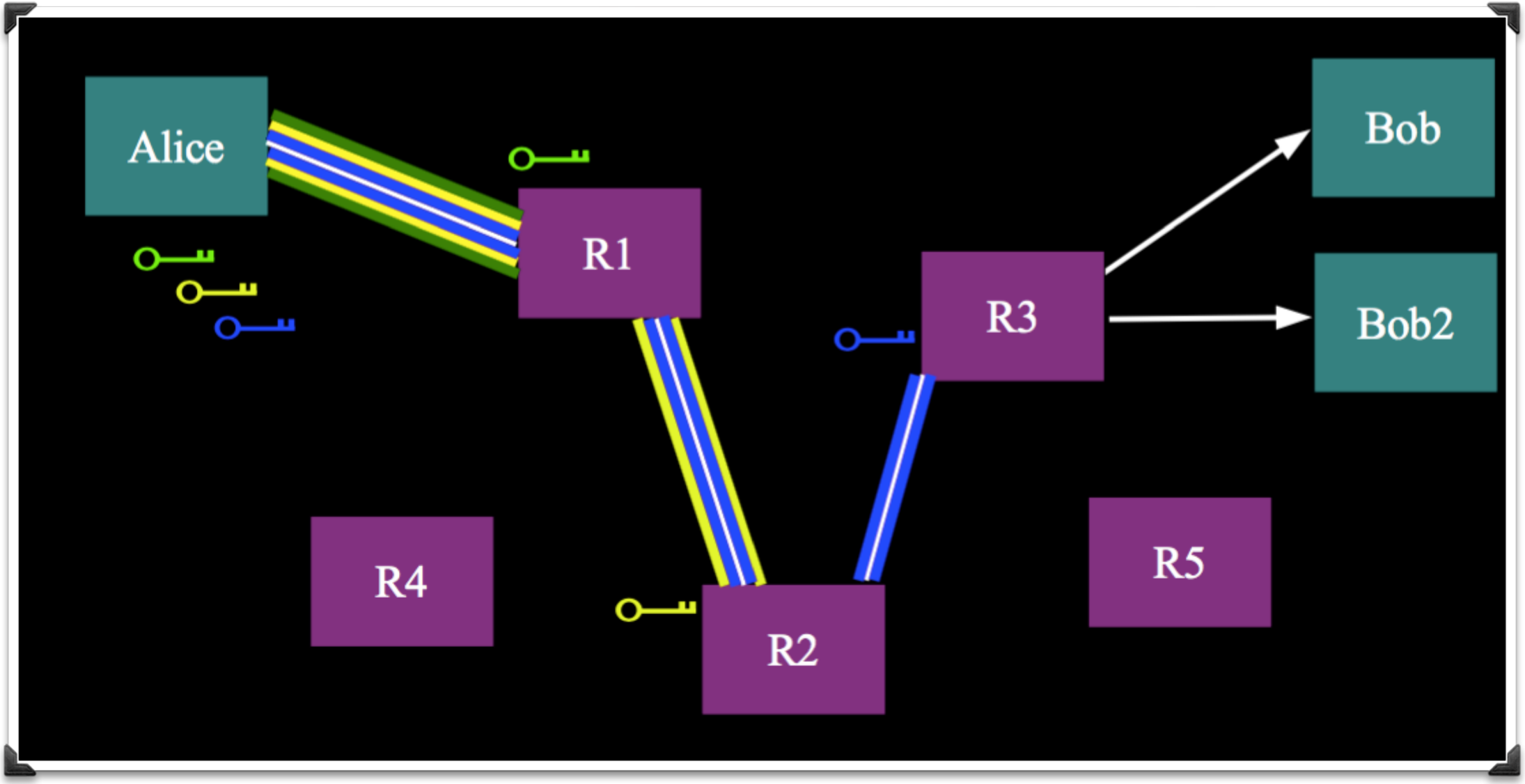
To construct a relay cell addressed to a given OR, Alice:

- assigns the digest;
- iteratively encrypts the cell payload with the symmetric key of each hop up to that OR.

To reply to Alice,

- OR encrypts the cell's relay header and payload with the single key it shares with Alice, and sends the cell back toward Alice along the circuit.
- subsequent ORs add further layers of encryption as they relay the cell back to Alice, who will have to decrypt all the layers one by one.

Tor: look at it as nested crypto-tunnels



Tor: destroy circuits

To tear down a circuit, Alice sends a destroy control cell.

Each OR in the circuit receives the destroy cell, closes all streams on that circuit, and passes a new destroy cell forward.

Just as circuits are built incrementally, they can also be torn down incrementally.

Part V

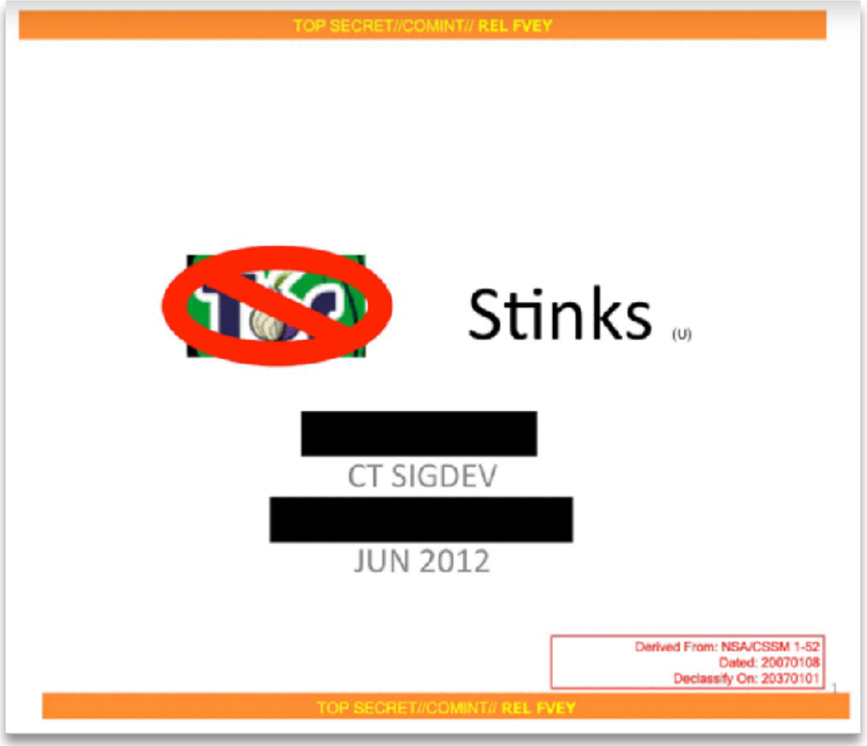
Tor vulnerabilities

Tor sucks!

News > US news > The NSA files

'Tor Stinks' presentation – read the full document

Top-secret presentation says 'We will never be able to de-anonymize all Tor users all the time' but 'with manual analysis we can de-anonymize a very small fraction of Tor users'

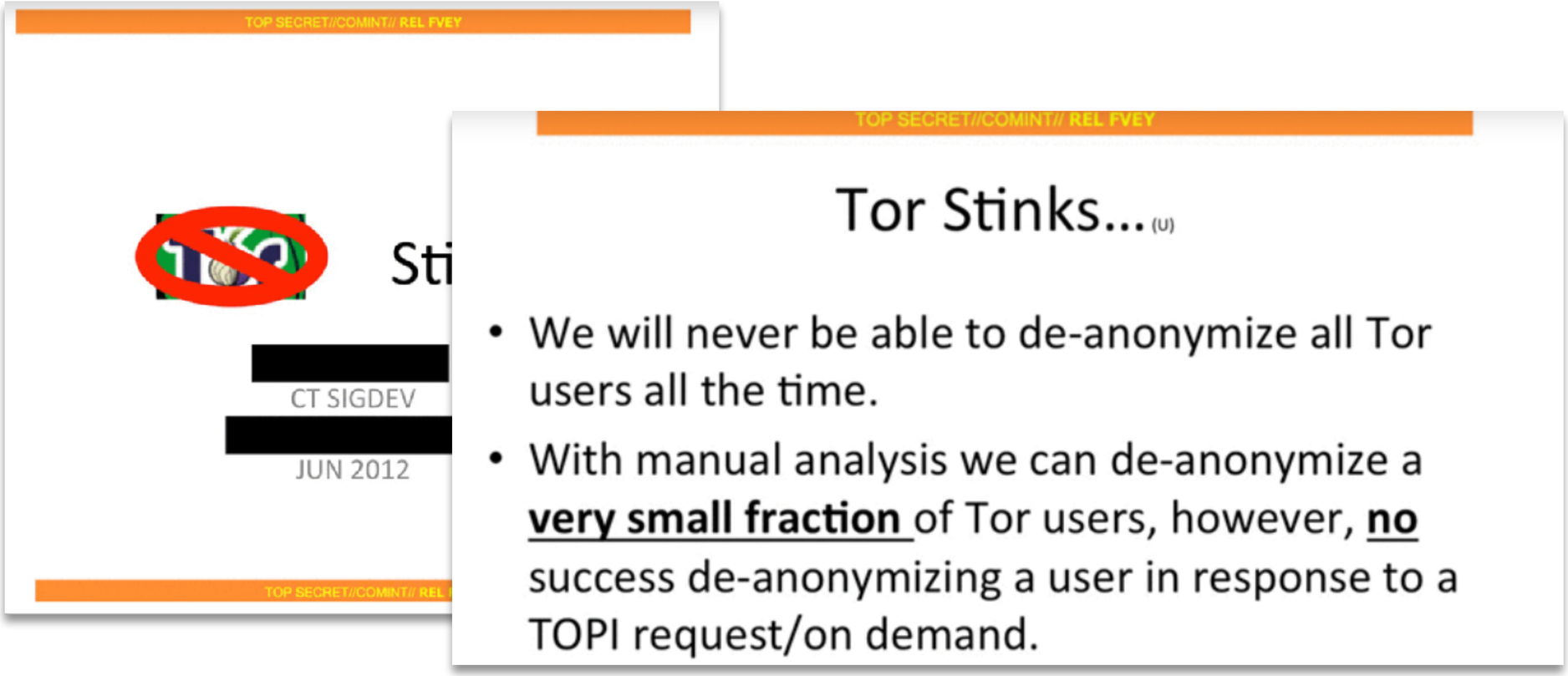


Tor sucks!

News > US news > The NSA files

'Tor Stinks' presentation – read the full document

Top-secret presentation says 'We will never be able to de-anonymize all Tor users all the time' but 'with manual analysis we can de-anonymize a very small fraction of Tor users'



attacks against Tor

UNIVERSITY OF Southampton

general attacks:

- denial of service
- OR/service/destination takeover

passive attacks:

- observing (and analyse) a user's traffic patterns
- observing user contents (may not be encrypted at the server end)
- observing configuration options (minority choices leave you exposed)
- end-to-end timing correlation (powerful technique)
- end-to-end size correlation (see above)
- website fingerprinting (build a db of website characteristics)
- external knowledge (rely on further "off-protocol" knowledge)

attacks against Tor

UNIVERSITY OF Southampton

active attacks:

- compromise keys (well, of course!)
- iterated compromise (travel up the chain, after compromising OR)
- run a recipient (can generate powerful timing attacks)
- run a onion proxy / a hostile OR (aims to spoof / be 1st & last in the chain)
- introduce timing in messages (powerful technique)
- clogging / congestion attacks (not effective)
- tagging messages (not effective)
- replay attacks (not effective)
- smear attacks (bring Tor into disrepute)
- cookies, history, browser window size, javascript refresh, …

directory attacks

- subvert Tor infrastructure (takeover of compromise ORs and IPs)

circuit fingerprinting

UNIVERSITY OF Southampton

Tor connection vulnerability uncloaks hidden web services

Can "circuit fingerprinting" reveal the true location of Tor websites and services?

By Charlie Os

0

MIT researchers h
services in the De

As reported by N
Technology (MIT)
anonymous hidde

Researchers devise passive attacks for identifying Tor hidden services

Posted on 31 July 2015.

A group of MIT researchers have devised two attacks that could identify, with a high degree of certainty (88%), an anonymous hidden service or client.

Hidden services are servers configured to receive inbound connections only through Tor, and can be accessed via their .onion address. In theory, it should be impossible to discover their true location, but in practice, there are attacks, such as this latest one, that could "uncloak" them.

"Over the past few years, hidden services have witnessed various active attacks in the wild, resulting in several takedowns," the researchers noted. "To examine the security of the design of hidden services, a handful of attacks have been proposed against them. While

circuit fingerprinting

UNIVERSITY OF Southampton

Tor connection vulnerability uncloaks hidden web services

Can "circuit fingerprinting" reveal the true location of Tor websites and services?

By Charlie Os

0

MIT researchers h
services in the De

As reported by N
Technology (MIT)
anonymous hidde

Researchers devise passive attacks for identifying Tor hidden services

Posted on 31 July 2015.

A group of MIT researchers have devised two attacks that could identify, with a high degree o
client.

Hidden services are
only through Tor, an
theory, it should be i
practice, there are a
them.

"Over the past few y
active attacks in the
researchers noted. "To examine the security of the design of hidden services, a handful of attacks have been proposed against them. While

Typical attacks (all active):

(1) crafted signals to speed up discovery of entry points

(2) congestion attacks to bias entry point selection

They require a malicious client to continuously attempt to connect to the hidden service

This attack (passive):

(1) characterises "hidden service" traffic(2) exploits website profiling to denonymise user

circuit fingerprinting

Circuit Fingerprinting Attacks:
Passive Deanonymization of Tor Hidden Services

Albert Kwon[†], Mashael AlSabah[‡§†*], David Lazar[†], Marc Dacier[‡], and Srinivas Devadas[†]

[†]*Massachusetts Institute of Technology*, {kwonal, lazard, devadas}@mit.edu
[‡]*Qatar Computing Research Institute*, mdacier@qf.org.qa
[§]*Qatar University*, malsabah@qu.edu.qa

homework: *read and discuss this paper*

conclusion

think i'll stop here for now