Privacy Enhancing Technologies

Chapter: Anonymous Communication

Dr. Christiane Kuhn <christiane.kuhn@kit.edu>

Thanks to Patricia Arias-Cabarcos for allowing me to reuse and adapt slides of her lecture on anonymous communication.

Helmholtz Center for Applied Security Technology

Karlsruhe Institute of Technology

Privacy Enhancing Technologies

Chapter: Anonymous Communication

Dr. Christiane Kuhn <christiane.kuhn@kit.edu>

Thanks to Patricia Arias-Cabarcos for allowing me to reuse and adapt slides of her lecture on anonymous communication.

Helmholtz Center for Applied Security Technology

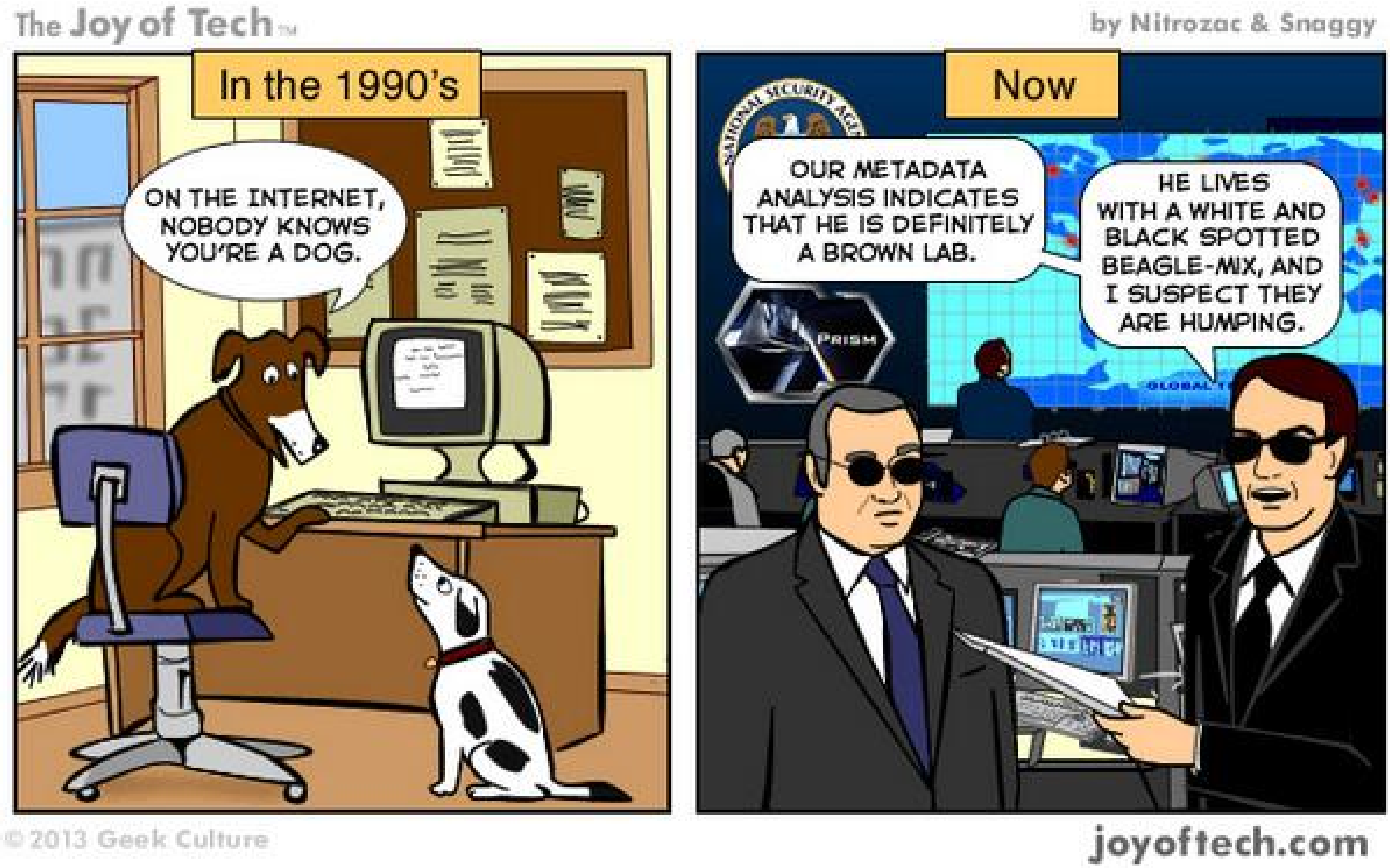
Can you click yes?

Learning Goals

- Understand the Problem
  - Motivation & Setting
  - Dimensions & Terminology

- Understand the Solution(-space)
  - Solution ideas and prominent protocols
  - Effects of design decisions

Motivation



The Joy of Tech™
by Nitrozac & Snaggy
In the 1990's
ON THE INTERNET, NOBODY KNOWS YOU'RE A DOG.
Now
OUR METADATA ANALYSIS INDICATES THAT HE IS DEFINITELY A BROWN LAB.
HE LIVES WITH A WHITE AND BLACK SPOTTED BEAGLE-MIX, AND I SUSPECT THEY ARE HUMPING.
PRISM
© 2013 Geek Culture
joyoftech.com

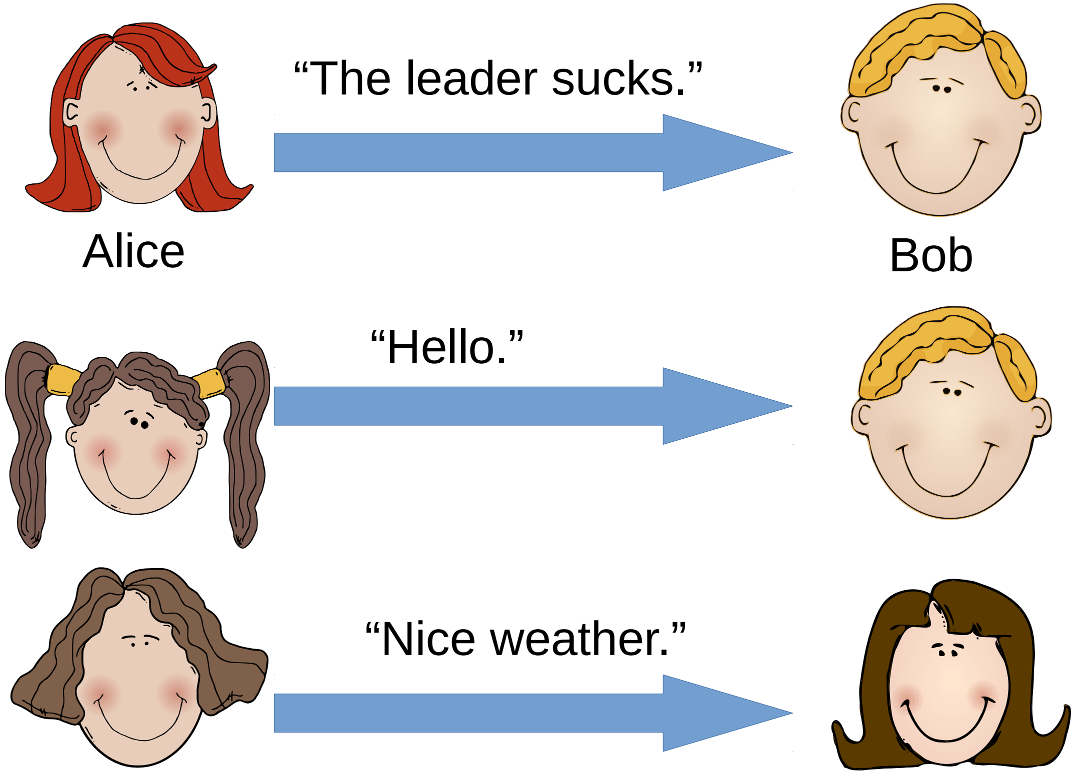
Motivation

- Protect Privacy in Communications to:
  - View sensitive content
  - Avoid impersonation
  - Avoid profiling and tracking by advertising companies (price discrimination)
  - Avoid profiling and tracking by governments (manipulation)
  - Guarantee freedom of speech
  - Enable applications: electronic voting, whistle blowing,…

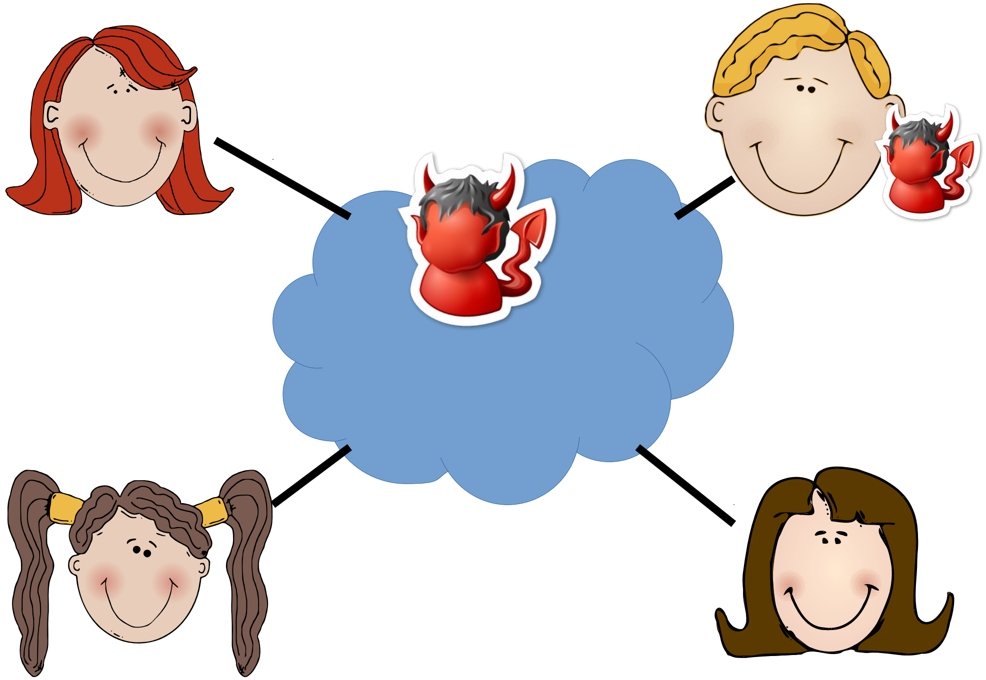
Setting

Communications that are happening

Sender message receiver



Network, on which they happen



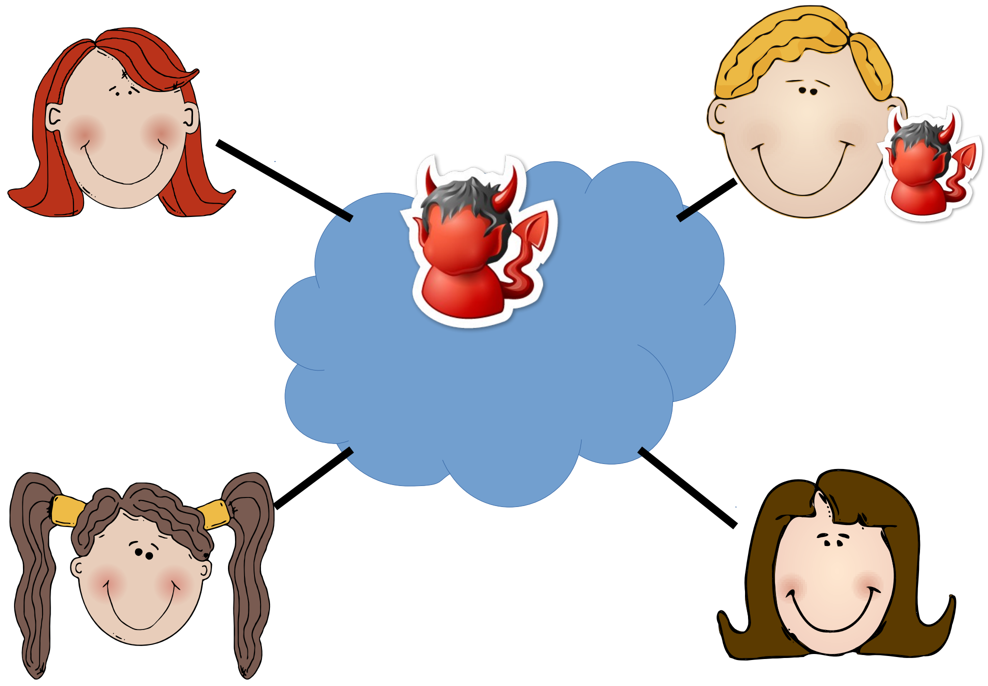
Does encryption protect Alice from the adversary?

Encryption is not enough

- Does not hide anything if the receiver is adversarial
- Does not hide meta data:
  - Sender-receiver relationships (IP addresses)
  - Activity
  - Cookies
  - Browser fingerprinting

    → all can be used to identify and profile users

✉ Encryption is an amazing tool, but not enough!



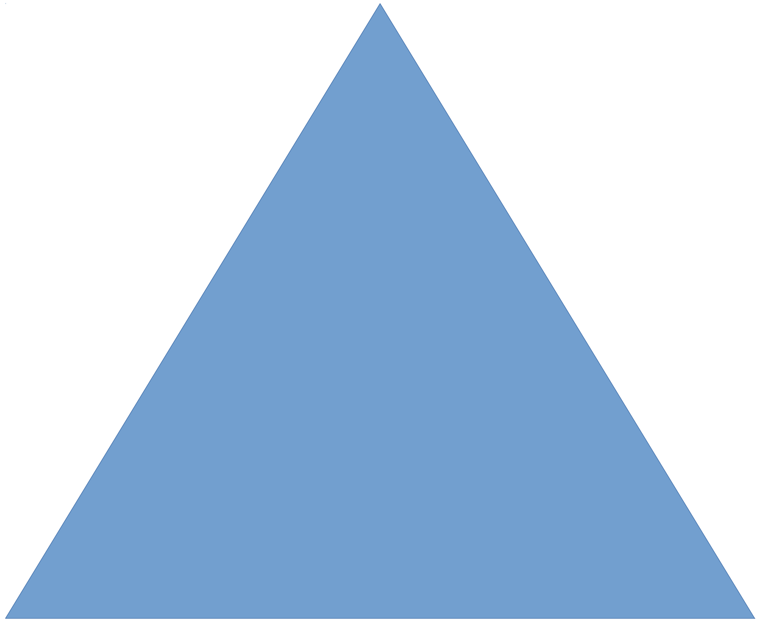
Learning Goals

- Understand the Problem
  - Motivation & Setting
  - Dimensions & Terminology

- Understand the Solution(-space)
  - Solution ideas and prominent protocols
  - Effects of design decisions

Criteria

What’s protected?

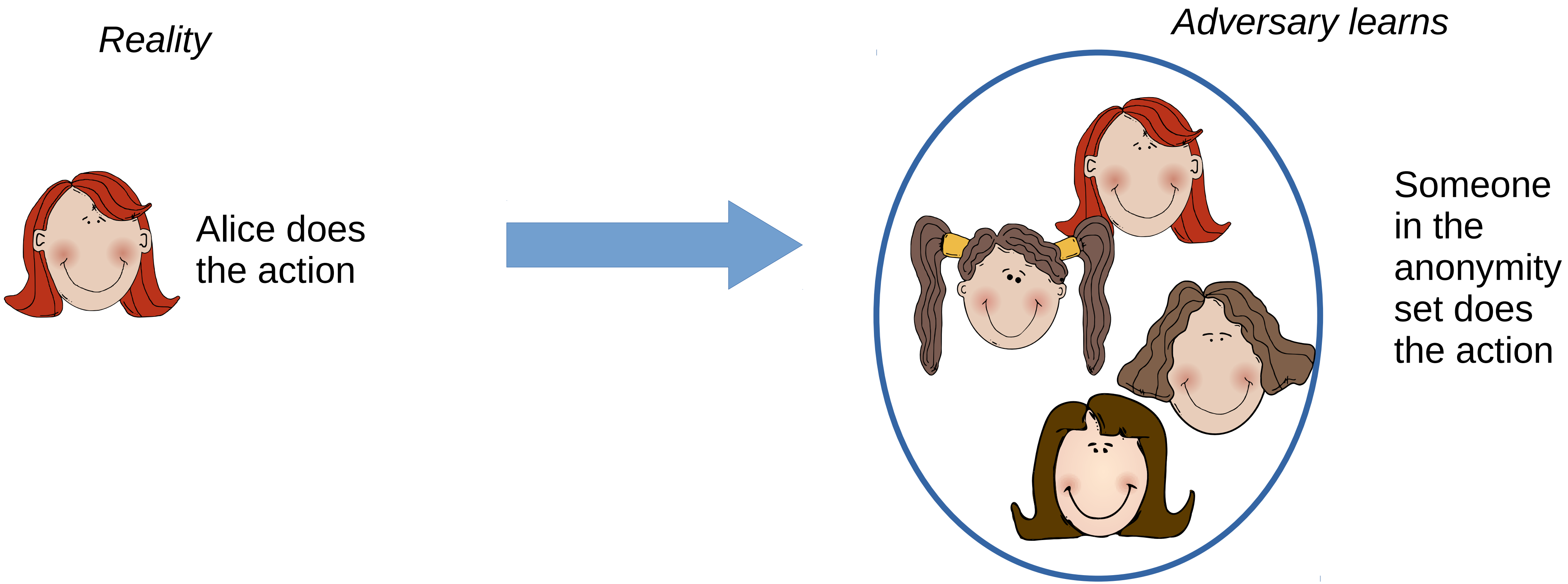


Against what adversary?

At what cost?

What’s protected? Terminology

Anonymity: “Anonymity of a subject means that the subject is not identifiable within a set of subjects, the **anonymity set**. ”



https://tools.ietf.org/id/draft-hansen-privacy-terminology-00.html

What’s protected?

Typically of interest as subjects: Senders, Receivers

→ we’ll focus on sender protection for this lecture

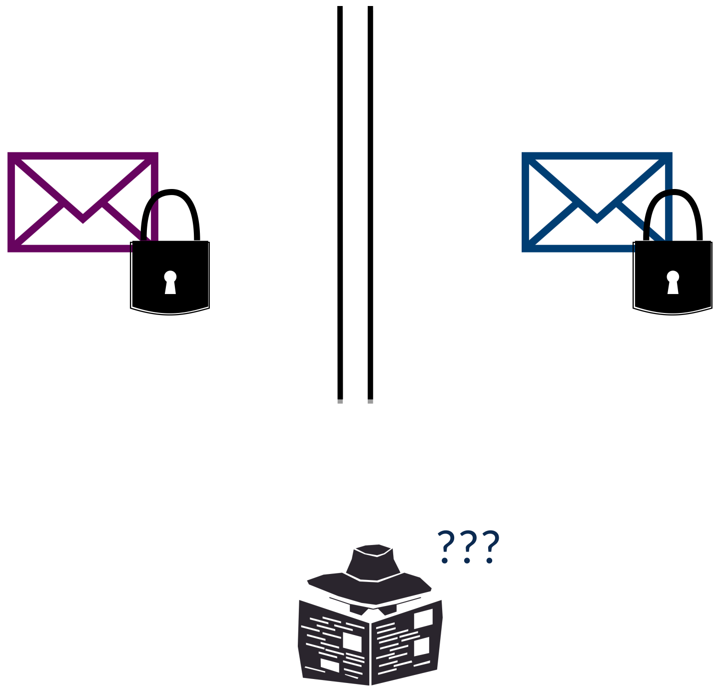
Most common goals (involving senders):

- Sender Anonymity – we do not learn who sends which message
- Relationship Anonymity – we do not learn who communicates with whom
- Sender Unobservability – we do not learn who sends something

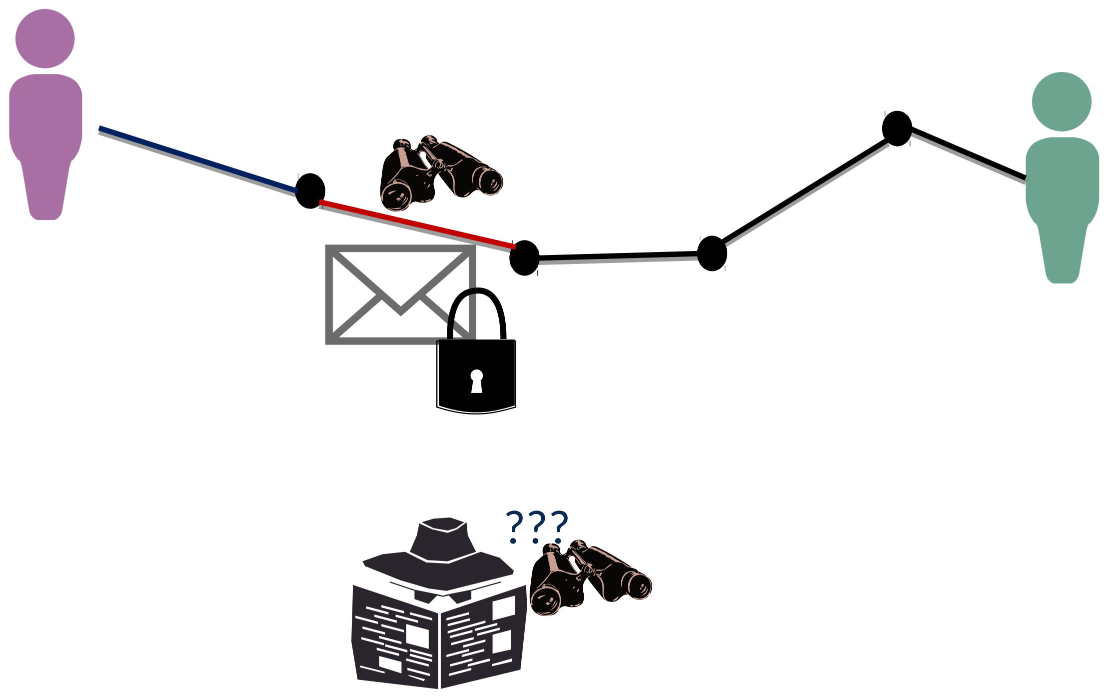
→ Informal goals lead to inconsistencies

Recall: IND-CPA

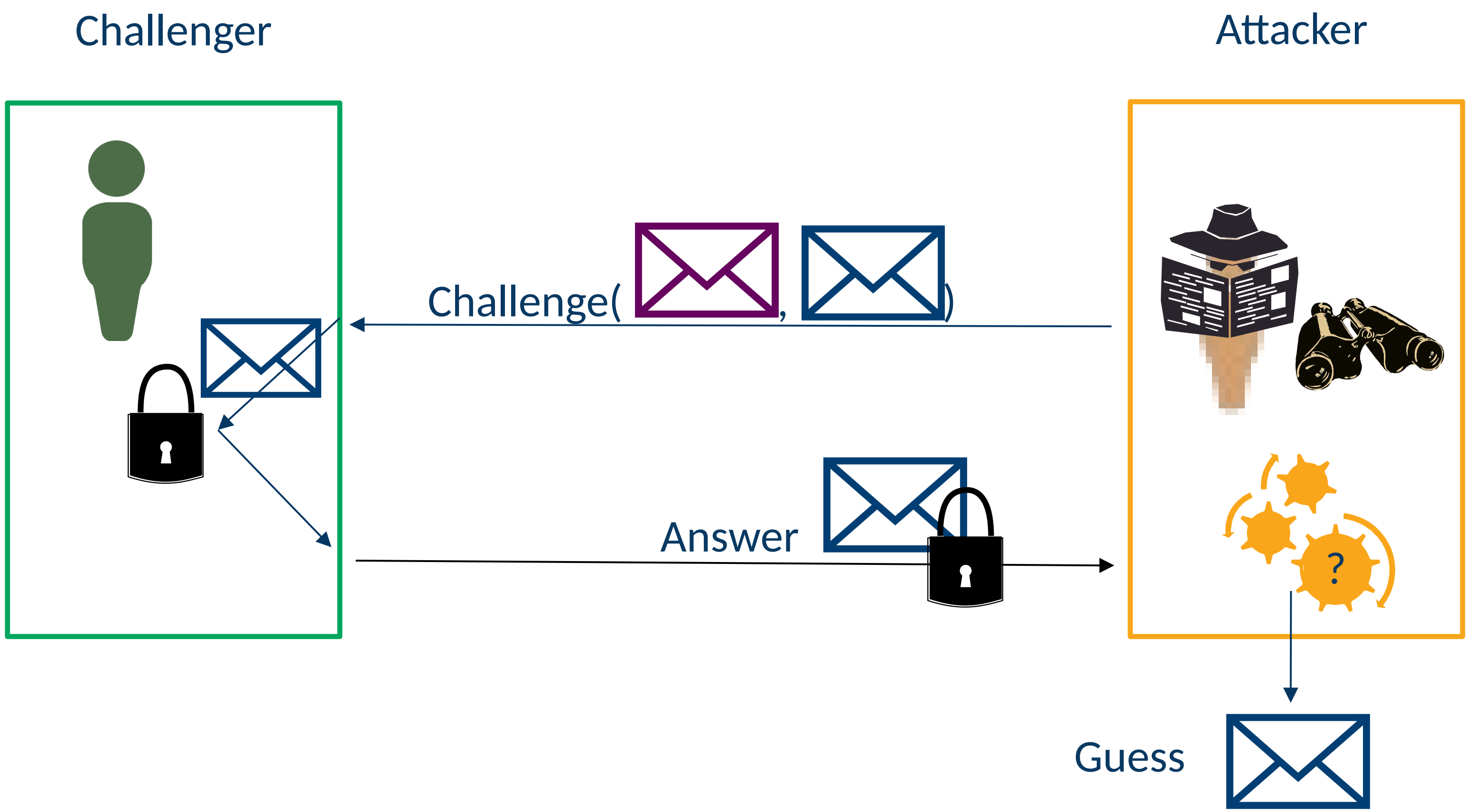
Indistinguishable cases



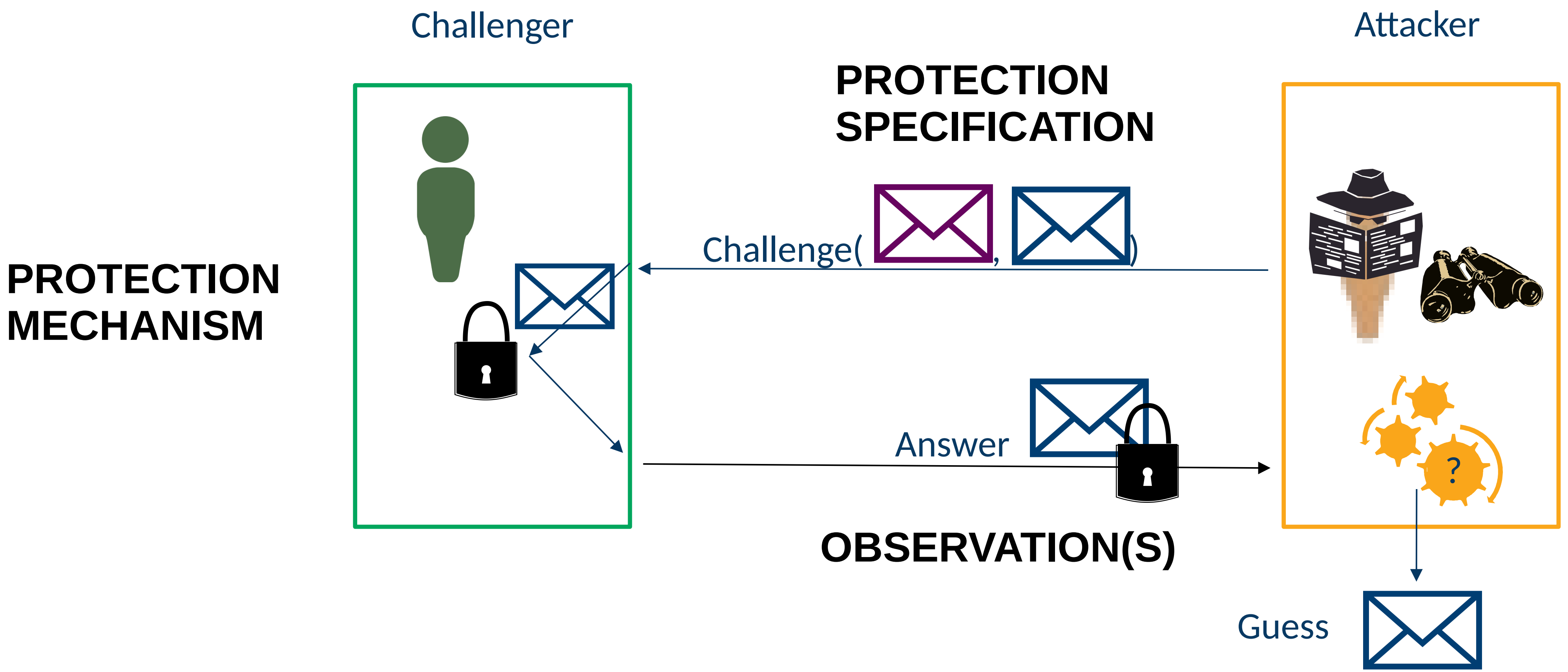
Setting



Recall: IND-CPA

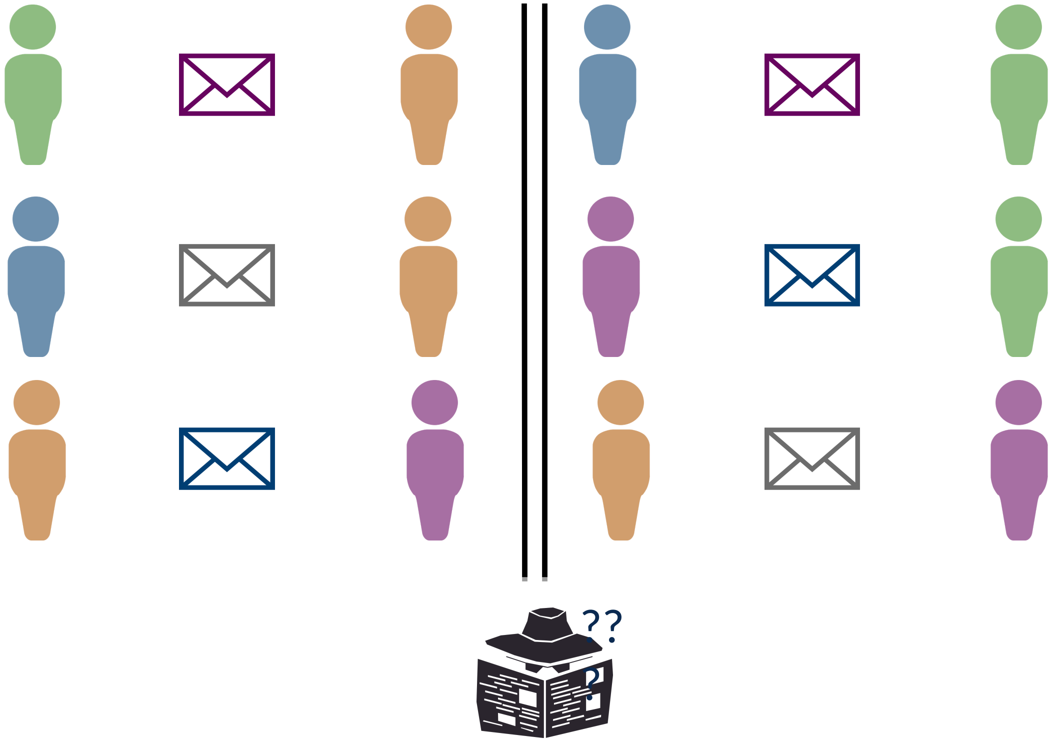


Recall: IND-CPA

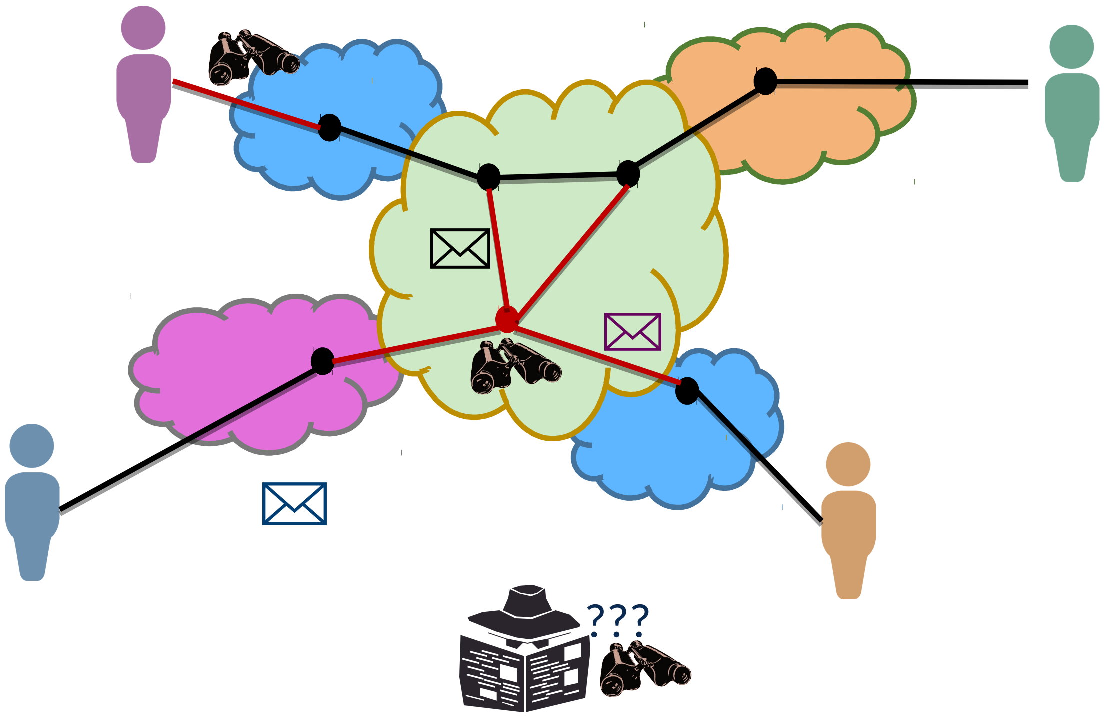


Formalizing Privacy

Indistinguishable cases

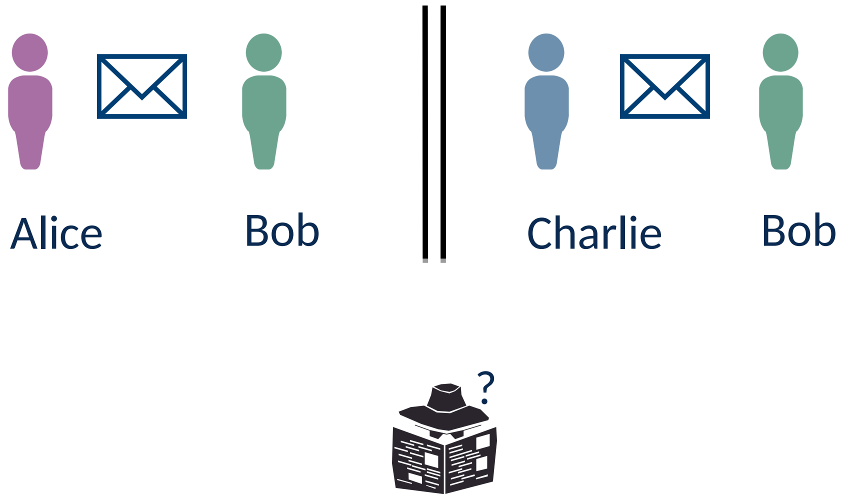


Setting

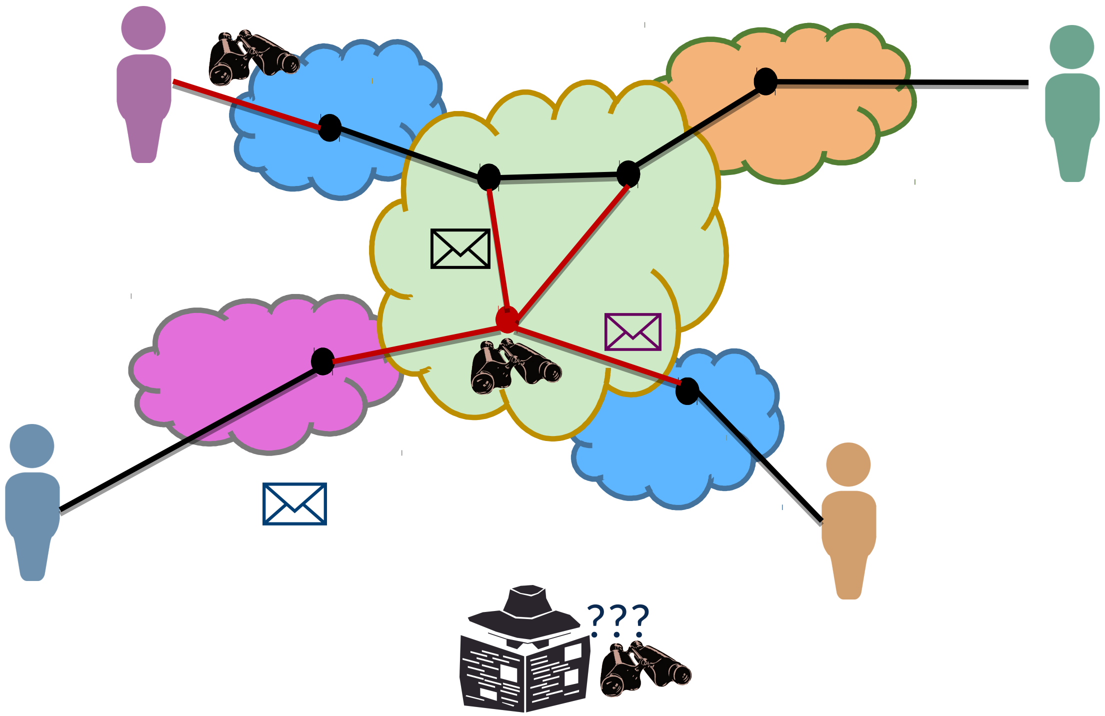


Formalizing Privacy

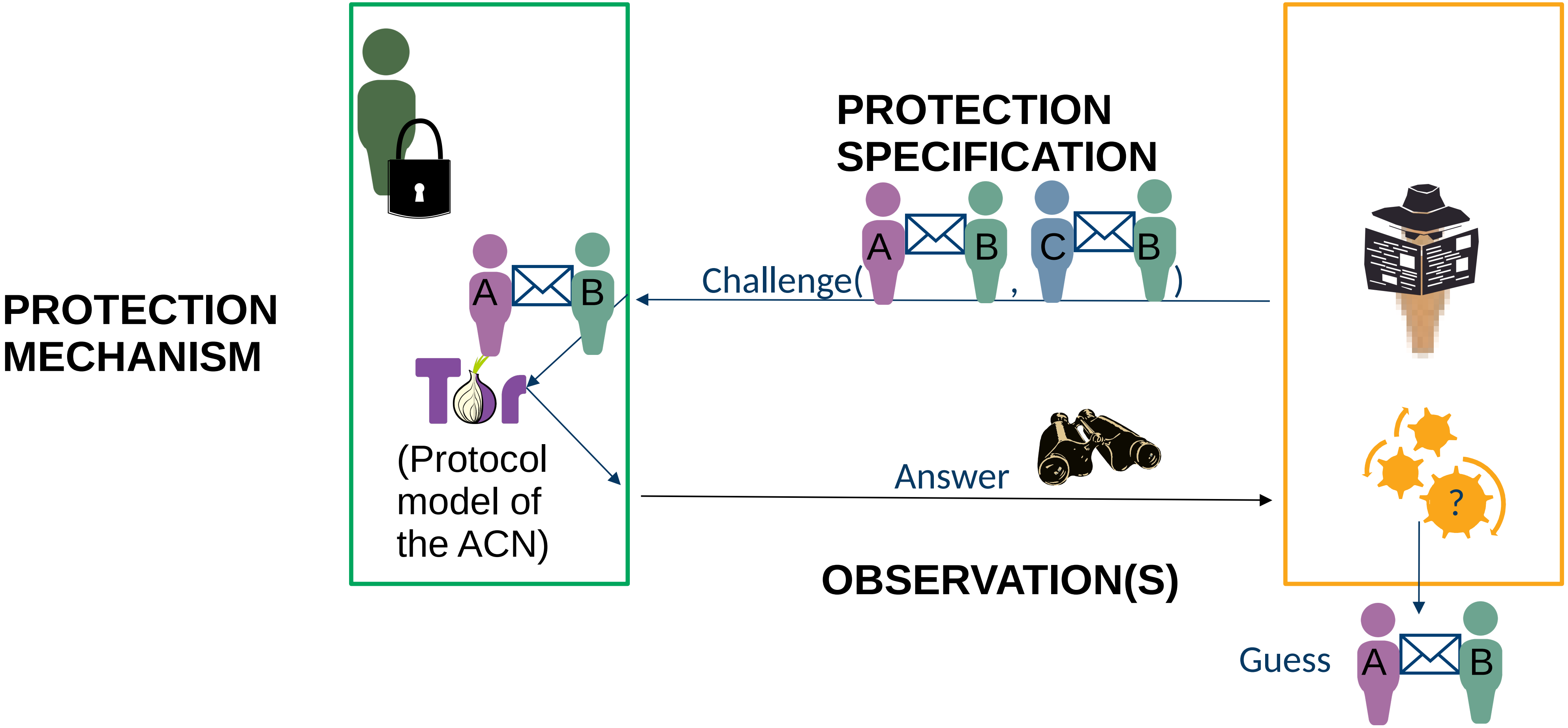
Indistinguishable cases



Setting



Formalizing Privacy for ACNs



Formalizing Privacy for ACNs

Karlsruhe Institute of Technology

Game:

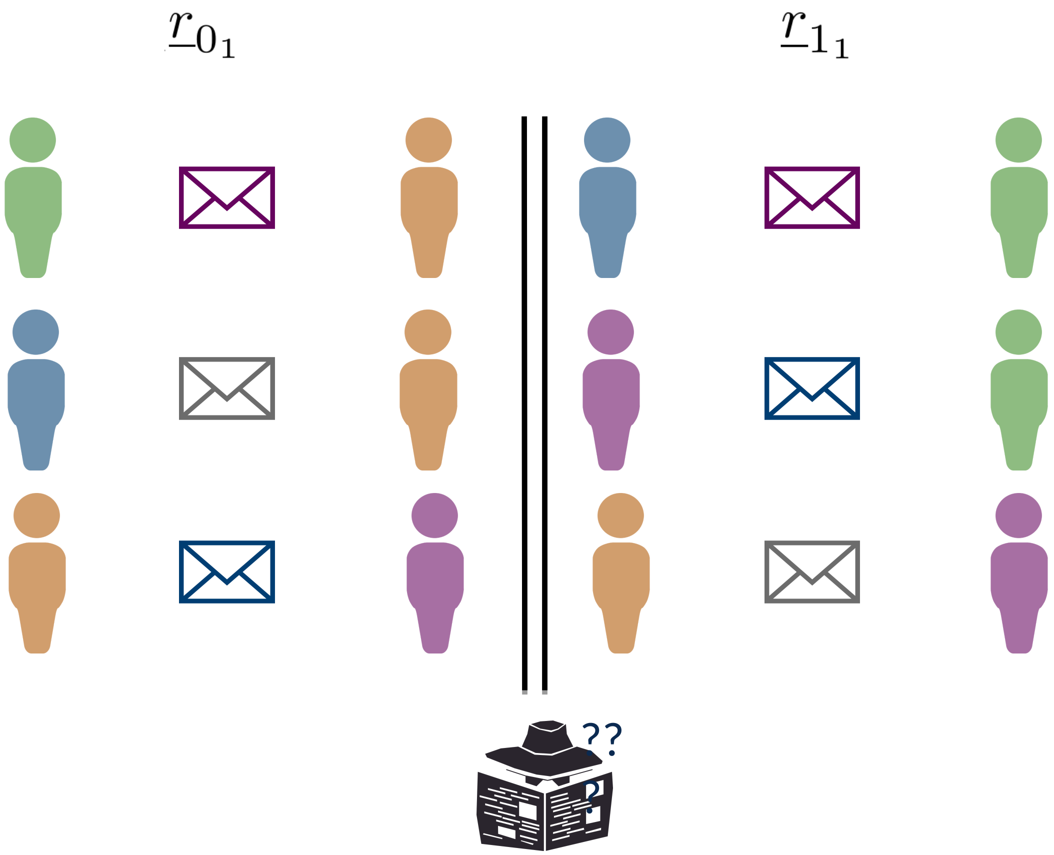
1. Ch randomly picks challenge bit b.
2. $\mathcal{A}$ sends a batch query, containing $\underline{r}_0$ and $\underline{r}_1$, to Ch.
3. Ch checks if the query is valid, i.e. both batches differ only in information that is supposed to be protected according to the analyzed notion X.
4. If the query is valid, Ch inputs the batch corresponding to b to Π.
5. Π's output $\Pi(\underline{r}_b)$ is handed to $\mathcal{A}$.
6. After processing the information, $\mathcal{A}$ outputs her guess g for b.

Achieving the privacy goal:

Protocol model achieves the privacy goal defined in the game iff for all PPT attackers

$$Pr[\mathcal{A}\text{'s } guess\ g = challenge\ bit\ b] - 0.5 \leq negl.$$

Formalizing the Batches



Communication r :

$$(u, u', m, aux)$$

Sender Receiver Message Auxiliary Info

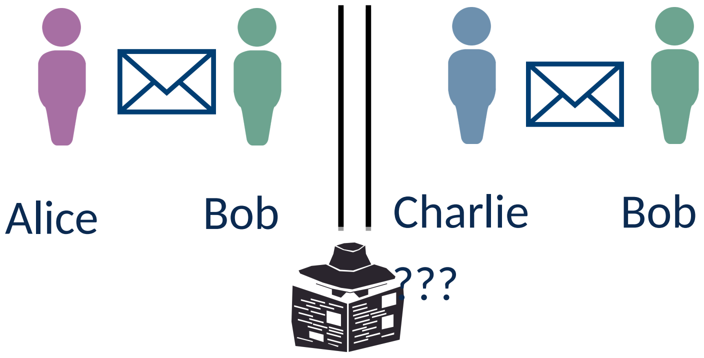
$\Diamond$ No comm.

Batches:

$$\underline{r}_b = (r_{b_1}, \ldots, r_{b_l})$$

1. communication … last communication

Sender Unobservability



For all communications

$$r_{0j} \in \{(u_{0_j}, u'_{0_j}, m_{0_j}, aux_{0_j}), \diamond\}$$

$$r_{1j} \in \{(u_{1_j}, u'_{1_j}, m_{1_j}, aux_{1_j}), \diamond\}$$

of the batches $\underline{r}_0, \underline{r}_1$ holds:

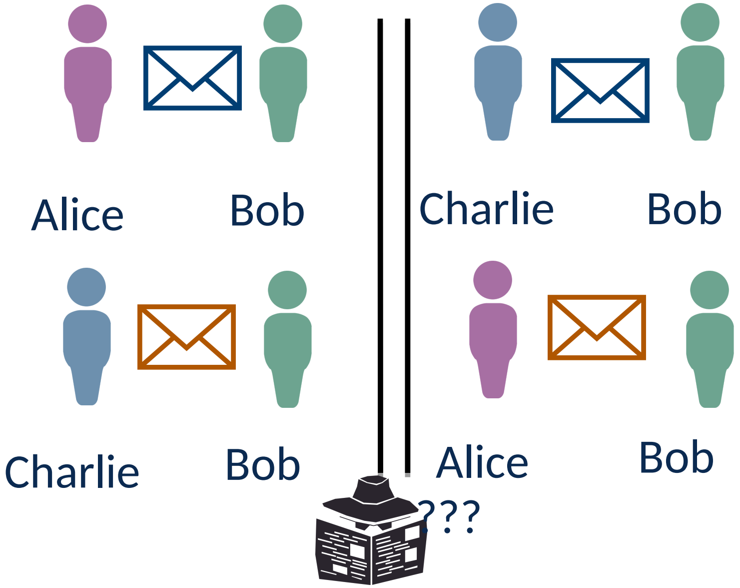
$$E_S : r_{1j} = (\mathbf{u_{1_j}}, u'_{0_j}, m_{0_j}, aux_{0_j})$$

$$\not\Diamond : \Diamond \notin \underline{r}_0 \wedge \Diamond \notin \underline{r}_1$$

Batches differ only, but arbitrarily in their senders.

Resulting anonymity set: All users (active or not).

Sender-Message Unlinkability*



Resulting anonymity set: All users that send to **the same** receiver in the same round.

CR — List of indexes of communications that differ in both batches

For batches defined as above, it holds:

$$
\begin{aligned}
M_{SR}: \quad & r_{0\,cr} = (\mathbf{u_0}, {u'}_{0_{cr}}, \mathbf{m_0}, aux_{0_{cr}}) \wedge \\
& r_{0\,cr+1} = (\mathbf{u_1}, {u'}_{0_{cr}}, \mathbf{m_1}, aux_{0_{cr}}) \wedge \\
& r_{1\,cr} = (\mathbf{u_1}, {u'}_{0_{cr}}, \mathbf{m_0}, aux_{0_{cr}}) \wedge \\
& r_{1\,cr+1} = (\mathbf{u_0}, {u'}_{0_{cr}}, \mathbf{m_1}, aux_{0_{cr}}) \\
& \textit{for every second } cr \in \text{CR}
\end{aligned}
$$

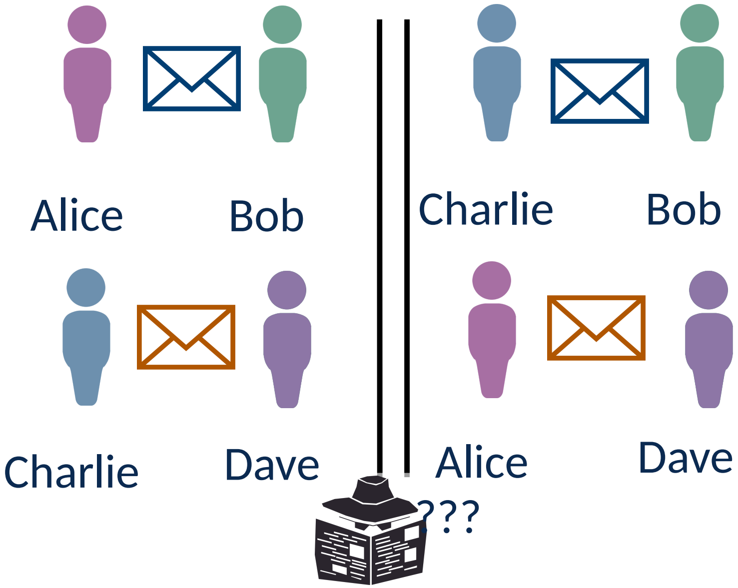
$$\oslash : \Diamond \notin \underline{r}_0 \wedge \Diamond \notin \underline{r}_1$$

Batches differ only in the senders of two consecutive communications with the same receiver. Those senders are switched. Everything else is equal.

*Corresponds to Sender-Message Pair Unlinkability in the paper, which uses additional technicalities to account for the order of communications.

Sender Unlinkability*

Karlsruhe Institute of Technology



$$Q : Q_0 = Q_1$$

$$Q_b := \{(u, n) \mid u\ send\ n\ messages\ in\ \underline{r}_b\}$$

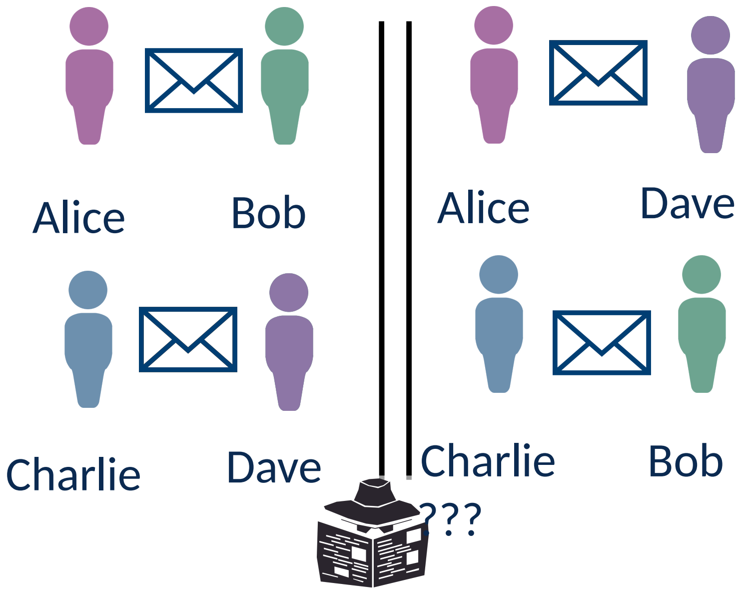
$$\not\Diamond : \Diamond \notin \underline{r}_0 \wedge \Diamond \notin \underline{r}_1$$

Batches differ only in their senders, but each sender’s sending frequency needs to be equal in both batches.

Resulting anonymity set: All users that send **a** message in the same round.

*Corresponds to Sender-Message Unlinkability in the paper (to avoid confusion with Twice-Sender Unlinkability, which is defined in the paper).

Sender-Receiver Unlinkability



As Sender-Message Unlinkability, but we allow receivers to differ and use the same messages.

*Corresponds to Sender-Receiver Pair Unlinkability in the paper, which uses additional technicalities to account for the order of communications.

What’s protected?

Karlsruhe Institute of Technology

- **Sender Unobservability**: we do not learn who sends something
- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)
- **Sender-Receiver Unlinkability:** we do not learn who sends to whom (same message)
- **Sender Unlinkability:** we do not learn who sends which message or to whom

More protection goals possible

What’s protected?

- **Sender Unobservability**: we do not learn who sends something
- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)
- **Sender-Receiver Unlinkability:** we do not learn who sends to whom (same message)
- **Sender Unlinkability:** we do not learn who sends which message or to whom

More protection goals possible

Is Sender-Message Unlinkability stronger than Sender Unobservability?

What’s protected?

Karlsruhe Institute of Technology

- **Sender Unobservability**: we do not learn who sends something
- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)

No! Counterexample: A protocol leaking (only!) sending activities per batch

(the same users are active in any batches accepted for Sender-Message Unlinkability)

Is Sender-Message Unlinkability stronger than Sender Unobservability?

What's protected?

- **Sender Unobservability**: we do not learn who sends something
- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)
- **Sender-Receiver Unlinkability:** we do not learn who sends to whom (same message)
- **Sender Unlinkability:** we do not learn who sends which message or to whom

More protection goals possible

Is Sender Unobservability stronger than Sender Unlinkability?

What's protected?

- **Sender Unobservability**: we do not learn who sends something
- **Sender Unlinkability:** we do not learn who sends which message or to whom

Yes! Any protocol that achieves Sender Unobservability also achieves Sender Unlinkability.

Indirect Proof: Assume protocol p does not achieve Sender Unlinkability, i.e. there is a successful valid attack on Sender Unlinkability, but achieves Sender Unobservability. The attack against Sender Unlinkability is also valid against Sender Unobservability. Contradiction with our assumption that Sender Unobservability is achieved.

Is Sender Unobservability stronger than Sender Unlinkability?

What’s protected?

Karlsruhe Institute of Technology

- **Sender Unobservability**: we do not learn who sends something
- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)
- **Sender-Receiver Unlinkability:** we do not learn who sends to whom (same message)
- **Sender Unlinkability:** we do not learn who sends which message or to whom

More protection goals possible

Is Sender-Receiver Unlinkability
stronger than Sender-Message Unlinkability?

KASTEL

What’s protected?

Karlsruhe Institute of Technology

- **Sender-Message Unlinkability:** we do not learn who send which message (same receiver)
- **Sender-Receiver Unlinkability:** we do not learn who sends to whom (same message)

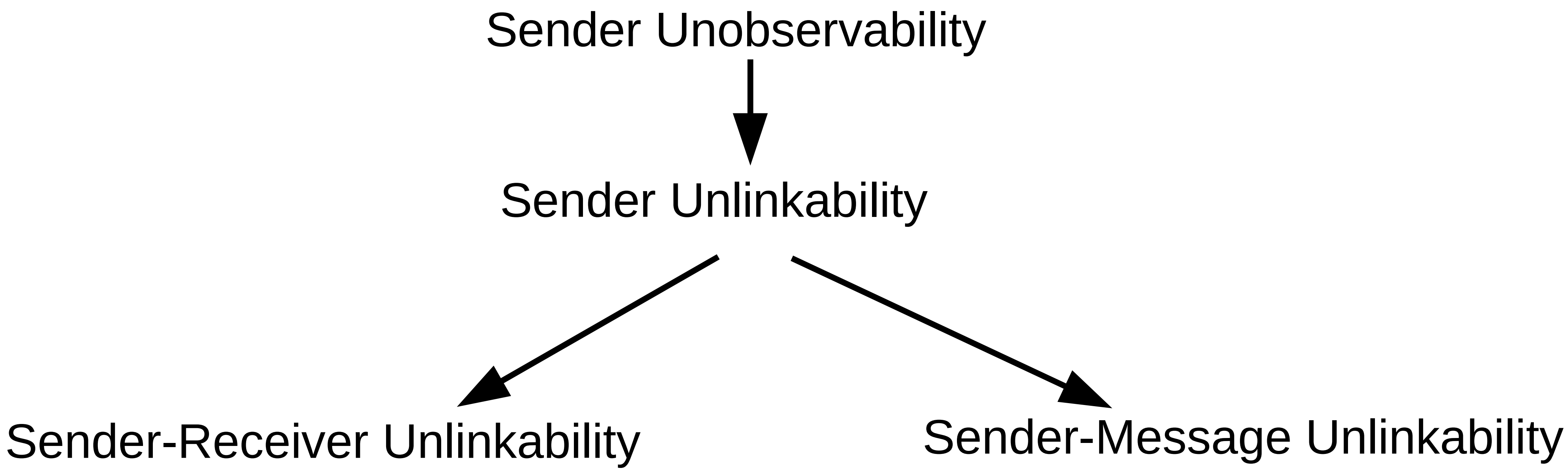
No! Counterexample: A protocol leaking (only!) the sender-message relationship. (Sender-Receiver Unlinkability always guarantees that the same sender-message-pairs are used in both batches)

Is Sender-Receiver Unlinkability stronger than Sender-Message Unlinkability?

KASTEL

What’s protected?

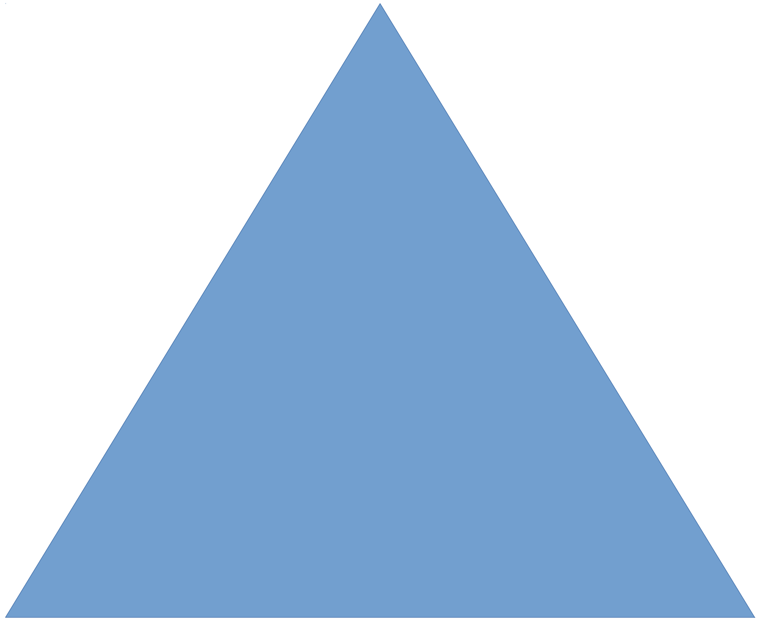
Karlsruhe Institute of Technology



KASTEL

Criteria

What’s protected?



Against what adversary?

At what cost?

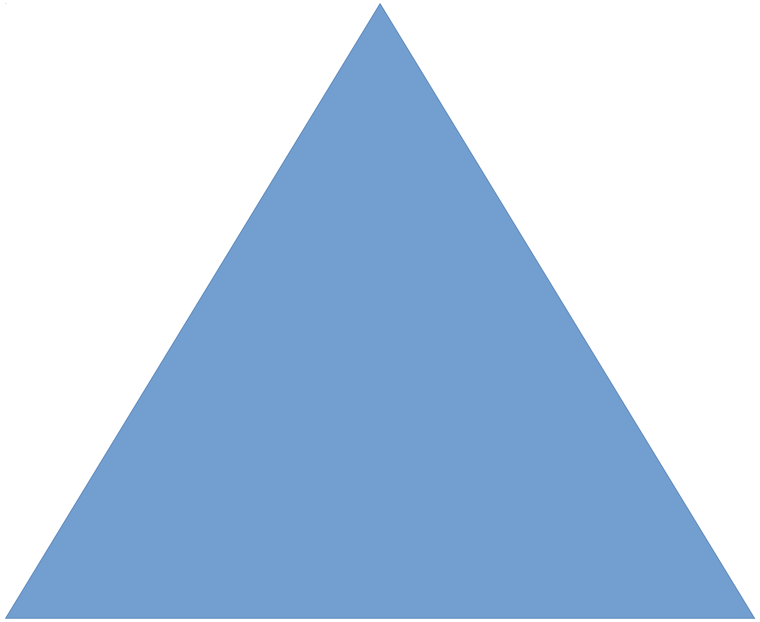
Against what adversary?

- Area? Local vs. Global, Links vs. Nodes etc.
- Actions? Eavesdropping (Passive)/ Modification, Dropping, Delay (Active)

  → we'll focus on passive adversaries for this lecture
- Participant? Internal vs. External
- Time? Temporary vs. Permanent
- Change resources/strategy? Static vs. Adaptive
- Restricted computation power?

Criteria

Karlsruhe Institute of Technology



KASTEL

At what cost?

- Latency
- Bandwidth

- Functionality
- Other security goals (availability)
- Additional assumptions (public key infrastructure etc.)

Learning Goals

- Understand the Problem
  - Motivation and Setting
  - Dimensions and Terminology

- Understand the Solution(-space)
  - Solution ideas and prominent protocols:
    - Random Walk
    - Onion Routing
    - Mix Networks
    - Dummy Traffic
    - DC Networks
  - Effects of design decisions

Learning Goals

Karlsruhe Institute of Technology

- Understand the Problem
  - Motivation and Setting
  - Dimensions and Terminology

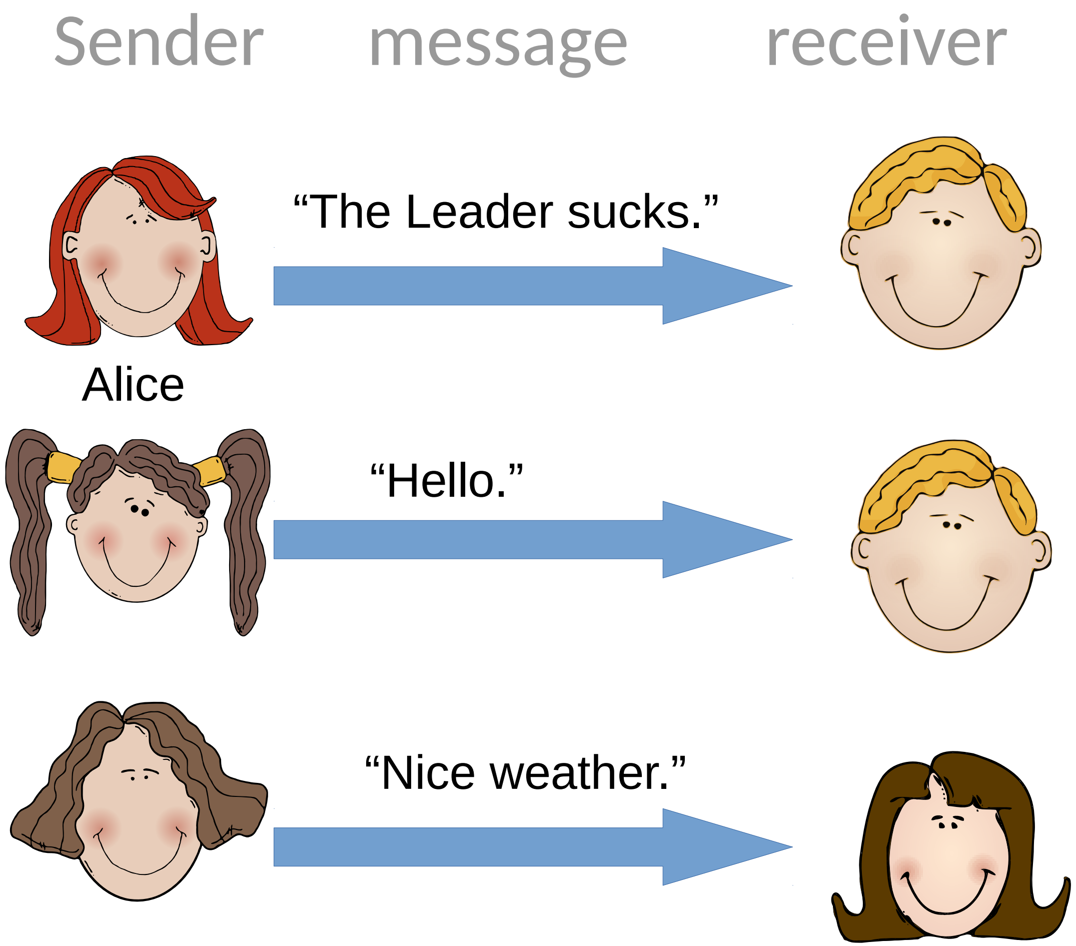
- Understand the Solution(-space)
  - Solution ideas and prominent protocols:
    - Random Walk
    - Onion Routing
    - Mix Networks
    - Dummy Traffic
    - DC Networks
  - Effects of design decisions

Solution ideas on slides with white background

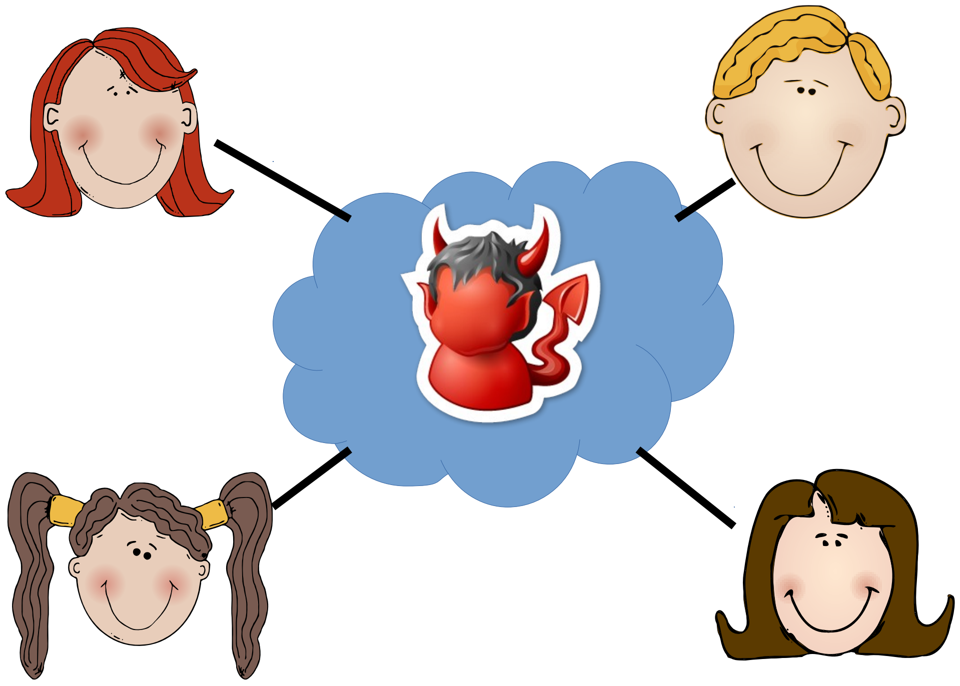
Prominent protocols on slides with green background

Setting

The Communications that happen

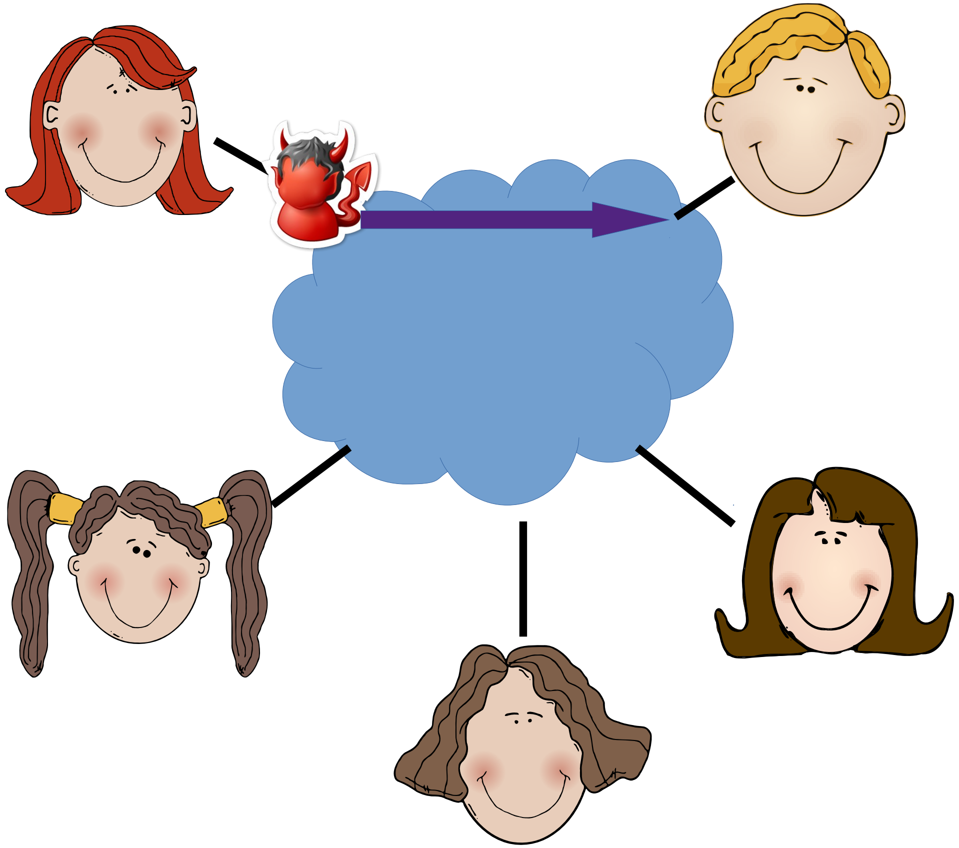


The network on which they happen



Without any protection

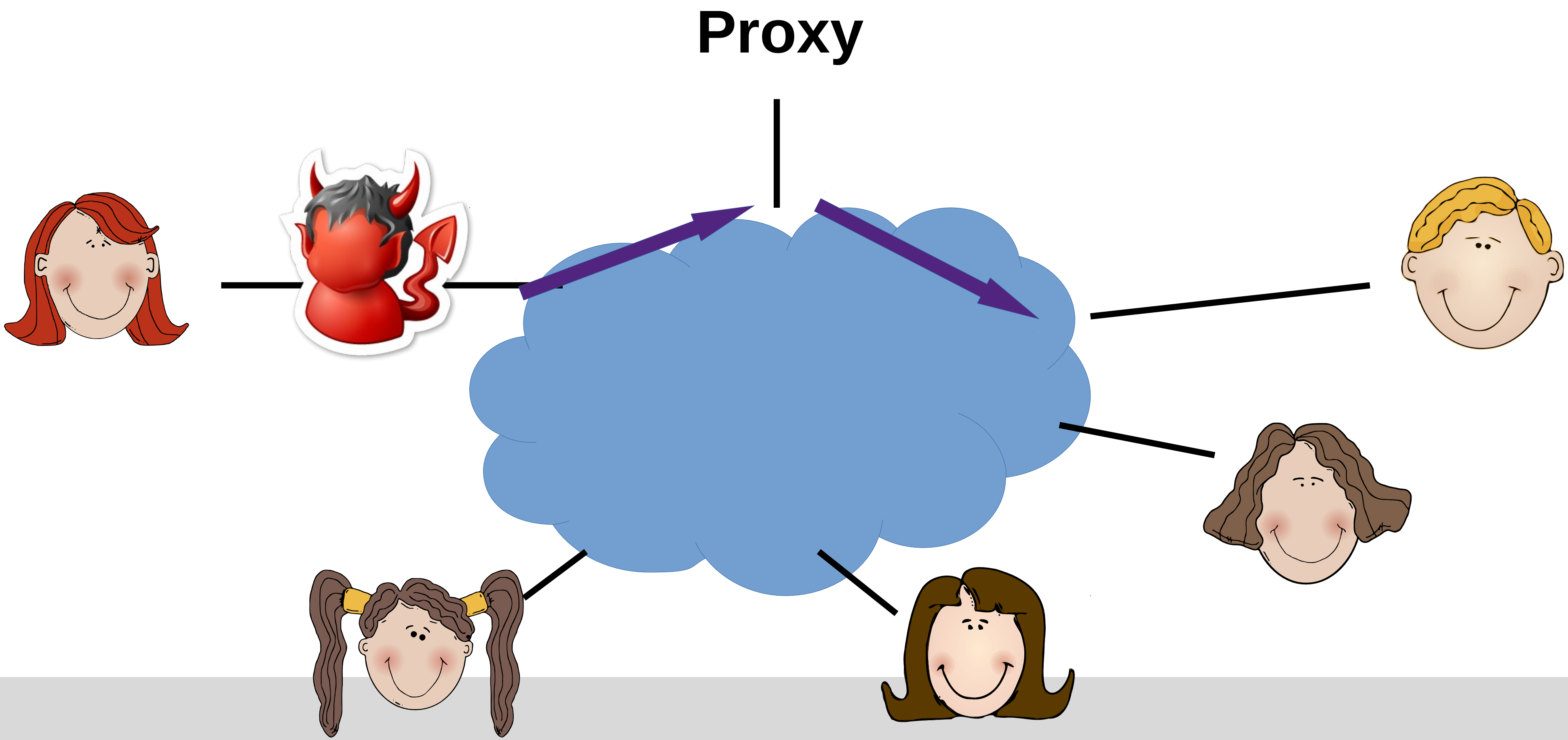
- Direct connection observable



Using a Proxy

Principle 1: Indirection

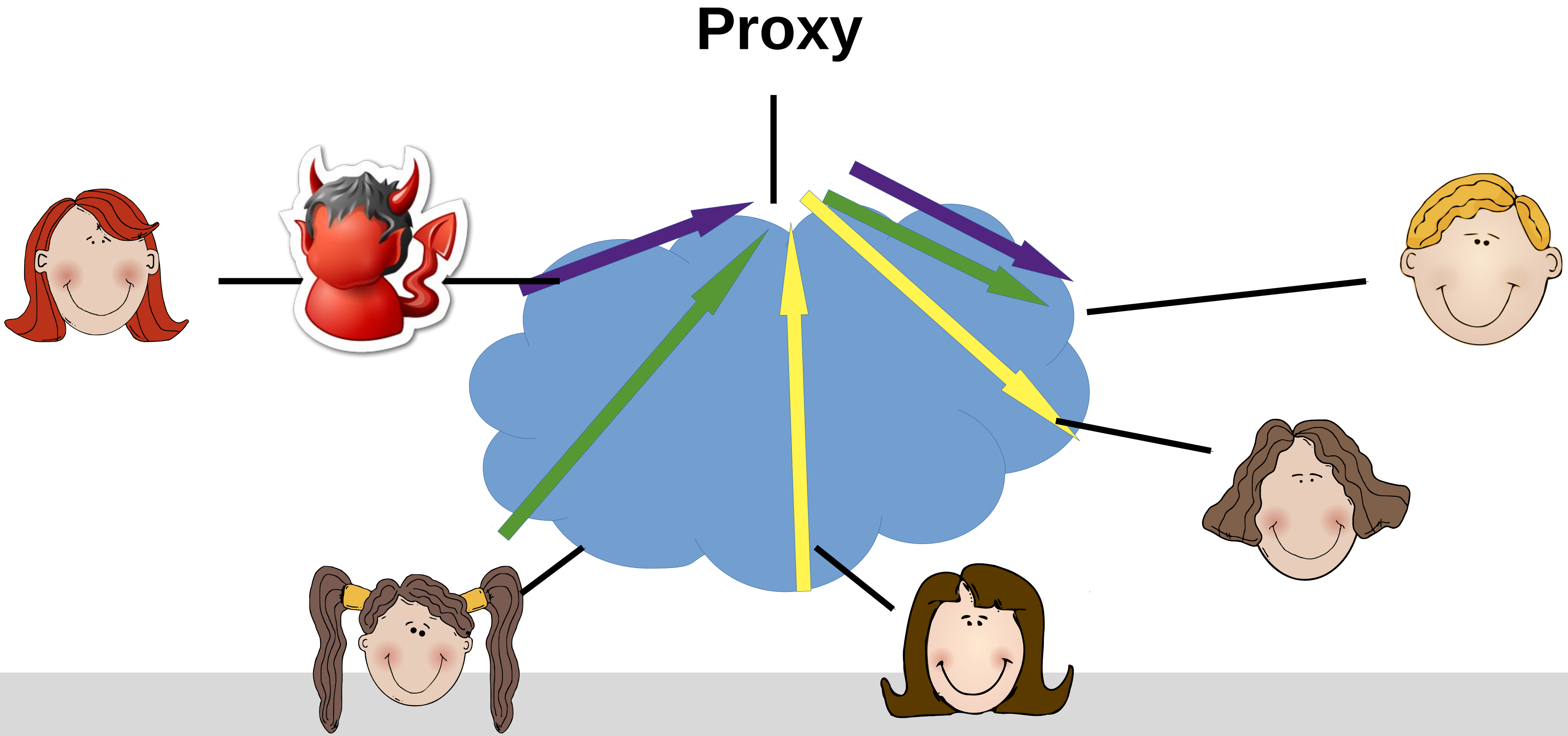
Alice sends message and receiver address to a proxy, who then forwards the message to the receiver



Using a Proxy

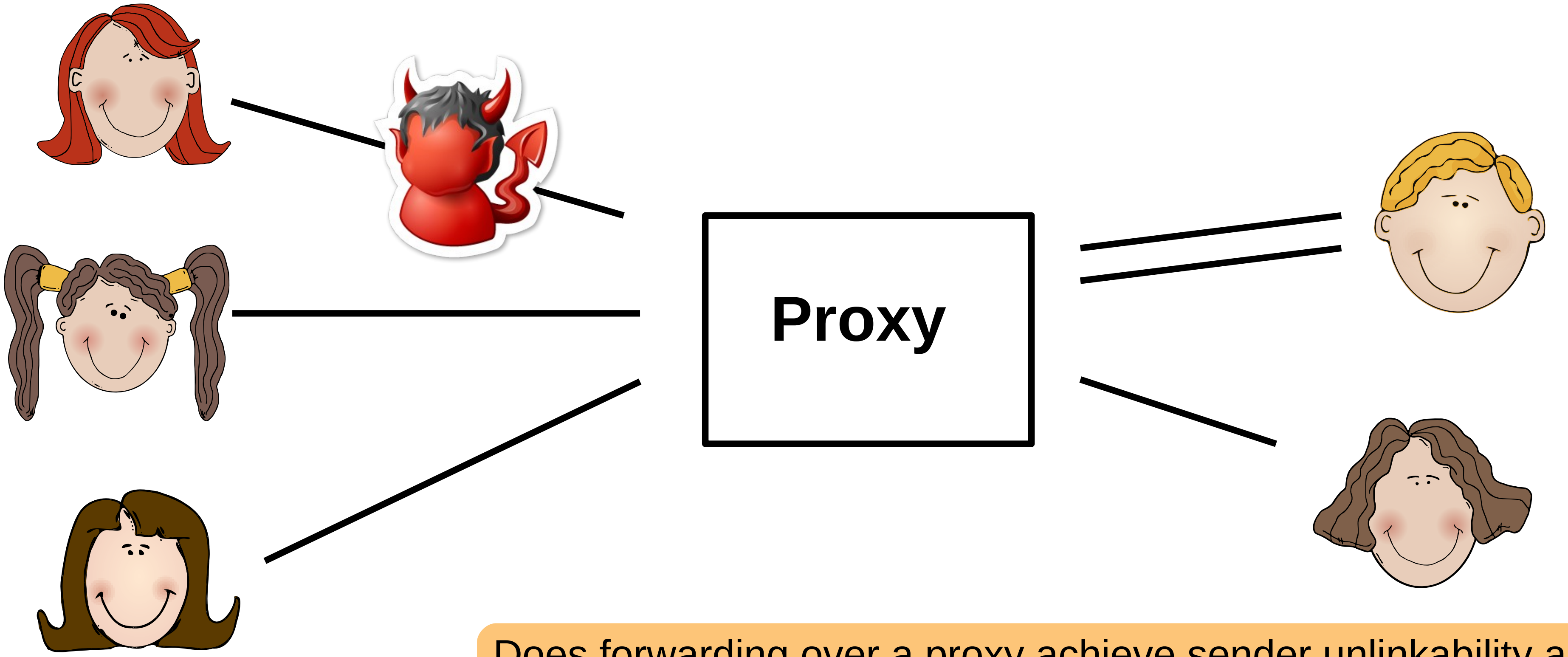
Principle 1: Indirection

Alice sends message and receiver address to a proxy, who then forwards the message to the receiver, all other senders do the same



Using a Proxy

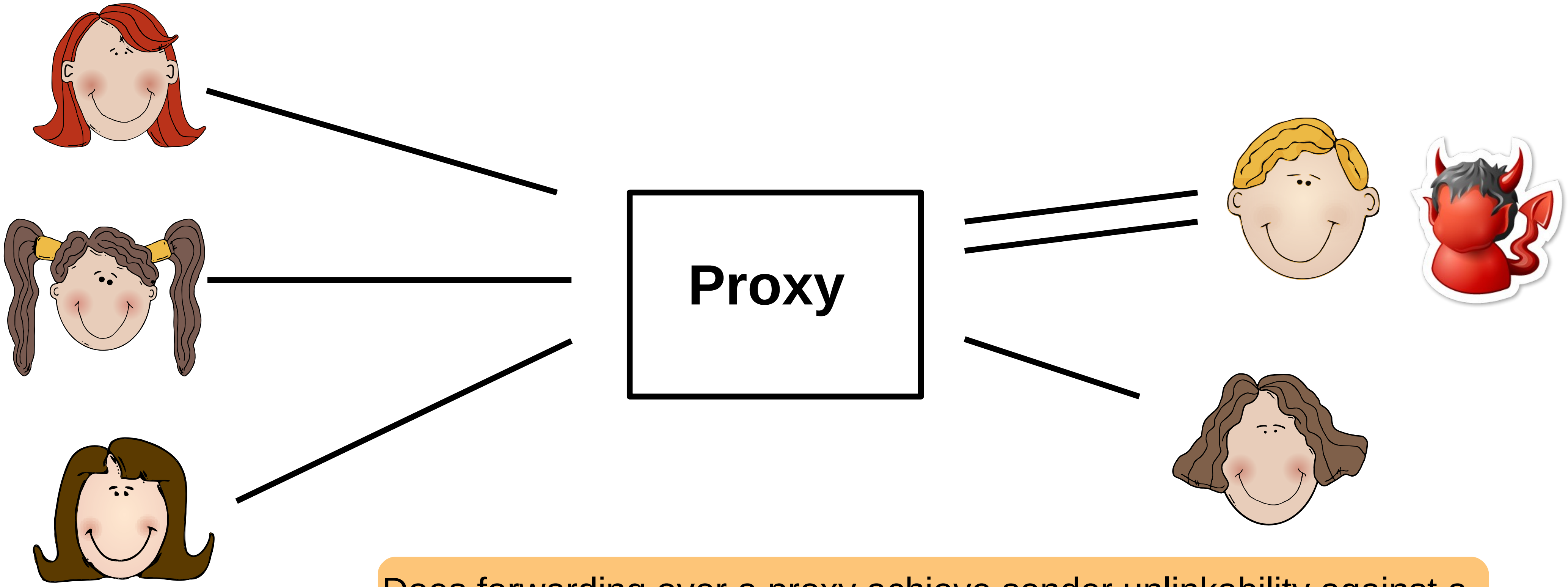
Principle 1: Indirection



Does forwarding over a proxy achieve sender unlinkability against a passive, local adversary at the senders?

Using a Proxy

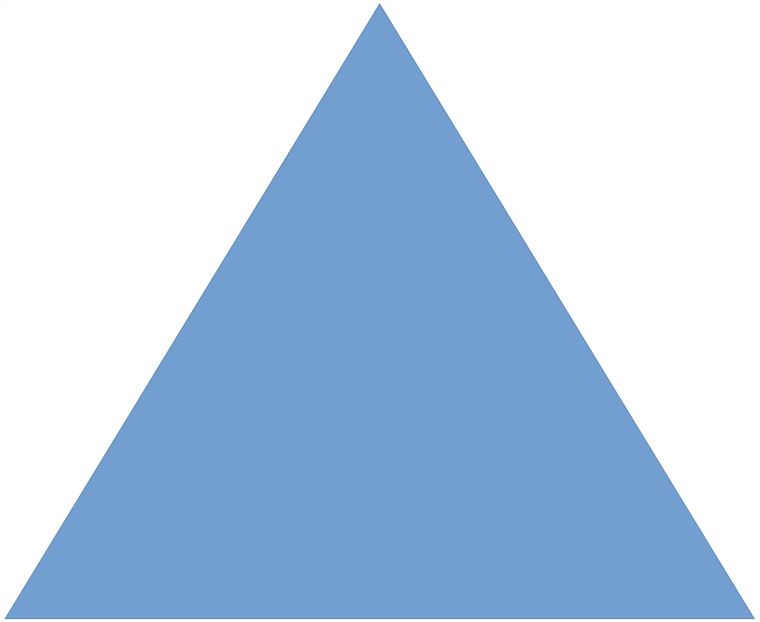
Principle 1: Indirection



Does forwarding over a proxy achieve sender unlinkability against a corrupt, passive receiver?

Using a Proxy

Sender Unobservability



Passive receiver as adversary

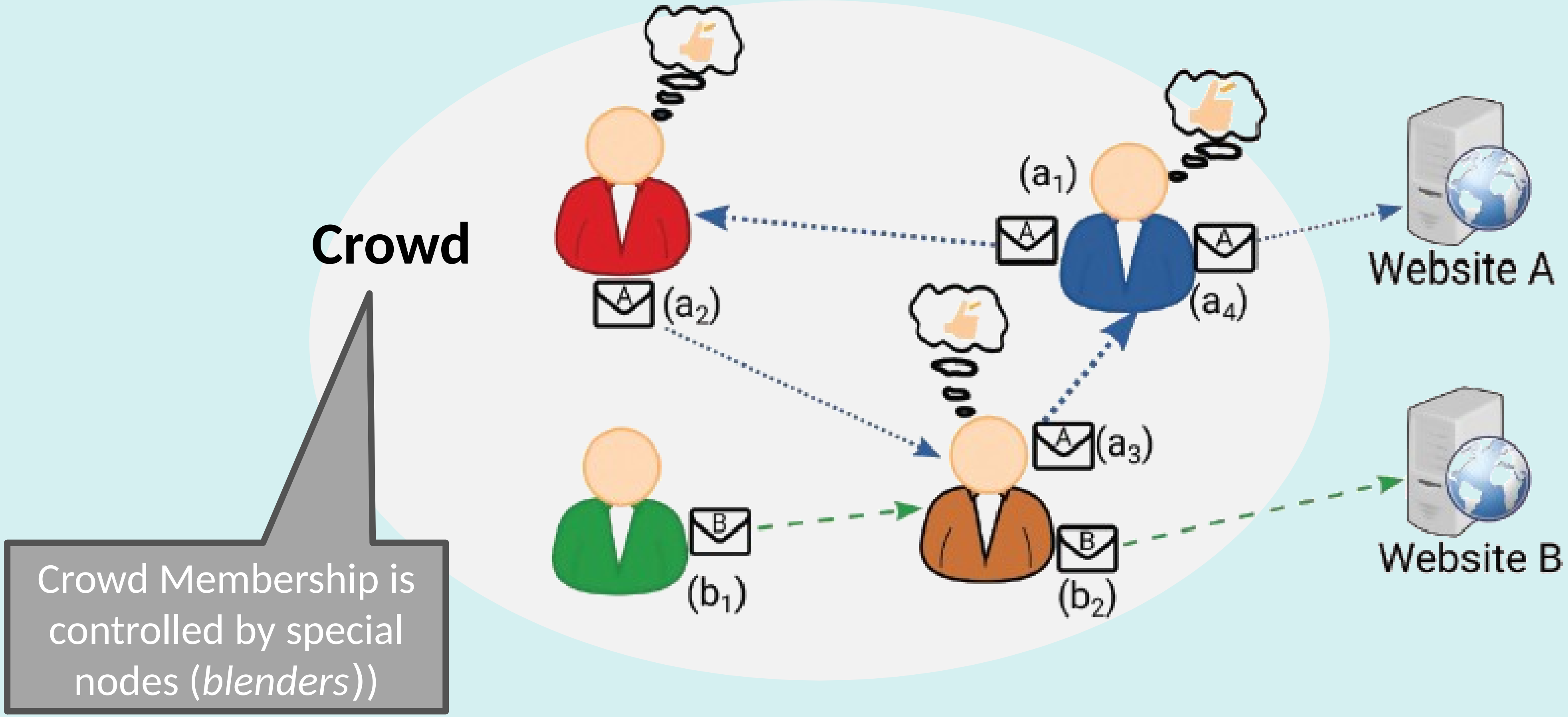
Slightly higher latency
need a proxy

Random Walk Protocols

- Typically use peer-to-peer network structure
- Forward message to randomly selected neighbor
- *Example: Crowds* (1998) for anonymous web browsing

Reiter, Michael K., and Aviel D. Rubin. "Crowds: Anonymity for web transactions." ACM transactions on information and system security (TISSEC) 1.1 (1998): 66-92.

Random Walk concept (Crowds)



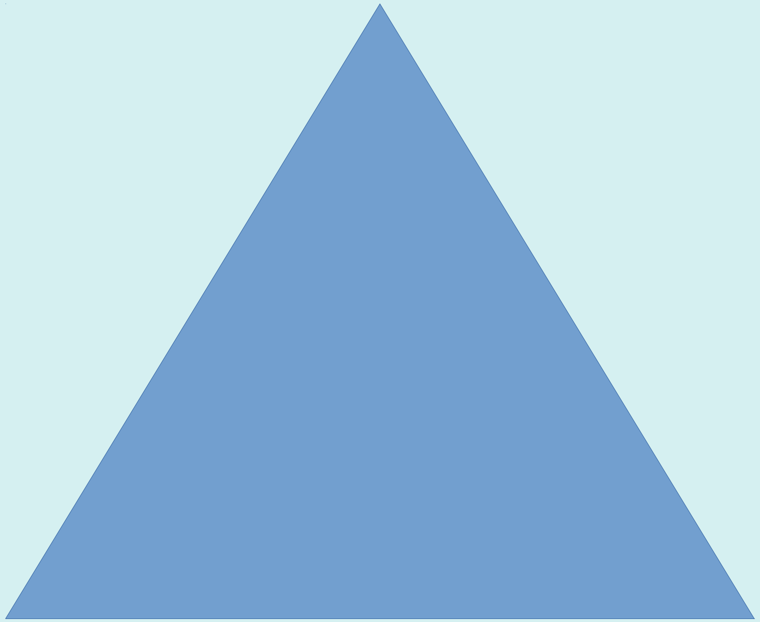
Crowds

- All nodes are grouped into „crowds“
- Nodes within a crowd might connect to each other for relaying a communication:
  - user randomly selects a node and sends her message (i.e., website request)
  - this node flips a biased coin to decide whether to send the request directly to the receiver or to forward it to another node selected uniform at random,
  - this continues until the message arrives at the destination.
  - The server replies are relayed through the same nodes in reverse order.

Can an internal adversary, corrupting n-2 participants, identify the sender of a message (with high probability)?

Crowds (with sufficiently biased coin)

Sender Unobservability



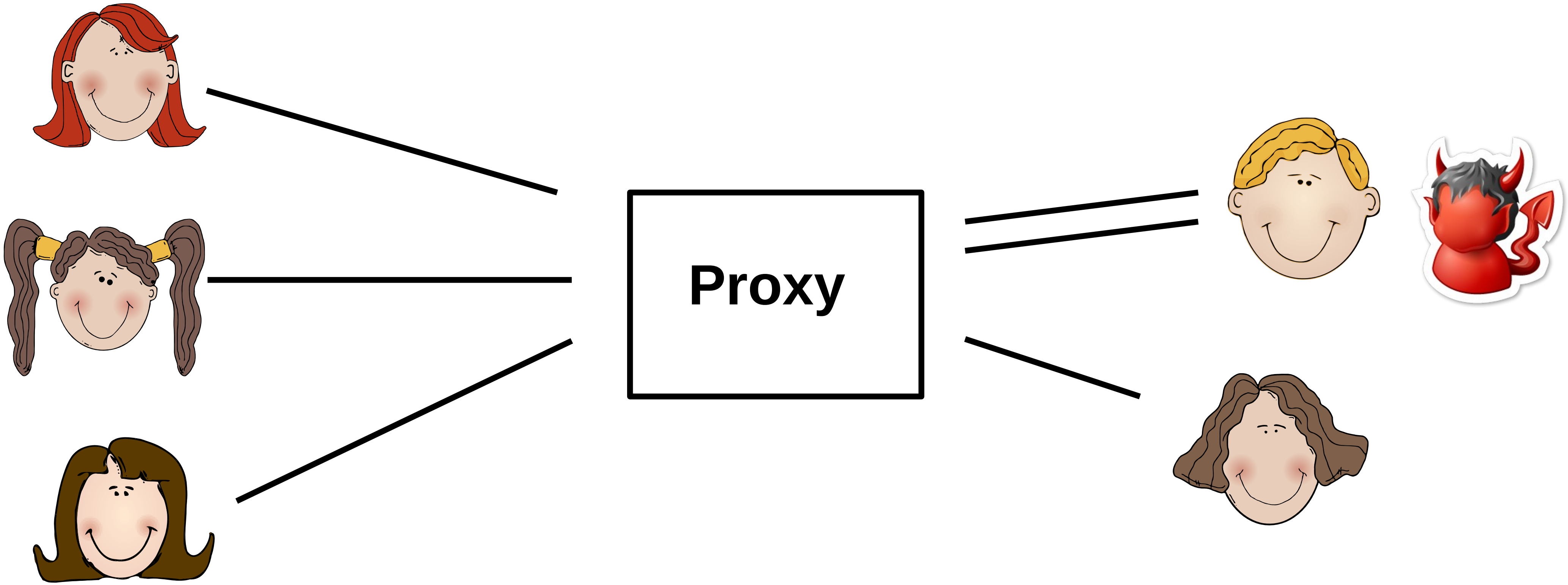
Passive **external** receiver

Higher latency
Management overhead
Availability risk (blenders)

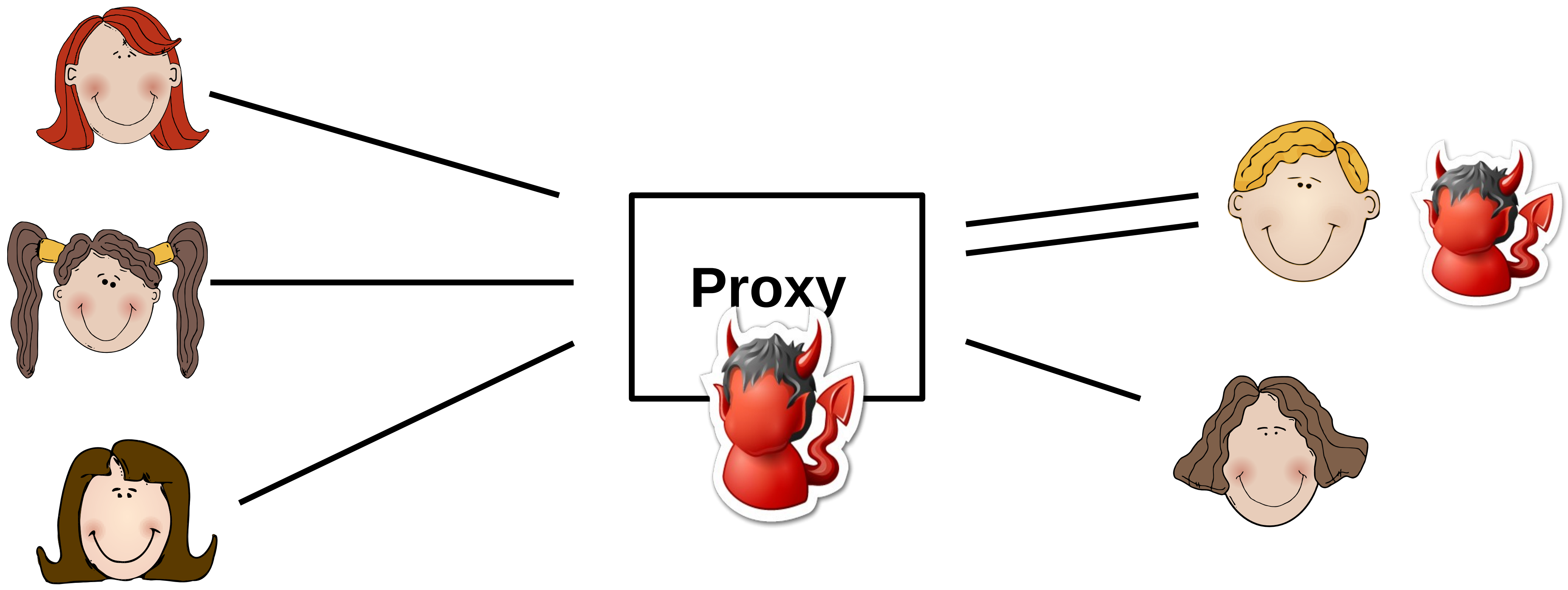
Summary Random walk

- Non-deterministic route selection
- Protection against external adversary
- Internal adversary improves estimation of sender based on timing information (predecessor attack)
  - Crowds is a representative example
    - Semi de-centralized
    - ✉ blenders are single points of failure

Using a Proxy



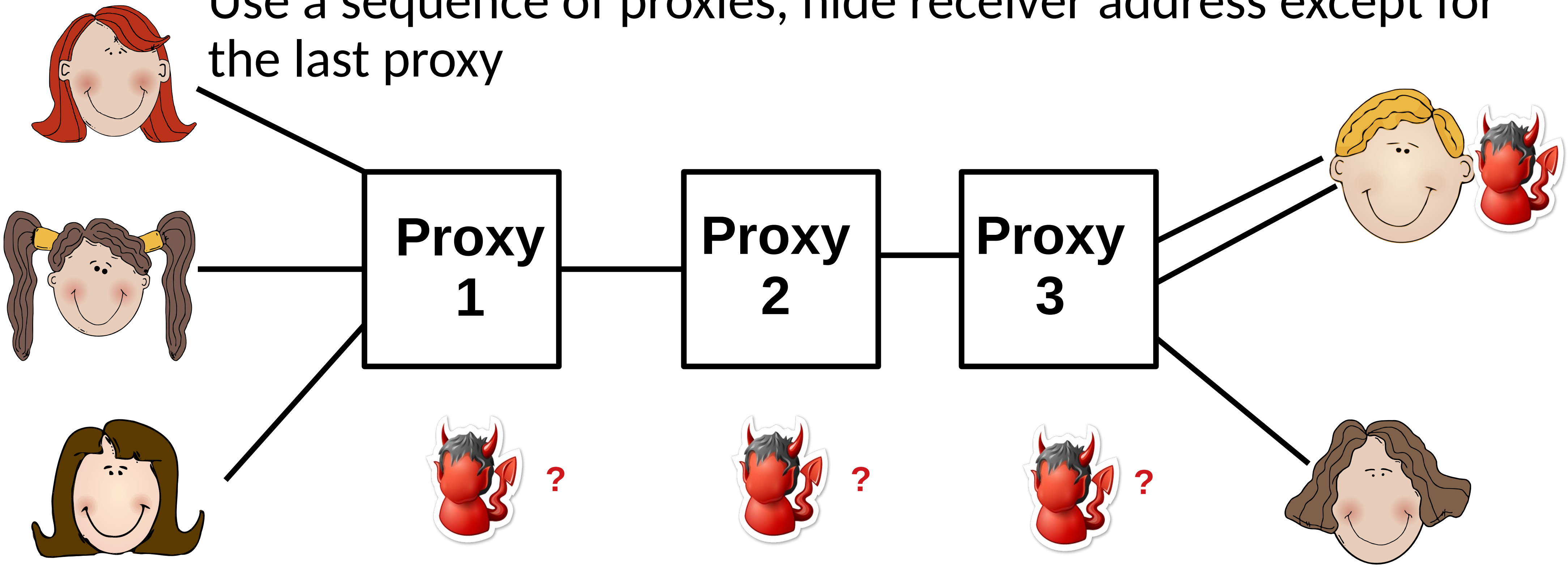
Using a Proxy



Using a Proxy Chain

Principle 2: Distribution of Trust

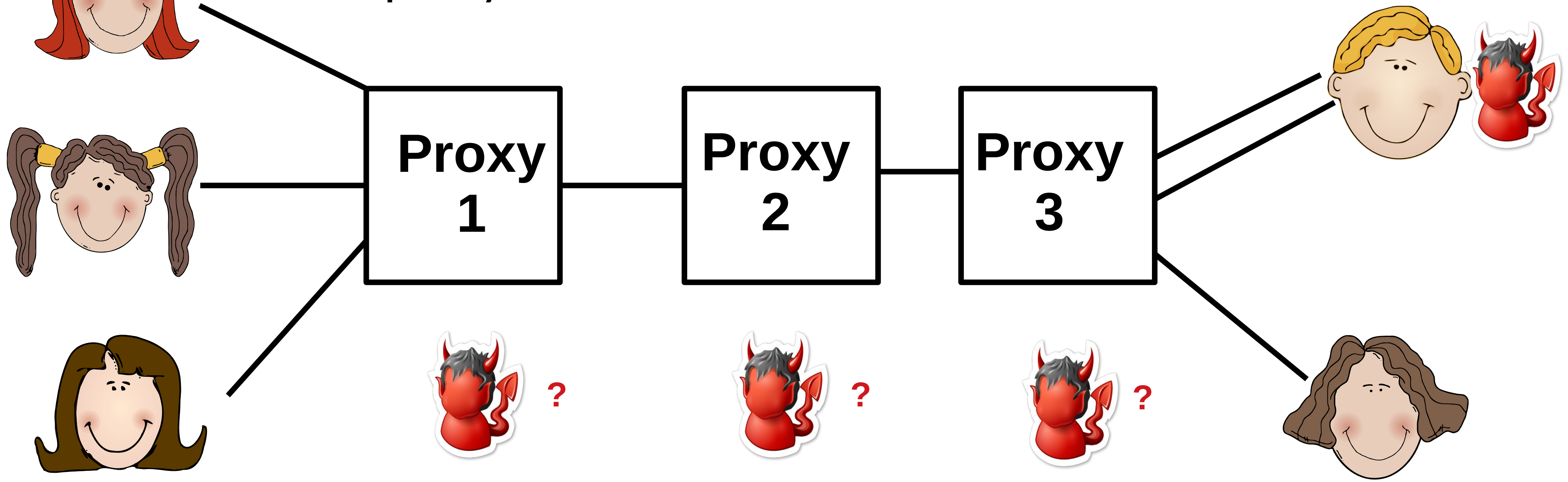
Use a sequence of proxies, hide receiver address except for the last proxy



Using a Proxy Chain

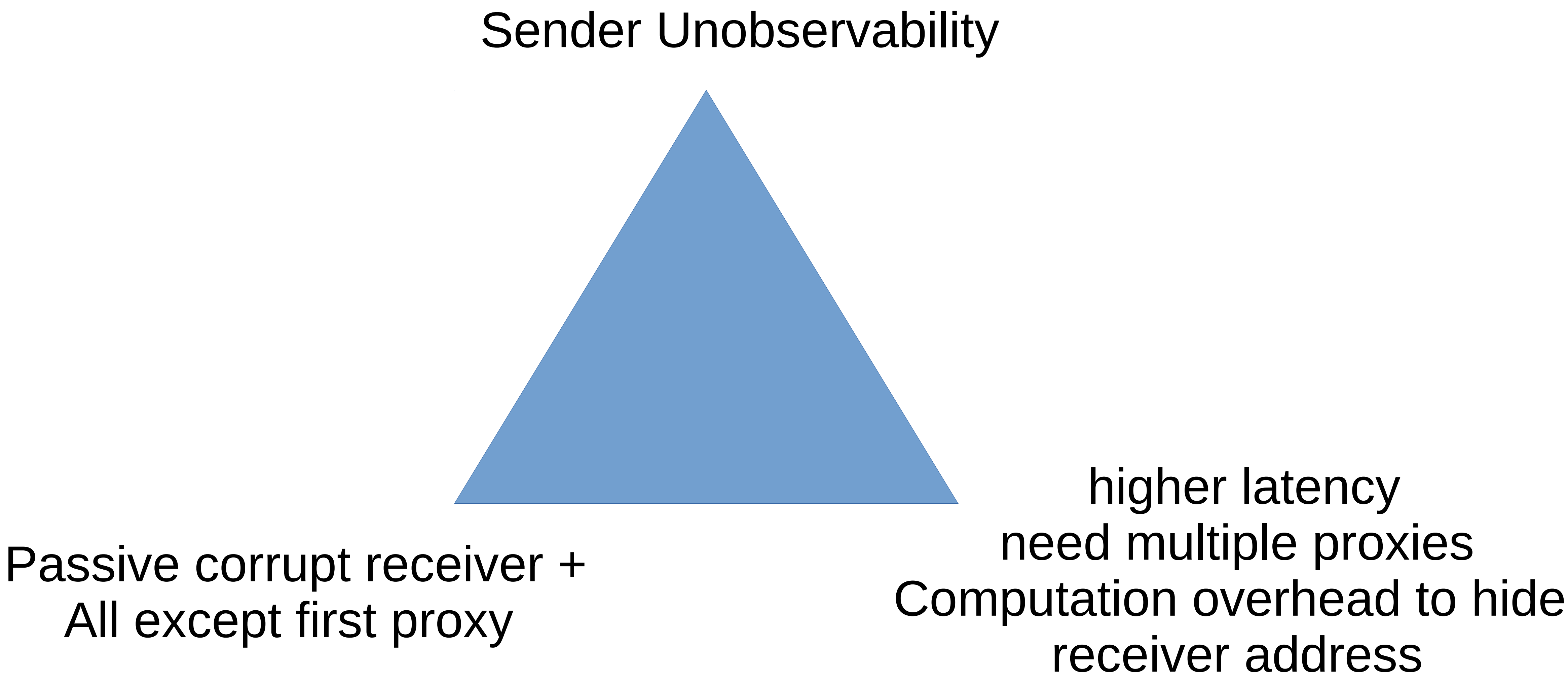
Principle 2: Distribution of Trust

Use a sequence of proxies, hide receiver address except for the last proxy

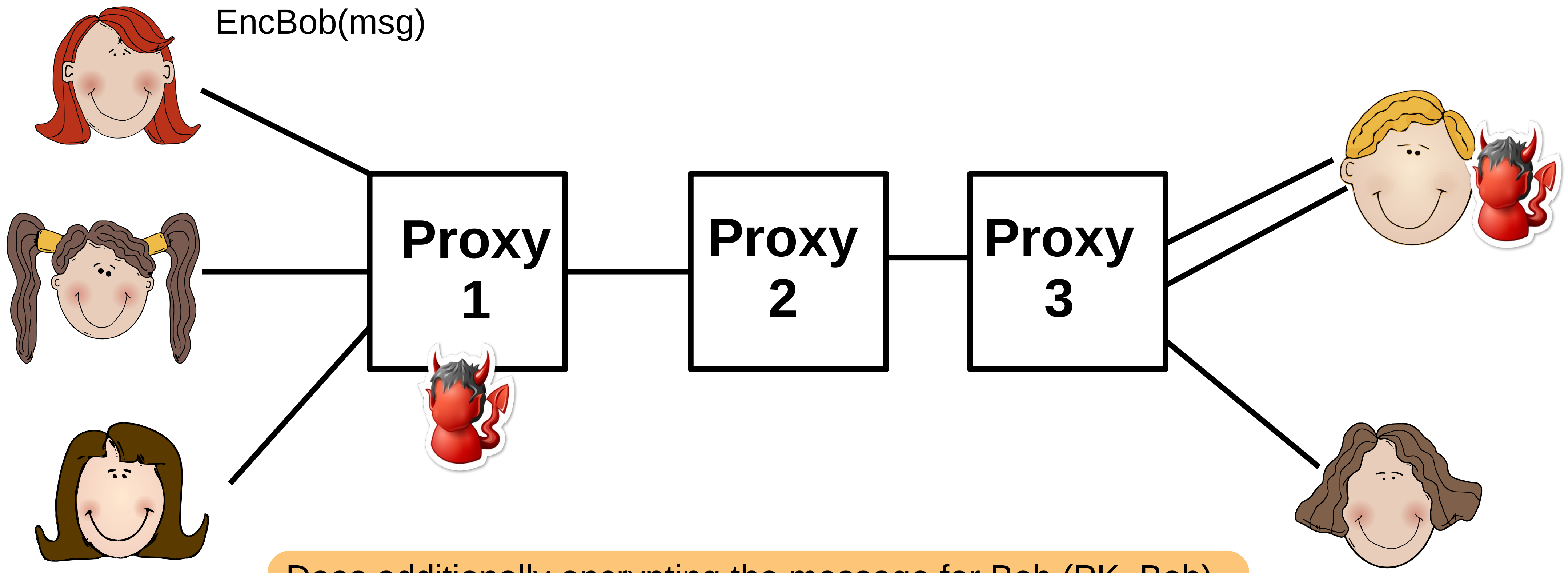


How many proxies need to be **corrupt** to break sender unlinkability against a corrupt receiver?

Using a Proxy Chain



Adding end-to-end encryption

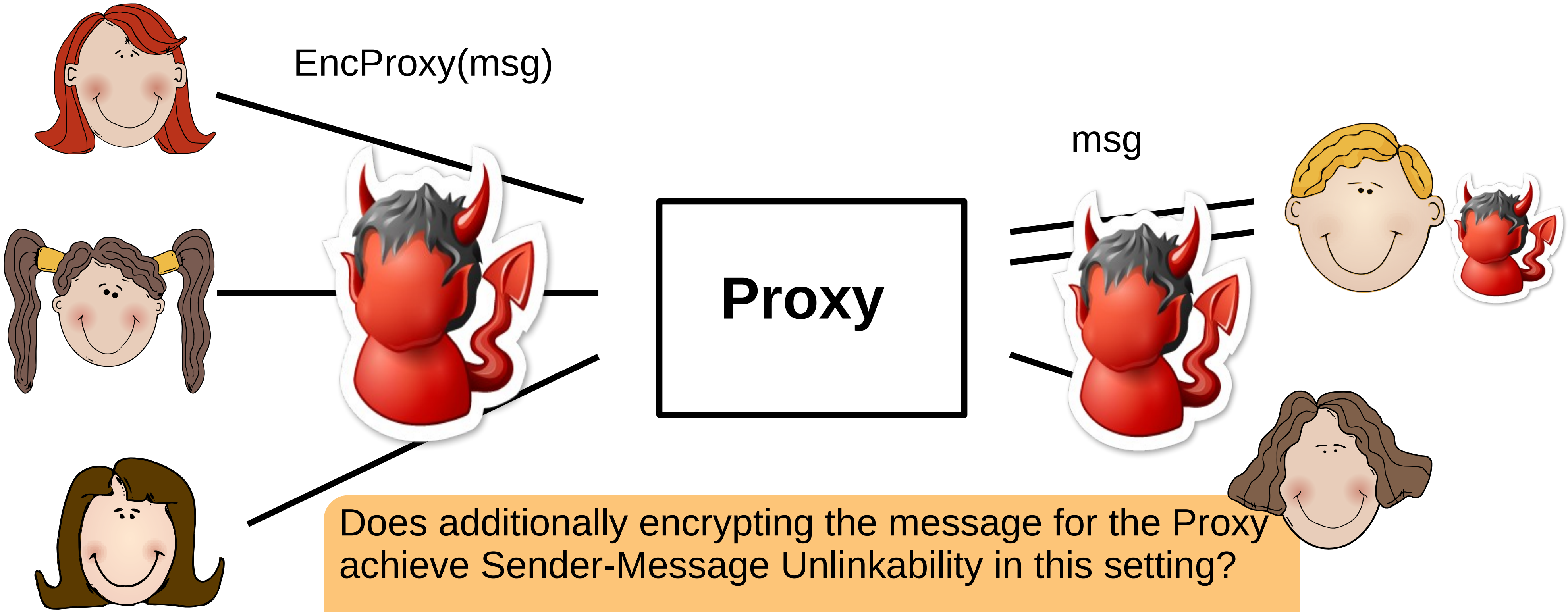


Does additionally encrypting the message for Bob (PK_Bob) achieve Sender-**Message** Unlinkability in this setting?

Adding Encryption

Principle 3: Unlink Observations

Principle 4: Randomize Observations

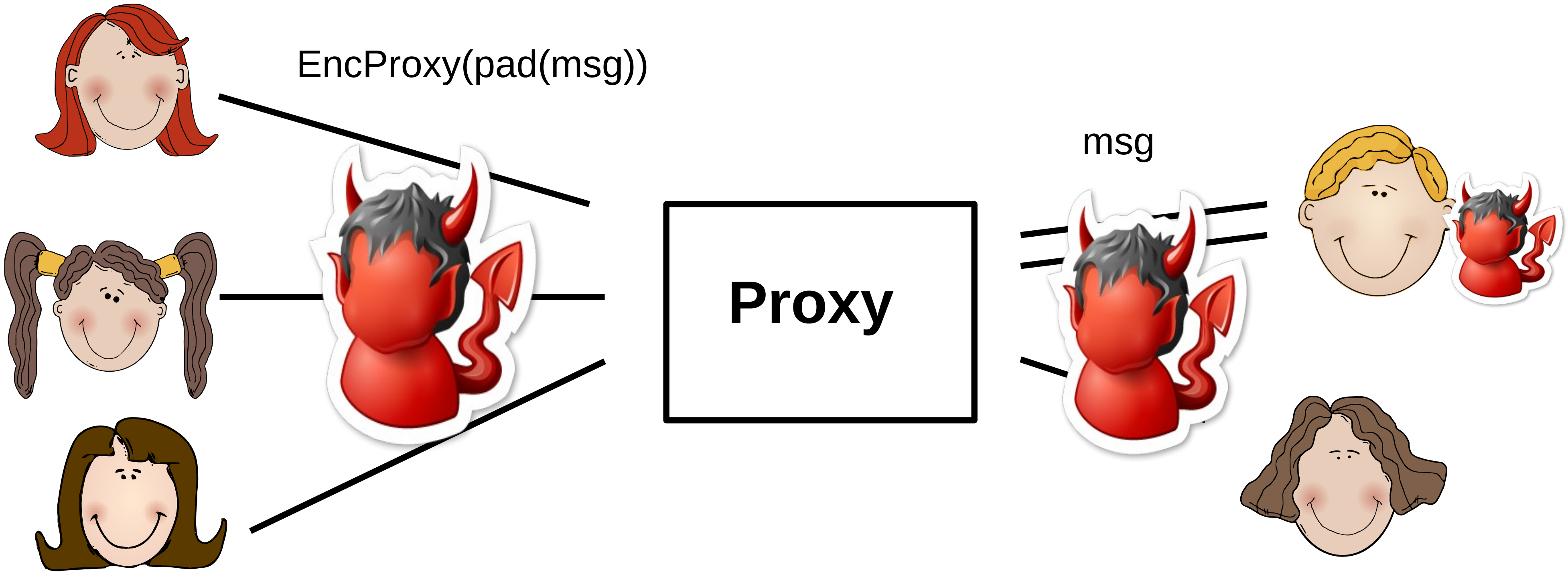


Does additionally encrypting the message for the Proxy achieve Sender-Message Unlinkability in this setting?

Padding against linking based on length

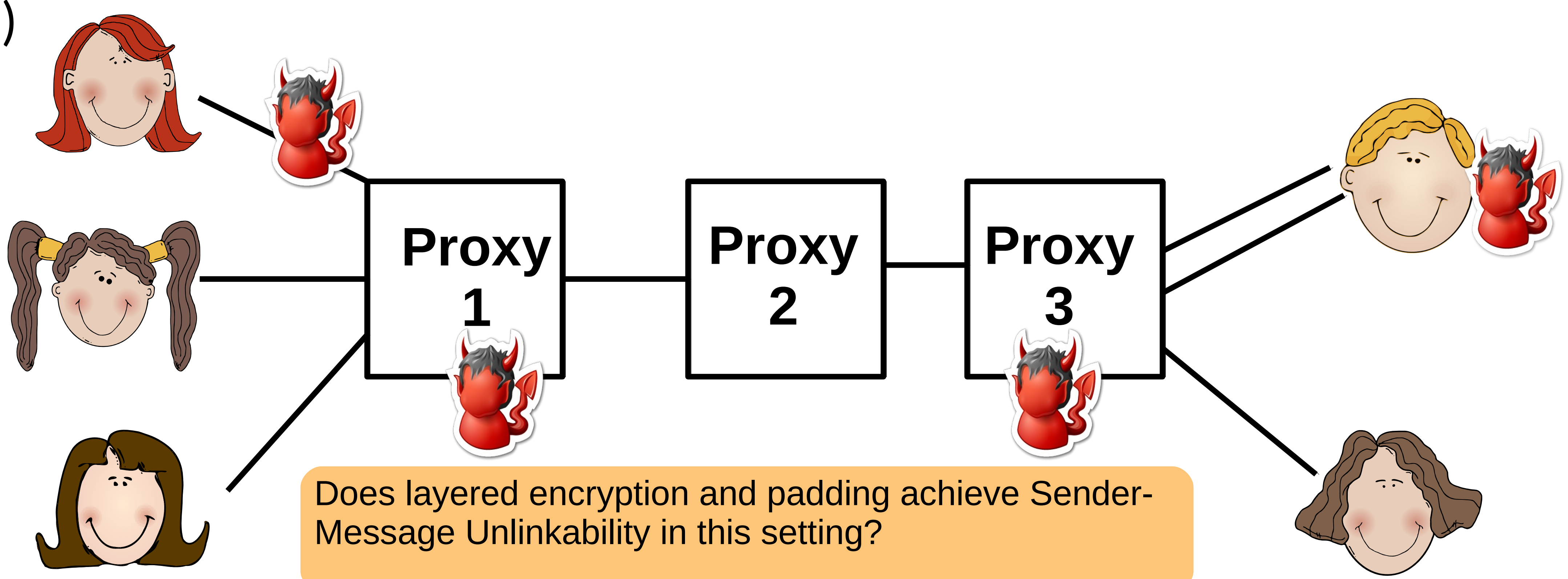
Principle 5: Fix Observations (& Principle 3)

Padding: add random bits to the message to ensure a fixed total length



Layered Encryption + Padding

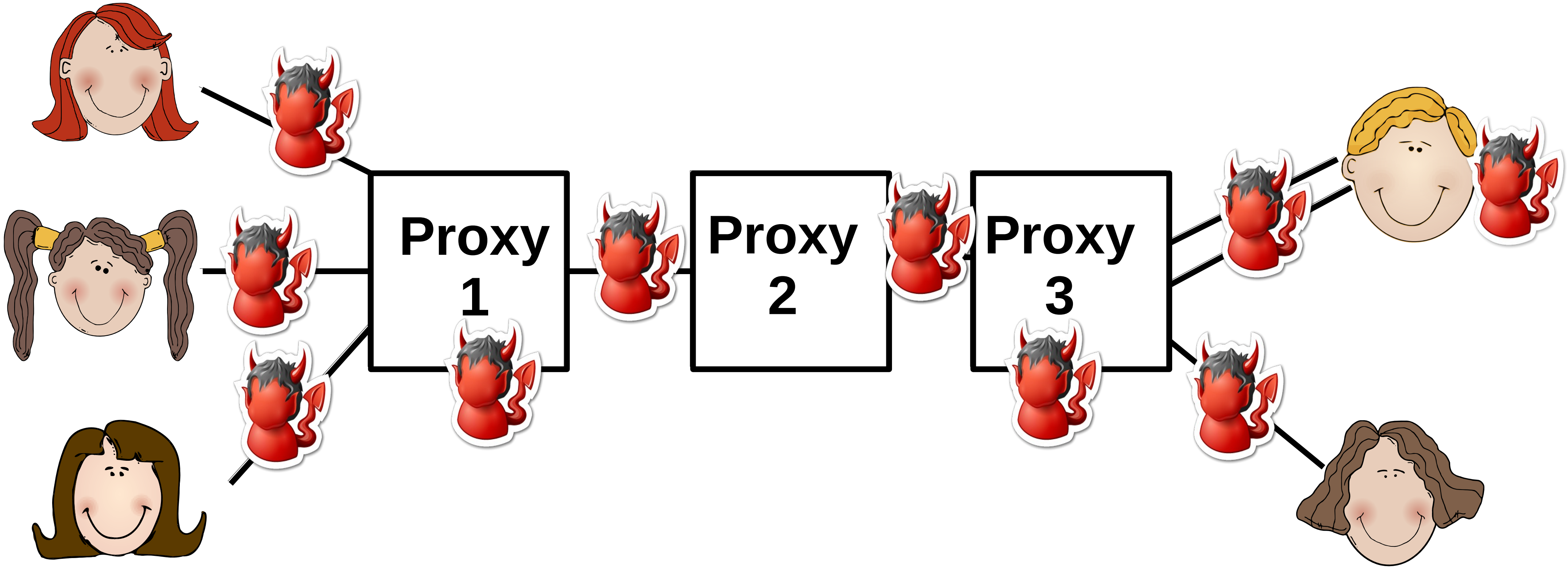
- Pad message to fixed length: pad(msg)
- EncProxy1(EncProxy2(EncProxy3(msg,Rec)))
- Usually for confidentiality: EncProxy1(EncProxy2(EncProxy3(EncRec(msg), Rec)))



Does layered encryption and padding achieve Sender-Message Unlinkability in this setting?

Layered Encryption + Padding

Karlsruhe Institute of Technology



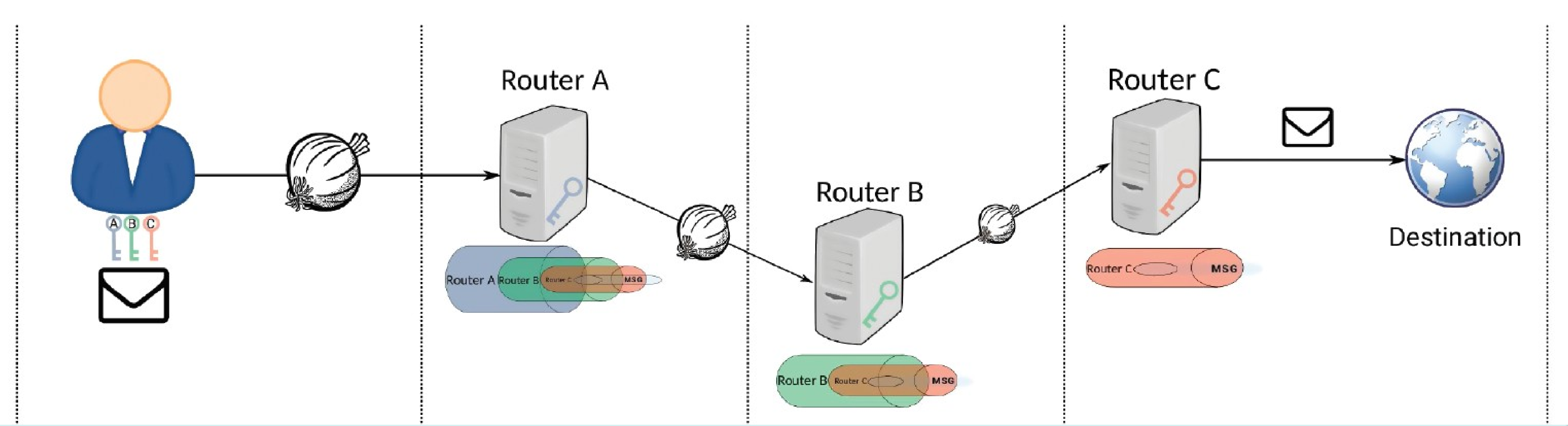
Unlinks sender & receiver, as well as sender & message cryptographically even against a global passive adversary and up to n-1 corrupt proxies!

Timing and Traffic Analysis attacks still possible

KASTEL

Protocol Class: Onion Routing

uses layered encryption and padding
here: proxies = routers (= relays)



Clever circuit setup: constructing symmetric keys for performance

Onion Routing concept

- Setup: Sender picks sequence of routers and exchanges symmetric keys
- Sending a message:
  - Pad and encrypt message in a layered fashion
  - **Include routing instruction into layered encryption**:
    EncRouter1(Router2, EncRouter2(Router3, EncRouter3(Rec, msg)))
  - Forwards result (=onion) to the first router

- Onion Routers (ORs):
  - Receive the onion, remove one layer of encryption, **pad it** and forward it to the next hop.
  - The first node (entry node) is aware of the identity of the sender and the next hop
  - The last node (exit node) is aware of the final destination, message and its predecessor node.

The Onion Router (Tor)

- Largest, most well deployed anonymity preserving service on the Internet
  - Publicly available since 2002
  - Continues to be developed and improved
  - Instrumental to the Arab Spring in 2010 and Snowden's revelations in 2013
- Currently, ~7,000* Tor relays around the world
  - All relays are run by volunteers
- ~ 3,000,000* users
- Extensions (better security, efficiency, deployability)

* https://metrics.torproject.org

Onion Routing protocols: TOR

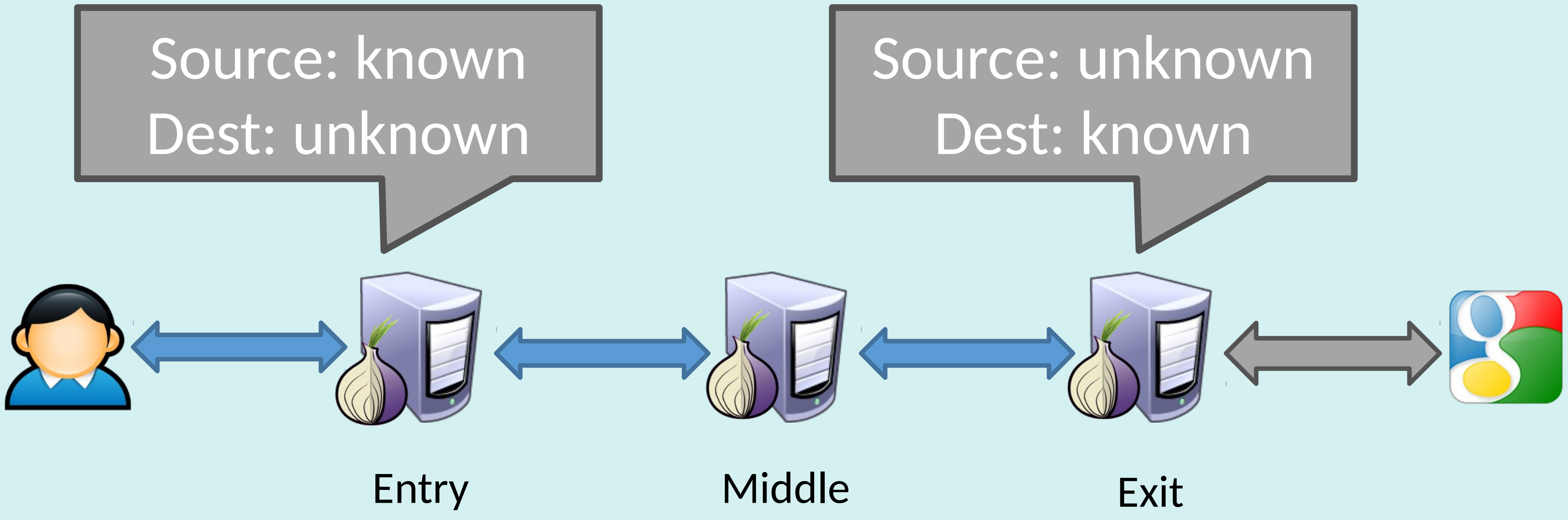
Karlsruhe Institute of Technology

- TOR has Authoritative Servers that:
  - Publish a list (called consensus) of available relays and their information (IP, keys)
  - Updates it regularly (typically every hour)
- Users run a SW called Onion Proxy that handles all TOR related processes
  - E.g., it gets the *consensus* and selects nodes (usually 3) to build a circuit
  - Node selection policy: high-bandwidth nodes with higher probability
  - Build new circuits periodically

Do we need to trust the authoritative servers?

KASTEL

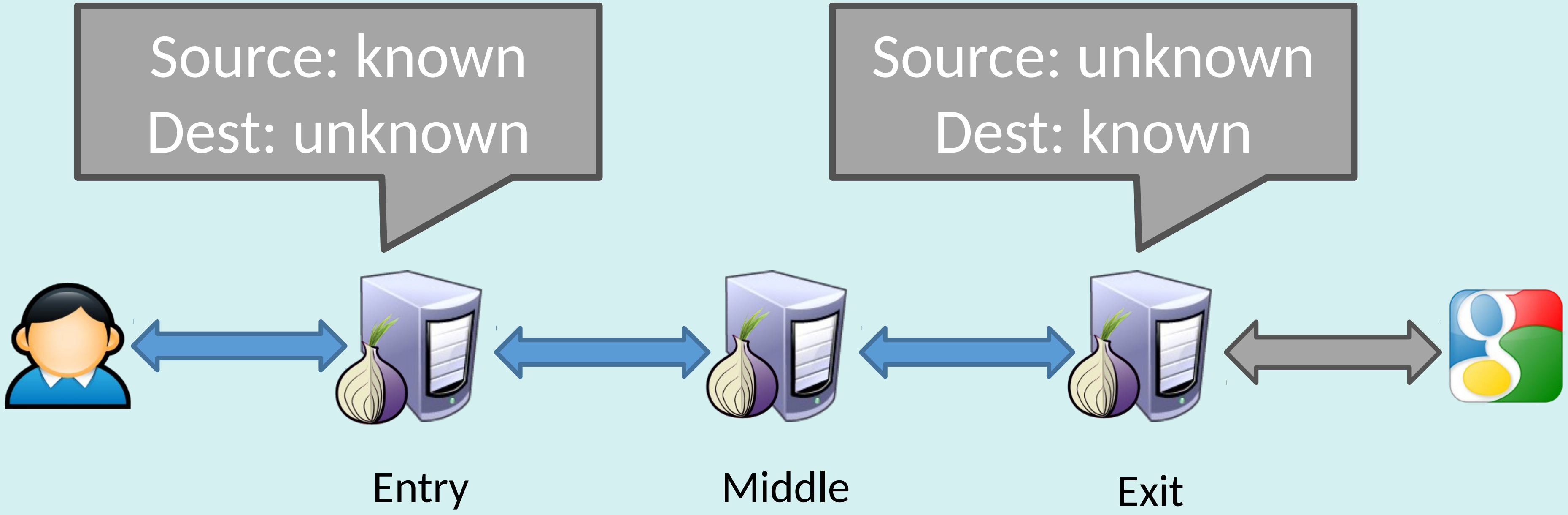
TOR’s Privacy



- Tor users can choose any number of relays
  - Default configuration is 3

Traffic Analysis and timing attacks if attacker controls entry and exit!

TOR’s Privacy



- Tor users can choose any number of relays
  - Default configuration is 3

Even worse: circuits are periodically renewed!

How can we mitigate the risk to pick a corrupt entry and exit node?

Traffic Analysis and timing attacks if attacker controls entry and exit!

Predecessor Attack

- Client periodically builds new circuits
  - Over time the chances to pick corrupt first and last relay increase!

- Mitigation: Guard nodes
  - Tor client selects a few relays at random to use as entry points
    - Pick stable and reliable guards (long uptimes, high bandwidth)
  - uses only those relays for her first hop during a few months

TOR and Onion Routing Summary

Karlsruhe Institute of Technology

- Use layered encryption, padding and a proxy-chain to distribute trust and unlink observations
- FIFO-like forwarding, no delay
- Susceptible to traffic analysis and timing attacks of the global passive adversary (or first and last router) → Guards as mitigation
- Sender Unlinkability for local adversaries
- Applicable to low latency services (e.g., browsing)

  ✉ more users = larger anonymity set

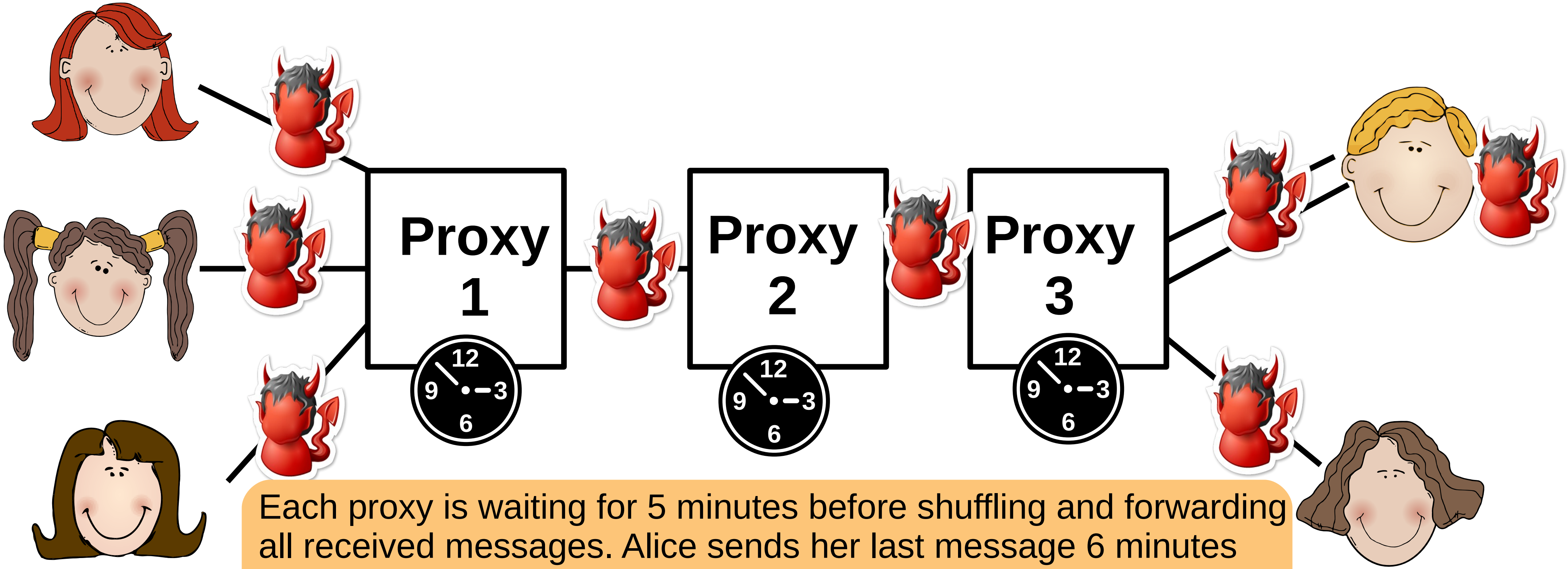
KASTEL

Protect against Timings – Mixing

Principle 3 & 4 (unlink & randomize observations)

Timings & traffic patterns are used for linking…

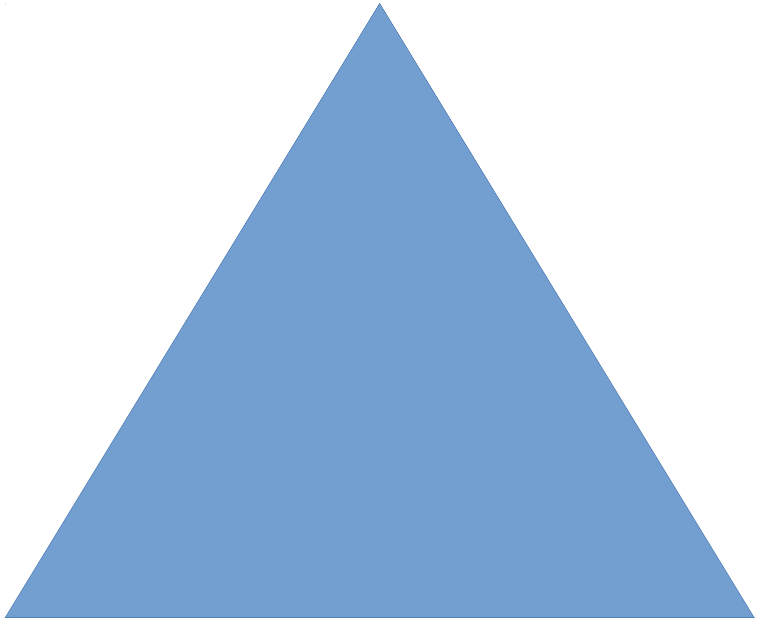
→ collect message at each proxy (delay) and forward in random order



Each proxy is waiting for 5 minutes before shuffling and forwarding all received messages. Alice sends her last message 6 minutes before Claire sends any message. Can the adversary tell whether a received message is from Alice or Claire?

Layered Encryption, padding and Mixing

Sender Unlinkability
(a batch corresponds to one round)



Global passive adversary, corrupt receiver and up to n-1 corrupt proxies

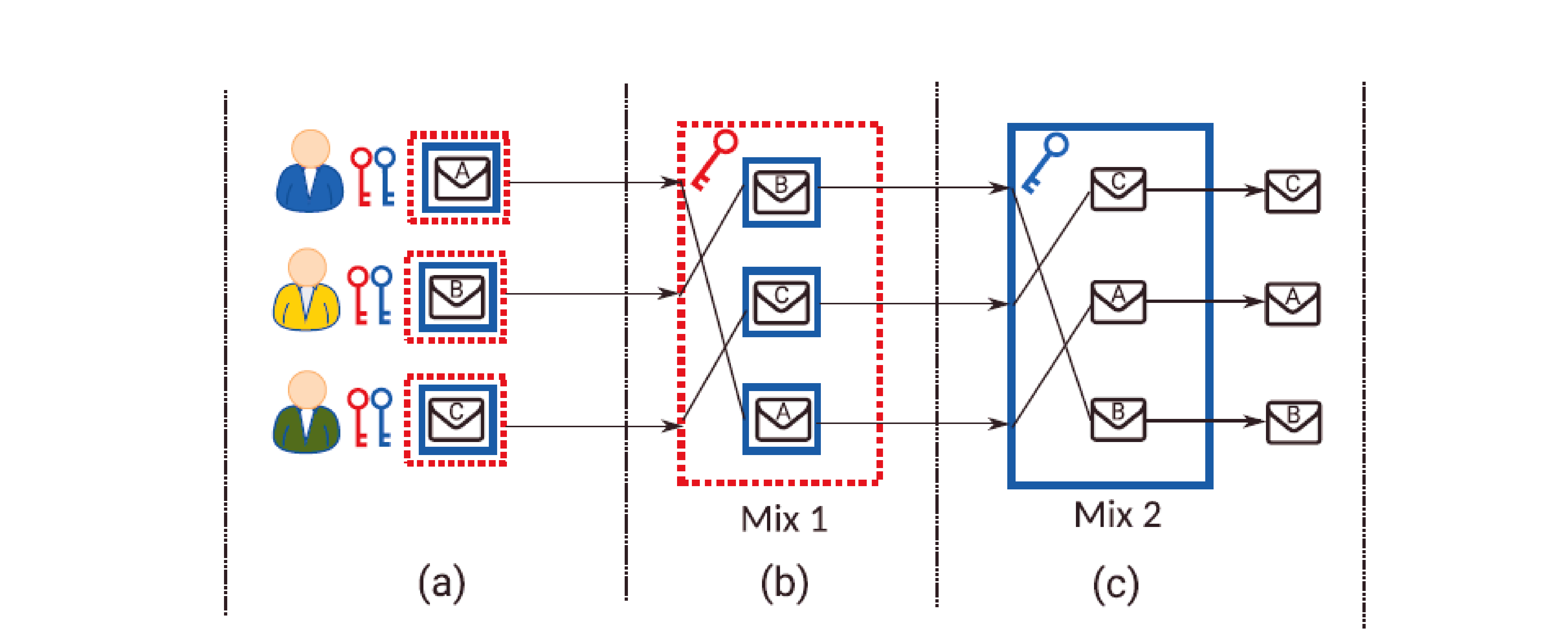
Much higher latency
slightly more computation at proxies
Need proxies

Mix Systems: concept

- originally proposed by Chaum (1981)
- Proxies = *mixes* (= *mix nodes* = *relays*):
  - cryptographically transform messages to unlink input and output messages based on content or size (layered encryption and padding)
  - **Shuffle** ("mix") input messages and output them in a reshuffled form to unlink messages based on their order/timing
- Different *(mix) node selection strategies* and *mixing strategies*

Chaum's Mix: Mix Cascade

relay messages through a **fixed** sequence of mix nodes



Chaum's Mixnet:

Karlsruhe Institute of Technology

- ***Mix Cascade***: relay messages through a fixed sequence of mixes
  - mixes are **selected deterministically**
  - *Fixed size* messages encrypted (in a layered fashion) with the public key of each mix in the cascade
  - Message transfer: each mix:

    - waits for messages (**until k received**)

    - decrypts the corresponding layer with its private key

    - shuffles messages (*sorts lexicographically*)

    - forwards batch of messages to the next mix
  - repeated until the last mix delivers the data to its final destination

KASTEL

Mix node selection strategies

- Availability drawback: Cascades = **single point of failure**
- *Improve Availability: Free-route* mix networks
  - route is not fixed, any sequence of nodes from the network can be used for relaying messages

Mixing strategies

- Flushing algorithm: specifies the precise timing when messages are forwarded
  - Threshold mixes: collect messages until a threshold is reached

Does the performance of treshold mixes decrease
(i.e. higher delivery latencies) if the traffic is low?

Mixing strategies

- Flushing algorithm: specifies the precise timing when messages are forwarded
  - Threshold mixes: collect messages until a threshold is reached

What could we do instead to avoid decrease
(i.e. higher delivery latency) if the traffic is low?

Mix Systems: mixing strategies

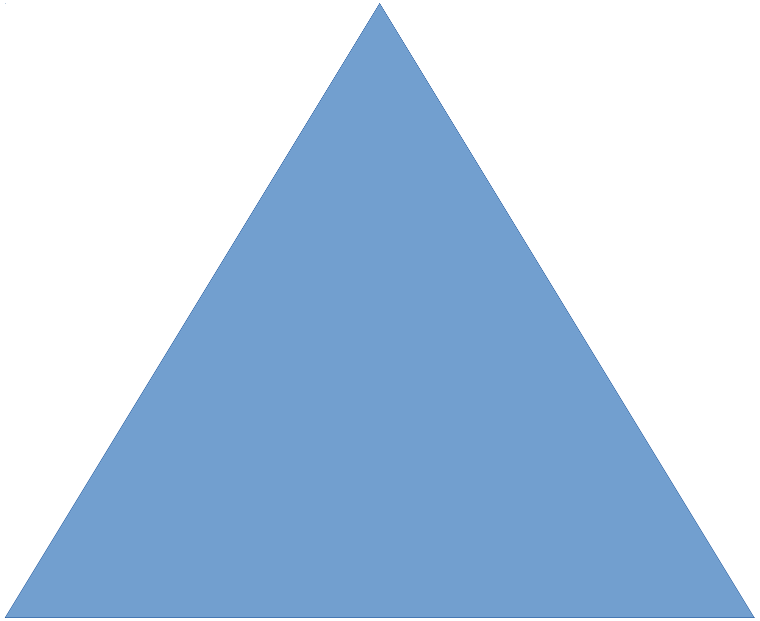
- Timed Mixes: enforce a time restriction for flushing out messages
  - privacy vulnerable to low traffic
- Threshold mixes: collect messages until a threshold is reached
  - Very high latency if the traffic load is low
- *Stop-and-Go* mixes: independent random delays are assigned to each mix
  - Performance is not dependent of the other users
  - Vulnerable when incoming traffic is low
- Pool Mixes: keep messages in pool, send out randomly selected messages, if new messages arrive
  - Suitable for fluctuating traffic

Mix Systems: Summary

- Layered encryption, padding and delaying in a proxy chain
- Show very heterogeneous designs: *free-route* vs. *Cascades*, *pool vs. Threshold vs. Stop-and-go vs. Timed*
- *Unlink senders from messages and receivers* ***also in the timing dimension*** *against global adversaries*
- High-latency
  - non-interactive services where users are willing to tolerate delays that can range from seconds to hours
  - suitable for services like e-mail and electronic voting

Layered Encryption, padding and Mixing

Sender Unlinkability



Global passive adversary, corrupt receiver and up to n-1 corrupt relays

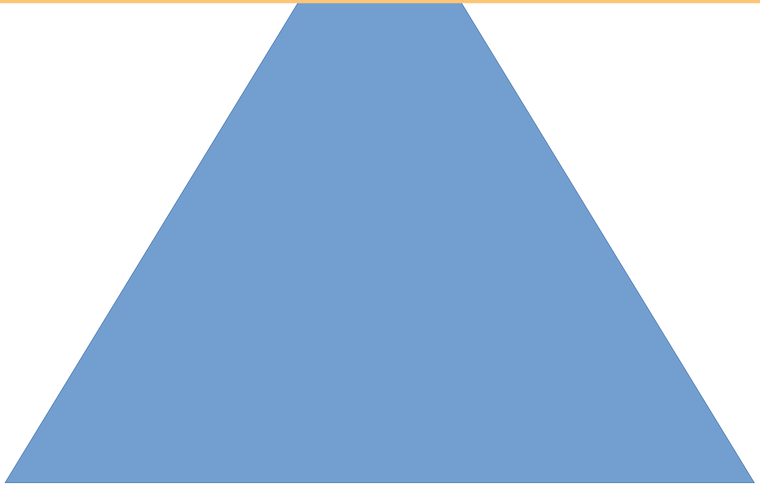
Much higher latency
slightly more computation at proxies
Need proxies

Layered Encryption, padding and Mixing

Karlsruhe Institute of Technology

Sender Unlinkability

What if I wanted to achieve Sender Unobservability instead?
Is there a way to increase the protection of a mixnet?

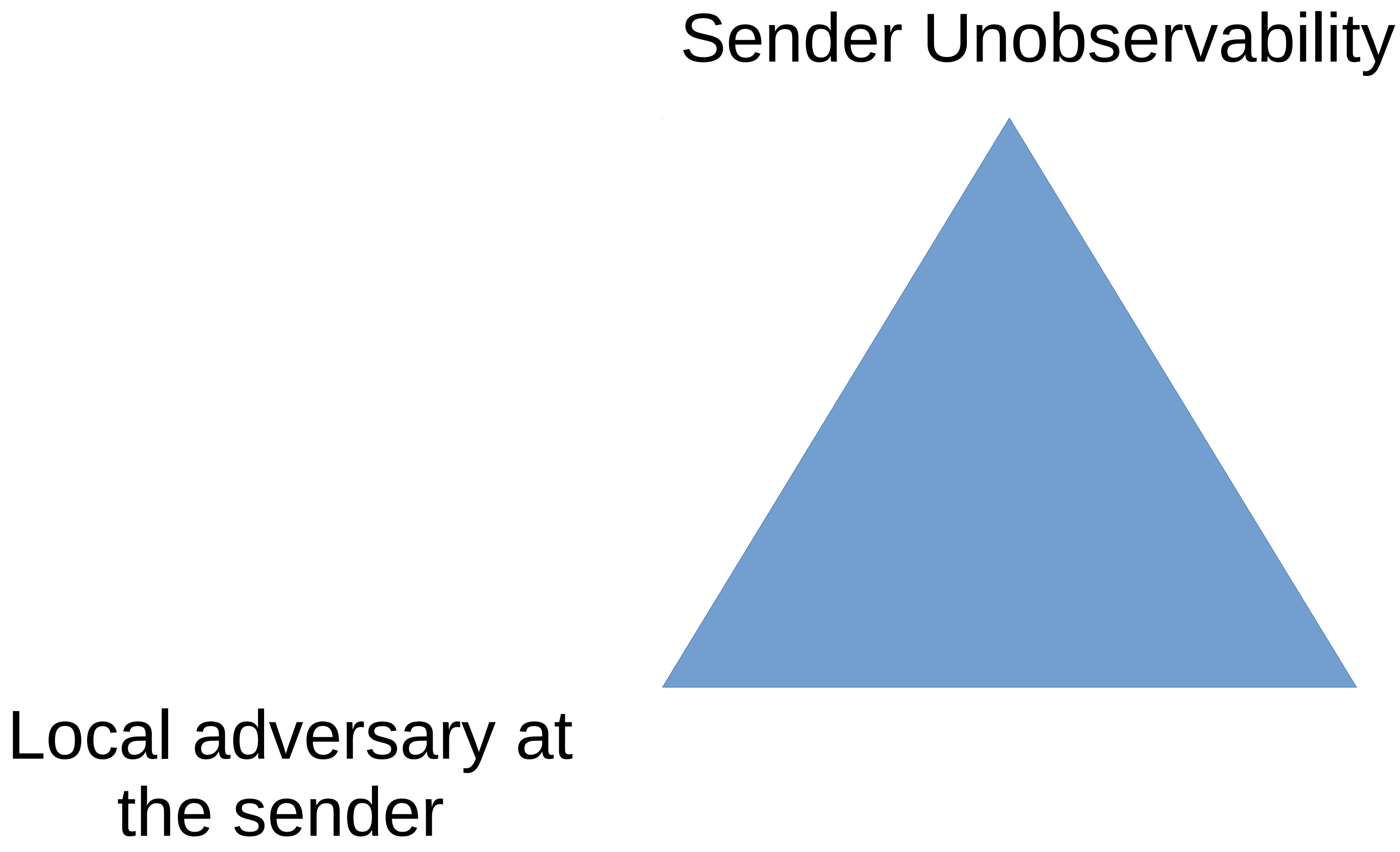


Global passive adversary, corrupt receiver and up to n-1 corrupt relays

Much higher latency
slightly more computation at proxies
Need proxies

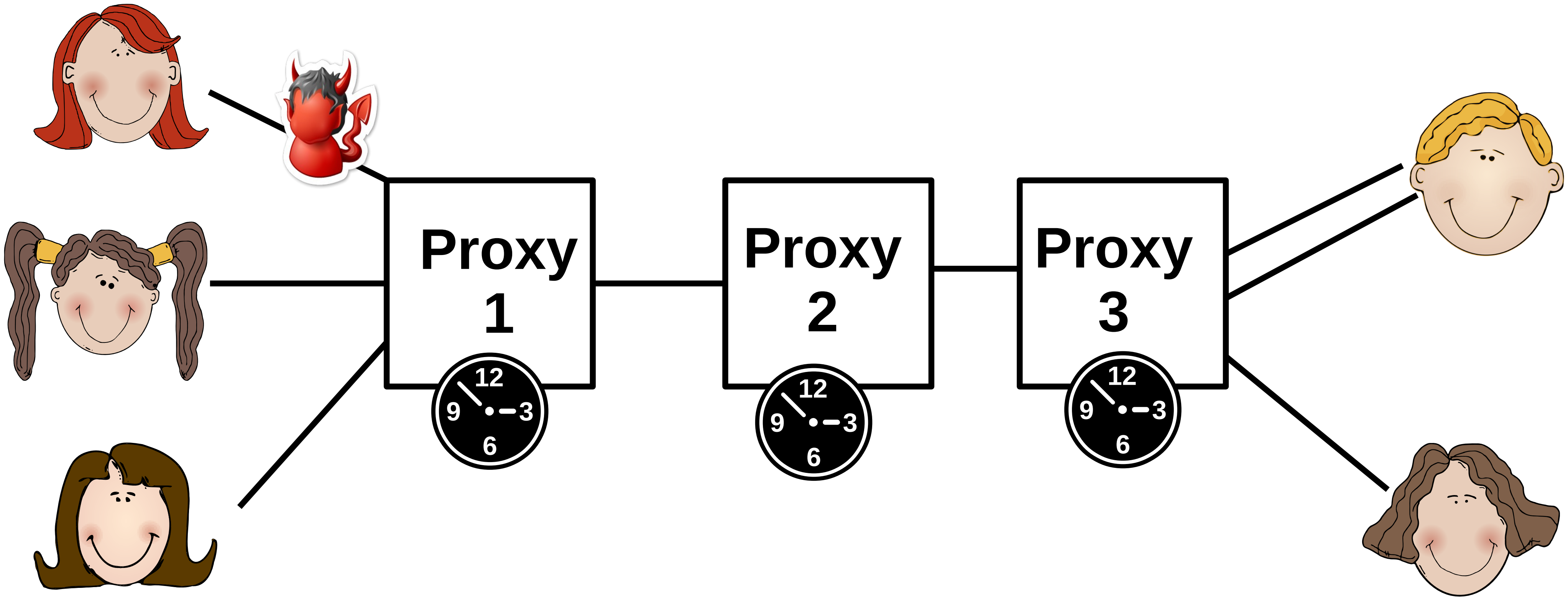
KASTEL

Hiding Activity and Frequencies



Hiding Activity and Frequencies

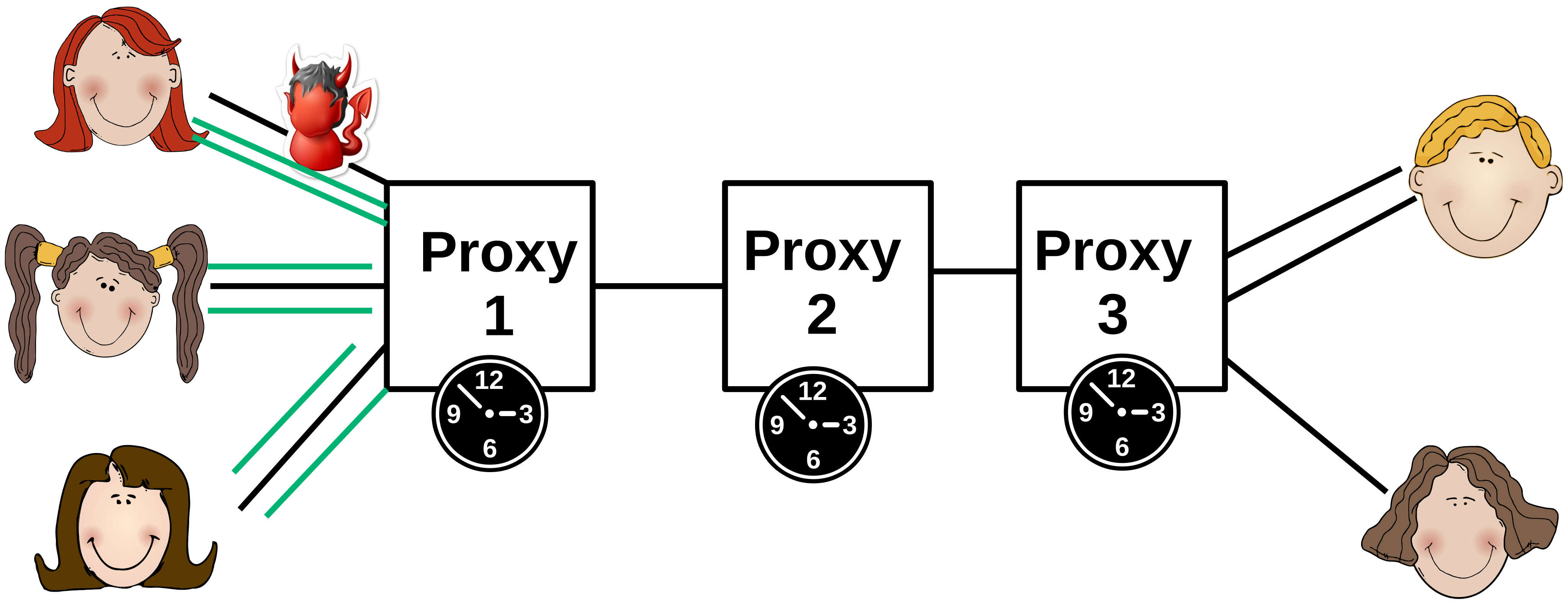
- Every packet is a “real” communication



Dummy Traffic

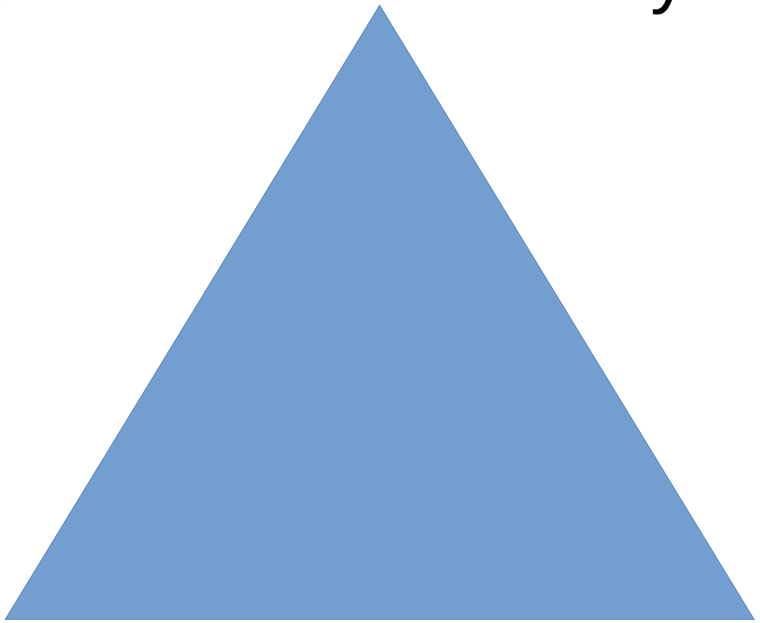
Principle 4 or 5 (randomize or fix observations)

- Add “fake” communications that are dropped at some party
- Need to be indistinguishable from real communications for the adversary



Adding Dummy Traffic on the first link to Mixnet

(upgrade to)
Sender Unobservability
[also increases anomity set]



Local adversary at
the first link

(additional)
Bandwidth overhead =
network load

Types of Dummy Traffic

- Strategy: Pad to a fixed number of communications or randomize number per round and user
- Area: end-to-end, link-based or anything in between
- Communication partner: real user or dedicated party
- Amount: e.g. >=1 (hide activity) or = max number of delivered messages (hide frequency)

- Combination of choices determines the cost in terms of bandwidth overhead

Dummy Traffic: Summary

- Usually combined with other techniques (e.g. Mixing, Onion Routing)
- Hide activity and sending/receiving frequencies
- Many variations with different cost and effects possible
- Improves anonymity set size

An Alternative Approach to Hide Senders?

For receivers: Broadcast! The message is received by everyone!

Can we make it look like the messages is sent from every user (without trusting all other users)?

An Alternative Approach to Hide Senders?

Karlsruhe Institute of Technology

For receivers: Broadcast! The message is received by everyone!

Can we make it look like the messages is sent from every user (without trusting all other users)?

YES, and Chaum knows how: we ensure that every user contributes a part needed to recover the final message…

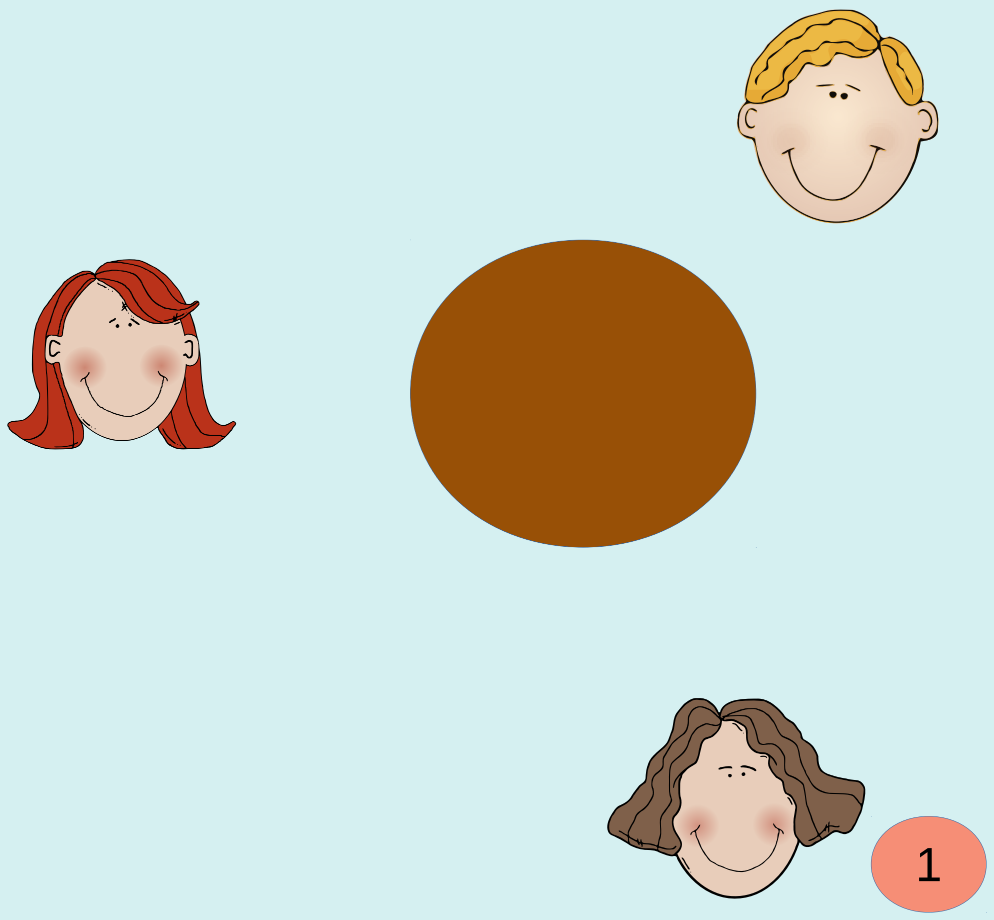
KASTEL

DC-Nets concept

- The idea of DC-Nets was first proposed by Chaum (1988)
- Inspired by a scenario:
  - 3 cryptographers went for dinner
  - they learn that the bill is payed

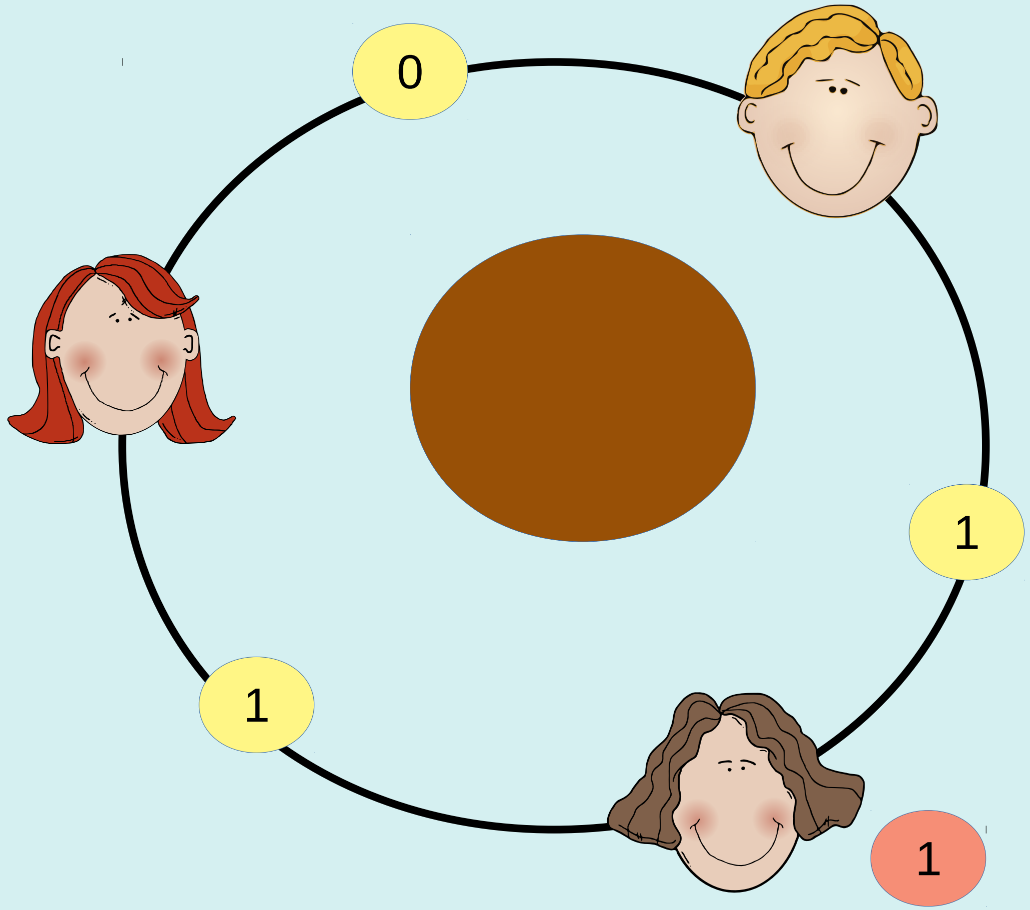
  Was the dinner payed anonymously by one of them or by the National Security Agency (NSA)?

  - can they figure this out while respecting anonymity?



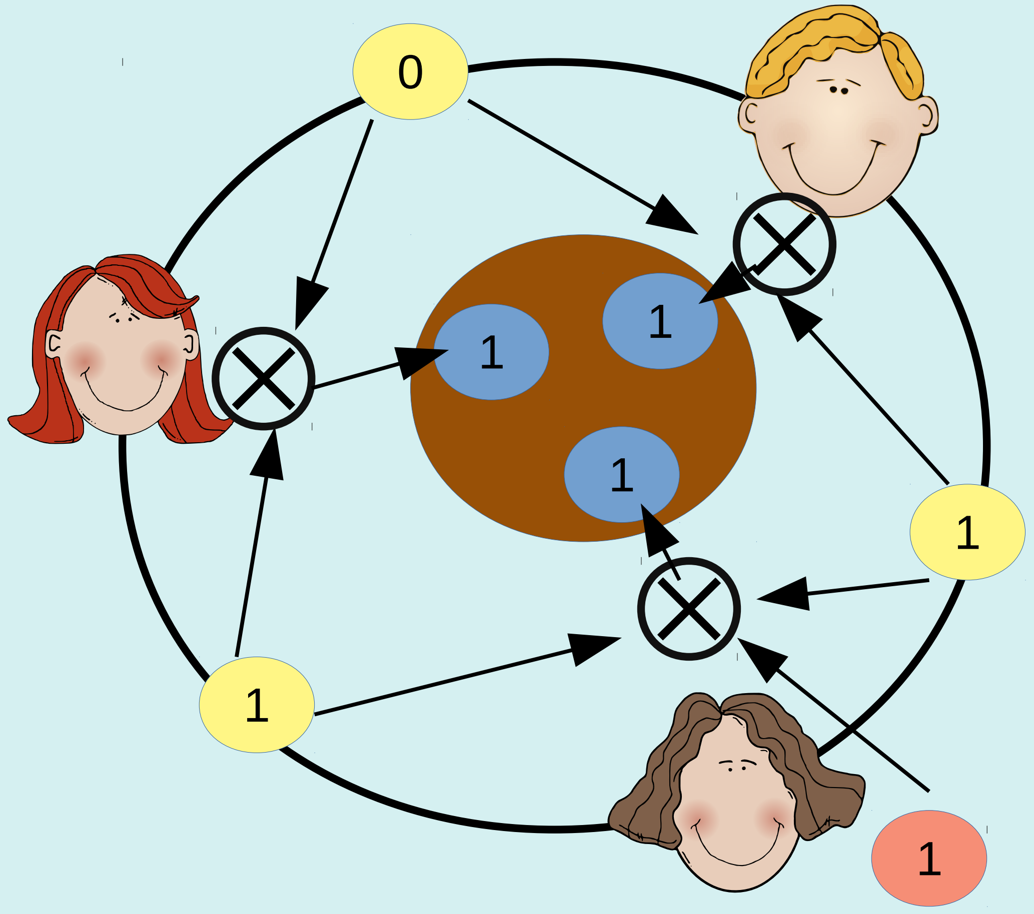
DC-Nets concept: superposed Sending

- Flip a coin with each neighbor



DC-Nets concept: superposed Sending

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result

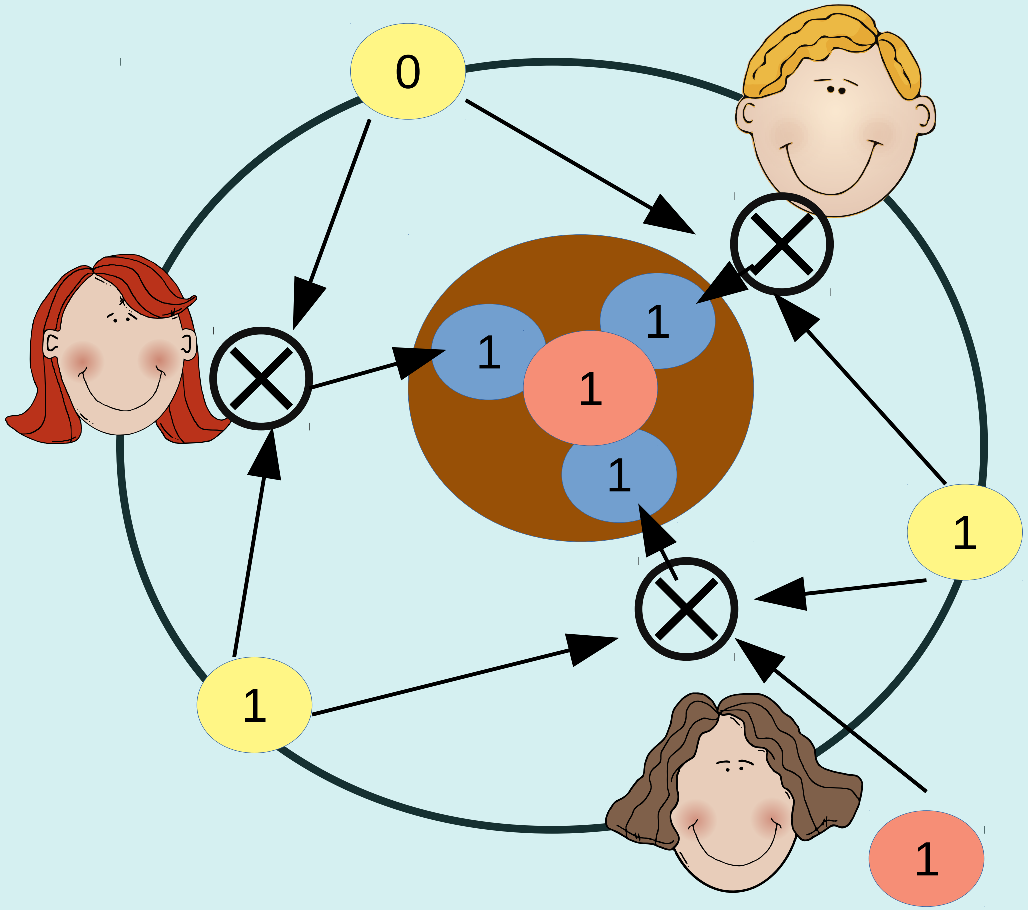


DC-Nets concept: superposed Sending

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result
- XOR all local results:

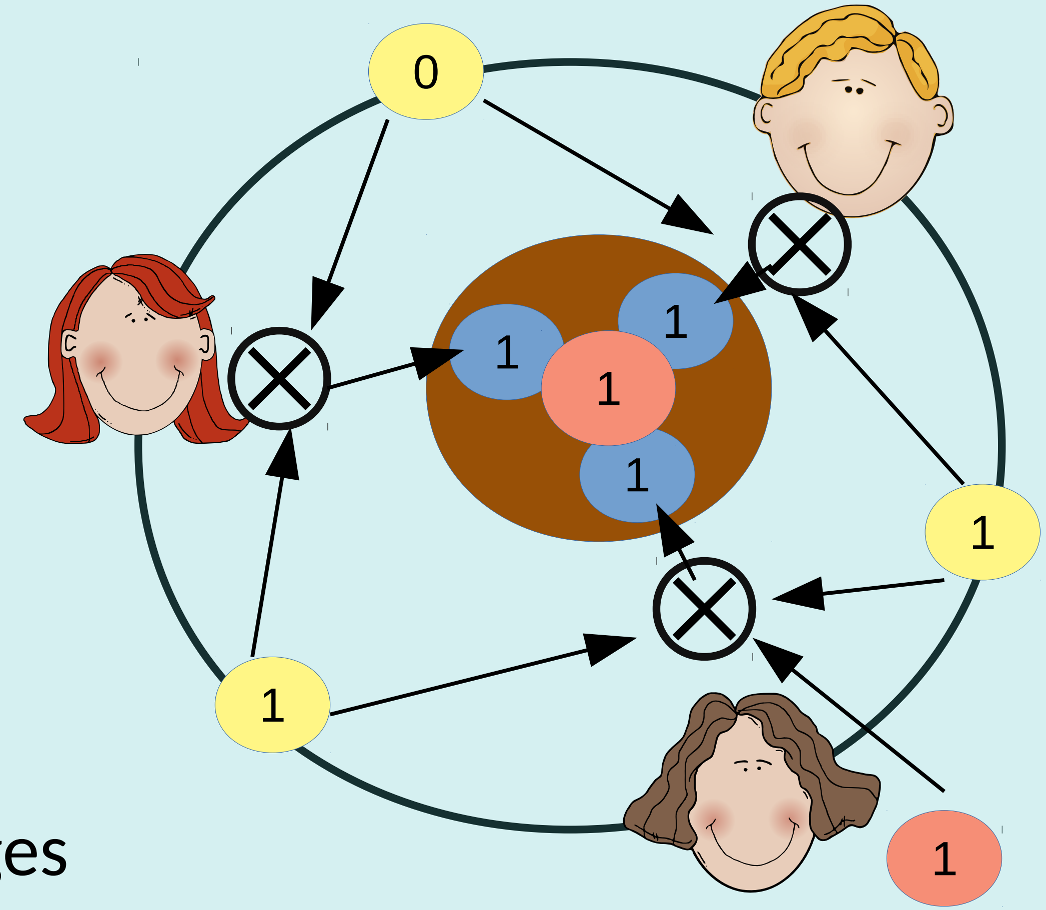
  0: NSA payed for the dinner

  1: A cryptographer payed for the dinner



DC-Nets concept: superposed Sending

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result
- XOR all local results:
  0: NSA payed for the dinner
  1: A cryptographer payed for the dinner



- Transmits 1 bit → Repeat for longer messages

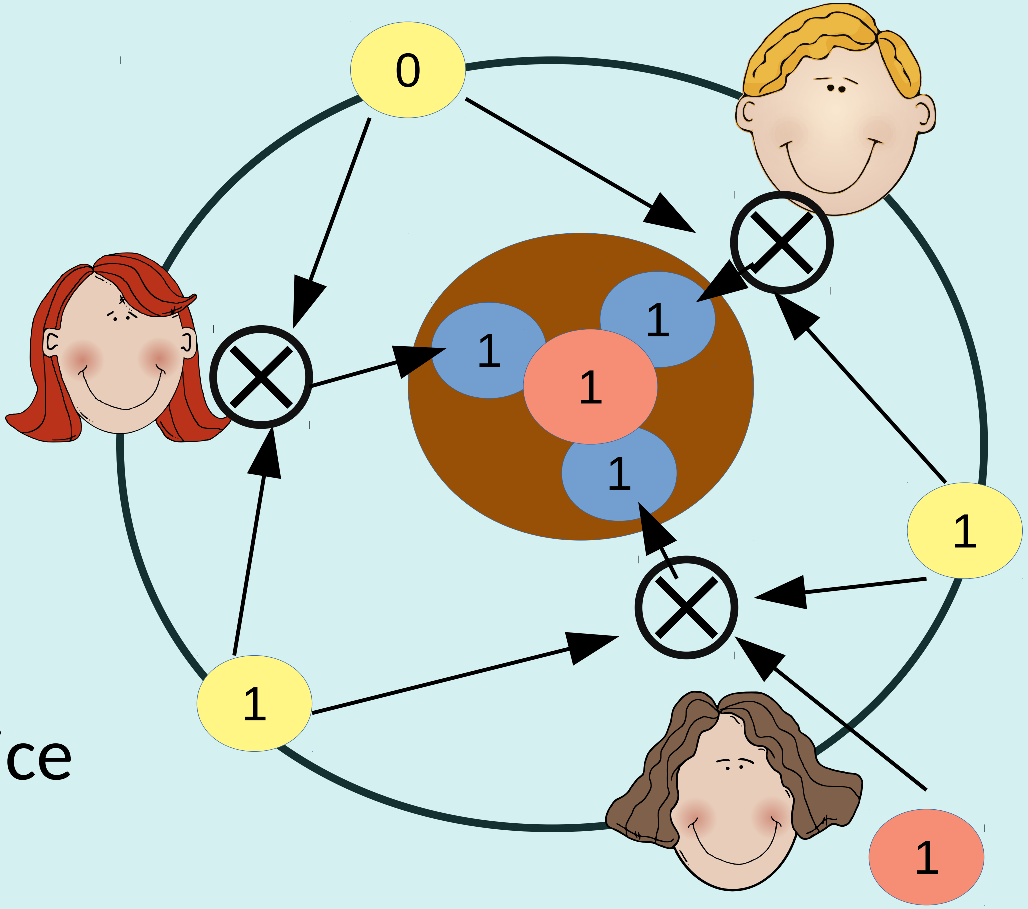
DC-Nets concept: superposed Sending

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result
- XOR all local results:

  0: NSA payed for the dinner

  1: A cryptographer payed for the dinner

- Correct? Yes, each coin flip result is used twice
  → cancel out in XOR



DC-Nets concept: superposed Sending

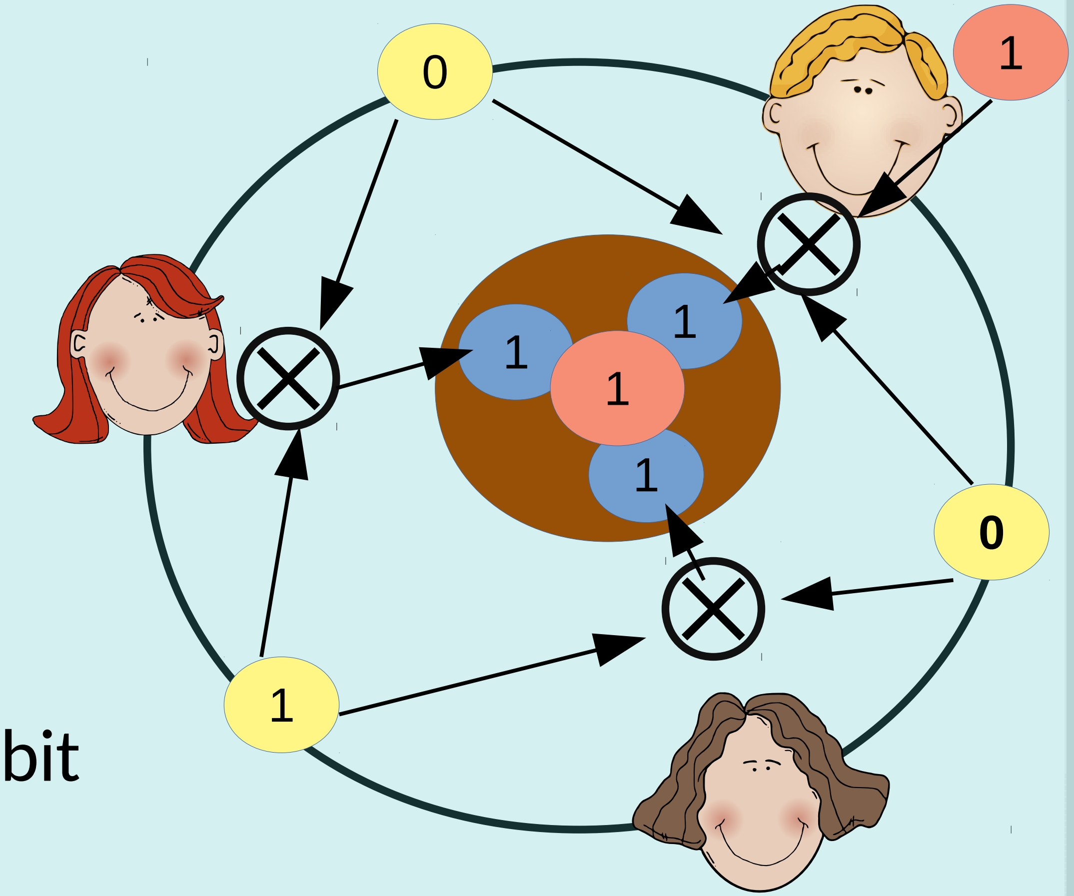
Karlsruhe Institute of Technology

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result
- XOR all local results:

  0: NSA payed for the dinner

  1: A cryptographer payed for the dinner

- Private? Yes, payer is protected by random bit shared with other honest cryptographer

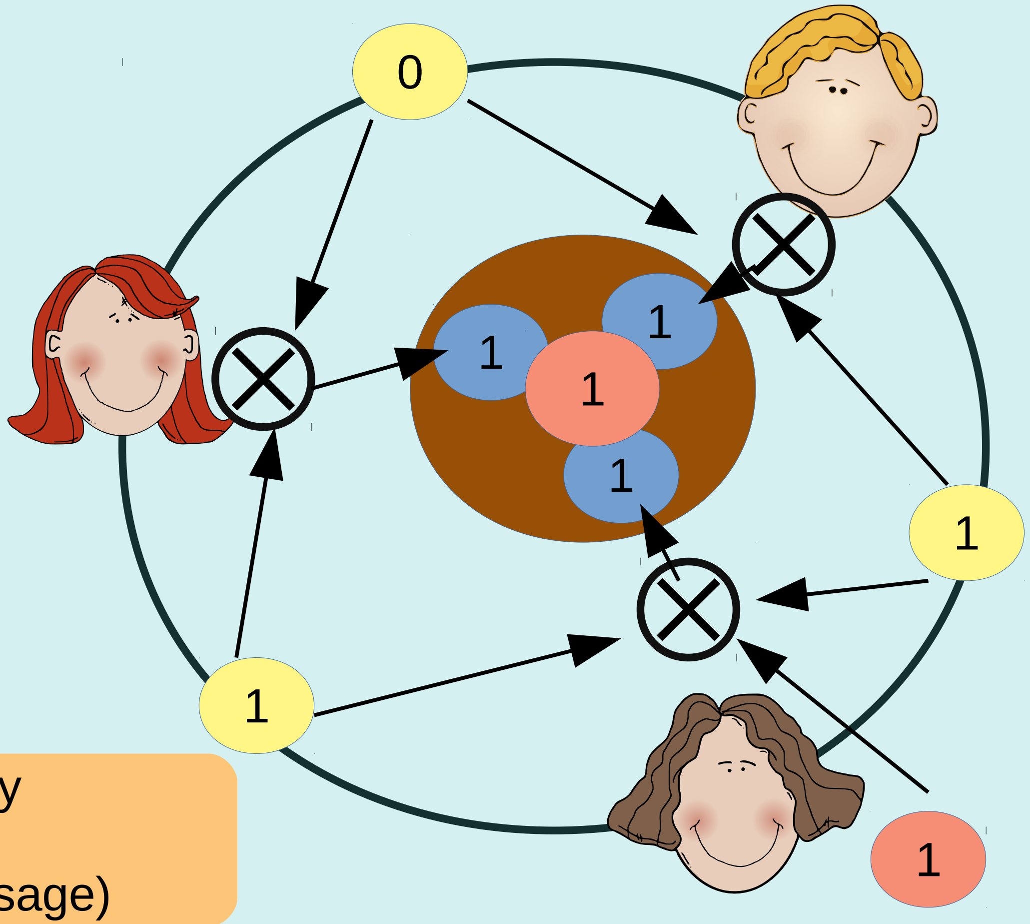


KASTEL

DC-Nets concept: superposed Sending

Karlsruhe Institute of Technology

- Flip a coin with each neighbor
- XOR coin results
- If you payed: reverse result of XOR
- Reveal local result
- XOR all local results:
  0: NSA payed for the dinner
  1: A cryptographer payed for the dinner



Does this approach achieve Sender Unobservability against a global passive adversary?
(assume that multiple users might want to send a message)

KASTEL

DC-Nets: protocol features

Assume: At most one person sends per round

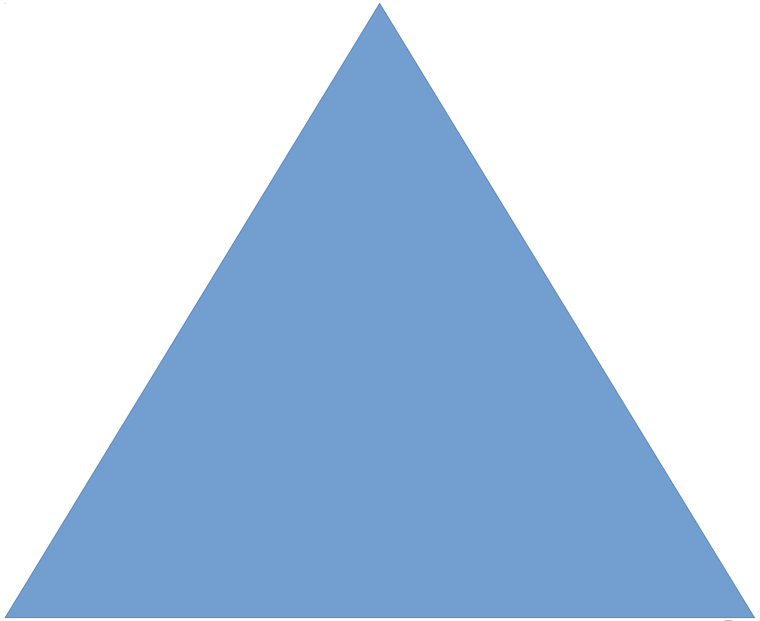
Collisions are possible!

- 1 sender: message is delivered
- 2 senders: both try to send and the output will be their messages XORed
- Can be used to disrupt the protocol (availability)

- New proposals introduce topology modifications and mechanisms to detect disruption

Superposed Sending (with collision prevention)

Sender Unobservability



Global passive adversary and up to n-2 corrupt participants

High bandwidth overhead
Collisions and DoS
Scalability issues

Learning Goals

- Understand the Problem
  - Motivation and Setting
  - Dimensions and Terminology

- Understand the Solution(-space)
  - Solution ideas and prominent protocols:
    - Random Walk
    - Onion Routing
    - Mix Networks
    - Dummy Traffic
    - DC Networks
  - Effects of design decisions

Summary Principles:

- Principle 1: Indirection
- Principle 2: Distribution of Trust
- Principle 3: Unlink Observations
- Principle 4: Randomize Observations
- Principle 5: Fix Observations

Summary Strategies:

- Proxy
- Proxy Chain
- Encryption
- Padding
- Delays (Mixing)
- Dummy Traffic
- Superposed Sending (DC-Nets)

Protocol classes

Name	Goal (Sender side)	Adversary	Cost
Random Walk	Sender Unobservability	External, passive	(Low) Latency
Onion routing	Sender Unlinkability	Local, passive adversary	Low Latency
Mixnets	Sender Unlinkability	Global, passive, corrupt up to n-1 mixes on path	High Latency
+ Dummy Traffic	Sender Unobservability	variable	Bandwidth
DC-Nets (no collisions)	Sender Unobservability	Global, passive, corrupt up to n-2 participants	Bandwidth, DoS vulnerability

Summary

- Criteria (the 3 “what”s)

- Overview over solution space
- Understanding of the interplay of adversary, goal and cost
- Understanding of combination of strategies in protocols

- We focused on passive attacks and sender protection (there is much more to learn if you’re interested!)

Further reading

- Protocol Overview: Shirazi, Fatemeh, et al. "A survey on routing in anonymous communication protocols." ACM Computing Surveys (CSUR) 51.3 (2018): 1-39.
- Goals: Kuhn, Christiane, et al. "On Privacy Notions in Anonymous Communication." Proceedings on Privacy Enhancing Technologies 2 (2019): 105-125.
- Crowds: Reiter, Michael K., and Aviel D. Rubin. "Crowds: Anonymity for web transactions." ACM transactions on information and system security (TISSEC) 1.1 (1998): 66-92.
- Tor: Dingledine, Roger, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Naval Research Lab Washington DC, 2004.

Further reading

Karlsruhe Institute of Technology

- Tor: https://www.torproject.org/
- Chaum Mix: Chaum, David L. "Untraceable electronic mail, return addresses, and digital pseudonyms." Communications of the ACM 24.2 (1981): 84-90.
- DC-Net: Chaum, David. "The dining cryptographers problem: Unconditional sender and recipient untraceability." Journal of cryptology 1.1 (1988): 65-75.
- Predecessor attacks: Wright, Matthew K., et al. "The predecessor attack: An analysis of a threat to anonymous communications systems." ACM Transactions on Information and System Security (TISSEC) 7.4 (2004): 489-522.

KASTEL