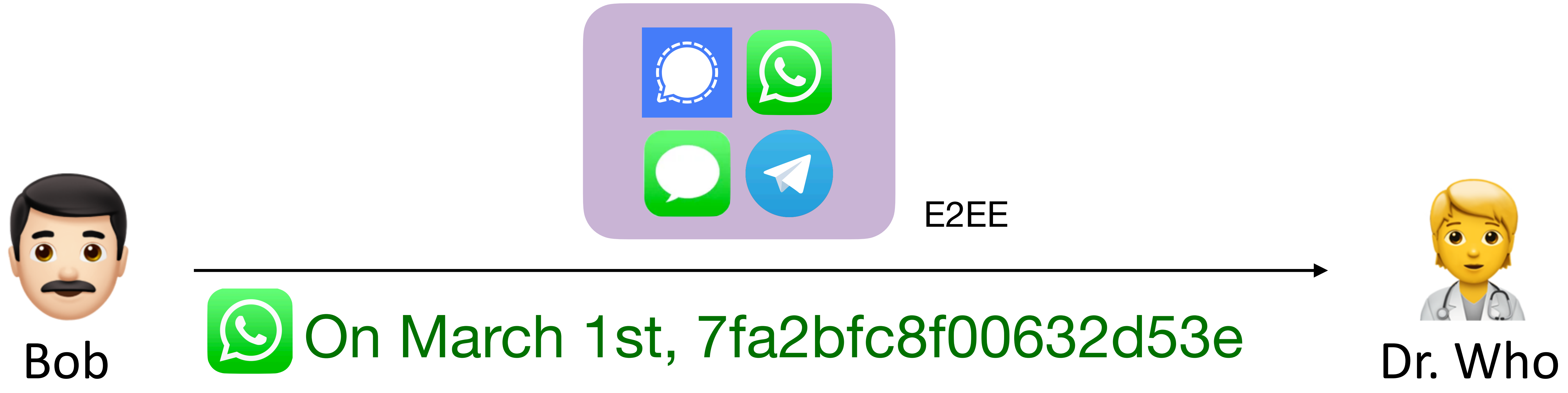
Boomerang: Metadata-private Messaging under Hardware Trust

Peipei Jiang[1,2] Qian Wang[1] Jianhao Cheng[1]
Cong Wang[2] Lei Xu[3] Xinyu Wang[4]
Yihao Wu[1] Xiaoyuan Li[1] Kui Ren[5]

[1]Wuhan University
[2]City University of Hong Kong
[3]Nanjing University of Science and Technology
[4]Tencent Inc.
[5]Zhejiang University

E2EE protects only payload, NOT metadata



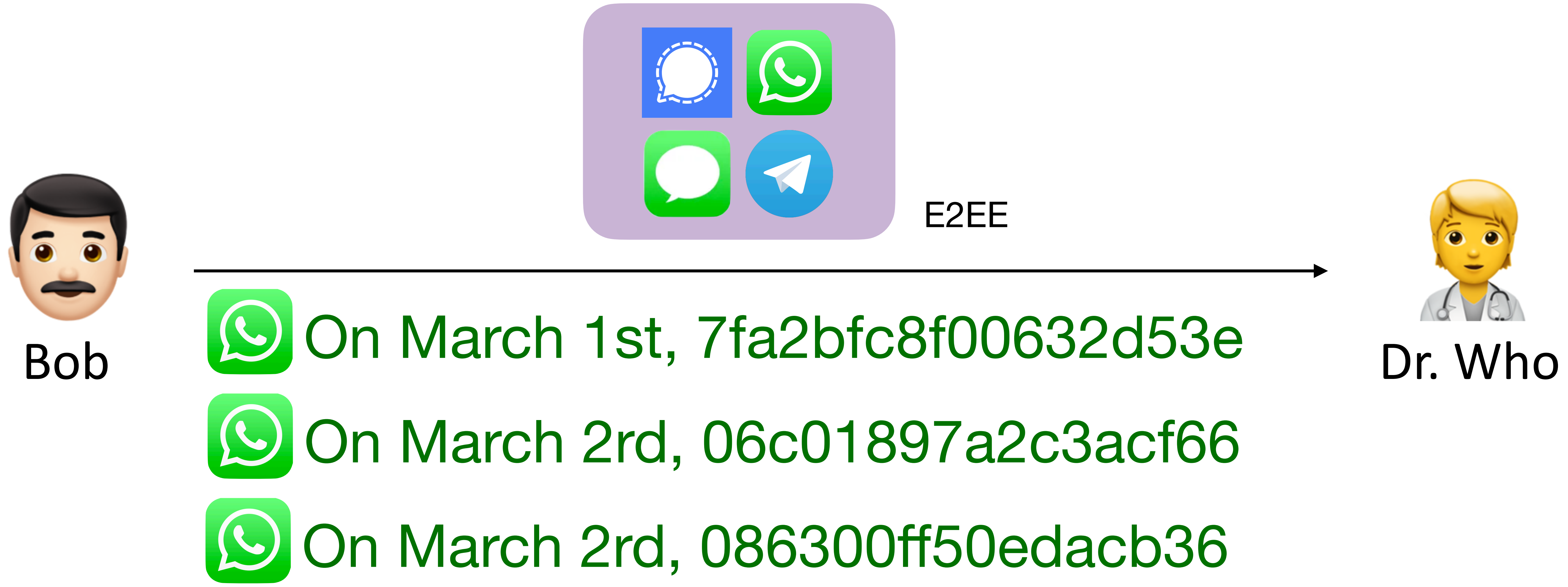
On March 1st, 7fa2bfc8f00632d53e

From: Bob
To: Dr. Who
Timestamp: 1647836660
Content: Doctor, I feel sick.

→ E2EE

From: Bob
To: Dr. Who
Timestamp: 1647836660
Content: 086300ff50edacb36

E2EE protects only payload, NOT metadata



On March 1st, 7fa2bfc8f00632d53e

On March 2rd, 06c01897a2c3acf66

On March 2rd, 086300ff50edacb36

From: Bob
To: Dr. Who
Timestamp: 1647836660
Content: Doctor, I feel sick.

→ E2EE

Exposed comm. metadata

From: Bob
To: Dr. Who
Timestamp: 1647836660
Content: 086300ff50edacb36

Metadata can be privacy-revealing

Sender identity

Receiver identity

Bob

Dr. Who

On March 1st, 7fa2bfc8f00632d53e

On March 2rd, 06c01897a2c3acf66

On March 2rd, 086300ff50edacb36

Time

Duration

Volume

Metadata can be privacy-revealing

Sender identity

Receiver identity

Bob

Dr. Who

On March 1st, 7fa2bfc8f00632d53e

On March 2rd, 06c01897a2c3acf66

On March 2rd, 086300ff50edacb36

Time

Duration

Volume

“Bob might have got some health condition.”

Who might see metadata?

- Your ISP sees your traffic data
- E2EE service providers
- Governments around the world
- Data brokers
- Advertisers
- …

whistleblowers

tracking general users

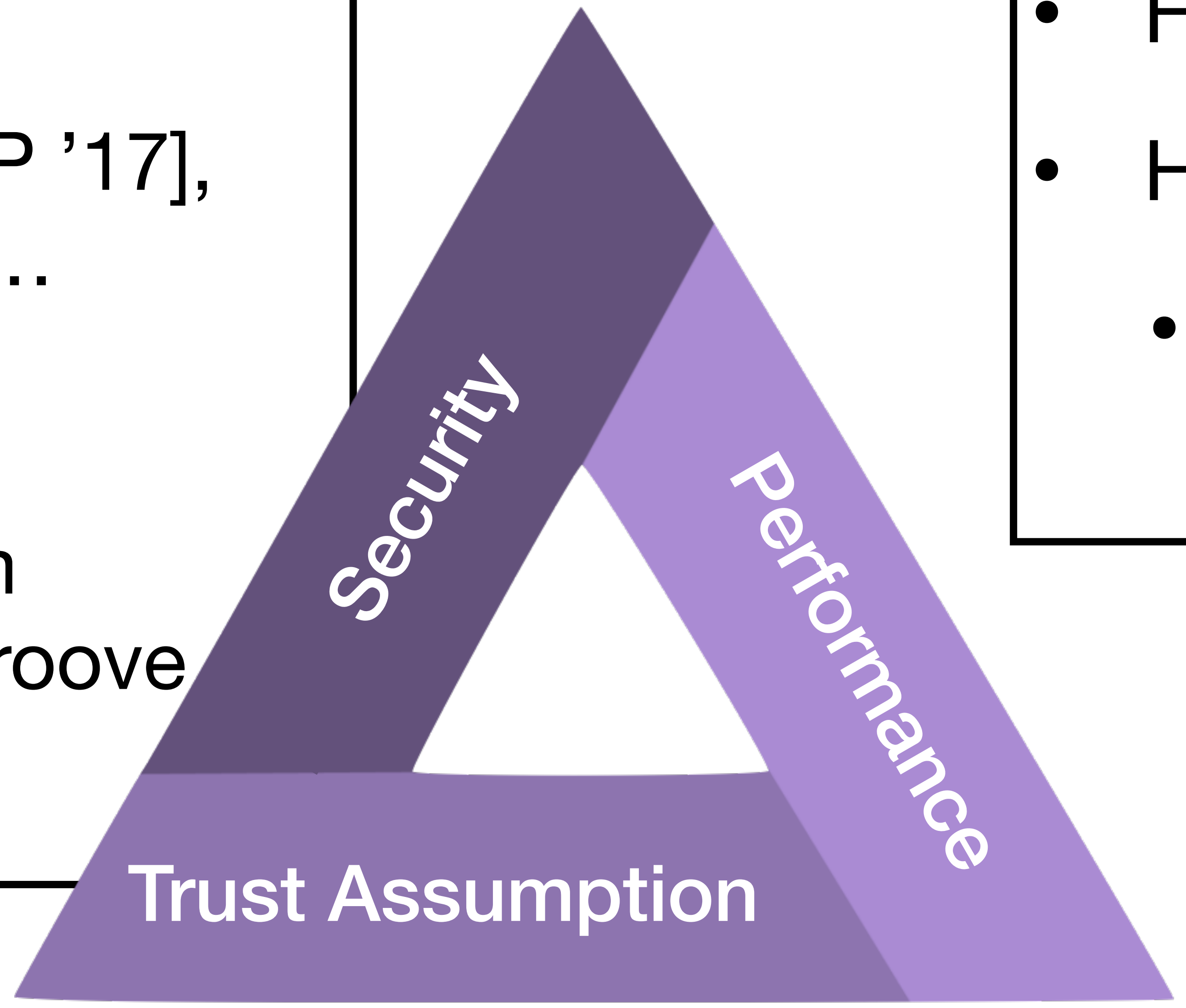
https://www.wired.com/story/opinion-data-brokers-are-a-threat-to-democracy/, April 13th, 2021

https://arstechnica.com/tech-policy/2021/03/t-mobile-will-tell-advertisers-how-you-use-the-web-starting-next-month/, March 10th, 2021

Progresses on metadata-private messaging

Balancing act among: security, performance, and trust assumption

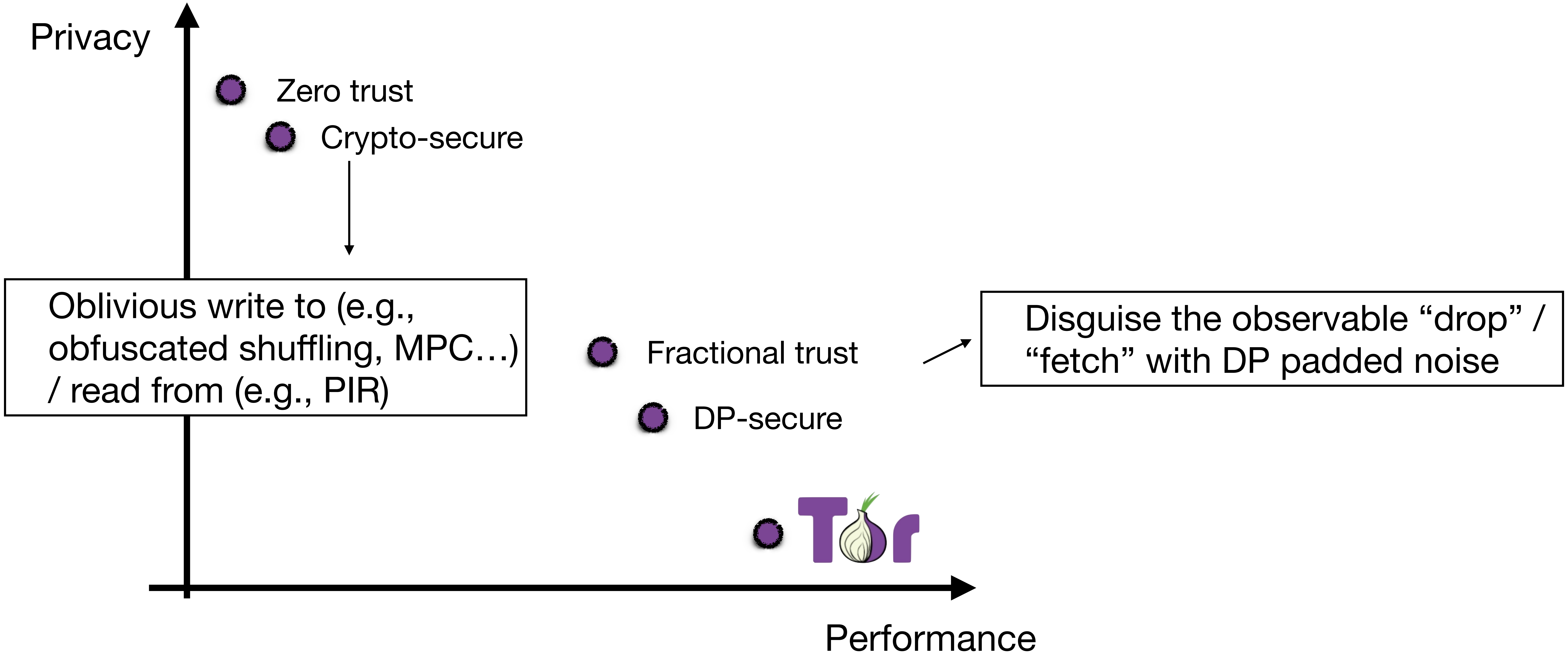
- Cryptographic security
  - E.g., Pung [OSDI ’16], Atom [SOSP ’17], XRD [NSDI ’20], Addra [OSDI’ 21]…
- Differential privacy security
  - E.g., Vuvuzela [SOSP’15], Stadium [SOSP’17], Karaoke [OSDI’ 18], Groove [OSDI ’22]…



- High throughput
- Horizontal scalability
  - E.g., Tor, Stadium [SOSP’17], Karaoke [OSDI’ 18], Yodel [SOSP’21]…

- Fractional trust
- Zero trust
  - E.g., Pung [OSDI ’16], Addra [OSDI ’21]…

Progresses on metadata-private messaging



Zero trust

Crypto-secure

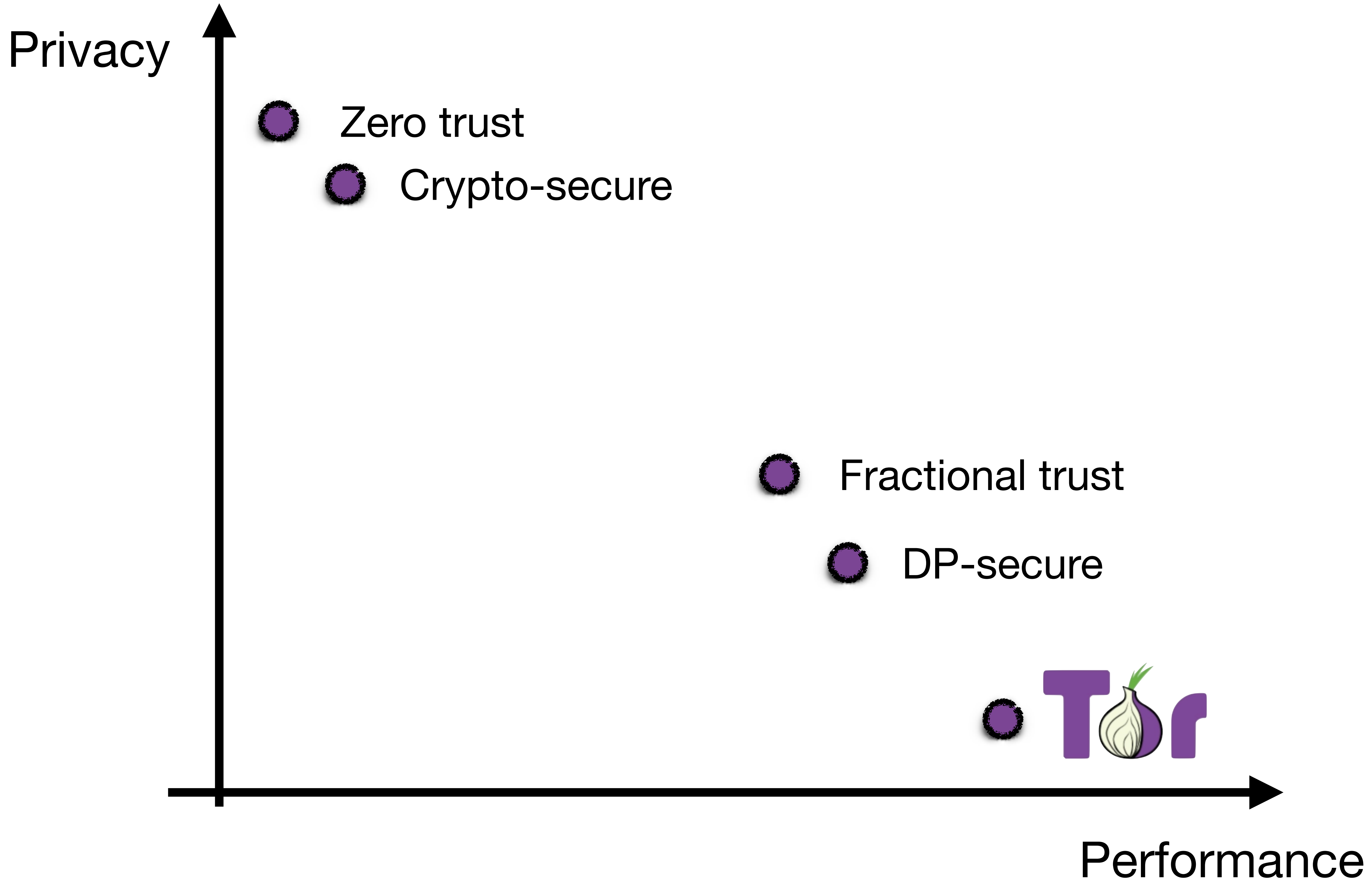
Oblivious write to (e.g., obfuscated shuffling, MPC…) / read from (e.g., PIR)

Fractional trust

DP-secure

Disguise the observable “drop” / “fetch” with DP padded noise

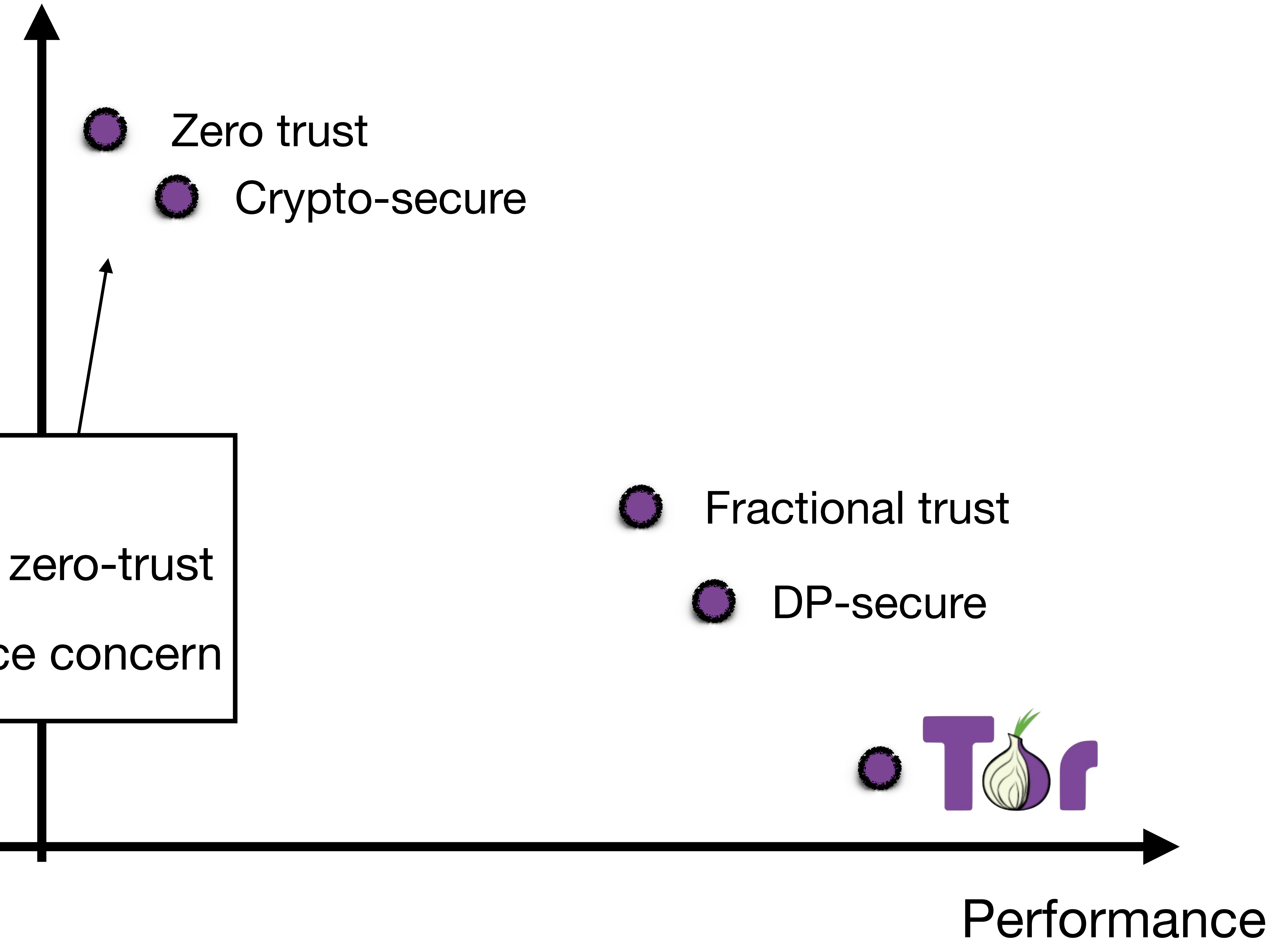
Motivation of Boomerang



Motivation of Boomerang

For whistleblowers

- Crypto secure & zero-trust
- Less performance concern



Privacy

Zero trust

Crypto-secure

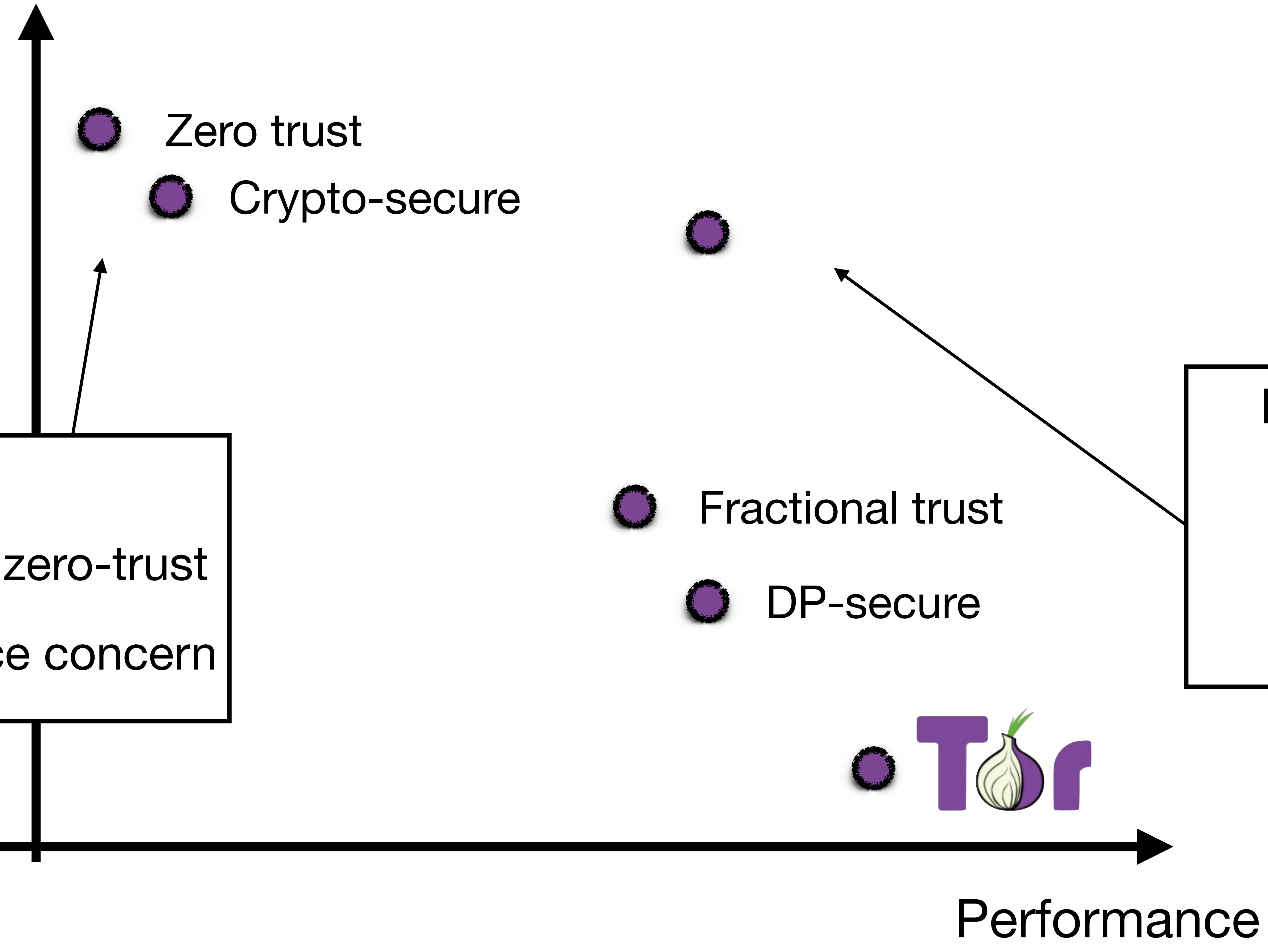
Fractional trust

DP-secure

Tor

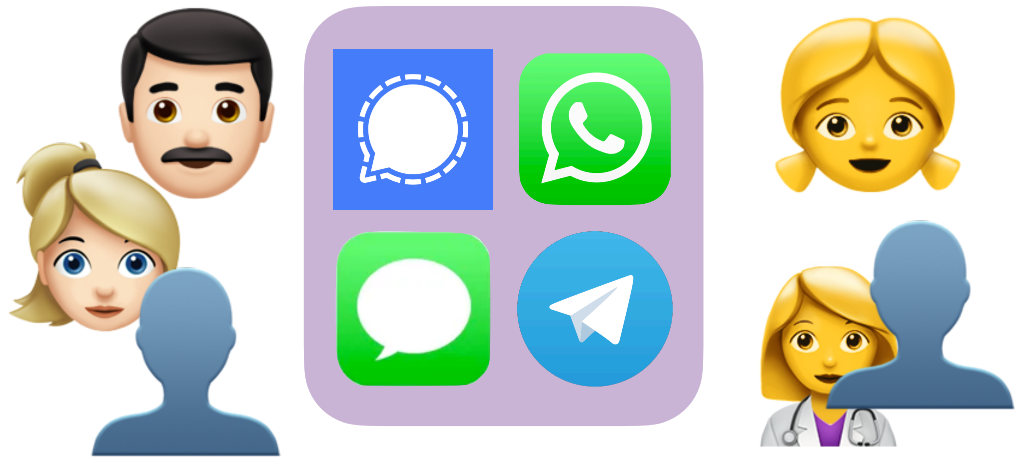
Performance

Motivation of Boomerang



For whistleblowers

- Crypto secure & zero-trust
- Less performance concern



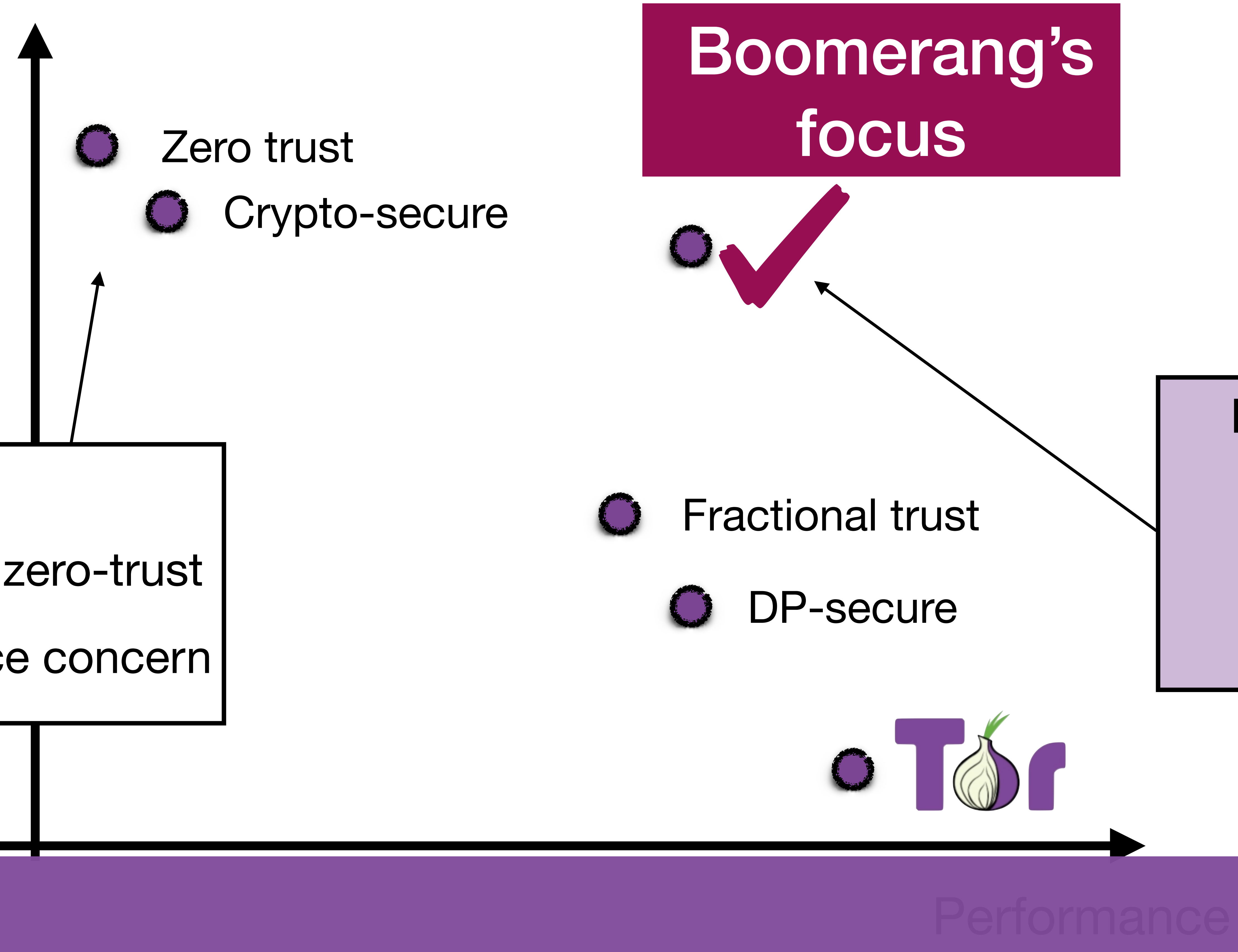
For mass general users

- More than E2EE messaging
- Lower operational cost and good performance

Motivation of Boomerang

For whistleblowers

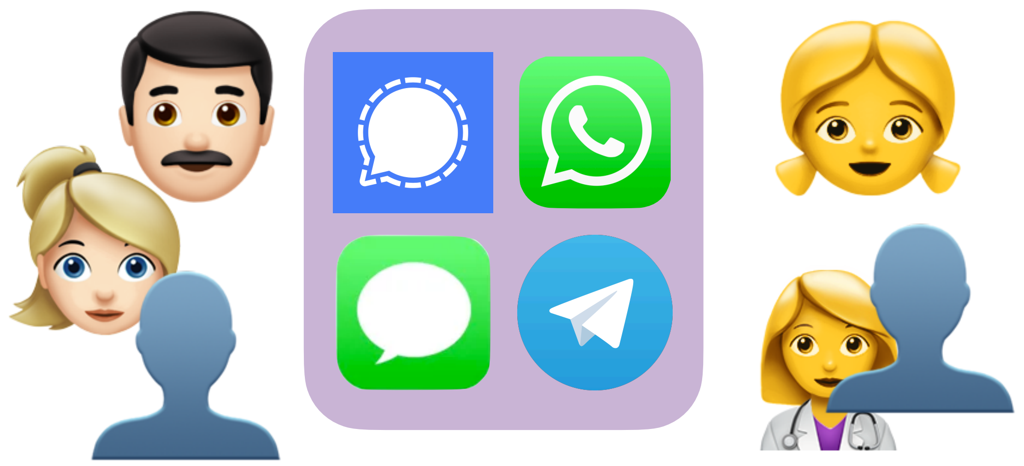
- Crypto secure & zero-trust
- Less performance concern



Boomerang's focus

For mass general users

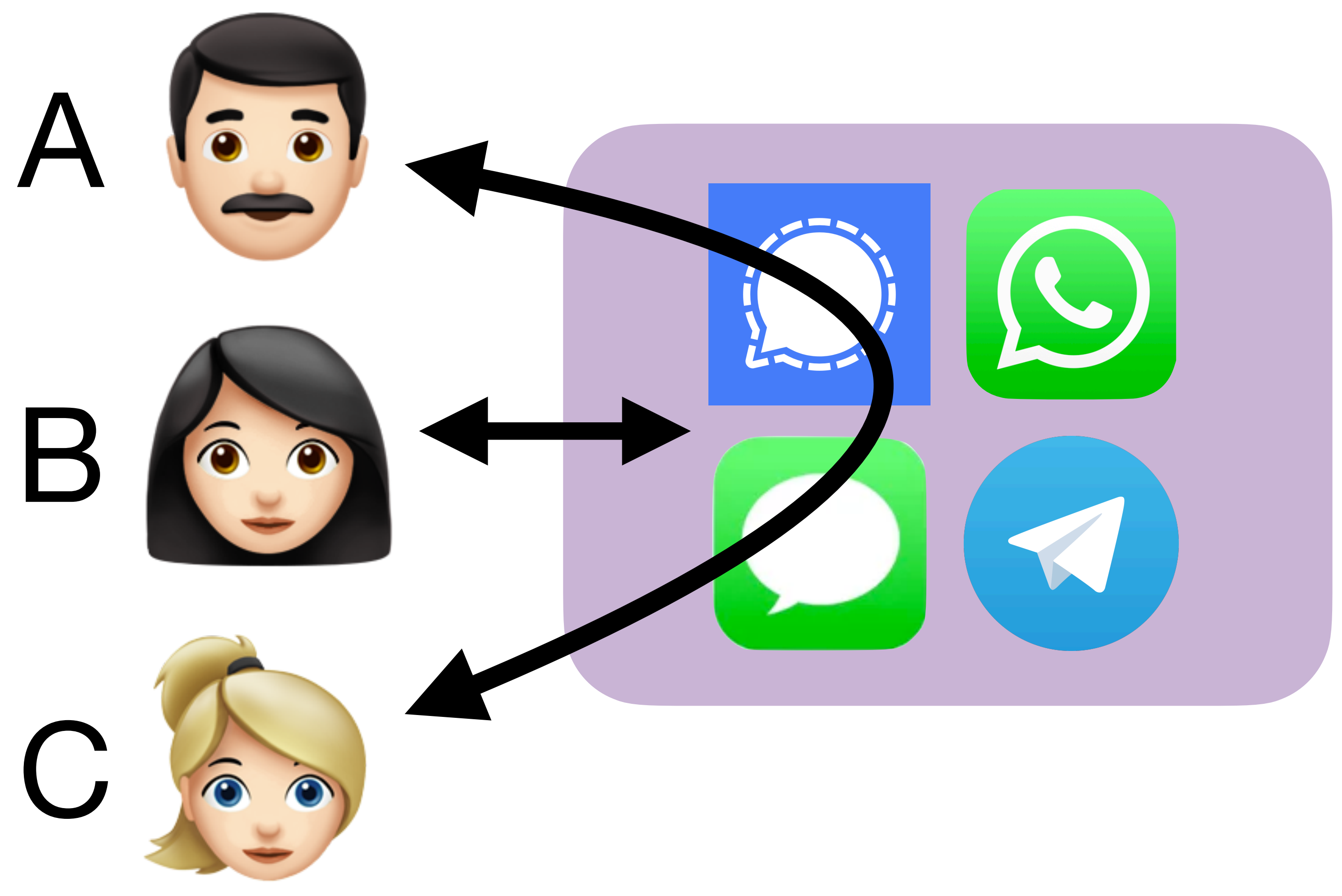
- More than E2EE messaging
- Lower operational cost and good performance



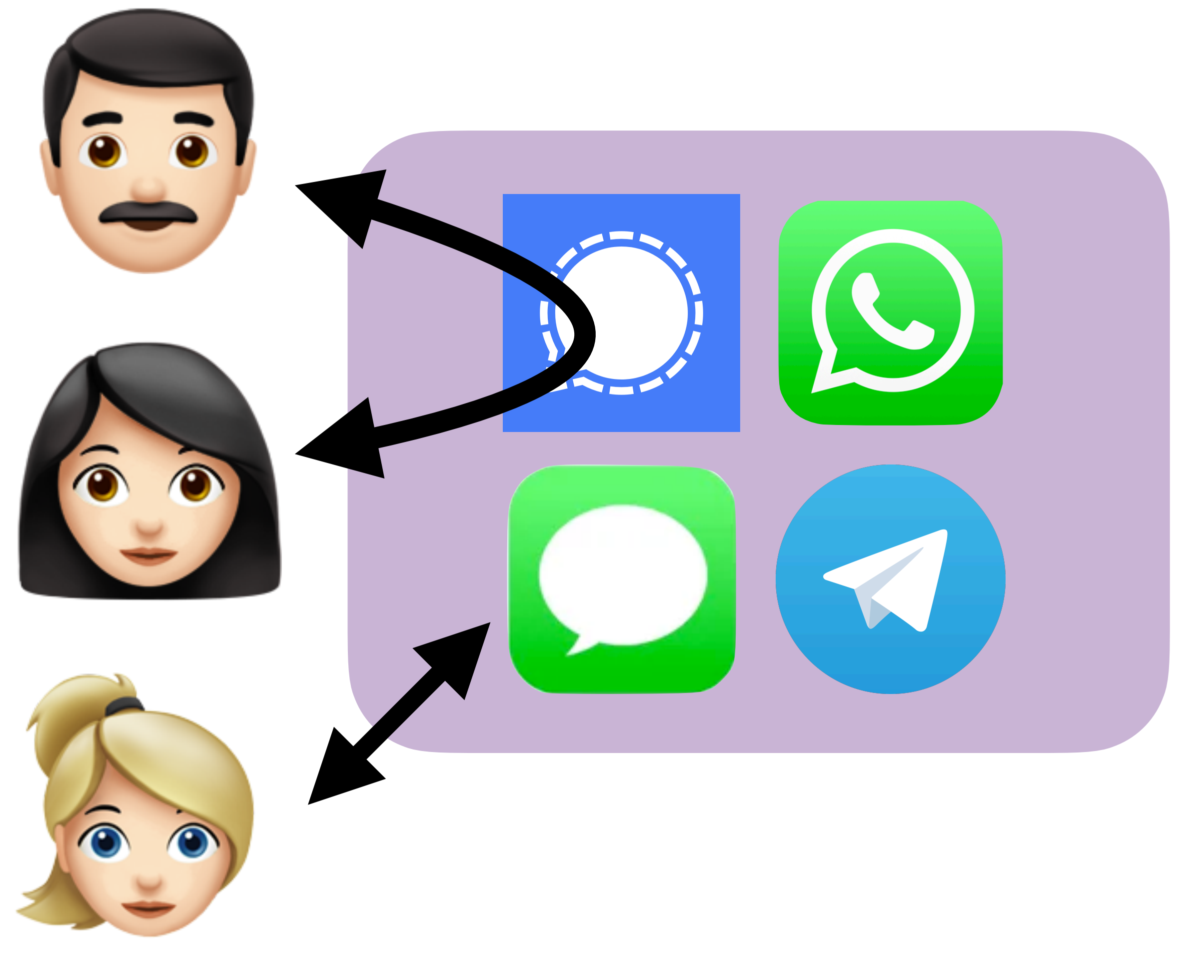
Boomerang: A performant system with cryptographic security under hardware trust

Metadata-private messaging: Goals & Threats

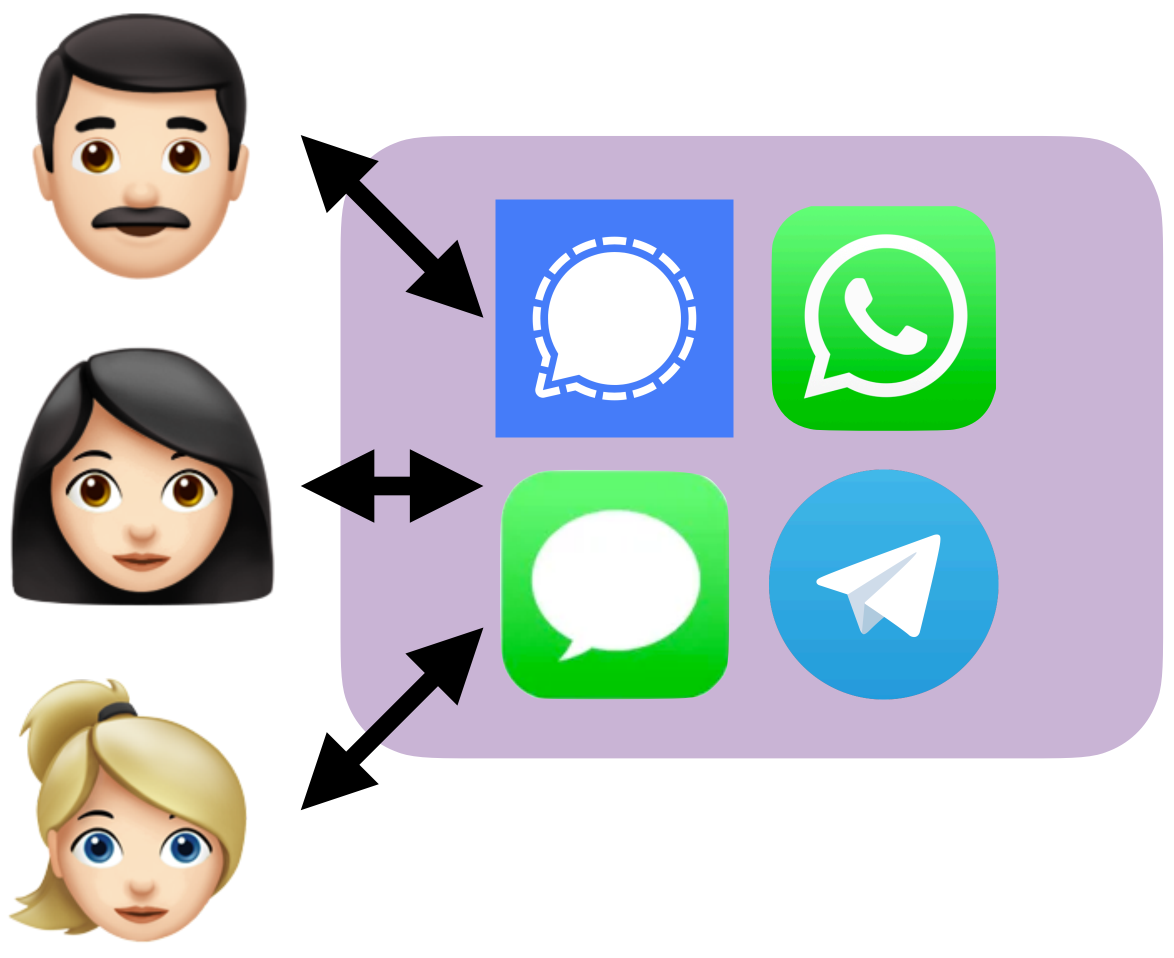
Goal: Hiding who is talking to whom



A talks to C



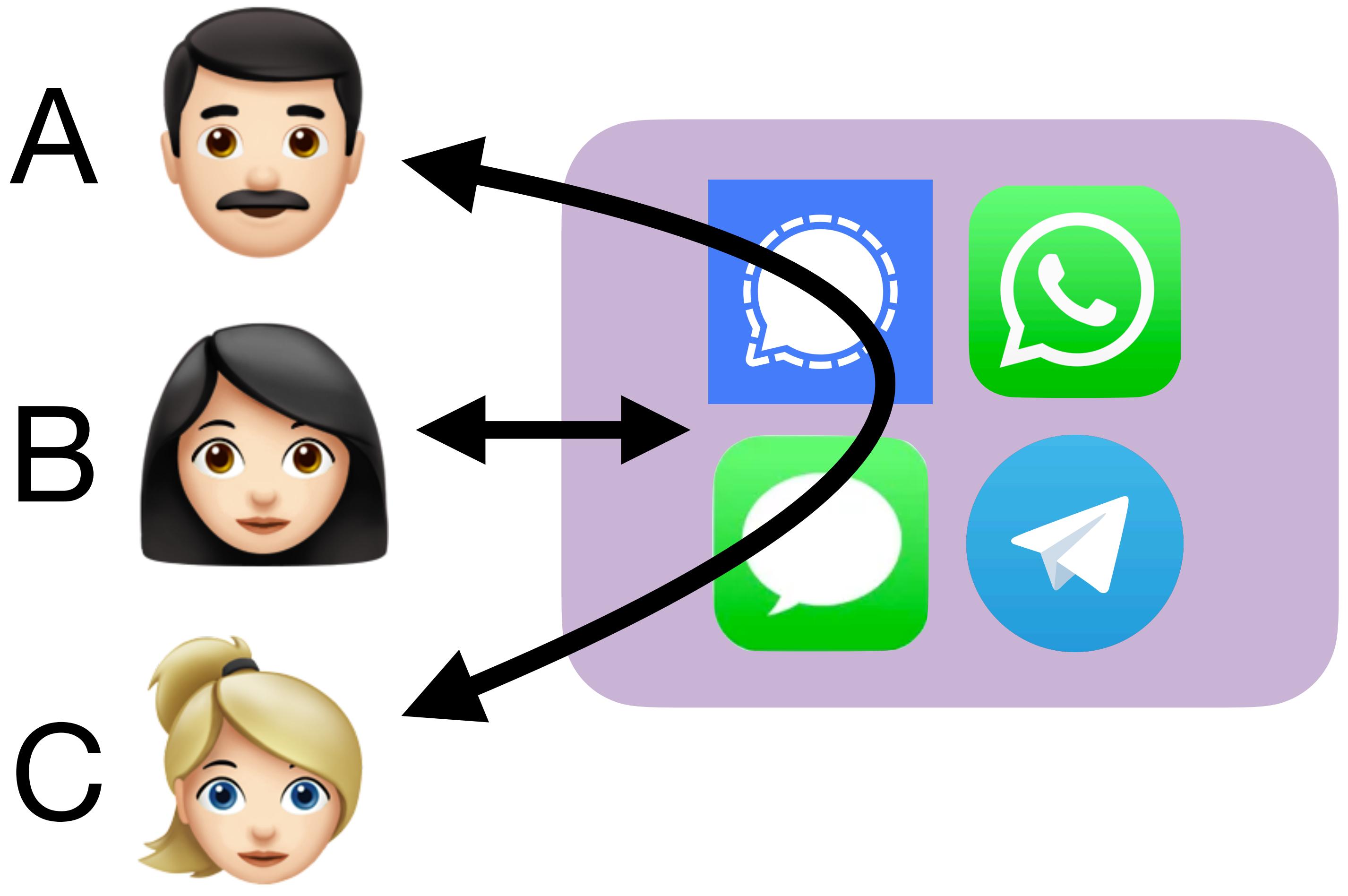
A talks to B



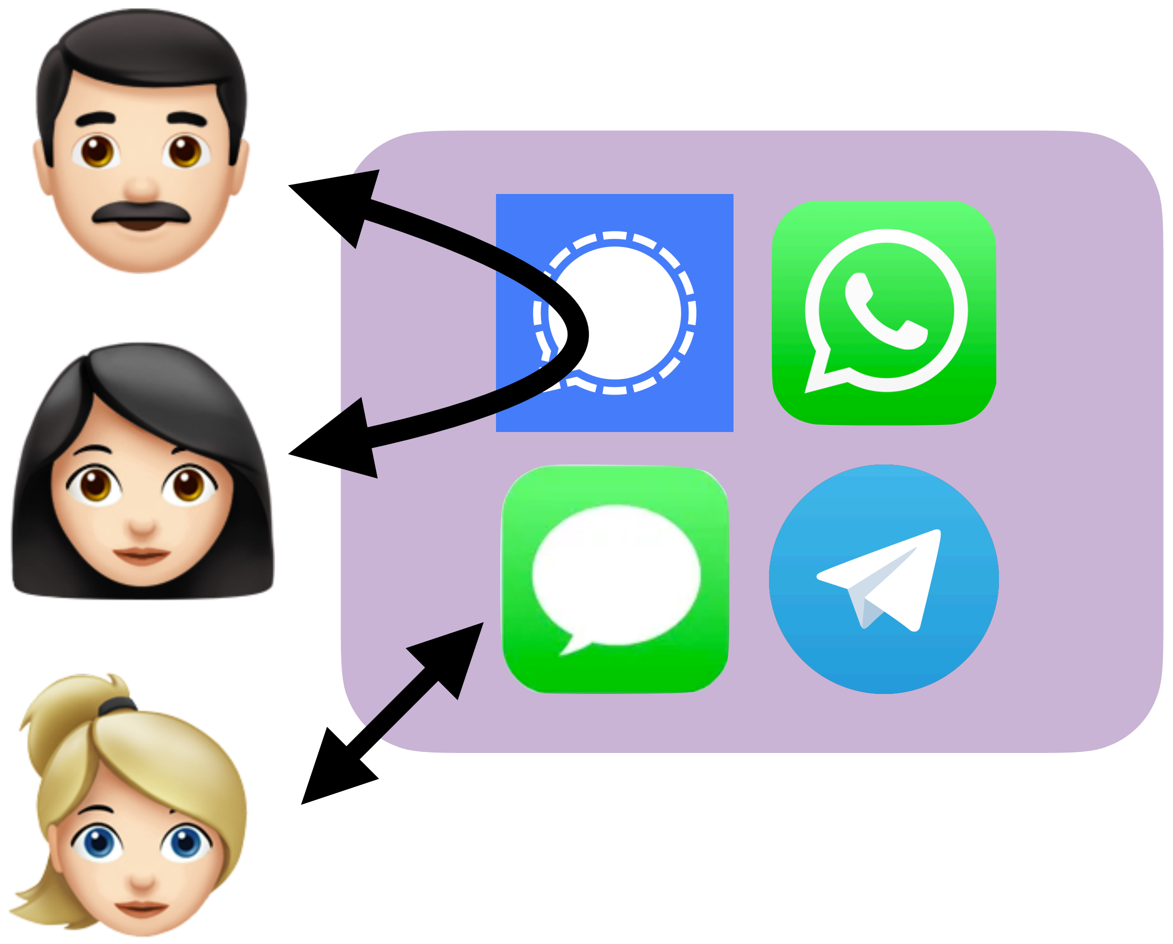
A talks to nobody

Metadata-private messaging: Goals & Threats

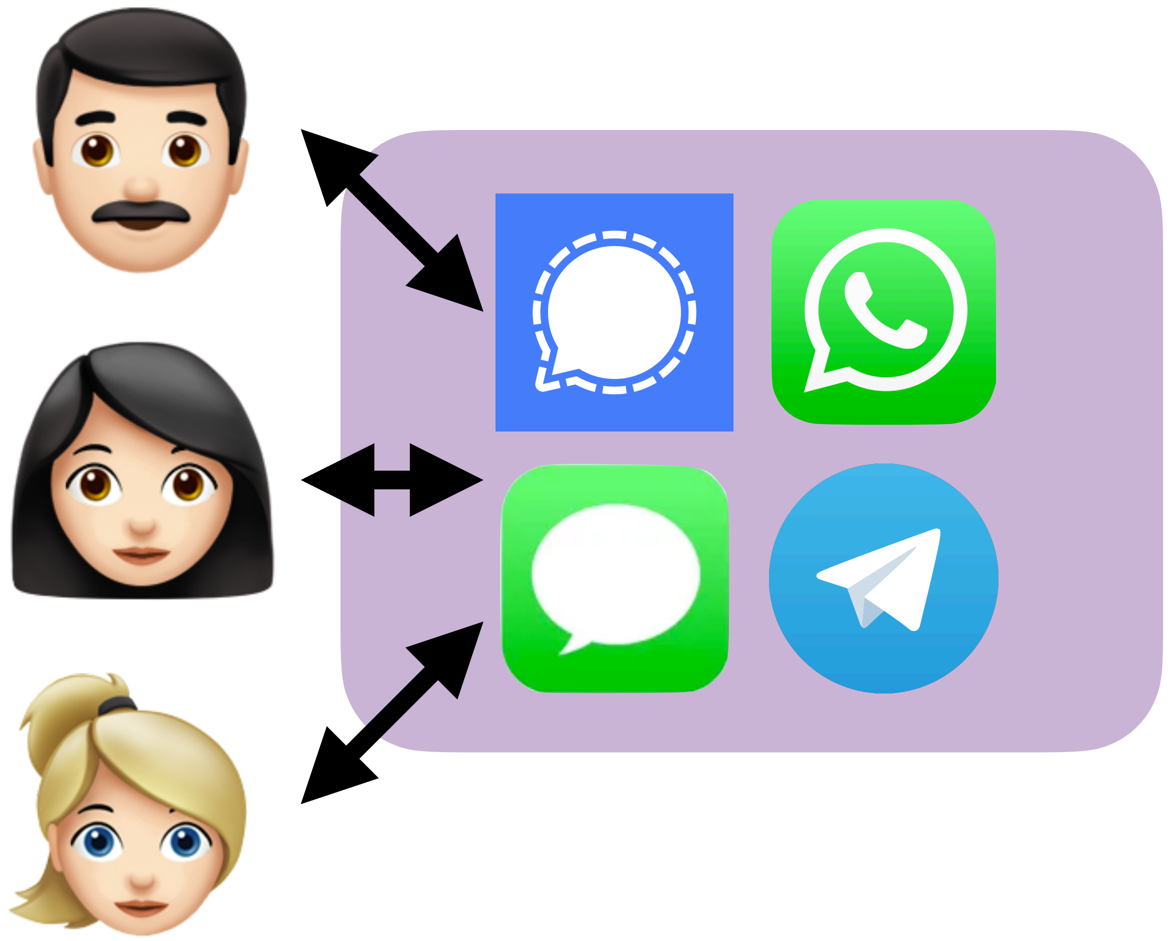
Goal: Hiding who is talking to whom



A talks to C



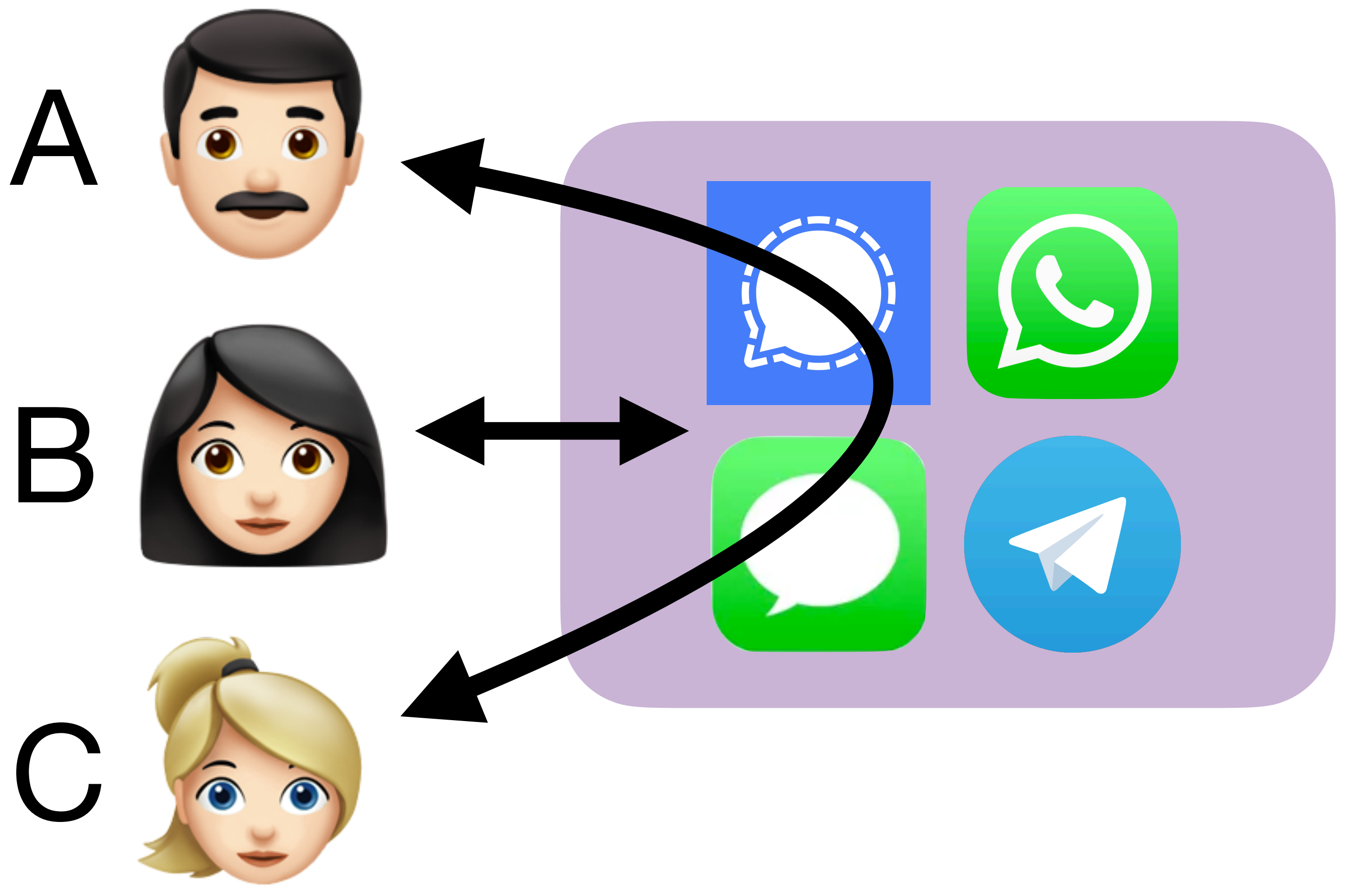
A talks to B



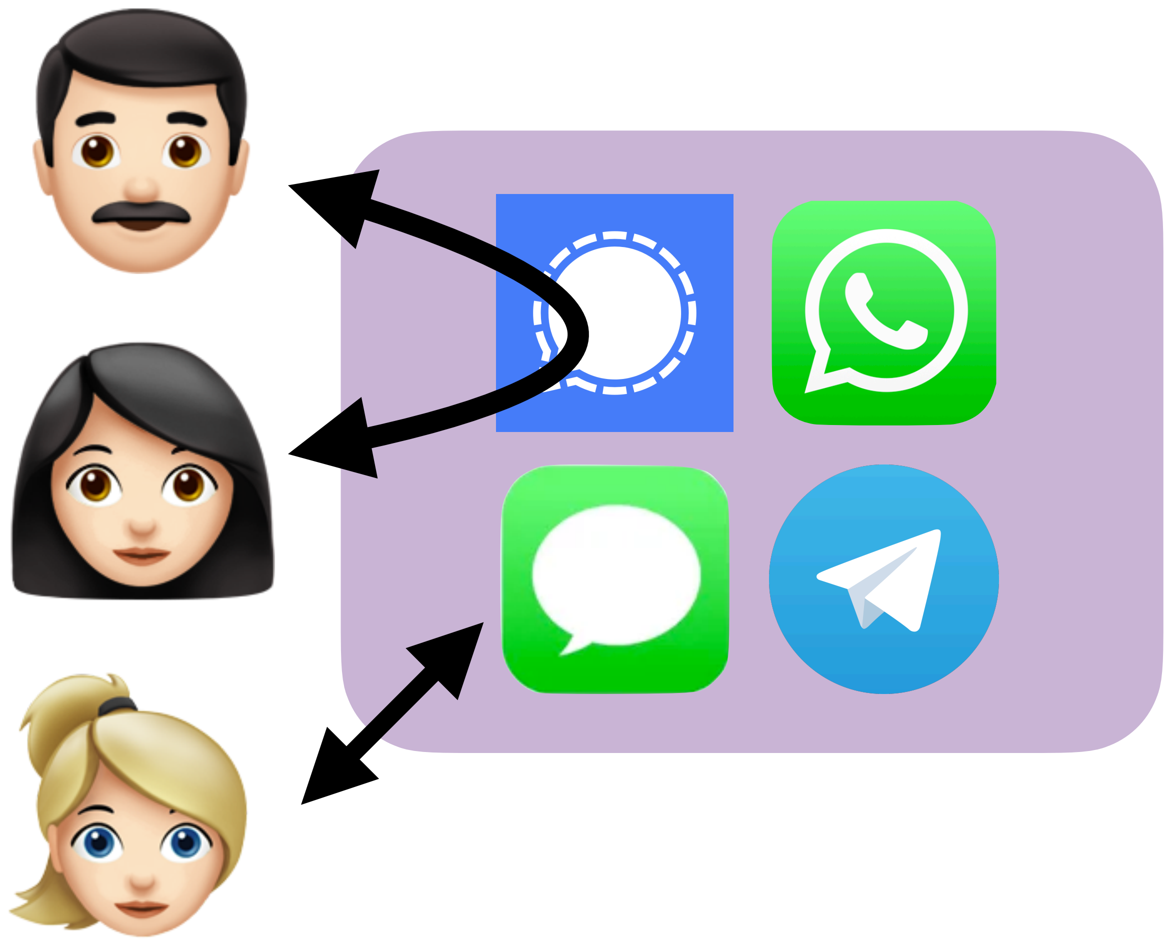
A talks to nobody

Metadata-private messaging: Goals & Threats

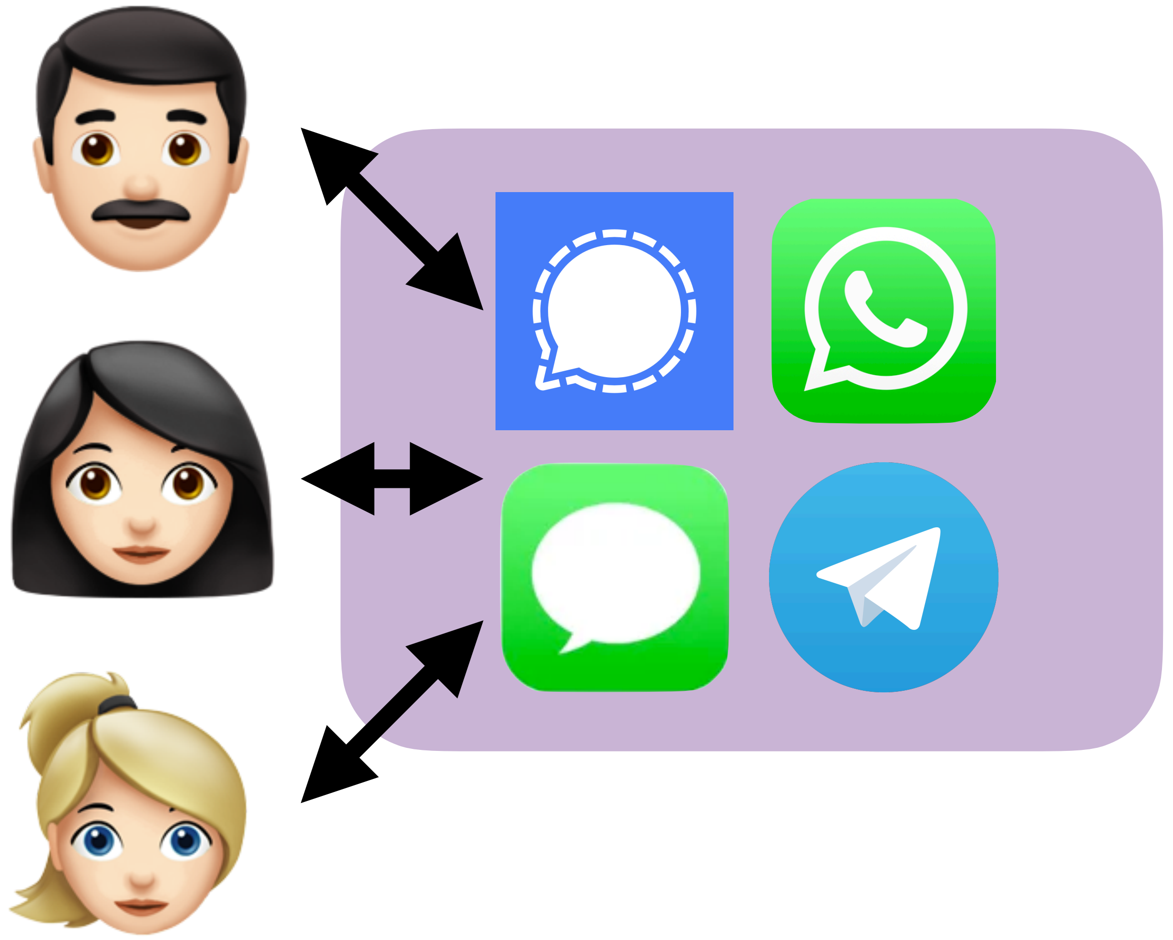
Goal: Hiding who is talking to whom



A talks to C



A talks to B

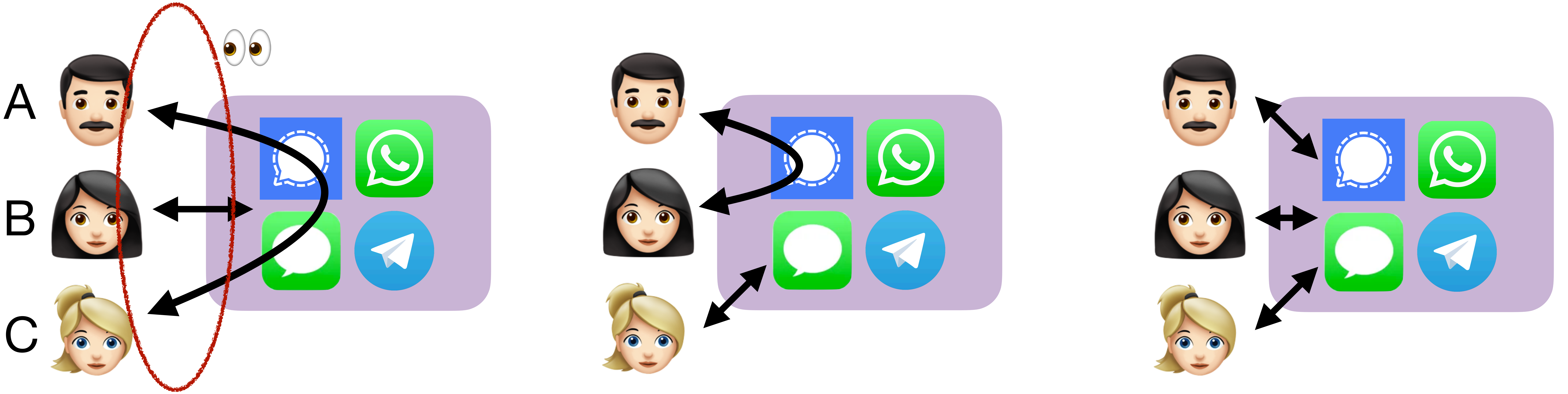


A talks to nobody

Passive attackers

Metadata-private messaging: Goals & Threats

Goal: Hiding who is talking to whom



A talks to C

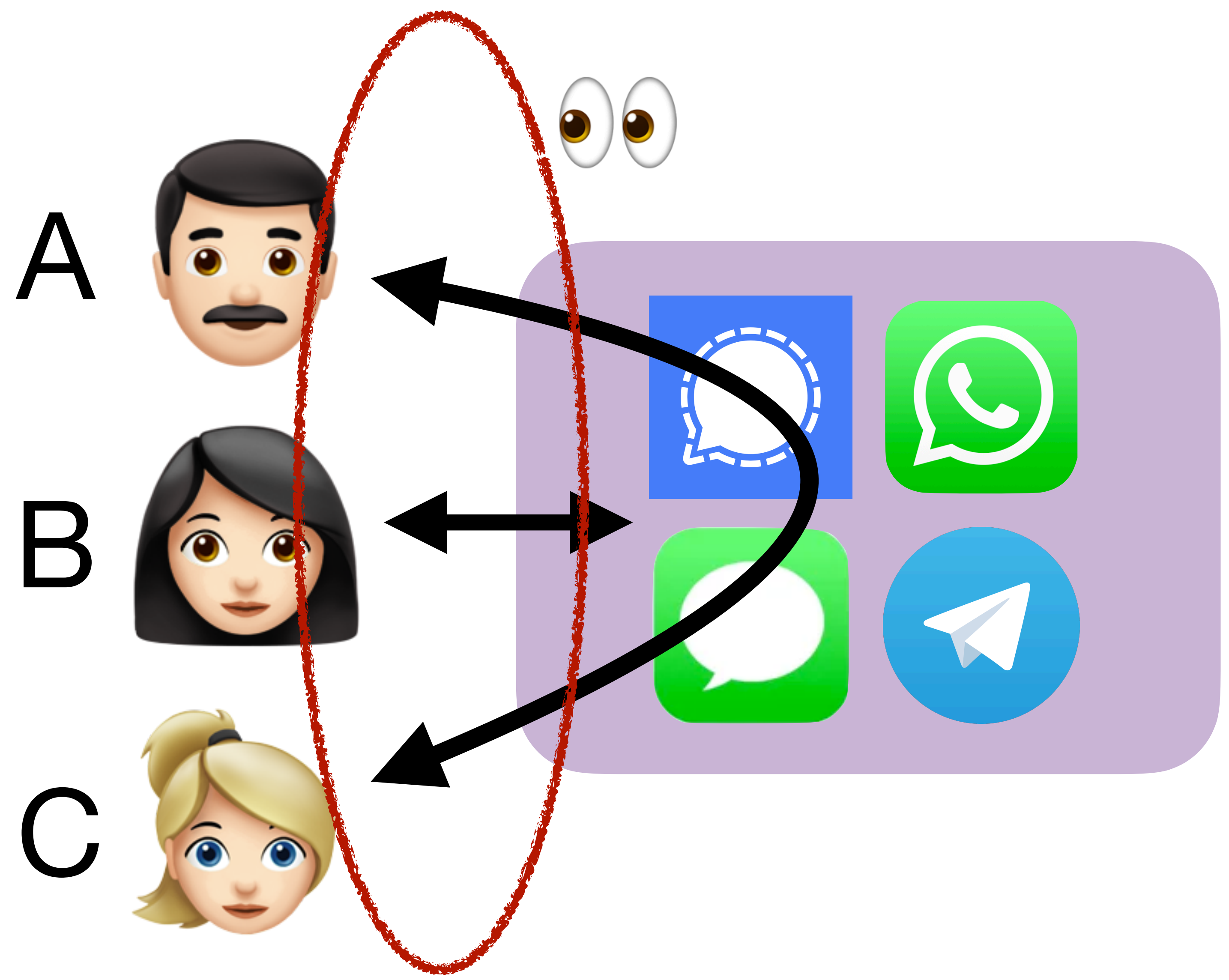
A talks to B

A talks to nobody

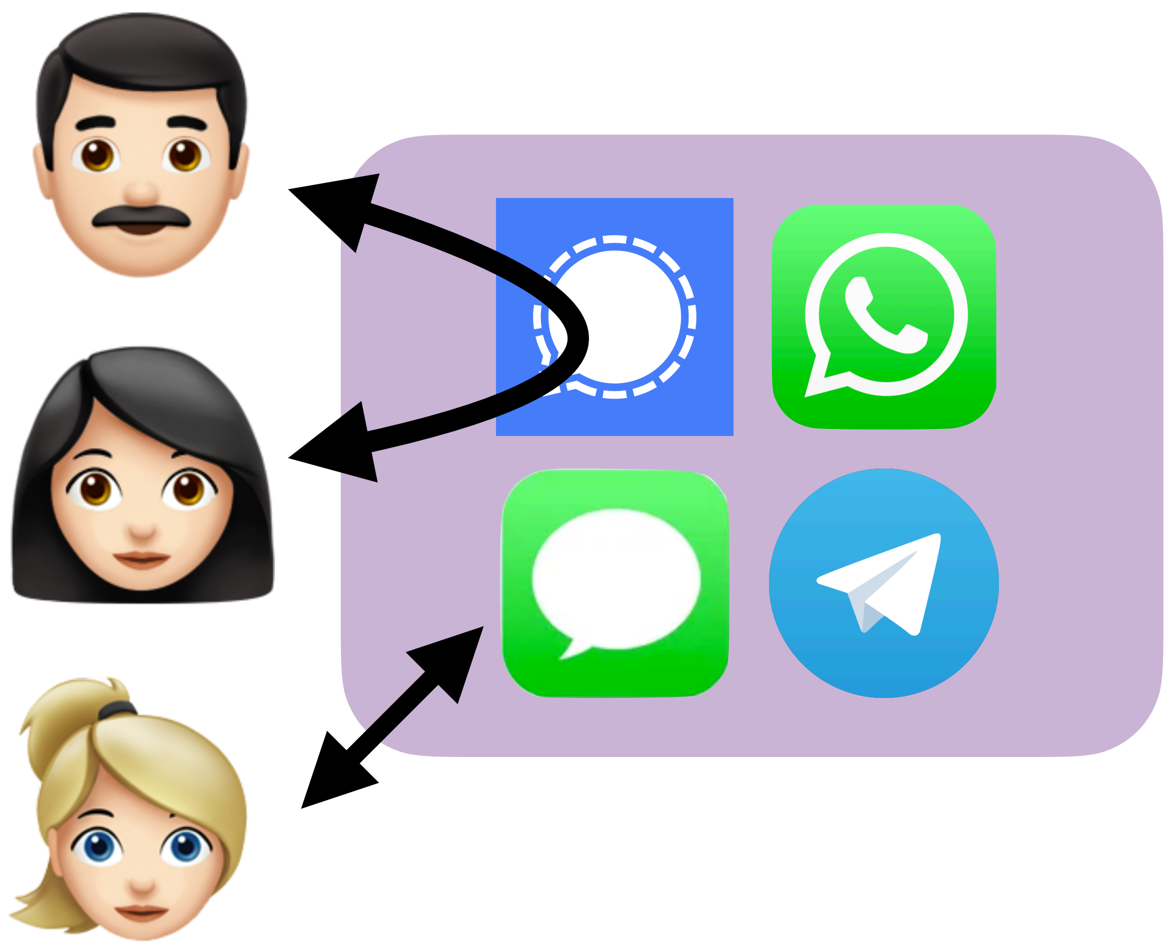
Passive attackers

Metadata-private messaging: Goals & Threats

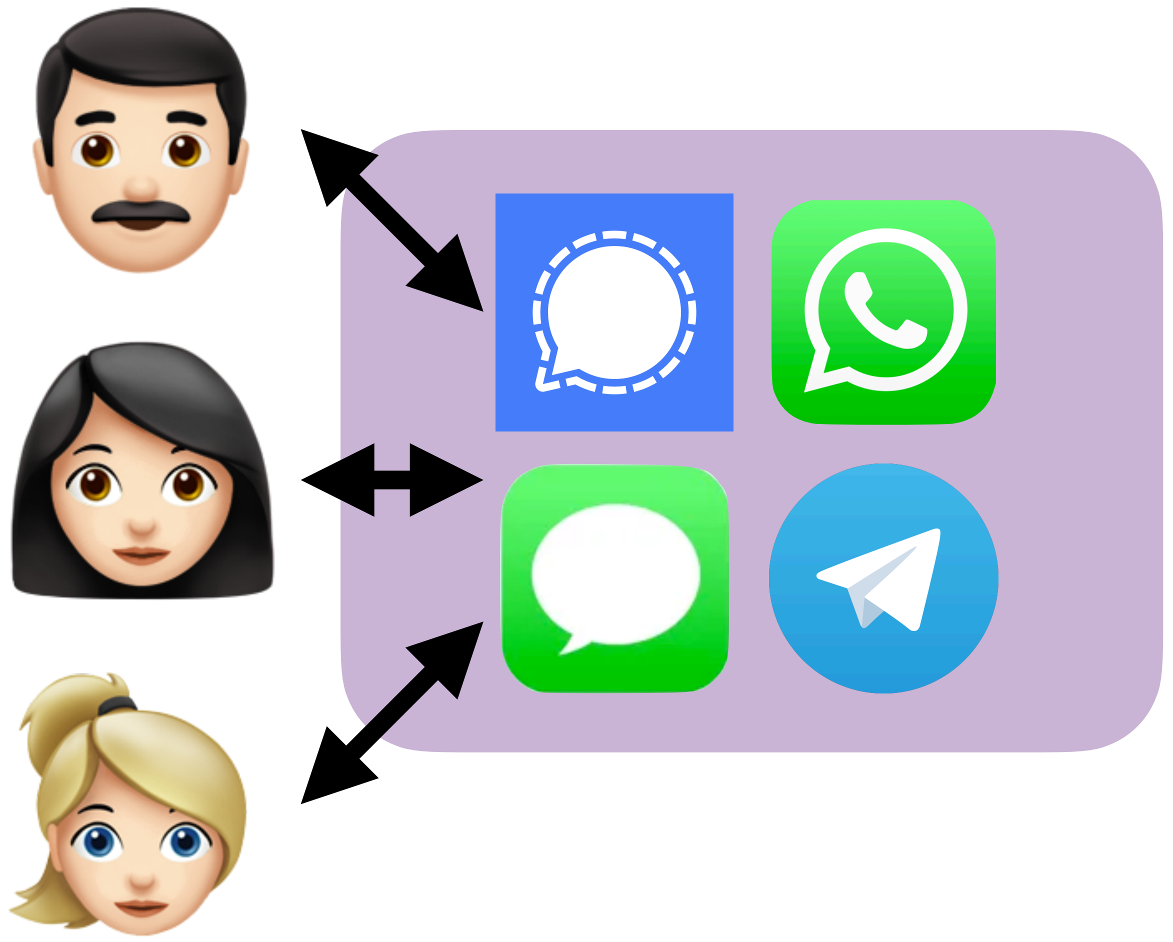
Goal: Hiding who is talking to whom



A talks to C



A talks to B



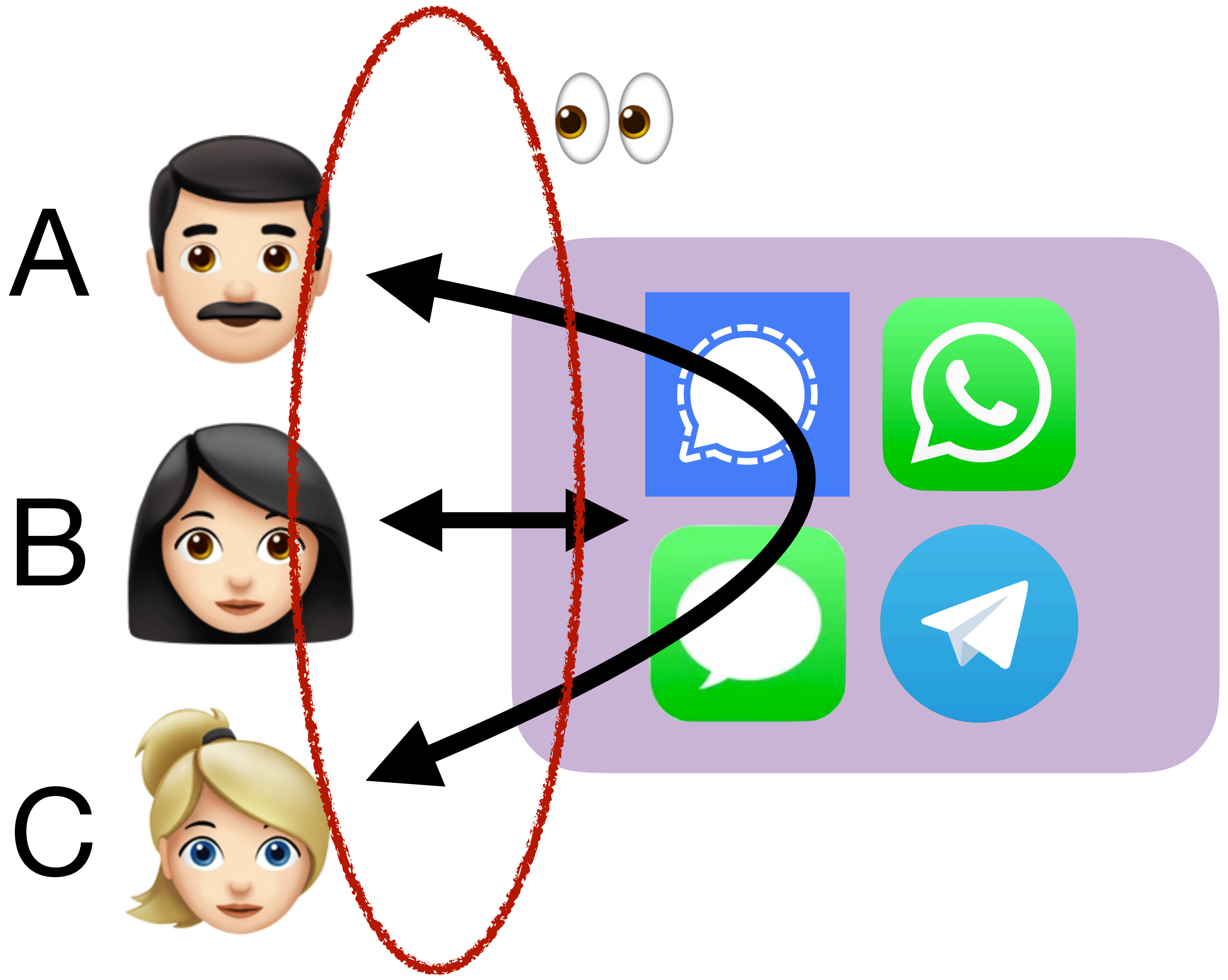
A talks to nobody

Passive attackers

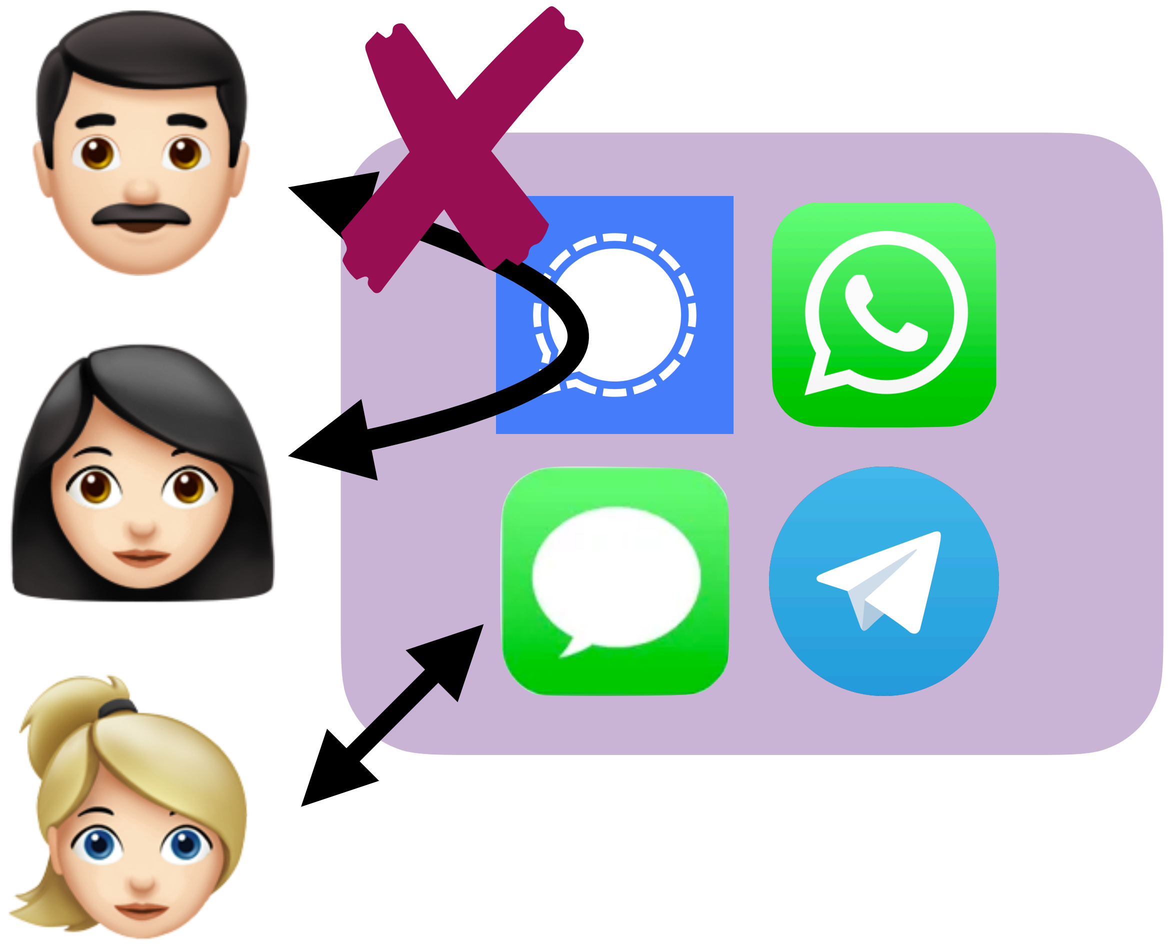
Active attackers

Metadata-private messaging: Goals & Threats

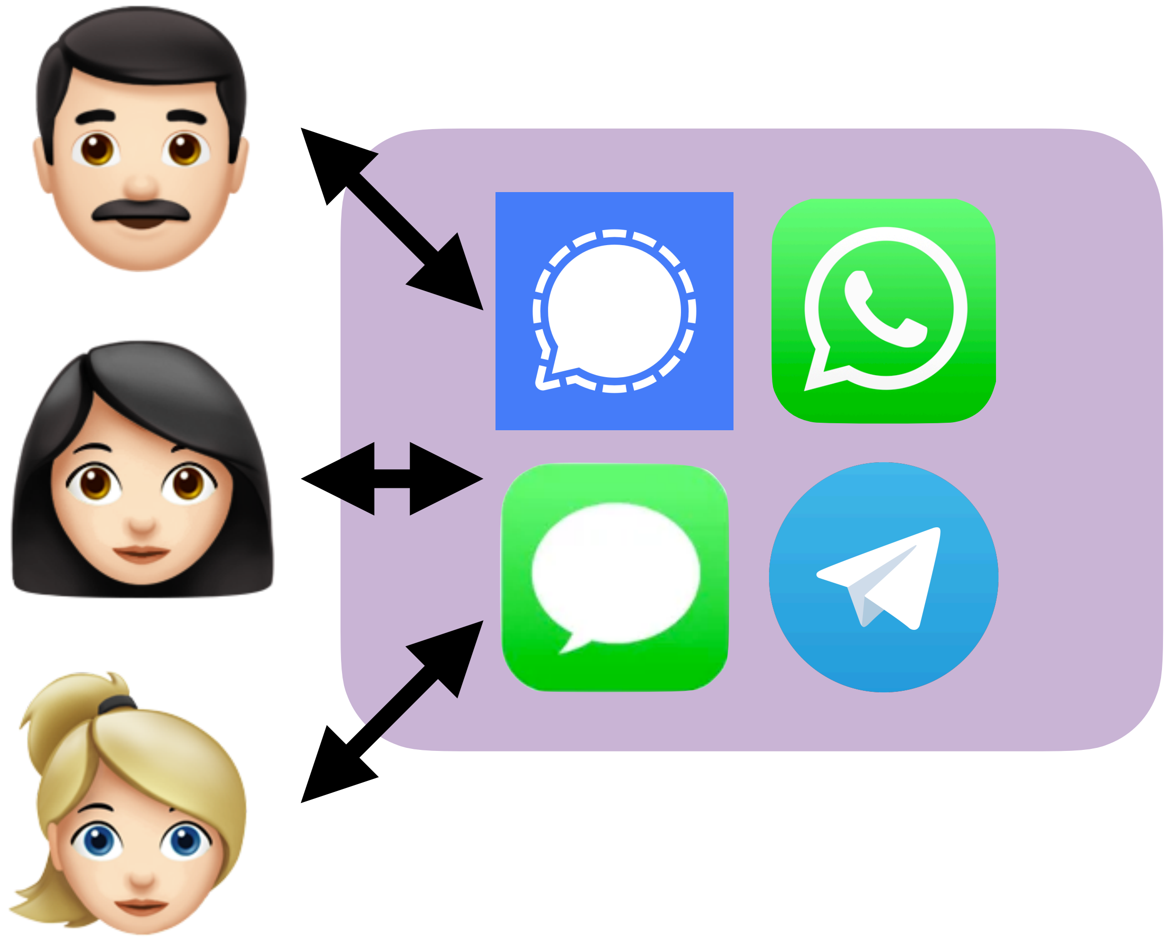
Goal: Hiding who is talking to whom



A talks to C



A talks to B



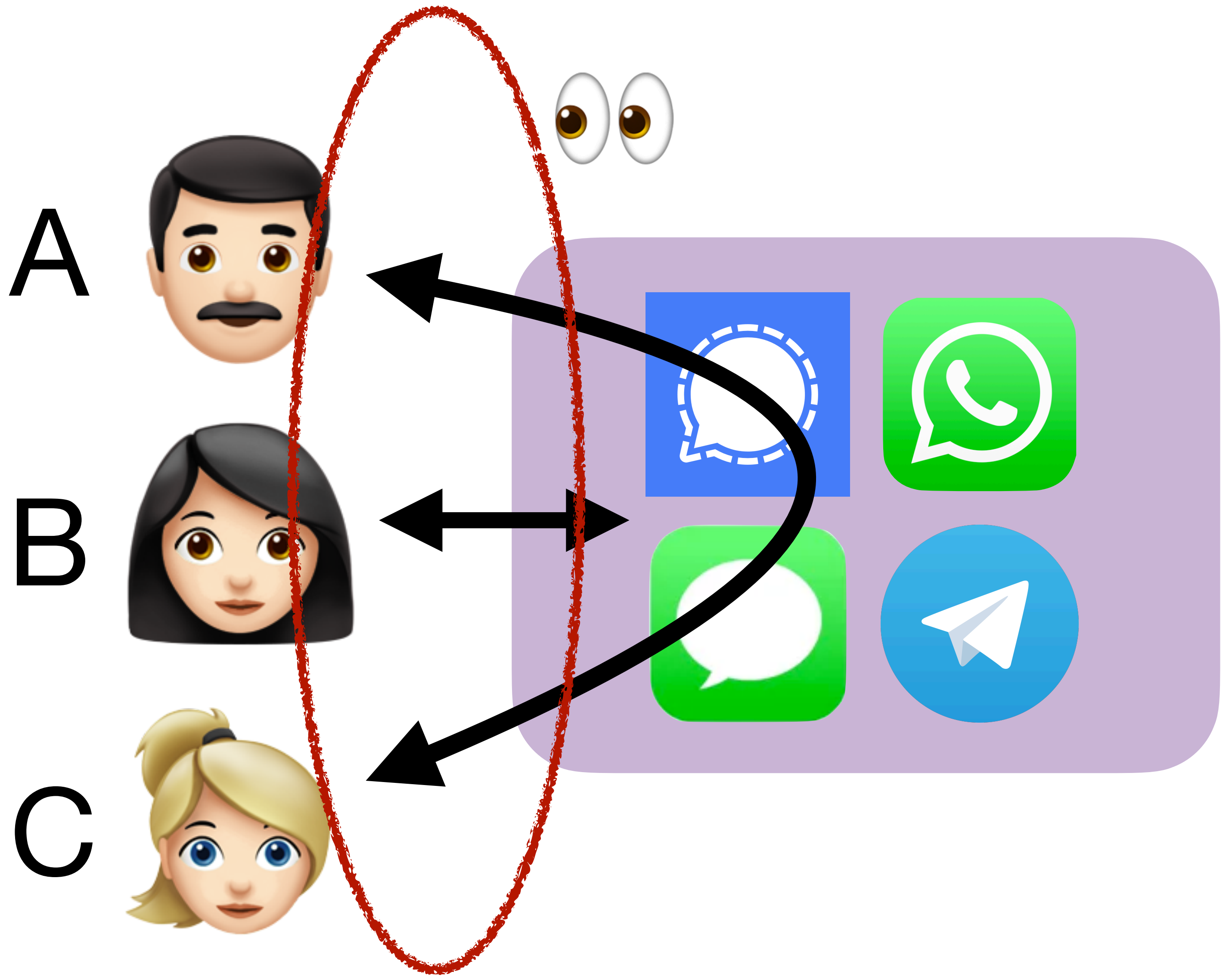
A talks to nobody

Passive attackers

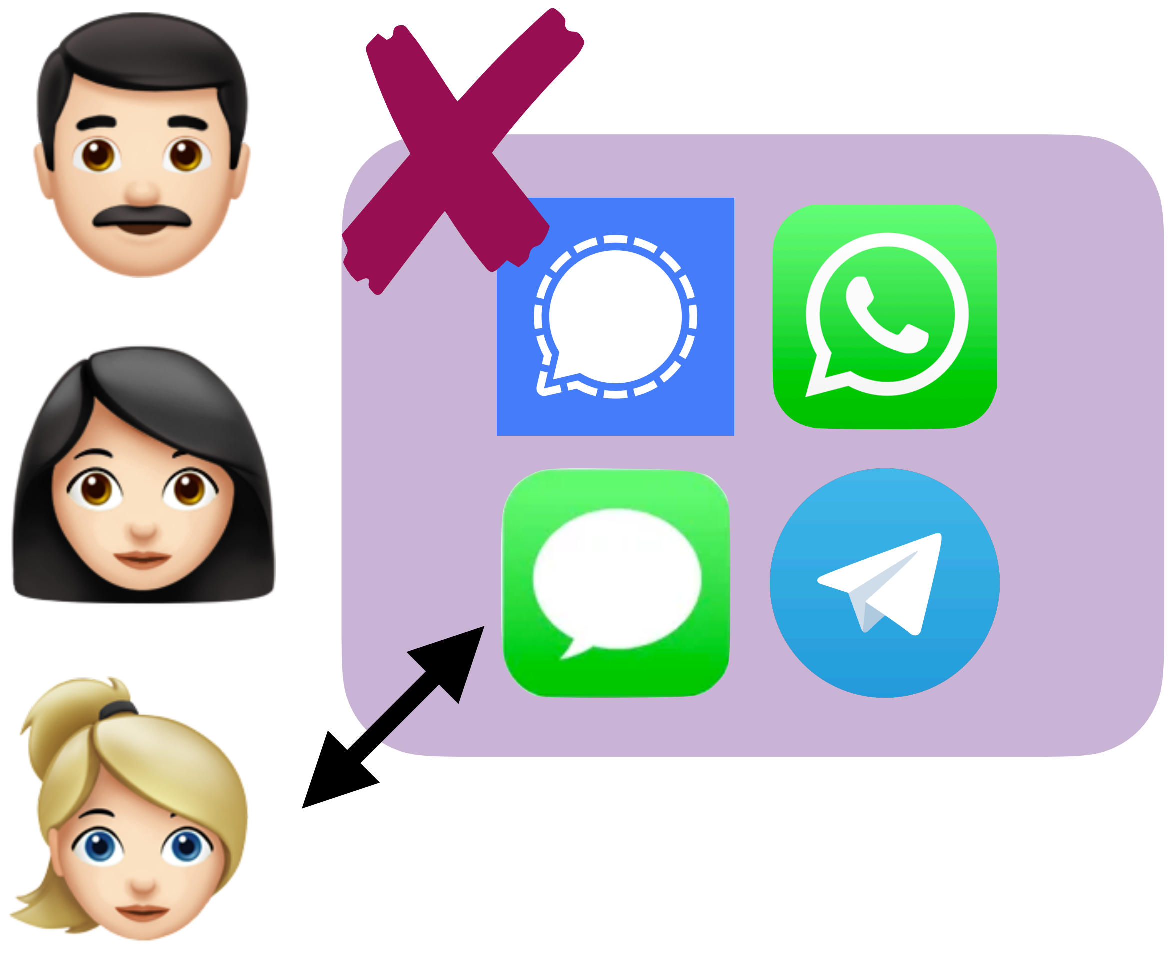
Active attackers

Metadata-private messaging: Goals & Threats

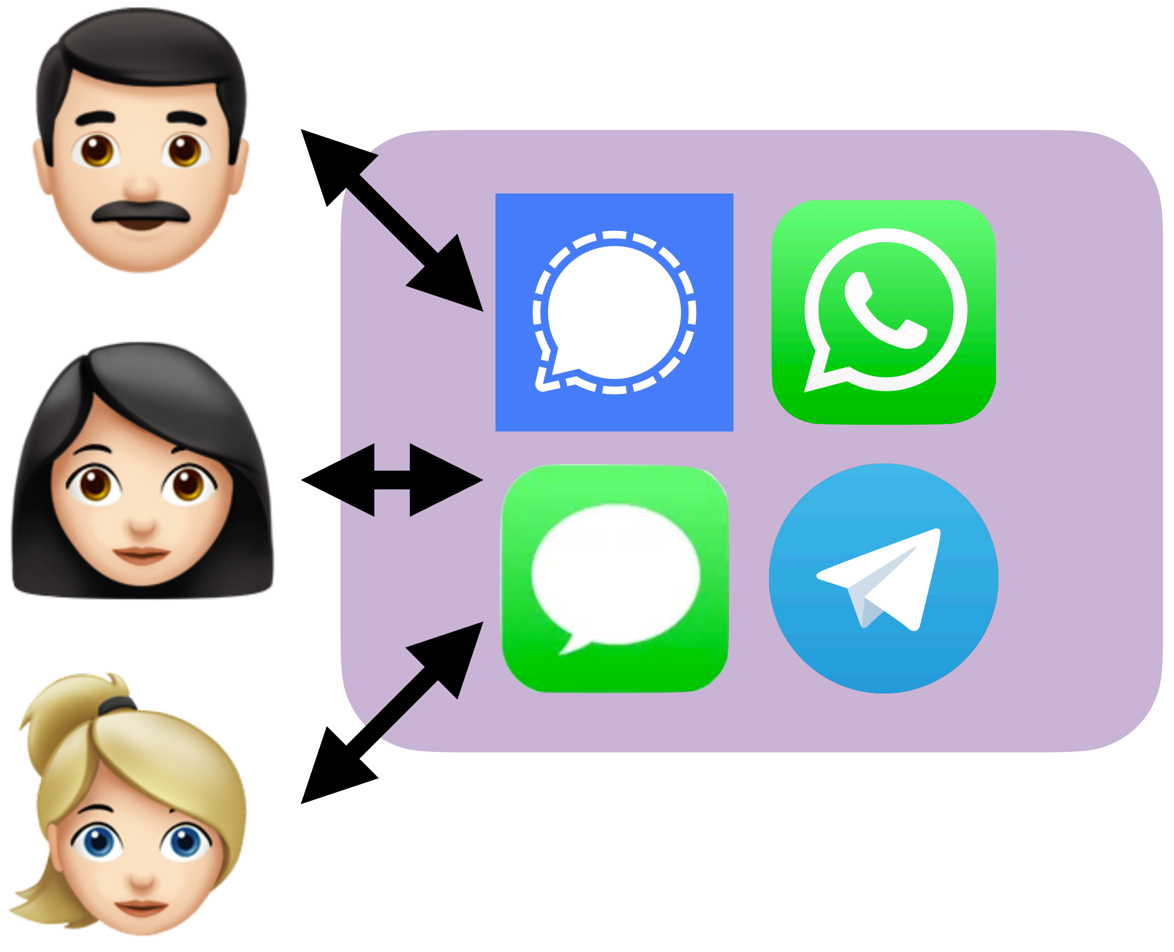
Goal: Hiding who is talking to whom



A talks to C



A talks to B



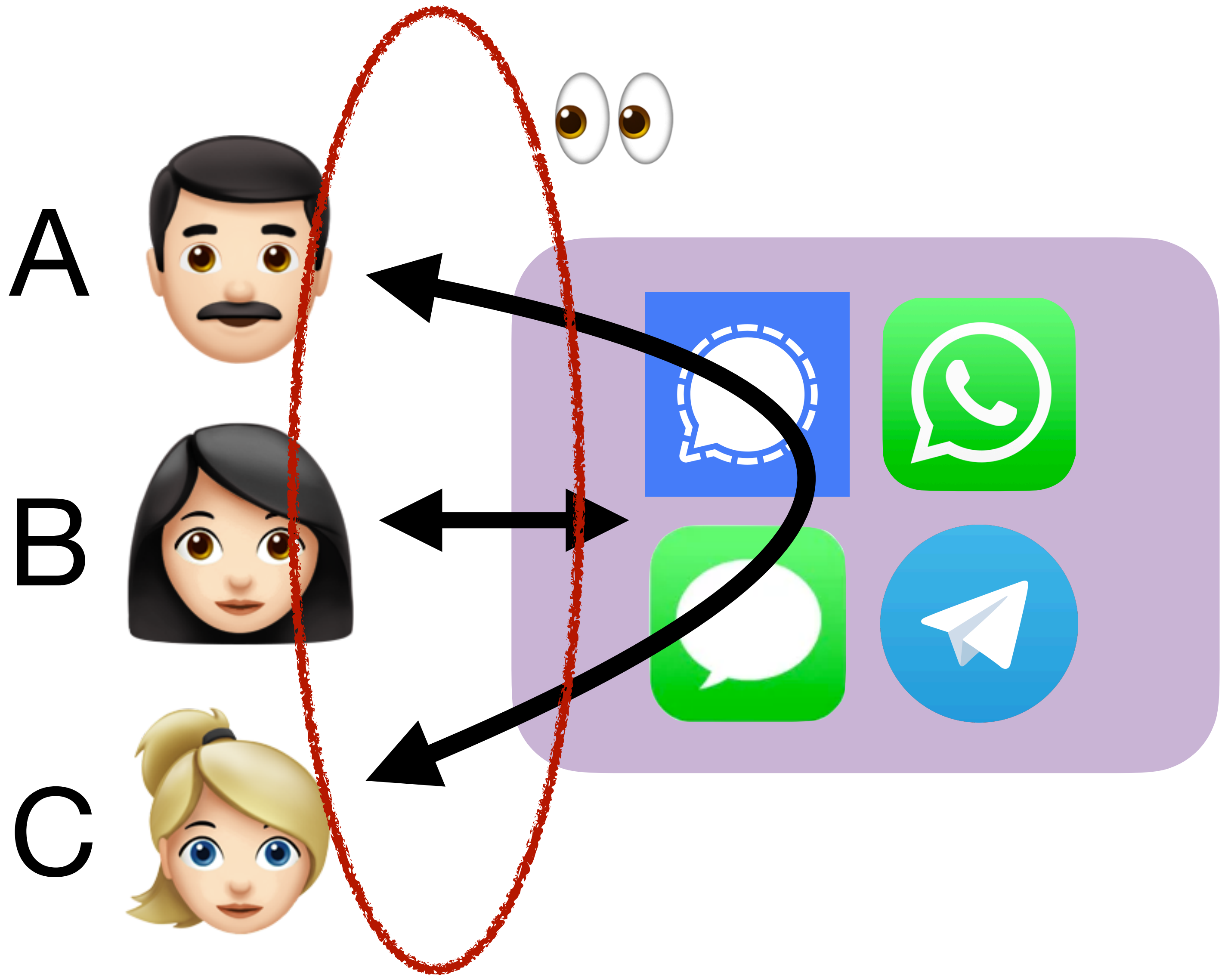
A talks to nobody

Passive attackers

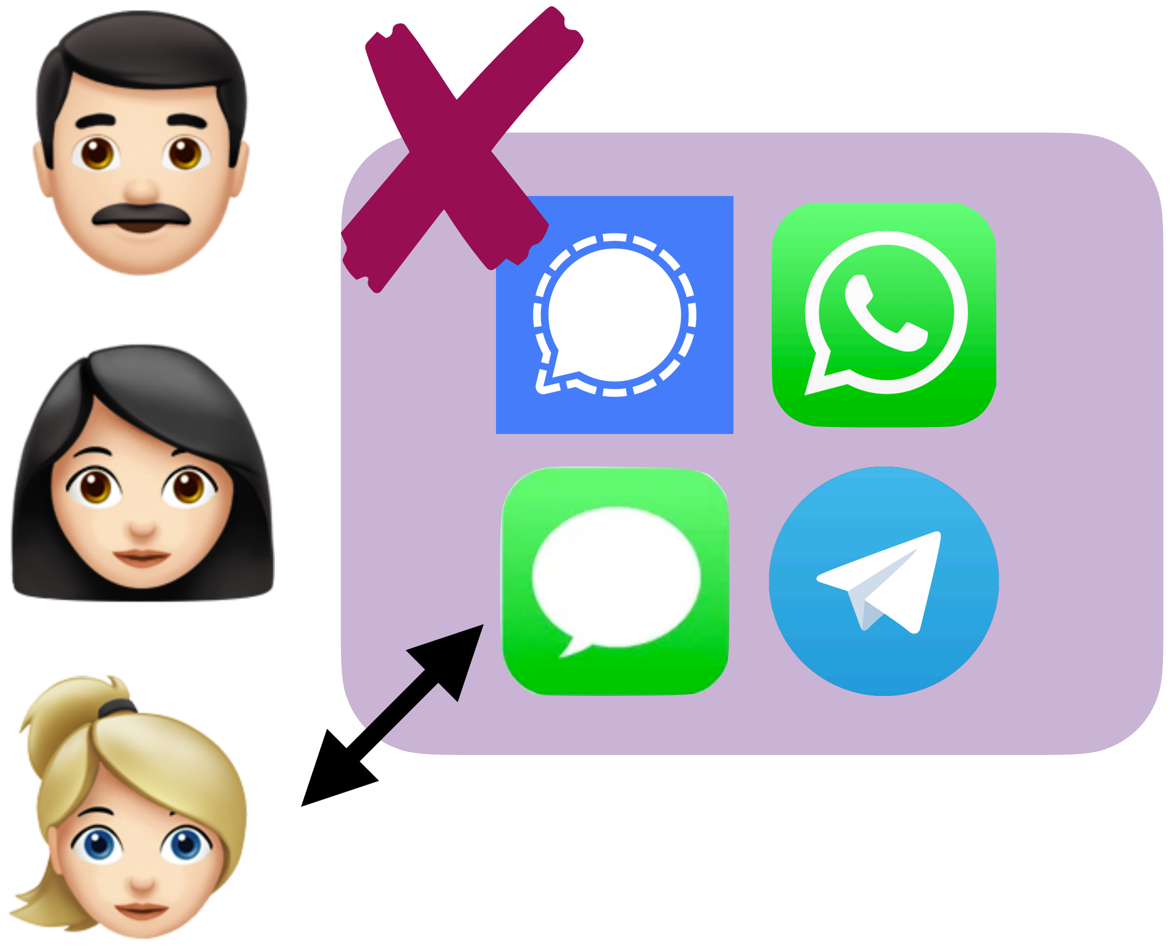
Active attackers

Metadata-private messaging: Goals & Threats

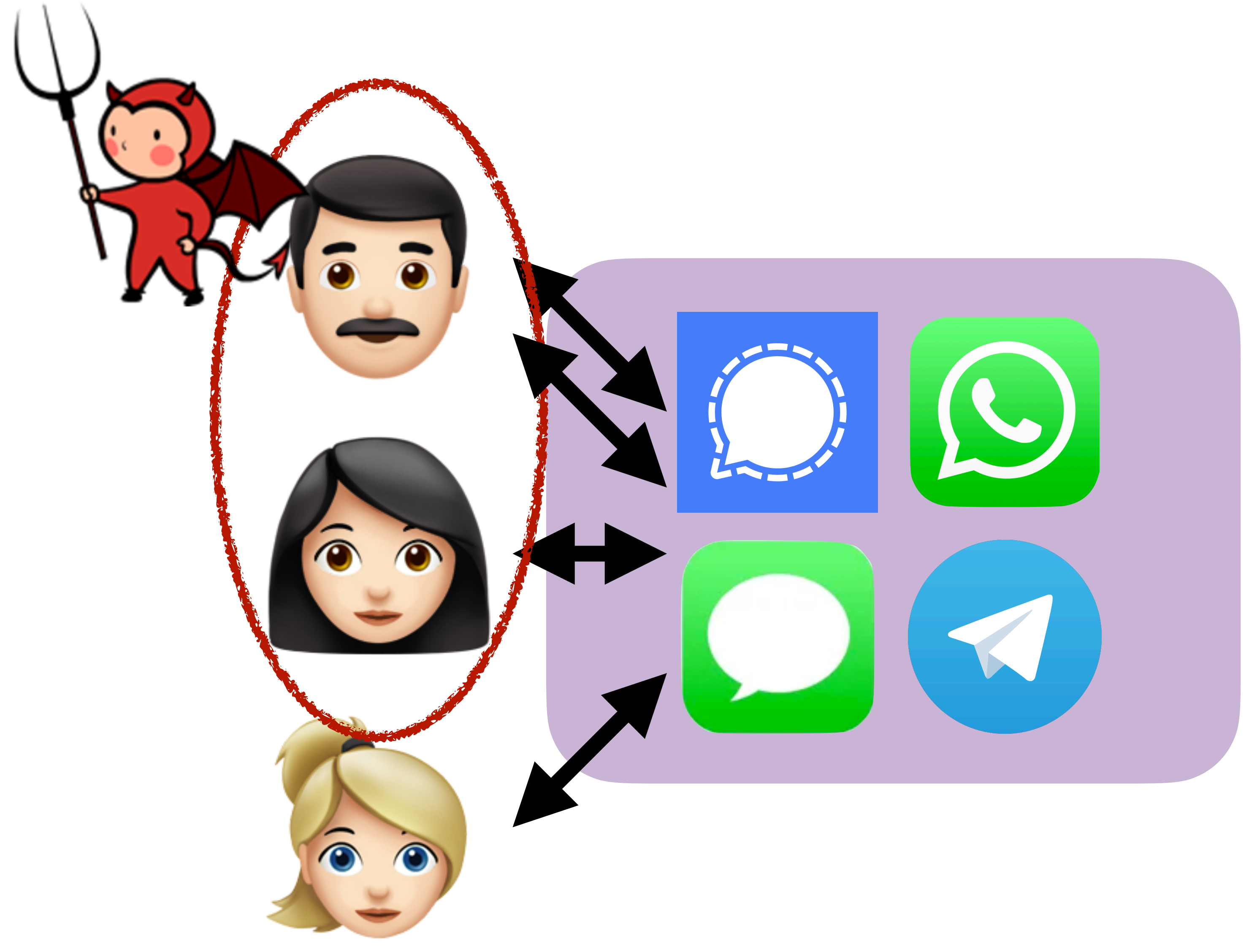
Goal: Hiding who is talking to whom



A talks to C



A talks to B



A talks to nobody

Passive attackers

Active attackers

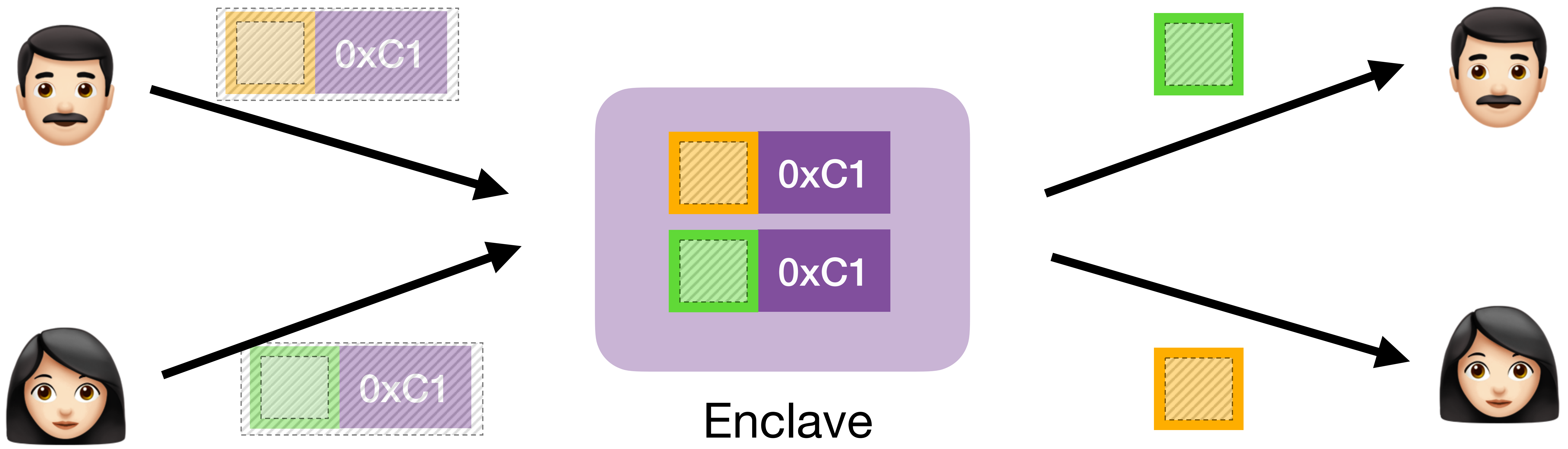
What are the challenges?

- Enclaves have unique threat models and attack surfaces
  - Memory access pattern protection
- Powerful attackers
  - Actively interfere with traffic and/or control a subset of clients
- Scalability as a security demand
  - Privacy loves company (more clients are always better)

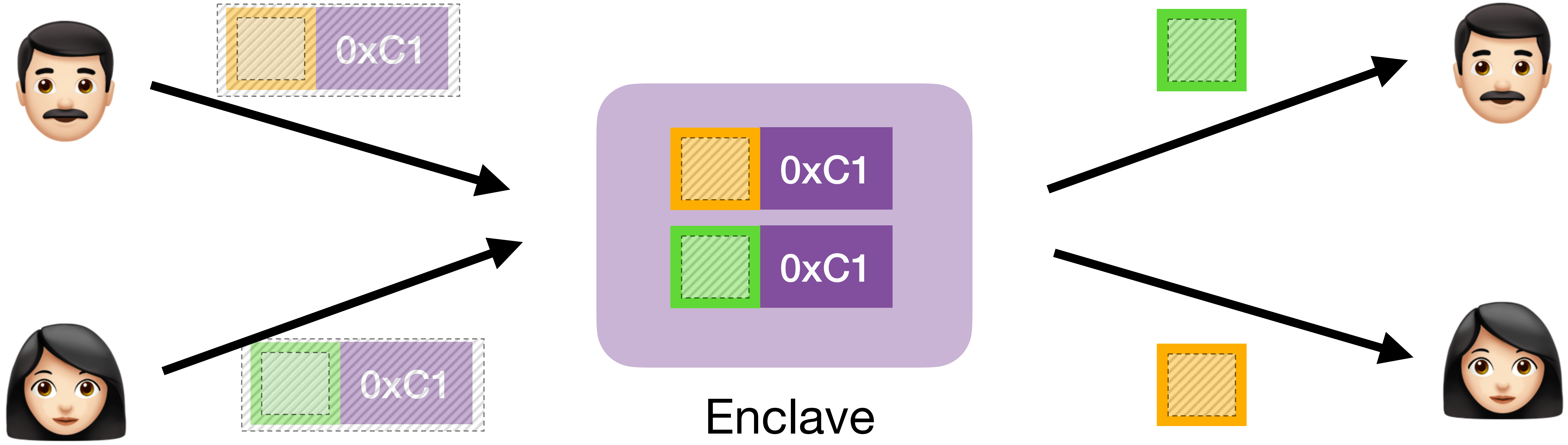
Technical overview

- Basic single-server Boomerang
  - Oblivious pairwise message exchange using one secure enclave
  - Proactive defense against active attackers
- Scalable multi-server Boomerang+
  - Security-aware load-balancing for horizontal scalability

Boomerang design

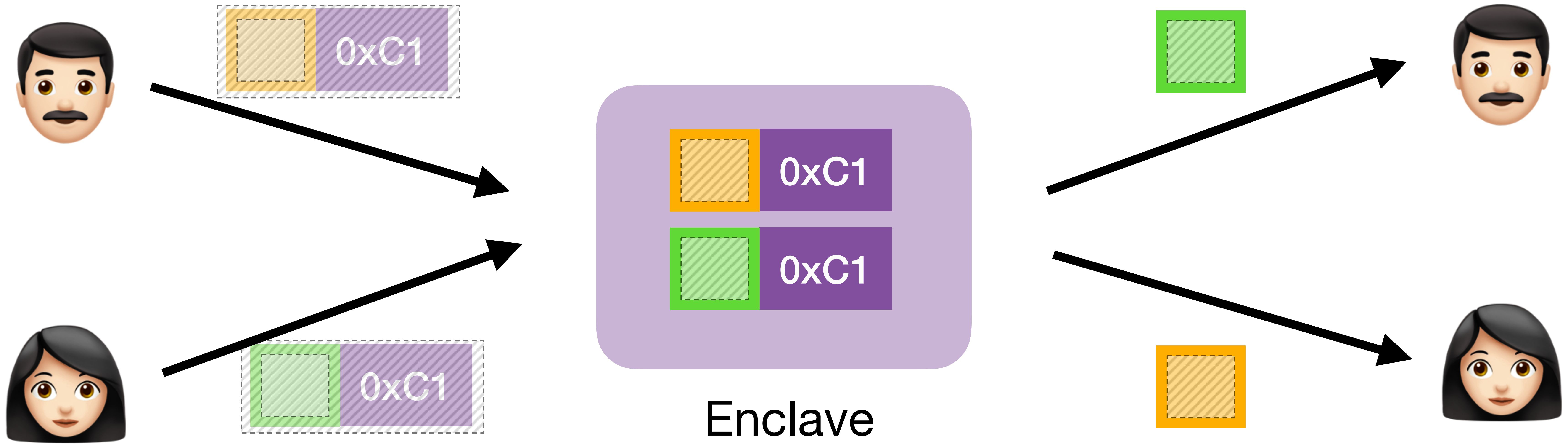


Boomerang design



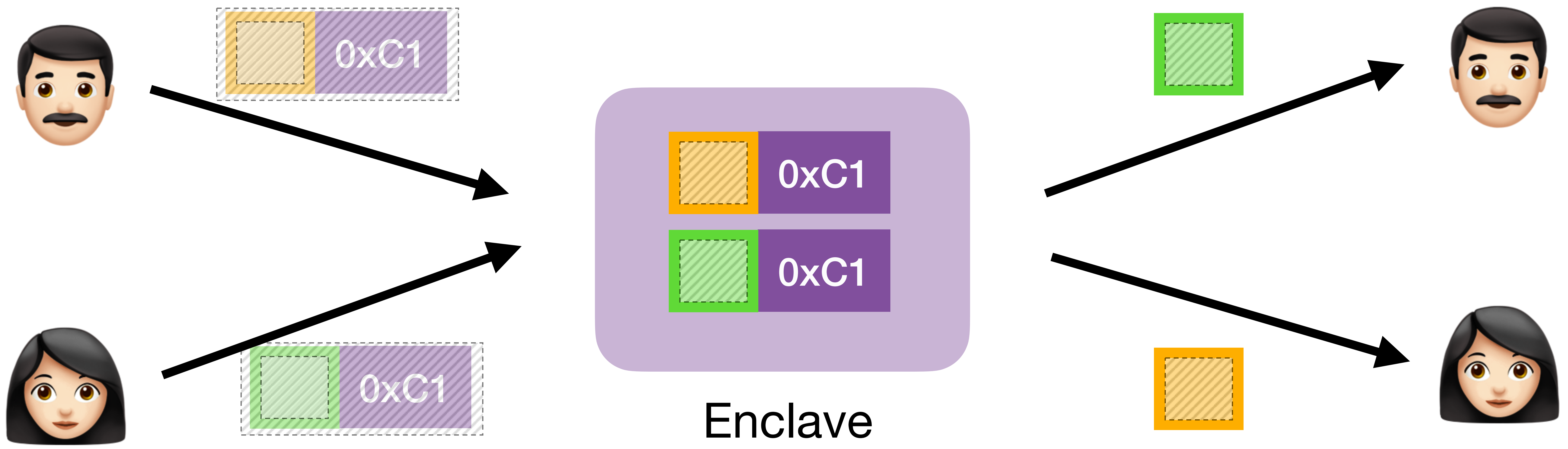
- A pair of buddies send messages tagged with pairwise private labels

Boomerang design



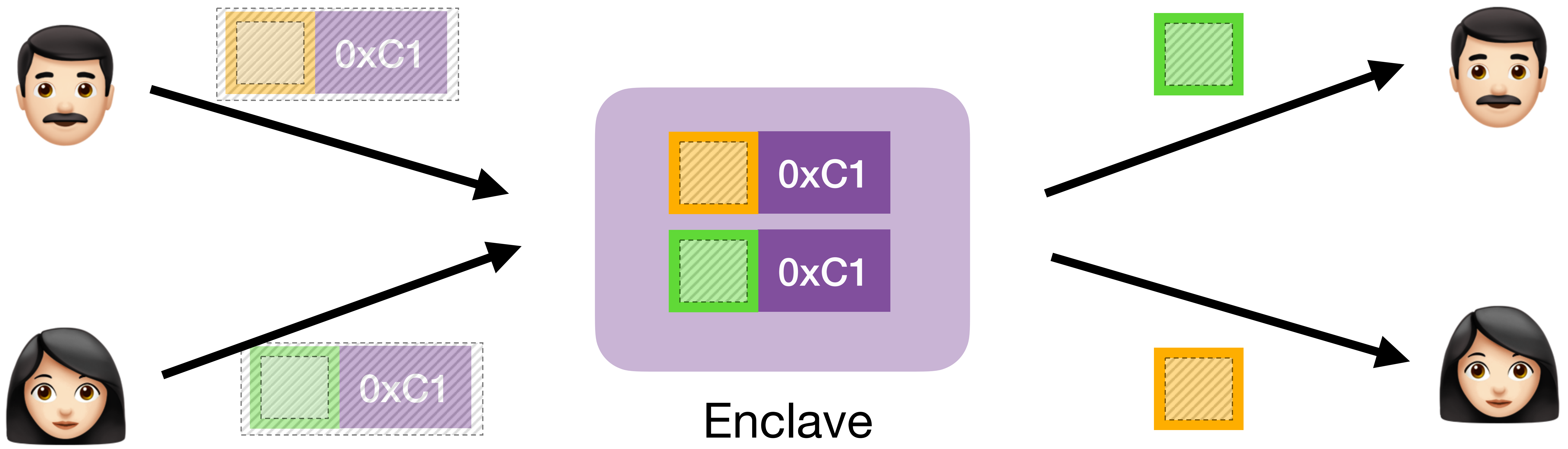
- A pair of buddies send messages tagged with pairwise private labels
- Server swaps any pair of messages with same labels

Boomerang design



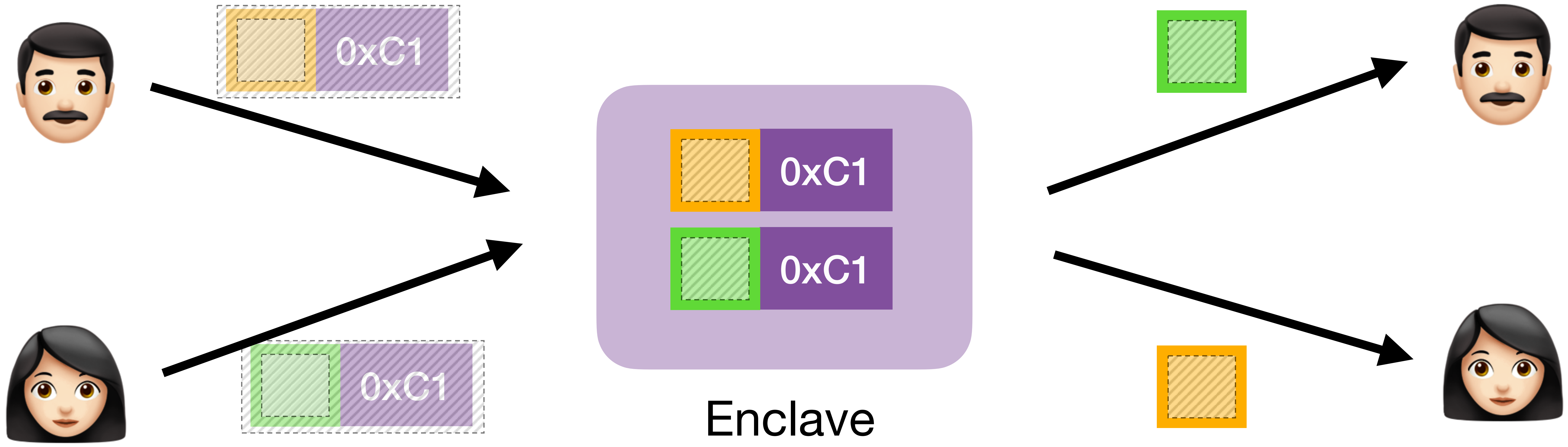
- A pair of buddies send messages tagged with pairwise private labels
- Server swaps any pair of messages with same labels
    - A regular label shows up twice each round

Boomerang design



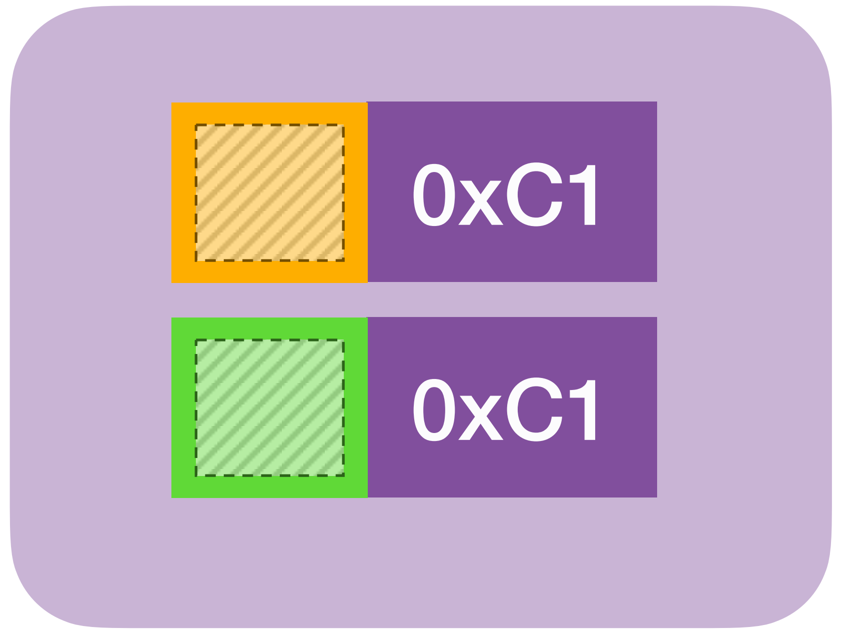
- A pair of buddies send messages tagged with pairwise private labels
- Server swaps any pair of messages with same labels
    - A regular label shows up twice each round
- But powerful attackers might disrupt the regular label pattern

Boomerang design

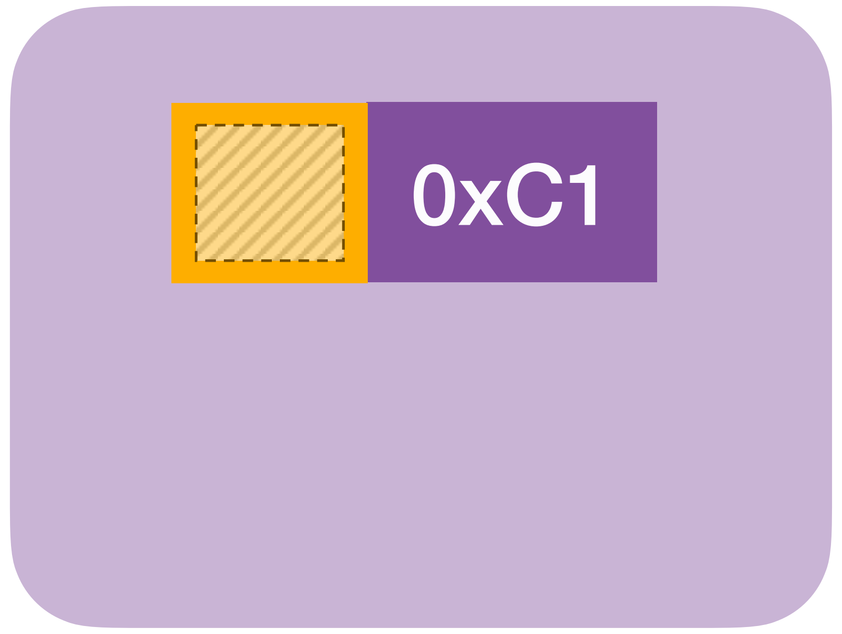


- A pair of buddies send messages tagged with pairwise private labels
- Server swaps any pair of messages with same labels
  - A regular label shows up twice each round
- But powerful attackers might disrupt the regular label pattern
  - Block selected clients or control a subset of clients to gain advantages

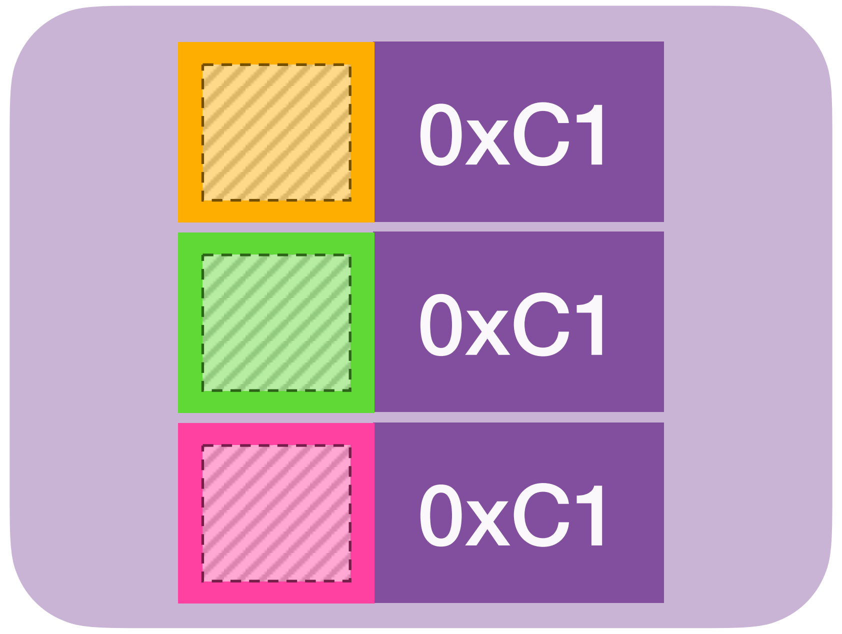
Needs to fix irregular label patterns…



Double pattern
(Regular)

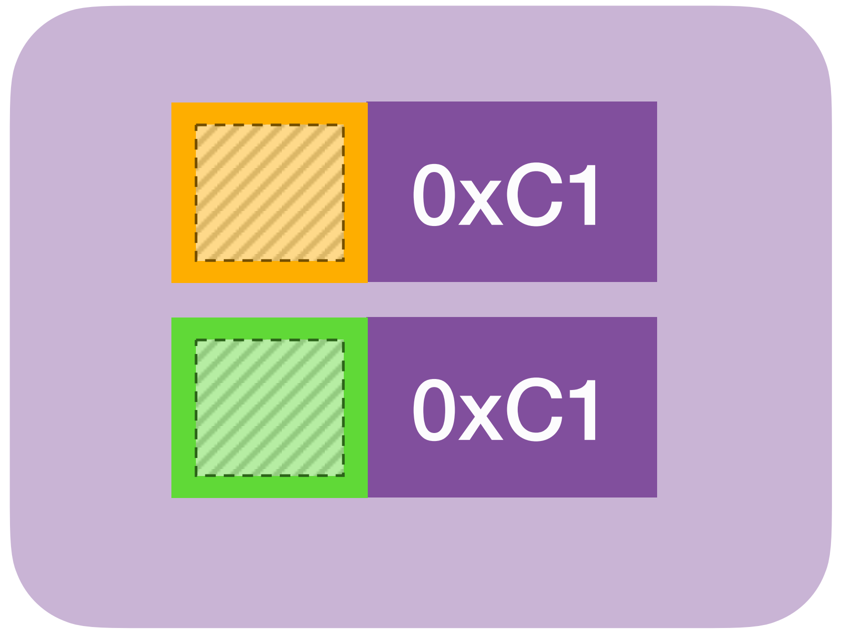


Single pattern

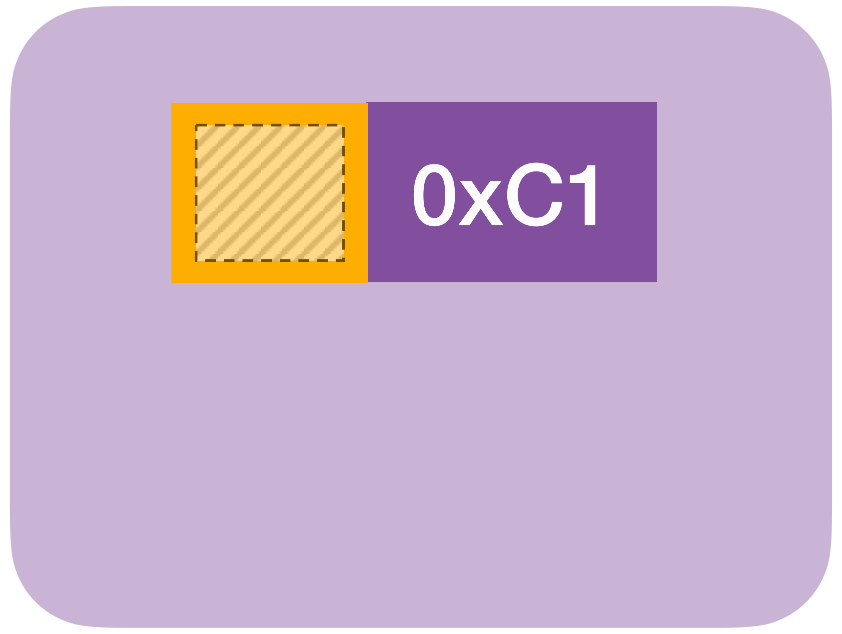


More-than-two pattern

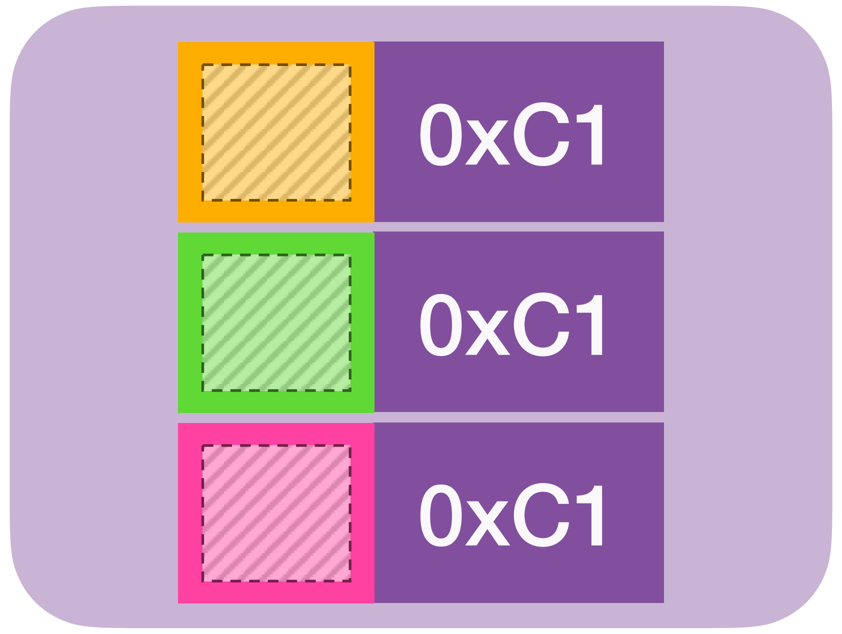
Needs to fix irregular label patterns…



Double pattern
(Regular)



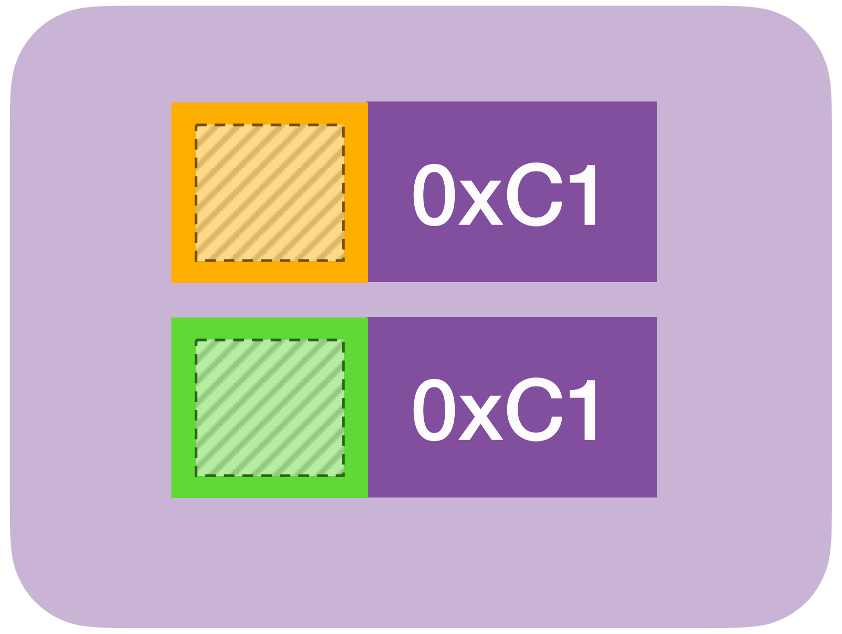
Single pattern



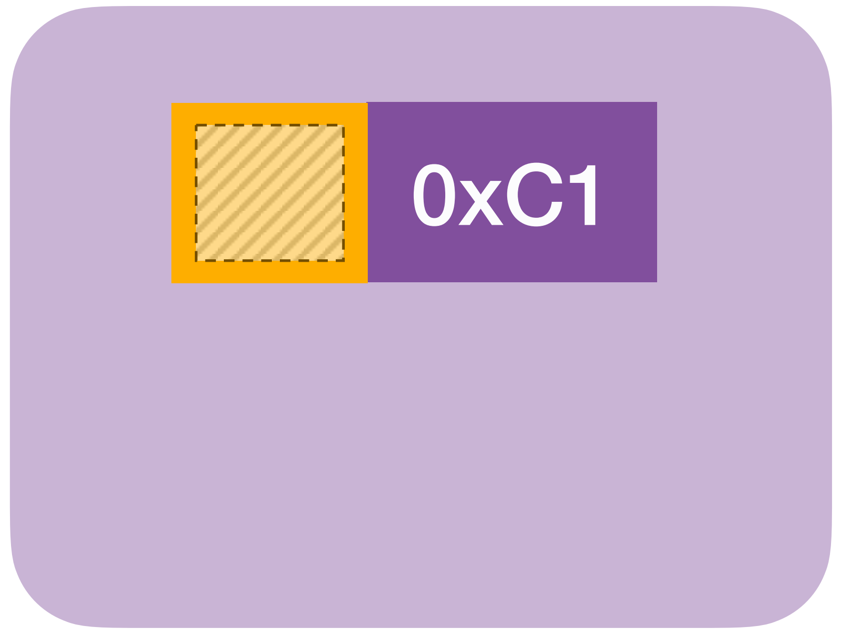
More-than-two pattern

- Build oblivious algorithms for enclaves to **proactively** detect and patch messages with irregular labels, and “return” them back to senders

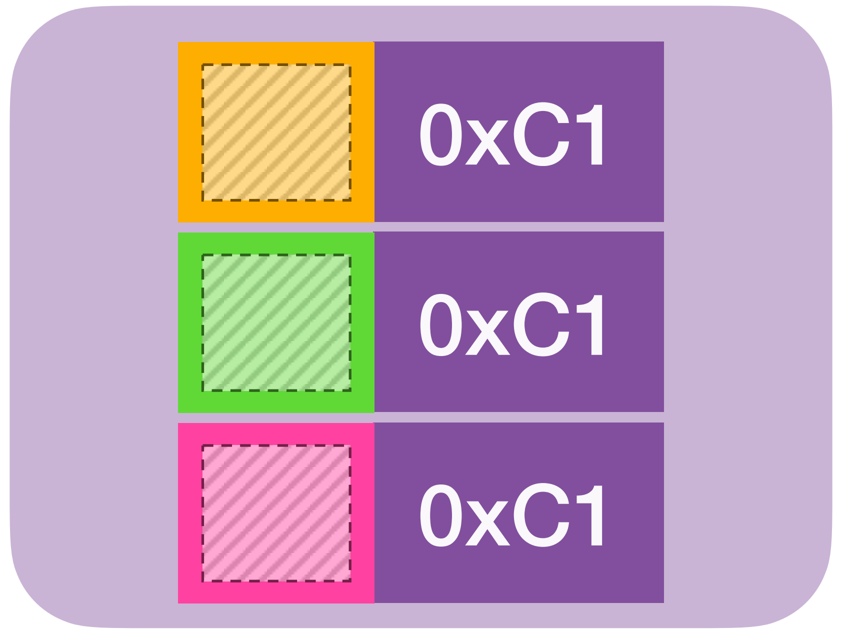
Needs to fix irregular label patterns…



Double pattern
(Regular)



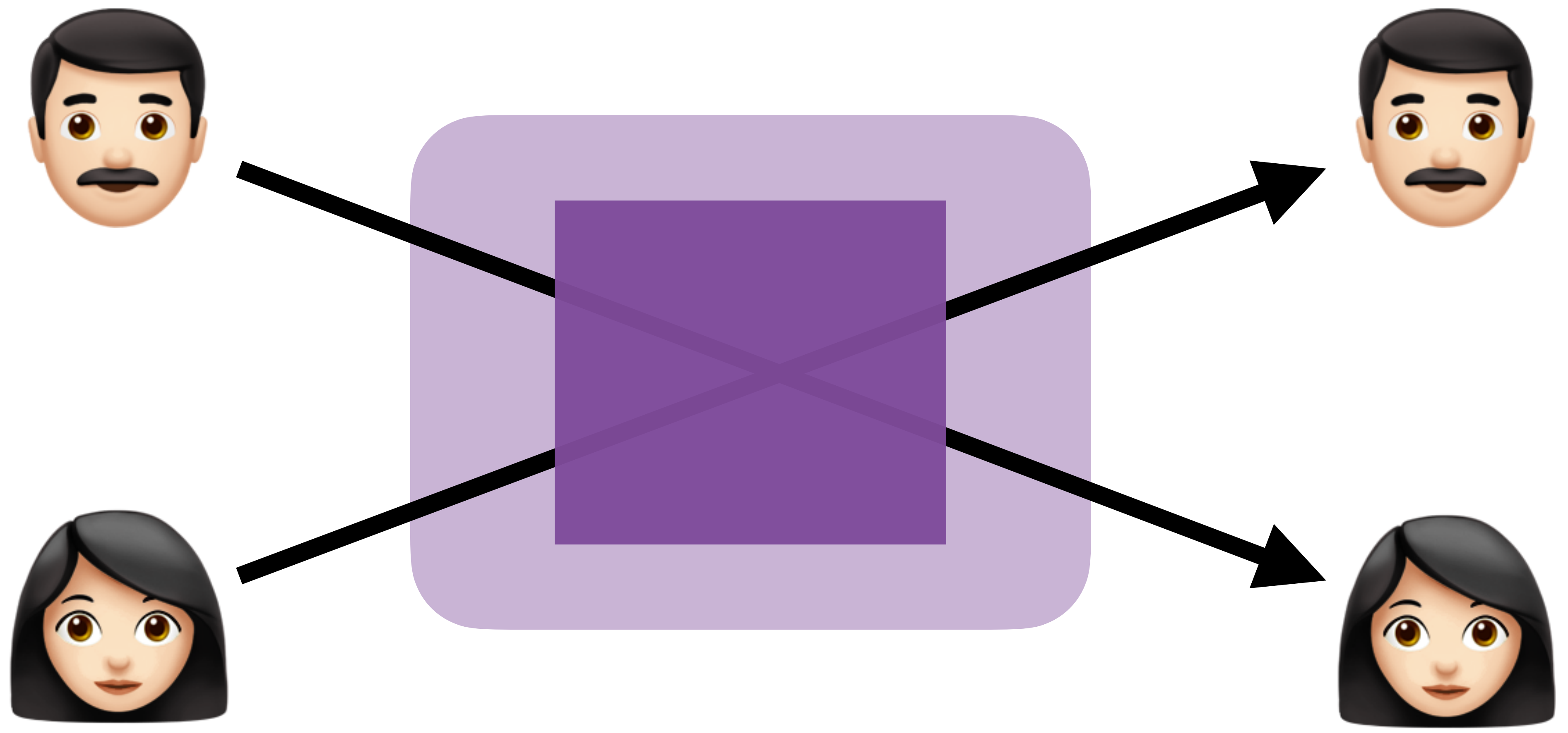
Single pattern



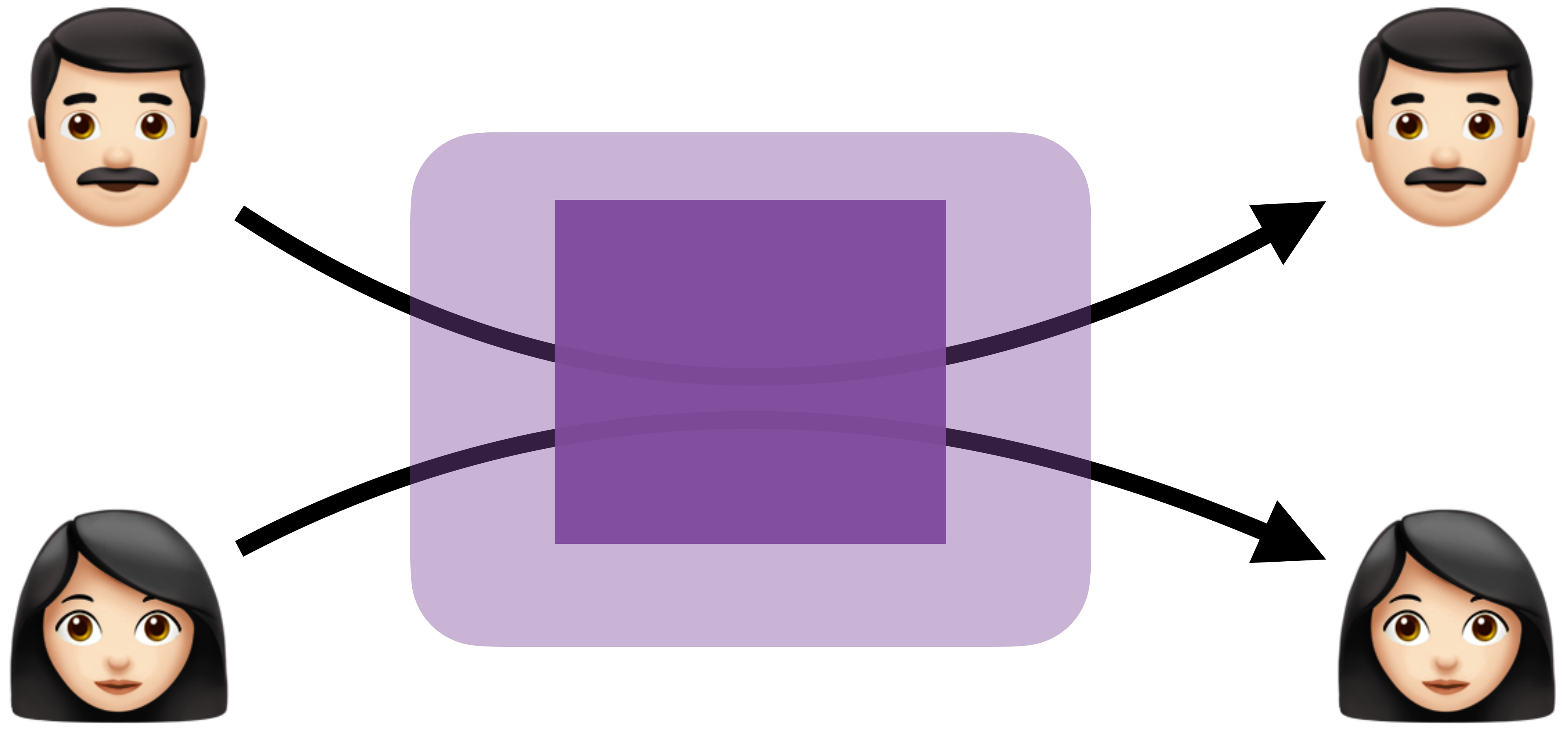
More-than-two pattern

- Build oblivious algorithms for enclaves to **proactively** detect and patch messages with irregular labels, and “return” them back to senders
- In this way, we contain the “disruptions” within problematic clients, isolated from remaining ones

Attacker blocks Alice, and infer if …

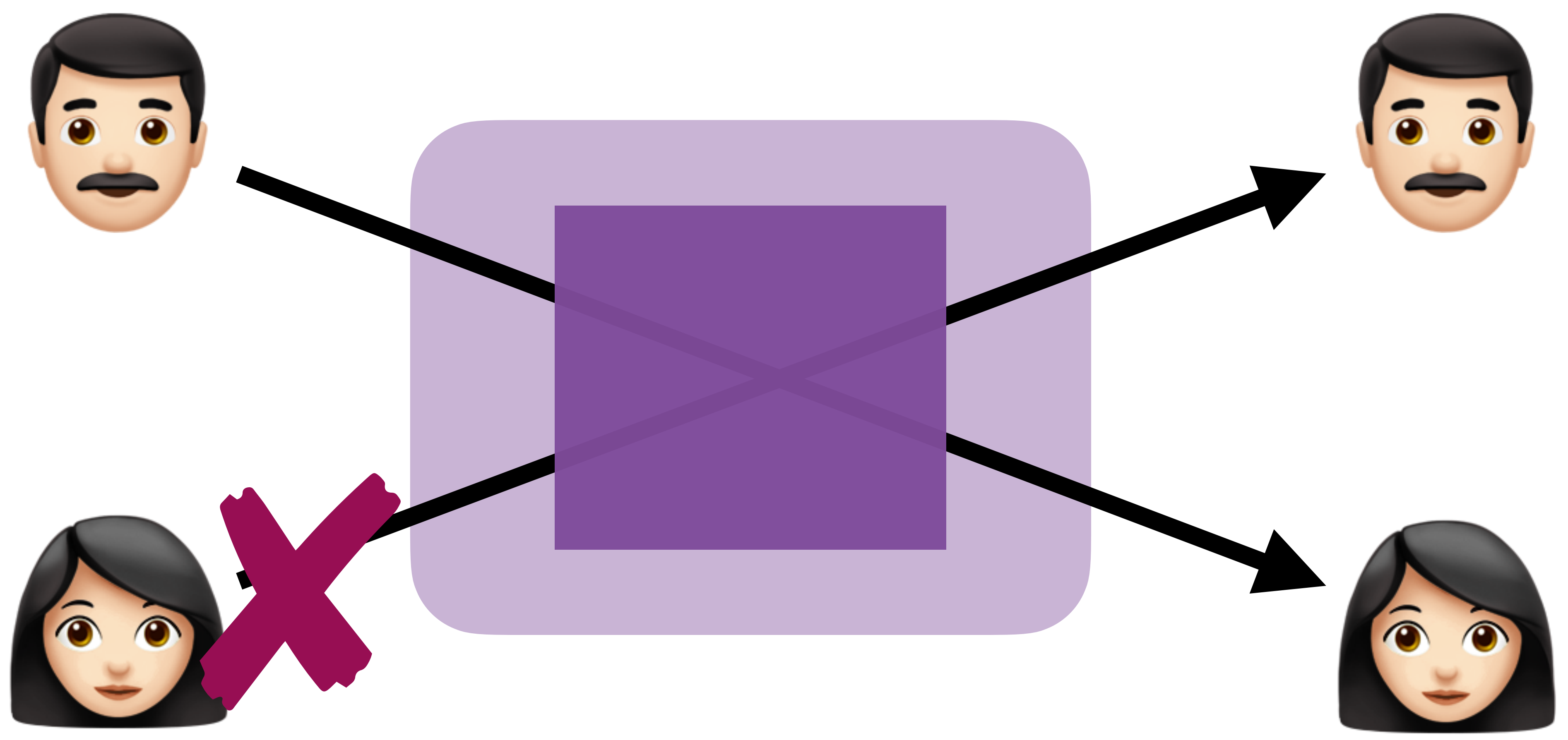


👀 Alice is talking to Bob

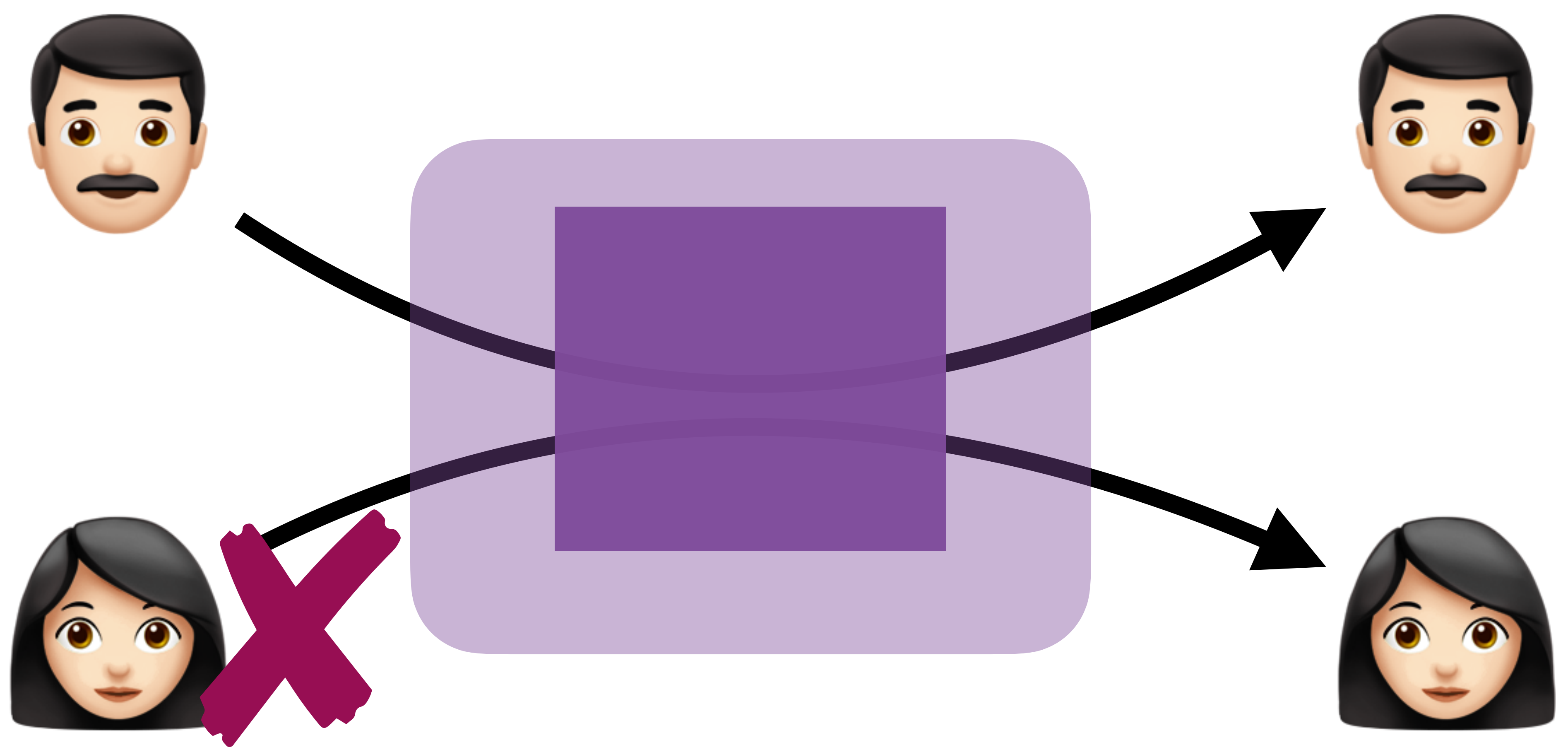


👀 Alice is talking to herself

Attacker blocks Alice, and infer if …

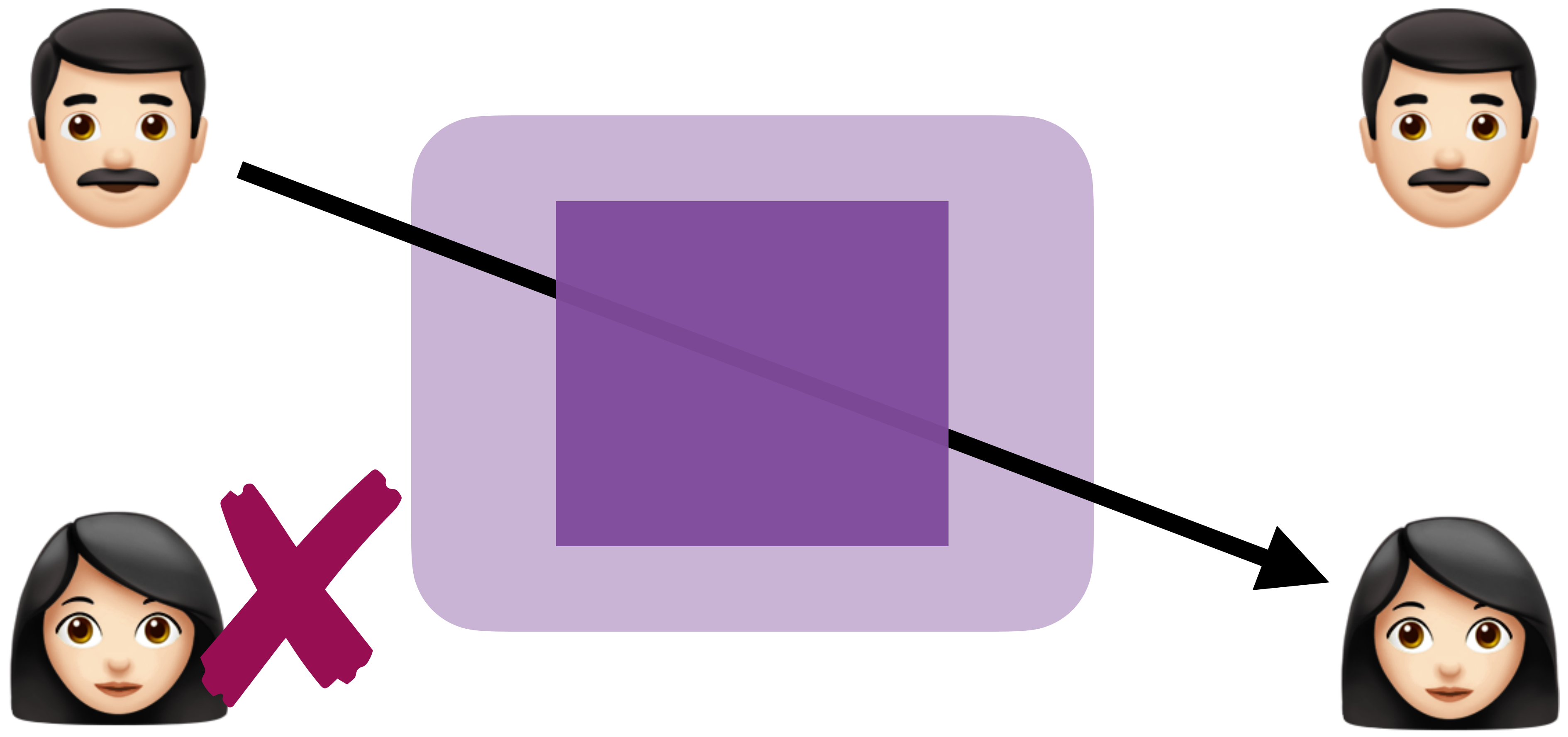


👀 Alice is talking to Bob

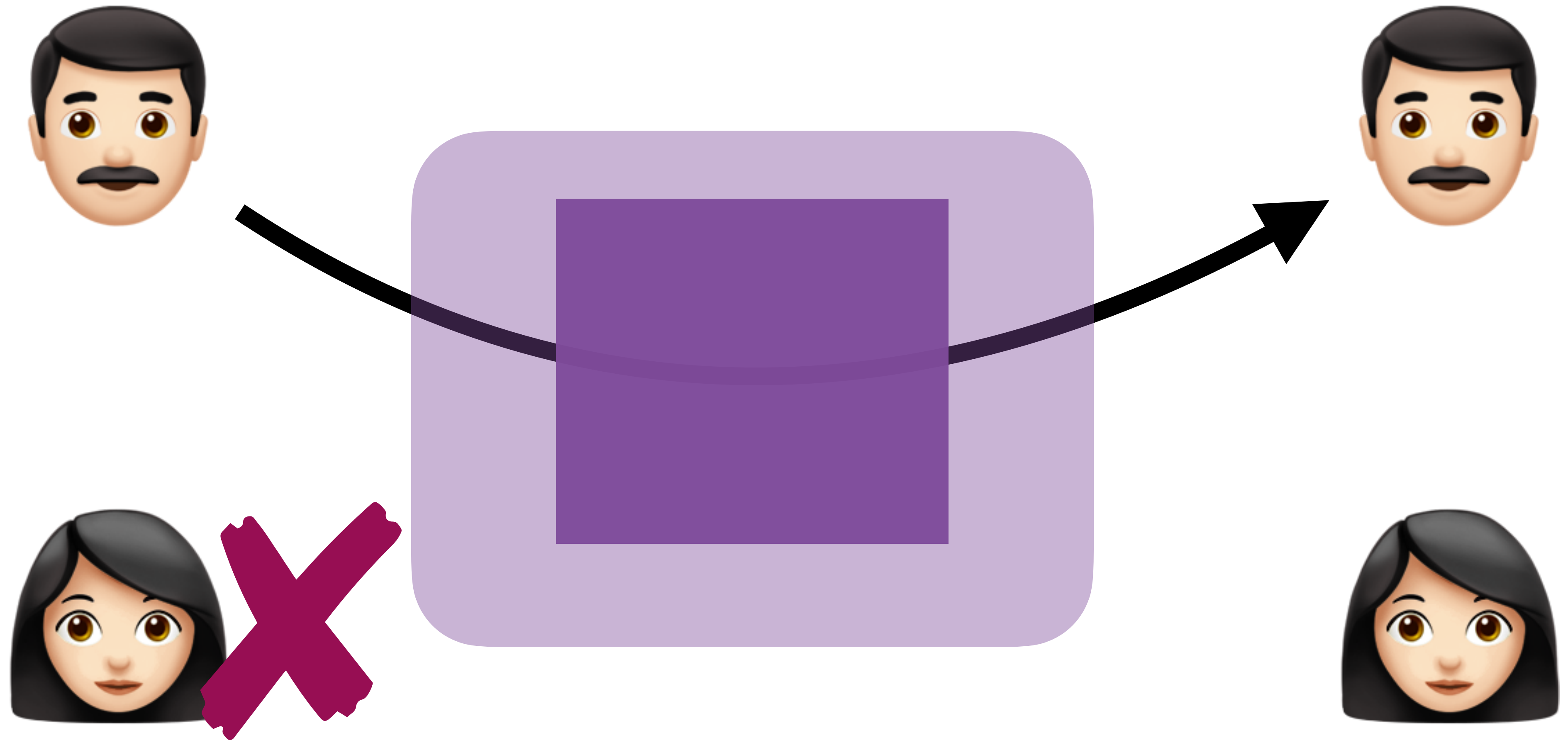


👀 Alice is talking to herself

Attacker blocks Alice, and infer if …

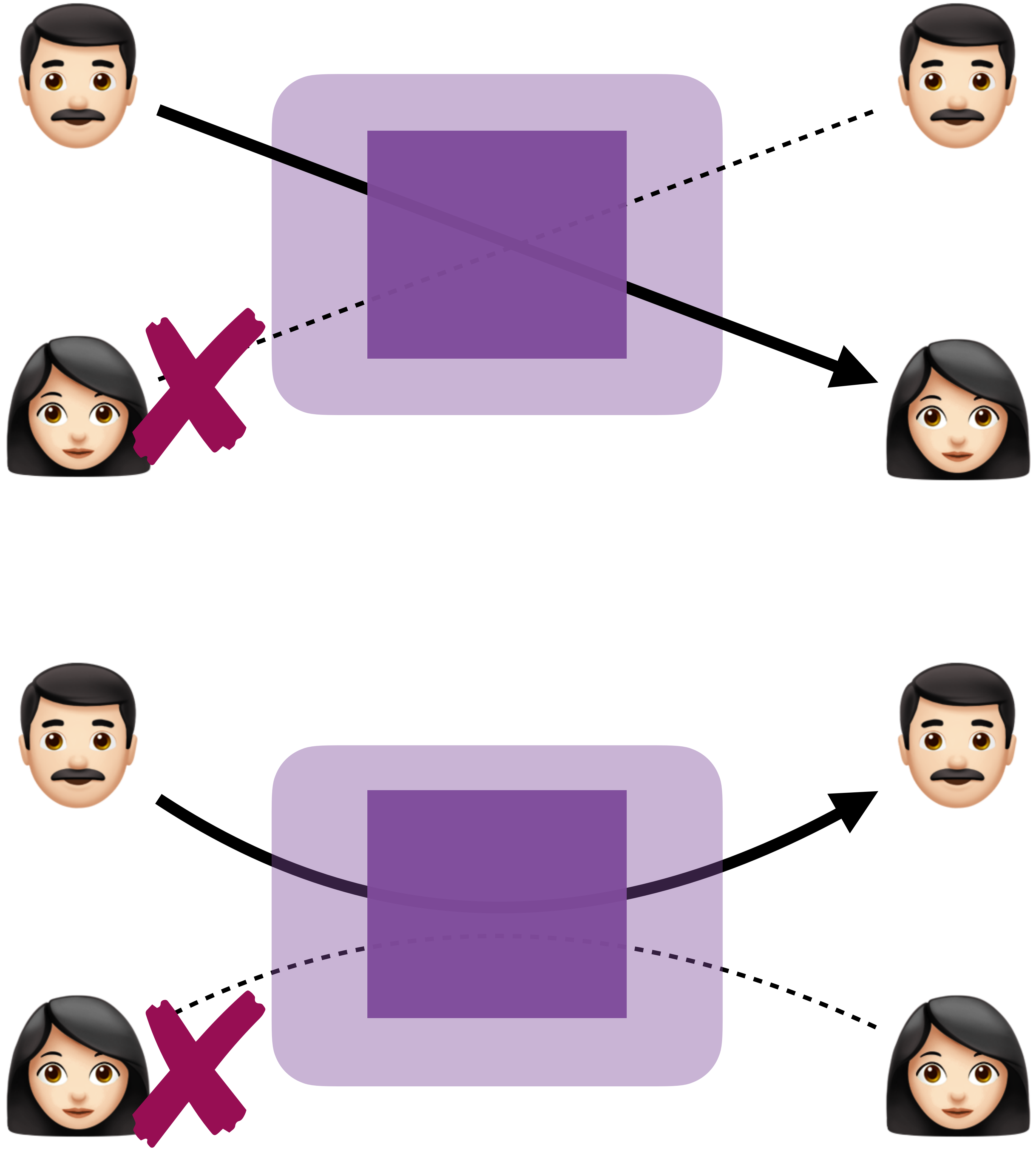


👀 Alice is talking to Bob

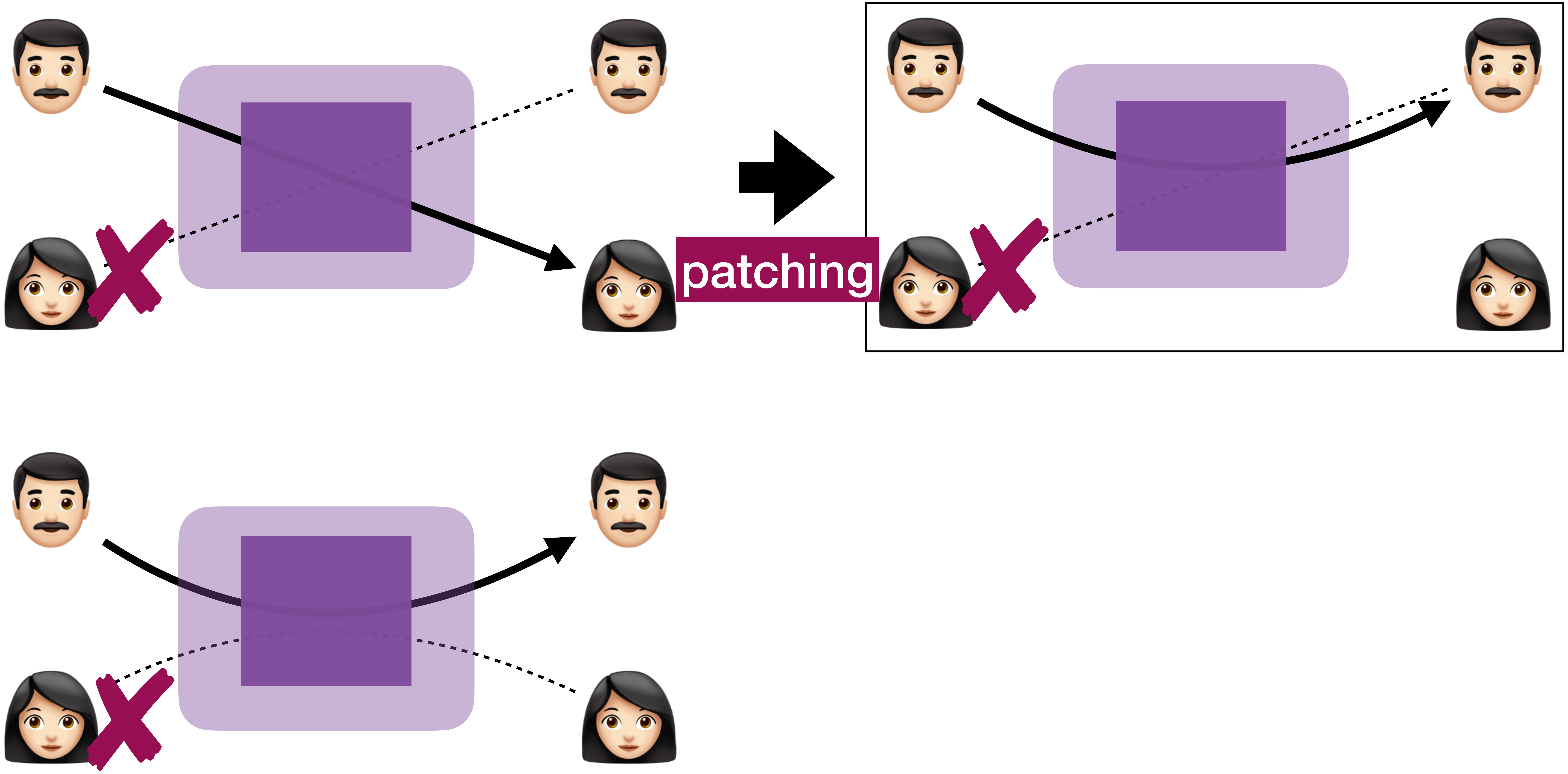


👀 Alice is talking to herself

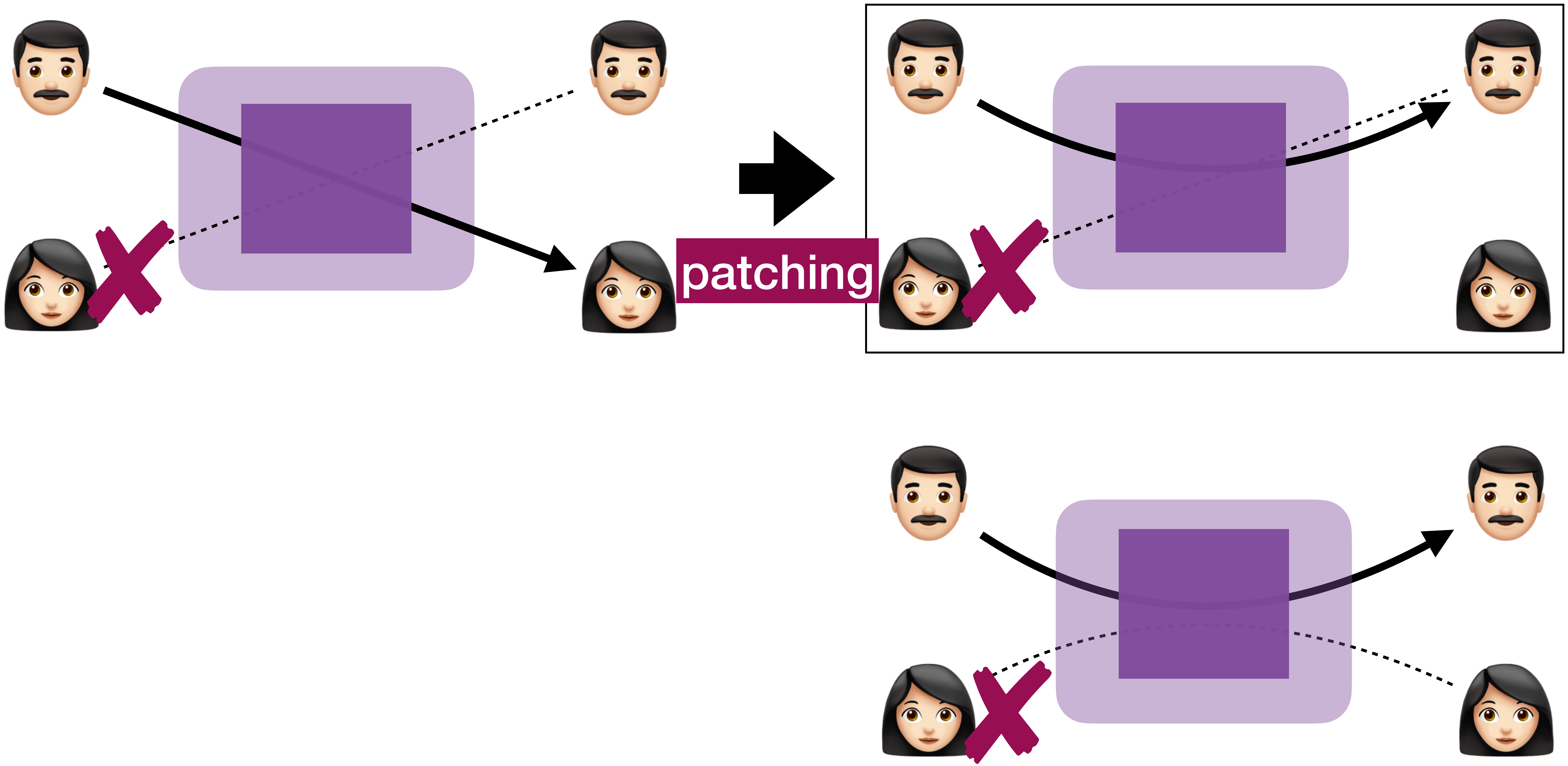
Boomerang with proactive pattern patching



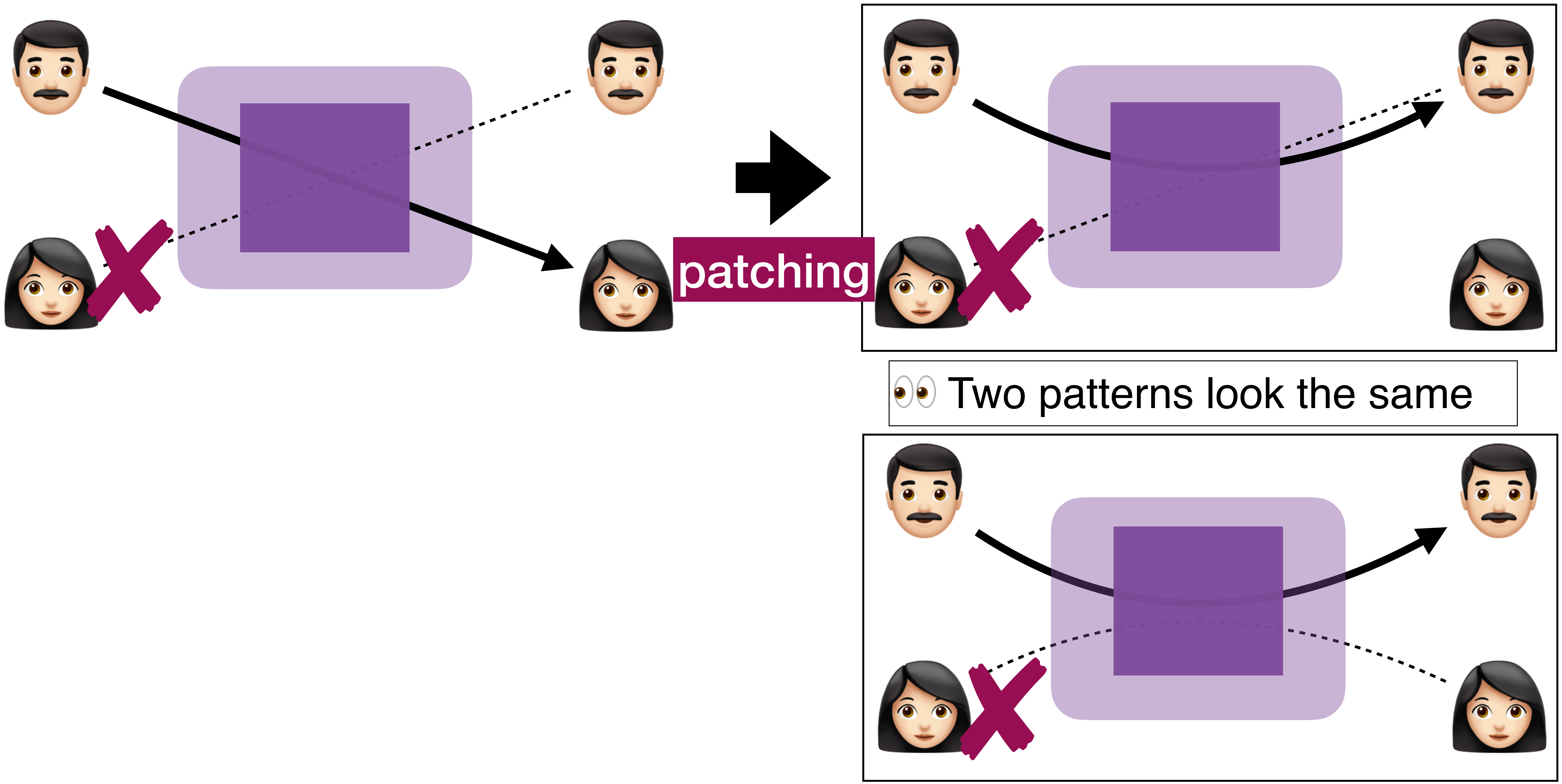
Boomerang with proactive pattern patching



Boomerang with proactive pattern patching



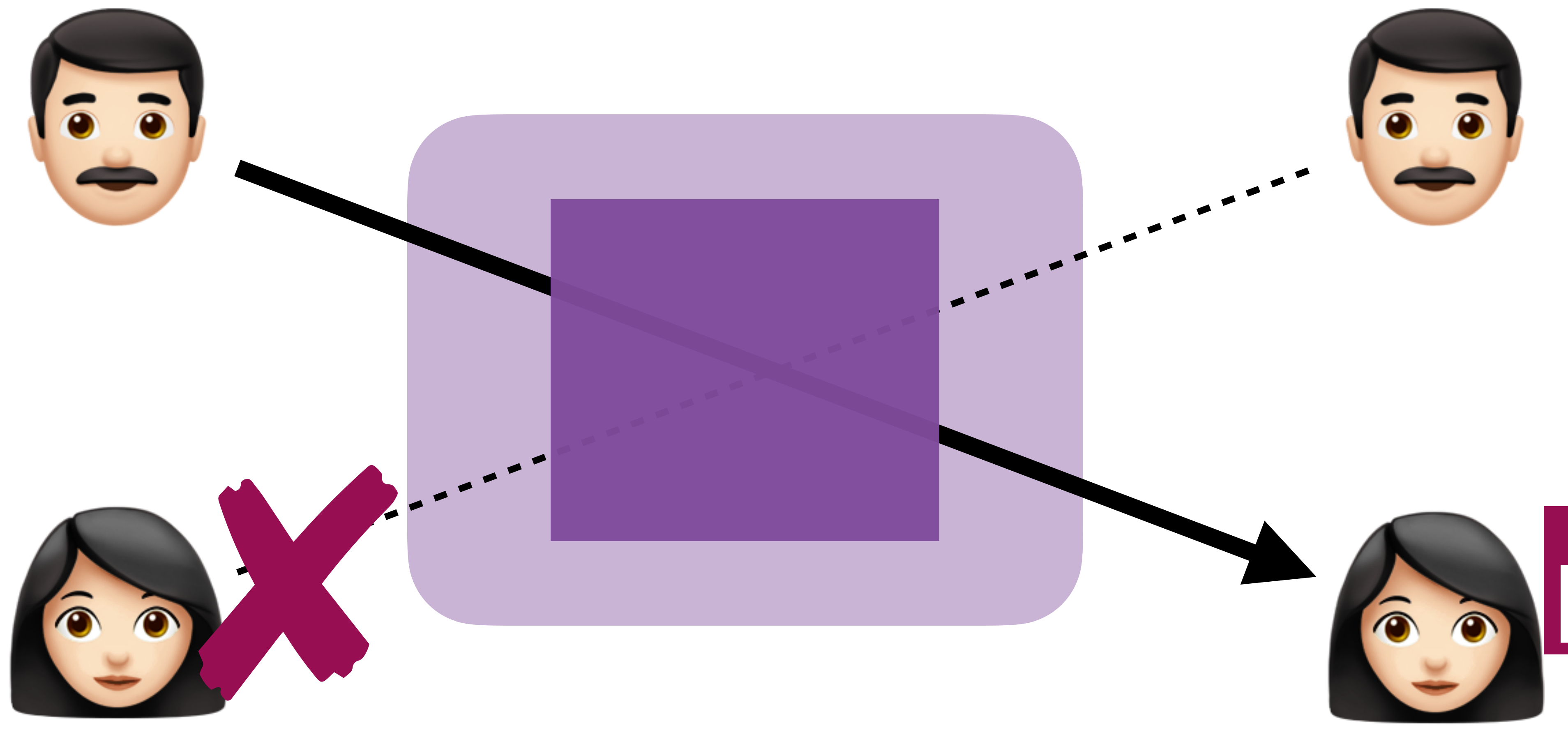
Boomerang with proactive pattern patching

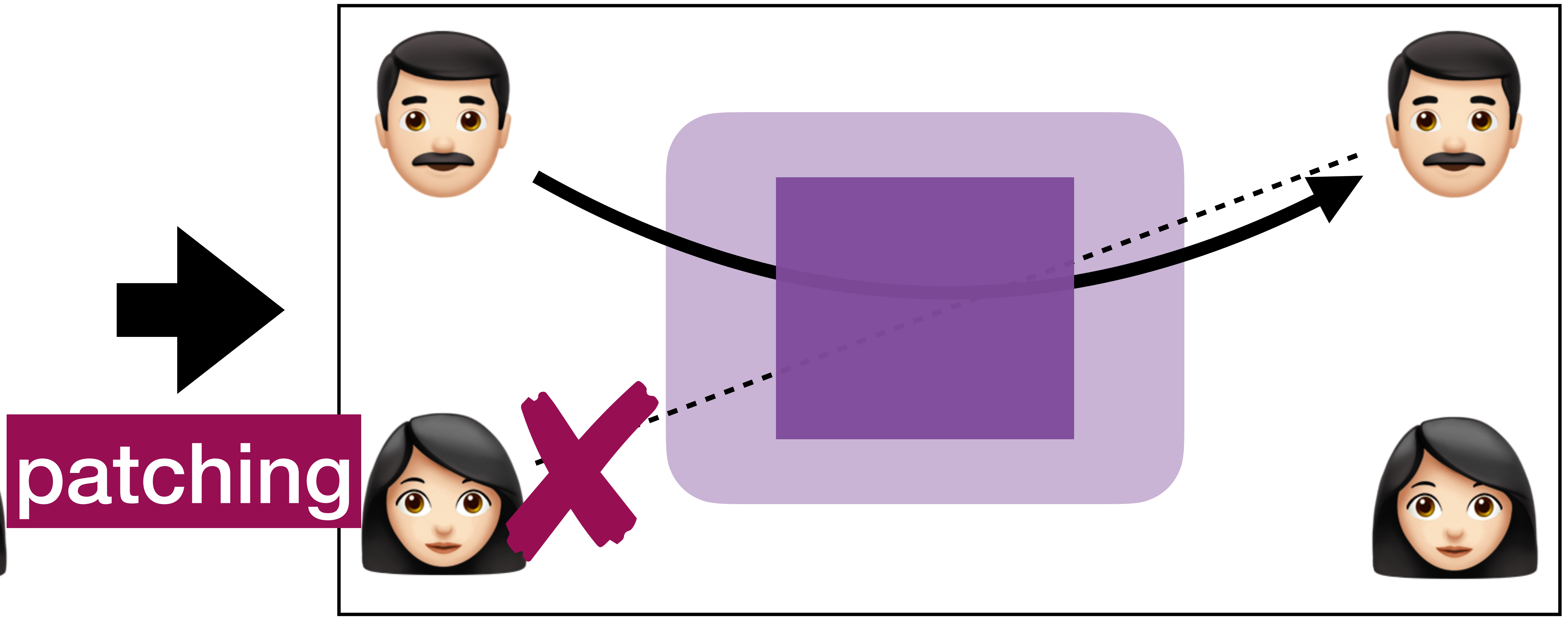


patching

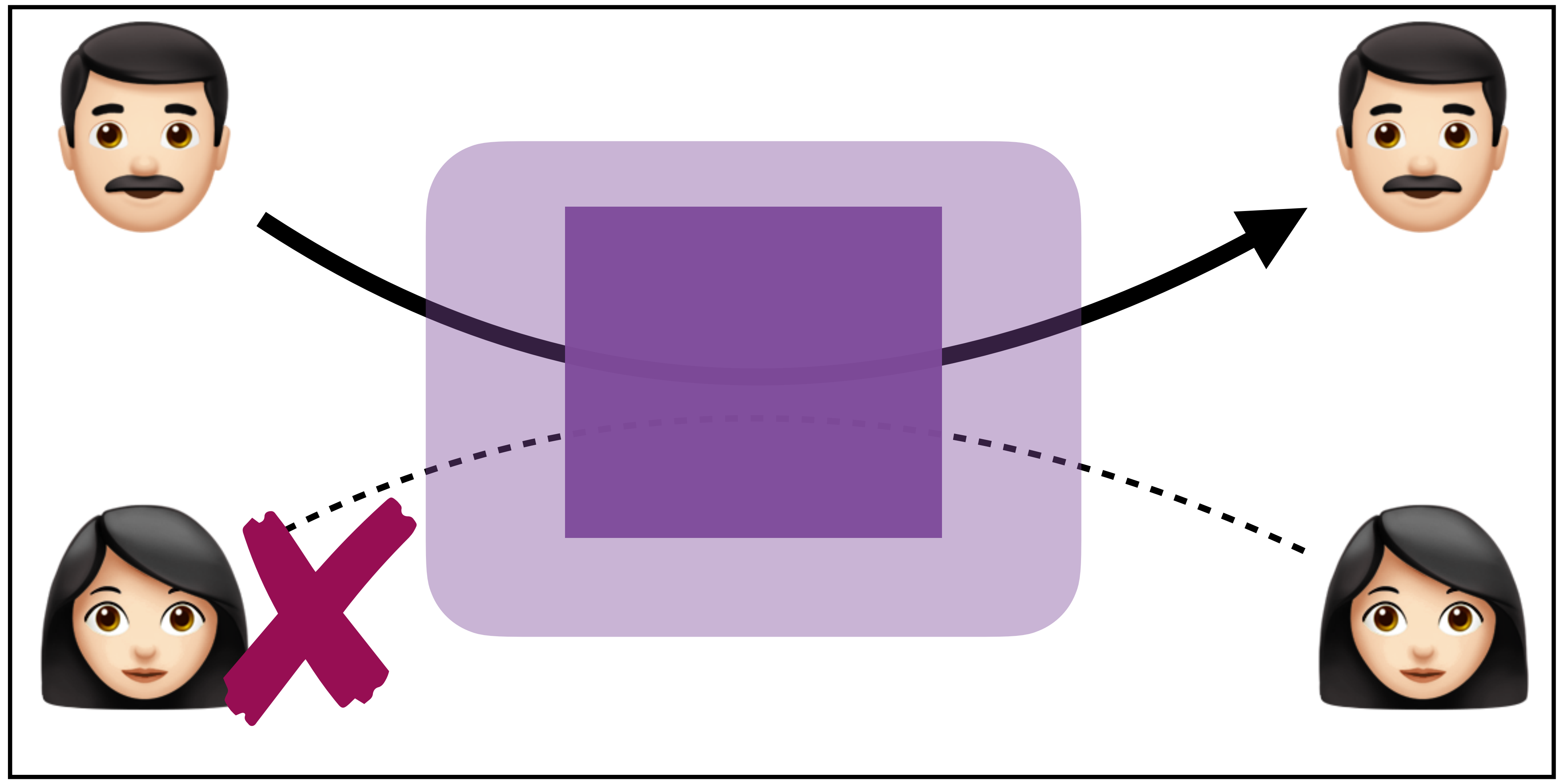
👀 Two patterns look the same

Boomerang with proactive pattern patching





👀 Two patterns look the same



Enclave **detects** irregular label patterns, and proactively **bounces them back** to the senders.

Like a "boomerang"

Oblivious irregular pattern detection & patching

Oblivious irregular pattern detection & patching

- Obliviously sort the labels

Oblivious irregular pattern detection & patching

- Obliviously sort the labels

Label	Sender
0x0C	A
0x12	B
0x12	C
0x12	D
0x1C	E
0x1C	F
0x3F	G

Enclave’s view of message sequence

Oblivious irregular pattern detection & patching

- Obliviously sort the labels
- Linear scan with a sliding window

Label	Sender
0x0C	A
0x12	B
0x12	C
0x12	D
0x1C	E
0x1C	F
0x3F	G

Enclave’s view of message sequence

Oblivious irregular pattern detection & patching

- Obliviously sort the labels
- Linear scan with a sliding window
  - **Detect** and identify different patterns

Label	Sender	Pattern
0x0C	A	Single
0x12	B	>2
0x12	C	>2
0x12	D	>2
0x1C	E	Double
0x1C	F	Double
0x3F	G	Single

Enclave’s view of message sequence

Oblivious irregular pattern detection & patching

- Obliviously sort the labels
- Linear scan with a sliding window
  - **Detect** and identify different patterns
  - **Patch** irregular pattens

Label	Sender	Pattern
0x0C	A	Single
0x12	B	>2
0x12	C	>2
0x12	D	>2
0x1C	E	Double
0x1C	F	Double
0x3F	G	Single

Enclave's view of message sequence

Oblivious irregular pattern detection & patching

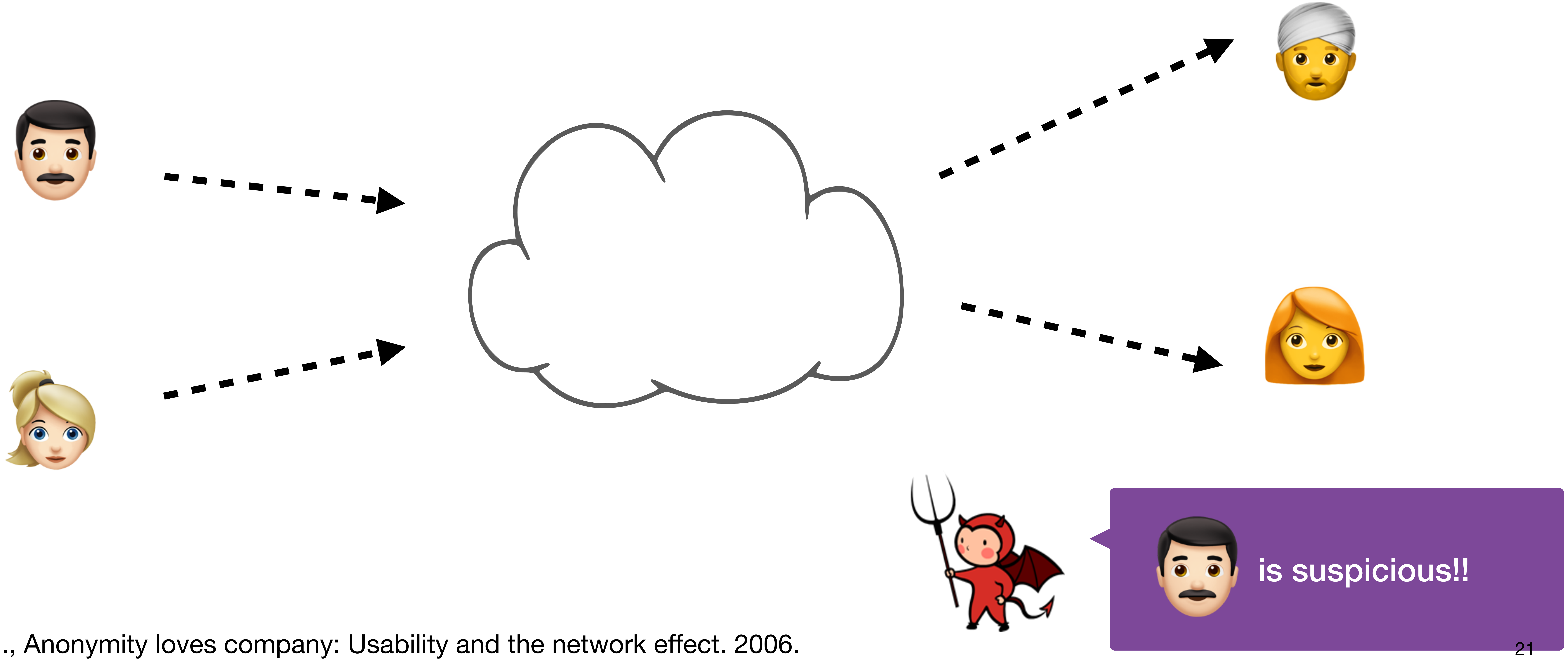
- Obliviously sort the labels
- Linear scan with a sliding window
    - **Detect** and identify different patterns
    - **Patch** irregular pattens
- All based on generic oblivious primitives (details see the paper)

Label	Sender	Pattern
0x0C	A	Single
0x12	B	>2
0x12	C	>2
0x12	D	>2
0x1C	E	Double
0x1C	F	Double
0x3F	G	Single

Enclave’s view of message sequence

Boomerang+: horizontal scaling

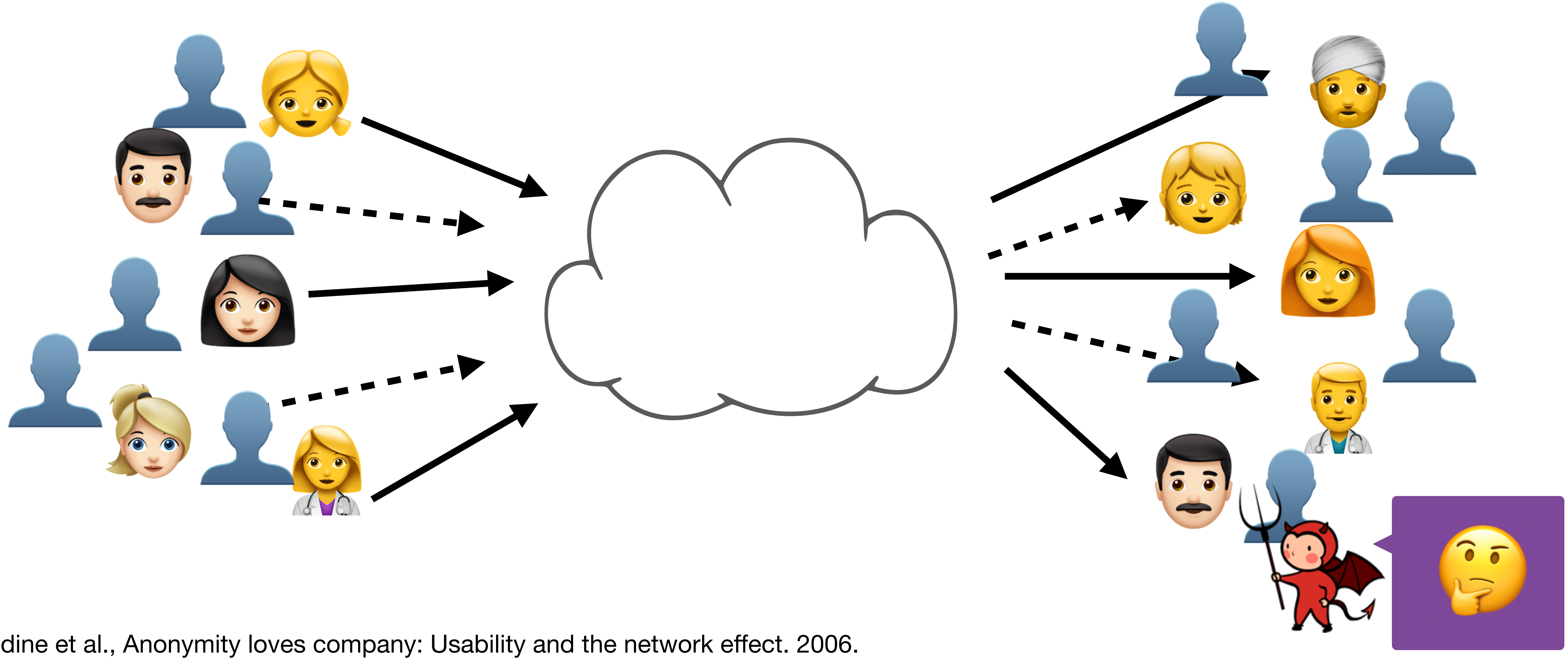
- Anonymity loves company



Dingledine et al., Anonymity loves company: Usability and the network effect. 2006.

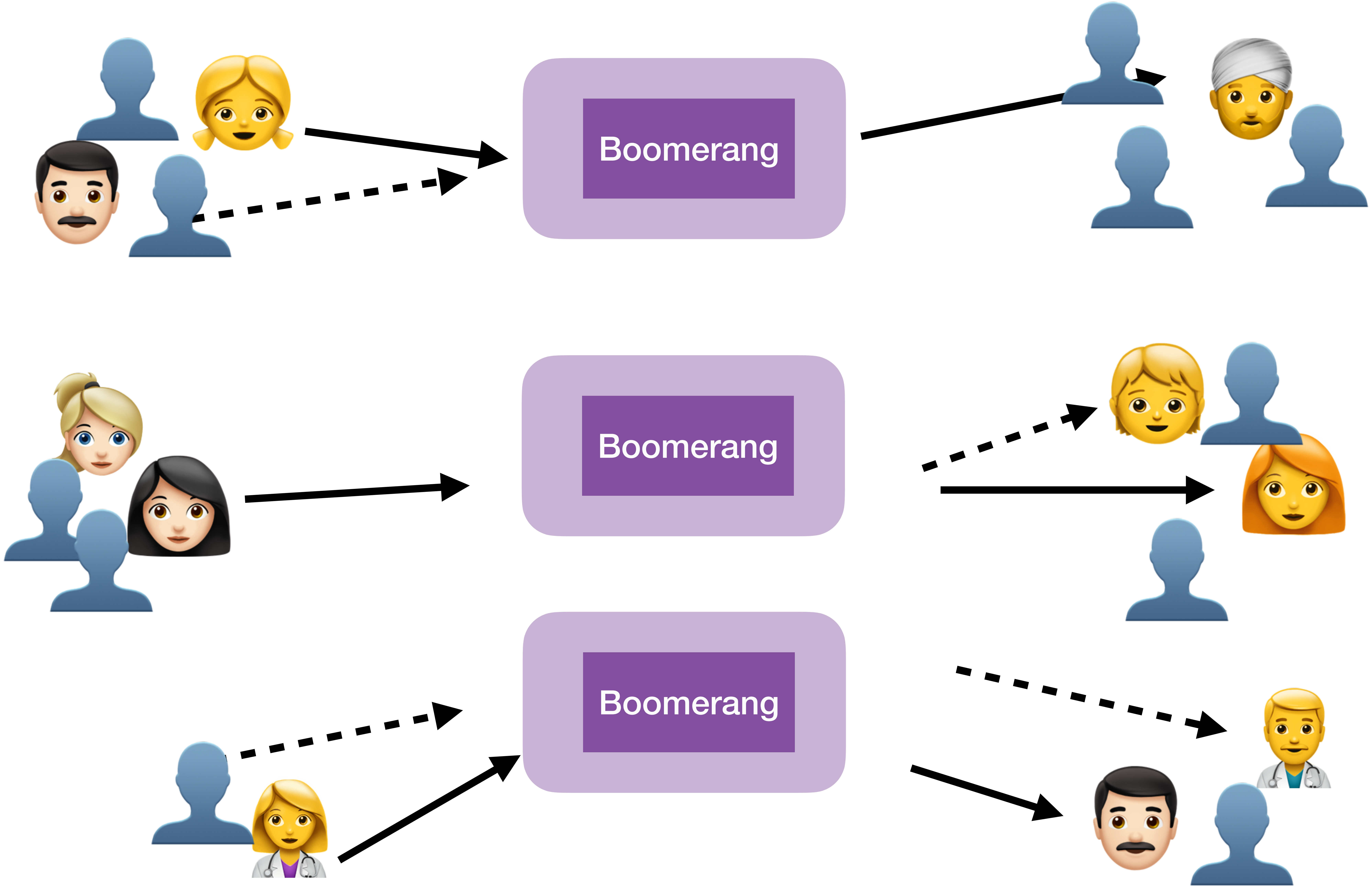
Boomerang+: horizontal scaling

- Anonymity loves company

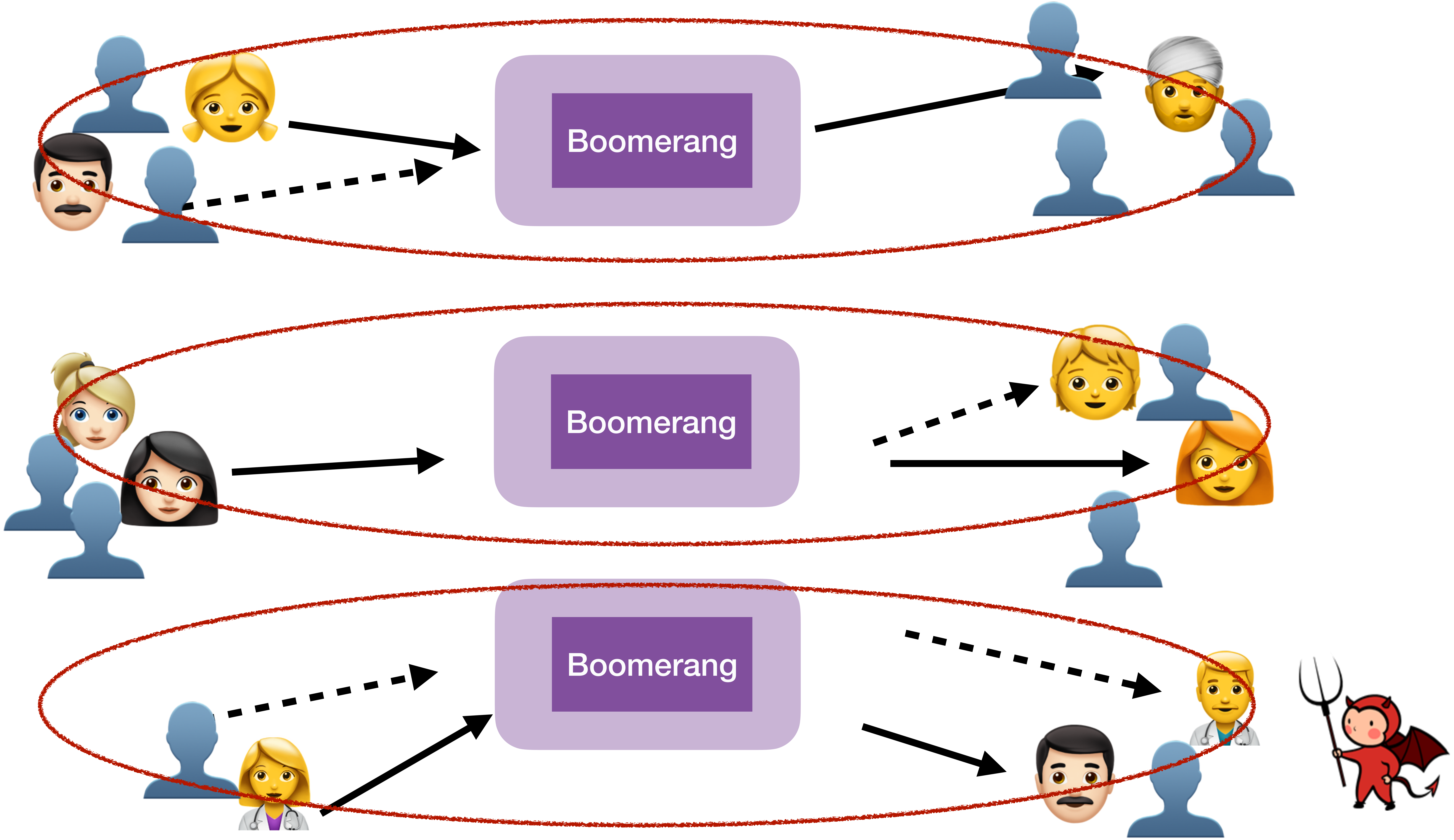


Dingledine et al., Anonymity loves company: Usability and the network effect. 2006.

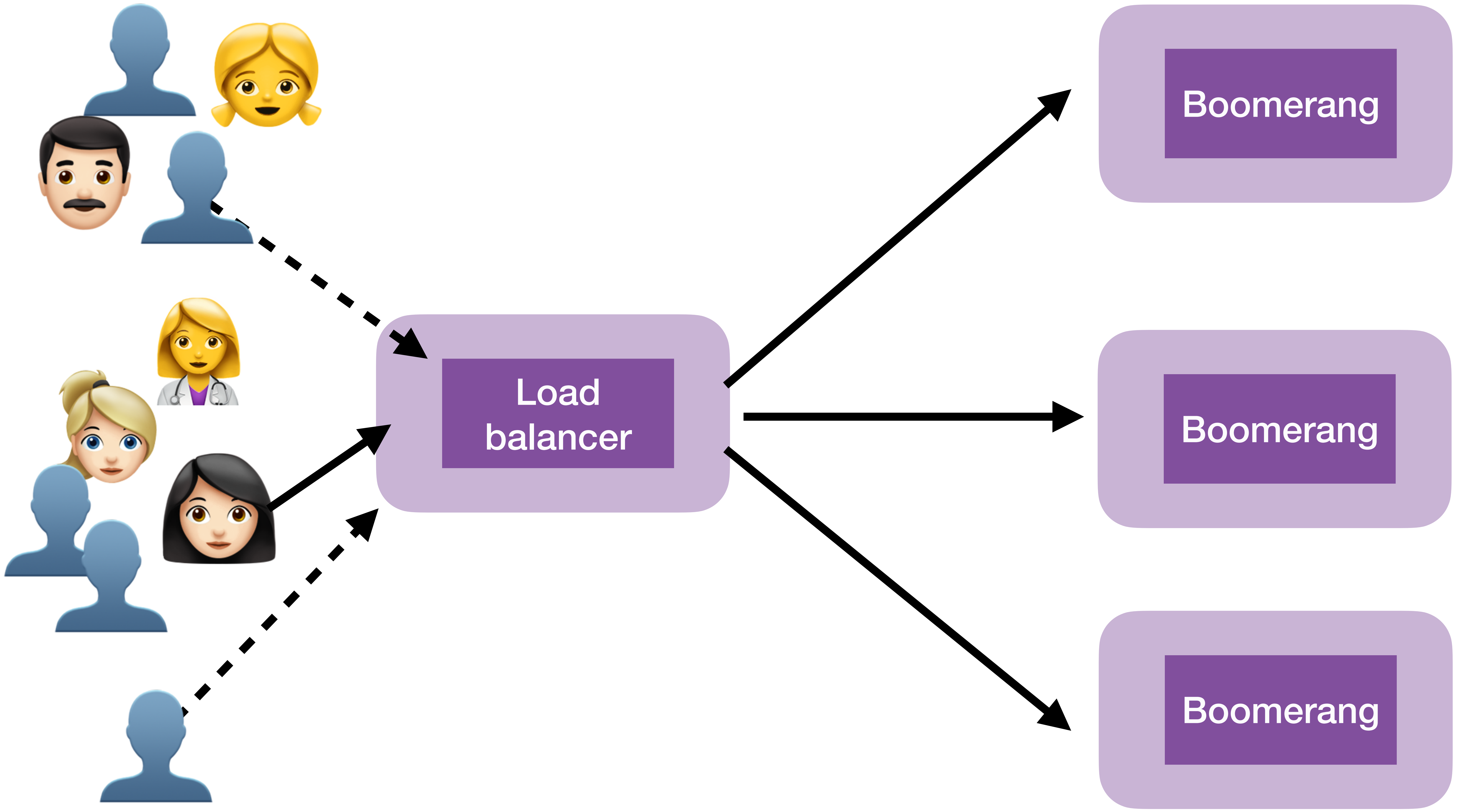
Intuition 1: duplicate boomerang servers



Intuition 1: duplicate boomerang servers



Intuition 2: use load balancers to balance traffic



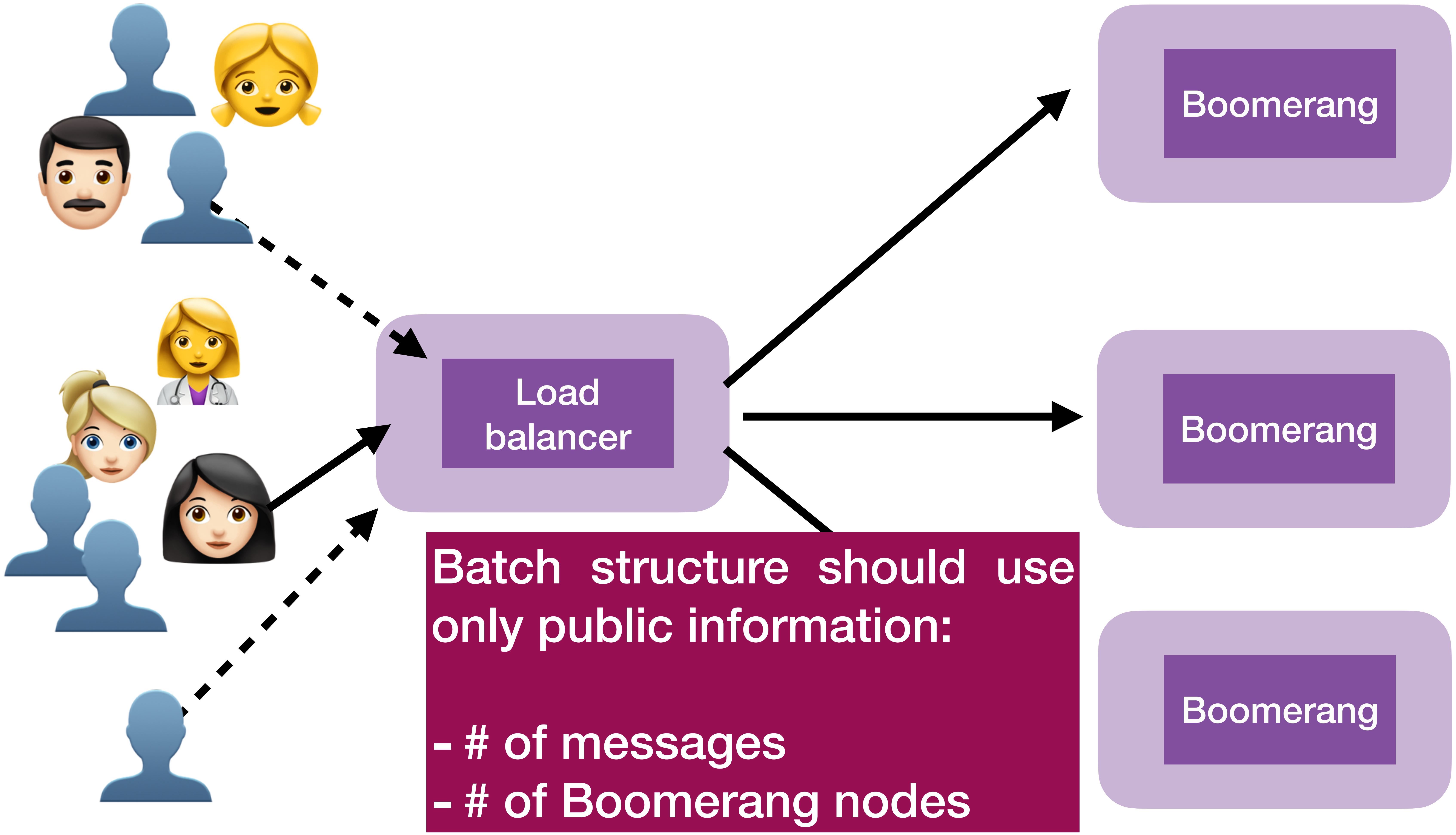
Load balancer

Boomerang

Boomerang

Boomerang

Intuition 2: use load balancers to balance traffic

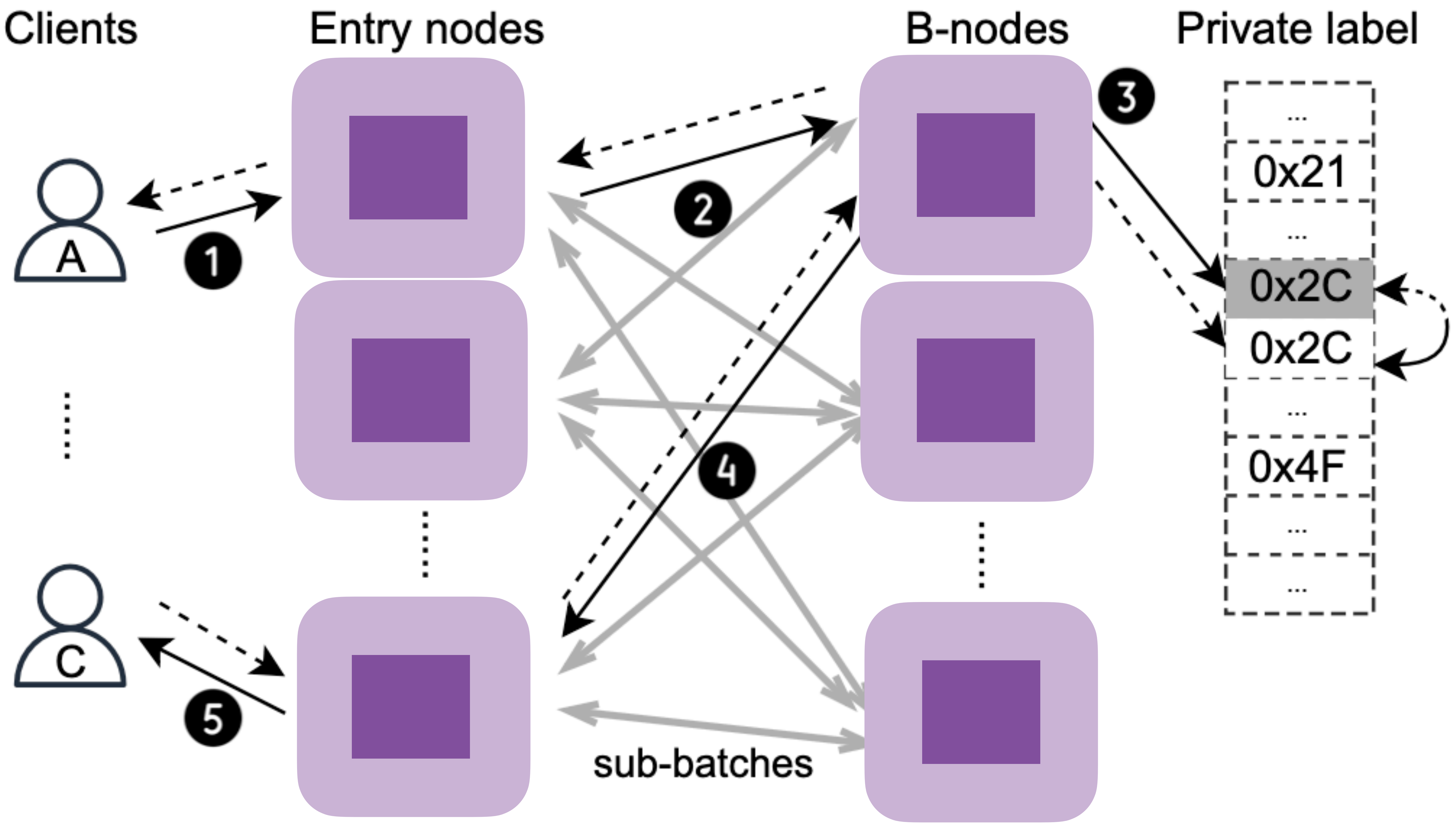


Batch structure should use only public information:

- # of messages
- # of Boomerang nodes

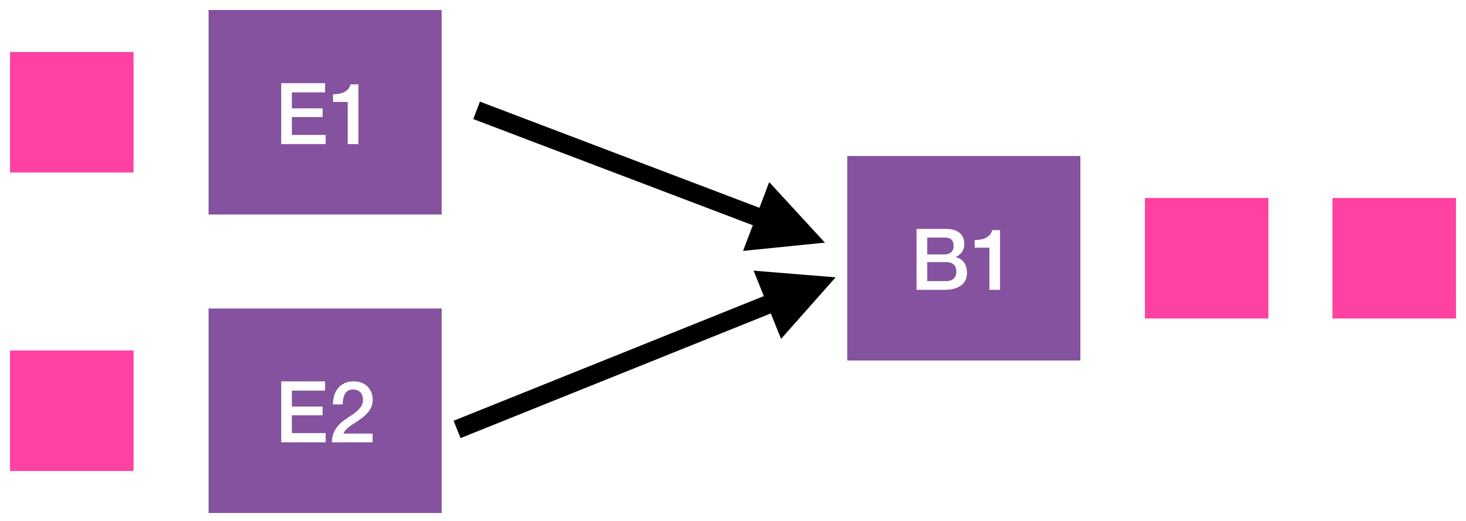
Dauterman et al., Snoopy: Surpassing the scalability bottleneck of oblivious storage. In *Proc. of ACM SOSP*, 2021.
Vuppalapati et al., SHORTSTACK: Distributed, Fault-tolerant, Oblivious Data Access. In *Proc. of* USENIX OSDI, 2022.

Boomerang+: 2-layer design



Entry node: sub-batch assignment

Function goal



always place messages between buddies to one single Boomerang node

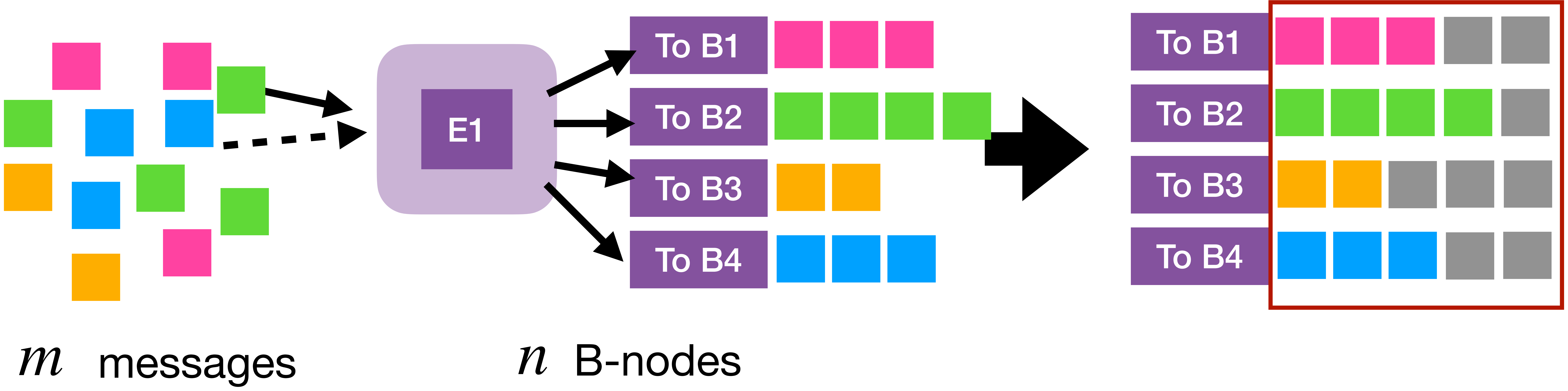
$$\mathtt{br\_id} = \mathsf{H}_k(\mathtt{priv\_label}||\mathtt{round\_num})\%n.$$

Entry node: oblivious batching

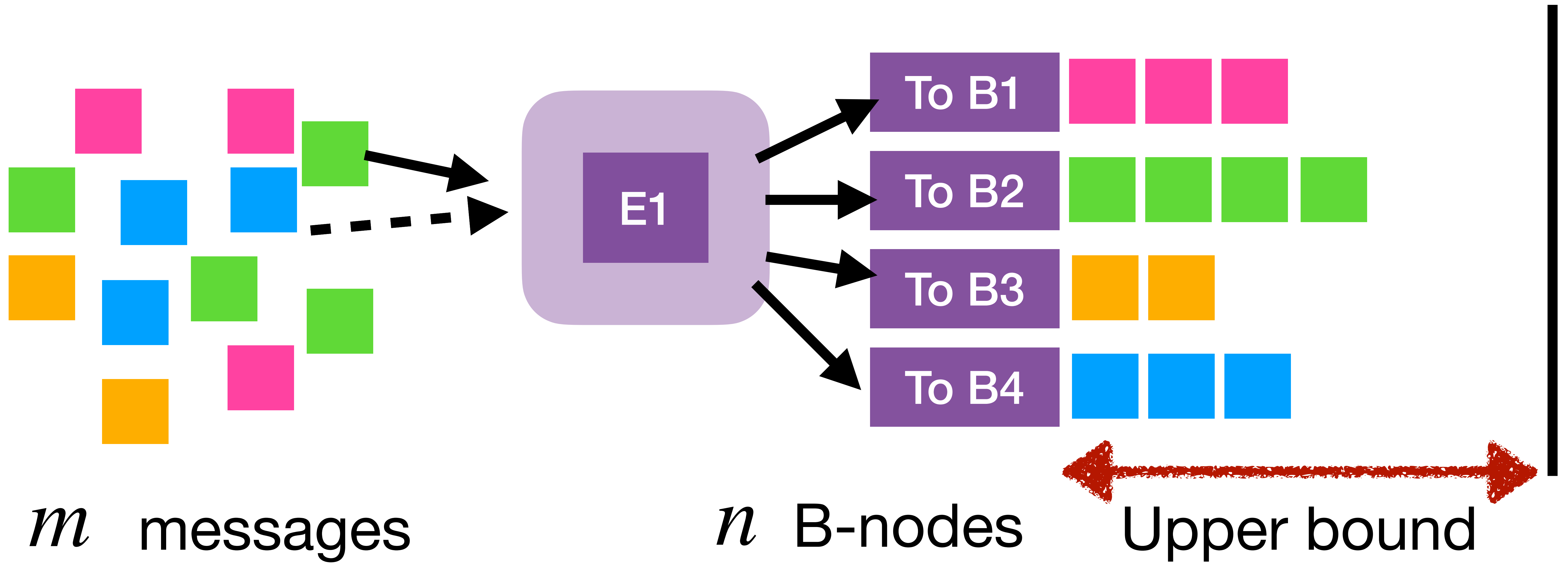
Security goal

Content-independent padding

How to choose the size? 🤔



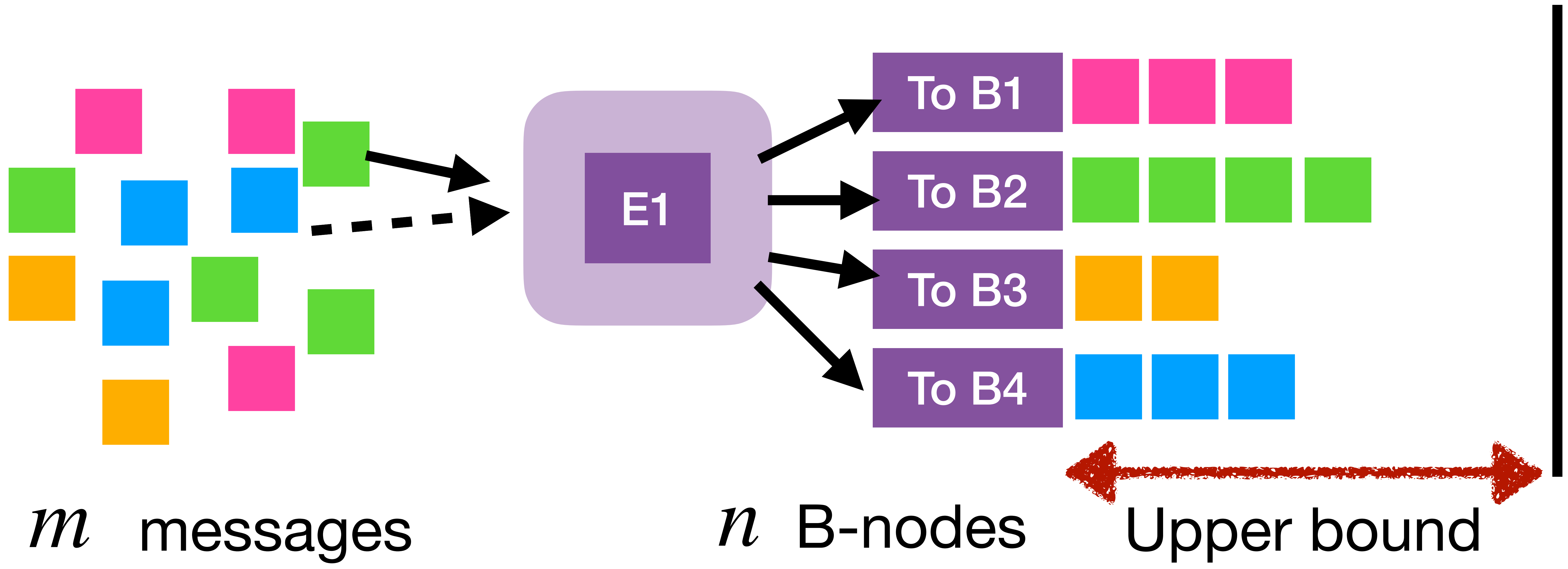
Setting a batch size for Boomerang+



M. Raab and A. Steger, “balls into bins” - A simple and tight analysis. *Proc. RANDOM*, 1998.

Setting a batch size for Boomerang+

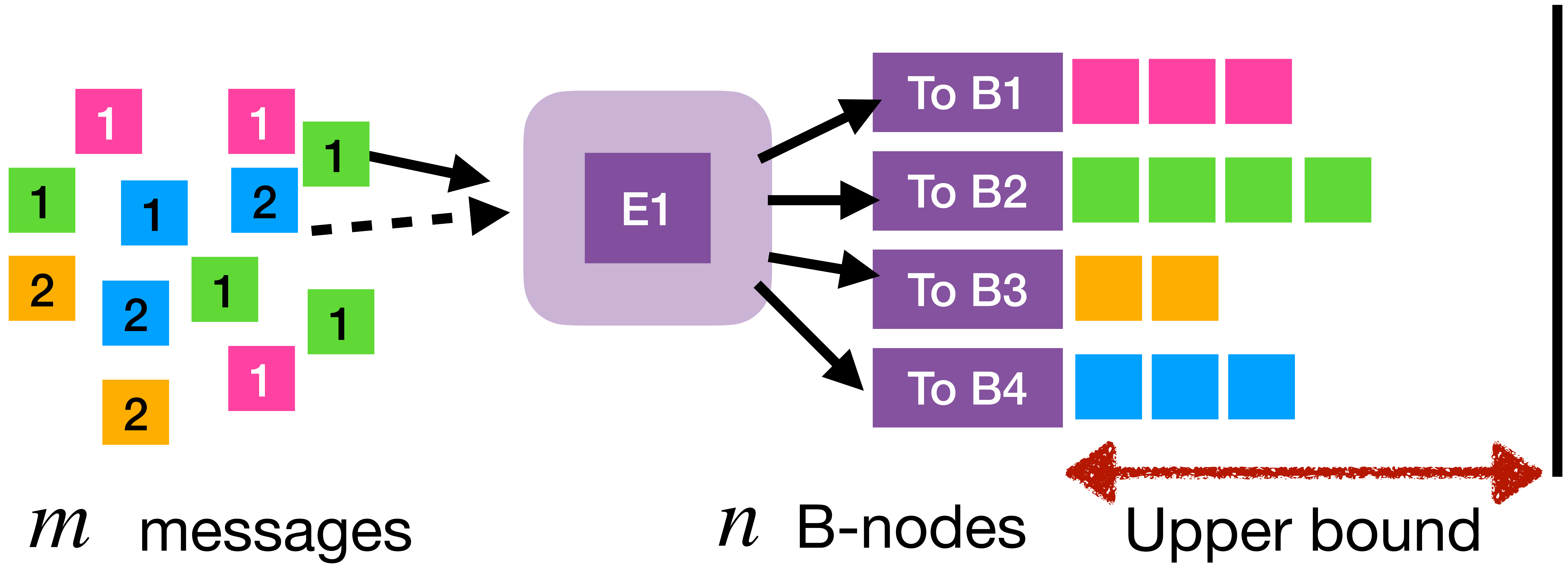
- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible



M. Raab and A. Steger, “balls into bins” - A simple and tight analysis. *Proc. RANDOM*, 1998.

Setting a batch size for Boomerang+

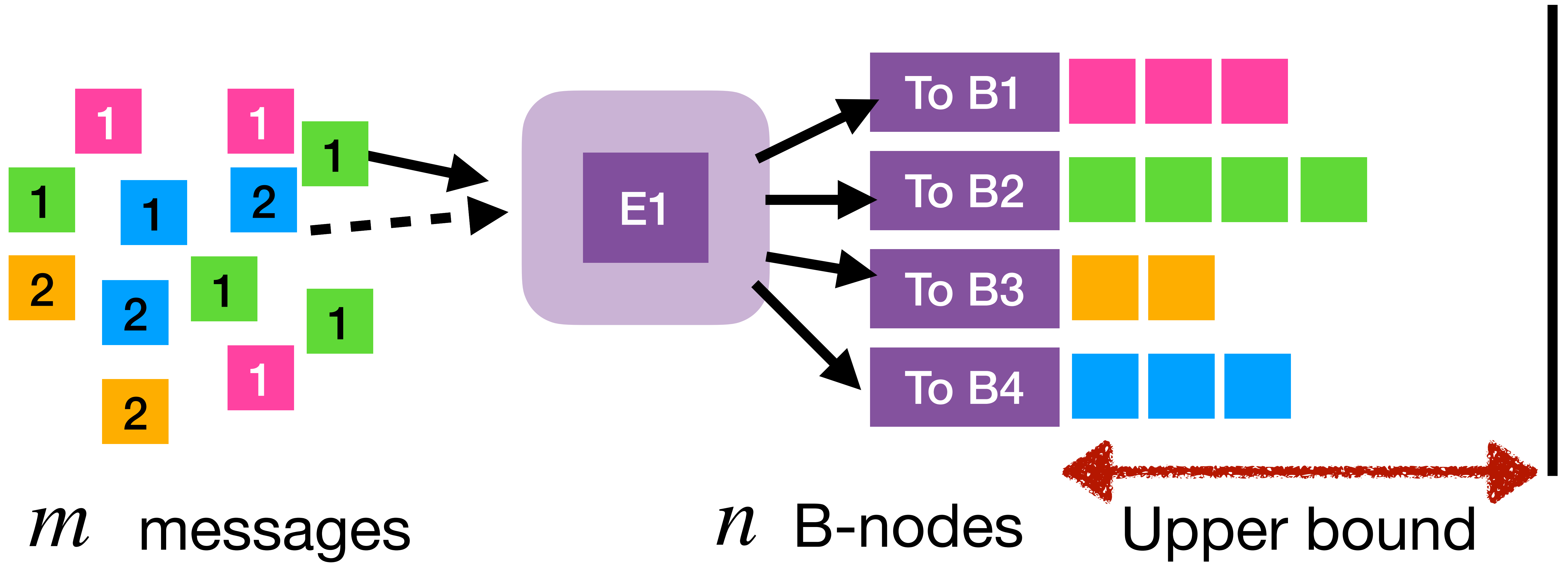
- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible



M. Raab and A. Steger, “balls into bins” - A simple and tight analysis. *Proc. RANDOM*, 1998.

Setting a batch size for Boomerang+

- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible
- Solution: **weighted balls-into-bins** game
  - Single, double, more-than-two (which we patch)



m messages

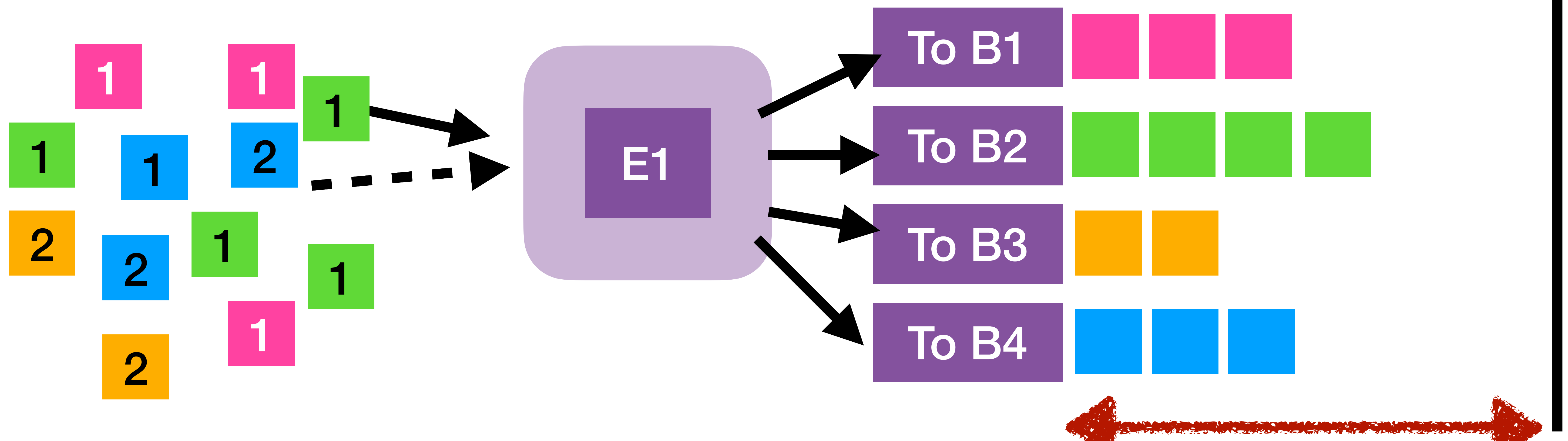
n B-nodes

Upper bound

M. Raab and A. Steger, "balls into bins" - A simple and tight analysis. *Proc. RANDOM*, 1998.

Setting a batch size for Boomerang+

- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible
- Solution: **weighted balls-into-bins** game
  - Single, double, more-than-two (which we patch)



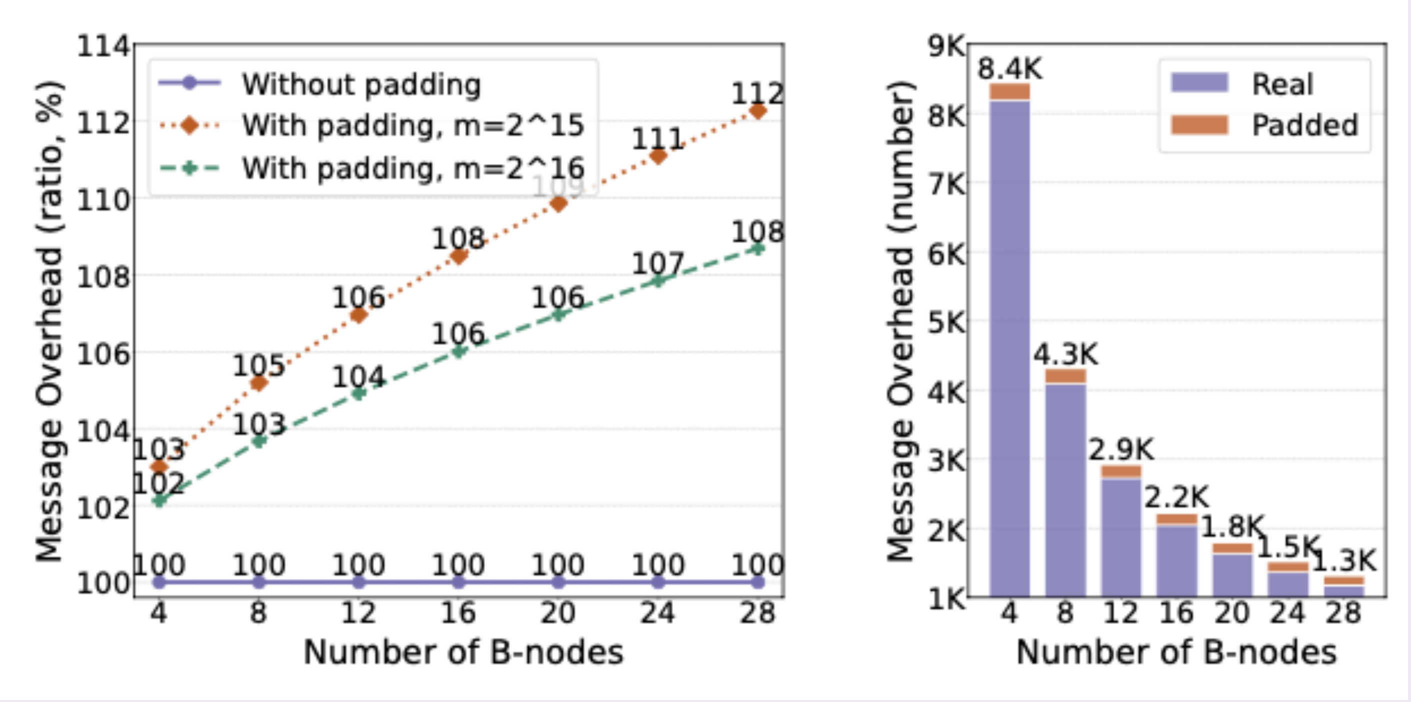
m messages

n B-nodes

Upper bound

Our results:

$$B = \left\lceil \frac{m}{n} + 4\sqrt{\frac{m \ln n}{3n}\left(1 - \frac{1}{\lambda}\frac{\ln \ln n}{2 \ln 2}\right)} \right\rceil$$



M. Raab and A. Steger, “balls into bins” - A simple and tight analysis. *Proc. RANDOM*, 1998.

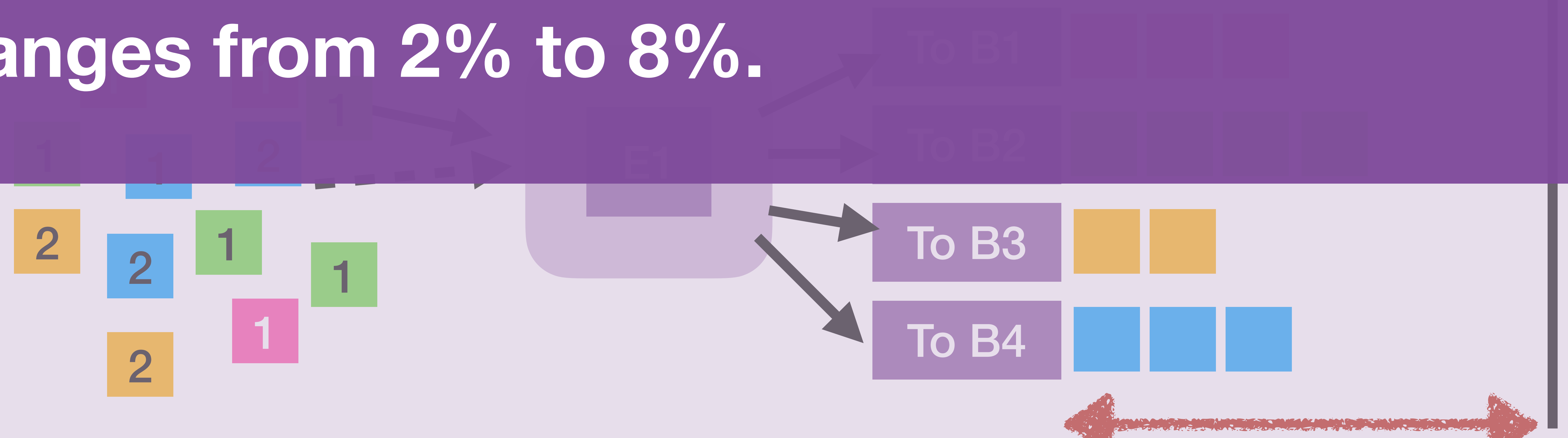
Setting a batch size for Boomerang+

- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible
- Solution: weighted balls-into-bins game
  - Single, double, more-than-two (which we patch)

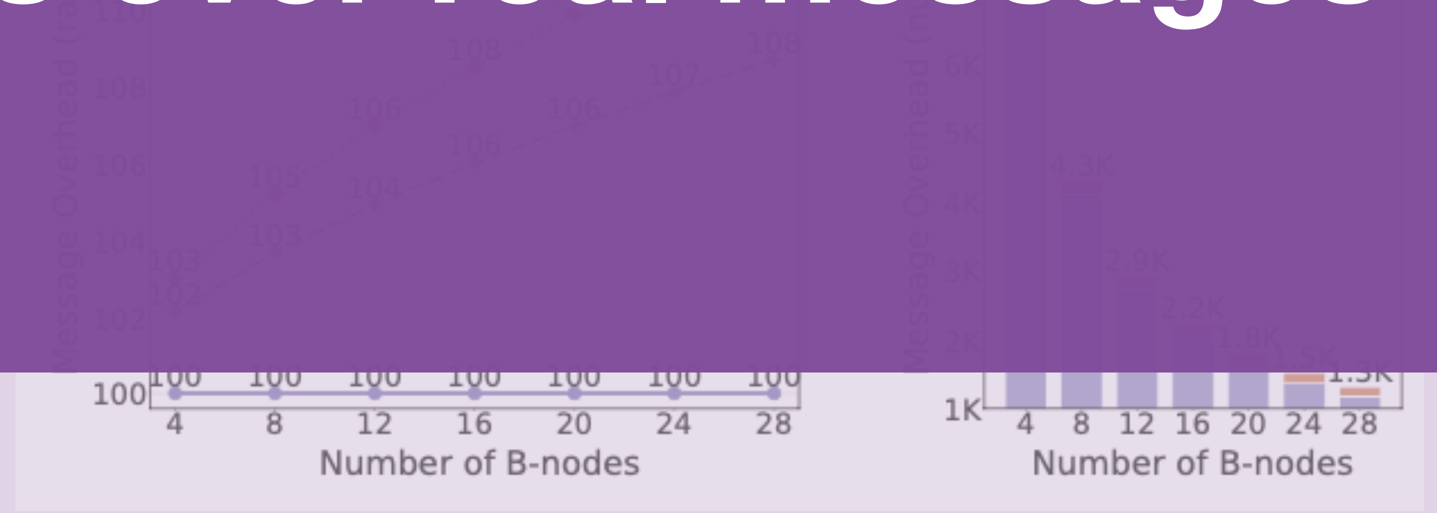
Our results:

$$B = \left\lceil \frac{m}{n} + 4\sqrt{\frac{m\ln n}{3n}\left(1 - \frac{1}{\lambda}\frac{\ln\ln n}{2\ln 2}\right)} \right\rceil$$

- **With λ = 128 and m = 2^16, when we scale the number of Boomerang nodes n from 4 to 28, the ratio of extra paddings over real messages ranges from 2% to 8%.**



m messages n B-nodes Upper bound



M. Raab and A. Steger, "balls into bins" - A simple and tight analysis. *Proc. RANDOM*, 1998.

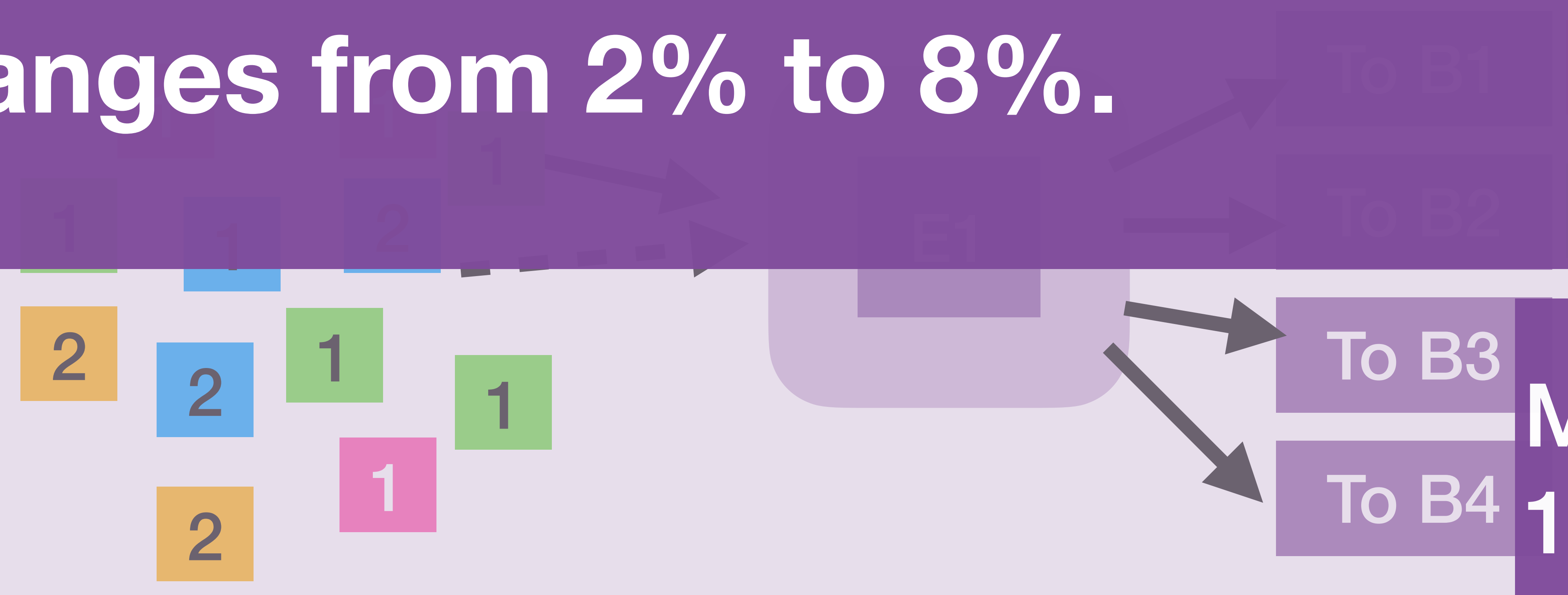
Setting a batch size for Boomerang+

- Requirements:
  - For security —> using public information for padding
  - For functionality —> no overflow
  - For efficiency —> as small as possible
- Solution: weighted balls-into-bins game
  - Single, double, more-than-two (which we patch)

Our results:

$$B = \left\lceil \frac{m}{n} + 4\sqrt{\frac{m \ln n}{3n}\left(1 - \frac{1}{\lambda}\frac{\ln\ln n}{2\ln 2}\right)}\right\rceil$$

- **With λ = 128 and m = 2^16, when we scale the number of Boomerang nodes n from 4 to 28, the ratio of extra paddings over real messages ranges from 2% to 8%.**



m messages

n B-nodes

More details:

1. Server churn
2. Oblivious padding and batching
3. Message exchange on Boomerang nodes

M. Raab and A. Steger, “balls into bins” - A simple and tight analysis. *Proc. RANDOM*, 1998.

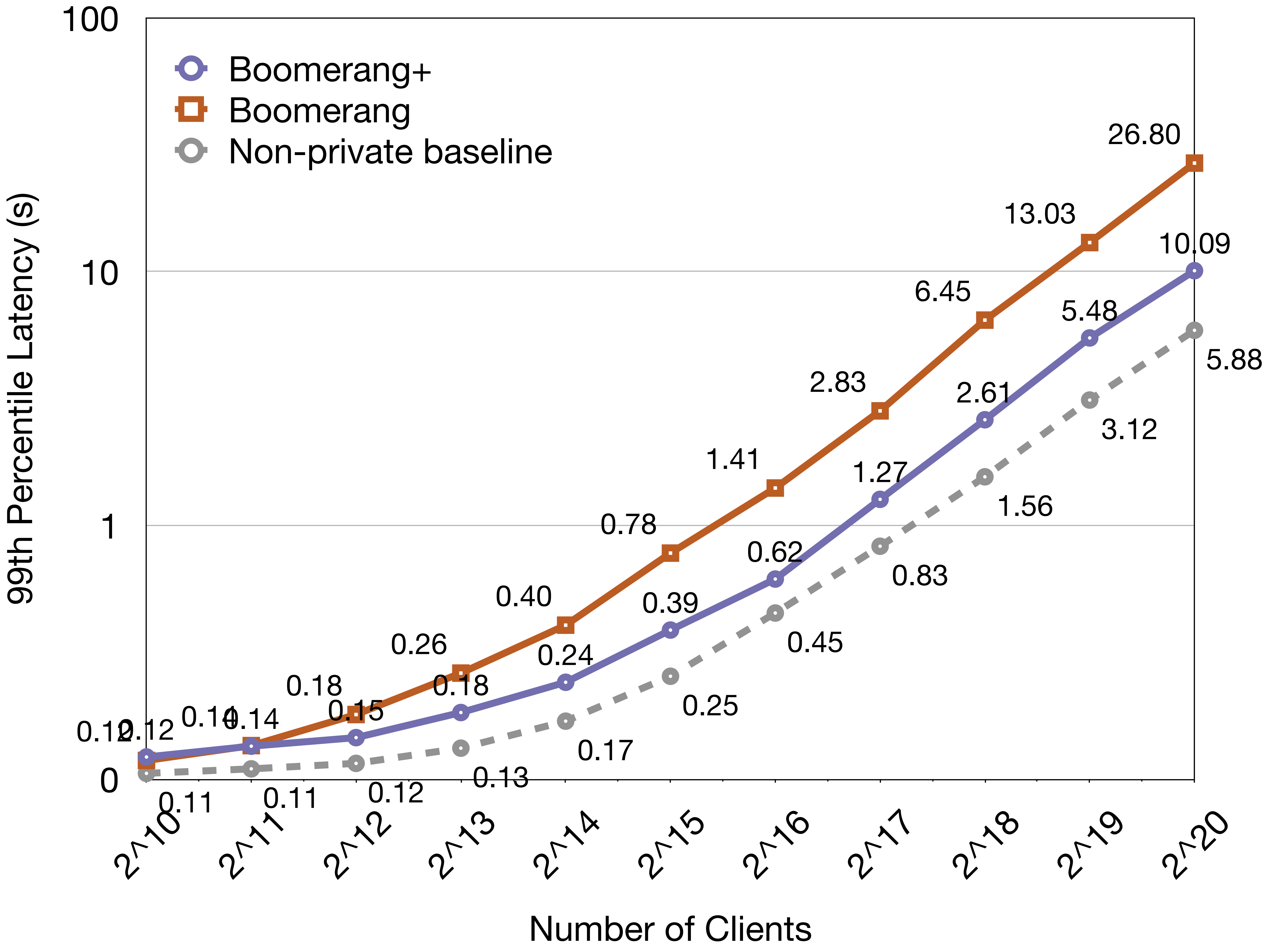
Evaluation

- How fast are Boomerang and Boomerang+?
- Can Boomerang+ scale by adding servers?

Implementation

- ~4,000 lines of C++ code
- 16 M6ce.4XLARGE128 instances with Intel Xeon Ice Lake processors with Intel SGX support (16 vCPU, 128 GB of memory, and 13 Gbps of network bandwidth)
- Compared with 3 existing systems with crypto guarantee (Pung, Addra, XRD)

Evaluation: how fast are Boomerang(+)?



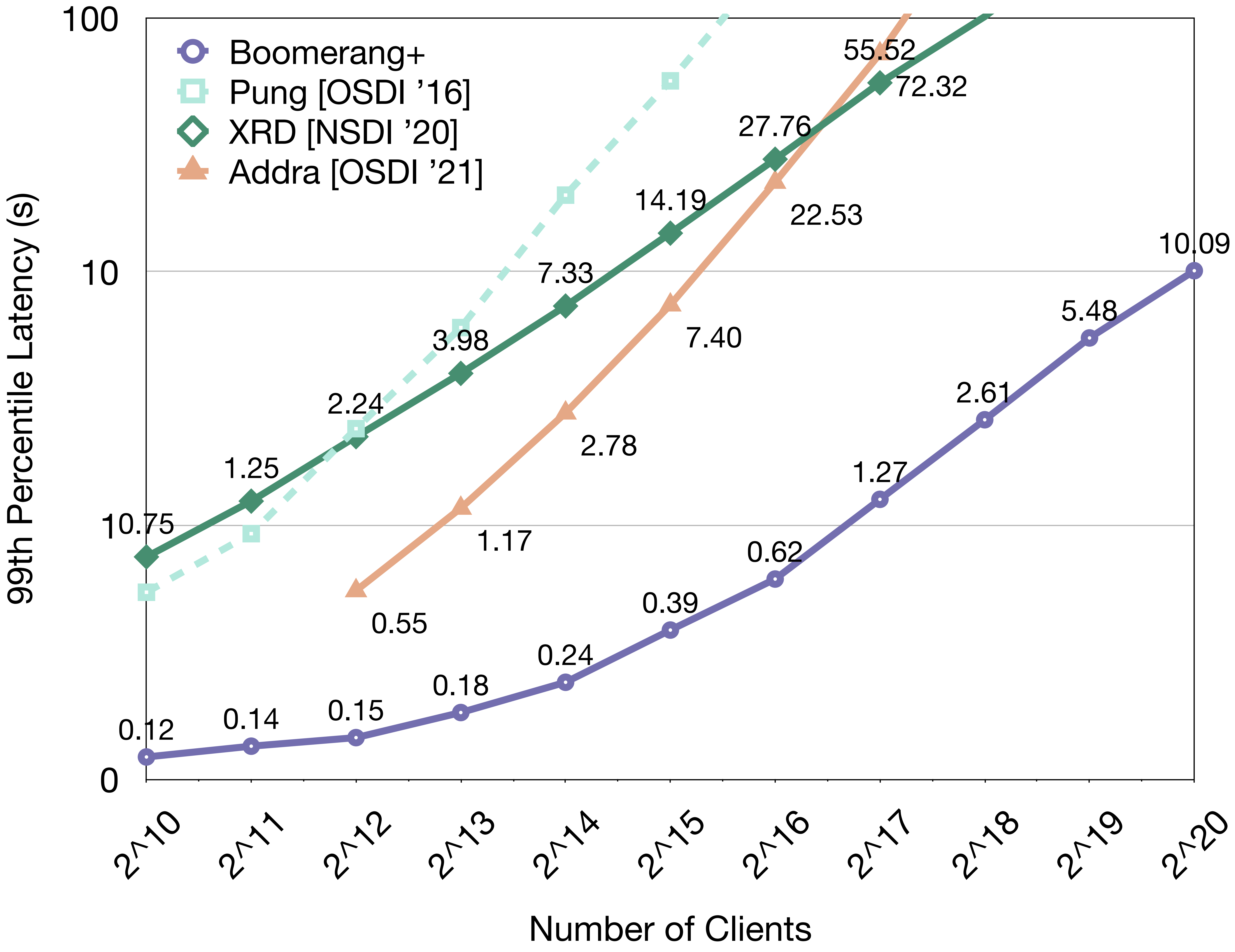
Boomerang+ (on 16 servers):

- For 2^16 clients, latency = 0.62s
- For 2^20 clients, latency = 10.09s

Boomerang (on 1 server):

- For 2^16 clients, latency = 1.41s
- For 2^20 clients, latency = 26.80s

Evaluation: how fast are Boomerang(+)?

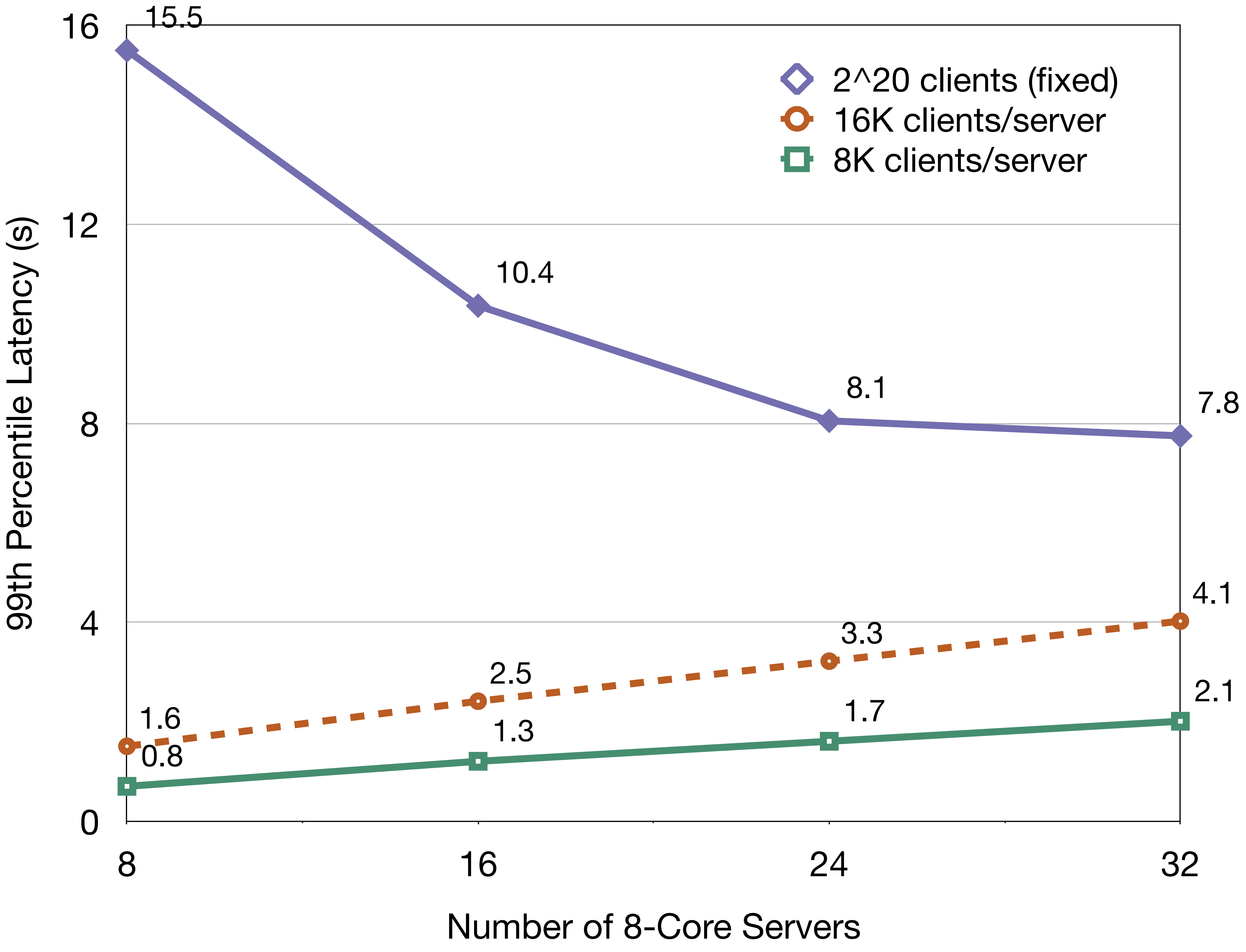


Try our best to make the comparison fair:

keep the same budget for cloud servers.

As expected, Boomerang+ runs faster than crypto-based systems, thanks to the trusted hardware

Evaluation: can Boomerang+ scale by adding servers?



Boomerang+ can reduce latency by adding **more servers**:

- **16 servers**: **10.4s** over 2^20 clients
- **32 servers**: **7.8s** over 2^20 clients

Boomerang+ can remain low latency while **keeping a constant per server workload**:

- **8K clients/server**: **0.8s~2.1s**
- **16K clients/server**: **1.6s~4.1s**

Takeaways

- Enclave-based metadata-private messaging for low cost and accessibility to the mass
  - Oblivious detection and patching algorithms (boomerang trick)
  - A scalable design with oblivious batching algorithms
  - Code available at https://github.com/CongGroup/boomerang

- Future work
  - A planner that can adaptively allocate the entry nodes and B-nodes
  - Boomerang as a backend mixing network for other applications

Takeaways

- Enclave-based metadata-private messaging for low cost and accessibility to the mass
  - Oblivious detection and patching algorithms (boomerang trick)
  - A scalable design with oblivious batching algorithms
  - Code available at https://[illegible]/boomerang

Thank you! Questions?

pp.jiang@my.cityu.edu.hk

- Future work
  - A planner that can adaptively allocate the entry nodes and B-nodes
  - Boomerang as a backend mixing network for other applications