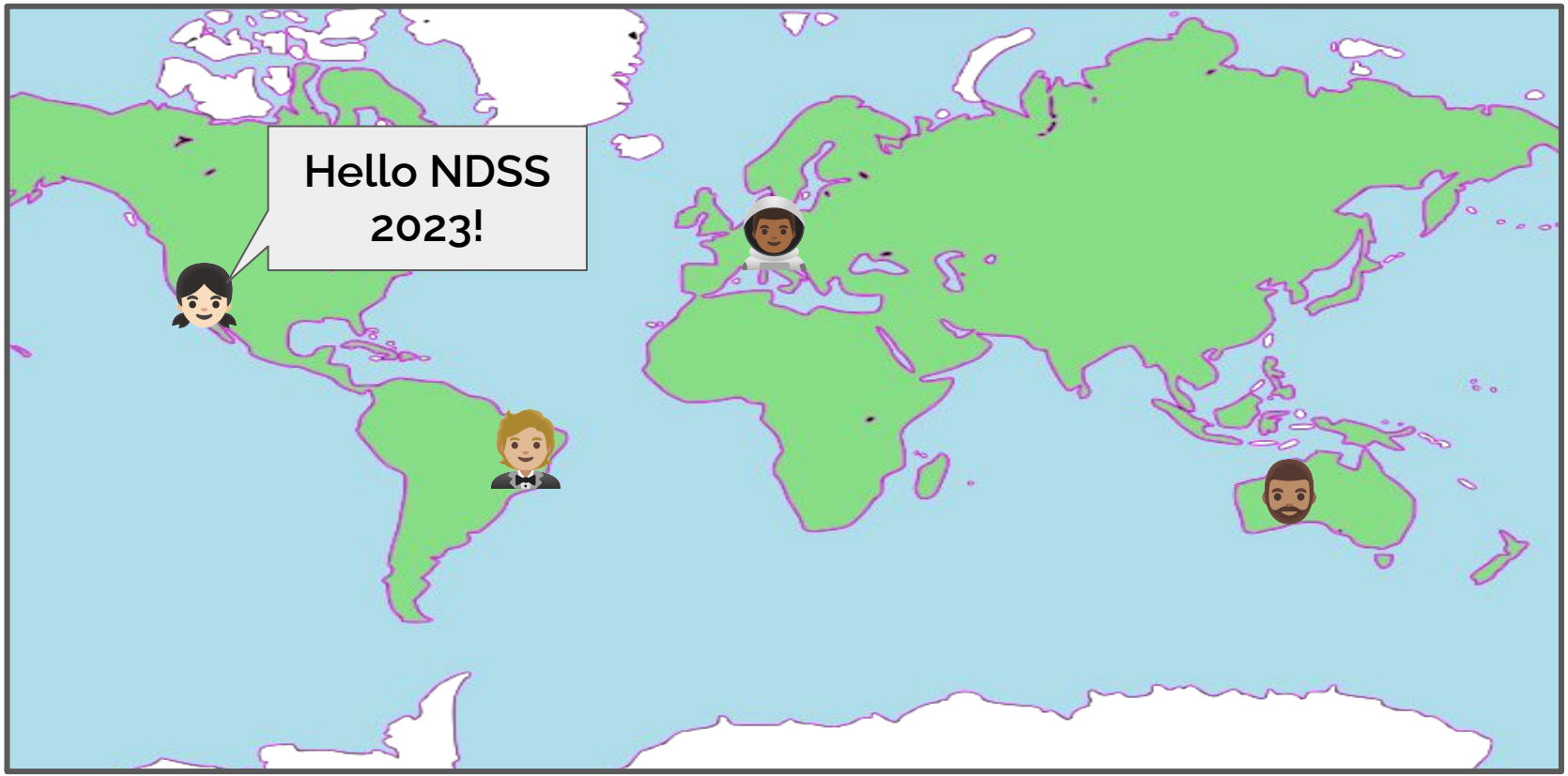
Trellis: Robust and Scalable Metadata Private Anonymous Broadcast

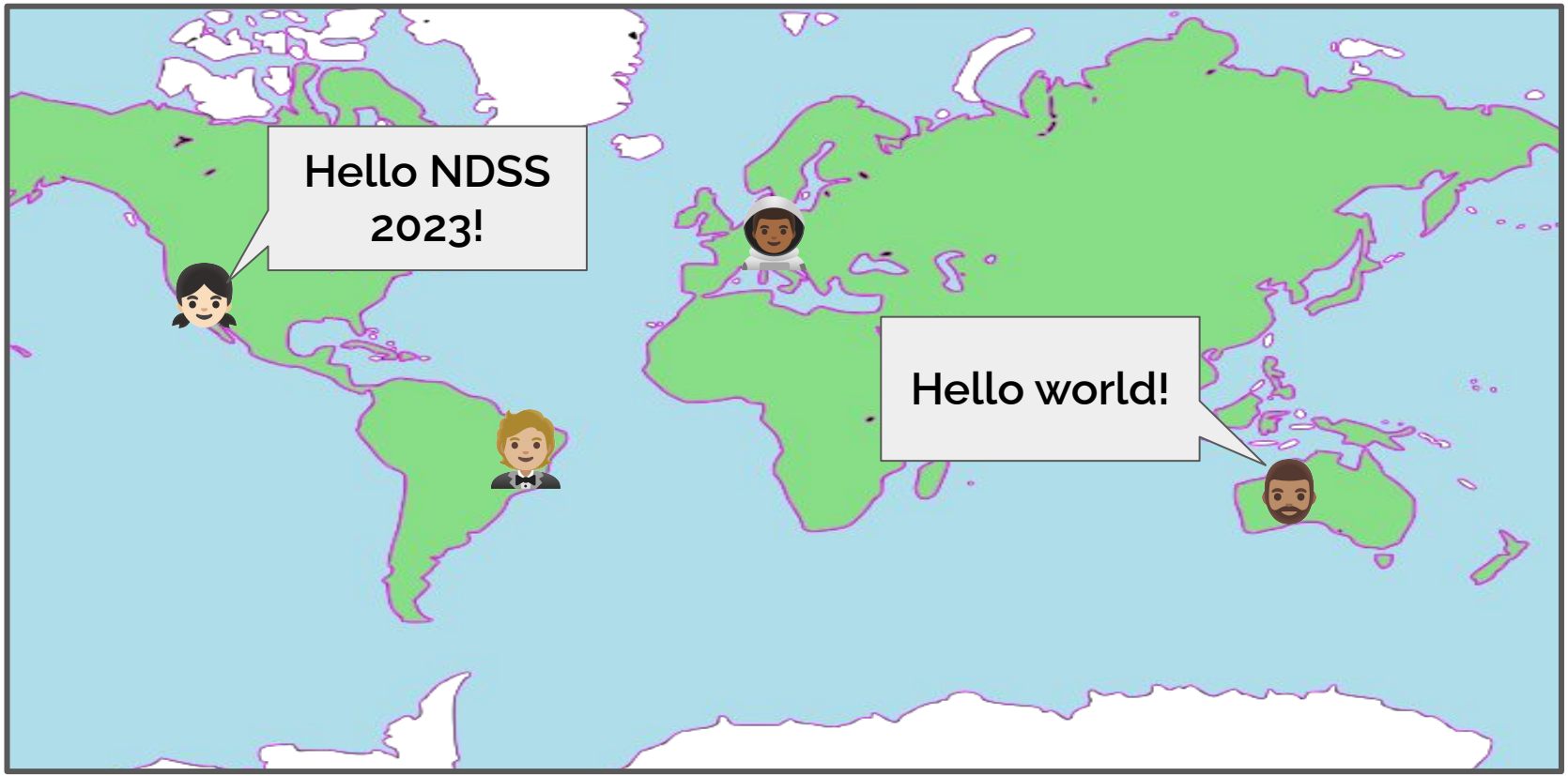
Simon Langowski, Sacha Servan-Schreiber, and Srini Devadas

MIT CSAIL

Broadcast

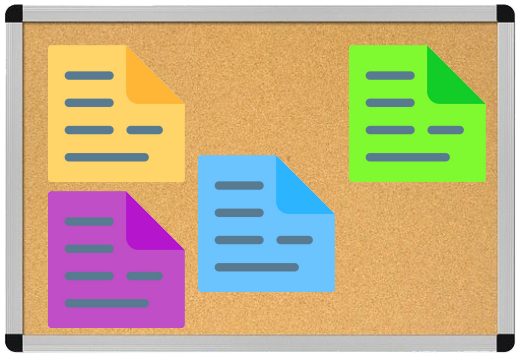


Broadcast



Broadcast: A public bulletin board…

Broadcast: A public bulletin board...



Broadcast: …or an online forum

User297: I really like potatoes.

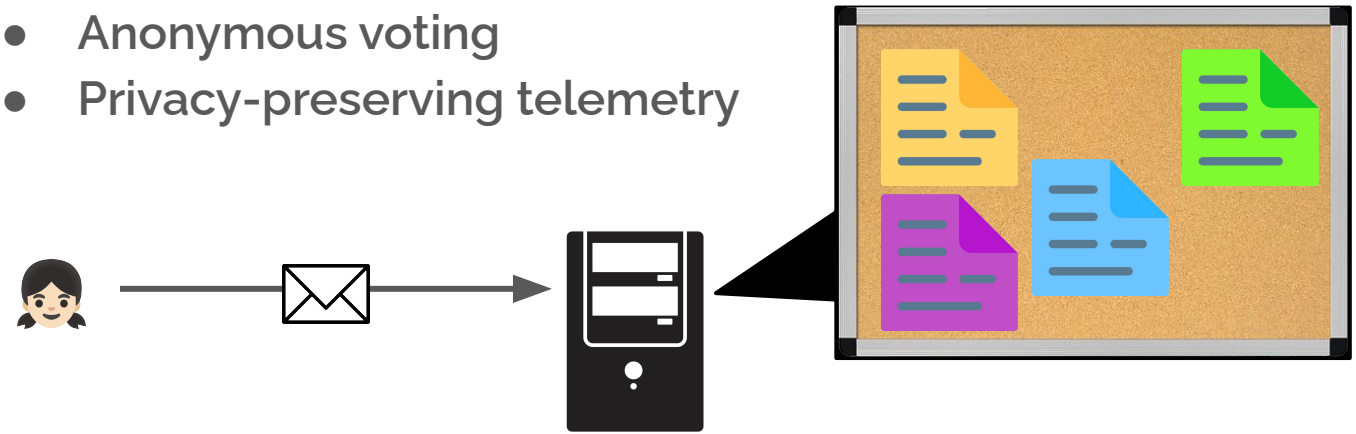
Iguana25: Carrots are better.

thellama: Why not both?

Anonymous broadcast

Anonymous broadcast is a building block in privacy-preserving systems

- Anonymous shuffling
- Anonymous communication
- Anonymous voting
- Privacy-preserving telemetry

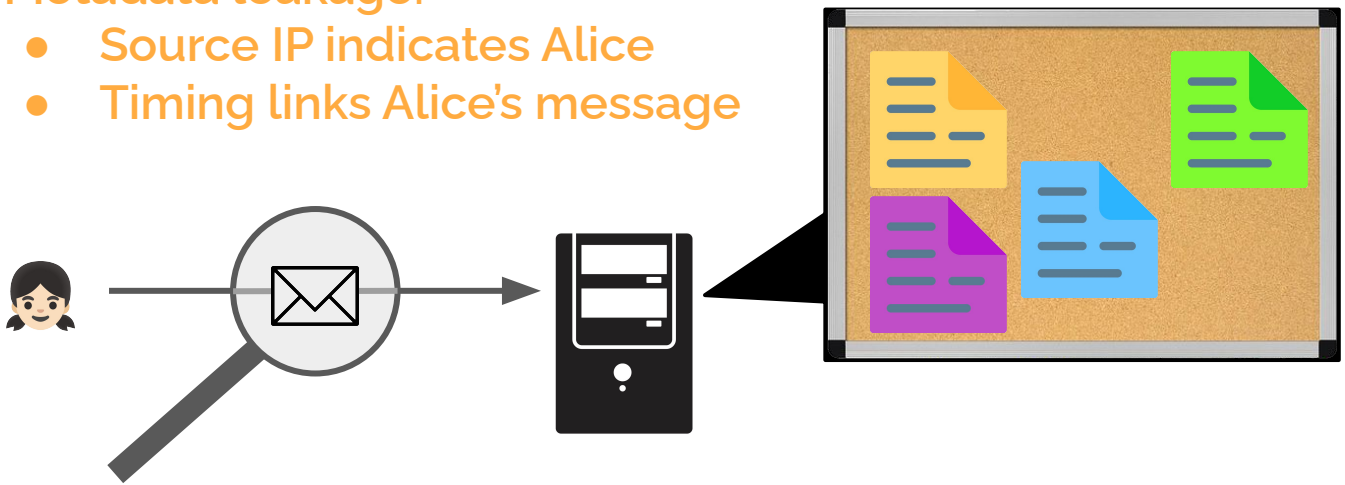


Anonymity vs. metadata

- Forum is hosted on a server
- Obvious from traffic **metadata** which message belongs to which user

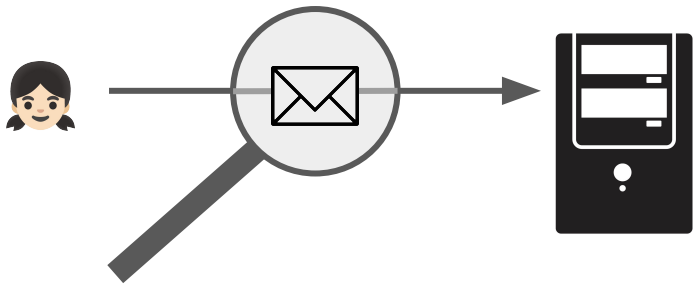
Metadata leakage:

- Source IP indicates Alice
- Timing links Alice's message



Metadata privacy

- Message **contents** can be protected by encryption and authentication (end-to-end encryption with TLS).
- But **metadata** is leaked to network observers:



Types of Metadata

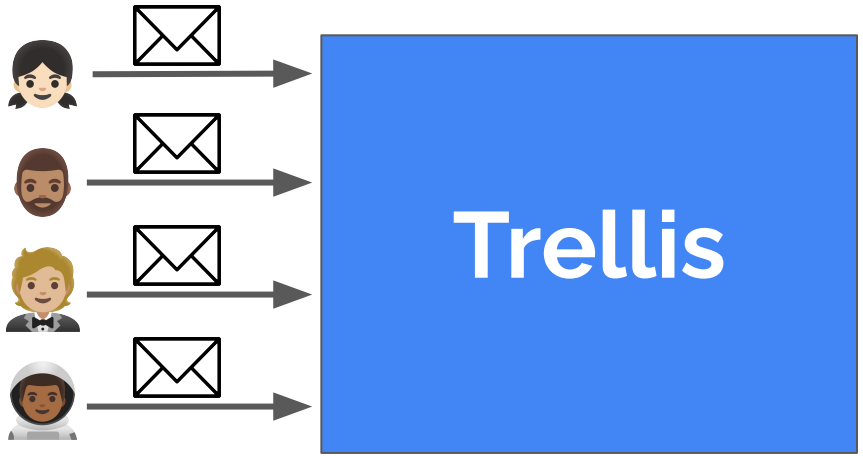
- Time sent,
- Source IP,
- Message size,
- etc...

Trellis: An anonymous broadcast system

Senders

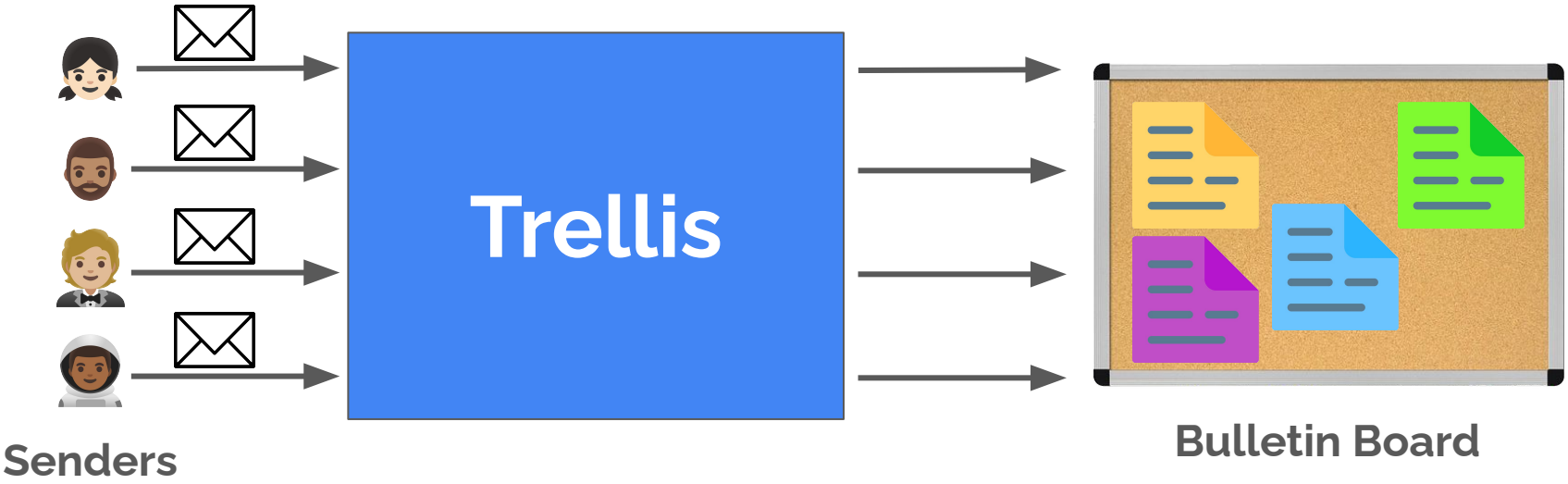
Trellis

Trellis: An anonymous broadcast system

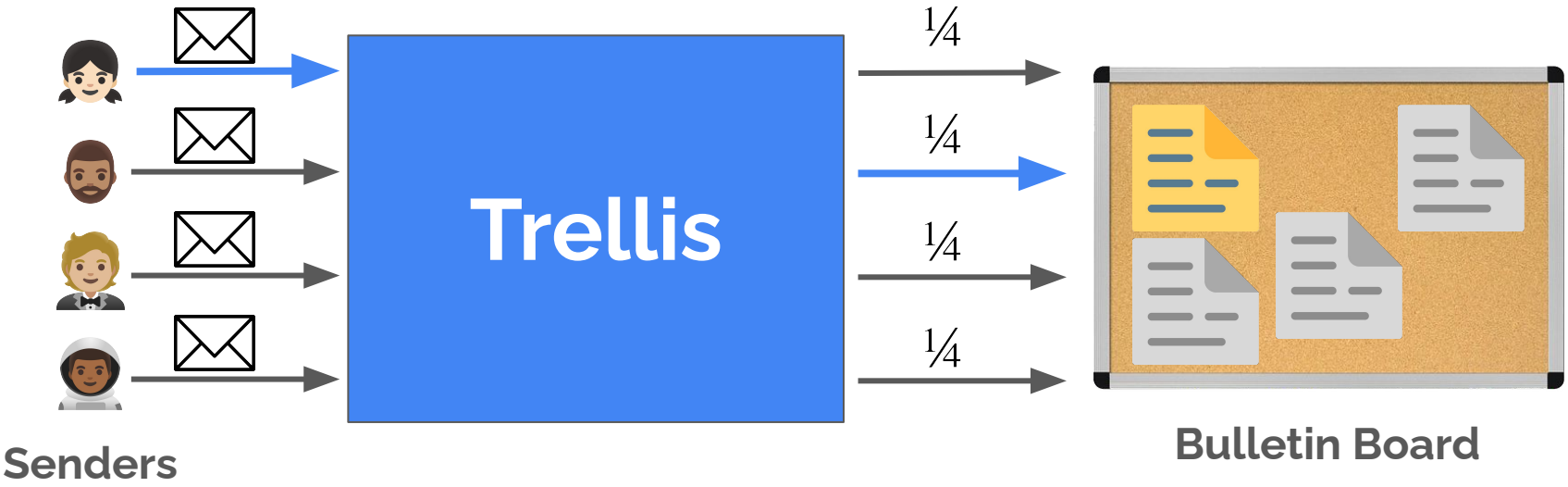


Senders

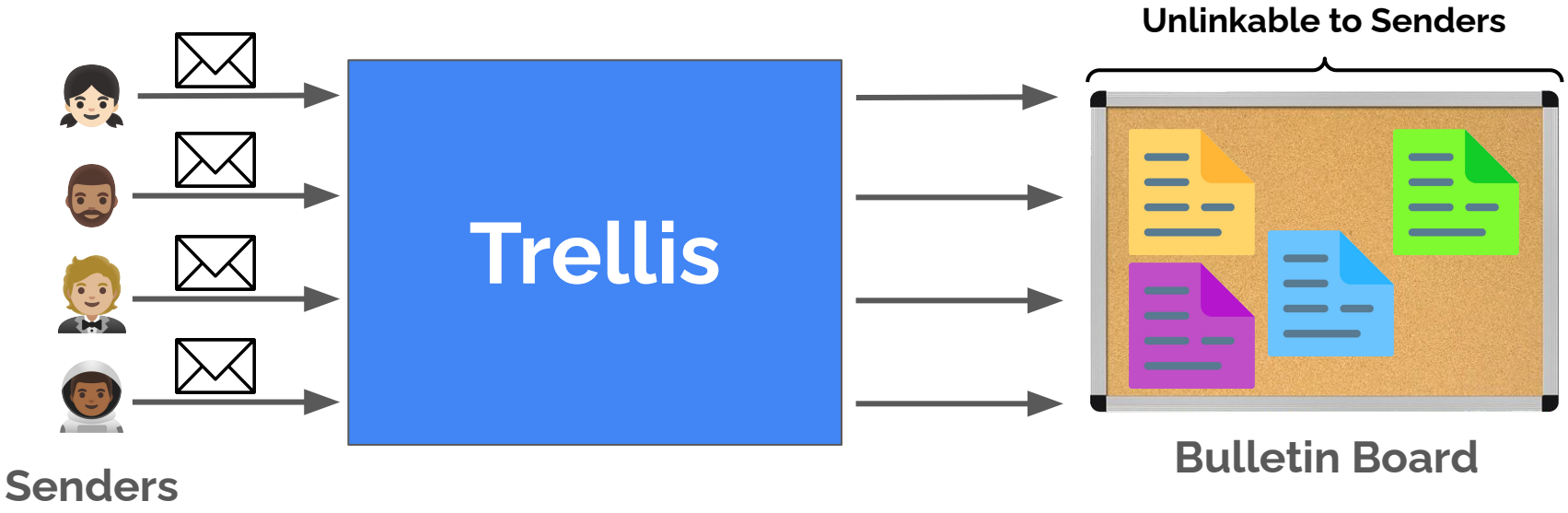
Trellis: An anonymous broadcast system



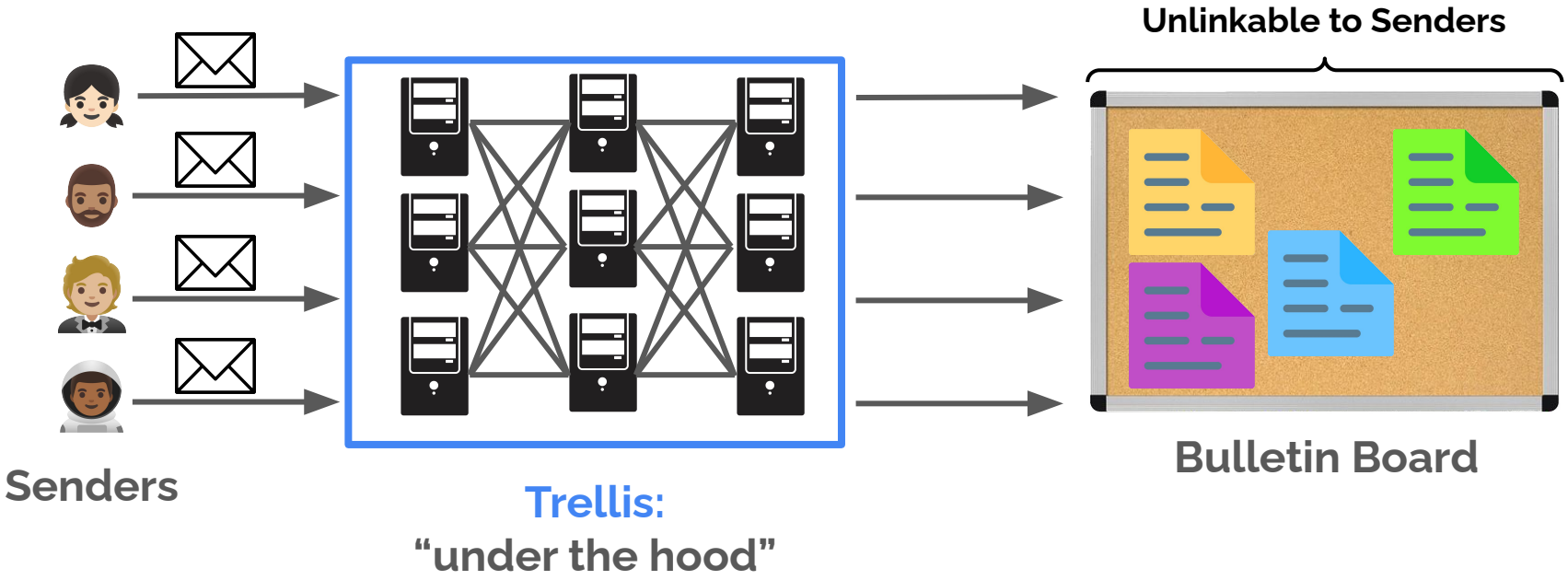
Trellis: An anonymous broadcast system



Trellis: An anonymous broadcast system

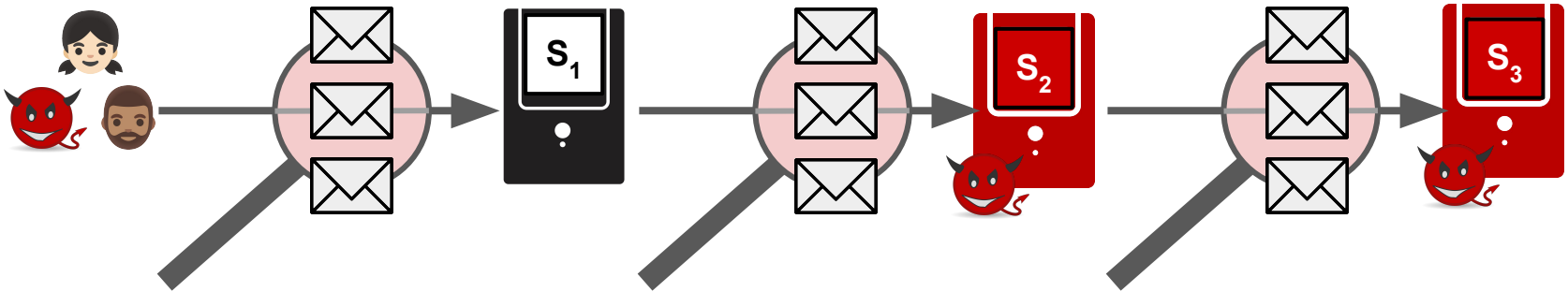


Trellis: An anonymous broadcast system



Adversary model

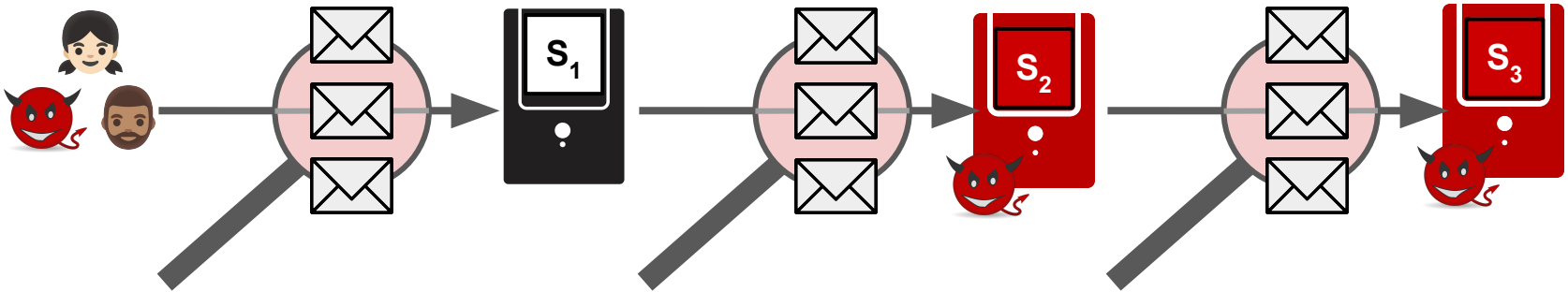
- The adversary controls a constant fraction *f* of the servers in Trellis
- The adversary can observe all network traffic
- Some number of colluding (sybil) users work with the adversary.



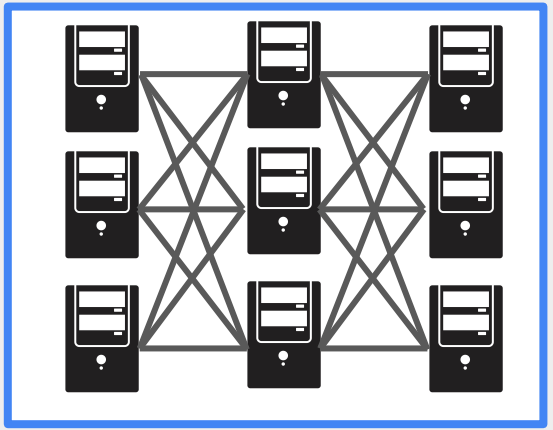
Adversary model

f=0.3

- The adversary controls a constant fraction *f* of the servers in Trellis
- The adversary can observe all network traffic
- Some number of colluding (sybil) users work with the adversary.

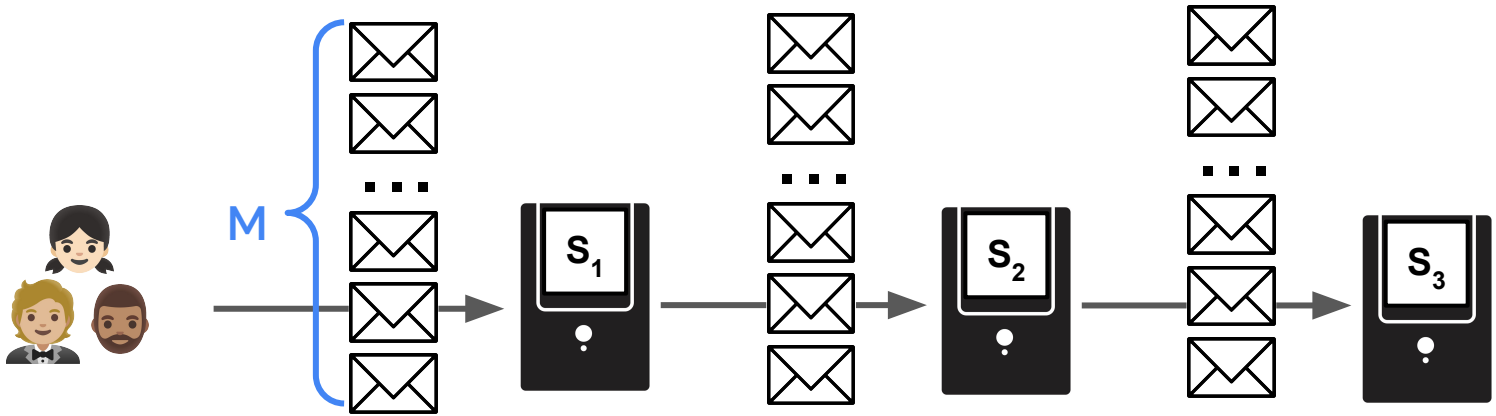


Constructing Trellis



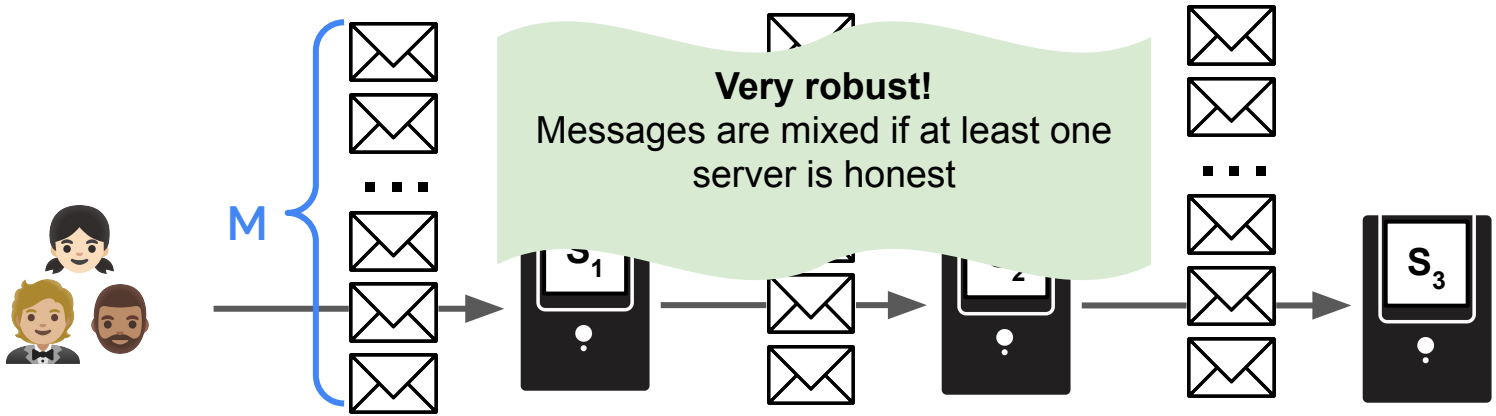
Starting point: Mixnets [Chaum'81]

- N servers arranged in a chain,
- Each server shuffles and forwards the M messages to the next server
- Doesn't scale: each server has O(M) work and O(M) communication!



Starting point: Mixnets [Chaum'81]

- N servers arranged in a chain,
- Each server shuffles and forwards the M messages to the next server
- Doesn't scale: each server has O(M) work and O(M) communication!



Desired Properties

Desired Properties

- Robustness

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial

- Scalability

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial

- Scalability
  - Performance improves with more servers

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial

- Scalability
  - Performance improves with more servers
  - Scale linearly with the number of messages

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial
- Scalability
  - Performance improves with more servers
  - Scale linearly with the number of messages
- Metadata-privacy

Desired Properties

- Robustness
  - All messages are posted even if servers are adversarial
- Scalability
  - Performance improves with more servers
  - Scale linearly with the number of messages
- Metadata-privacy
  - Anonymity even when the network is observed

Desired Properties

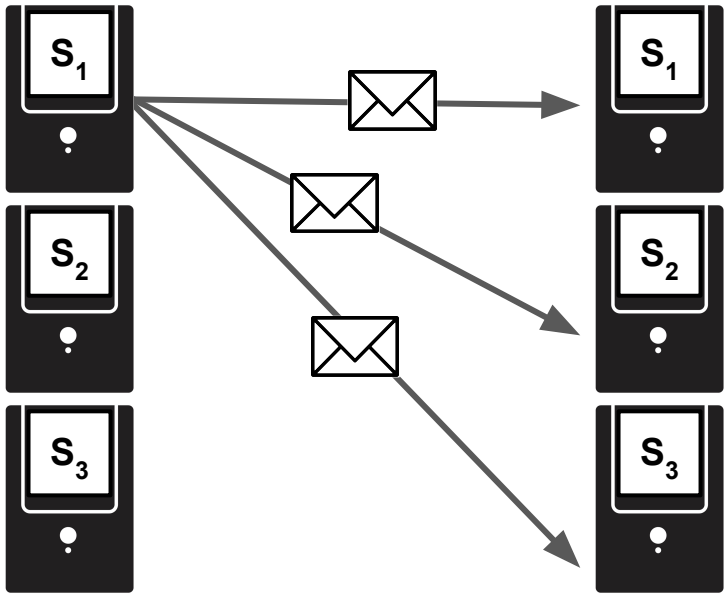
- Robustness
  - All messages are posted even if servers are adversarial
- Scalability
  - Performance improves with more servers
  - Scale linearly with the number of messages

} Often in conflict!

- Metadata-privacy
  - Anonymity even when the network is observed

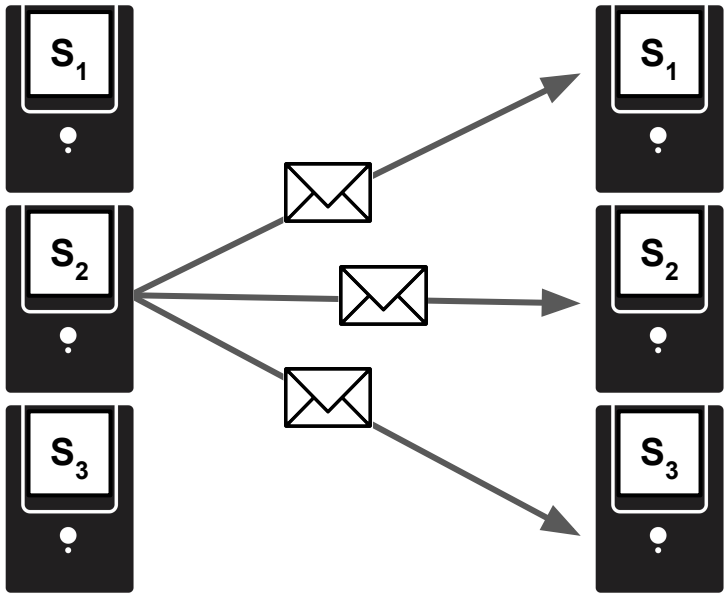
Scalable but not robust: Parallel Mixnets [GJ'04]

Scalability: Each server divides the envelopes among *all* the servers



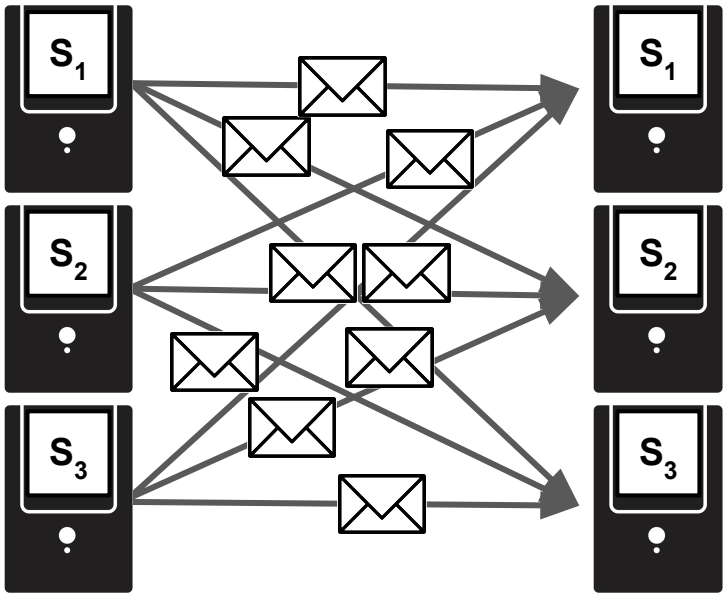
Scalable but not robust: Parallel Mixnets [GJ'04]

Scalability: Each server divides the envelopes among *all* the servers



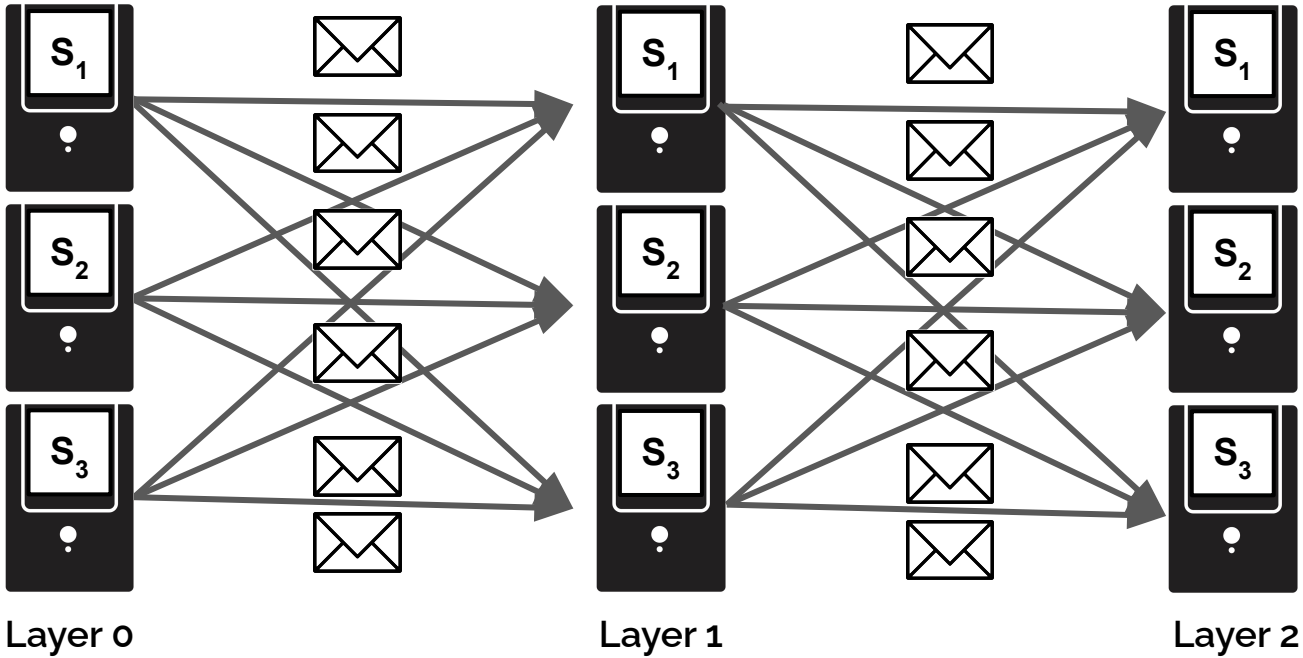
Scalable but not robust: Parallel Mixnets [GJ'04]

Scalability: Each server divides the envelopes among *all* the servers



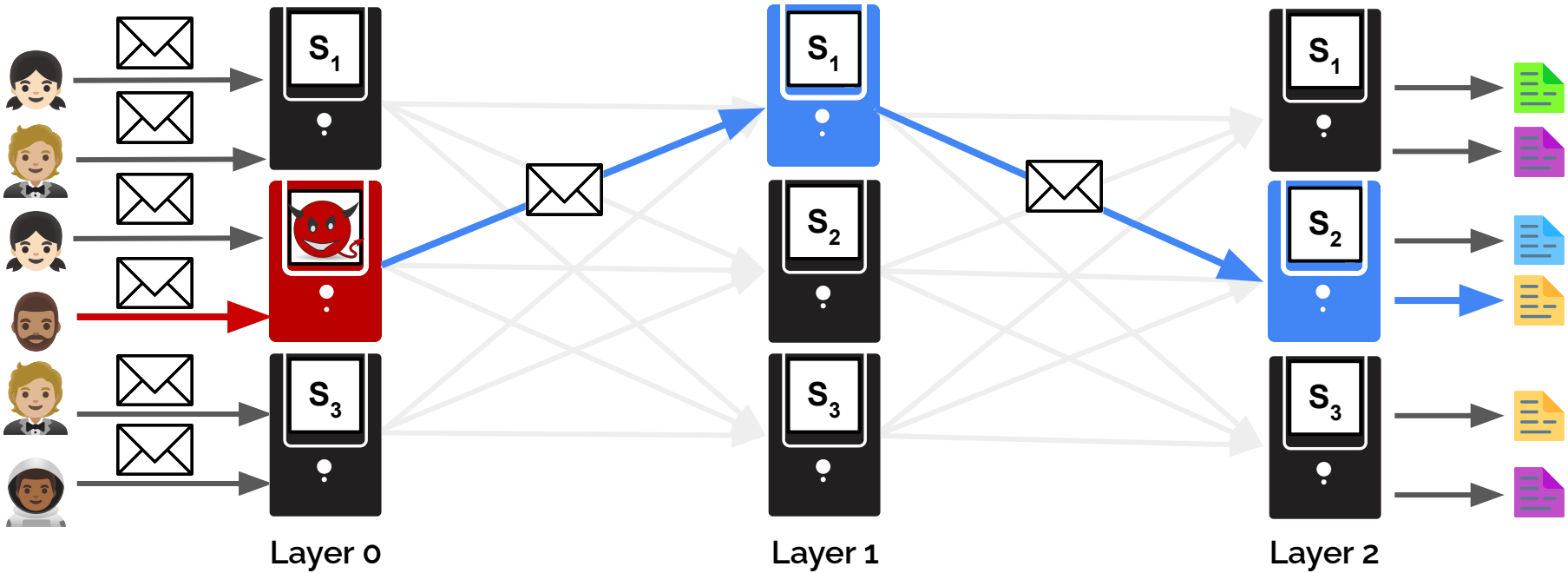
Scalable but not robust: Parallel Mixnets [GJ'04]

Scalability: Each server divides the envelopes among *all* the servers



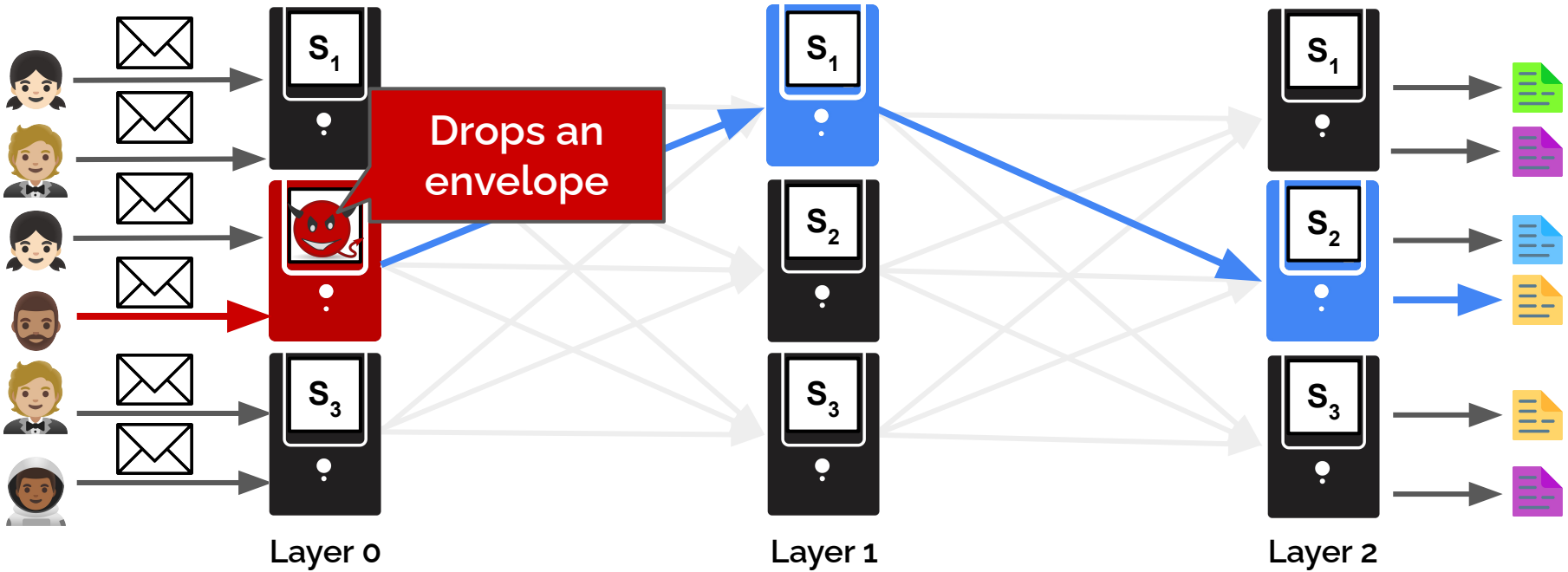
Scalable but not robust: Parallel Mixnets

Problem: Adversary controlled paths



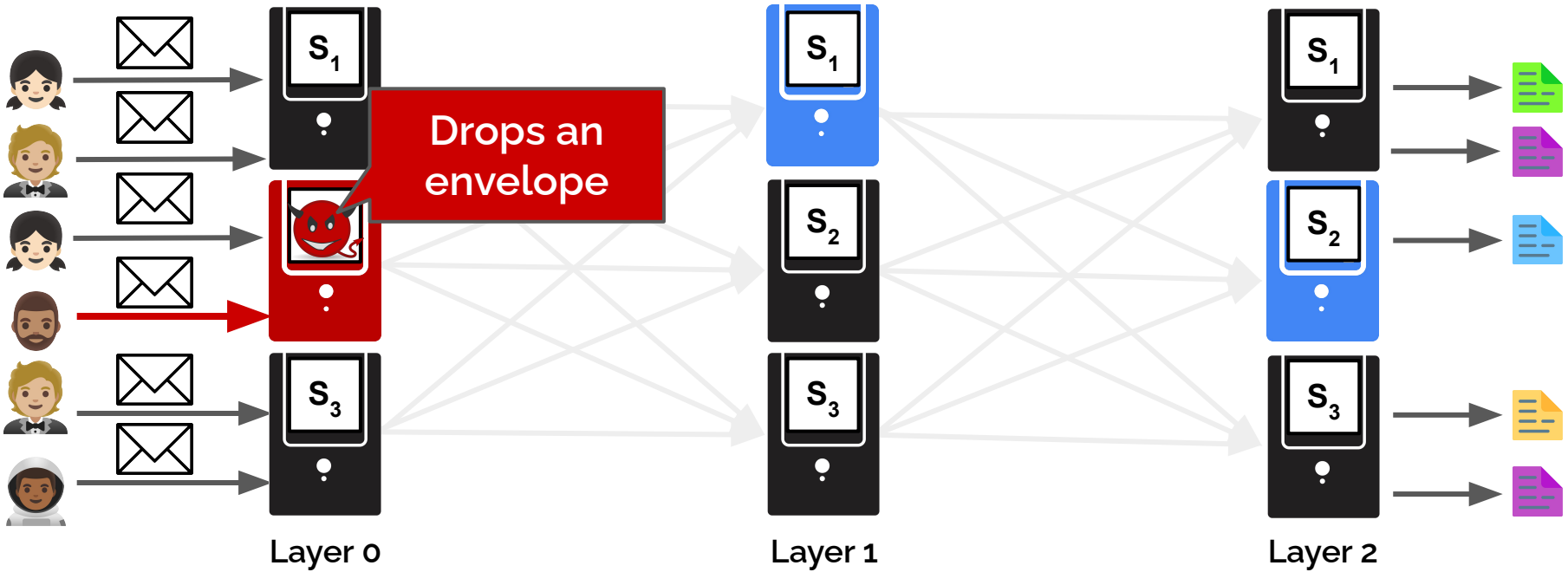
(Covertly) Dropped Messages

Problem: Adversary controlled paths



(Covertly) Dropped Messages

Problem: Adversary controlled paths



New tool: Anonymous routing tokens (ART)

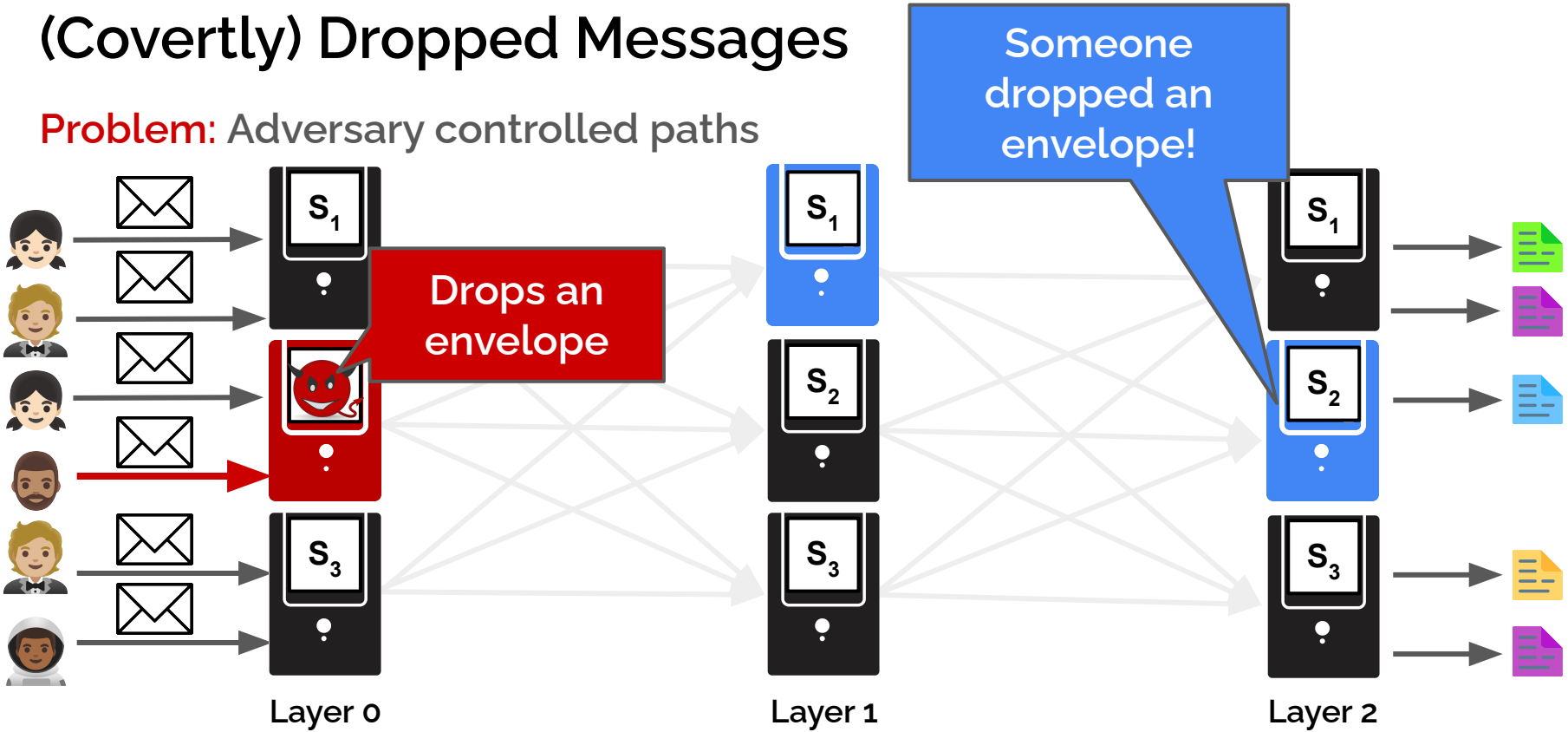
- Allows servers to know which envelopes to expect
- Servers anonymously check that an envelope was received from each assigned user

ART Tokens				
Token ID	**Encryption key**	**Verification key**	**Forwarding server**	**Received?**
14	0x13a	0xfdf9130	3	Yes
55	0x41f	0x12ea0	1	Yes
39	0x556	0xbbc908	3	**No**

Tokens are random ⇒ unlinkable to user

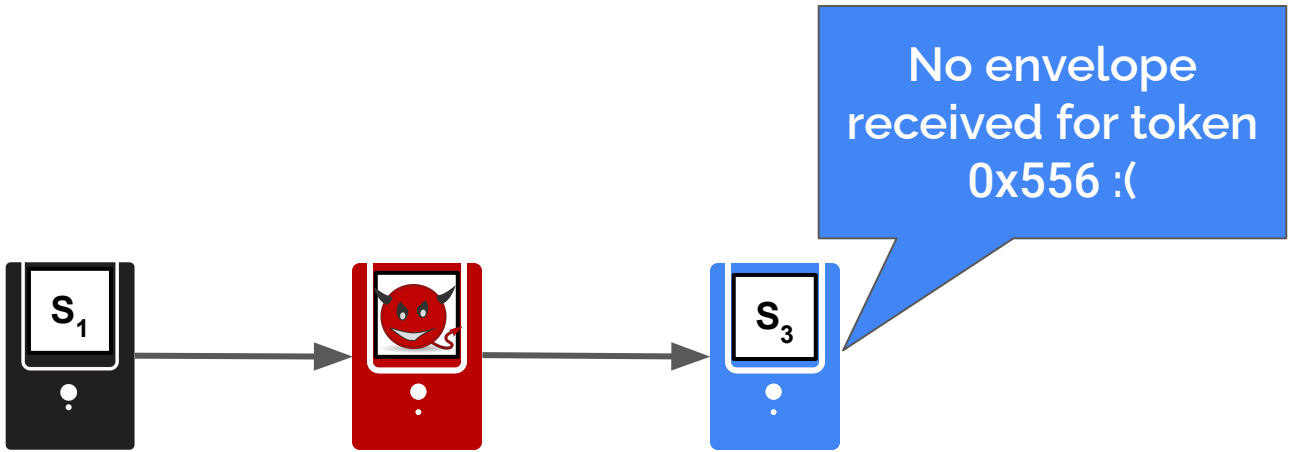
(Covertly) Dropped Messages

Problem: Adversary controlled paths



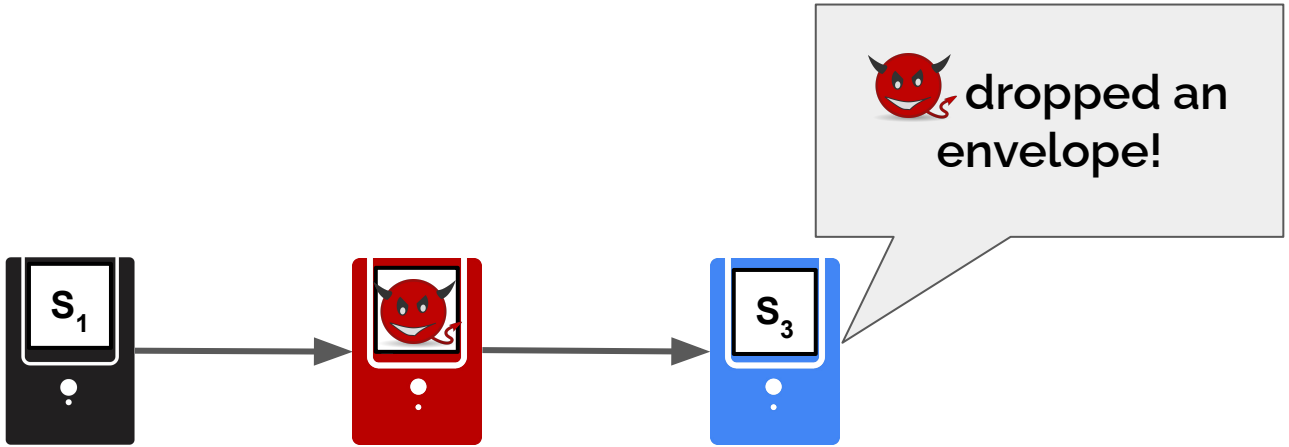
Robustness in Trellis using ARTs

- Deviations are always caught at the next honest server



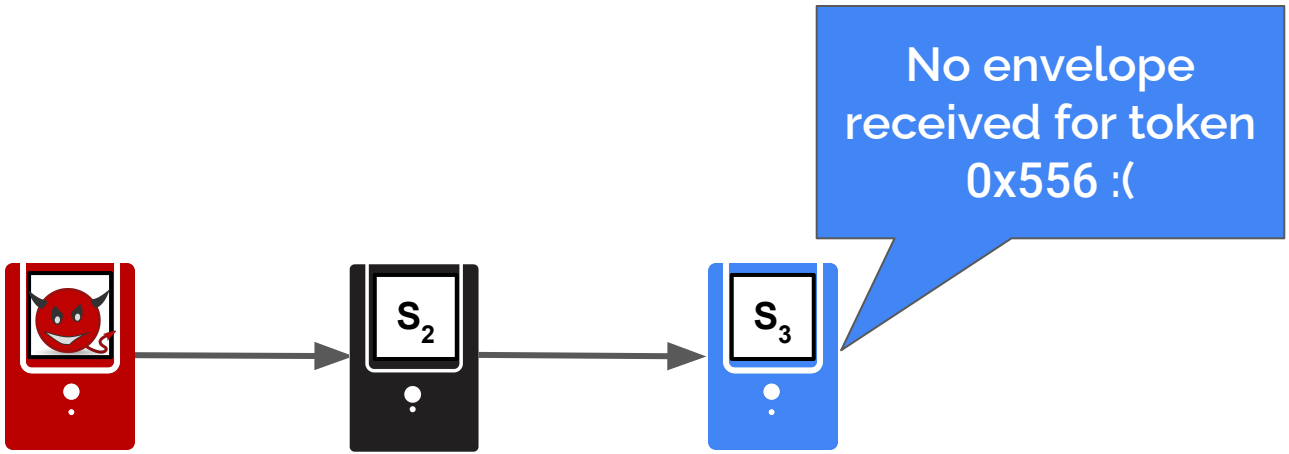
Robustness in Trellis using ARTs

- Deviations are always caught at the next honest server
- Blame previous server for being malicious!



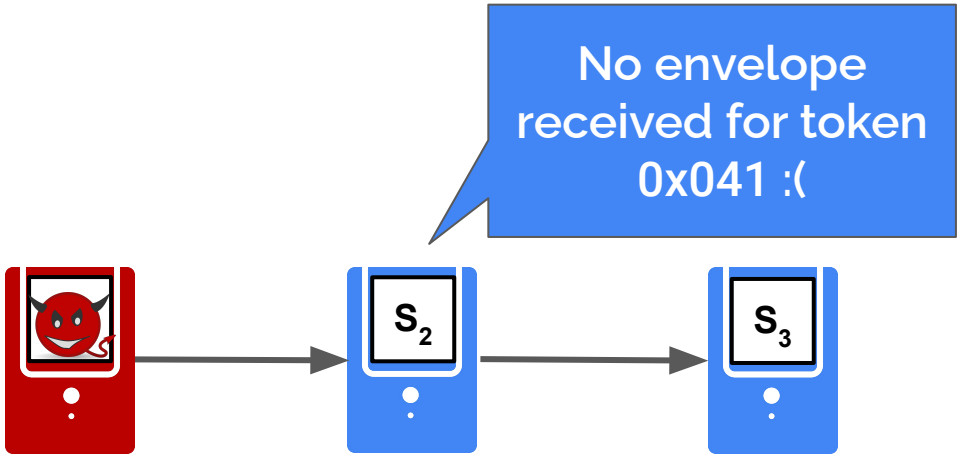
Robustness in Trellis using ARTs

- Deviations are always caught at the next honest server



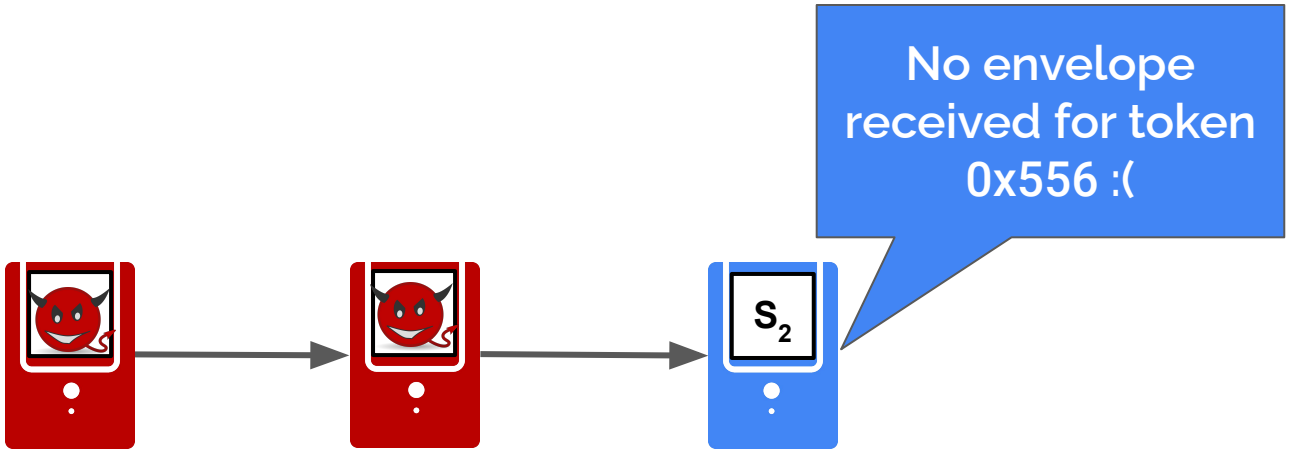
Robustness in Trellis using ARTs

- Deviations are always caught at the next **honest** server
- Honest server should have reported earlier



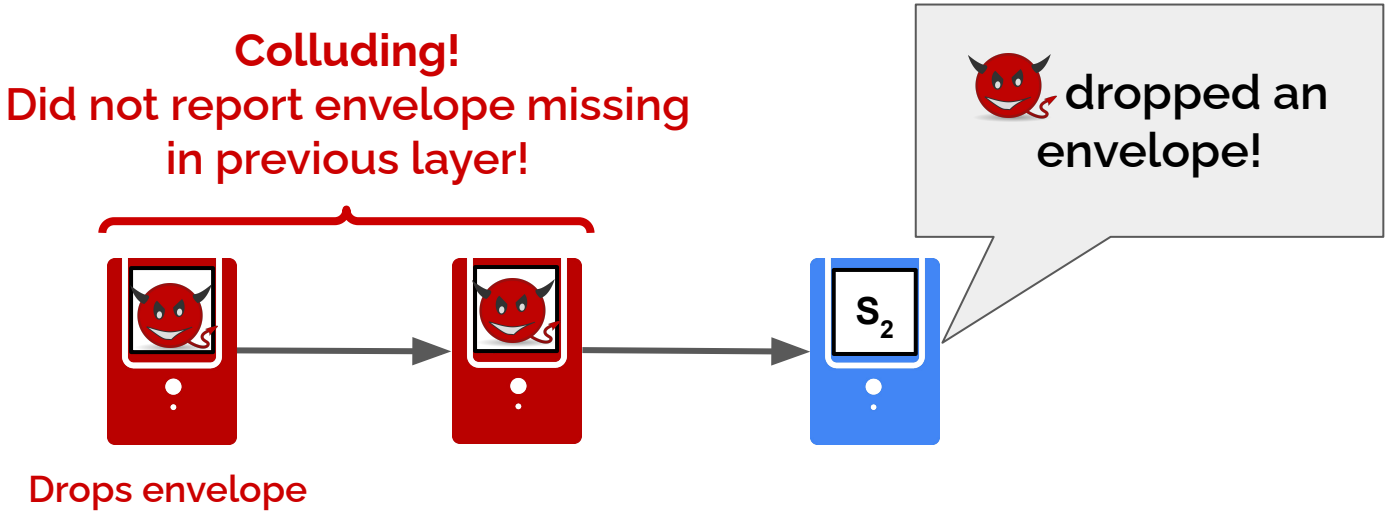
Robustness in Trellis using ARTs

- Deviations are always caught at the next **honest** server.
- An honest server would have blamed if had not received the envelope
- Guarantee: One adversary server is removed for each server deviation



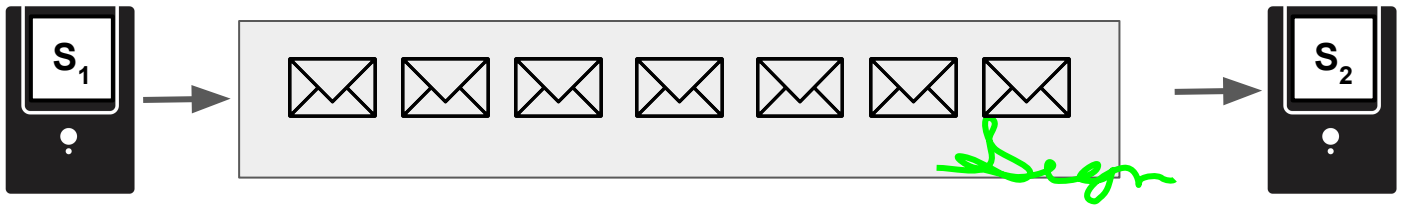
Robustness in Trellis using ARTs

- Deviations are always caught at the next **honest** server.
- An honest server would have blamed if had not received the envelope
- Guarantee: One adversary server is removed for each server deviation



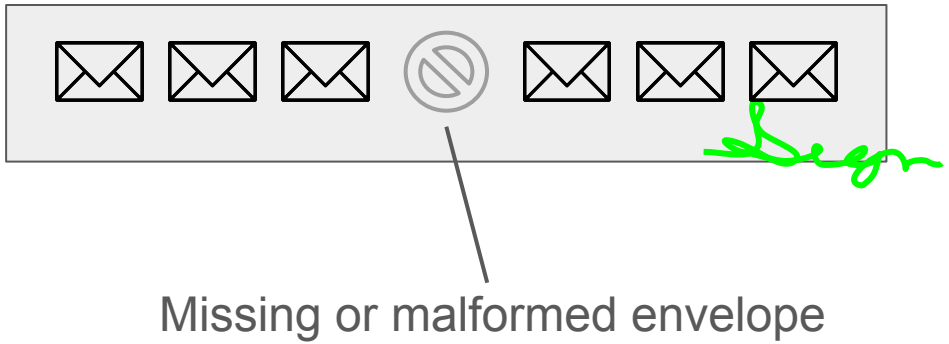
Publicly verifiable evidence

- Each server digitally signs the batch of envelopes it sends

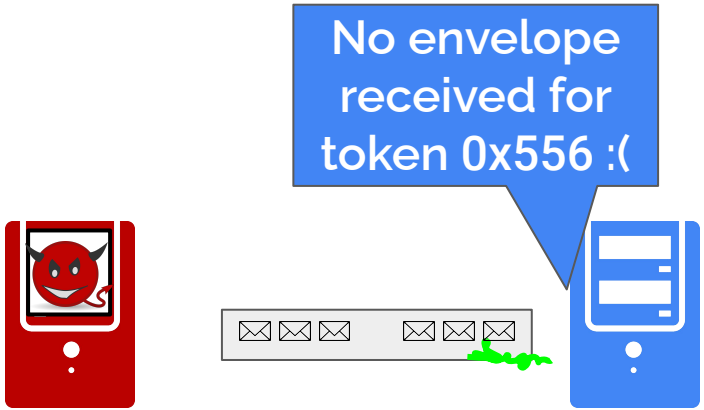


Publicly verifiable blame

- Servers use signatures to provide proof of “input”
- Allows a server to prove that an envelope wasn’t sent

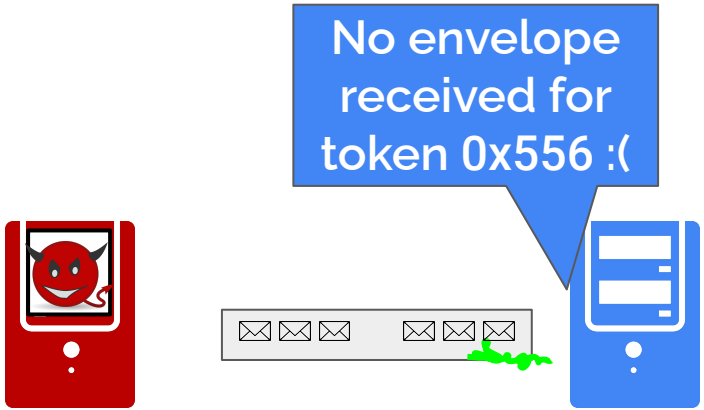


Blame with evidence

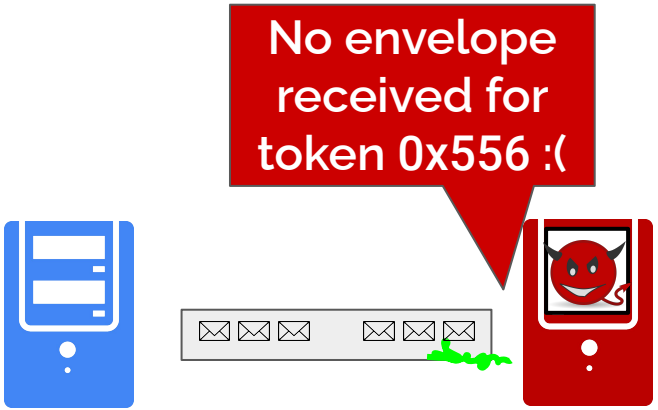


- Adversary dropped message
- Next server reveals signature with envelope missing

Blame with evidence

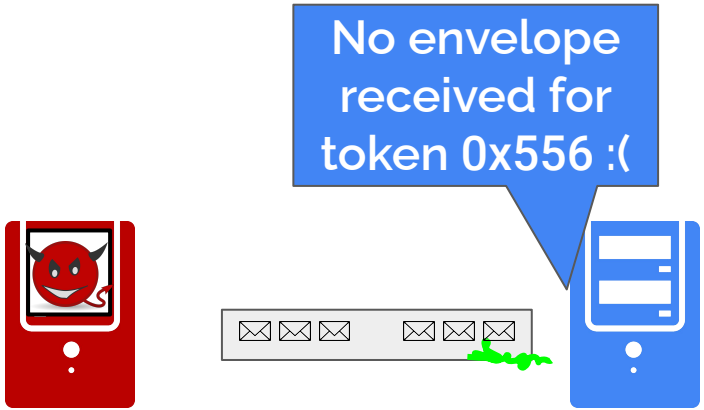


- Adversary dropped message
- Next server reveals signature with envelope missing

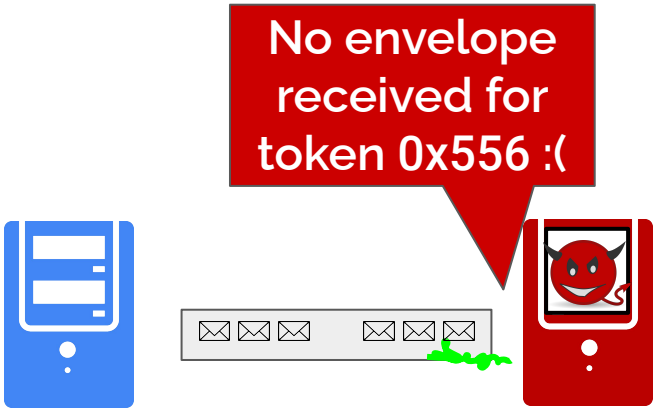


- Malicious invocation
- Can't show signed envelopes from honest server

Blame with evidence



- Adversary dropped message
- Next server reveals signature with envelope missing



- Malicious invocation
- Can't show signed envelopes from honest server

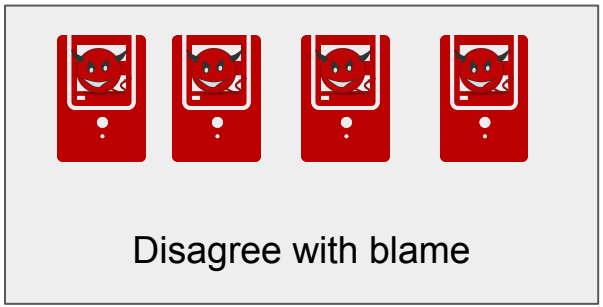
In both cases, the adversary already knows the revealed information!

Blame with evidence

- Evidence allows users and servers to decide which are honest

Observation: Honest servers will always evaluate evidence correctly

- Honest servers refuse servers who evaluate differently
- Users route through honest servers in future
- Every deviation removes at least one malicious server



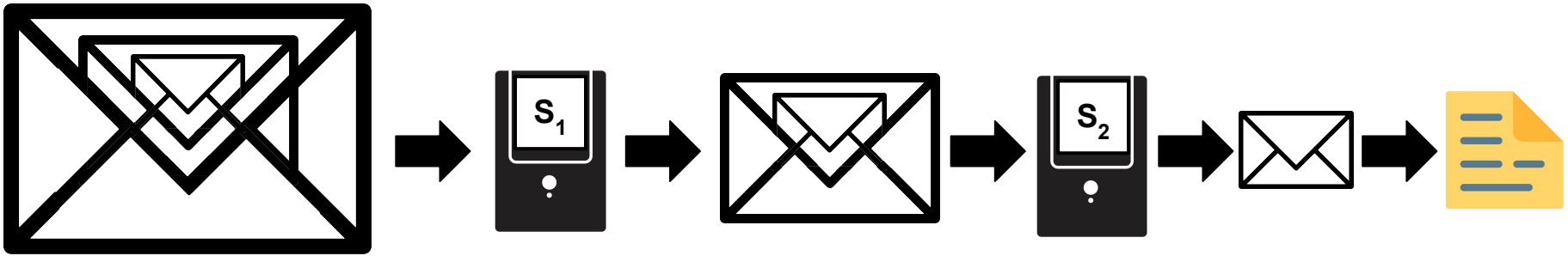
Constructing Trellis: Boomerang encryption

How do servers know which users are assigned to them?

Deliver anonymous routing tokens via **Boomerang encryption**

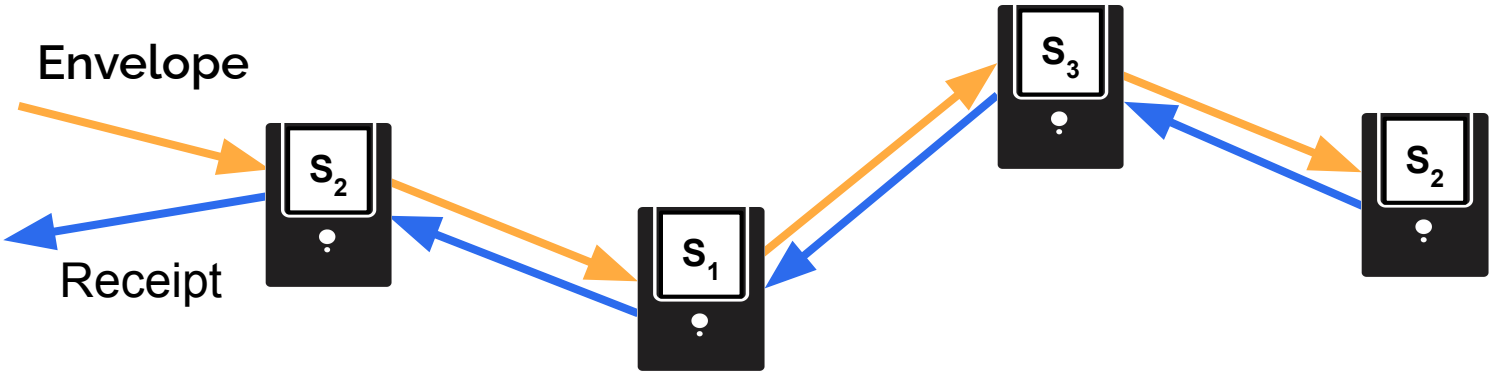
Onion envelopes

- A message is recursively encrypted to each of the servers
- Each server removes one layer of encryption
- All layers of decryption must be removed to get the message



New tool: Boomerang encryption

Provide **delivery receipts** by onion encrypting along the path again but in **reverse**



Only if all servers along the path decrypt the envelope, can the correct receipt be returned

Trellis overview

- **Incremental Path Establishment**:
  - Users use Boomerang Encryption to obtain receipts of correct Anonymous Routing Token delivery
  - Anonymous Routing Tokens guarantee delivery in later rounds

Trellis overview

- Incremental Path Establishment:
- **Broadcast rounds:**
  - Deliver messages along the established paths
  - See paper for full details

Trellis overview

- Incremental Path Establishment:
- Broadcast rounds:
- **Blame protocols on demand:**
  - Remove malicious parties with evidence
  - Discard malicious messages with evidence
  - Recover from deviations
  - See paper for full details

Evaluation

AWS machines

- Up to 256 m5.xlarge (4 core, 16GB) instances

Across four AWS regions

- Oregon, Virginia, Frankfurt, and Stockholm
- Roughly matches the Tor network geographic distribution

Simulate limited bandwidth and latency conditions

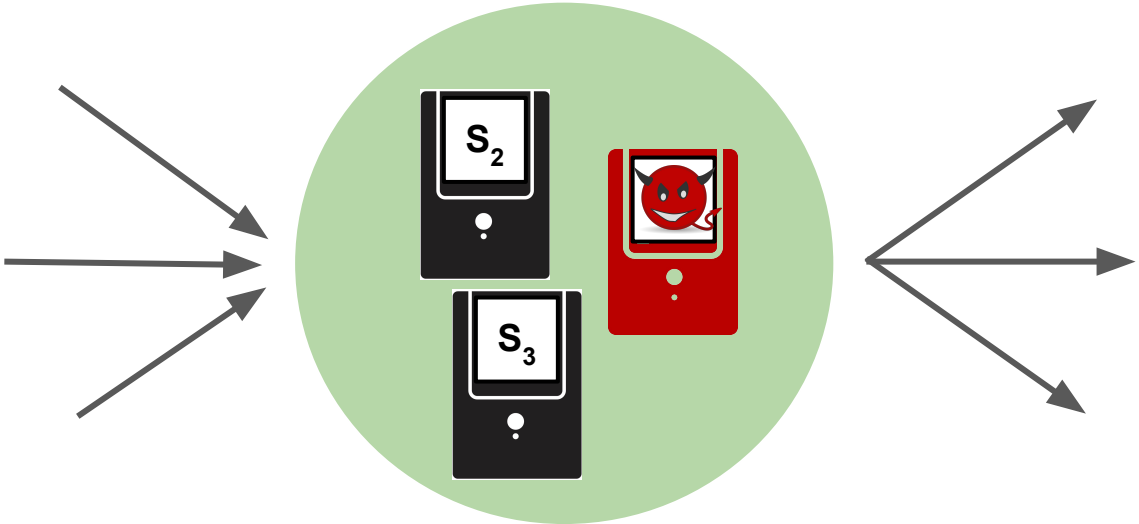
- Tail latency impacts time before we can send next layer!

Atom [KCGDF'17]

Main idea: Use "anytrust" groups for robustness

Servers check each other's work:

- If the group is large enough, it contains at least one honest server with high probability



Atom [KCGDF'17]

Main idea: Use "anytrust" groups for robustness

Servers check each other's work:

- If the group is large enough, it contains at least one honest server with high probability

Problem: large *computational* overheads

- Zero-knowledge proofs to verify server actions ⇒ High computational overheads
- Work increases with group size

Atom vs. Trellis

Atom [KCGDF'17]

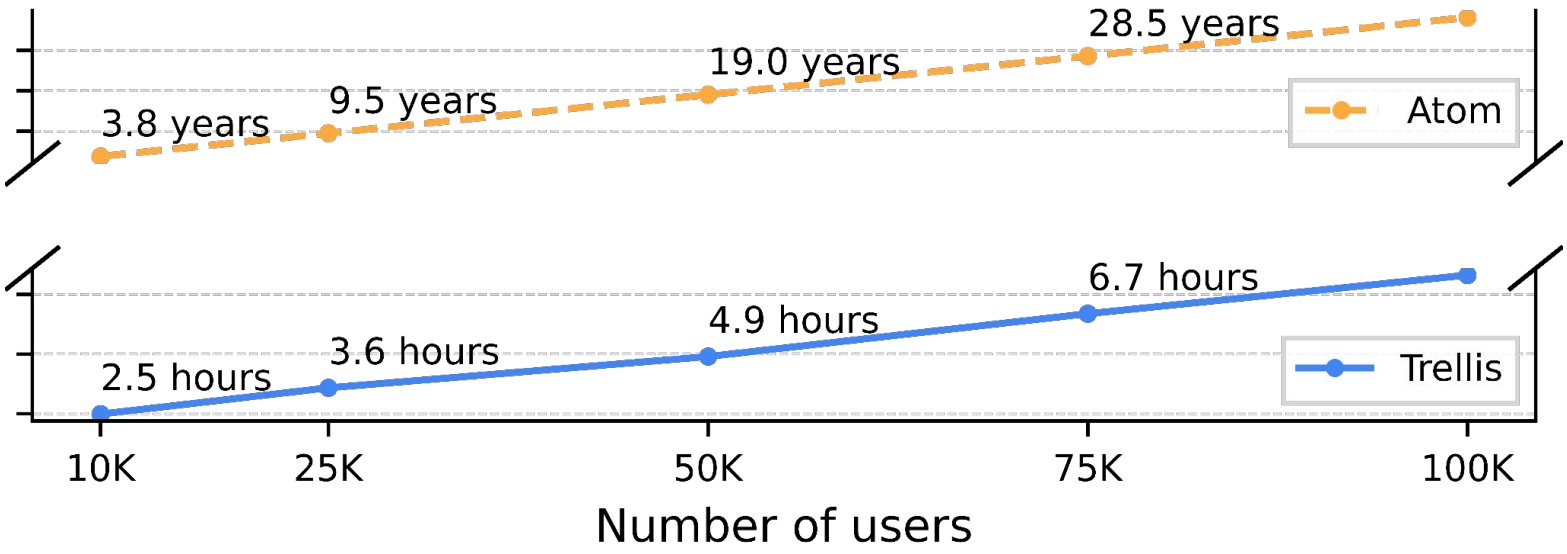
- Uses zero knowledge proofs to show server correctness
- Every server action is checked by a group of servers with large network overhead
- Large proofs for large messages cause computational slow down

Trellis

- Servers only "check" signatures
- A deviation results in the malicious party being removed
- Protocols remain efficient even for large messages
- (Theoretically) supports dishonest majority ($f > 0.5$)

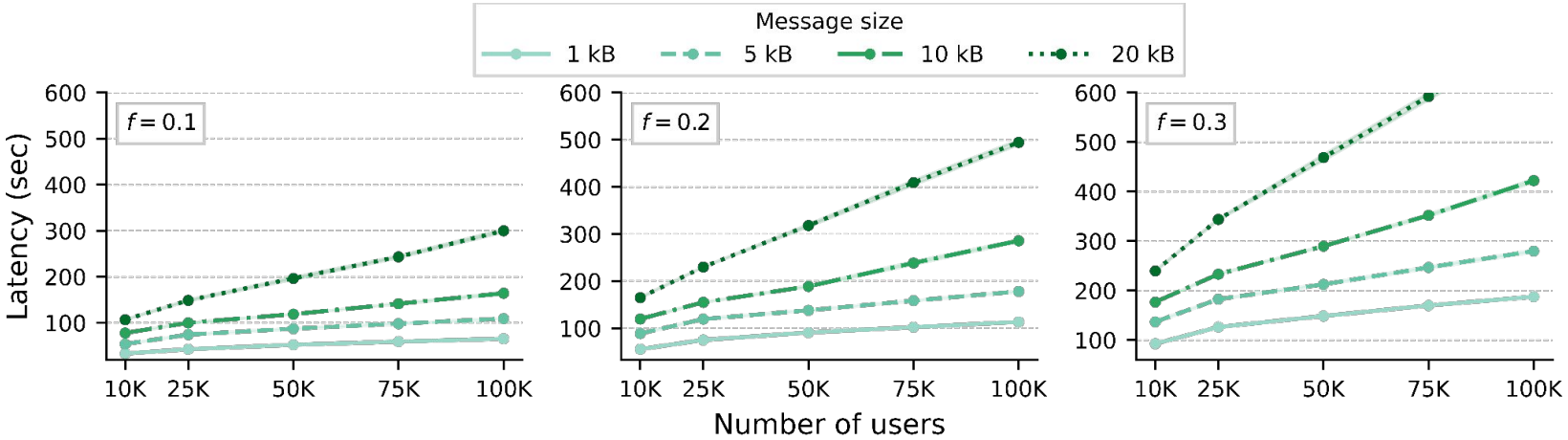
Atom vs. Trellis

- Orders of magnitude faster than Atom (f=0.2, 128 servers)
- Especially efficient with long messages (1MB here).



Trellis: Message Scaling

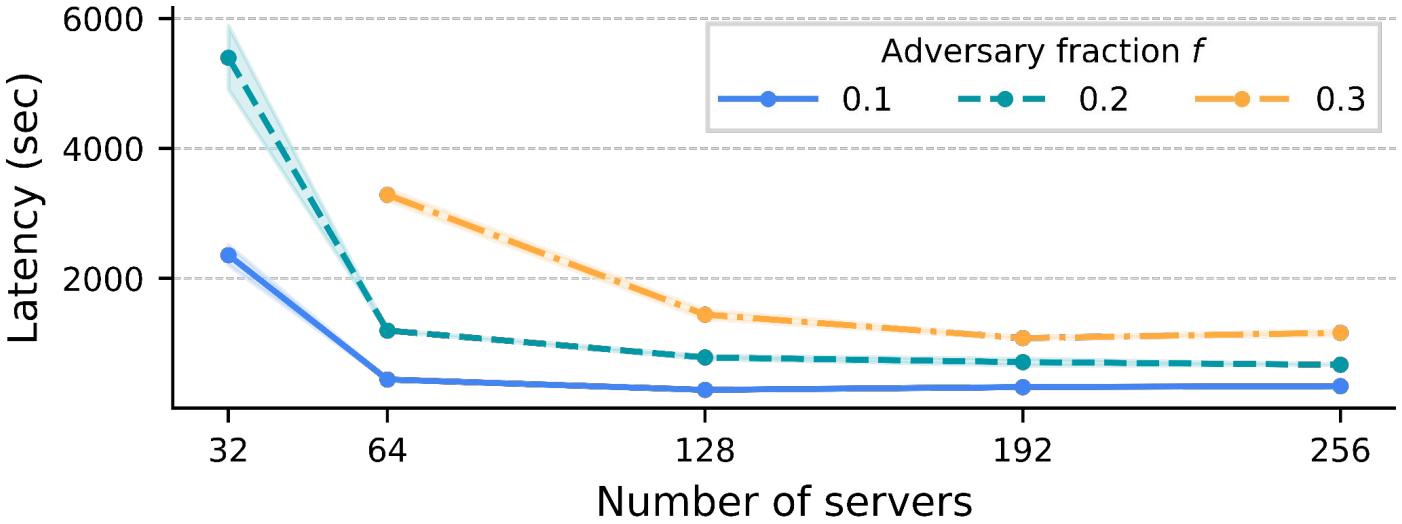
- Scales linearly with size of message
- Scales nearly-linearly with number of messages



Trellis: 128 servers, 200Mbps network

Trellis: Server scaling

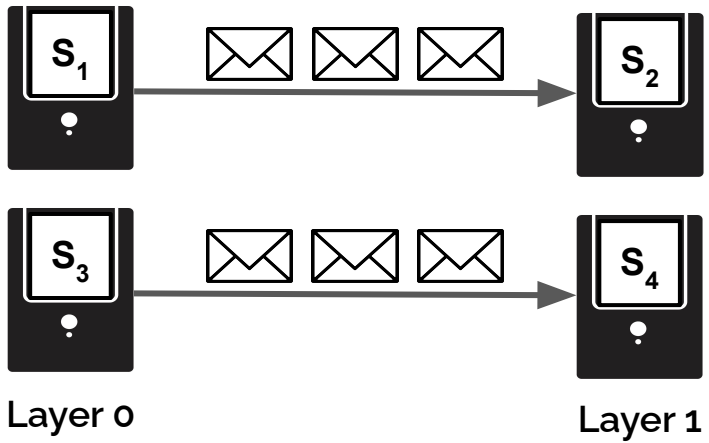
- Network bound: tail latency is constant
- Network bound: At least one dummy envelope is required for every pair of servers



Trellis: 200Mbps network, with 2 million, 1Kb messages

Overheads of metadata privacy

- Bandwidth: dummy envelopes are needed as cover traffic to and between servers.
- Latency: servers must wait for all other servers to send envelopes to hide timing information.
- Required for metadata private systems to hide metadata



Questions?

See paper for things not covered!

For example:

- Trust graphs for managing dishonest majority and unresponsive servers
- Number of layers required to achieve mixing
- Full details on all the blame protocols

Paper ePrint: https://eprint.iacr.org/2022/1548

Code: https://github.com/SimonLangowski/trellis

Contact: Simon (slangows@mit.edu) | Sacha (3s@mit.edu)

References

[KCGDF'17]: Kwon, A., Corrigan-Gibbs, H., Devadas, S., & Ford, B. (2017, October). Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles* (pp. 406-422).

[Chaum'81] Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, *24*(2), 84-90.

[GJ'04] Golle, P., & Juels, A. (2004, October). Parallel mixing. In *Proceedings of the 11th ACM conference on Computer and communications security* (pp. 220-226).