Nearly Optimal Parallel Broadcast in the Plain Public Key Model

Ran Gelles (Bar-Ilan University), Christoph Lenzen (CISPA), Julian Loss (CISPA), Sravya Yandamuri (Duke University and Common Prefix)

Funded by
the European Union

Byzantine Broadcast

Byzantine Broadcast

Sender

Byzantine Broadcast

Sender

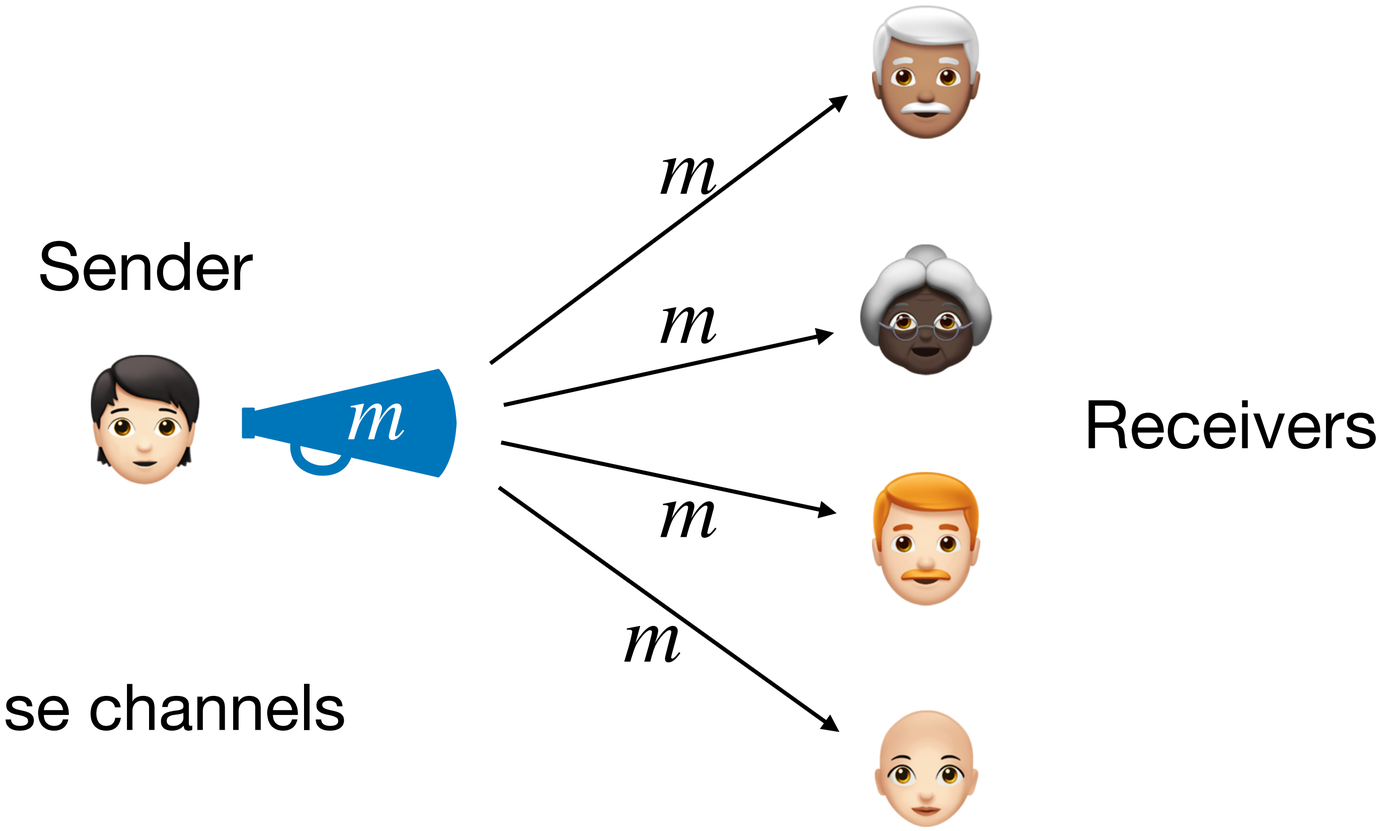
Receivers

Byzantine Broadcast

Sender

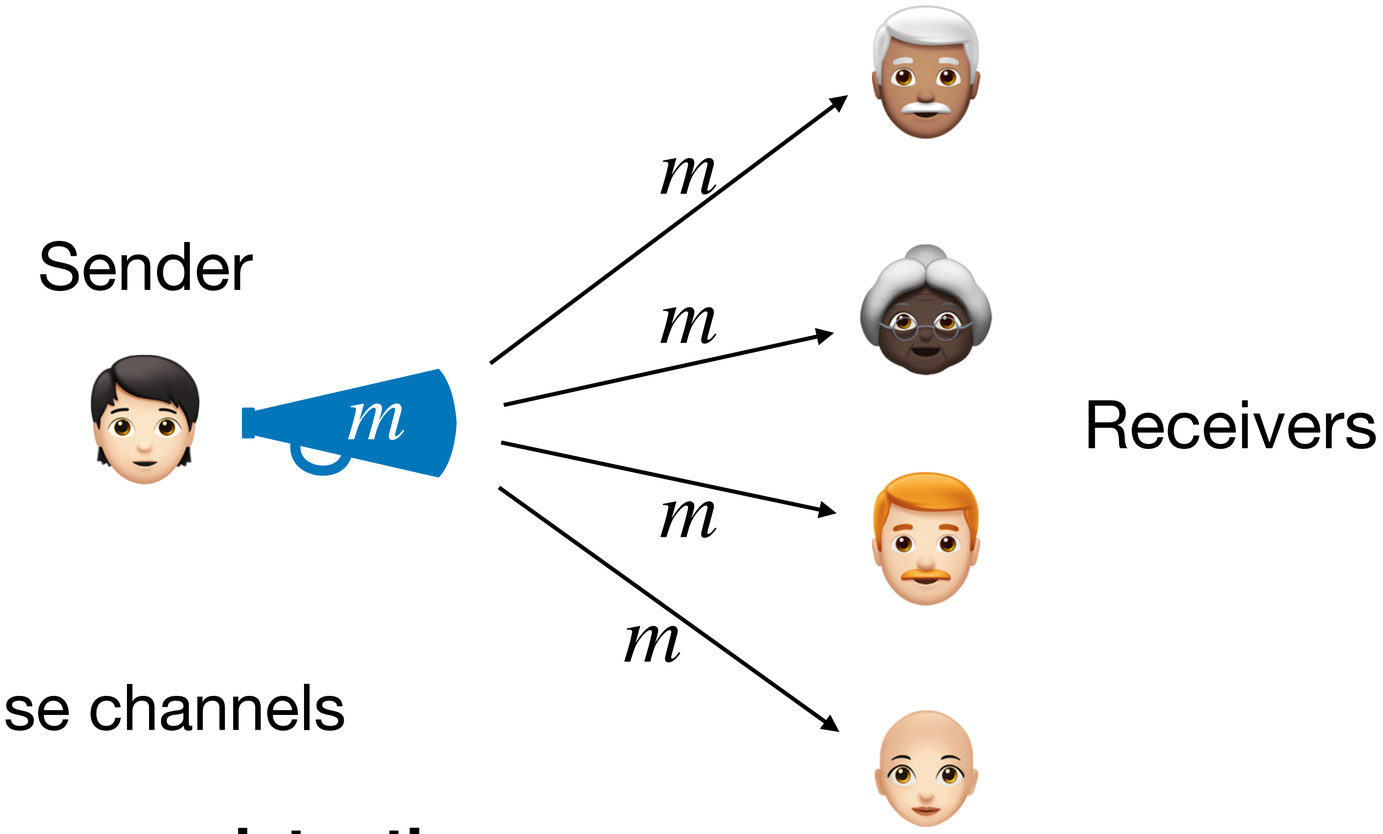
Receivers

Byzantine Broadcast



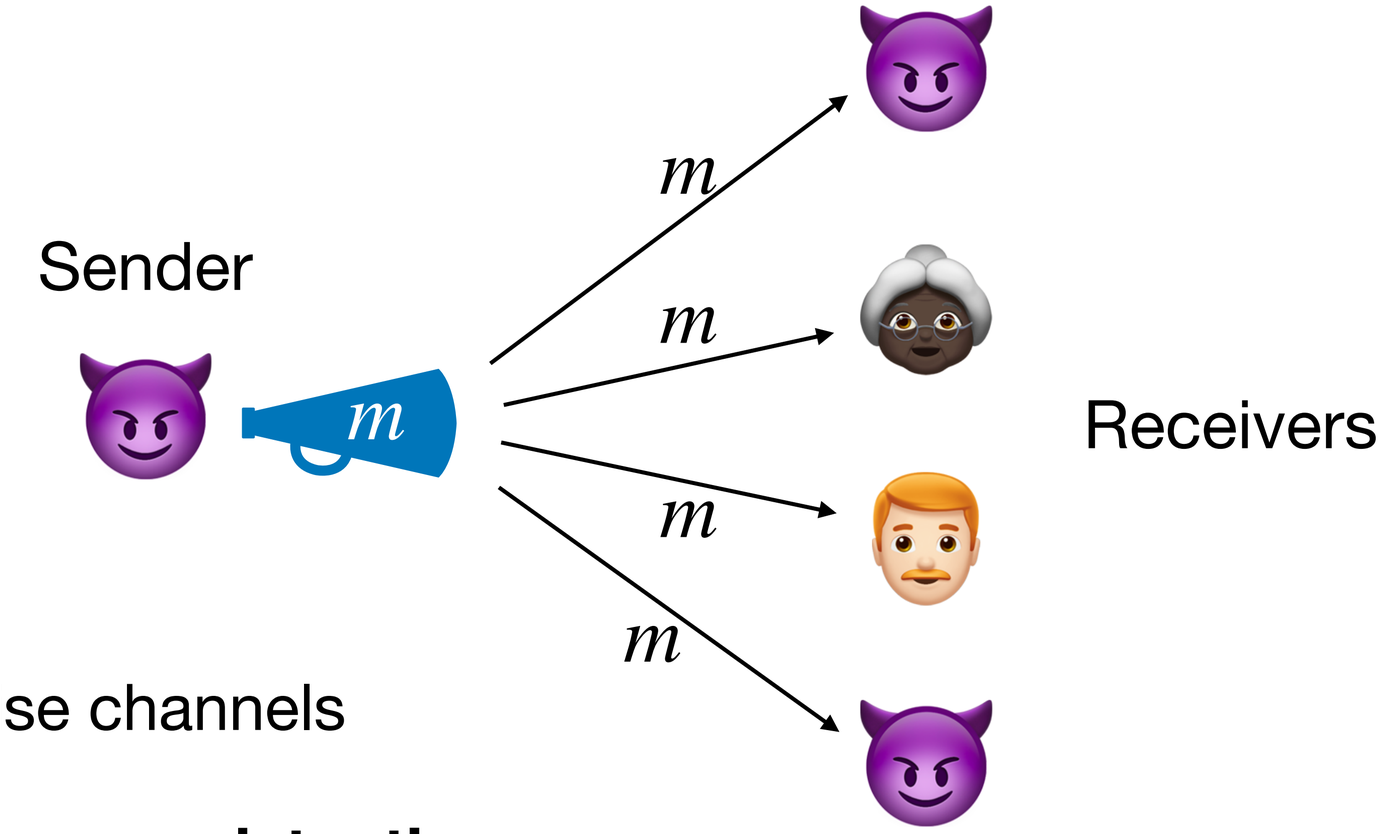
- **Setting:** n parties, pairwise channels

Byzantine Broadcast



- **Setting:** n parties, pairwise channels
- **Goal:** Sender distributes m **consistently**

Byzantine Broadcast



- **Setting:** n parties, pairwise channels
- **Goal:** Sender distributes m **consistently**
- **Problem: Majority** of $t < n$ parties is malicious

This Work: Efficient Parallel Brodcast

This Work: Efficient Parallel Brodcast

- Like Broadcast, but everybody sends $\Longrightarrow$ n consistent outputs

This Work: Efficient Parallel Brodcast

- Like Broadcast, but everybody sends $\Longrightarrow$ n consistent outputs
- Many applications e.g. MPC and Secret Sharing

What Makes a Protocol Efficient?

What Makes a Protocol Efficient?

- Want to run protocol in large-scale networks $\Longrightarrow$ n could be very large

What Makes a Protocol Efficient?

- Want to run protocol in large-scale networks $\Longrightarrow$ n could be very large
- **Communication Complexity:** how many bits does protocol exchange?

What Makes a Protocol Efficient?

- Want to run protocol in large-scale networks $\Longrightarrow$ n could be very large
- **Communication Complexity:** how many bits does protocol exchange?
- Should scale well as function of n, e.g. $O(n^2)$ is better than $O(n^4)$

What Makes a Protocol Efficient?

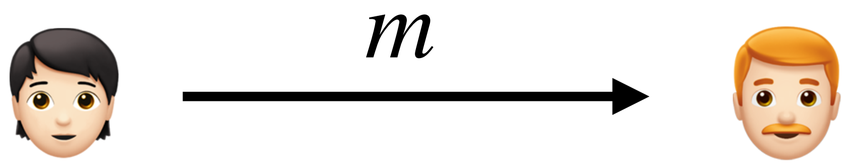
- Want to run protocol in large-scale networks $\Longrightarrow$ n could be very large
- **Communication Complexity:** how many bits does protocol exchange?
- Should scale well as function of n, e.g. $O(n^2)$ is better than $O(n^4)$
- **Our work:** improve communication complexity

Synchronous Network Model

Synchronous Network Model

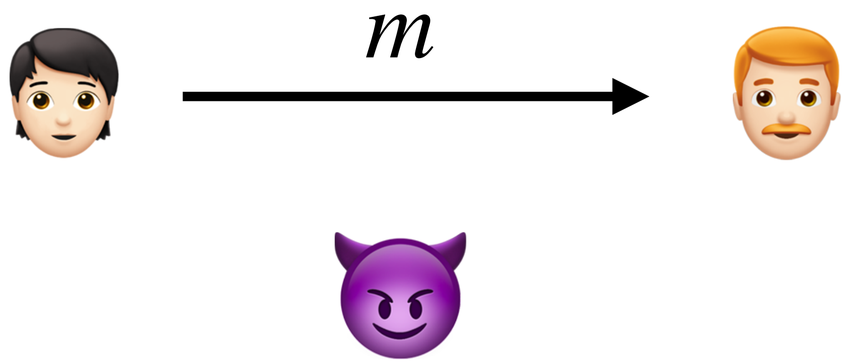
- Shared clock

Synchronous Network Model



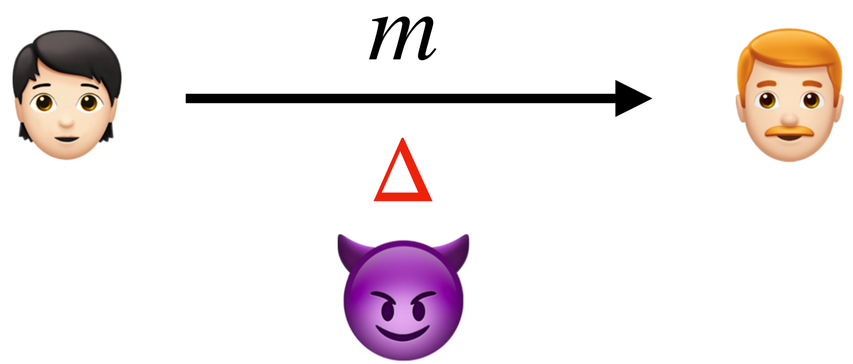
- Shared clock

Synchronous Network Model



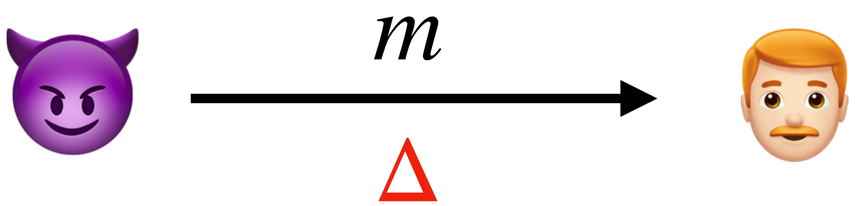
- Shared clock

Synchronous Network Model



- Shared clock
- Honest messages delivered in Δ time, **no drops**

Synchronous Network Model



- Shared clock
- Honest messages delivered in Δ time, **no drops**
- Up to $t < n/2$ **malicious** corruptions

Synchronous Network Model (cont.)

Synchronous Network Model (cont.)

- **Adaptive:** Adversary corrupts parties **at any point of execution**

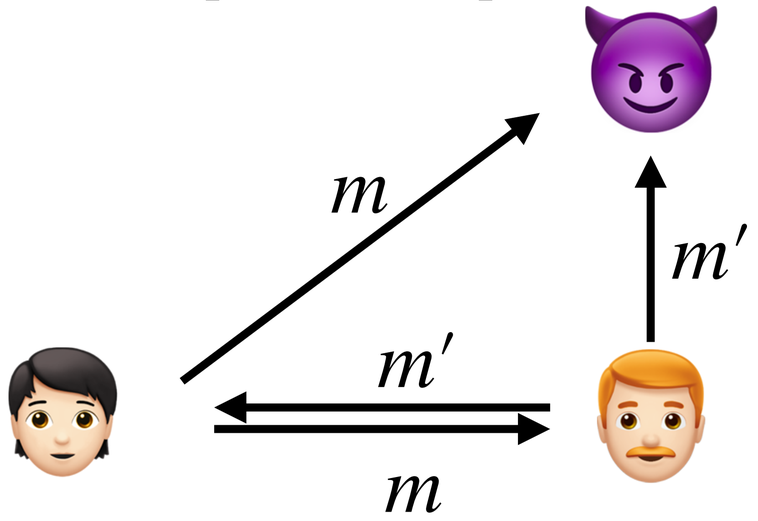
Synchronous Network Model (cont.)

- **Adaptive:** Adversary corrupts parties **at any point of execution**

Synchronous Network Model (cont.)

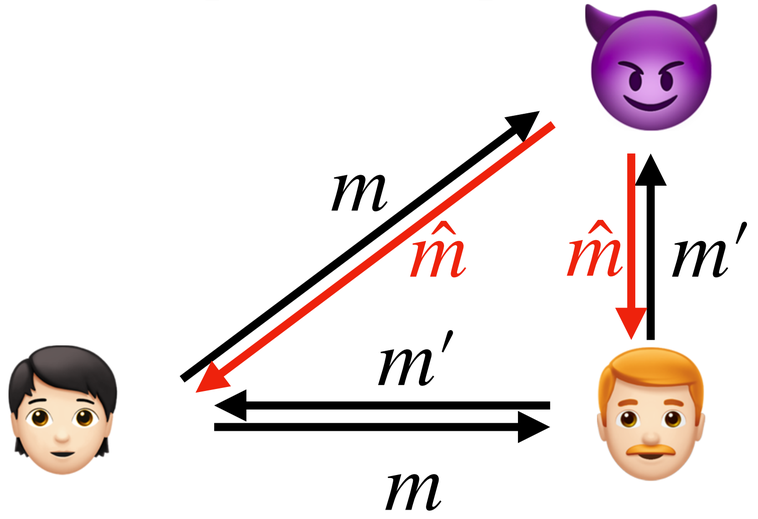
- **Adaptive:** Adversary corrupts parties **at any point of execution**
- **Rushing:** Learns honest messages **immediately** in each round

Synchronous Network Model (cont.)



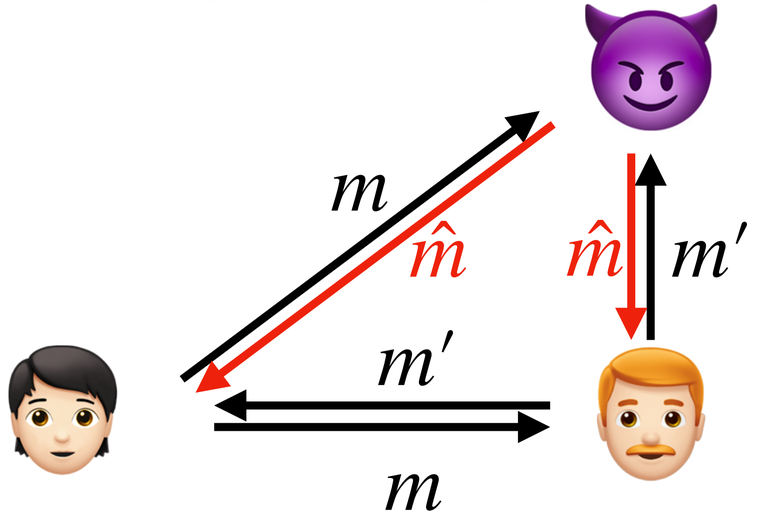
- **Adaptive:** Adversary corrupts parties **at any point of execution**
- **Rushing:** Learns honest messages **immediately** in each round

Synchronous Network Model (cont.)



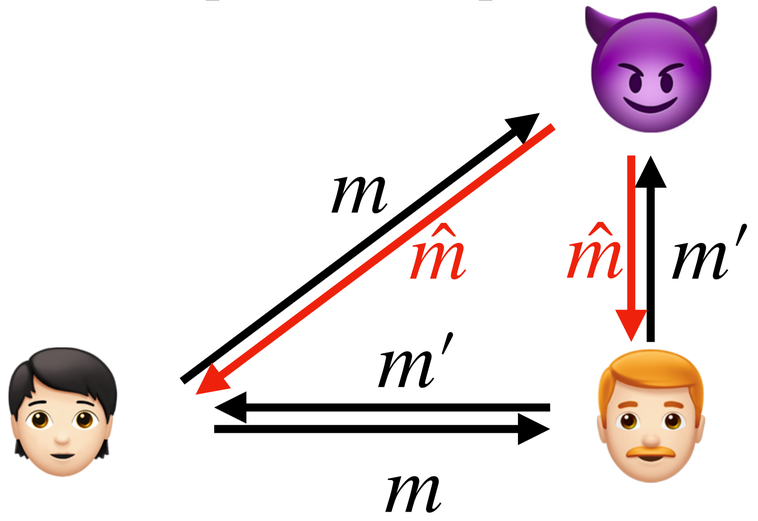
- **Adaptive:** Adversary corrupts parties **at any point of execution**
- **Rushing:** Learns honest messages **immediately** in each round

Synchronous Network Model (cont.)



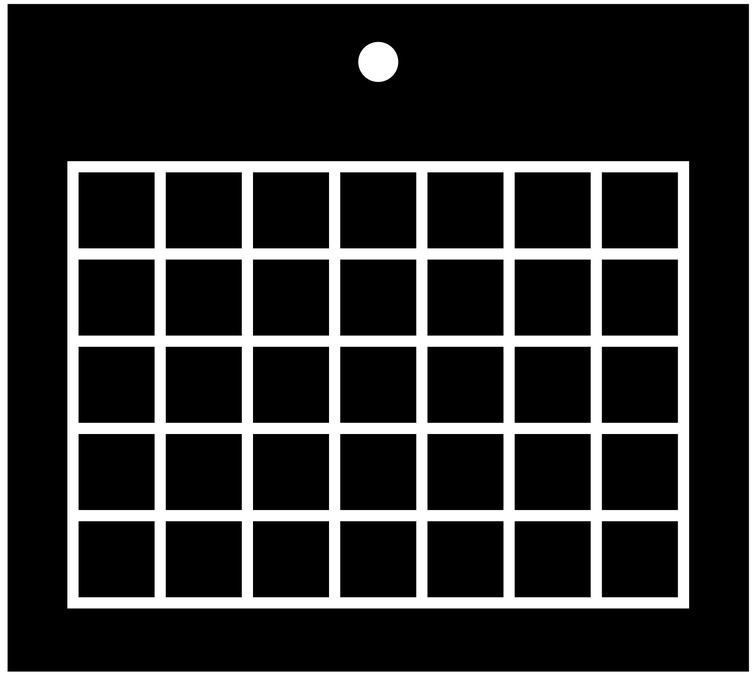
- **Adaptive:** Adversary corrupts parties **at any point of execution**
- **Rushing:** Learns honest messages **immediately** in each round
  - Can not drop messages that party sent while it was still honest

Synchronous Network Model (cont.)

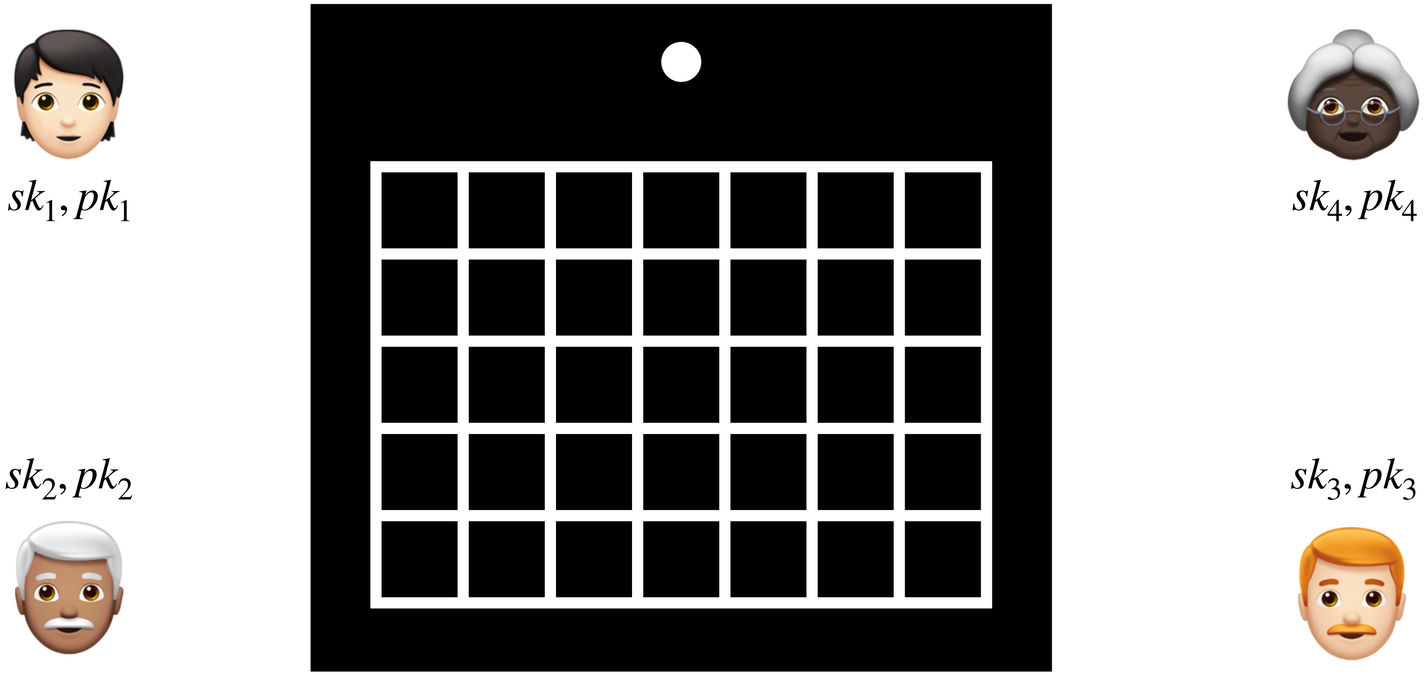


- **Adaptive:** Adversary corrupts parties **at any point of execution**
- **Rushing:** Learns honest messages **immediately** in each round
  - Can not drop messages that party sent while it was still honest
  - Can not corrupt during **atomic send operation**

Dolev-Yao Model

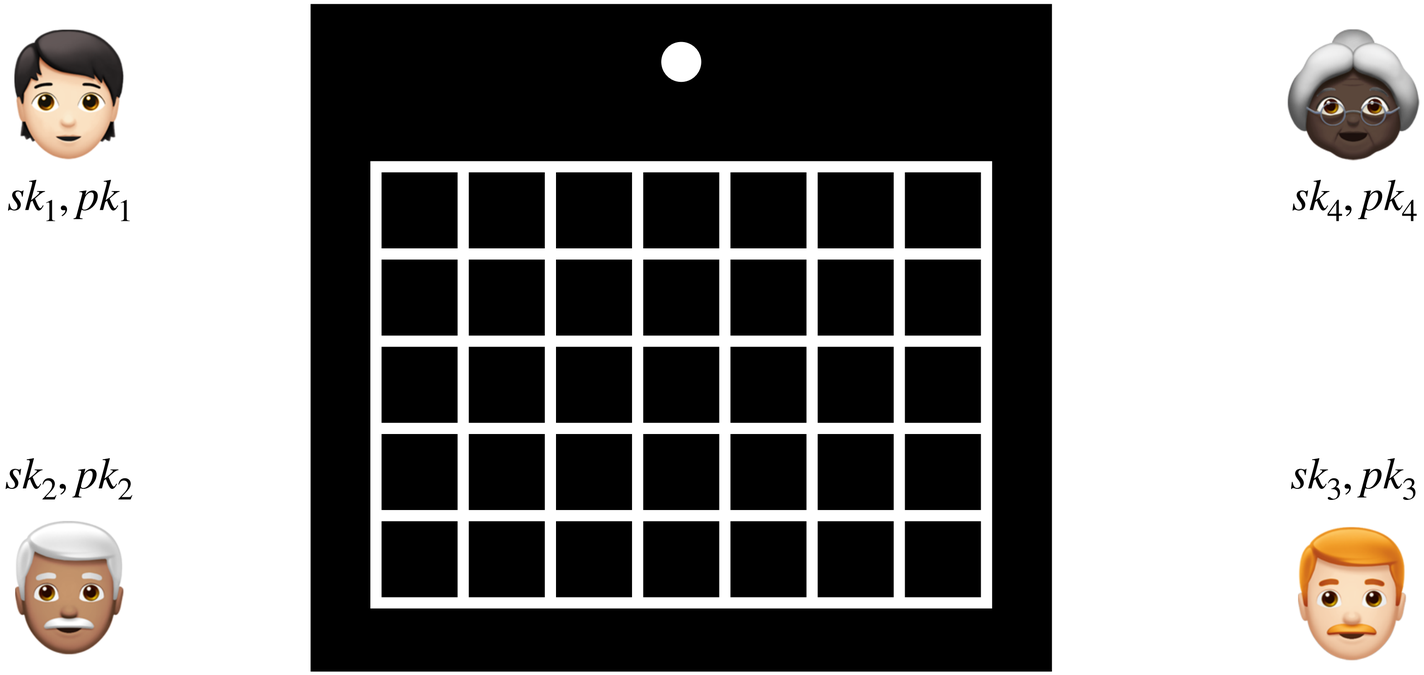


Dolev-Yao Model



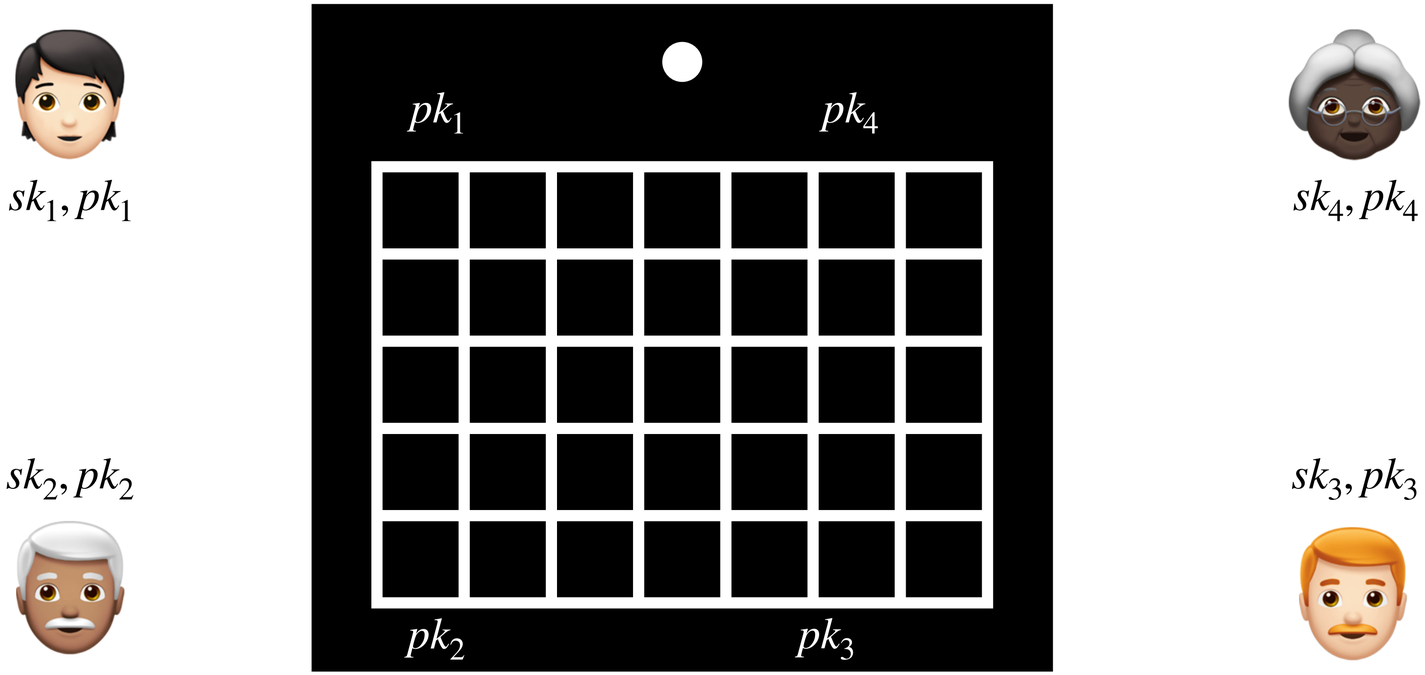
- Only allows for `basic crypto' (hash functions, signatures, public/secret key encryption)

Dolev-Yao Model



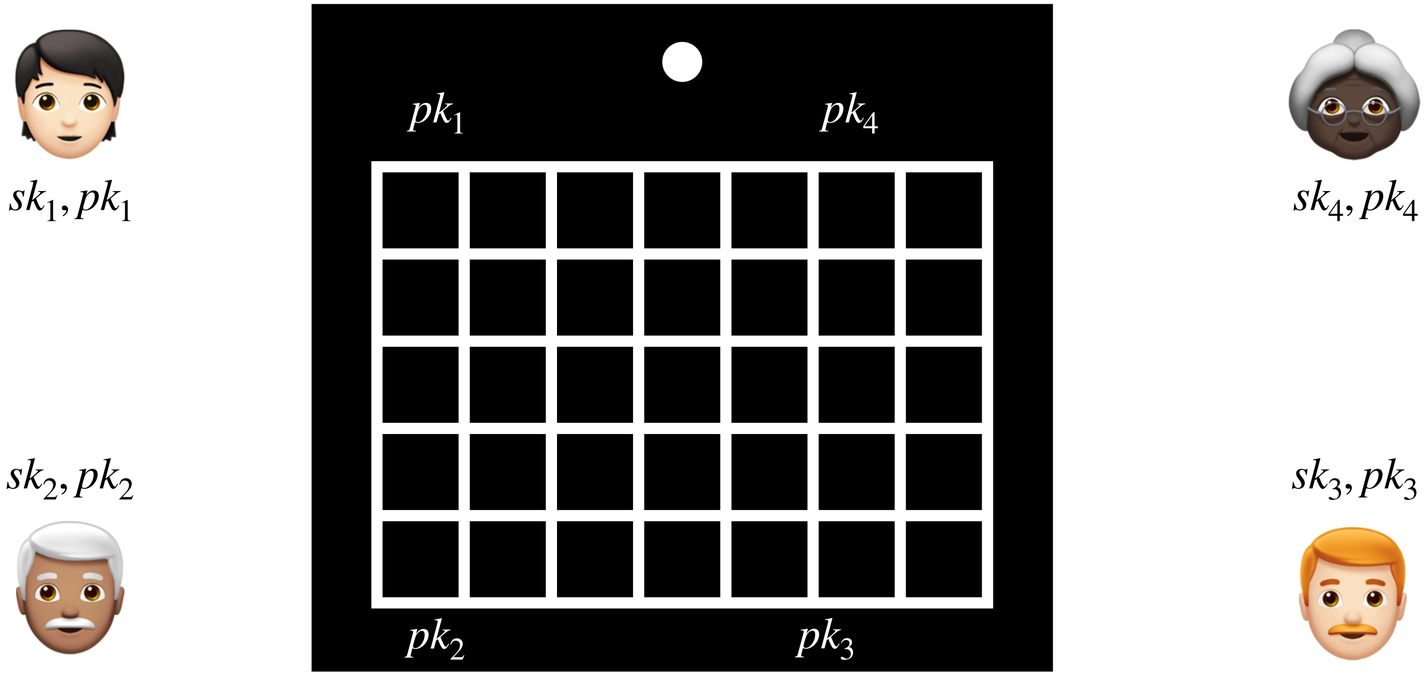
- Only allows for `basic crypto' (hash functions, signatures, public/secret key encryption)
- Parties generate **their own** keys, post them to public bulletin board **before start of execution**

Dolev-Yao Model



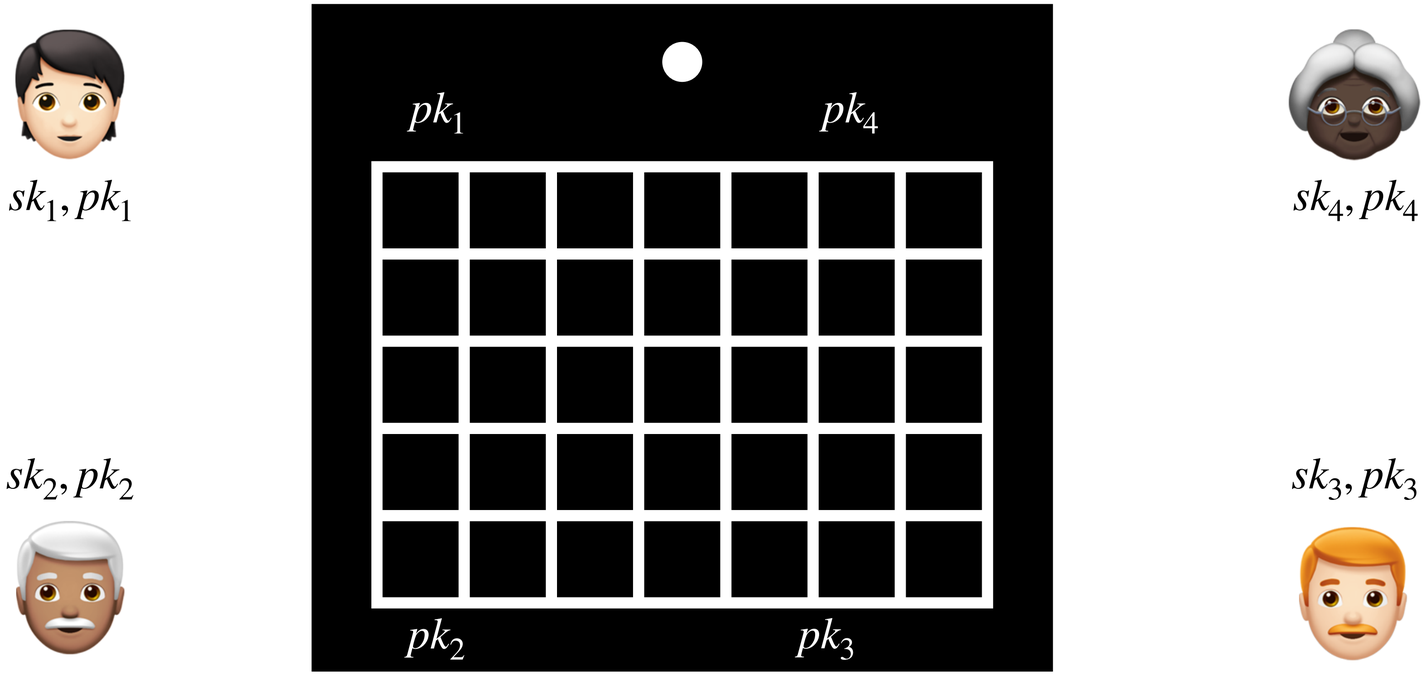
- Only allows for `basic crypto' (hash functions, signatures, public/secret key encryption)
- Parties generate **their own** keys, post them to public bulletin board **before start of execution**

Dolev-Yao Model



- Only allows for `basic crypto' (hash functions, signatures, public/secret key encryption)
- Parties generate **their own** keys, post them to public bulletin board **before start of execution**
- Pro: Crypto compatible with DY model is **simple** and **very efficient**

Dolev-Yao Model



- Only allows for `basic crypto' (hash functions, signatures, public/secret key encryption)
- Parties generate **their own** keys, post them to public bulletin board **before start of execution**
- Pro: Crypto compatible with DY model is **simple** and **very efficient**
- Con: Harder to construct **asymptotically efficient** protocols

Some Notation

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)
- λ : statistical security parameter; want security with probability $1 - 2^{-\lambda}$

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)
- λ : statistical security parameter; want security with probability $1 - 2^{-\lambda}$
- n : number of parties, $t < n$ parties are malicious

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)
- λ : statistical security parameter; want security with probability $1 - 2^{-\lambda}$
- n : number of parties, $t < n$ parties are malicious
- Typical values: $\lambda \approx 100$, $\kappa \approx 256$

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)
- λ : statistical security parameter; want security with probability $1 - 2^{-\lambda}$
- n : number of parties, $t < n$ parties are malicious
- Typical values: $\lambda \approx 100$, $\kappa \approx 256$
- $n \geq \kappa \geq \lambda$

Some Notation

- κ : normalized cryptographic parameter size (hashes, signatures etc.)
- λ : statistical security parameter; want security with probability $1 - 2^{-\lambda}$
- n : number of parties, $t < n$ parties are malicious
- Typical values: $\lambda \approx 100$, $\kappa \approx 256$
- $n \geq \kappa \geq \lambda$
- λ and κ factors matter in practice!

Our Result: New Protocol in DY Model

Our Result: New Protocol in DY Model

Our Result: New Protocol in DY Model

- **Resilience:** $t < n/2$ in DY model

Our Result: New Protocol in DY Model

- **Resilience:** $t < n/2$ in DY model
- **Communication:** $O(n^2 \cdot \ell \cdot \log(n) + n \cdot \kappa \cdot \log^4(n))$

Fundamental Operation: Distributing Certificate

Fundamental Operation: Distributing Certificate

- **Certificate** C is a **list of** $t + 1$ **signatures** on some message

Fundamental Operation: Distributing Certificate

C

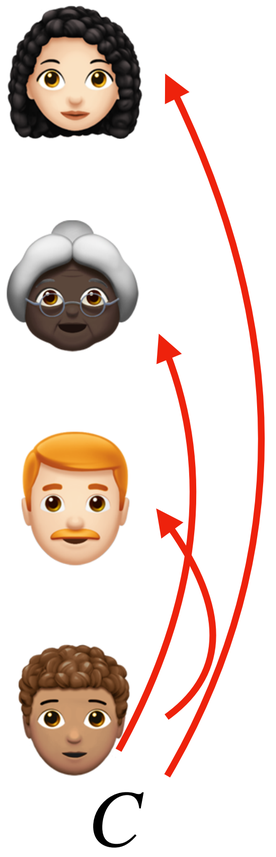
- **Certificate** C is a **list of** $t + 1$ **signatures** on some message

Fundamental Operation: Distributing Certificate

C

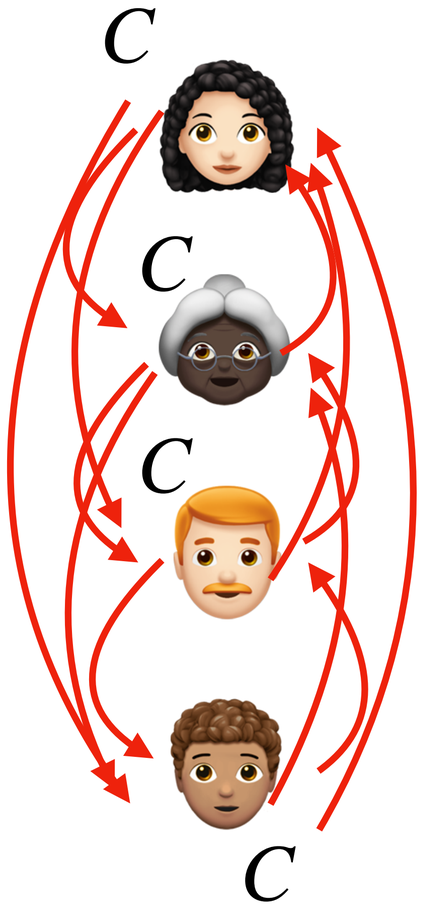
- **Certificate** C is a **list of** $t + 1$ **signatures** on some message
- DY model $\Longrightarrow$ C has size $O(t \cdot \kappa) = O(n \cdot \kappa)$

Fundamental Operation: Distributing Certificate



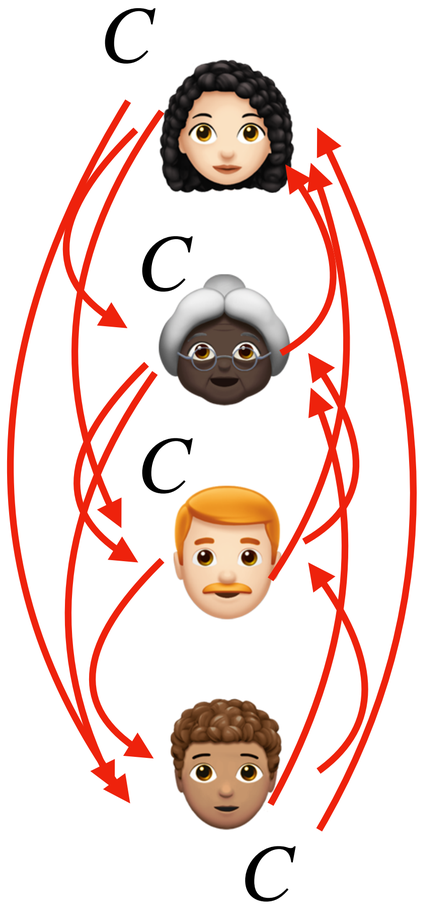
- **Certificate** C is a **list of** $t + 1$ **signatures** on some message
- DY model $\Longrightarrow$ C has size $O(t \cdot \kappa) = O(n \cdot \kappa)$

Fundamental Operation: Distributing Certificate



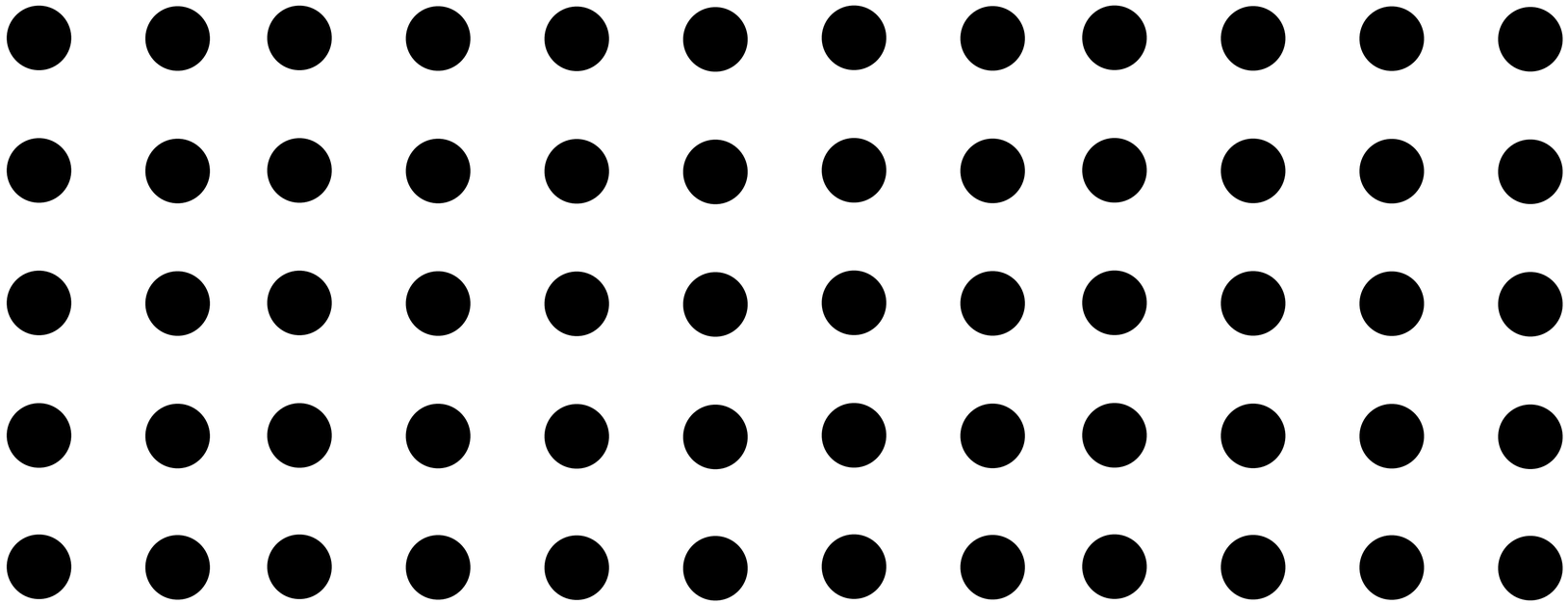
- **Certificate** C is a **list of** $t + 1$ **signatures** on some message
- DY model $\Longrightarrow$ C has size $O(t \cdot \kappa) = O(n \cdot \kappa)$

Fundamental Operation: Distributing Certificate

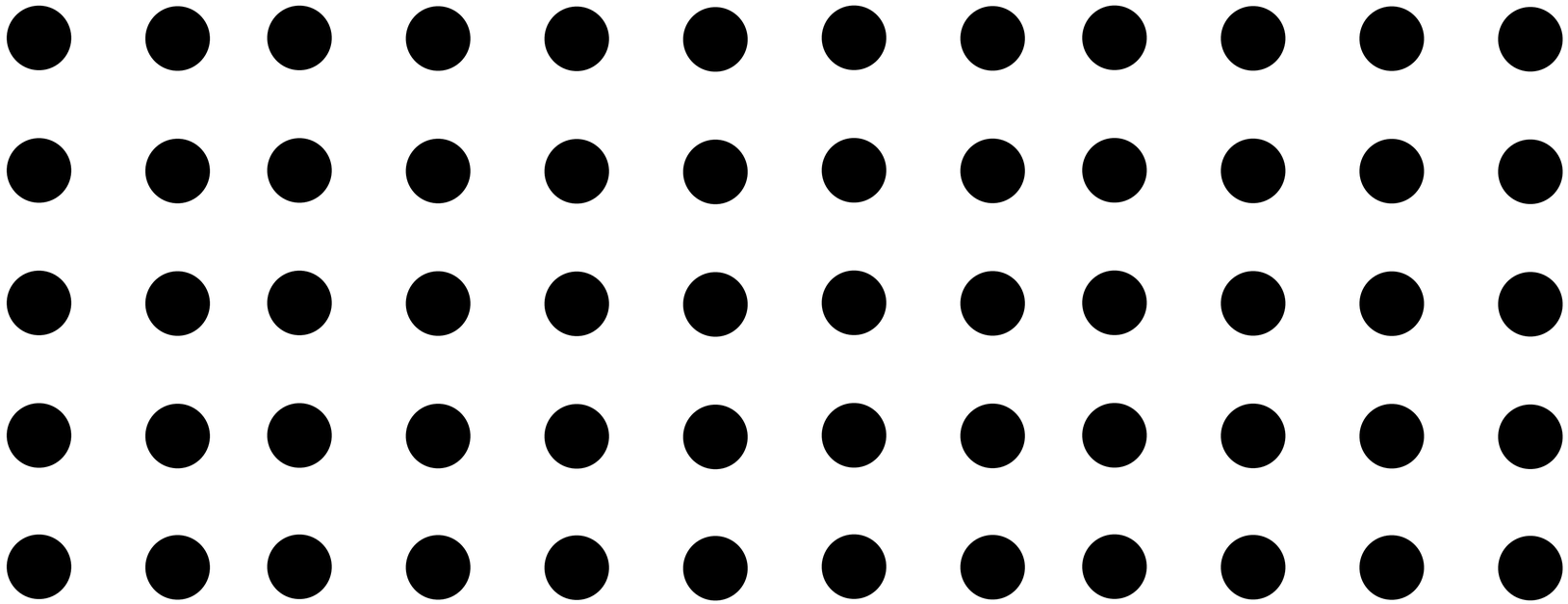


- **Certificate** C is a **list of** $t+1$ **signatures** on some message
- DY model $\Longrightarrow$ C has size $O(t \cdot \kappa) = O(n \cdot \kappa)$
- Naive echoing: $O(n^3 \cdot \kappa)$ bits

Idea: Replace Send-to-All with Gossip

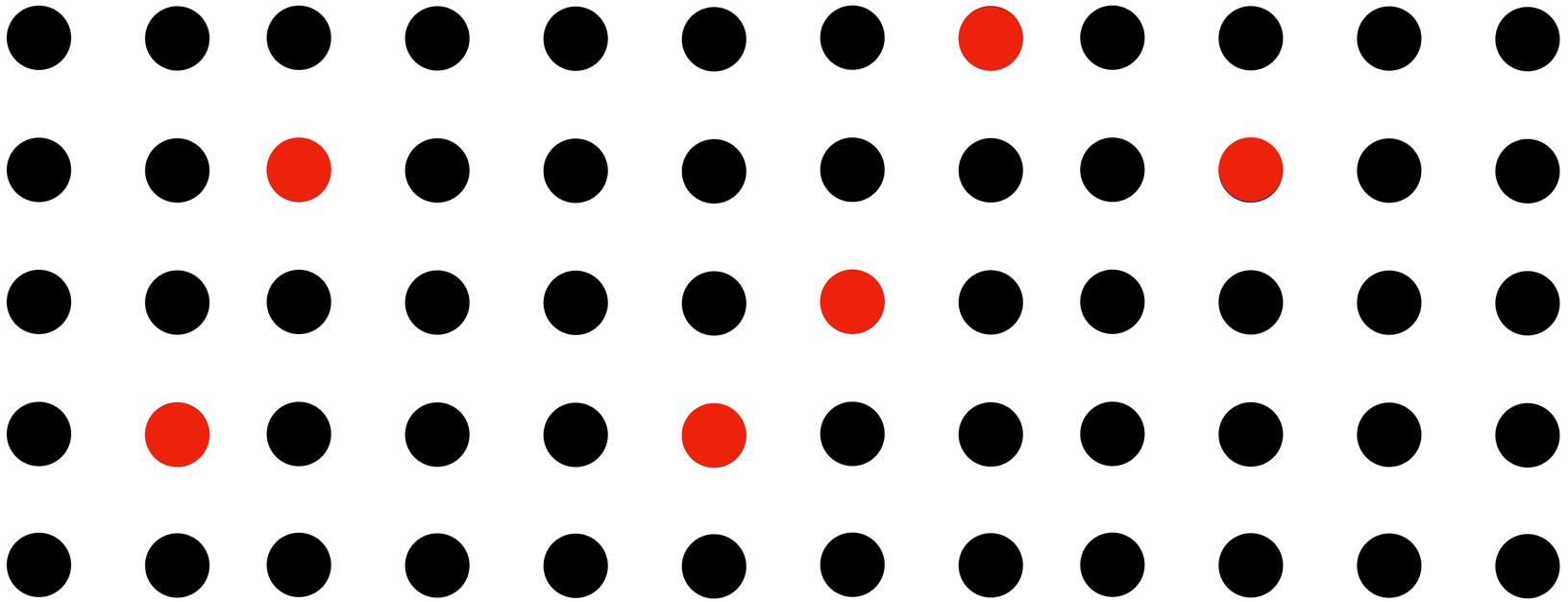


Idea: Replace Send-to-All with Gossip



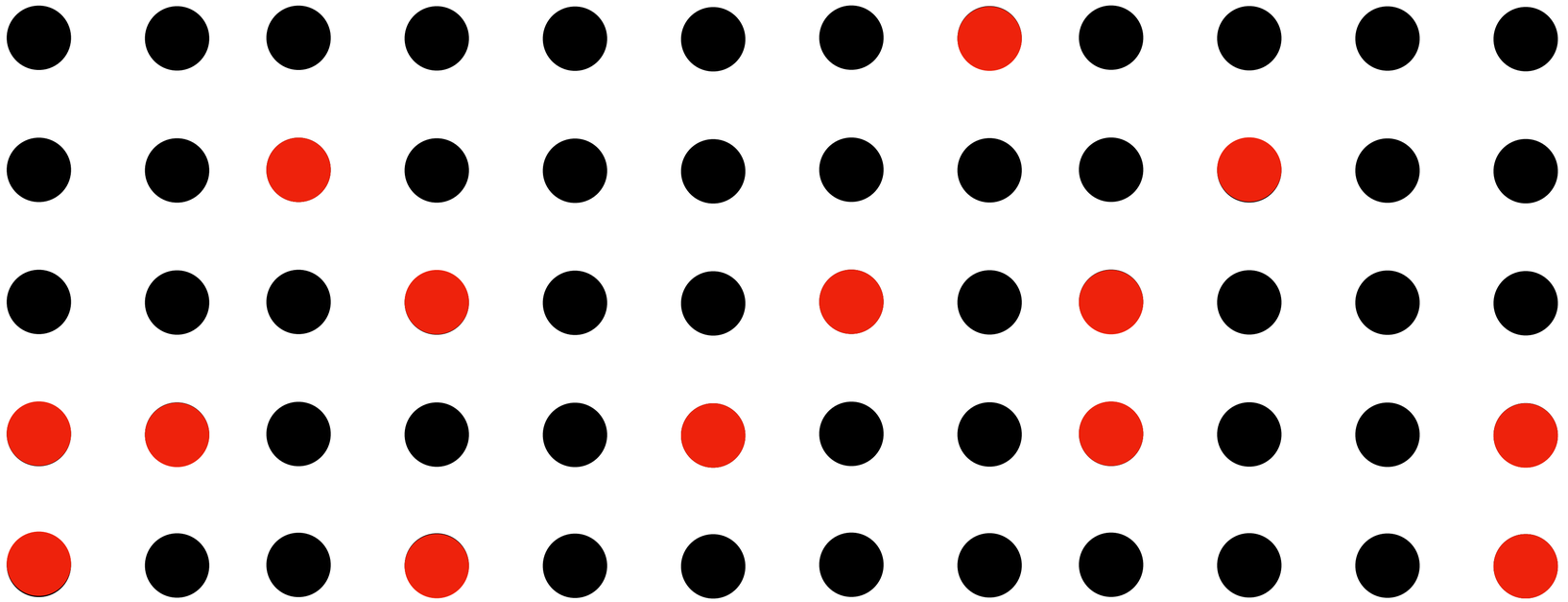
- **Gossip:** Tell message to few neighbours (only once), they do the same

Idea: Replace Send-to-All with Gossip



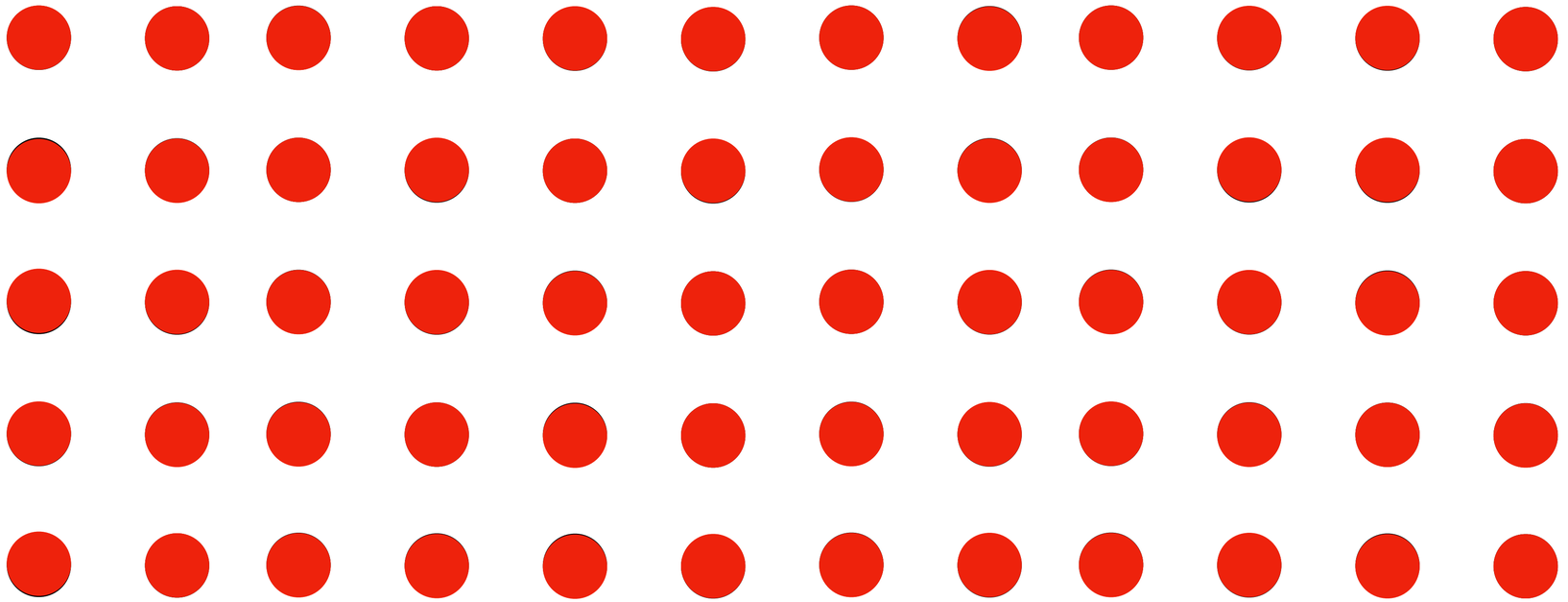
- **Gossip:** Tell message to few neighbours (only once), they do the same

Idea: Replace Send-to-All with Gossip



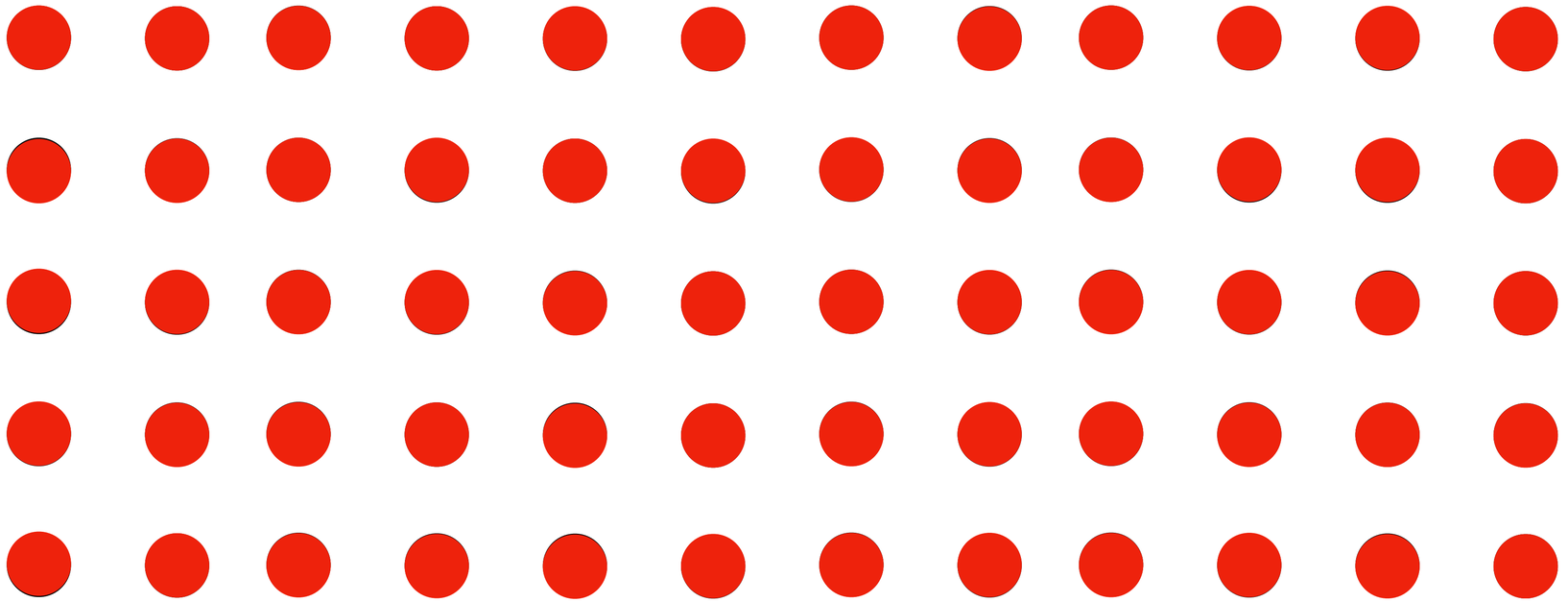
- **Gossip:** Tell message to few neighbours (only once), they do the same

Idea: Replace Send-to-All with Gossip



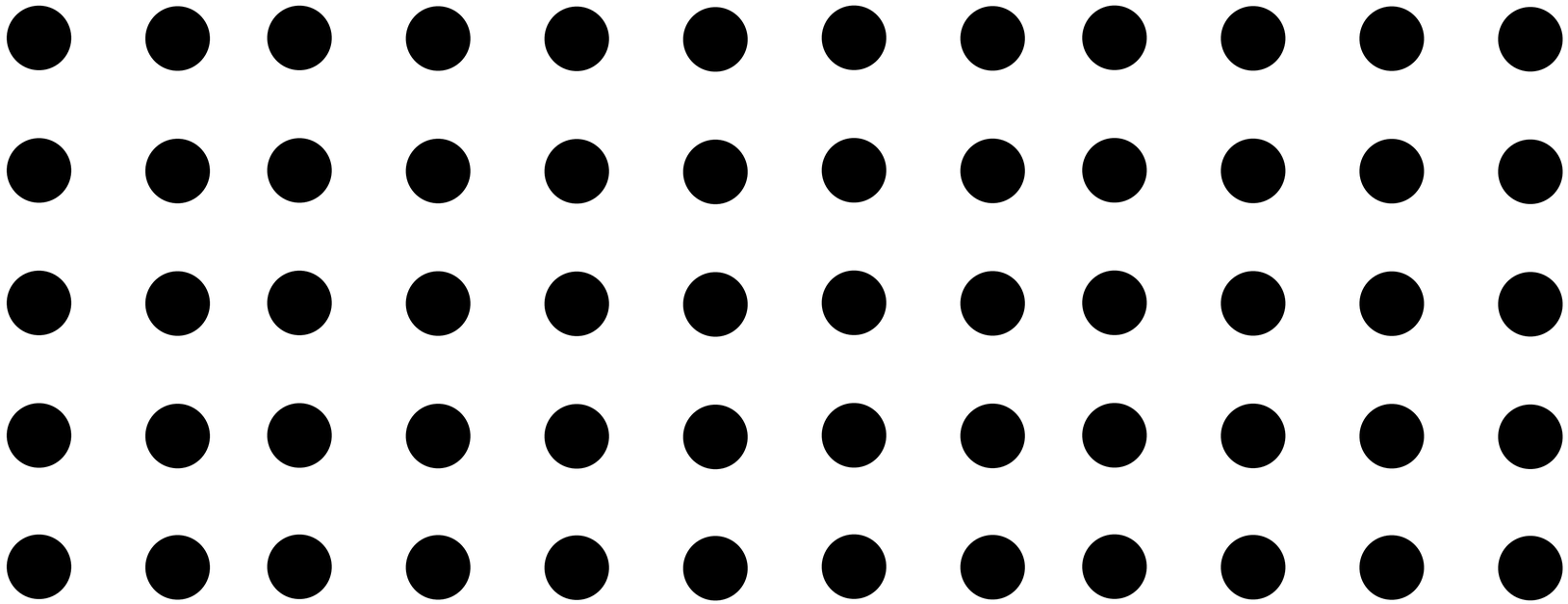
- **Gossip:** Tell message to few neighbours (only once), they do the same

Idea: Replace Send-to-All with Gossip

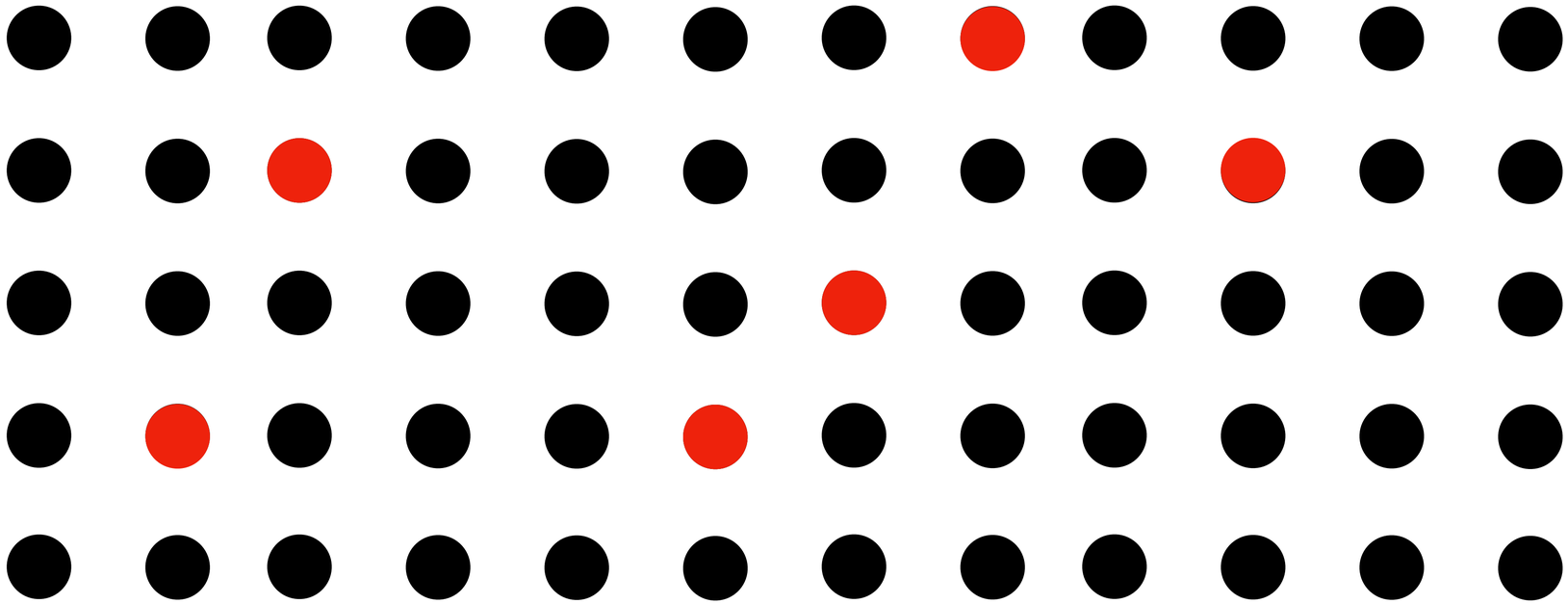


- **Gossip:** Tell message to few neighbours (only once), they do the same
- Everybody learns the message in $\log(n)$ rounds with $\tilde{O}(n)$ complexity

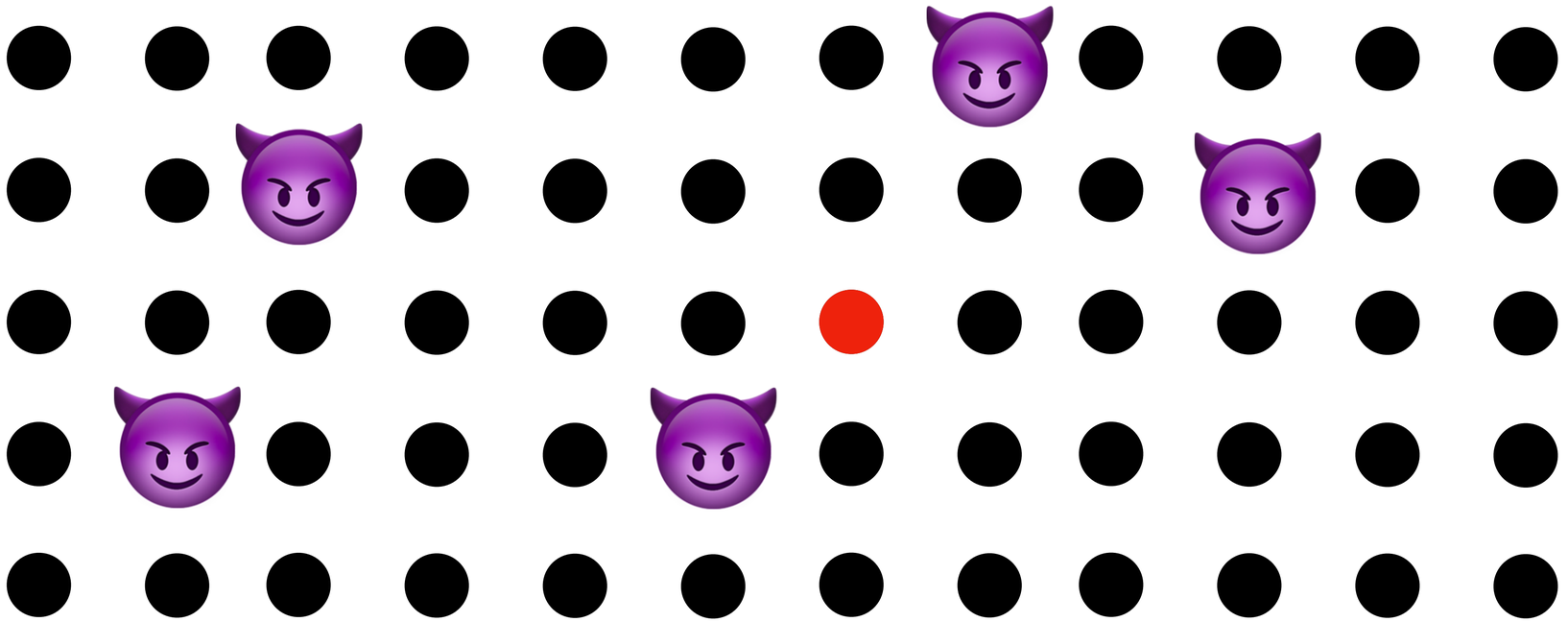
Not Possible versus Adaptive Adversary!



Not Possible versus Adaptive Adversary!

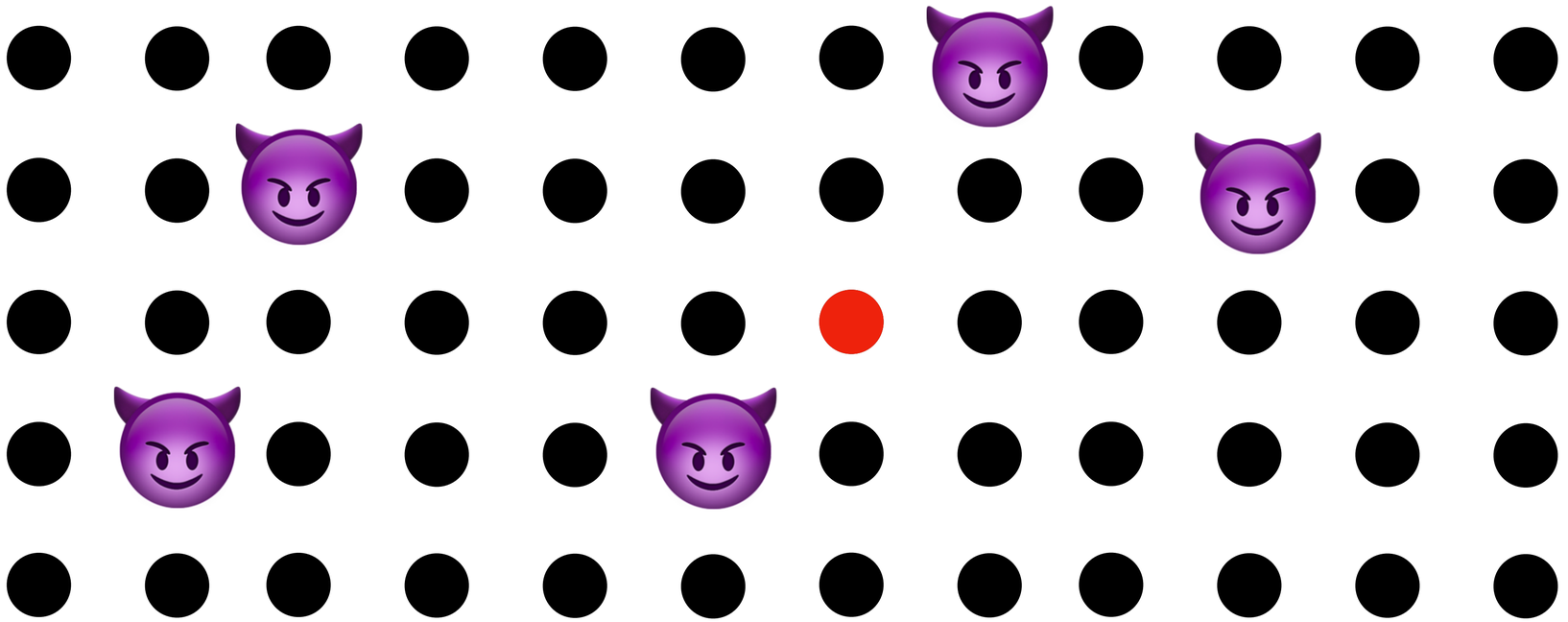


Not Possible versus Adaptive Adversary!



- Adaptive adversary can **eclipse** the sender

Not Possible versus Adaptive Adversary!



- Adaptive adversary can **eclipse** the sender
- Shuts down the process before the message spreads

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)

- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)

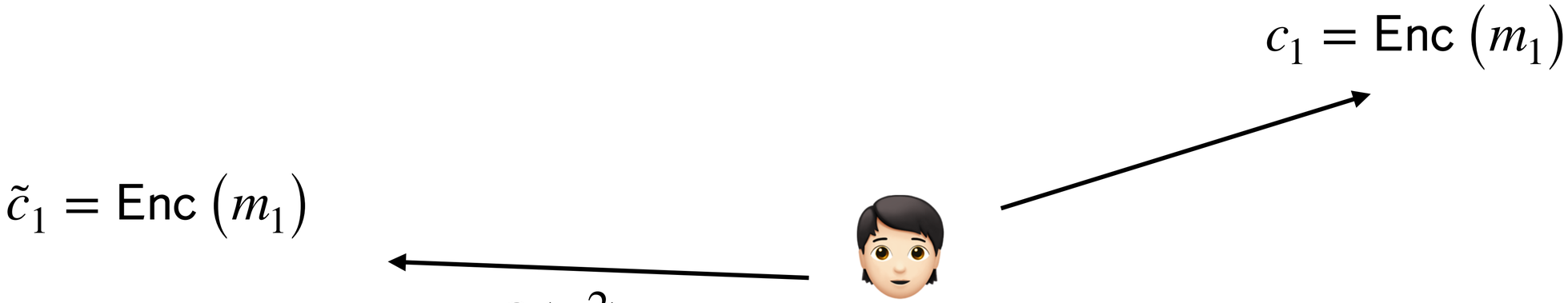
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)

$c_1 = \text{Enc}\left(m_1\right)$

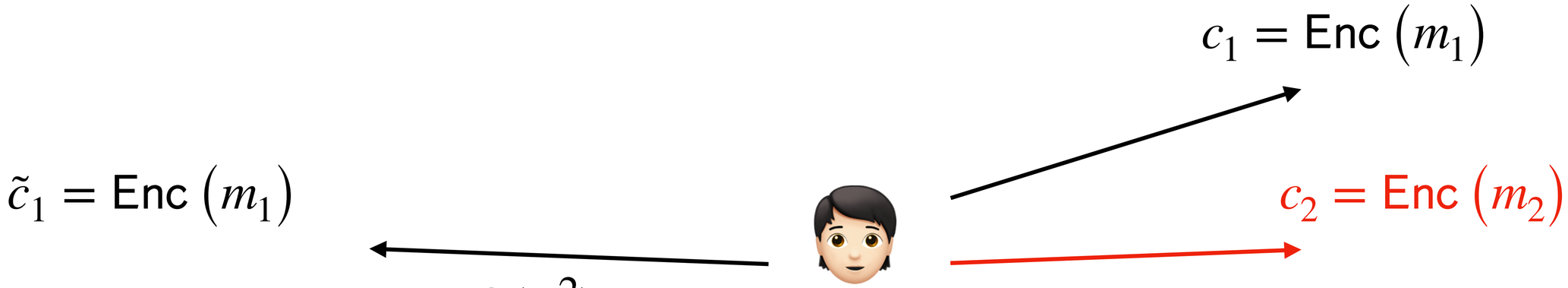
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



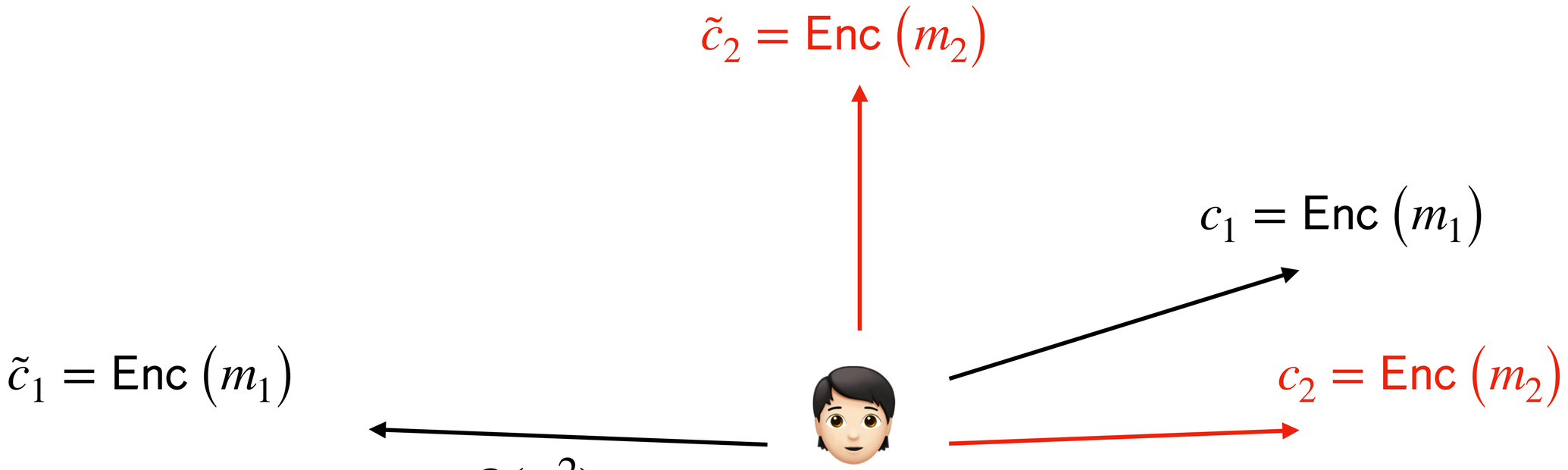
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



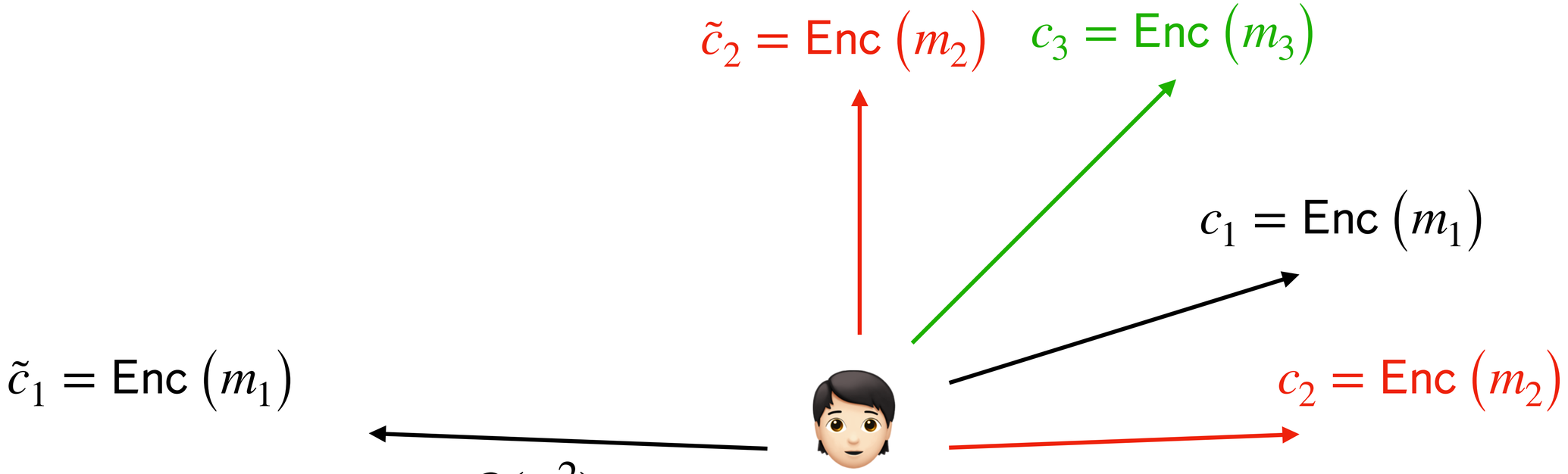
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



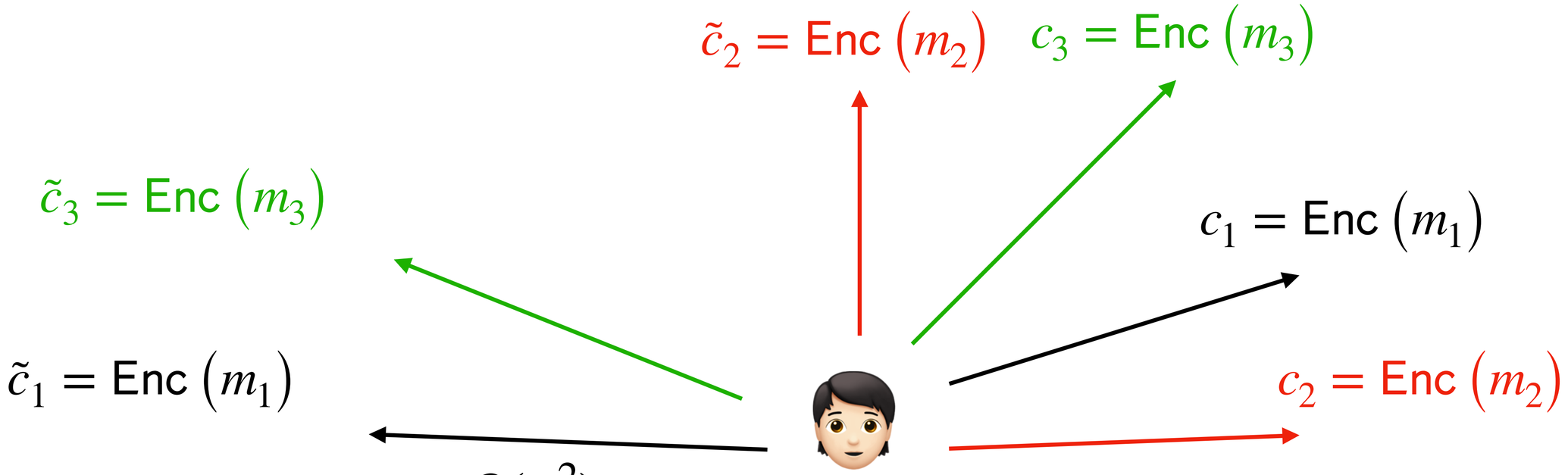
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



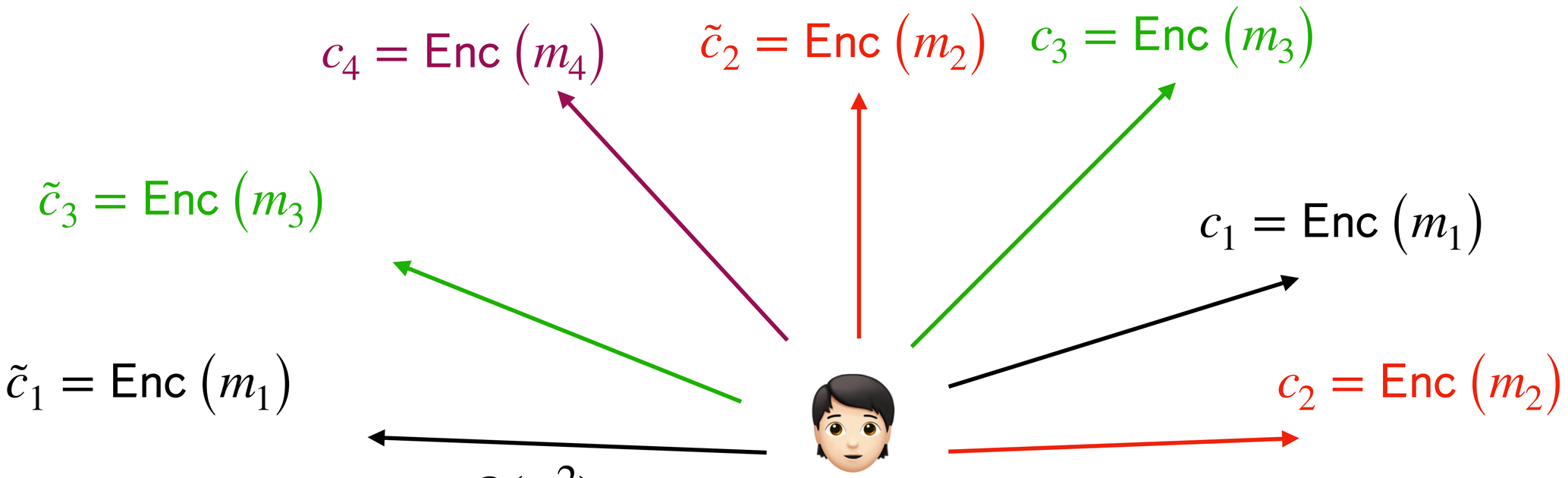
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



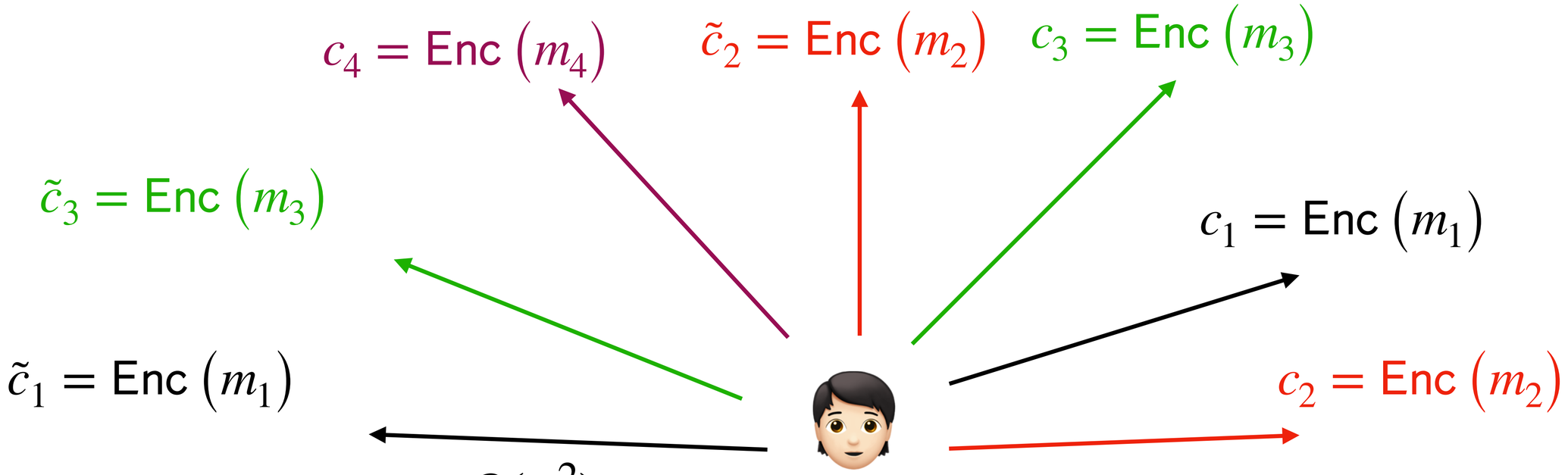
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



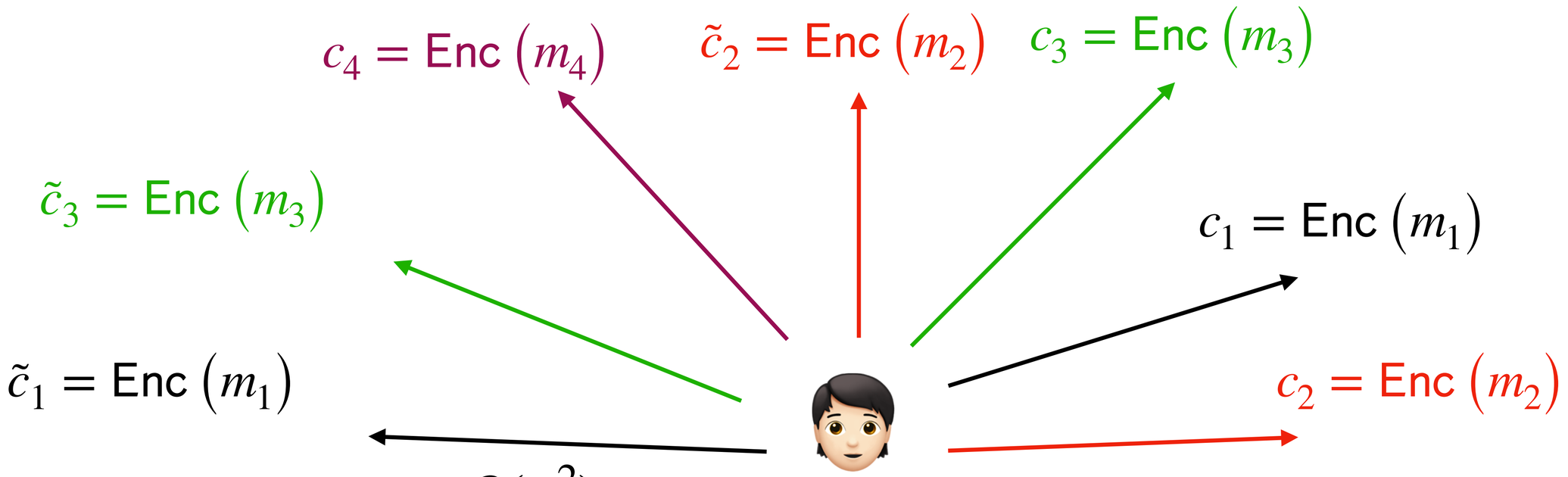
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



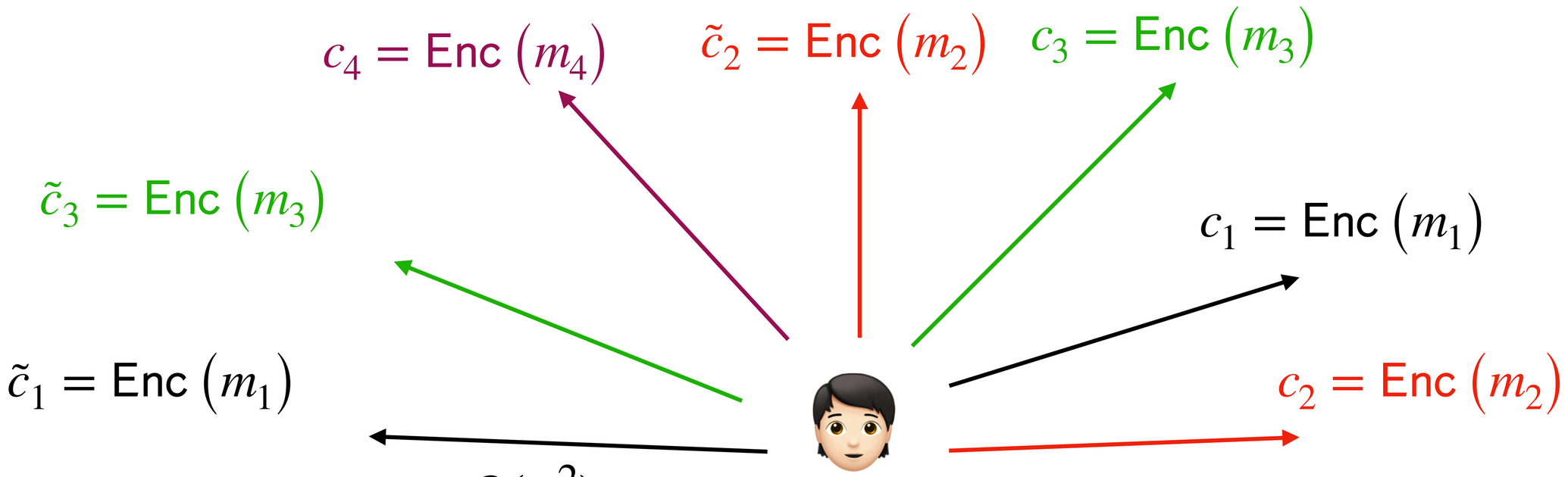
- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)
- Send **same sized ciphertext** to **every party** in **every gossip round**

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)
- Send **same sized ciphertext** to **every party** in **every gossip round**
- Instances provide **cover traffic** for each other

Key idea: Cover Traffic (Tsimos et al. CRYPTO 22)



- PBC inherently costs $O(n^2)$ communication $\Longrightarrow$ all-to-all communication is ok!
- Idea: run n gossip instances in parallel (one per sender)
- Send **same sized ciphertext** to **every party** in **every gossip round**
- Instances provide **cover traffic** for each other
- **Still costs $O(n^2\kappa\lambda)$ bits**

Two-Step Approach

Two-Step Approach

- Step 1: Reduce PBC to Binary Consensus

Two-Step Approach

- Step 1: Reduce PBC to Binary Consensus
- **Step 2:** Binary consensus in $O(n^2 \cdot \kappa)$ bits using novel DY-based gossip

Pull-Based Gossip

$\sigma_2(m)$

$\sigma_4(m')$
$\sigma_3(m')$

$\sigma_1(m)$

$\sigma_2(m)$
$\sigma_3(m')$

Pull-Based Gossip

- Ask for (pull) signatures from other parties

Pull-Based Gossip

$\sigma_2(m)$

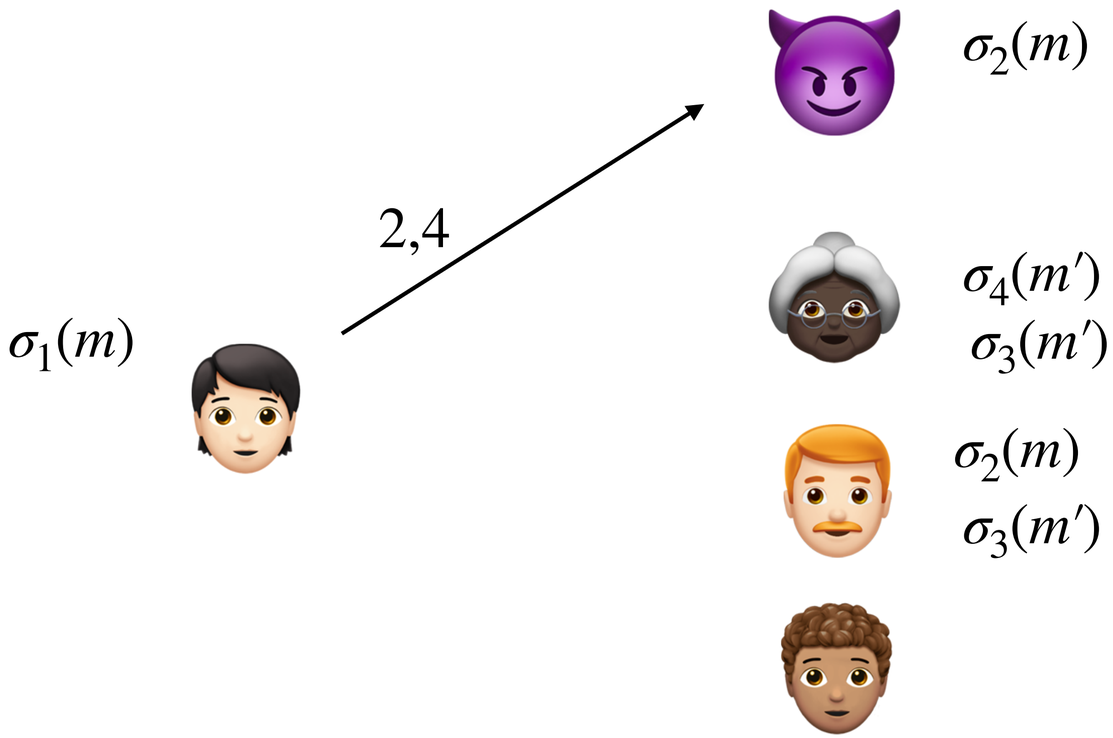
$\sigma_4(m')$
$\sigma_3(m')$

$\sigma_1(m)$

$\sigma_2(m)$
$\sigma_3(m')$

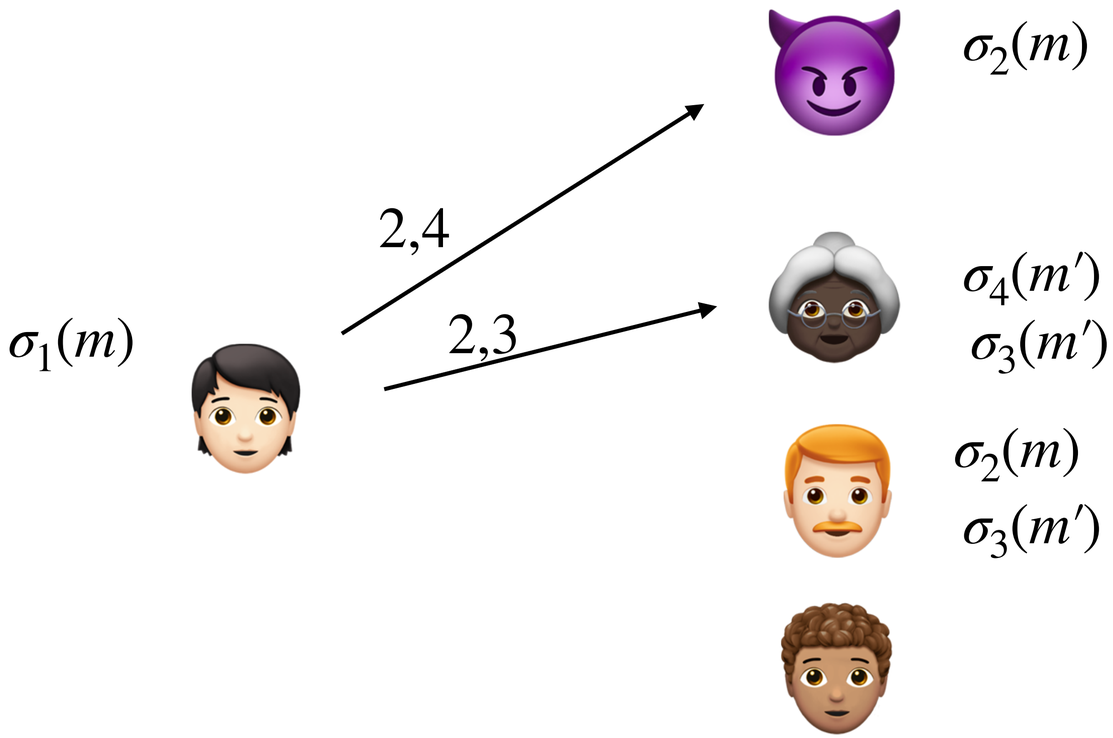
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



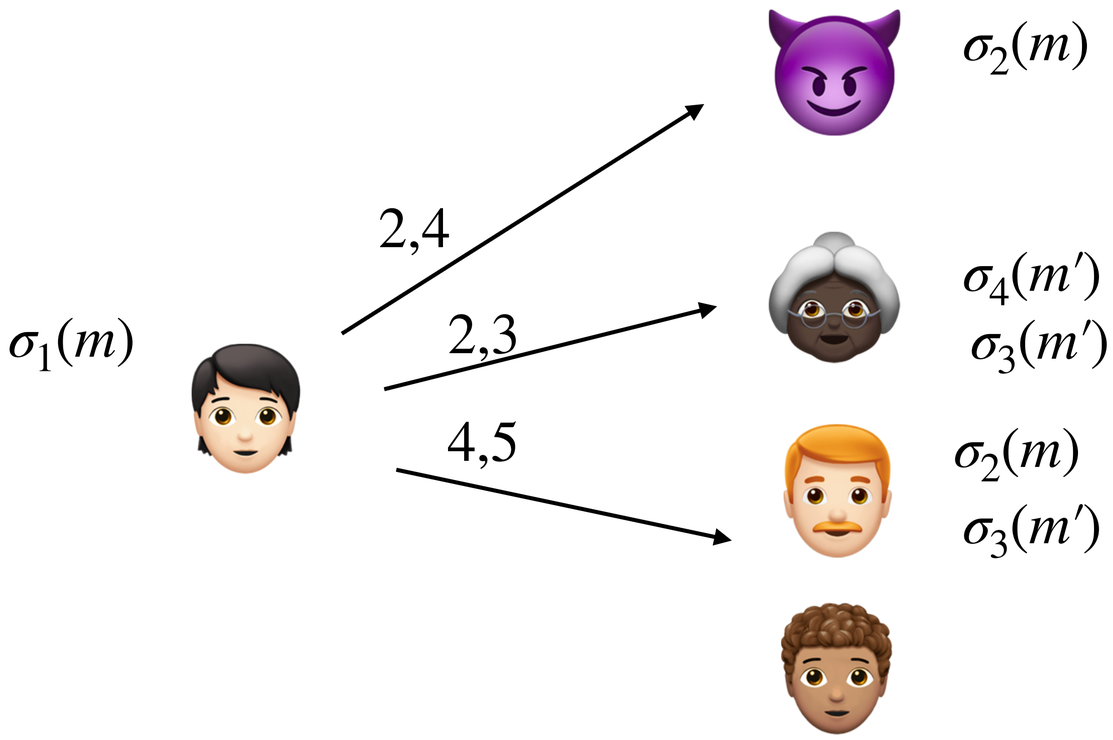
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



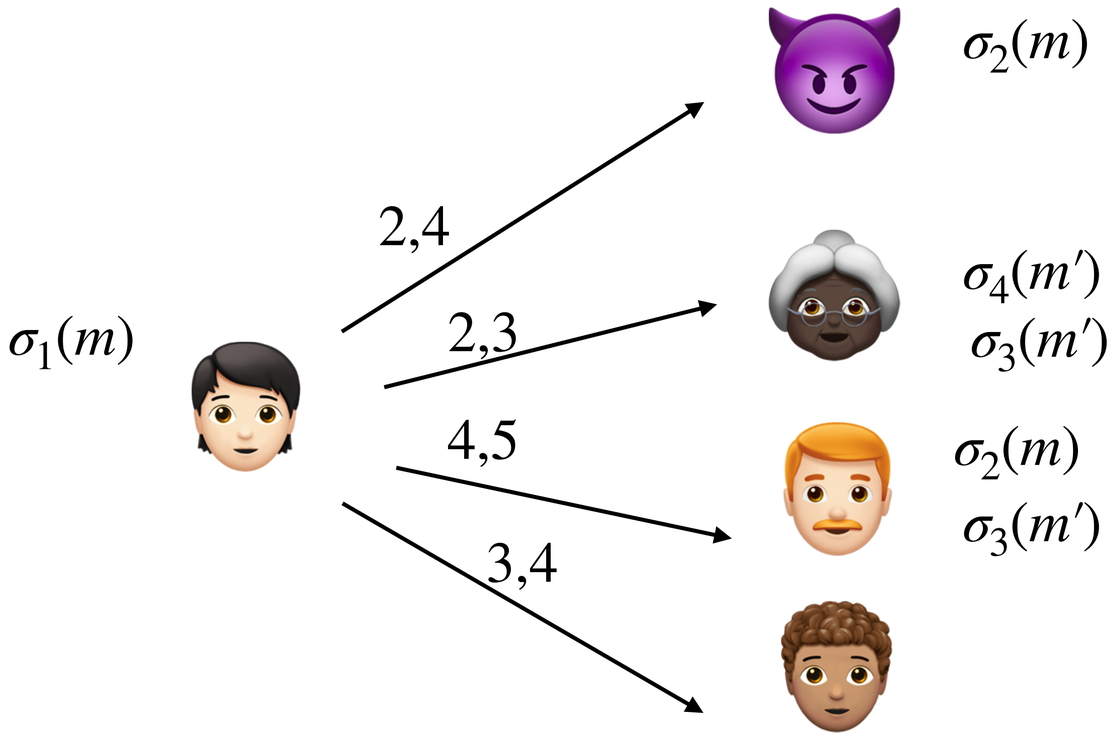
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



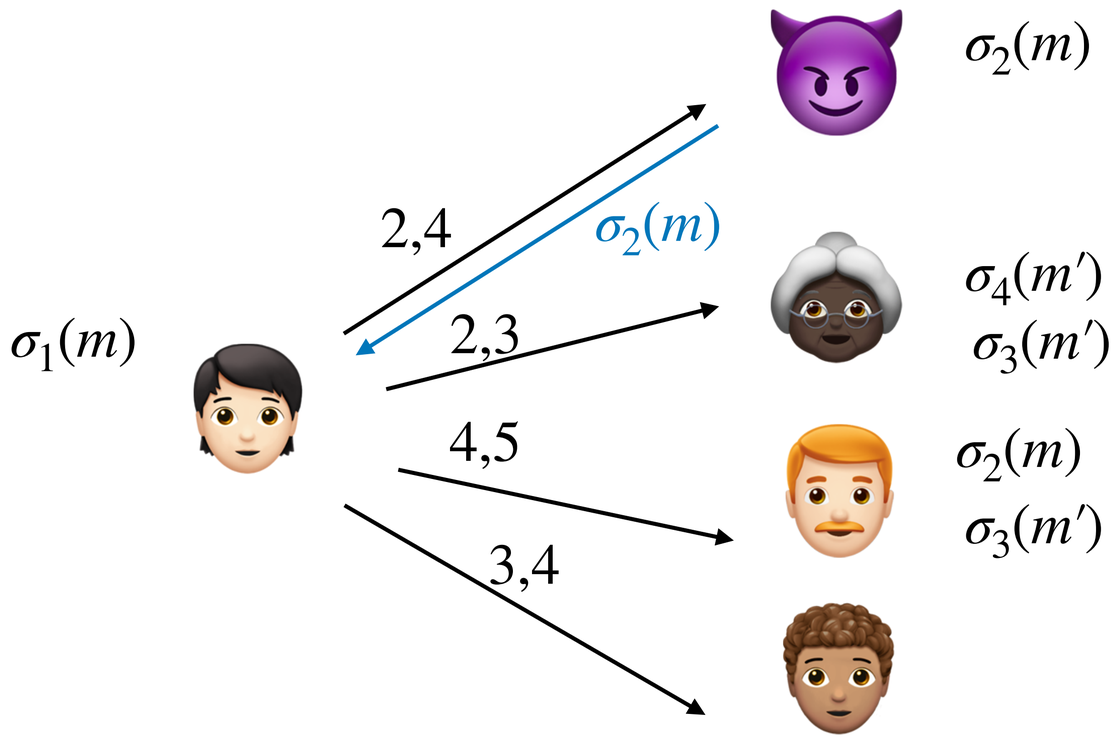
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



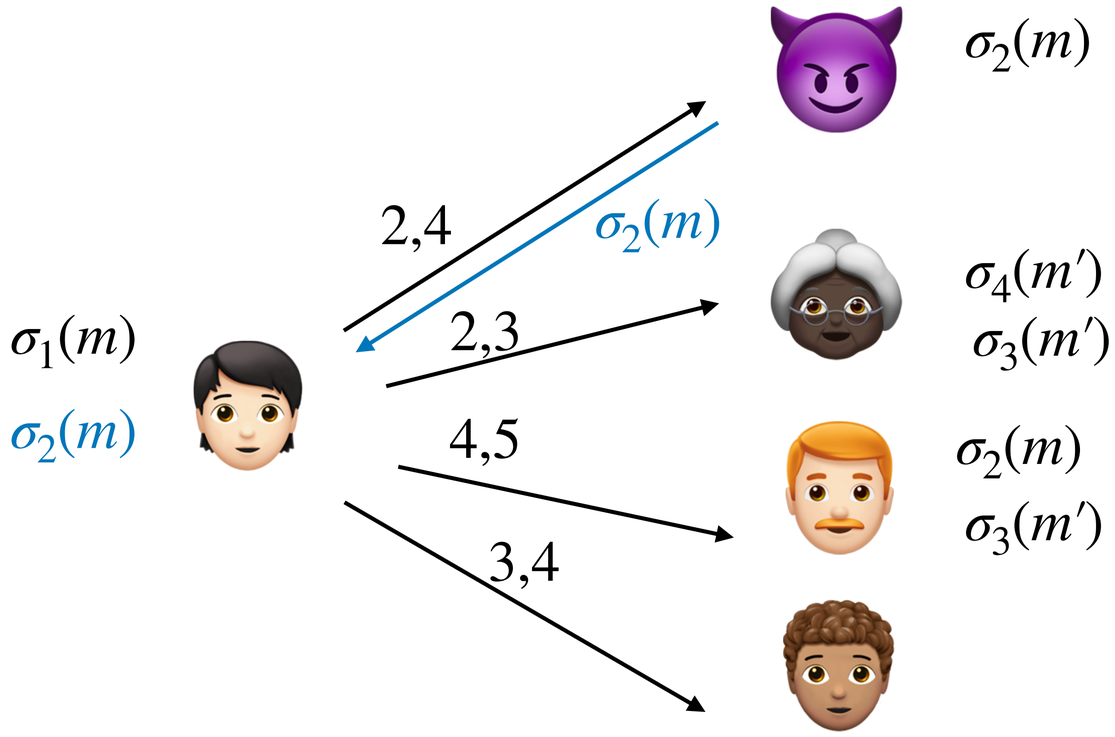
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



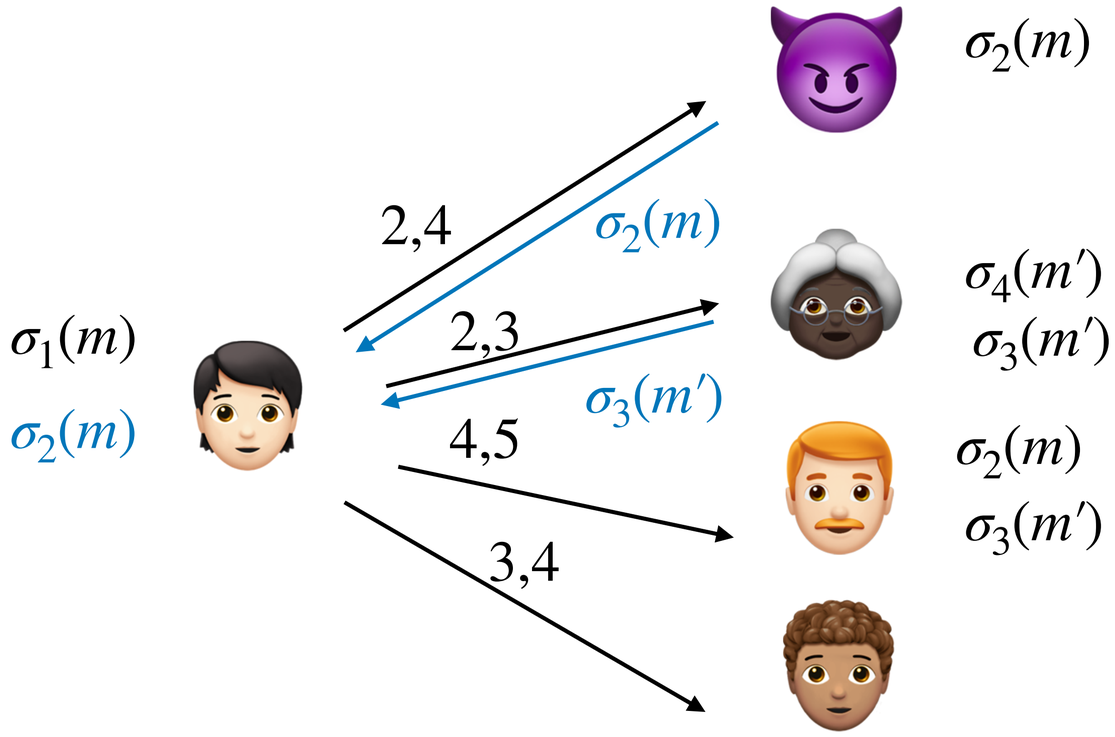
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



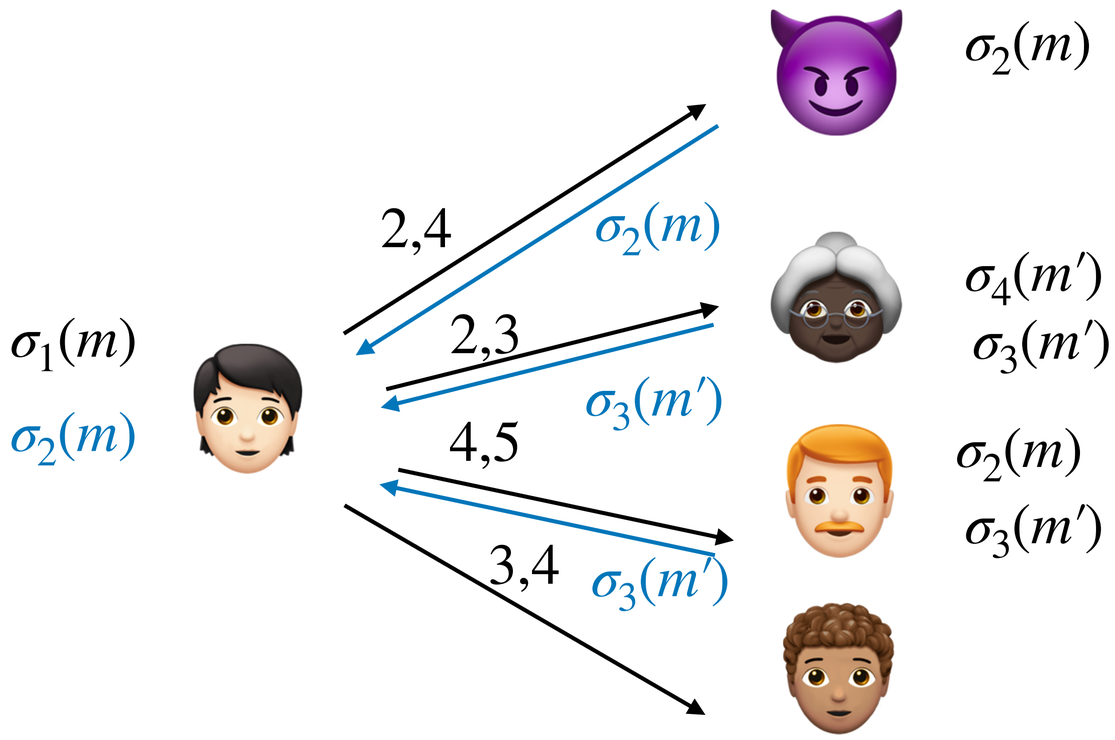
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



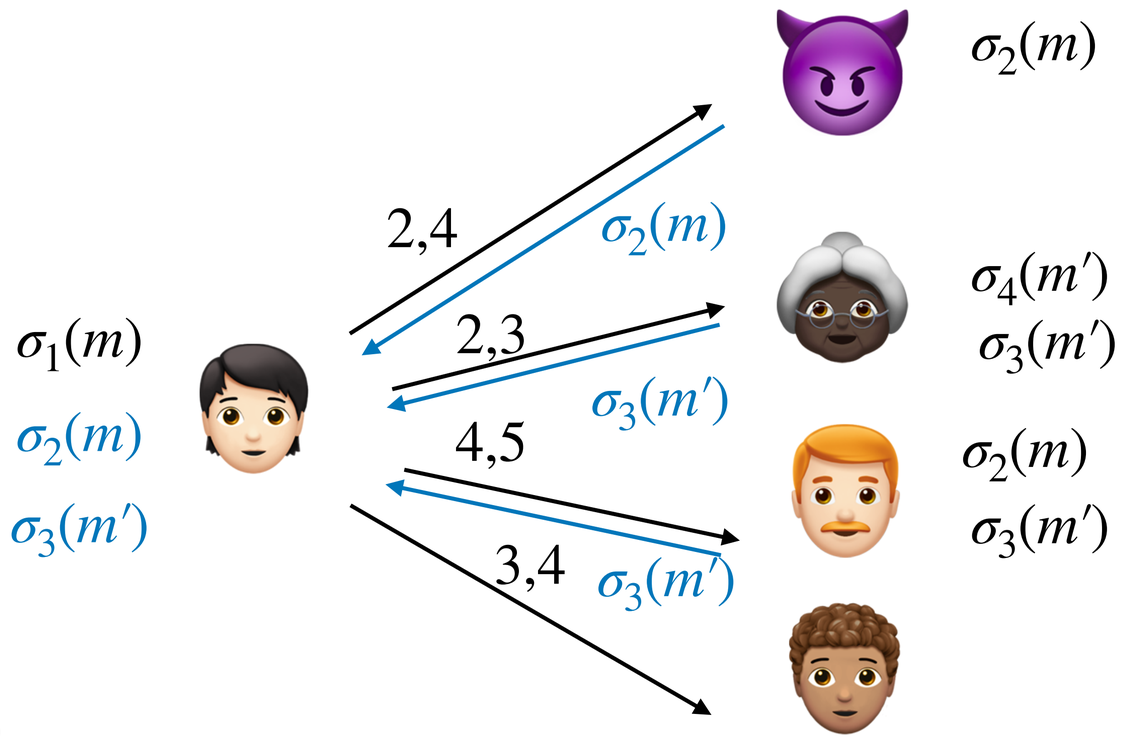
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



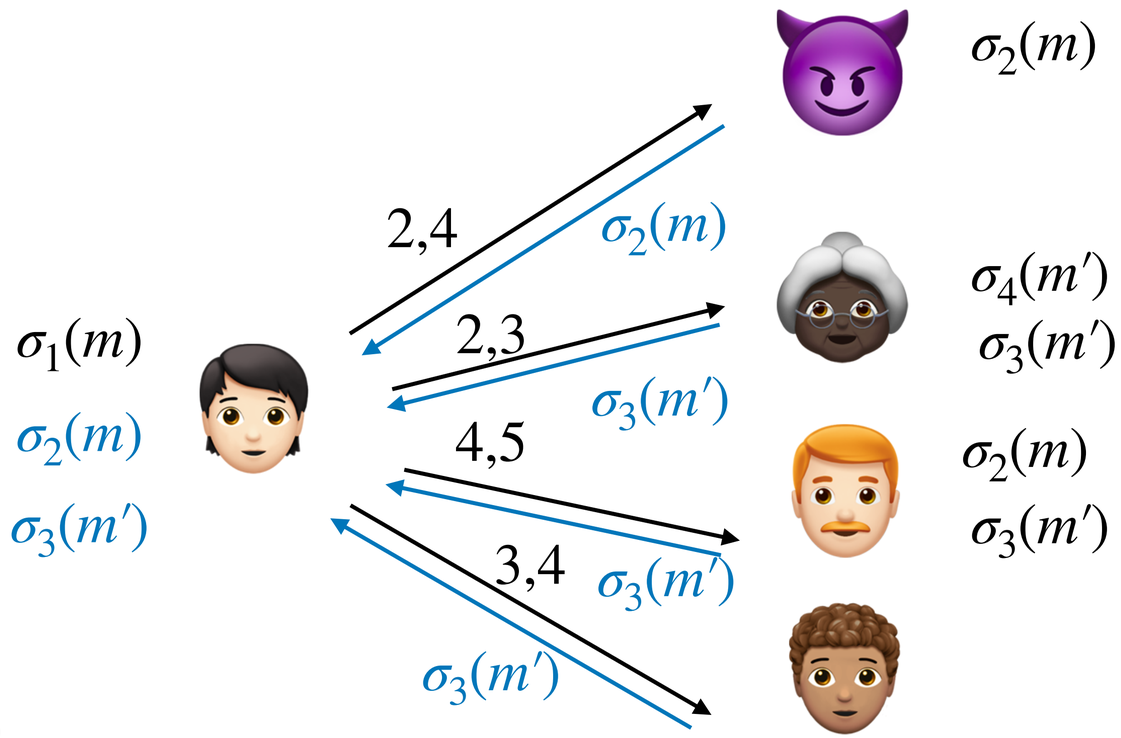
- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Pull-Based Gossip



- Ask for (pull) signatures from other parties
- Pull only indices you are missing

Issue with Naive Pulling

$\sigma_2(m)$

$\sigma_1(m)$

$\sigma_4(m')$
$\sigma_3(m')$

$\sigma_2(m)$
$\sigma_3(m')$

$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$

Issue with Naive Pulling

- Naive attempt: pull $O(1)$ missing indices

$\sigma_2(m)$

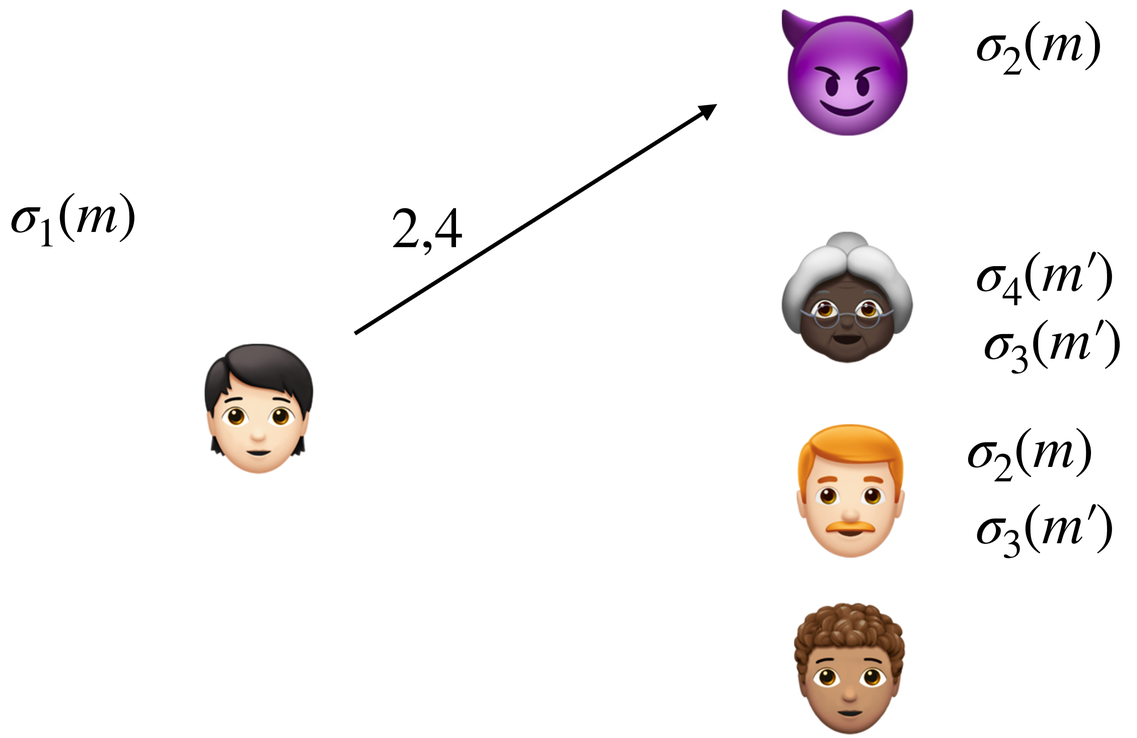
$\sigma_1(m)$

$\sigma_4(m')$
$\sigma_3(m')$

$\sigma_2(m)$
$\sigma_3(m')$

$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$

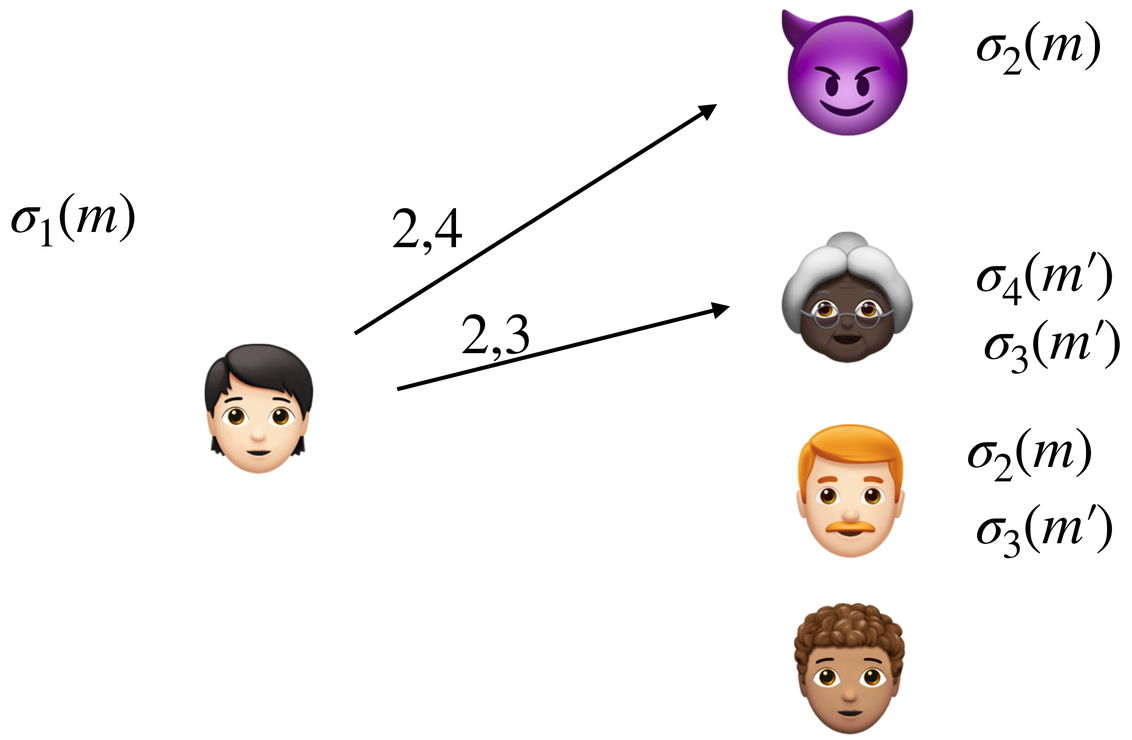
Issue with Naive Pulling



- Naive attempt: pull $O(1)$ missing indices

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

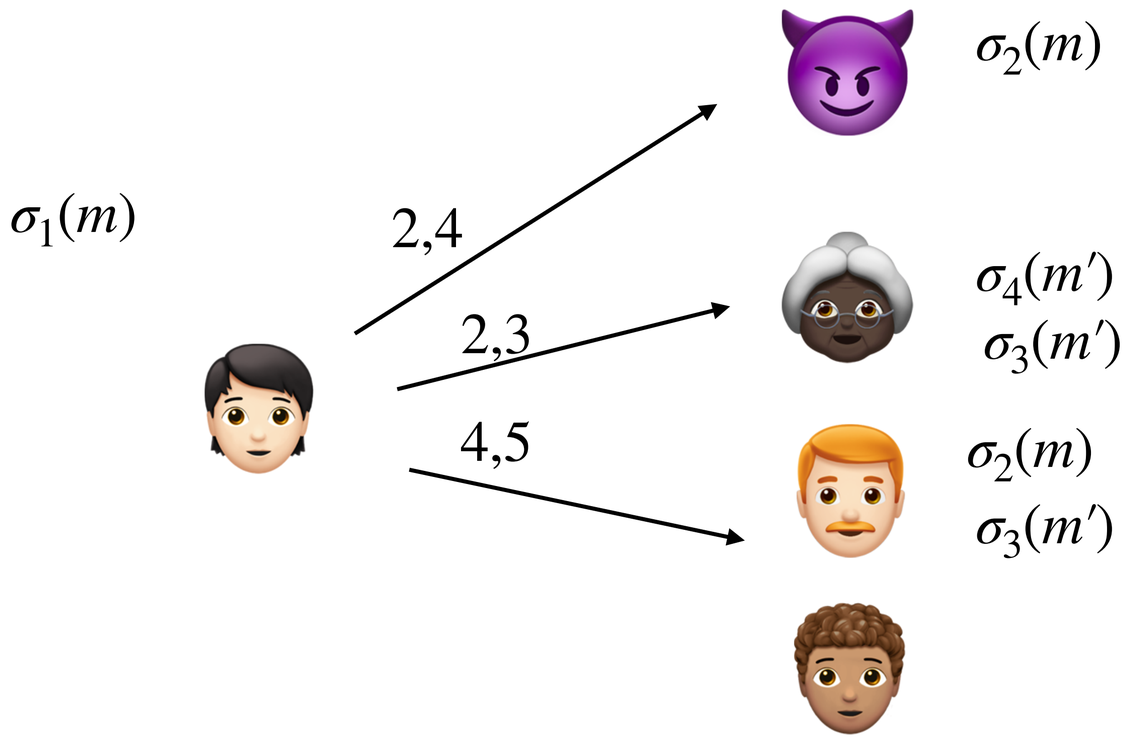
Issue with Naive Pulling



- Naive attempt: pull $O(1)$ missing indices

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

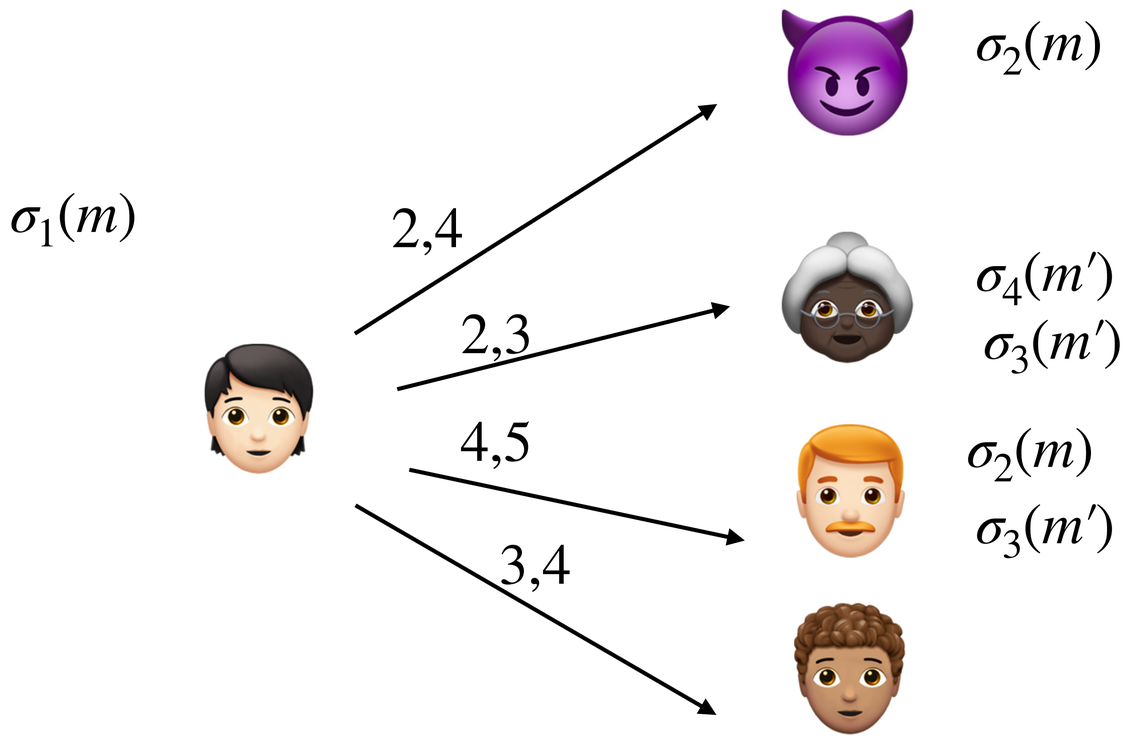


- Naive attempt: pull $O(1)$ missing indices

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

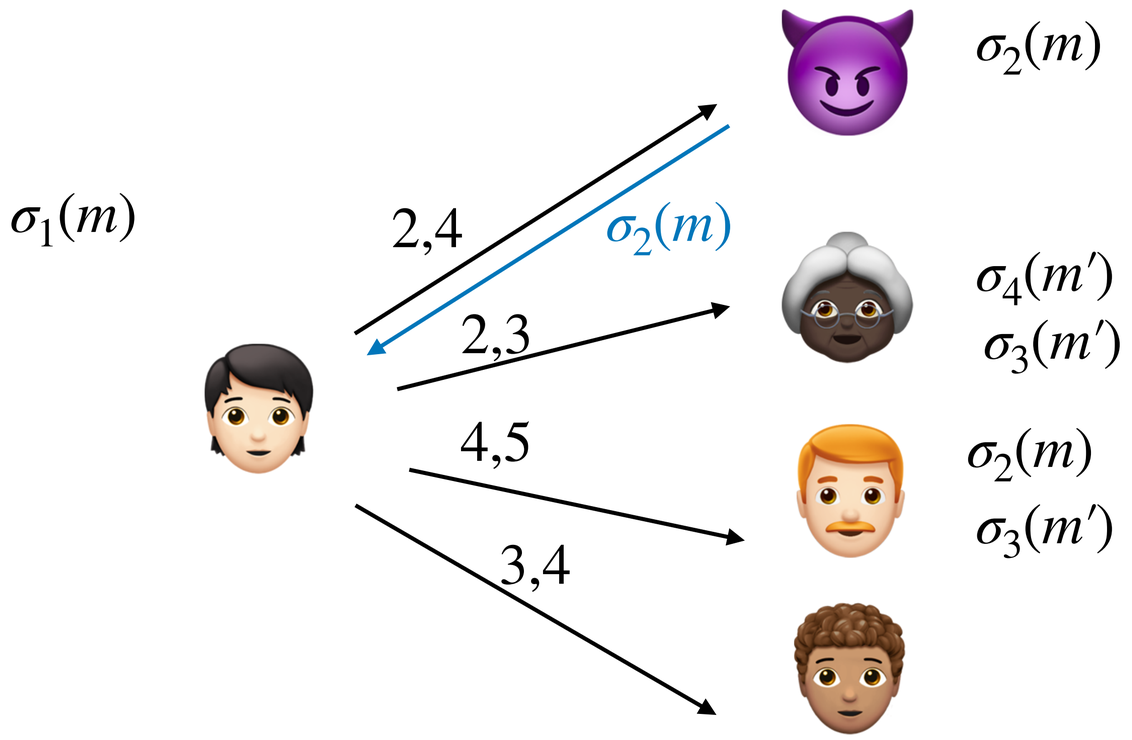
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

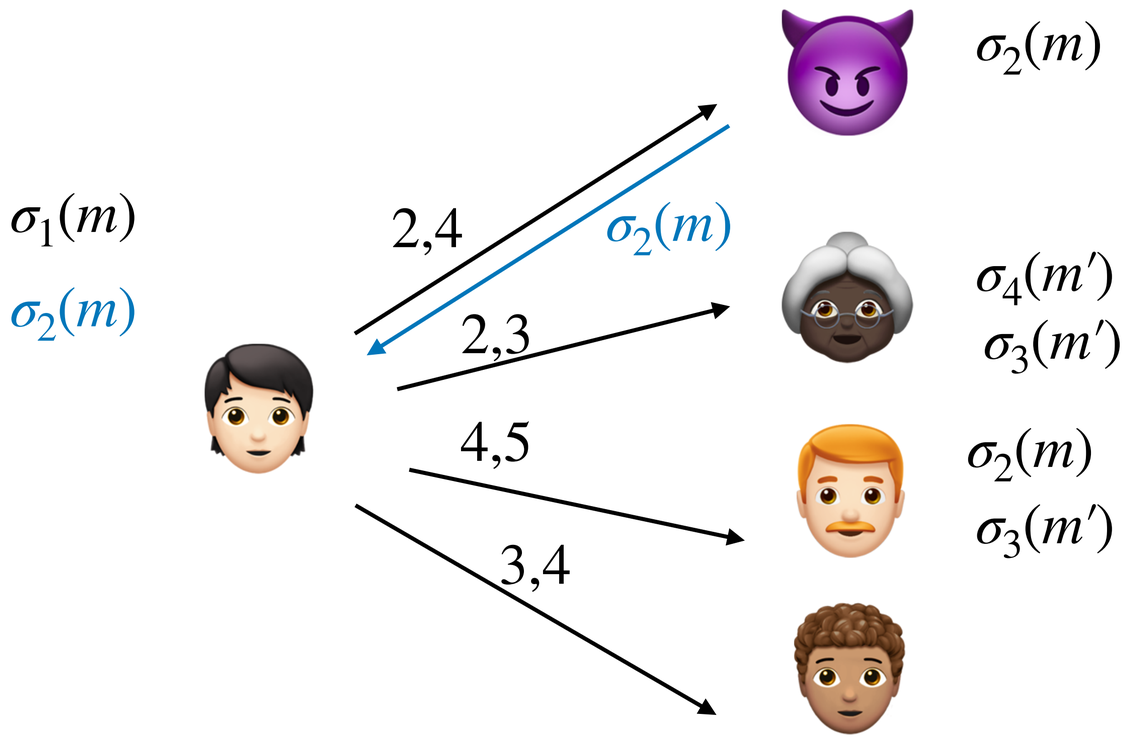
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

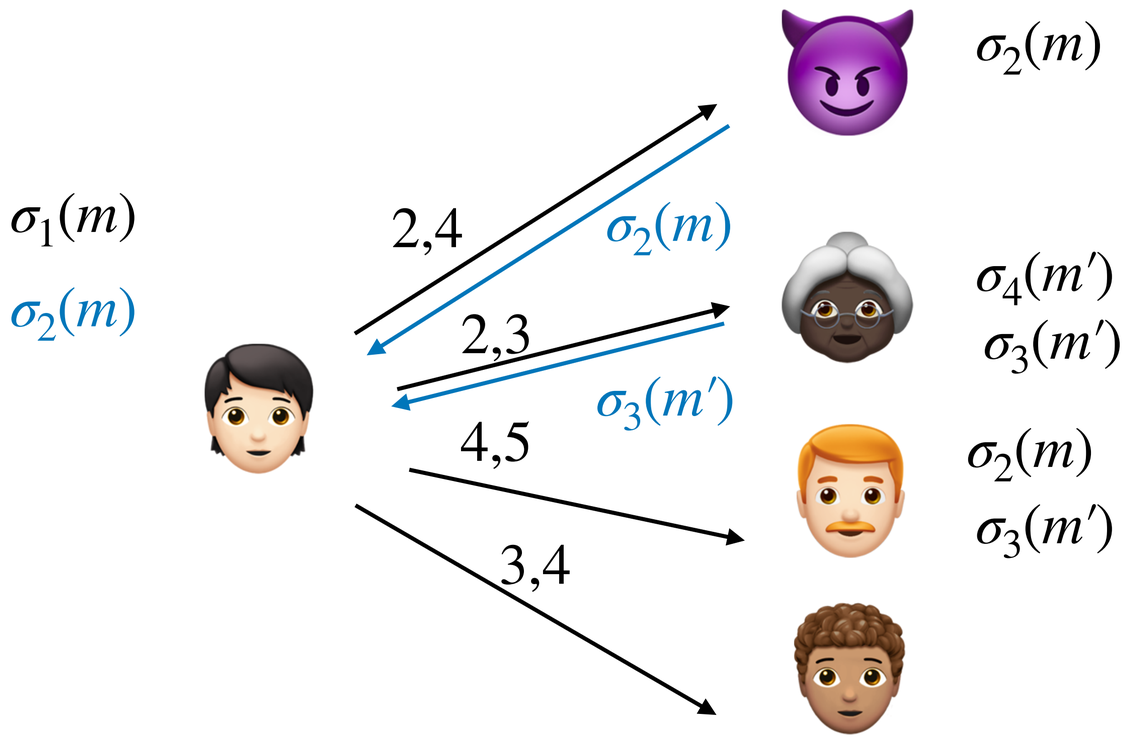
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

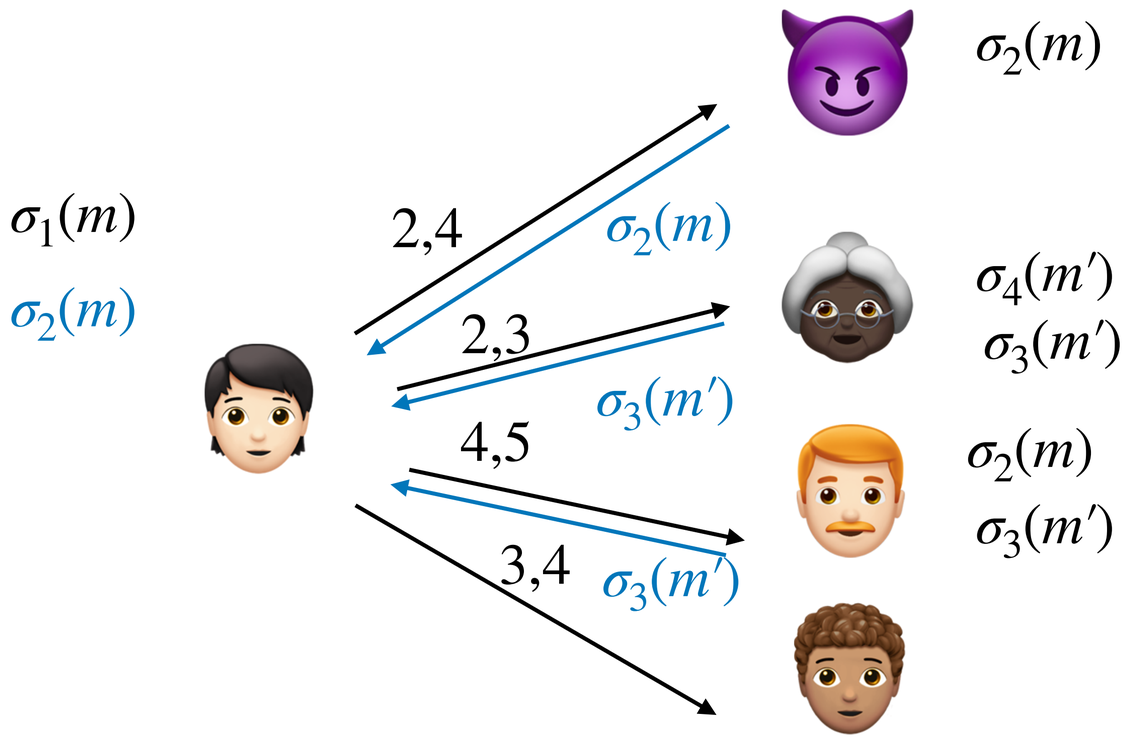
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

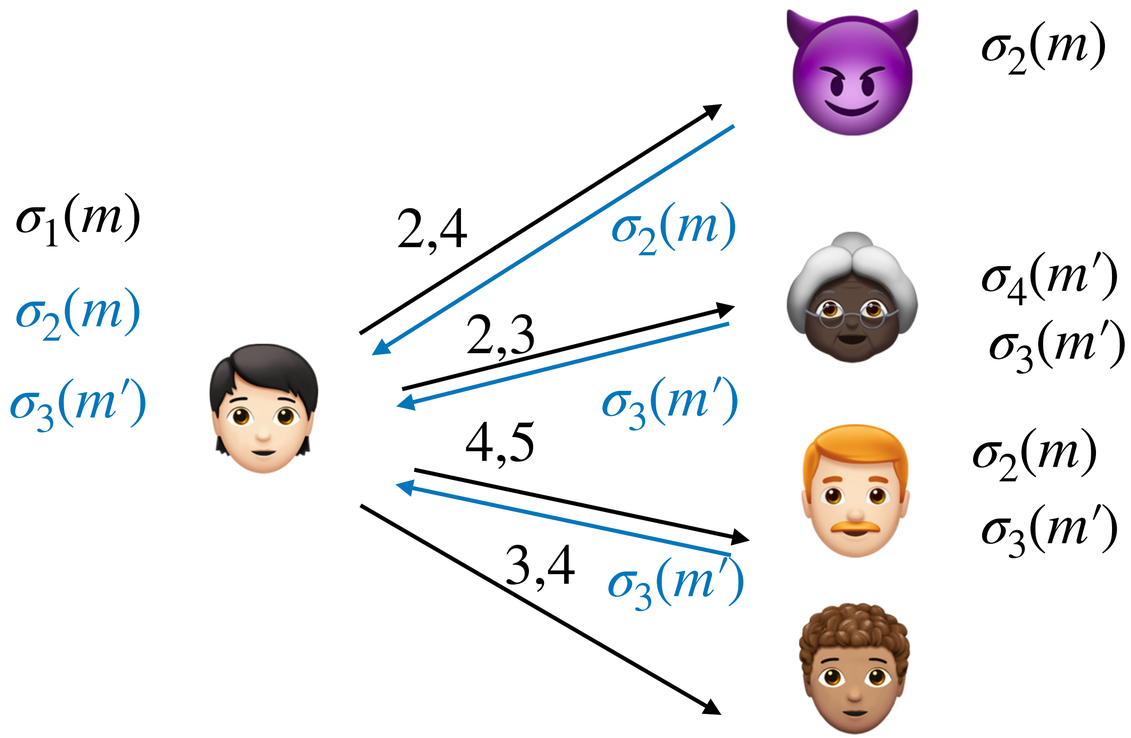
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

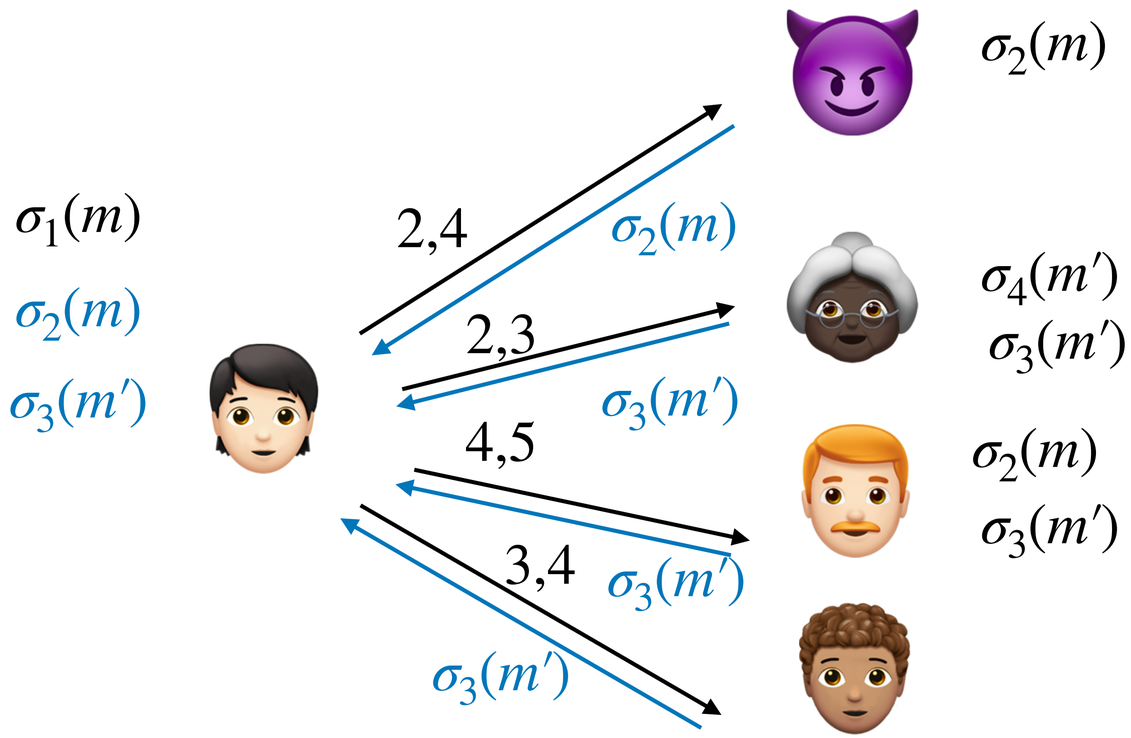
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

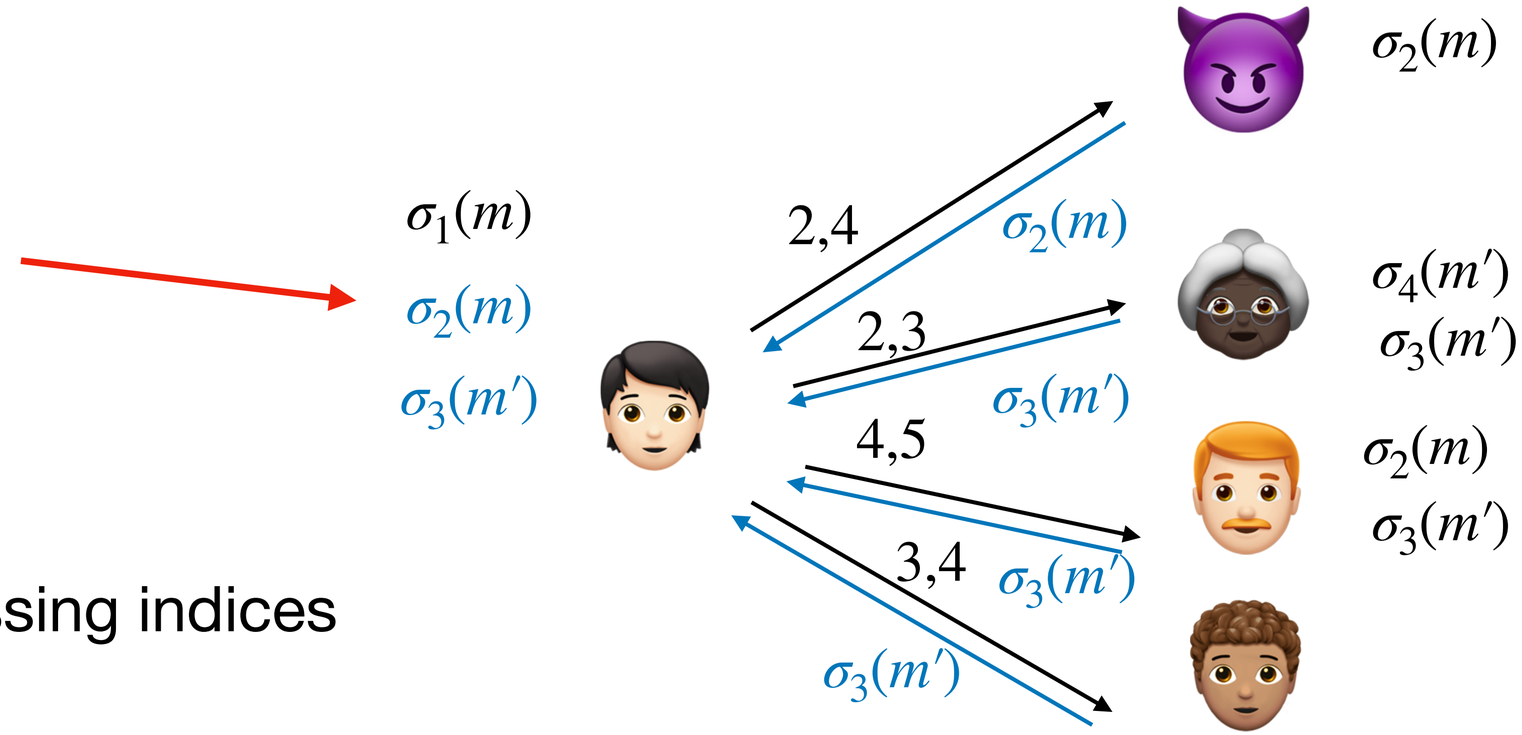
- Naive attempt: pull $O(1)$ missing indices



$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

P_1 will never query for a signature from P_2 again



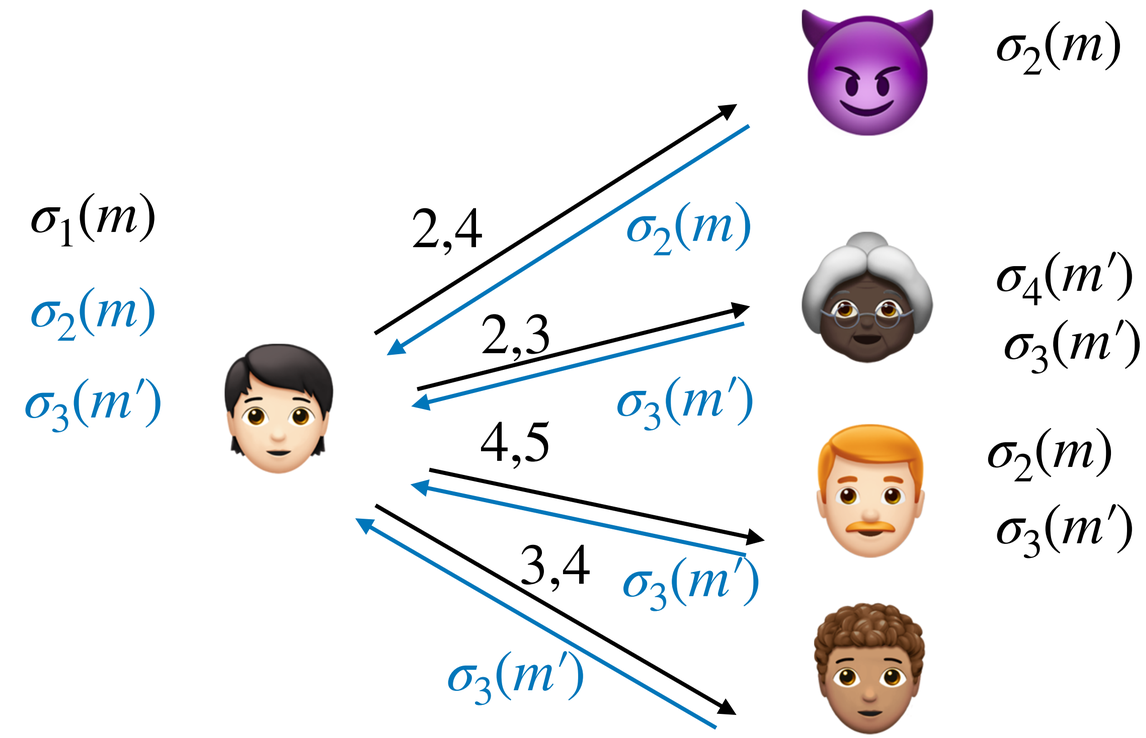
- Naive attempt: pull $O(1)$ missing indices

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

P_1 will never query for a signature from P_2 again

$\Longrightarrow$ P_1 never learns C' from an honest party!



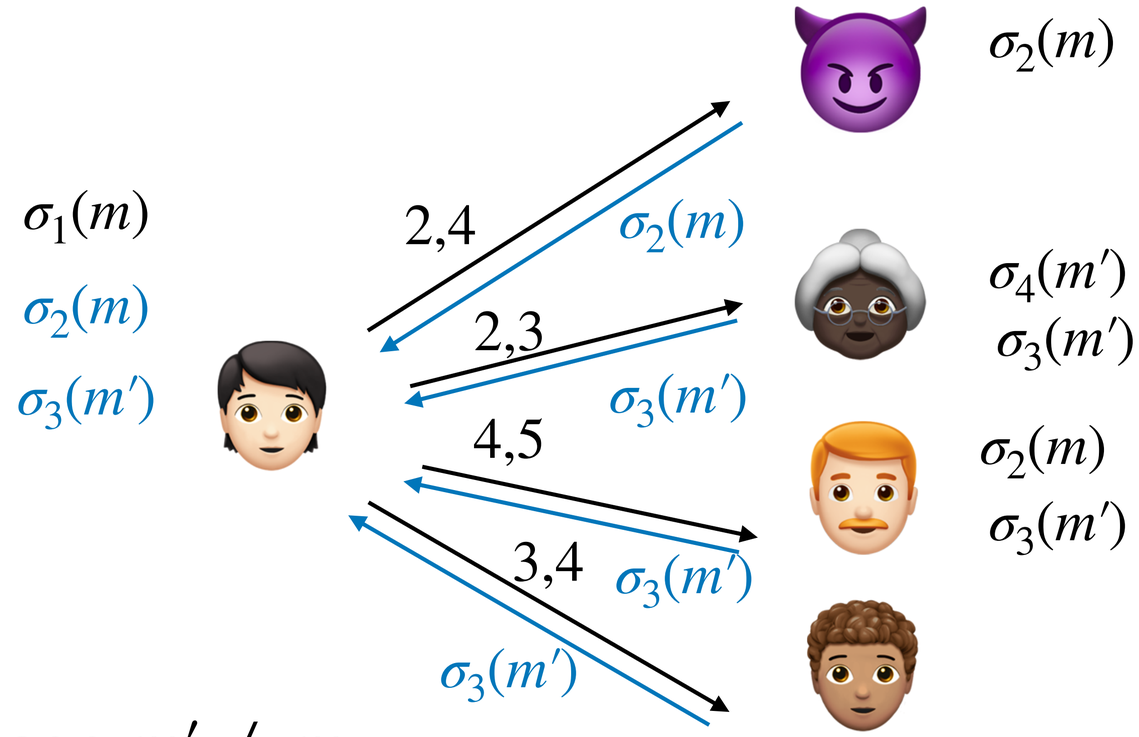
- Naive attempt: pull $O(1)$ missing indices

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

P_1 will never query for a signature from P_2 again

$\Longrightarrow$ P_1 never learns C' from an honest party!



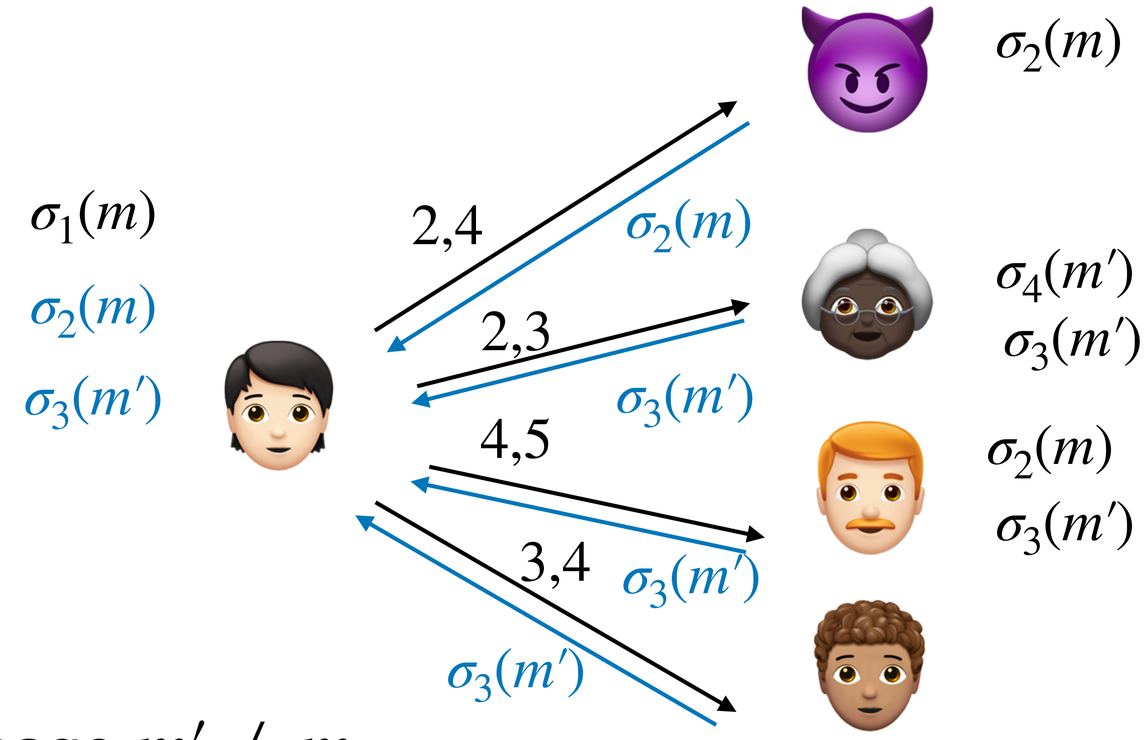
- Naive attempt: pull $O(1)$ missing indices
- **Issue:** Certificate could exist for **any** message $m' \neq m$

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

P_1 will never query for a signature from P_2 again

$\Longrightarrow$ P_1 never learns C' from an honest party!



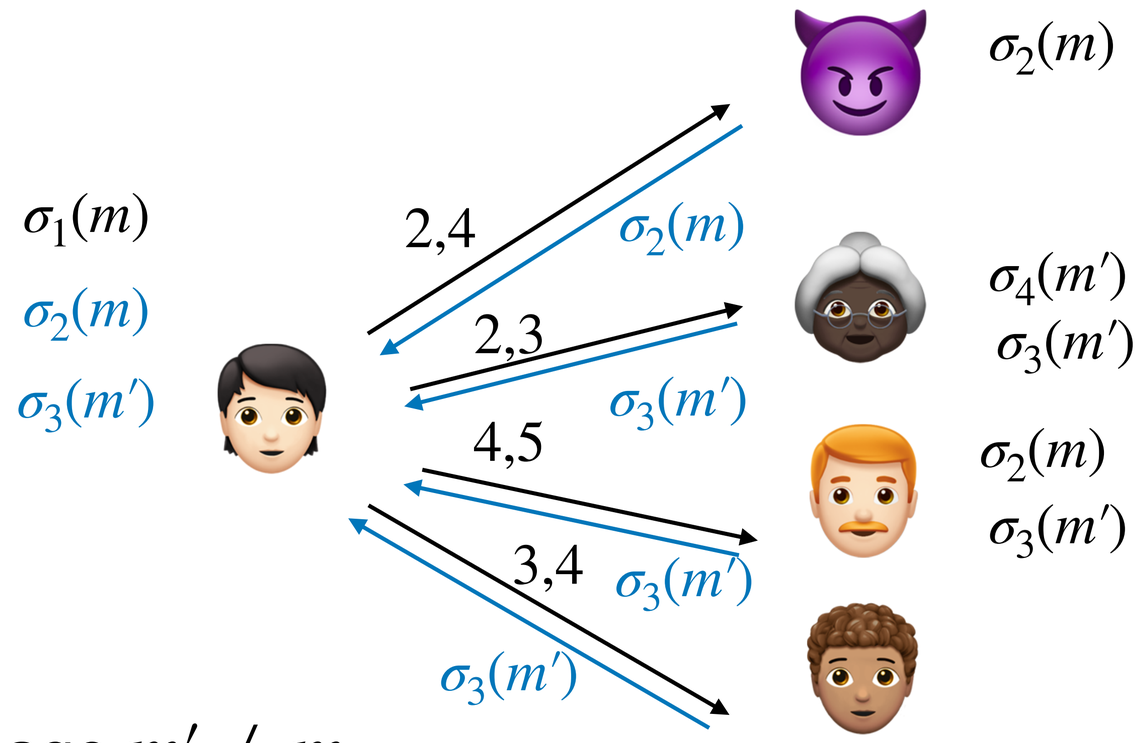
- Naive attempt: pull $O(1)$ missing indices
- **Issue:** Certificate could exist for **any** message $m' \neq m$
- $\sigma_i(m')$ for $m' \neq m$ is never pulled

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Issue with Naive Pulling

P_1 will never query for a signature from P_2 again

$\Longrightarrow$ P_1 never learns C' from an honest party!



- Naive attempt: pull $O(1)$ missing indices
- **Issue:** Certificate could exist for **any** message $m' \neq m$
- $\sigma_i(m')$ for $m' \neq m$ is never pulled
- Prevents signatures in certificate from spreading

$$C' = \{\sigma_2(m'), \sigma_3(m'), \sigma_5(m')\}$$

Pull-Based Gossip with Testing

P'

P

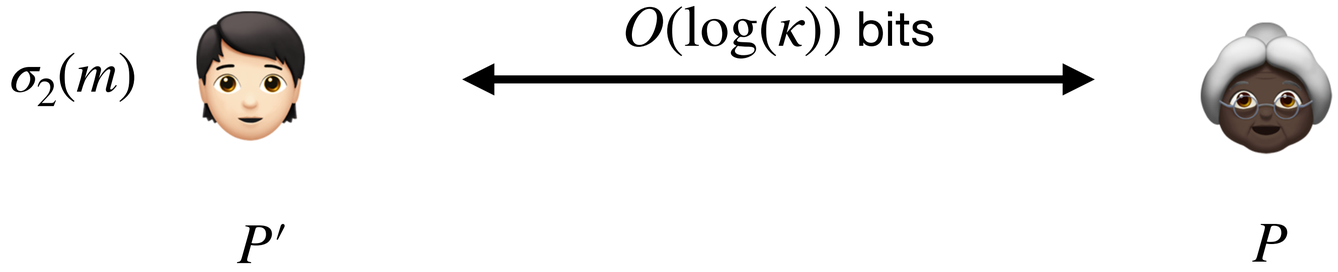
Pull-Based Gossip with Testing

$\sigma_2(m)$ 

P'

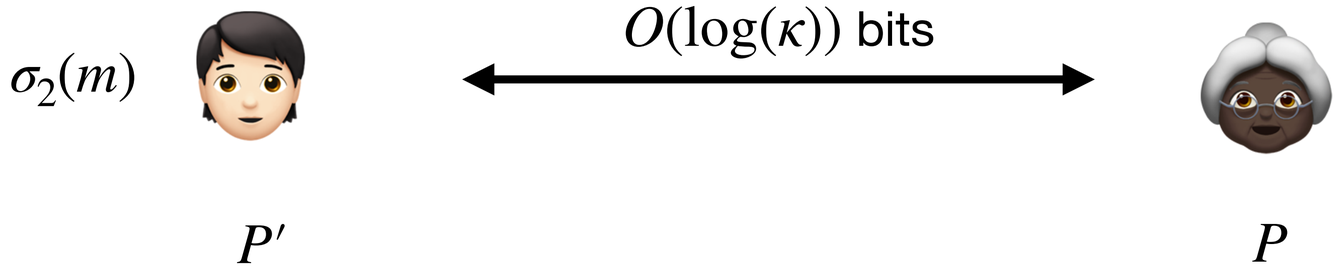
P

Pull-Based Gossip with Testing



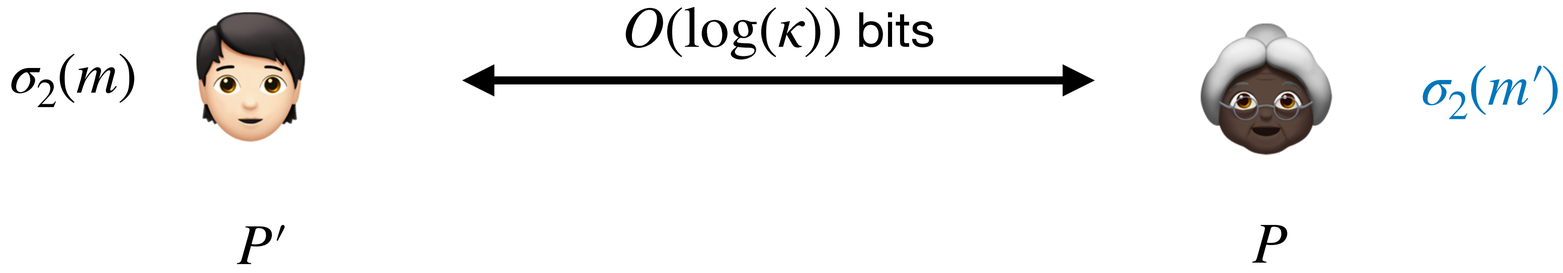
- Test for new information before pulling $\Longrightarrow O(\log(\kappa))$ extra bits per index

Pull-Based Gossip with Testing



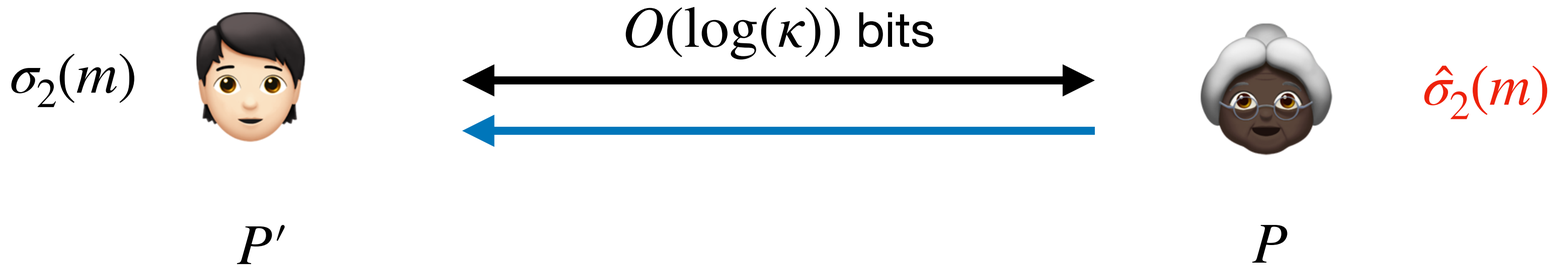
- Test for new information before pulling $\Longrightarrow O(\log(\kappa))$ extra bits per index
- Pull only if test indicates a different signature

Pull-Based Gossip with Testing



- Test for new information before pulling $\Longrightarrow O(\log(\kappa))$ extra bits per index
- Pull only if test indicates a different signature

Pull-Based Gossip with Testing



- Test for new information before pulling $\Longrightarrow O(\log(\kappa))$ extra bits per index
- Pull only if test indicates a different signature
- Technical issue: signatures on **same** message can be syntactically different

Issues with Naive Testing

P'

P

Issues with Naive Testing

C'

P'

P

- Incomplete certificate C'; missing **single index**

Issues with Naive Testing

C'

P'

C

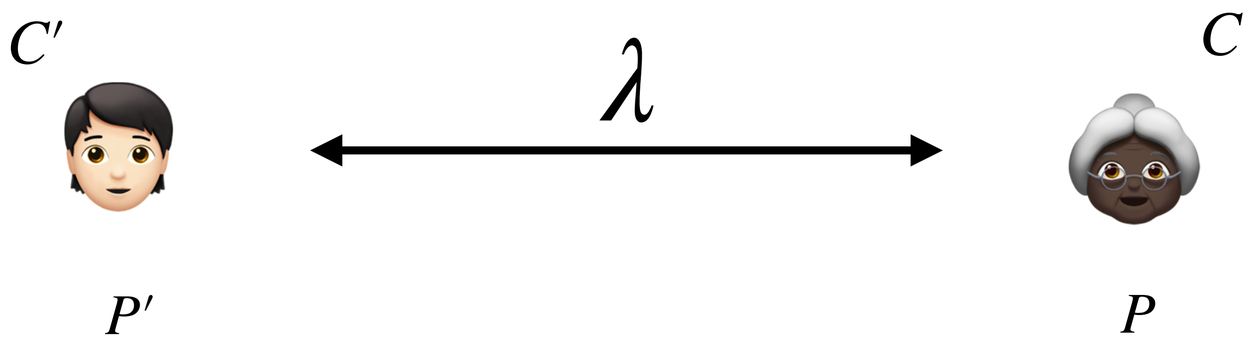
P

- Incomplete certificate C'; missing **single index**
- Complete C; has **different** signatures on superset of indices in C' on **same message** m

Issues with Naive Testing

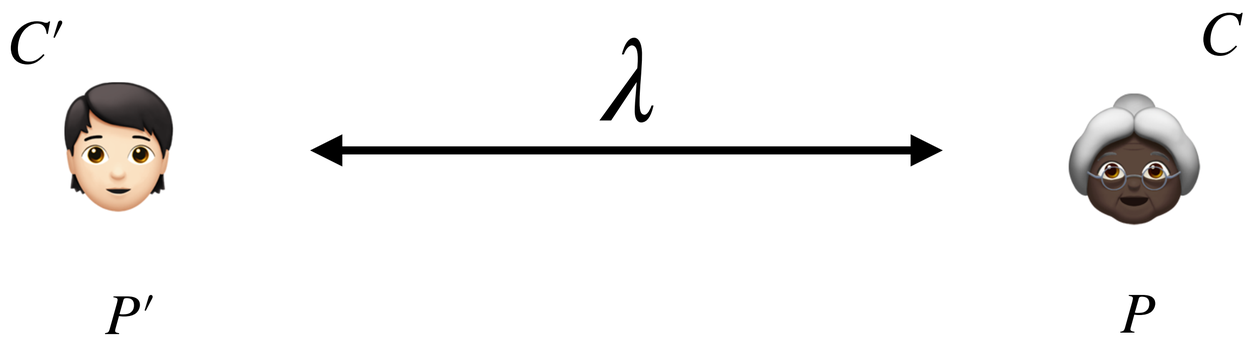
- Incomplete certificate C'; missing **single index**
- Complete C; has **different** signatures on superset of indices in C' on **same message** m
- Naive: P' tests for $O(\lambda)$ random indices missing from C' per gossip round

Issues with Naive Testing



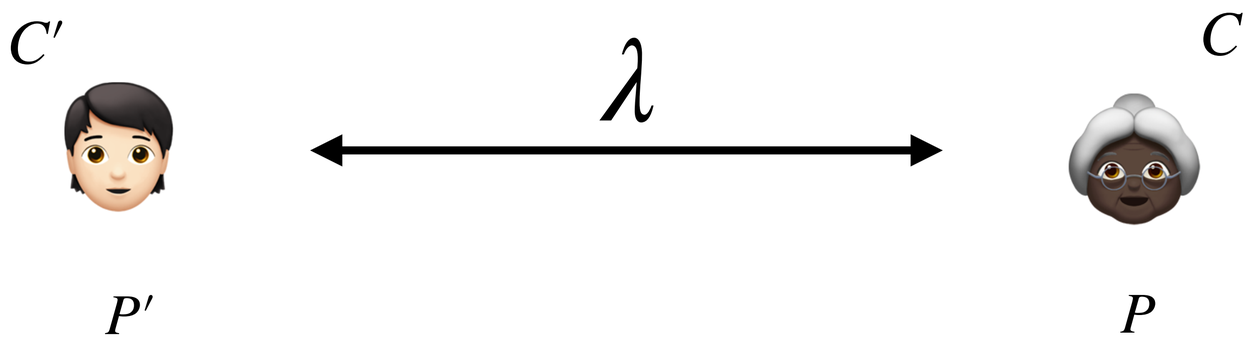
- Incomplete certificate C'; missing **single index**
- Complete C; has **different** signatures on superset of indices in C' on **same message** m
- Naive: P' tests for $O(\lambda)$ random indices missing from C' per gossip round

Issues with Naive Testing



- Incomplete certificate C'; missing **single index**
- Complete C; has **different** signatures on superset of indices in C' on **same message** m
- Naive: P' tests for $O(\lambda)$ random indices missing from C' per gossip round
- All tests positive $\implies$ missing index is pulled with probability $1/n$

Issues with Naive Testing



- Incomplete certificate C'; missing **single index**
- Complete C; has **different** signatures on superset of indices in C' on **same message** m
- Naive: P' tests for $O(\lambda)$ random indices missing from C' per gossip round
- All tests positive $\implies$ missing index is pulled with probability $1/n$
- λ rounds to learn missing index with probability $1 - 2^{-\lambda} \implies$ still costs $O(n^2 \cdot \kappa \cdot \lambda)$ bits

Increasing the Probability of Pulling

P_i

P

Increasing the Probability of Pulling

C'

P_i

C

P

Increasing the Probability of Pulling

C'

P_i

C

P

C is a full certificate for m

Increasing the Probability of Pulling

C'

P_i

C

P

C is a full certificate for m

- For each P_i:

Increasing the Probability of Pulling

C'

P_i

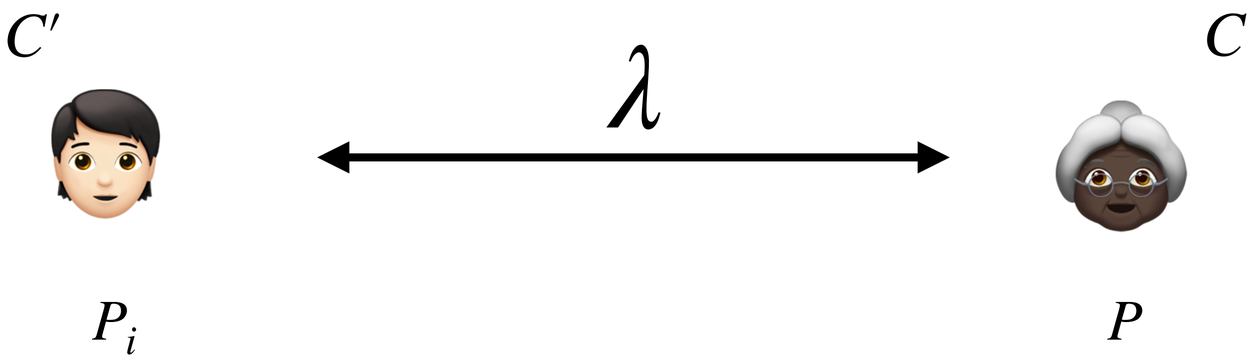
C

P

C is a full certificate for m

- For each P_i:
  - P samples $O(\lambda)$ signatures in C

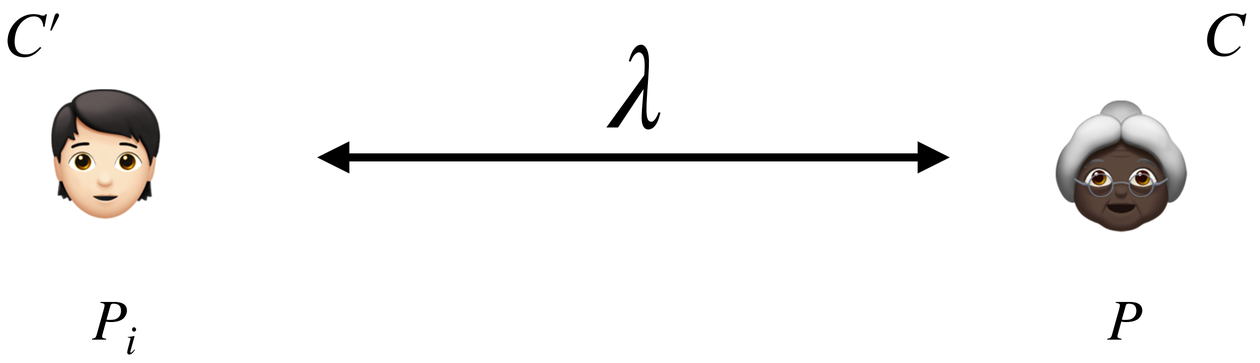
Increasing the Probability of Pulling



C is a full certificate for m

- For each P_i:
  - P samples $O(\lambda)$ signatures in C
  - P finds (via testing) $O(1)$ signatures among sample that P_i doesn't have

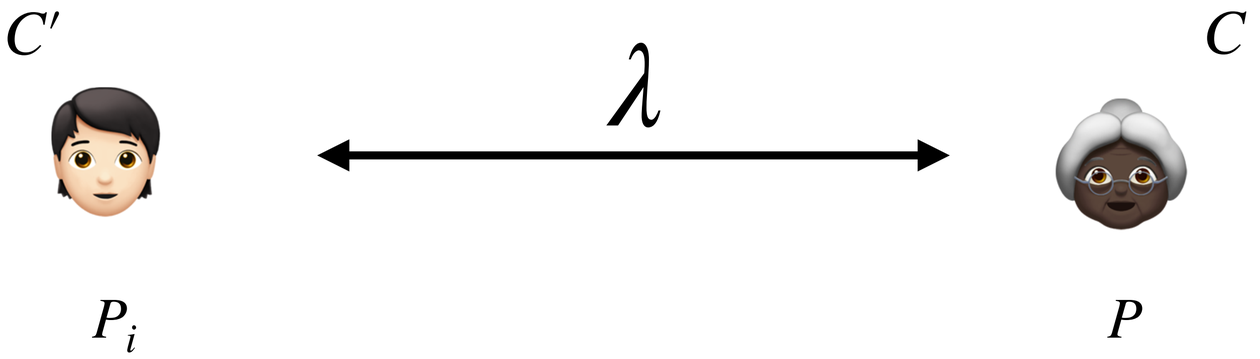
Increasing the Probability of Pulling



C is a full certificate for m

- For each P_i:
  - P samples $O(\lambda)$ signatures in C
  - P finds (via testing) $O(1)$ signatures among sample that P_i doesn't have
- P_i pulls missing index with probability $O(\lambda/n)$ (improved from naive $O(1/n)$ probability)

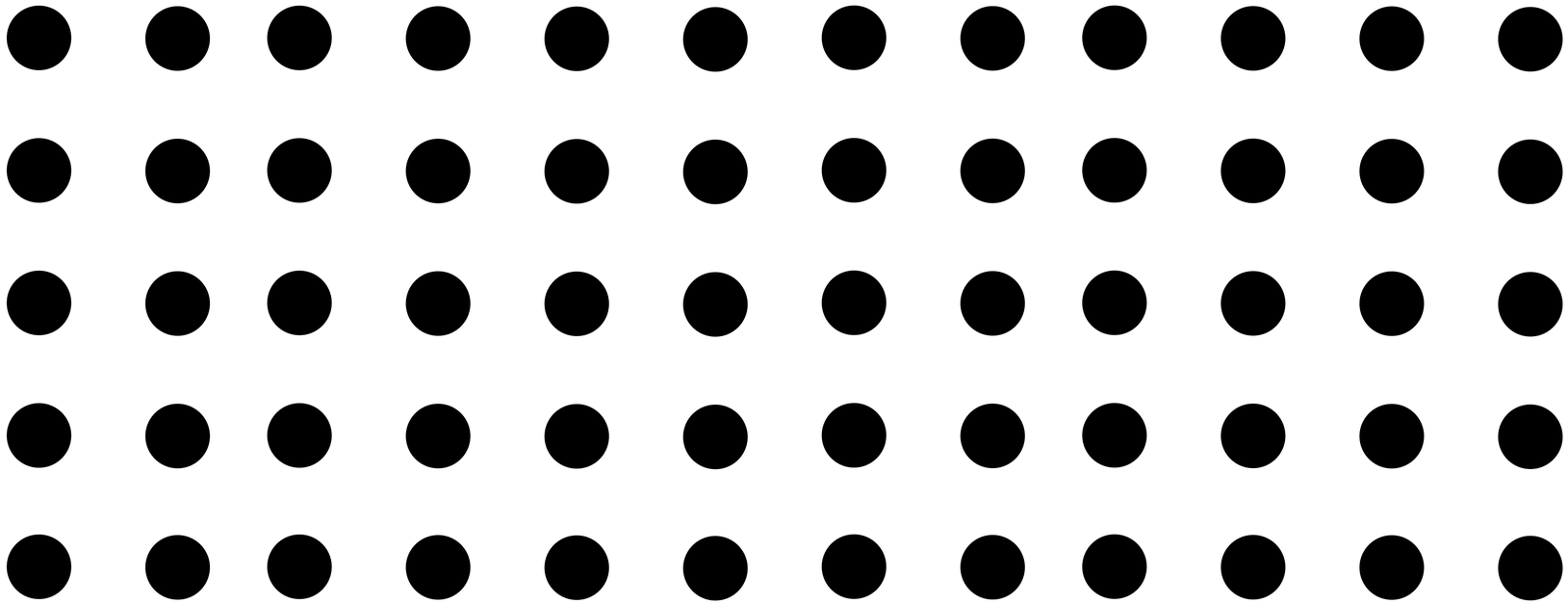
Increasing the Probability of Pulling



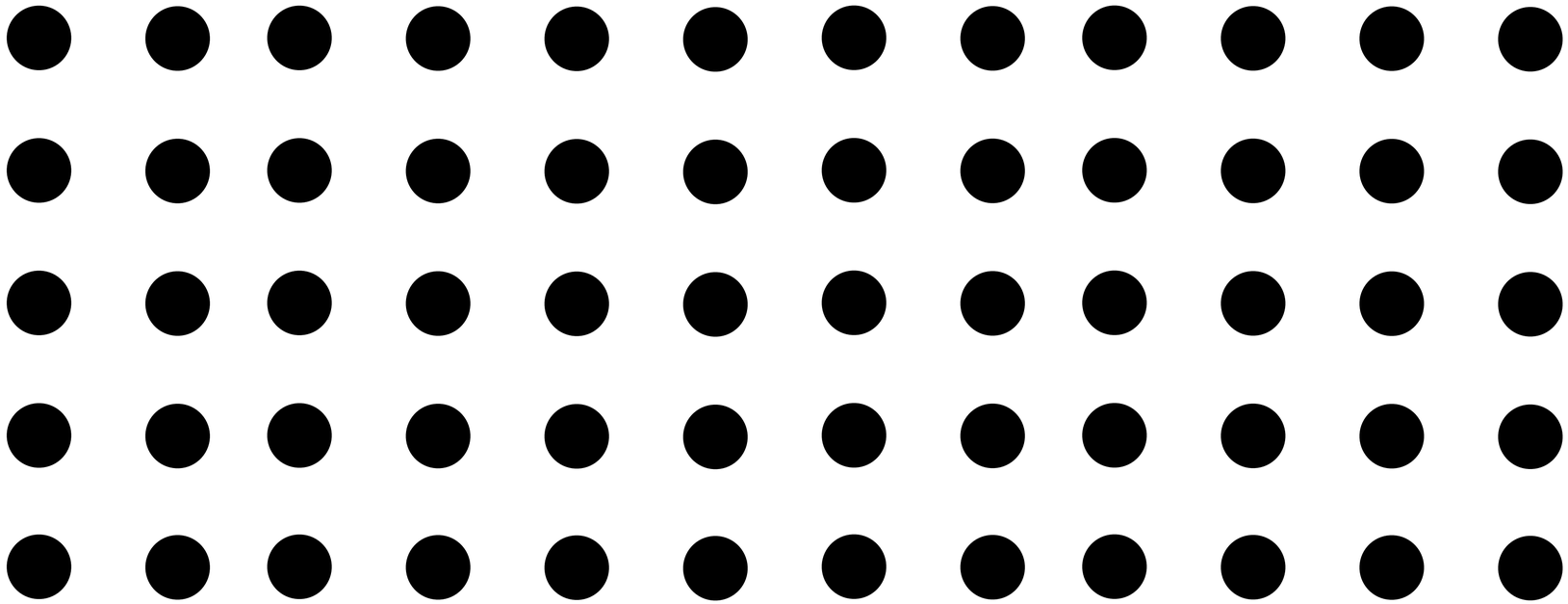
C is a full certificate for m

- For each P_i:
  - P samples $O(\lambda)$ signatures in C
  - P finds (via testing) $O(1)$ signatures among sample that P_i doesn't have
- P_i pulls missing index with probability $O(\lambda/n)$ (improved from naive $O(1/n)$ probability)
- Per round, λ honest parties learn a new signature from P with probability $1 - 2^{-\lambda}$

Alternating Push/Pull Paradigm



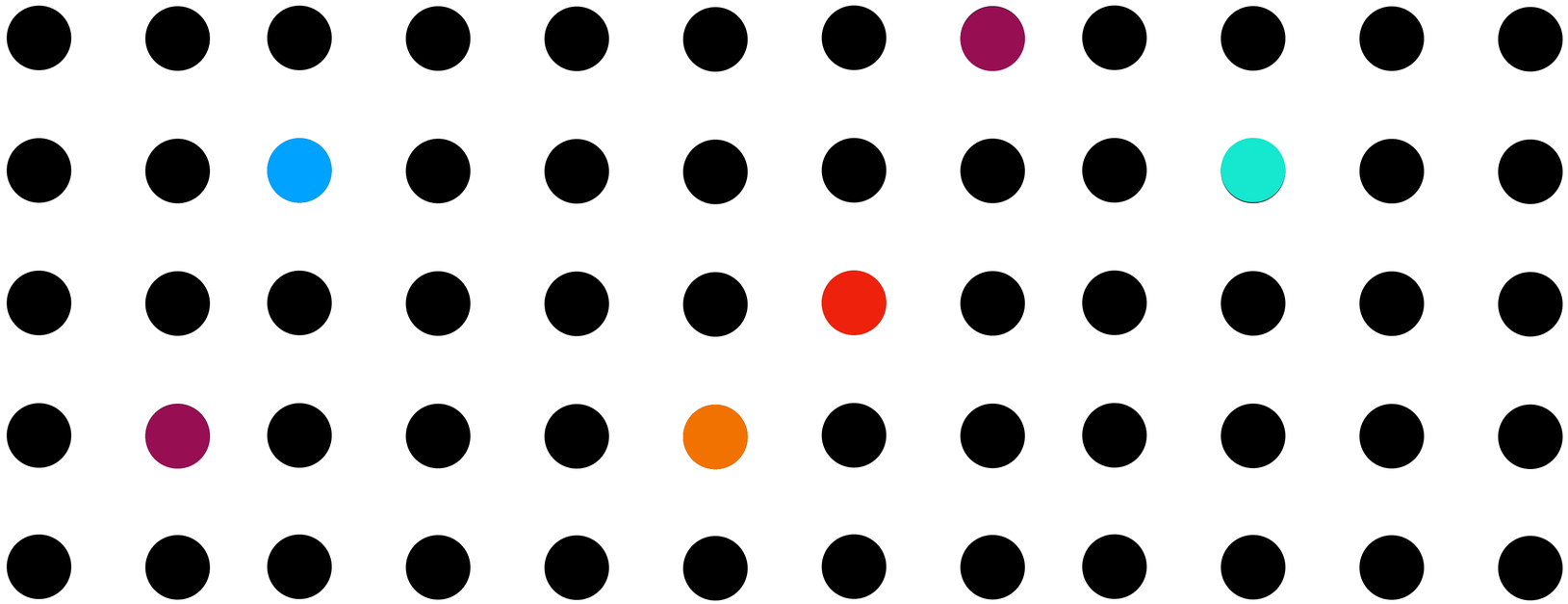
Alternating Push/Pull Paradigm



P

- Perform pulling step on certificate C

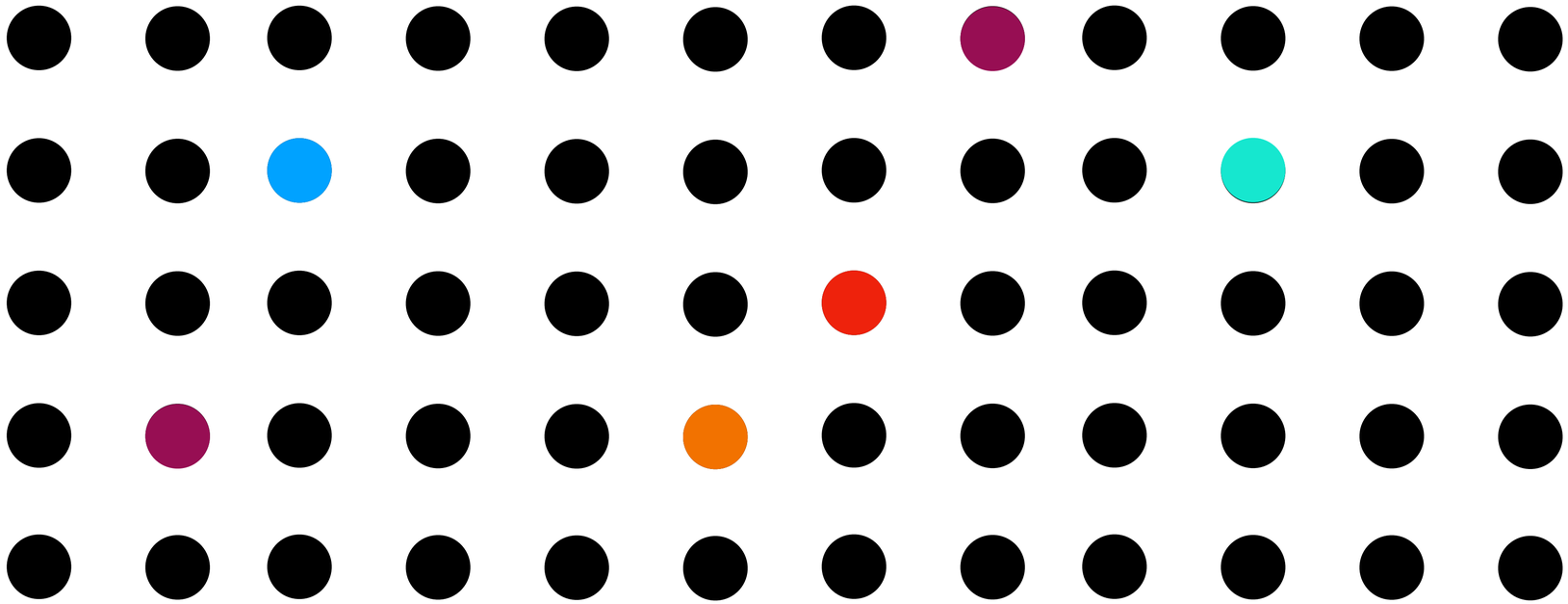
Alternating Push/Pull Paradigm



P

- Perform pulling step on certificate C

Alternating Push/Pull Paradigm

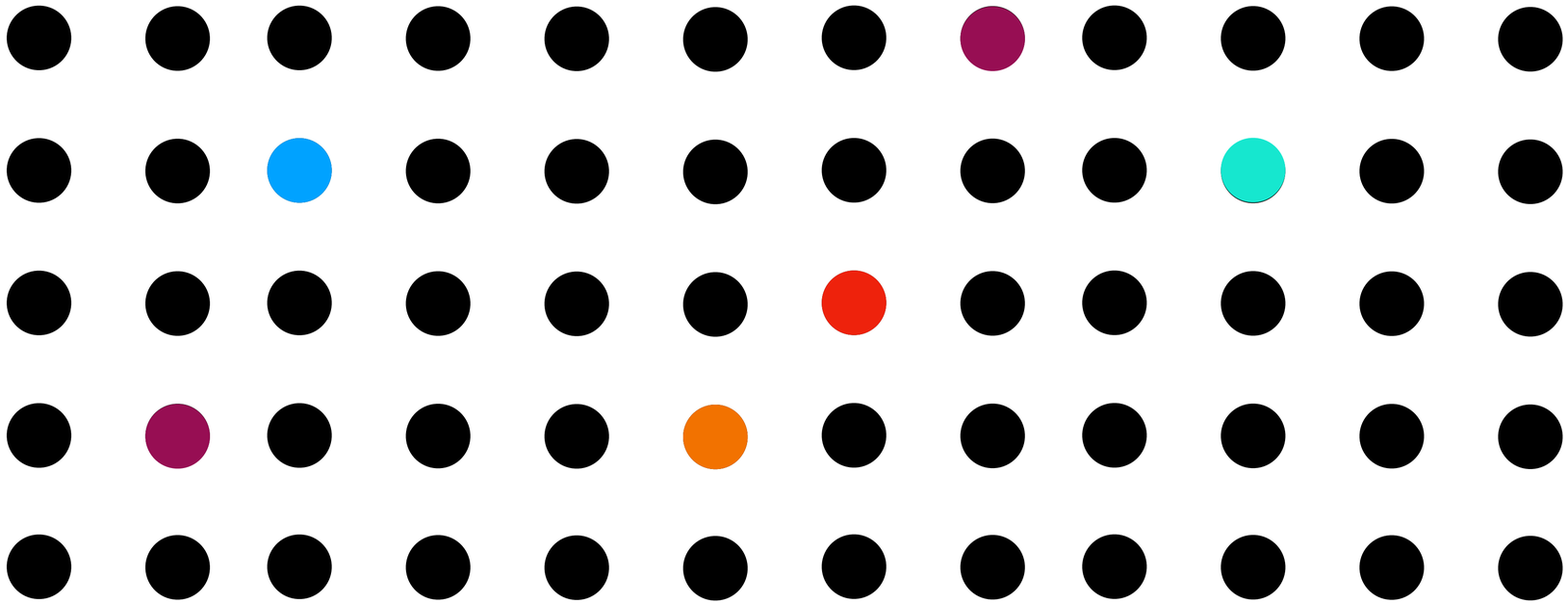


C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$

Alternating Push/Pull Paradigm

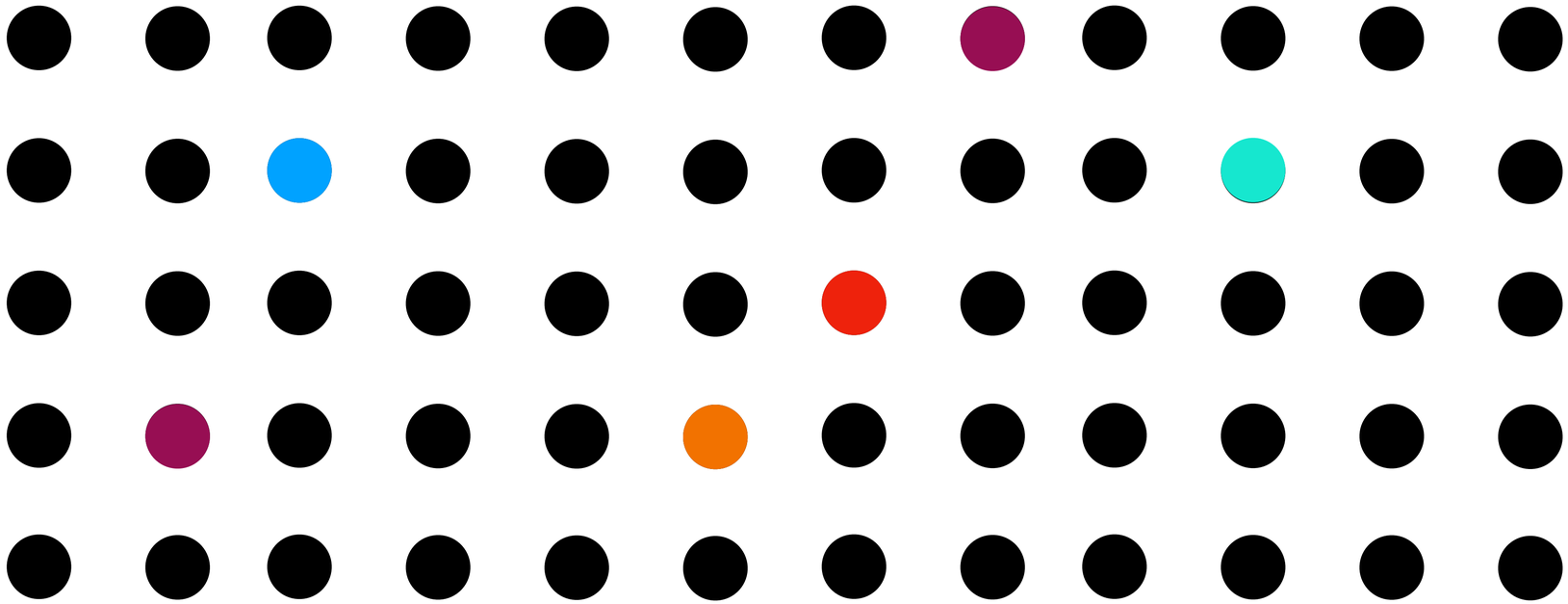


C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$
  - Later on, ensures λ parties learn some new signatures from C with probability $1 - 2^{-\lambda}$

Alternating Push/Pull Paradigm

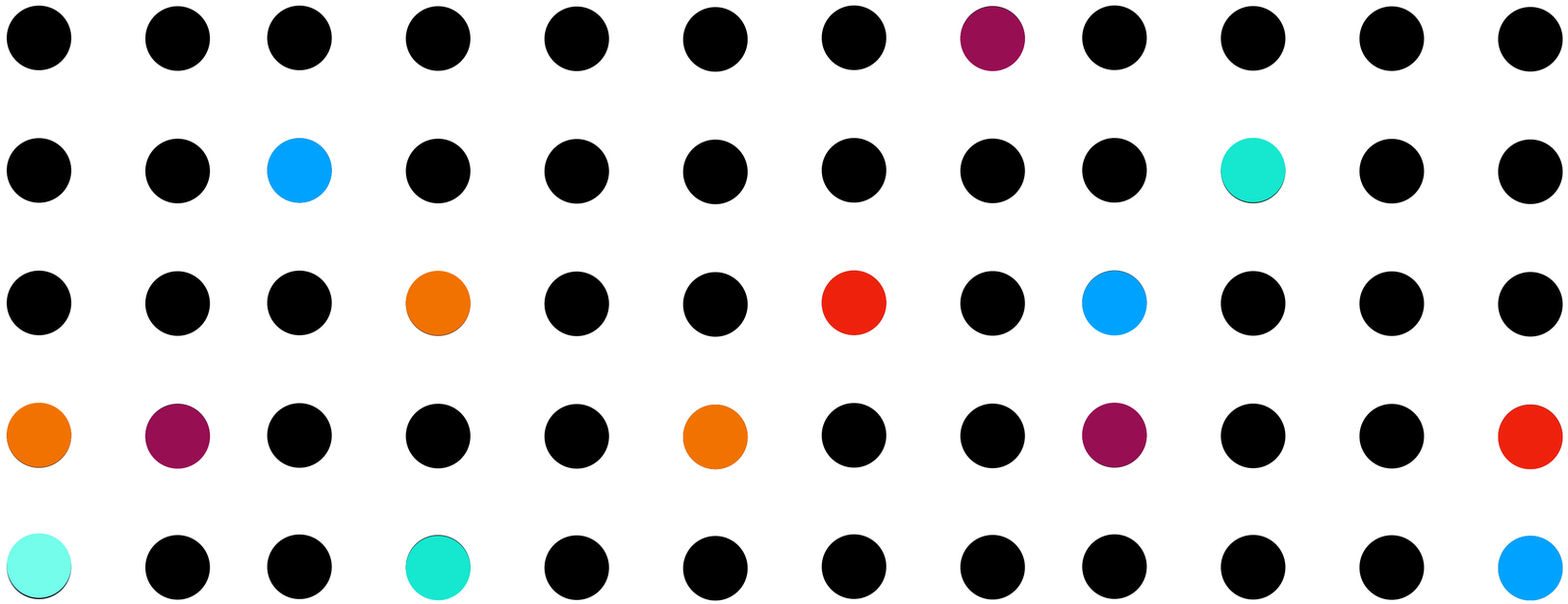


C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$
  - Later on, ensures λ parties learn some new signatures from C with probability $1 - 2^{-\lambda}$
- Push random $O(1)$ indices to all parties

Alternating Push/Pull Paradigm

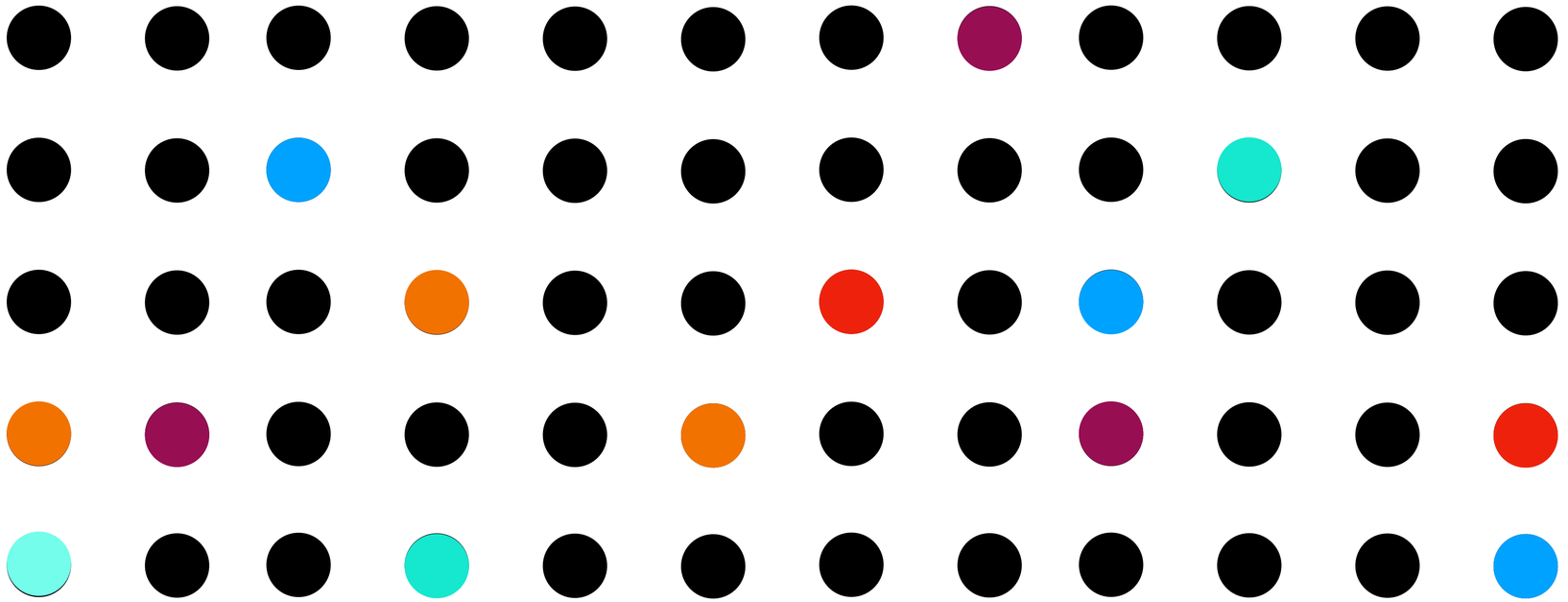


C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$
  - Later on, ensures λ parties learn some new signatures from C with probability $1 - 2^{-\lambda}$
- Push random $O(1)$ indices to all parties

Alternating Push/Pull Paradigm

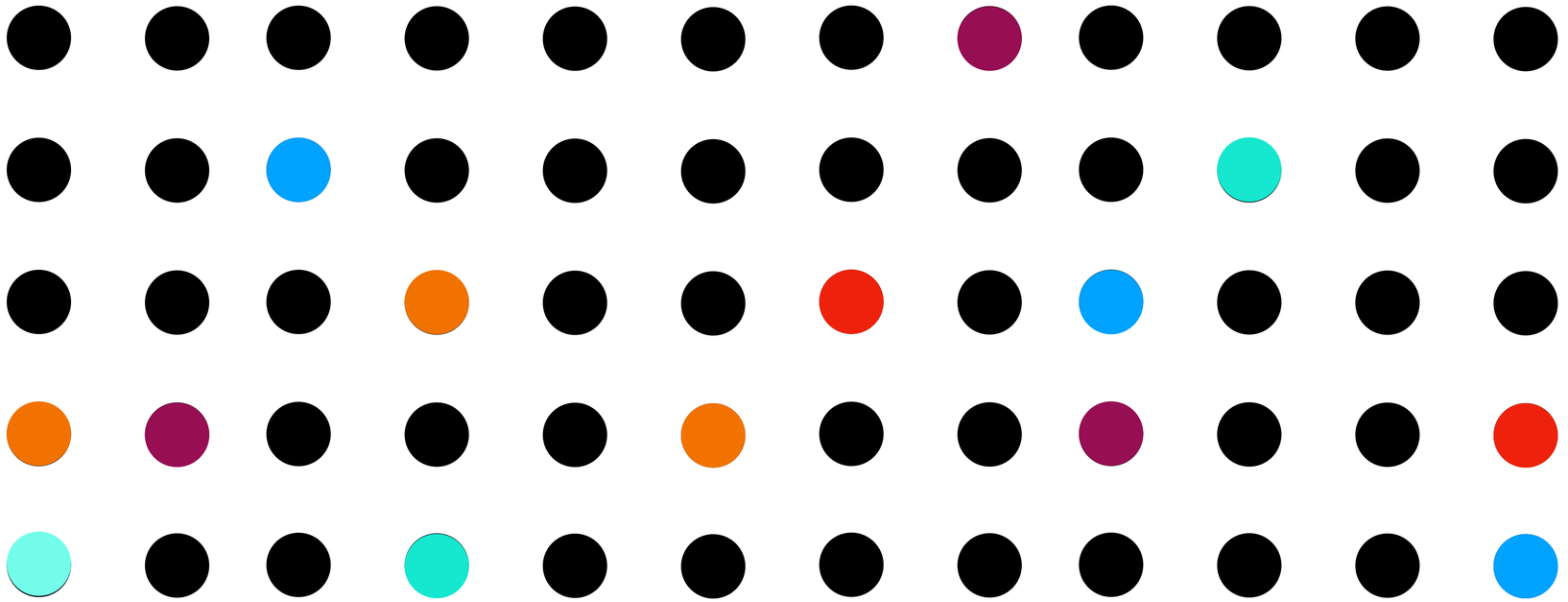


C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$
  - Later on, ensures λ parties learn some new signatures from C with probability $1 - 2^{-\lambda}$
- Push random $O(1)$ indices to all parties
  - Increases spread of signatures by constant factor with probability $1 - 2^{-\lambda}$ due to concentration over at least λ parties

Alternating Push/Pull Paradigm



Repeat $\log(n)$ times

C

P

- Perform pulling step on certificate C
  - Initially, ensures that $O(n)$ parties each learn $O(1)$ signatures from C with probability $1 - 2^{-n} \geq 1 - 2^{-\lambda}$
  - Later on, ensures λ parties learn some new signatures from C with probability $1 - 2^{-\lambda}$
- Push random $O(1)$ indices to all parties
  - Increases spread of signatures by constant factor with probability $1 - 2^{-\lambda}$ due to concentration over at least λ parties

Open Questions

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)
- Better round efficiency; can gossip run `in the background'?

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)
- Better round efficiency; can gossip run `in the background'?
- Applications to the information theoretic case with $t < n/3$

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)
- Better round efficiency; can gossip run `in the background'?
- Applications to the information theoretic case with $t < n/3$
  - Improvements to VSS?

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)
- Better round efficiency; can gossip run `in the background'?
- Applications to the information theoretic case with $t < n/3$
  - Improvements to VSS?
- Pulling-based gossip in the asynchronous model

Open Questions

- Apply pull-based gossip for the $t < n$ regime (in progress)
- Better round efficiency; can gossip run `in the background'?
- Applications to the information theoretic case with $t < n/3$
  - Improvements to VSS?
- Pulling-based gossip in the asynchronous model
    - Asynchronous common subset in $O(n^2 \cdot \ell)$ in the DY model?