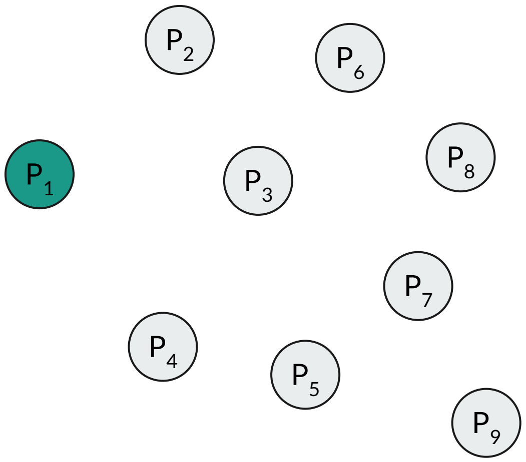
# Asymptotically Optimal Message Dissemination with Applications to Blockchains

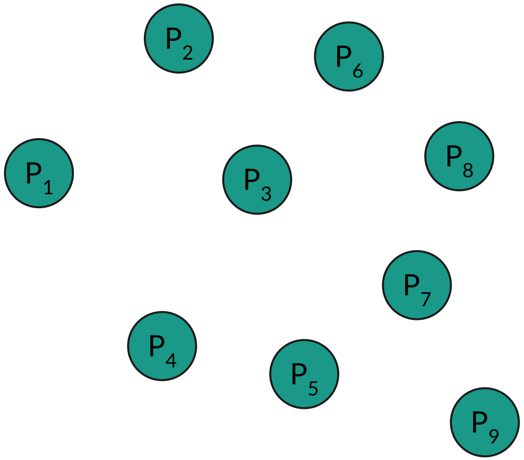
Chen-Da Liu-Zhang, Lucerne University of Applied Sciences and Arts & Web3 Foundation
Christian Matt, Primev
*Søren Eller Thomsen,* Partisia

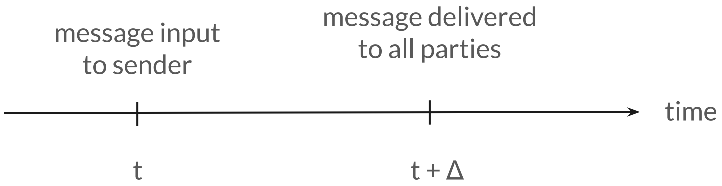
# The flooding functionality

# The flooding functionality

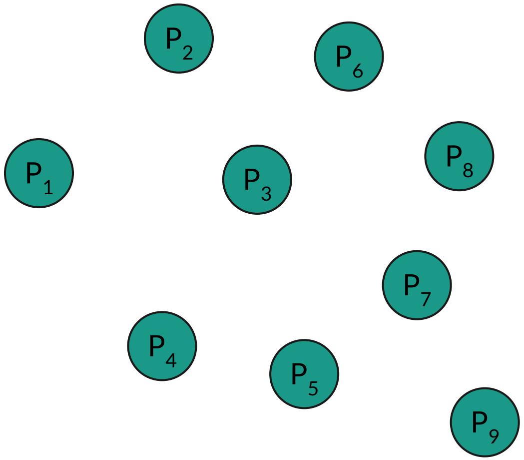



# The flooding functionality

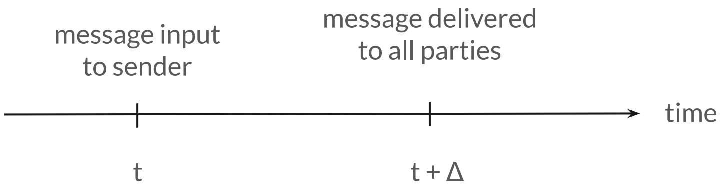


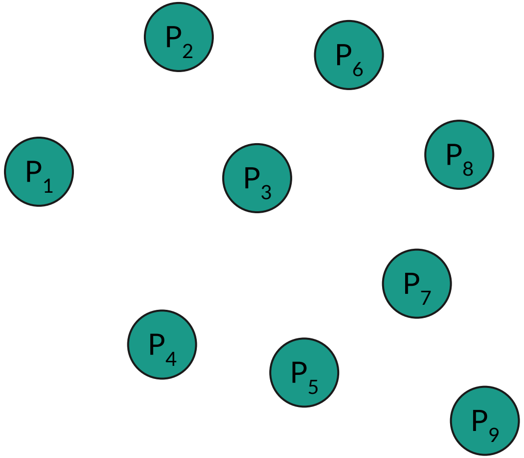
# The flooding functionality



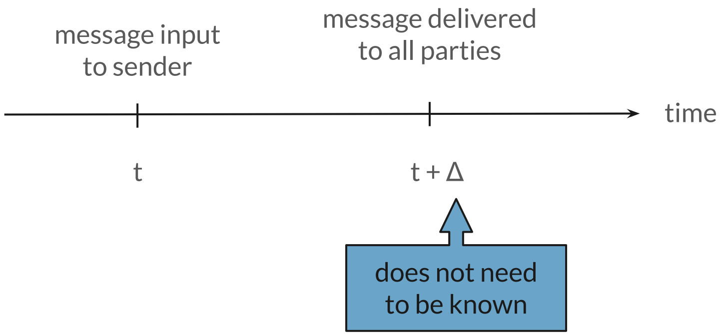

Probability that not all parties have received the message by time $t + \Delta$ must be negligible in security parameter



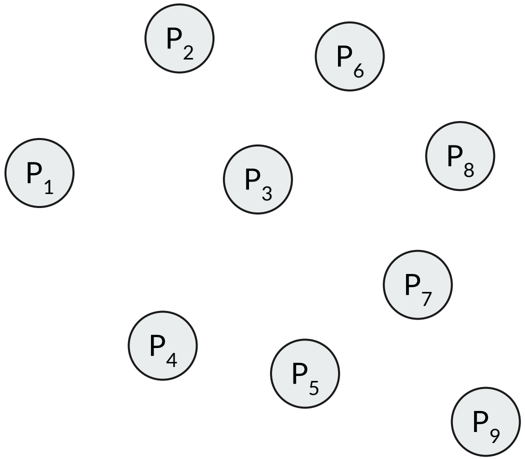
# The flooding functionality



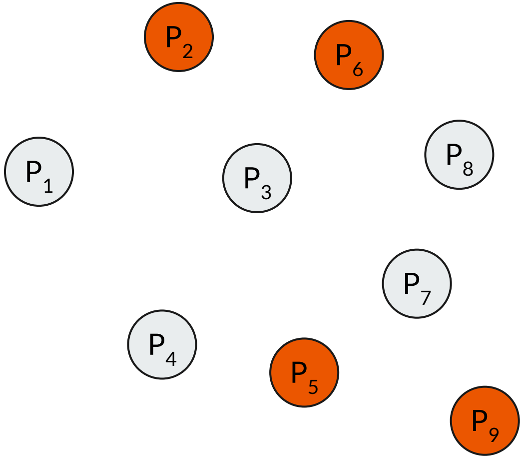

Probability that not all parties have received the message by time $t + \Delta$ must be negligible in security parameter

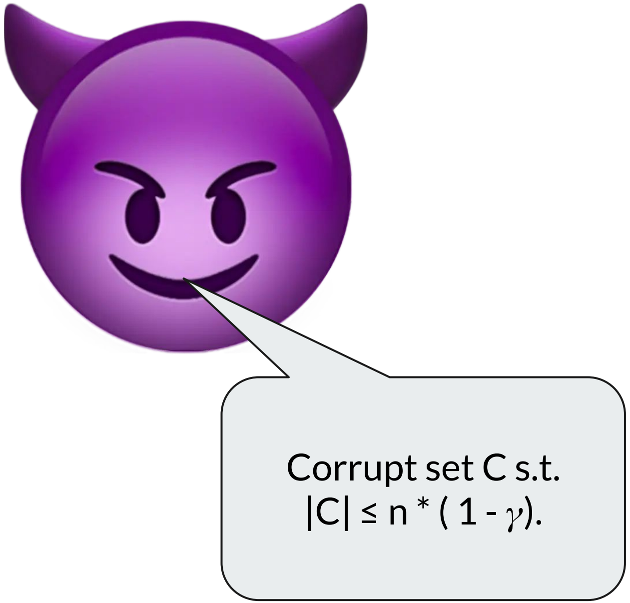


# Adversary

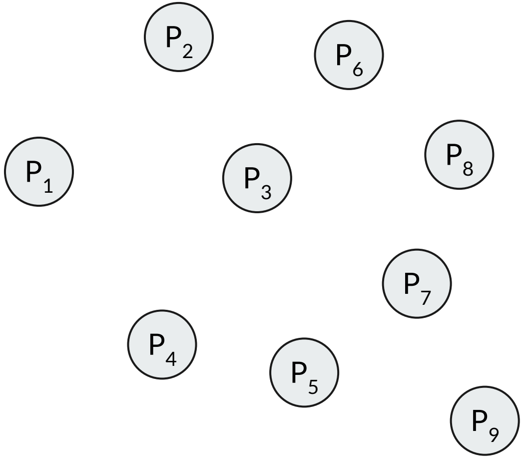



# Adversary



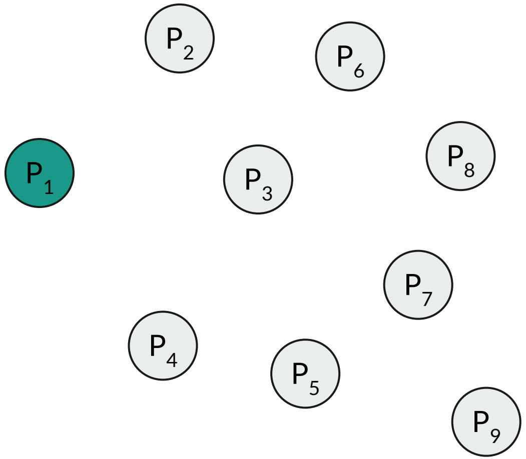
# Flooding protocols




**State of the art**

Each party forwards the message to a random subset of parties of (expected) size $\kappa$.

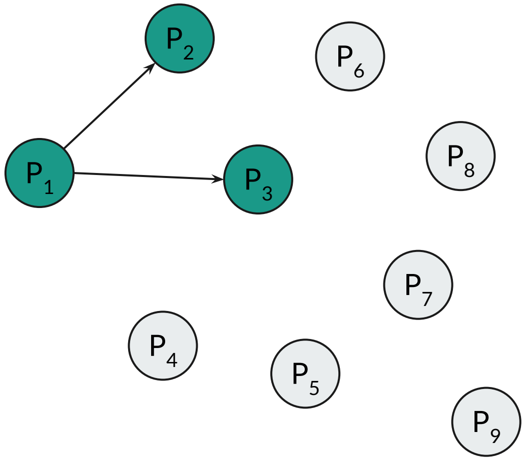
# Flooding protocols




**State of the art**

Each party forwards the message to a random subset of parties of (expected) size $\kappa$.

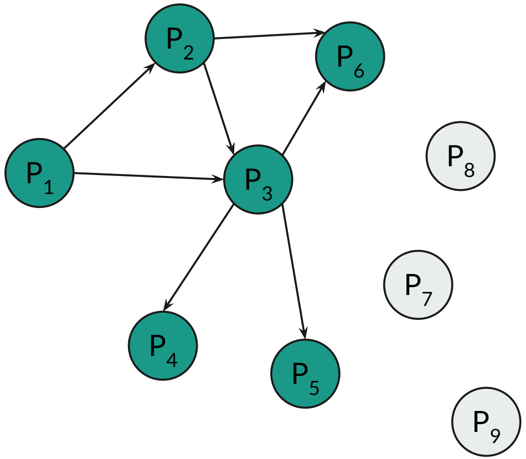
# Flooding protocols




**State of the art**

Each party forwards the message to a random subset of parties of (expected) size $\kappa$.

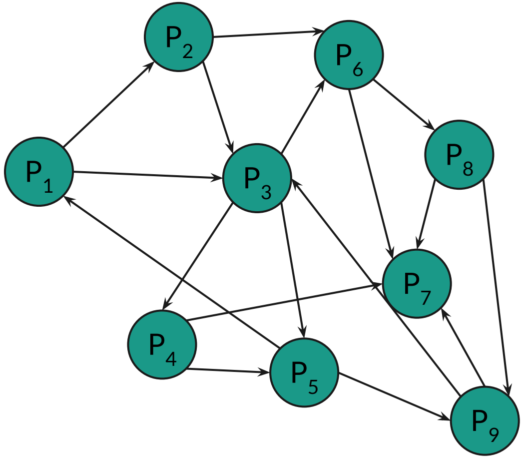
# Flooding protocols




**State of the art**

Each party forwards the message to a random subset of parties of (expected) size $\kappa$.

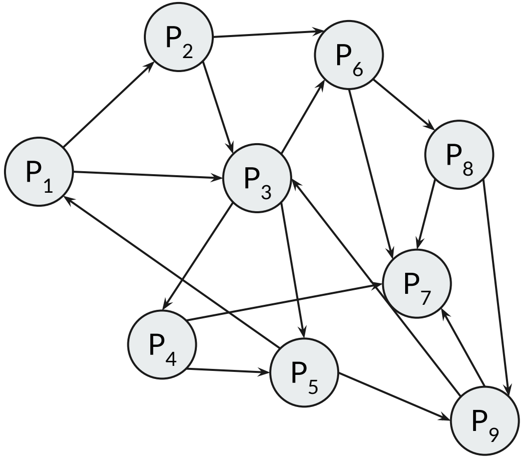
# Flooding protocols




**State of the art**

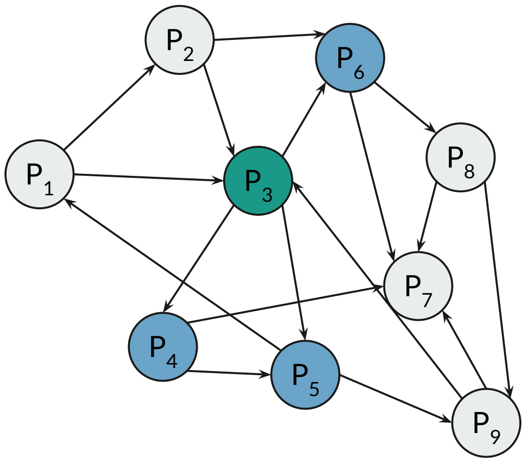
Each party forwards the message to a random subset of parties of (expected) size $\kappa$.

# KPIs for a flooding protocol



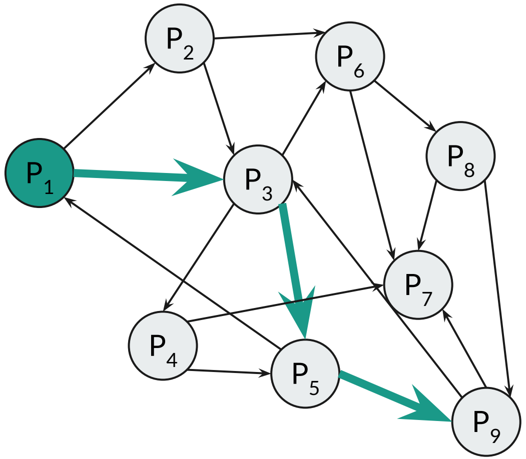

1. Max number of neighbors
2. Diameter
3. Max per party communication

# KPIs for a flooding protocol



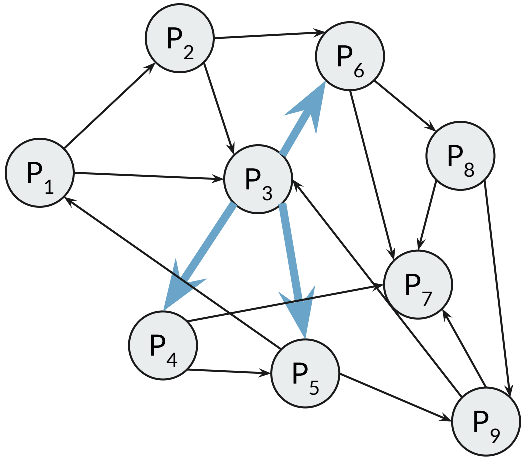

1. **Max number of neighbors**
2. Diameter
3. Max per party communication

# KPIs for a flooding protocol



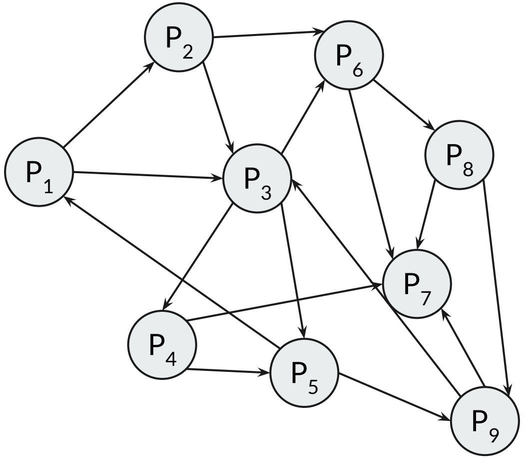

1. Max number of neighbors
2. **Diameter**
3. Max per party communication

# KPIs for a flooding protocol



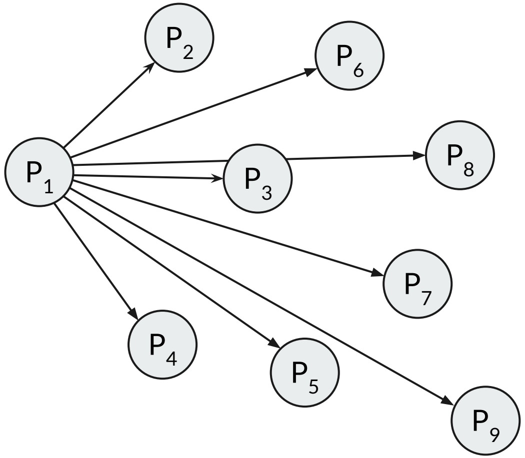

1. Max number of neighbors
2. Diameter
3. **Max per-party communication**

# State of the art flooding (in presence of 😈)

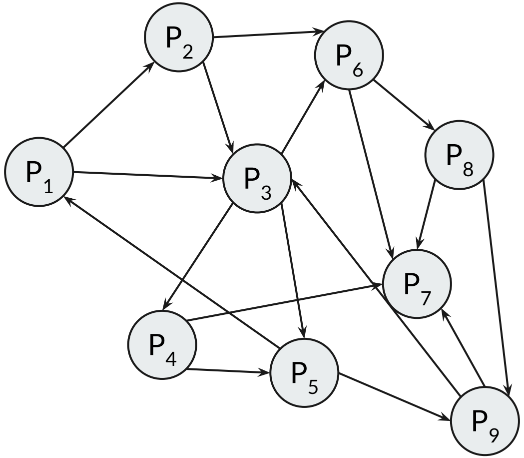



Each party forwards the message to a random subset of parties of (expected) size $\kappa$. [MNT22, LMMRT22]

# State of the art flooding (in presence of 😈)

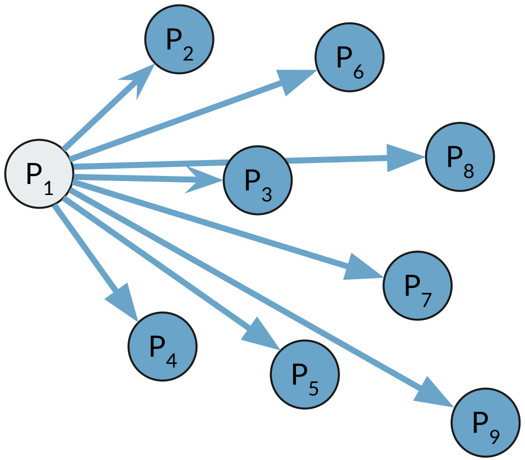



Sender forwards the message to all parties.

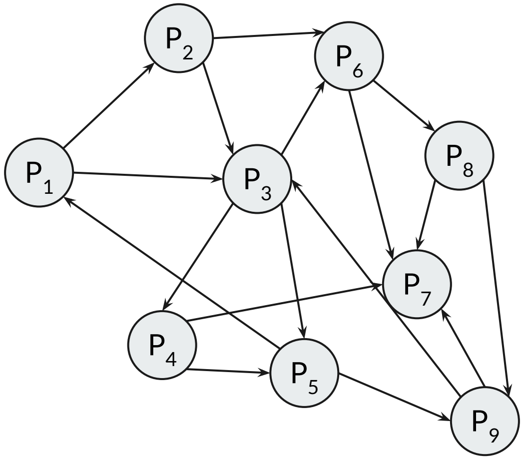



Each party forwards the message to a random subset of parties of (expected) size $\kappa$. [MNT22, LMMRT22]

## State of the art flooding (in presence of 😈)




Sender forwards the message to all parties.




Each party forwards the message to a random subset of parties of (expected) size $\kappa$. [MNT22, LMMRT22]

# Our results: Two asymptotically optimal protocols

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |

*n* = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
*l* = length of message.

# Our results: Two asymptotically optimal protocols

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |
| **ECFlood** | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(log(n))$ |

$n$ = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
$l$ = length of message.

# Our results: Two asymptotically optimal protocols

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |
| **ECFlood** | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(log(n))$ |
| Naive | $n - 1$ | $l \cdot (n - 1)$ | $1$ |

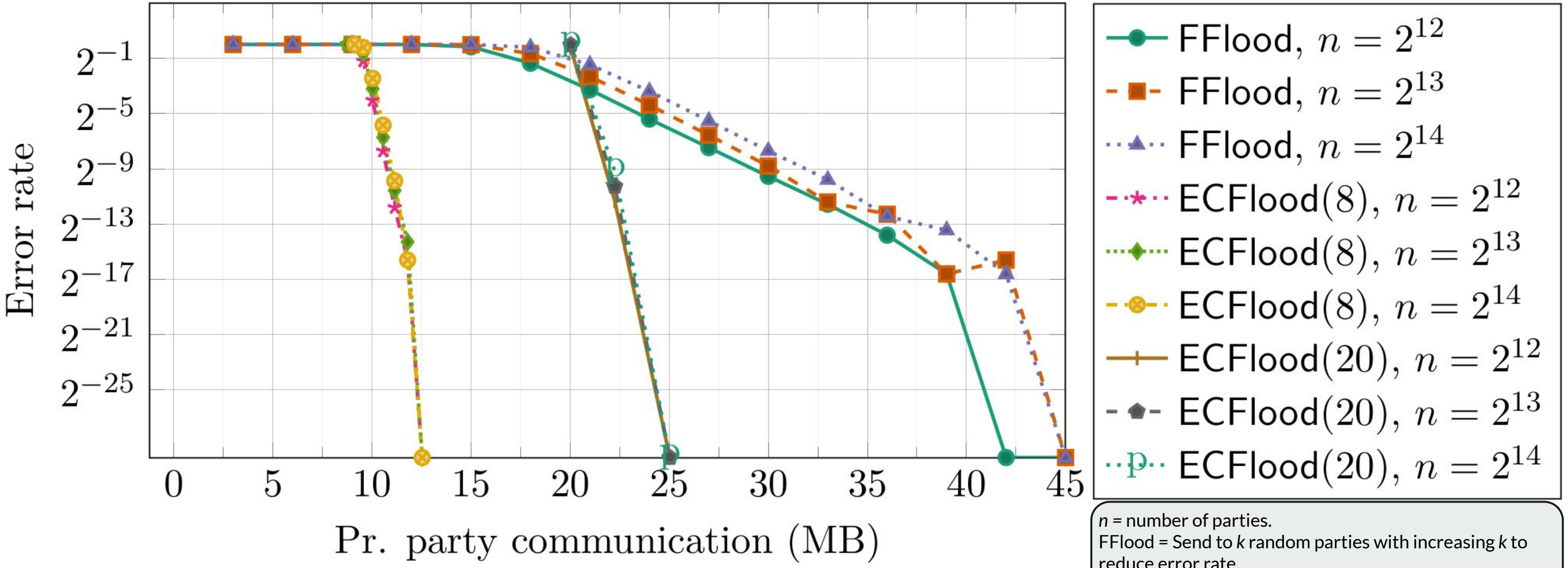
$n$ = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
$l$ = length of message.

# Our results: Two asymptotically optimal protocols

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |
| **ECFlood** | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(log(n))$ |
| Naive | $n - 1$ | $l \cdot (n - 1)$ | $1$ |
| **ECCast** | $n - 1$ | $O(l \cdot \gamma^{-1})$ | $2$ |

$n$ = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
$l$ = length of message.

# Efficiency evaluation of ECFlood




*n* = number of parties.
FFlood = Send to *k* random parties with increasing *k* to reduce error rate.
ECFlood (x) = Our new protocol with parameter x.

# Our asymptotically optimal protocol

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |
| **ECFlood** | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(log(n))$ |

$n$ = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
$l$ = length of message.

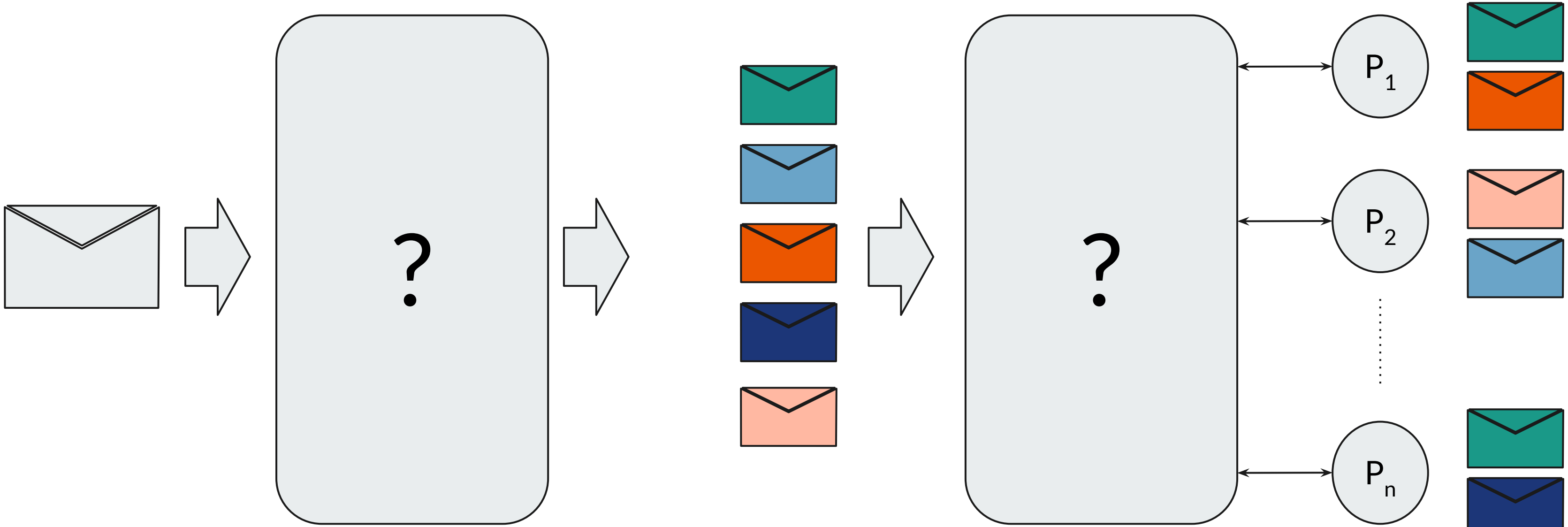
# Our asymptot tocol

Necessary for Erdős–Rényi flooding protocols shown in [KMG03] and necessary for fanout type flooding shown in [LMMRT22]

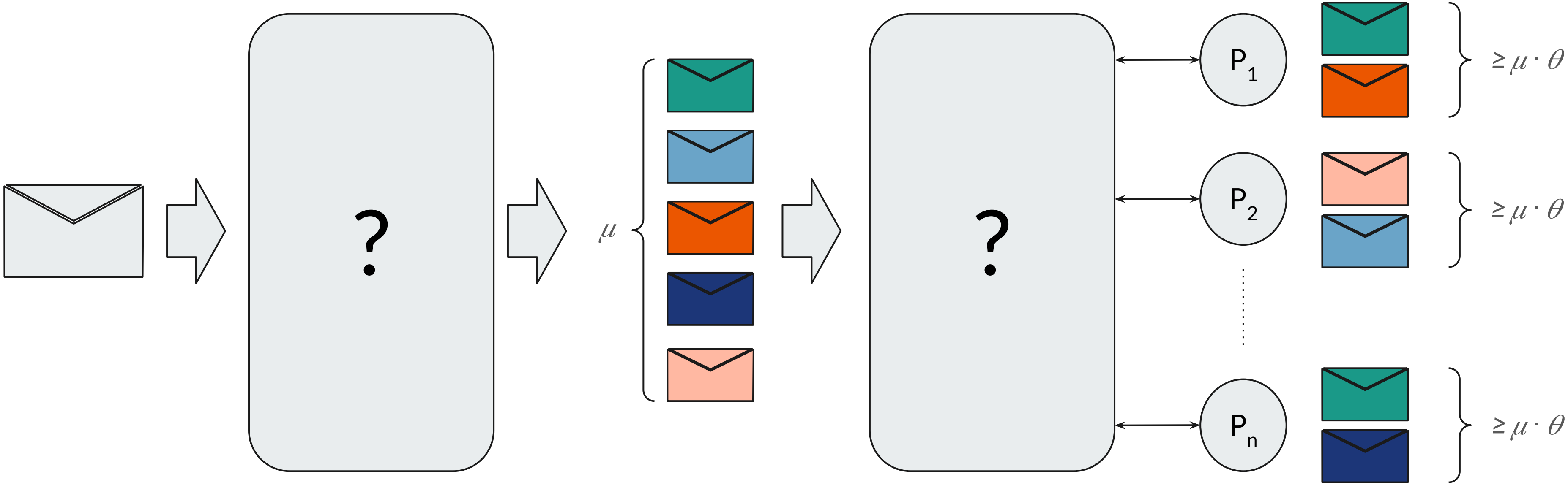
| Protocol | | ax per-party communication | Diameter |
|---|---|---|---|
| [MNT22, LMMRT22] | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1} \cdot (log(n) + \kappa))$ | $O(log(n))$ |
| **ECFlood** | $O(\gamma^{-1} \cdot (log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(log(n))$ |

$n$ = number of parties.
$\kappa$ = security parameter.
$\gamma$ = minimum fraction of honest parties.
$l$ = length of message.

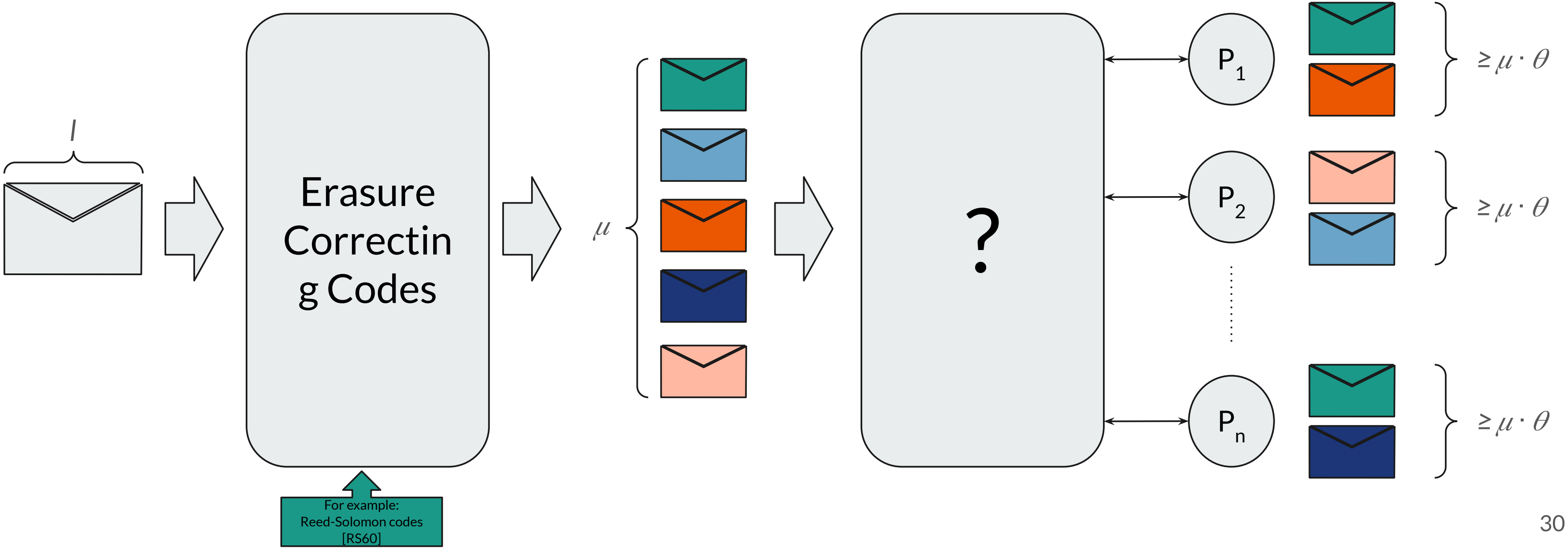
# Meet ECFlood



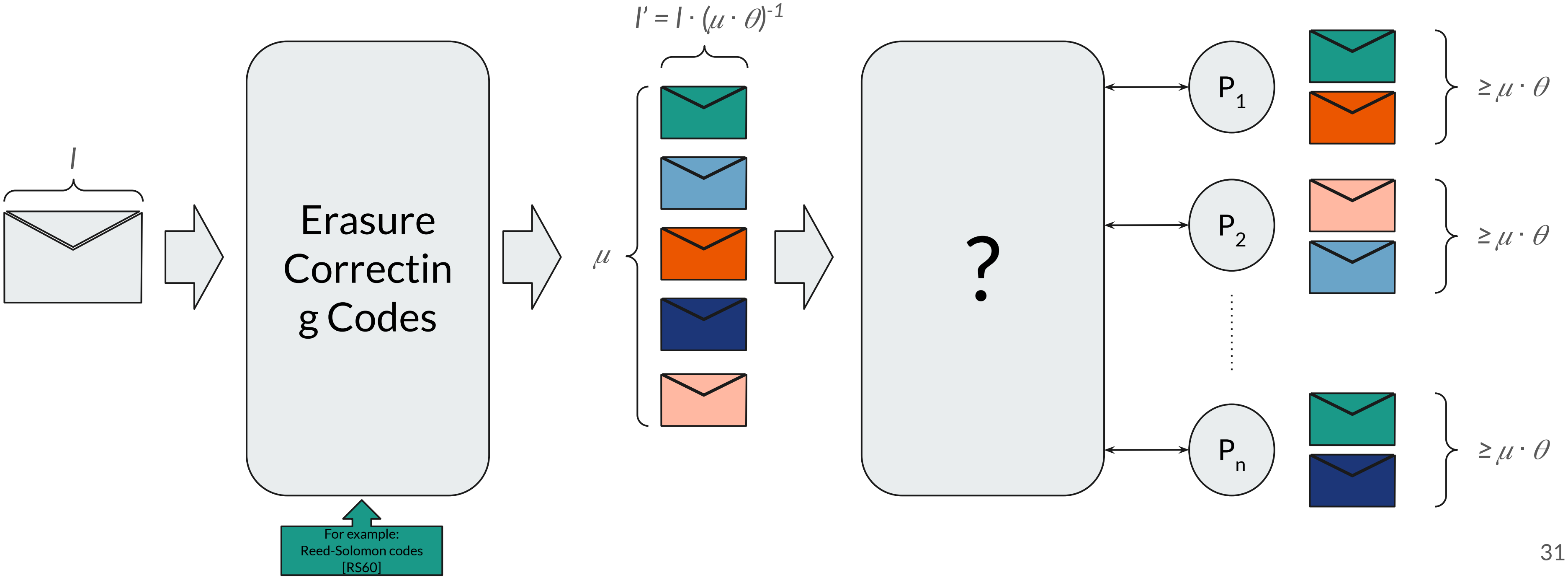
# Meet ECFlood



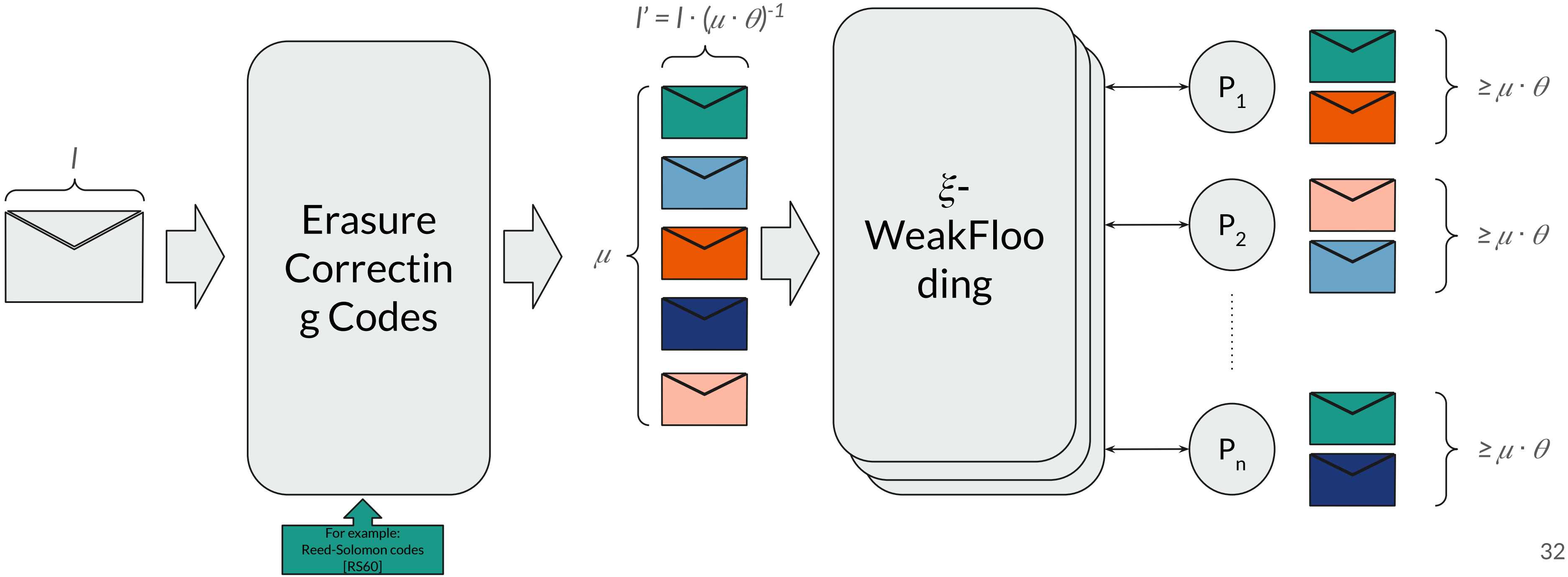
# Meet ECFlood



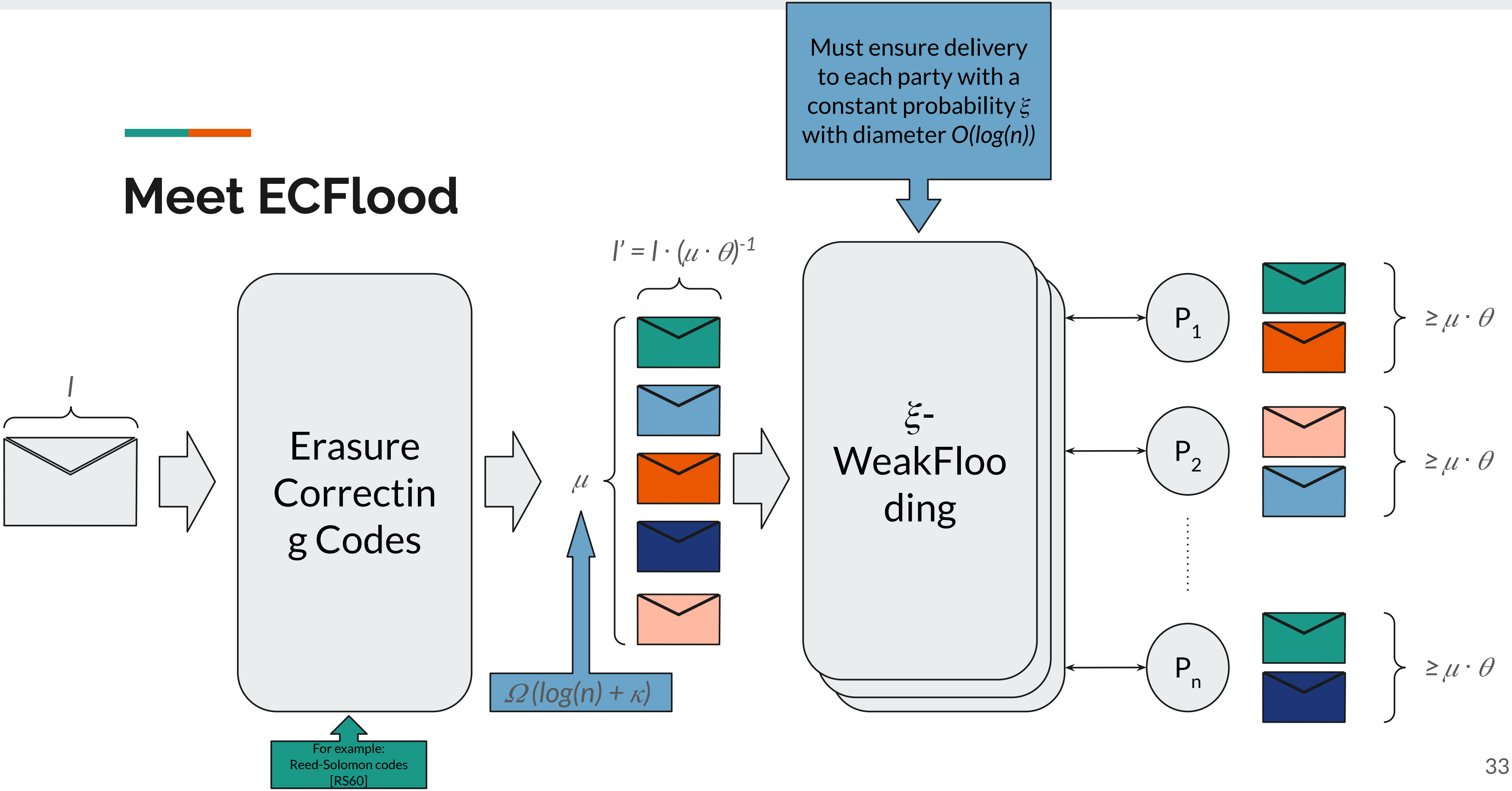
# Meet ECFlood



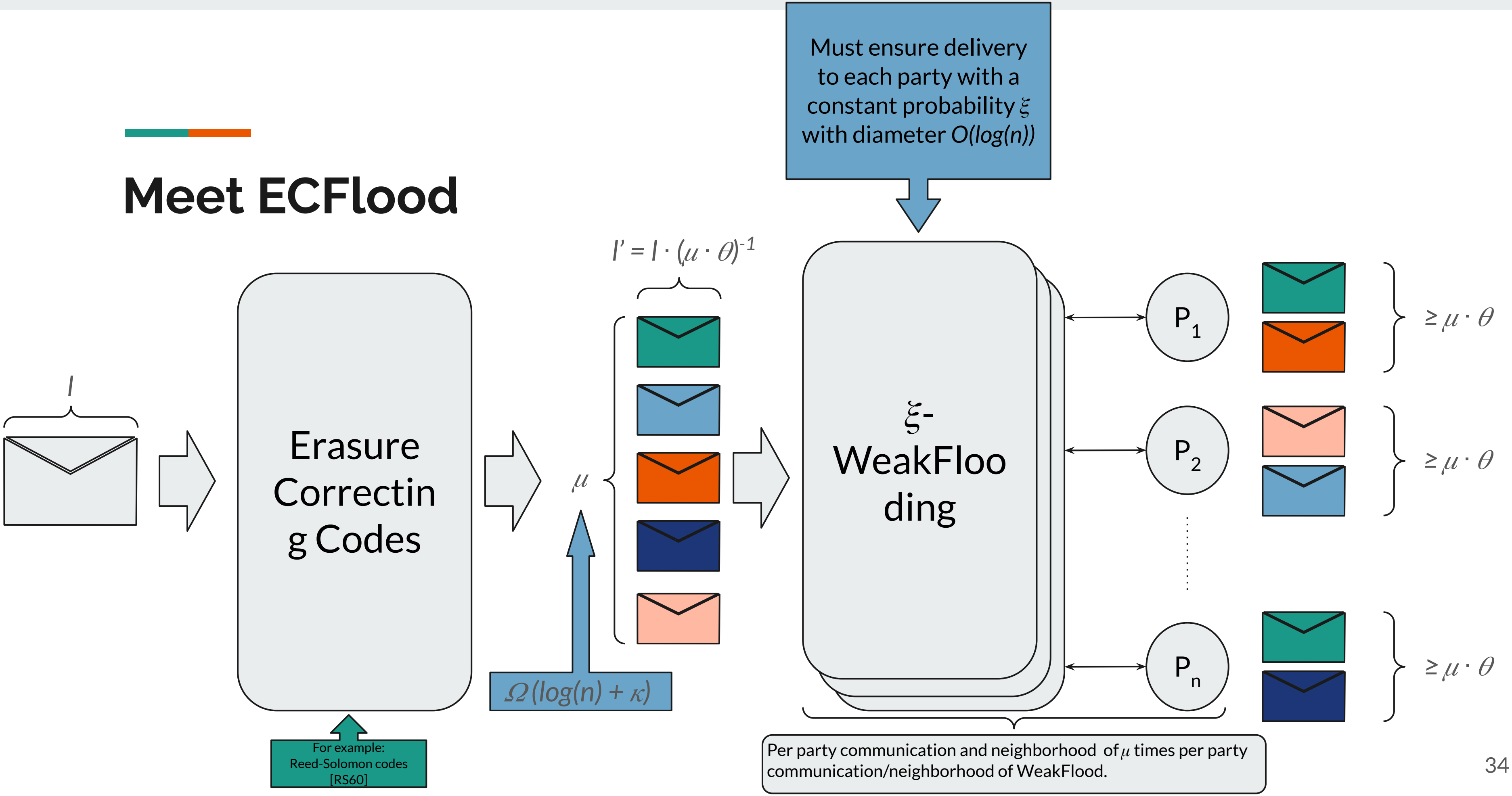
# Meet ECFlood



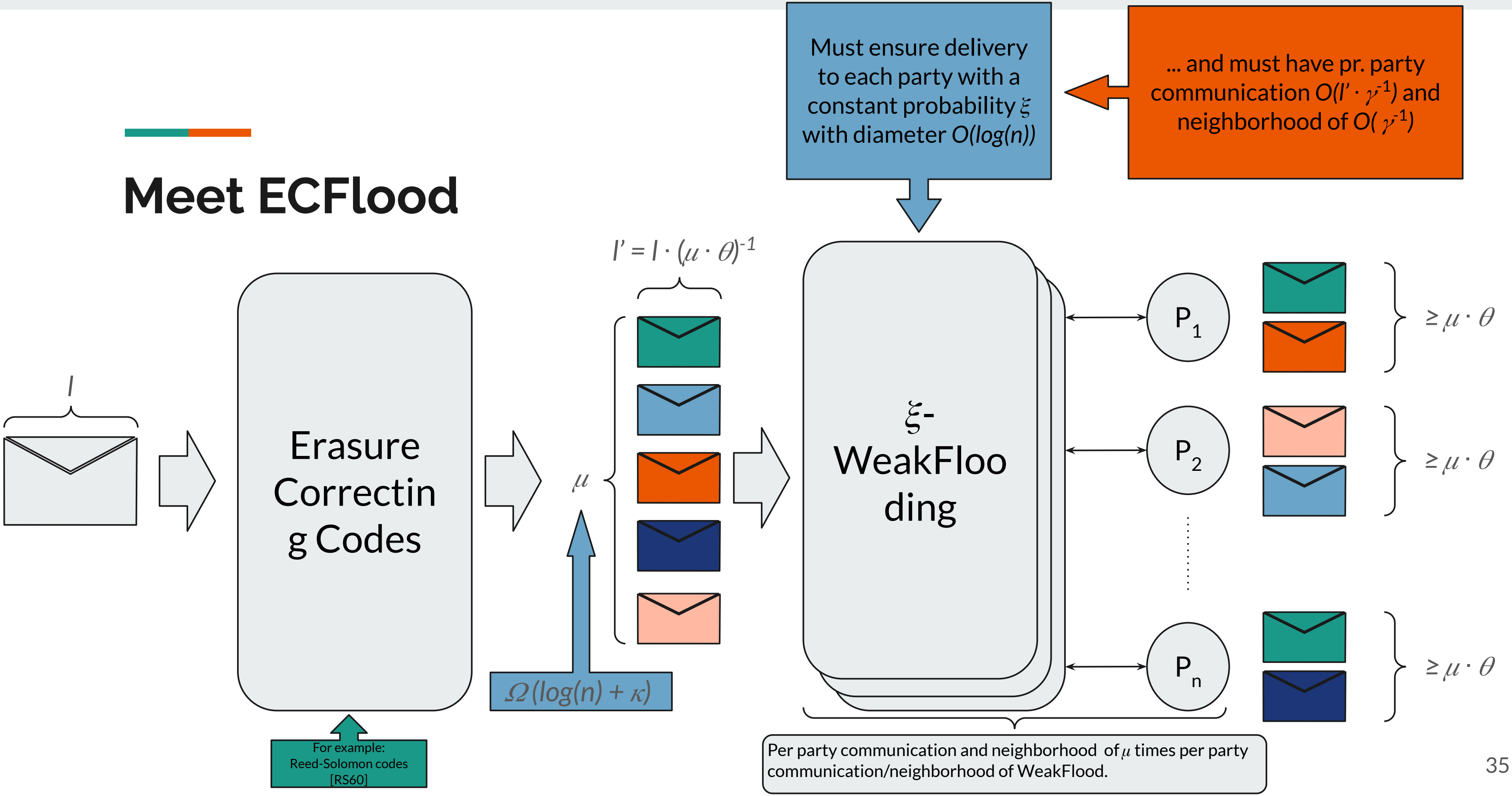
# Meet ECFlood



# Meet ECFlood



# Meet ECFlood

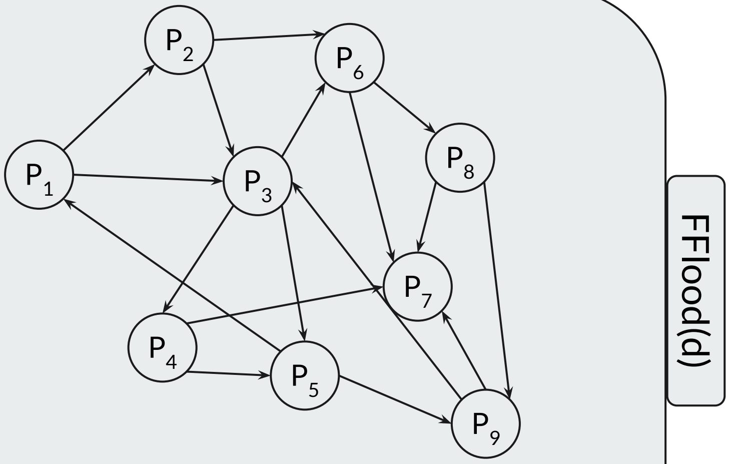


# Requirements for a WeakFlooding protocol

1. Must ensure delivery with diameter $O(\log(n))$ to each party with constant probability
2. Must have $O(\gamma^{-1})$ neighborhoods
3. Must have per party communication of $O(l' \cdot \gamma^{-1})$ for messages of length $l'$.

# So… Any candidates for a $\xi$-WeakFlooding protocol?

# So… Any candidates for a $\xi$-WeakFlooding protocol?

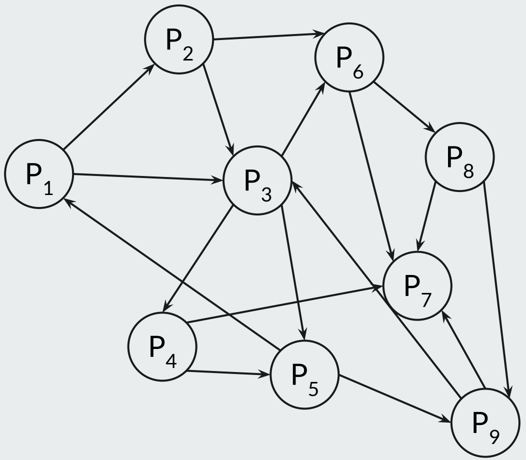



Each party forwards the message to a random subset of parties of size d.

# So… Any candidates for a $\xi$-WeakFlooding protocol?

For $d = O(\gamma^{-1})$, FFlood($d$):

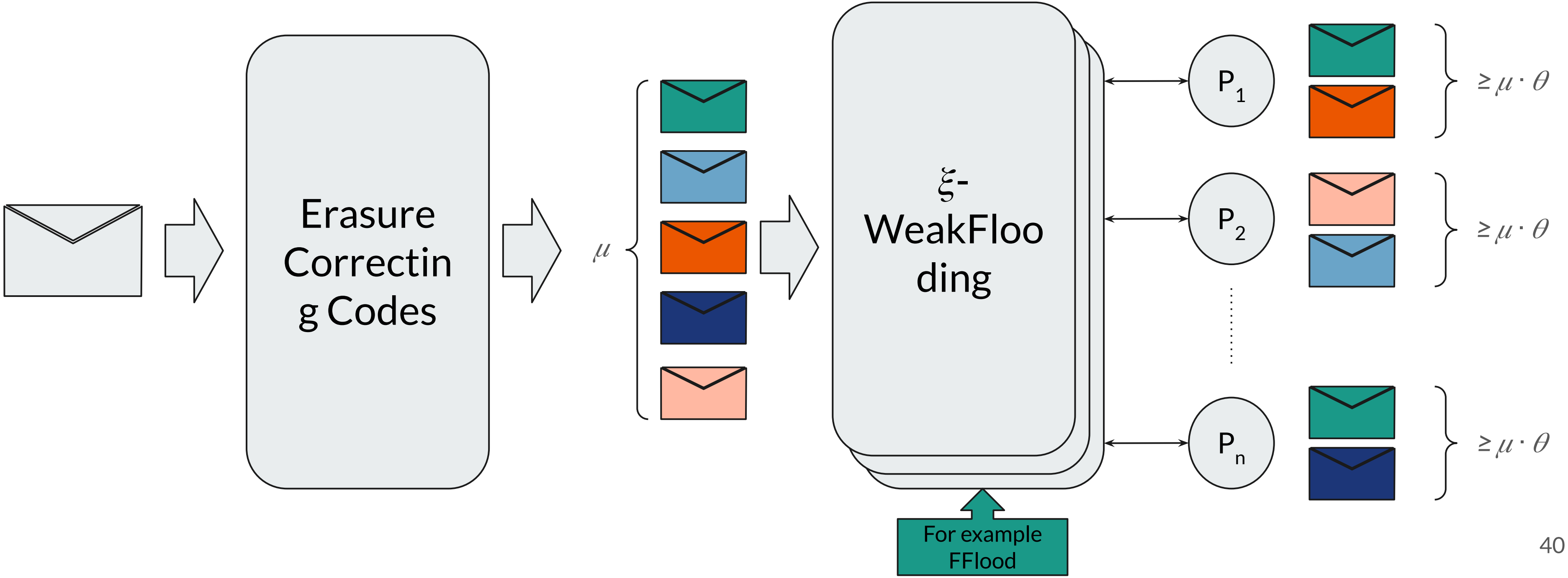
1. Ensures delivery with diameter $O(log(n))$ to each party with constant probability ✅
2. Has $d$ neighborhoods ✅
3. Has per party communication of $O(l' \cdot d)$ for messages of length $l'$ ✅



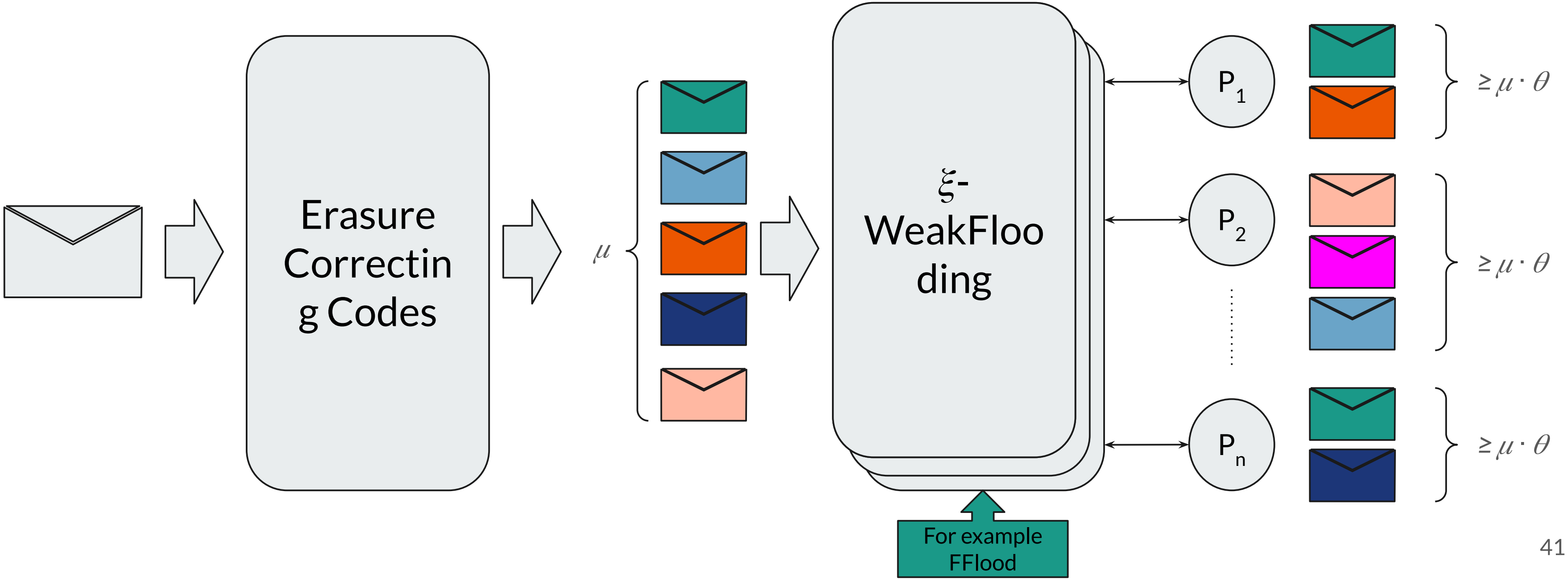

Each party forwards the message to a random subset of parties of size d.

FFlood(d)

# ECFlood

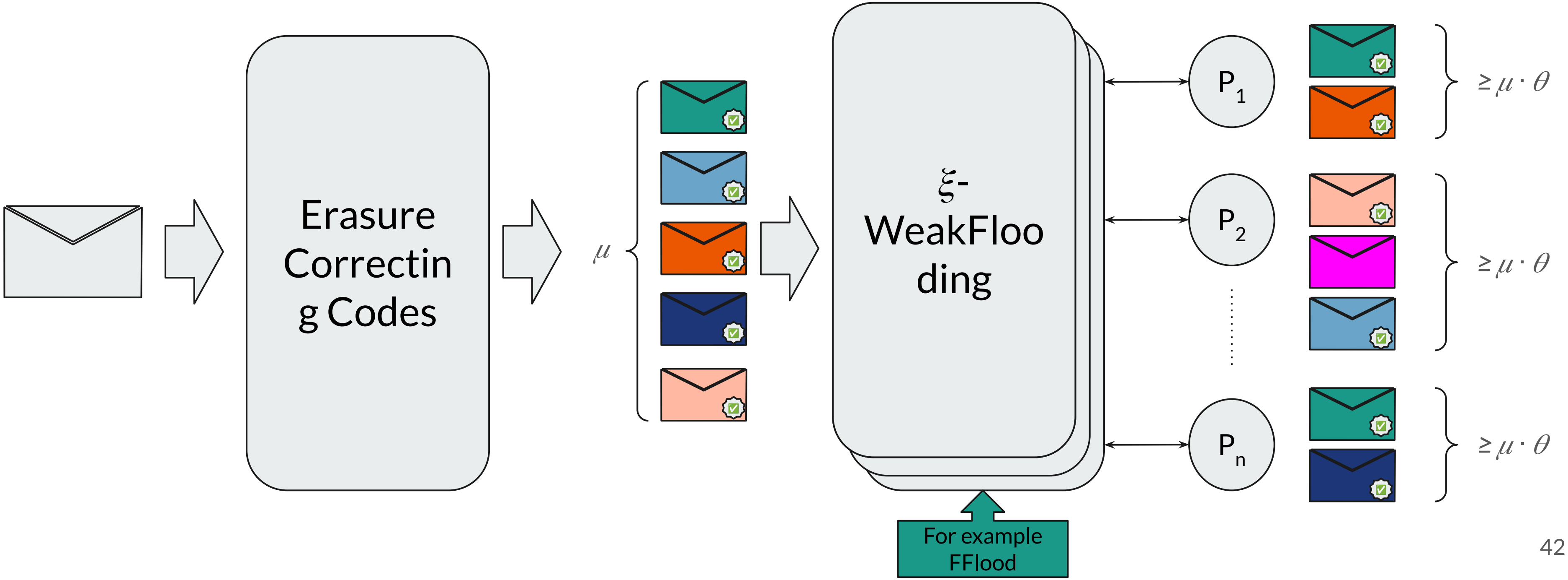

Erasure Correctin g Codes
$\mu$
$\xi$-WeakFlooding
For example FFlood
$P_1$
$P_2$
$P_n$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$

# ECFlood



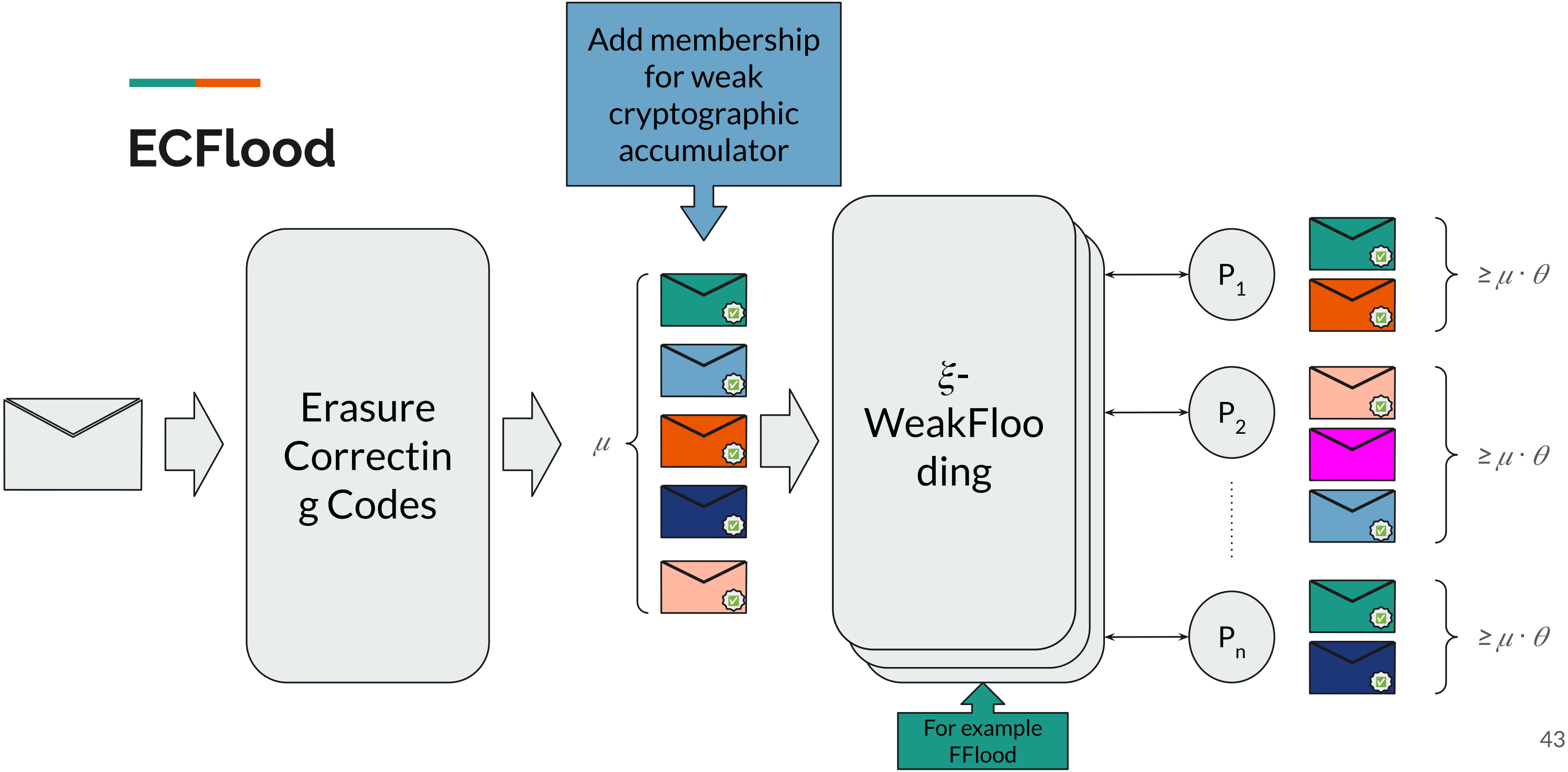
Erasure Correcting Codes
$\mu$
$\xi$-WeakFlooding
For example FFlood
$P_1$
$P_2$
$P_n$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$

# ECFlood

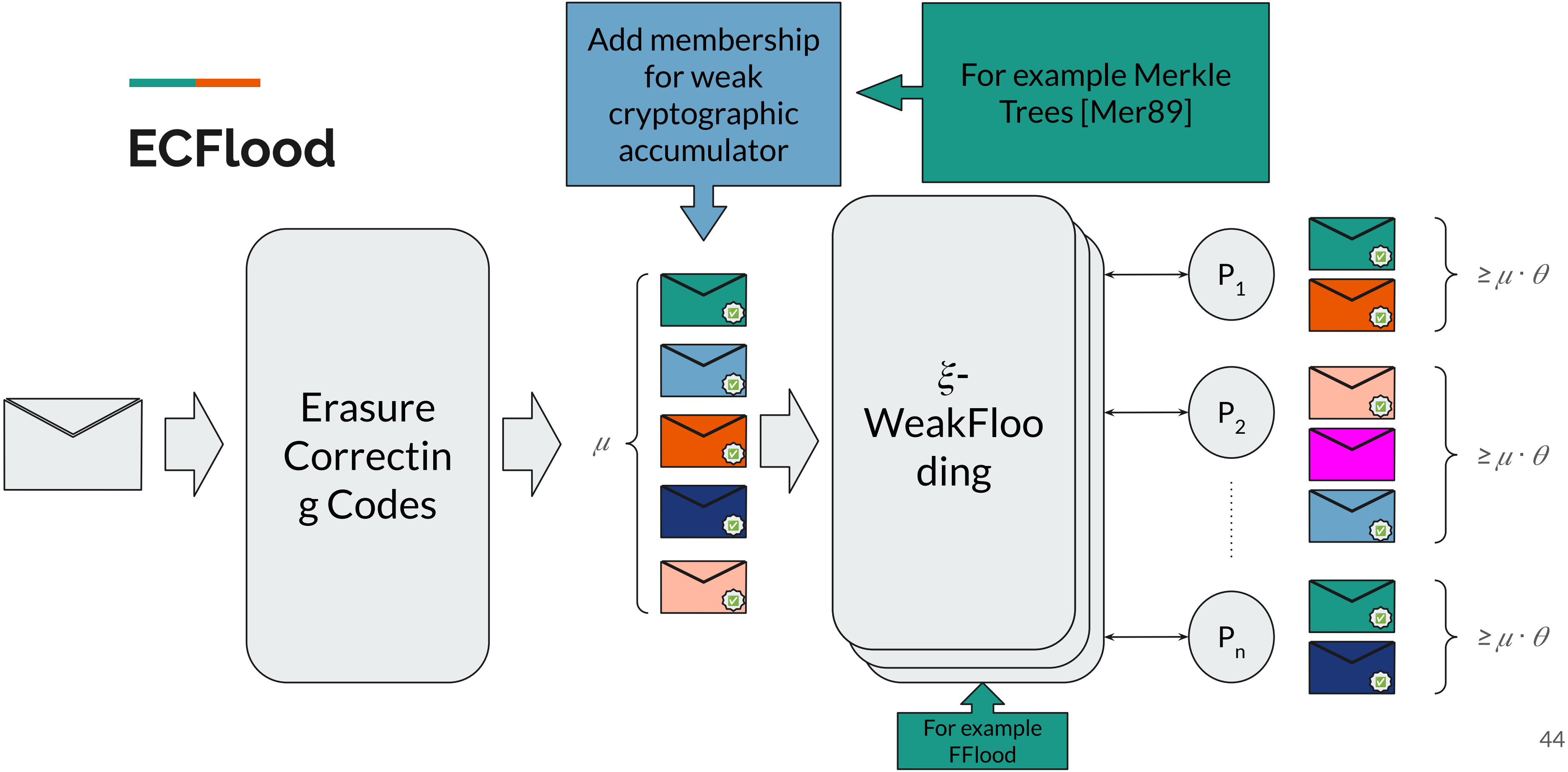


Erasure Correcting Codes
$\mu$
$\xi$-WeakFlooding
For example FFlood
$P_1$
$P_2$
$P_n$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$

# ECFlood


Add membership for weak cryptographic accumulator
Erasure Correcting Codes
$\mu$
$\xi$-WeakFlooding
For example FFlood
$P_1$
$P_2$
$P_n$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$

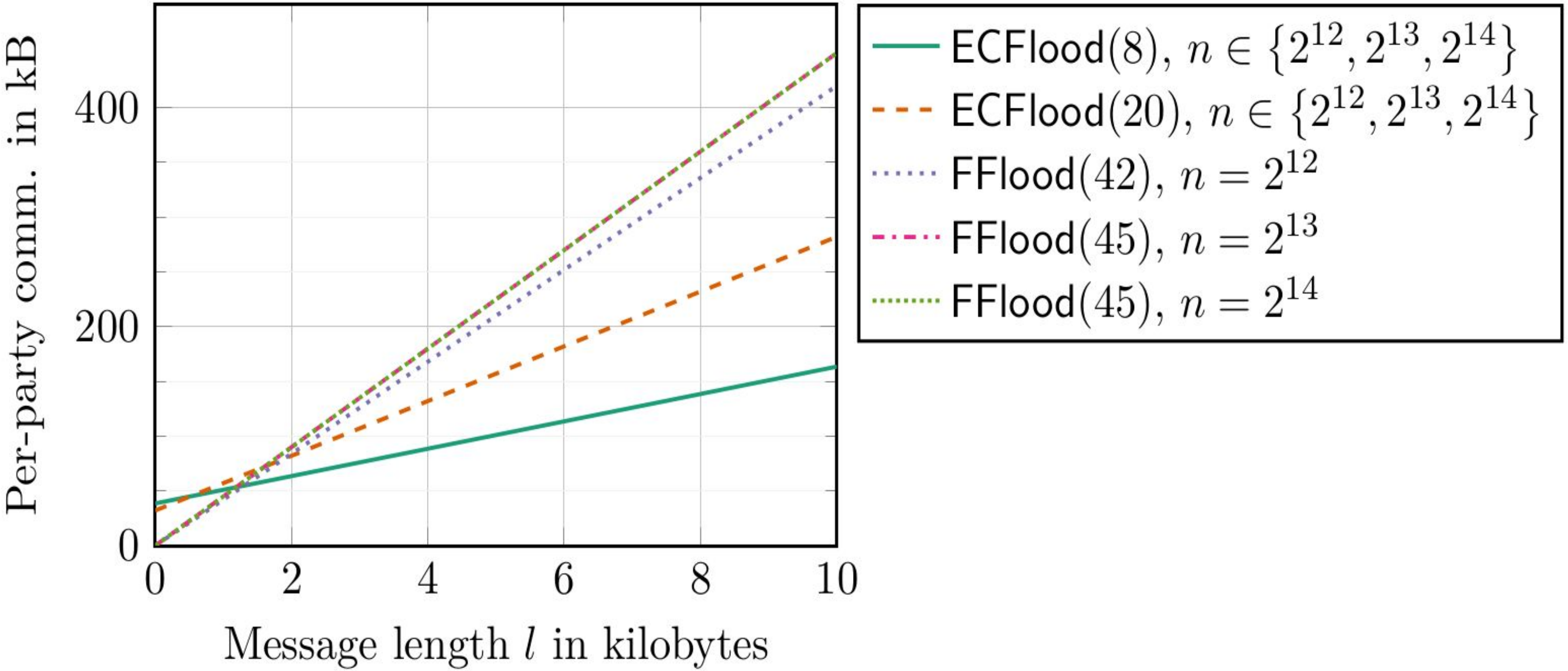
# ECFlood



Add membership for weak cryptographic accumulator
For example Merkle Trees [Mer89]
Erasure Correcting Codes
$\mu$
$\xi$-WeakFlooding
For example FFlood
$P_1$
$P_2$
$P_n$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$
$\geq \mu \cdot \theta$

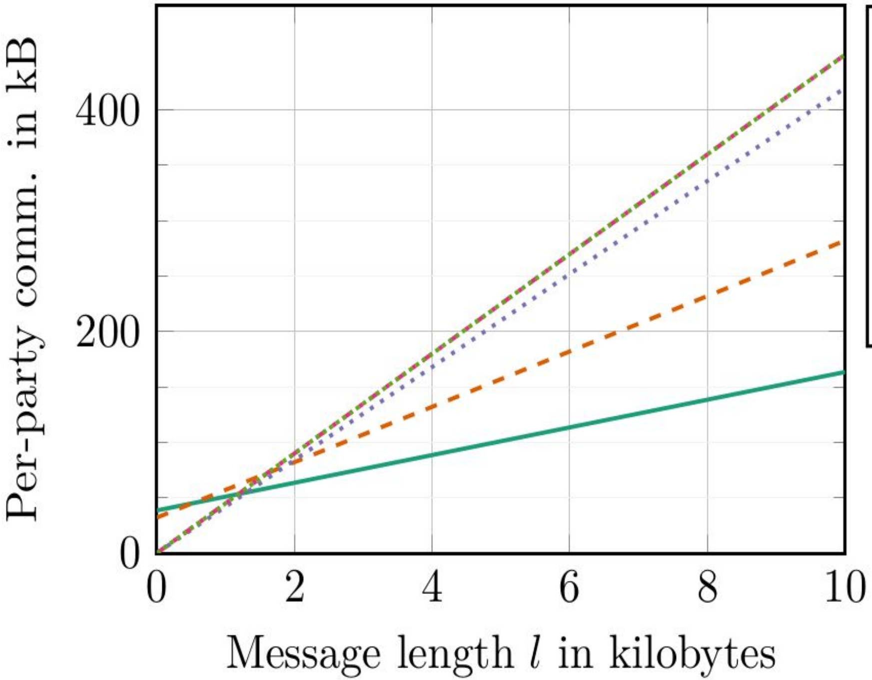
# How long should the message be?

*n* = number of parties.
FFlood = Send to *k* random parties with increasing *k* to reduce error rate.
ECFlood (x) = Our new protocol with parameter x.



*n* = number of parties.
FFlood = Send to *k* random parties with increasing *k* to reduce error rate.
ECFlood (x) = Our new protocol with parameter x.

# How long should the message be?




Optimal for:

$l = \Omega((\log(n) + \kappa) \cdot (\log(\log(n)) + \kappa))$

# Per-party communication lower bound

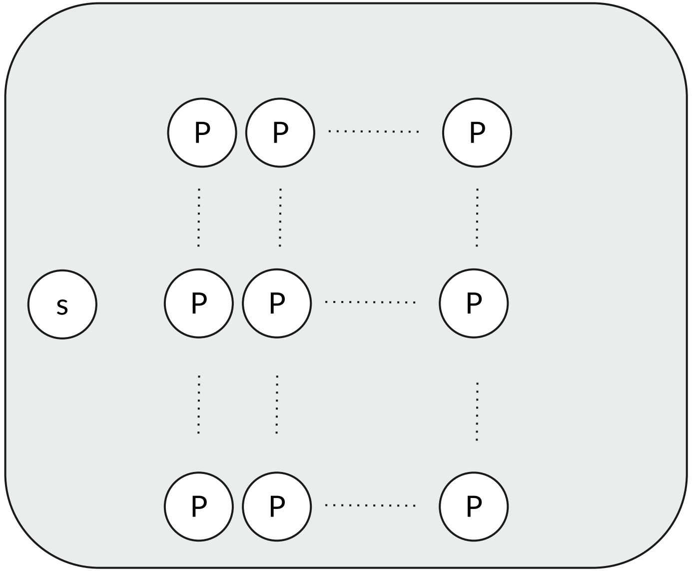
*Theorem:* Any flooding protocol must have max per party communication $\Omega(l \cdot \gamma^{-1})$.

# Per-party communication lower bound

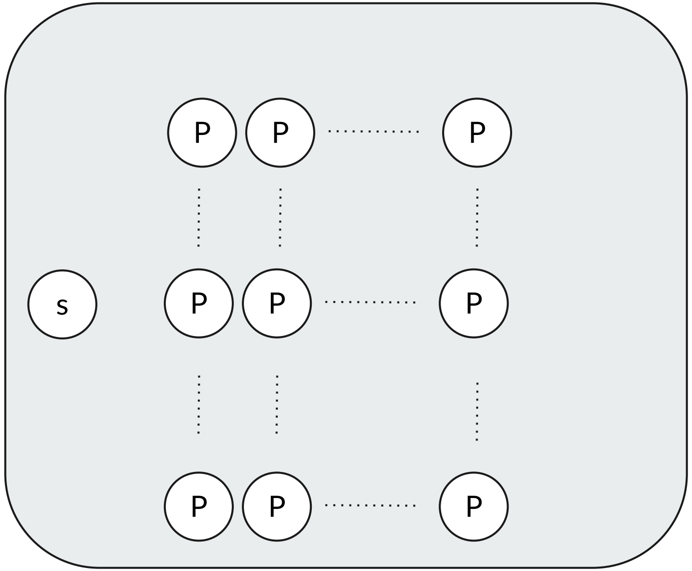
*Theorem:* Any flooding protocol must have max per party communication $\Omega(l \cdot \gamma^{-1})$.

| Protocol | Max neighbors | Max per-party communication | Diameter |
|---|---|---|---|
| **ECCast** | $n - 1$ | $O(l \cdot \gamma^{-1})$ | 2 |
| **ECFlood** | $O(\gamma^{-1} \cdot (\log(n) + \kappa))$ | $O(l \cdot \gamma^{-1})$ | $O(\log(n))$ |

# Optimality of $O(l \cdot \gamma^{-1})$ per party communication

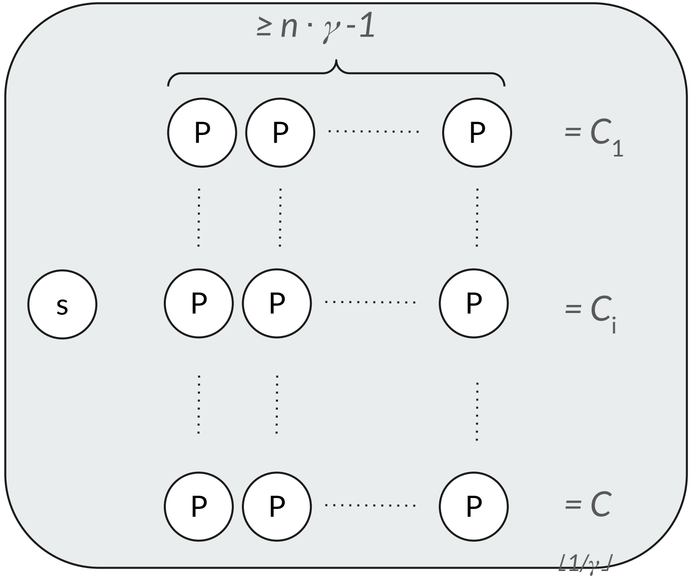


# Optimality of $O(l \cdot \gamma^{-1})$ per party communication




😈 strategy:

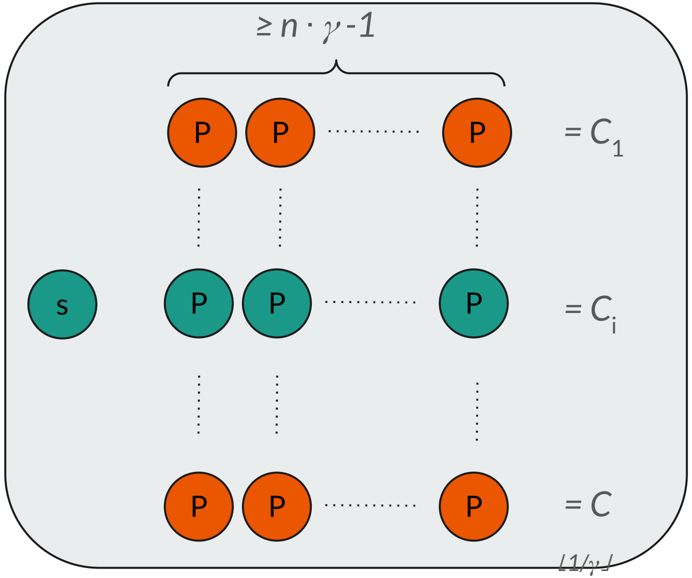
# Optimality of $O(l \cdot \gamma^{-1})$ per party communication




😈 strategy:

1. **Divide parties into sets of size ≈ $n \cdot \gamma$.**

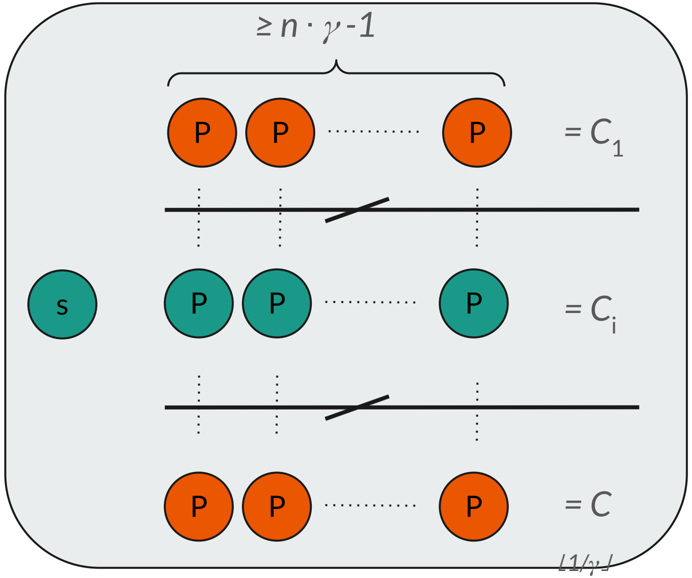
# Optimality of $O(l \cdot \gamma^{-1})$ per party communication




strategy:

1. Divide parties into sets of size ≈ $n \cdot \gamma$.
2. **Choose random *i* and corrupt everyone but sender and $C_i$.**

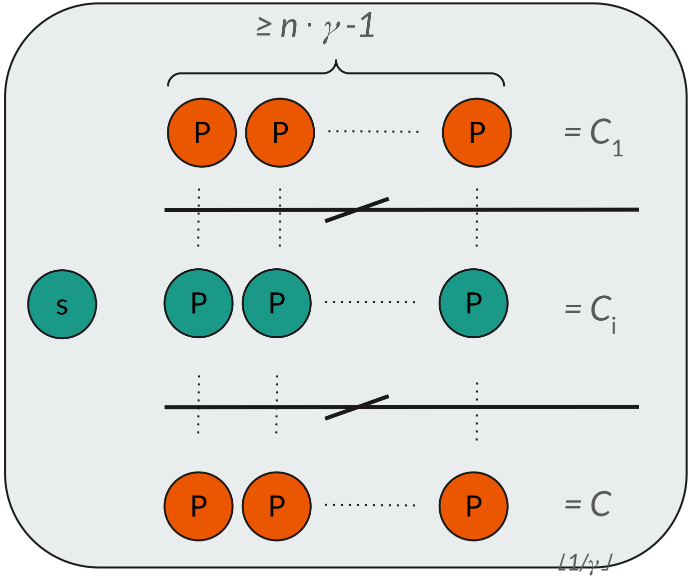
# Optimality of $O(l \cdot \gamma^{-1})$ per party communication



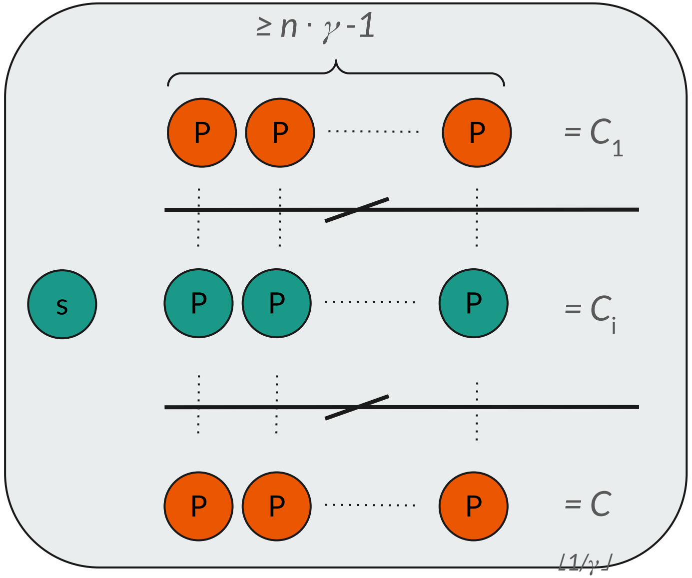

😈 strategy:

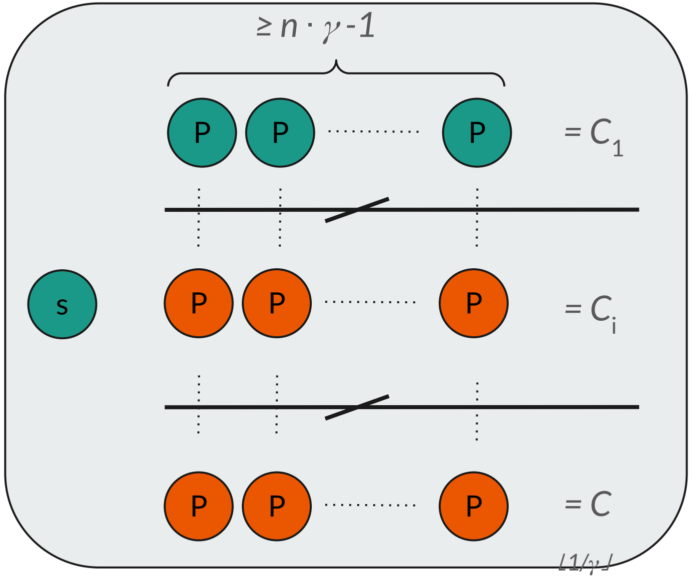
1. Divide parties into sets of size ≈ $n \cdot \gamma$.
2. Choose random *i* and corrupt everyone but sender and $C_i$.
3. **No dishonest cliques communicates with other cliques.**

# Optimality of $O(l \cdot \gamma^{-1})$ per party communication

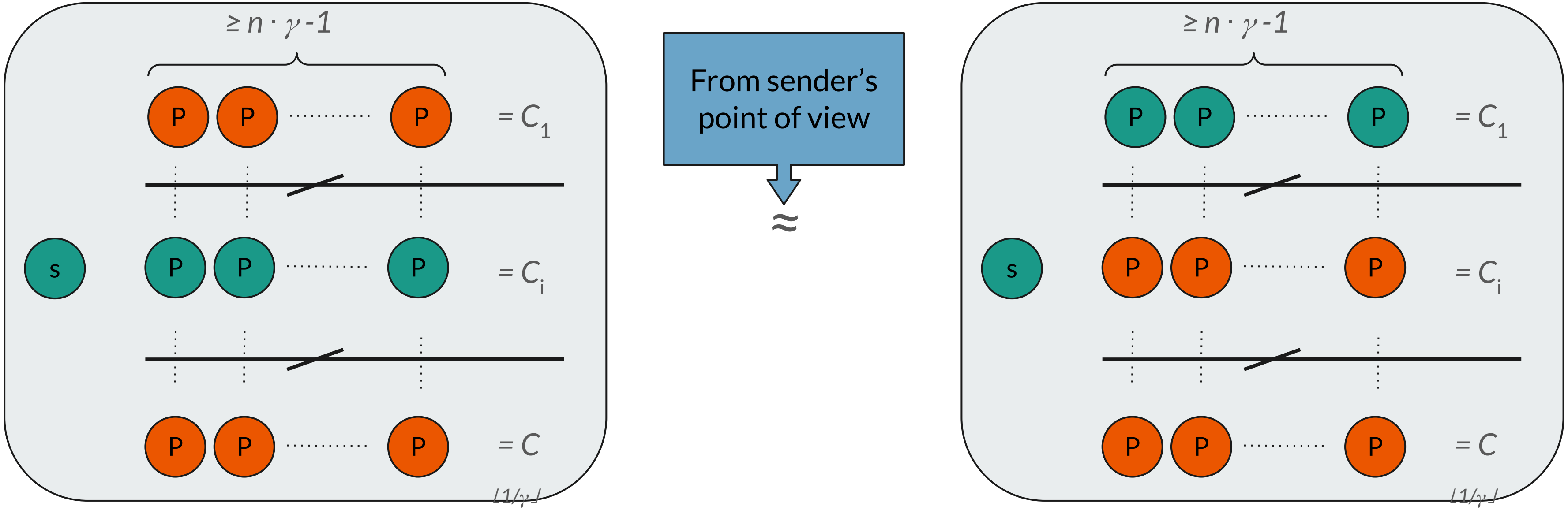


# Optimality of $O(l \cdot \gamma^{-1})$ per party communication

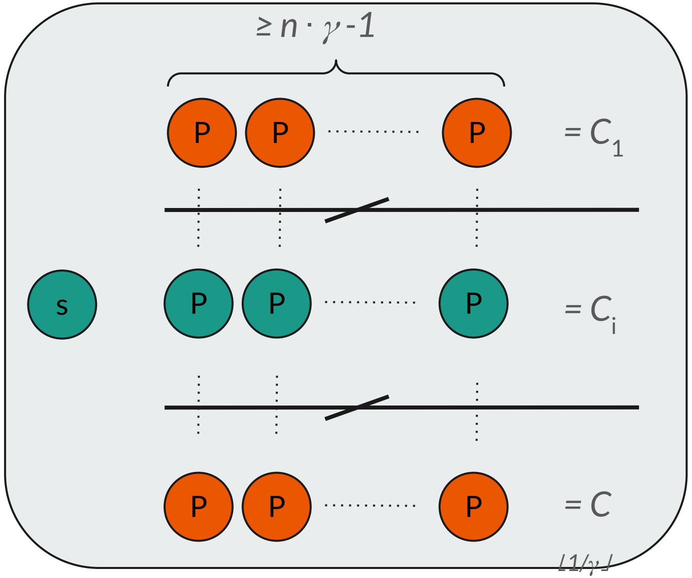


# Optimality of $O(l \cdot \gamma^{-1})$ per party communication



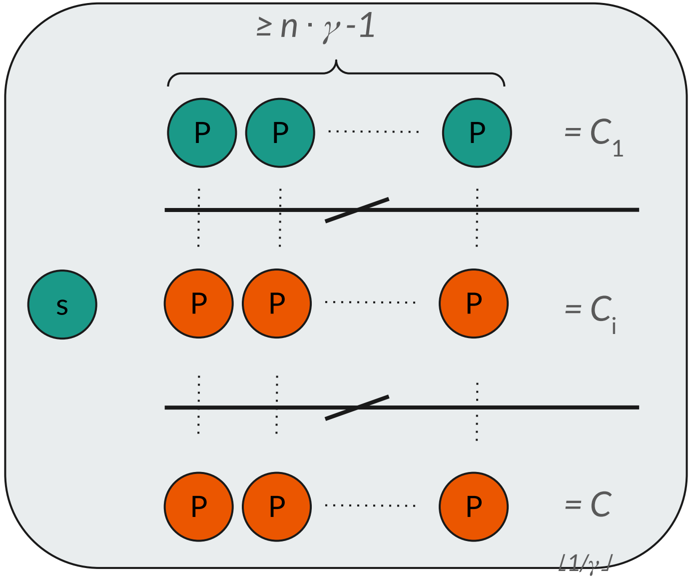
# Optimality of $O(l \cdot \gamma^{-1})$ per party communication




From sender's point of view

≈

Each $C_i$ must receive $l$ bits from the sender with overwhelming probability.



# Also in the paper

*Theorem:* Property-based flooding implies UC-flooding.

# Also in the paper

Because there are no secrecy requirements for flooding protocols they are easy to simulate

*Theorem*: Property-based flooding implies UC-flooding.

# Also in the paper

Because there are no secrecy requirements for flooding protocols they are easy to simulate

*Theorem:* Property-based flooding implies UC-flooding.

*Theorem:* Secure protocol in the non-weighted setting *implies* another protocol that is secure in the weighted setting.

# Also in the paper

Because there are no secrecy requirements for flooding protocols they are easy to simulate

*Theorem:* Property-based flooding implies UC-flooding.

Constructive proof by emulating parties as [LMMRT22]

*Theorem:* Secure protocol in the non-weighted setting *implies* another protocol that is secure in the weighted setting.

# Conclusion

In this talk:

1. Presented ECFlood: A flooding protocol with a logarithmic neighborhood, a logarithmic diameter, and only $O(l \cdot \gamma^{-1})$ per party communication.
2. Presented simulations showing practical advantages over existing approaches.
3. Shown the optimality of a $O(l \cdot \gamma^{-1})$ per party communication.

Details and additional results in the full version of the paper: https://eprint.iacr.org/2022/1723

Contact: soren.eller.thomsen@partisia.com

Partisia

# References

**[MNT22]**: Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. In Yevgeniy Dodis and Thomas Shrimpton, editors, advances in Cryptology – CRYPTO 2022, Cham, 2022. Springer Nature Switzerland.

**[LMMRT22]**: Liu-Zhang, CD., Matt, C., Maurer, U., Rito, G., Thomsen, S.E. (2022). Practical Provably Secure Flooding for Blockchains. In: Agrawal, S., Lin, D. (eds) Advances in Cryptology – ASIACRYPT 2022. Springer, Cham.

**[KMG03]:** Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. IEEE Trans. Parallel Distributed Syst.,14(3):248–258, 2003.

**[RS60]:** Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. Journal of The Society for Industrial and Applied Mathematics, 8:300–304, 1960.

**[Mer89]:** Ralph C. Merkle. A certified digital signature. In CRYPTO, volume 435 of Lecture Notes in Computer Science, pages 218–238. Springer, 1989.