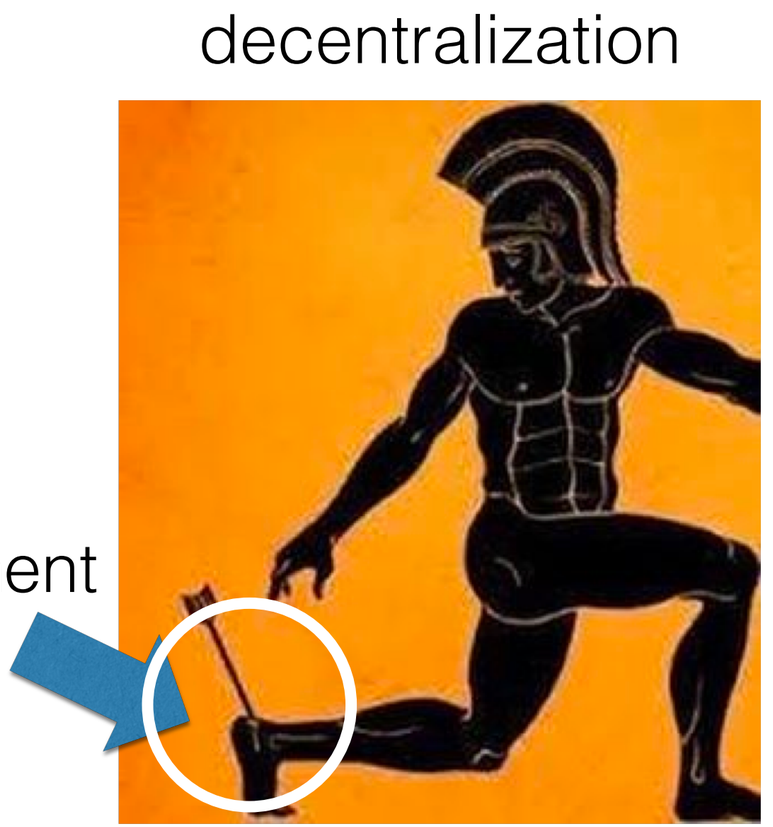
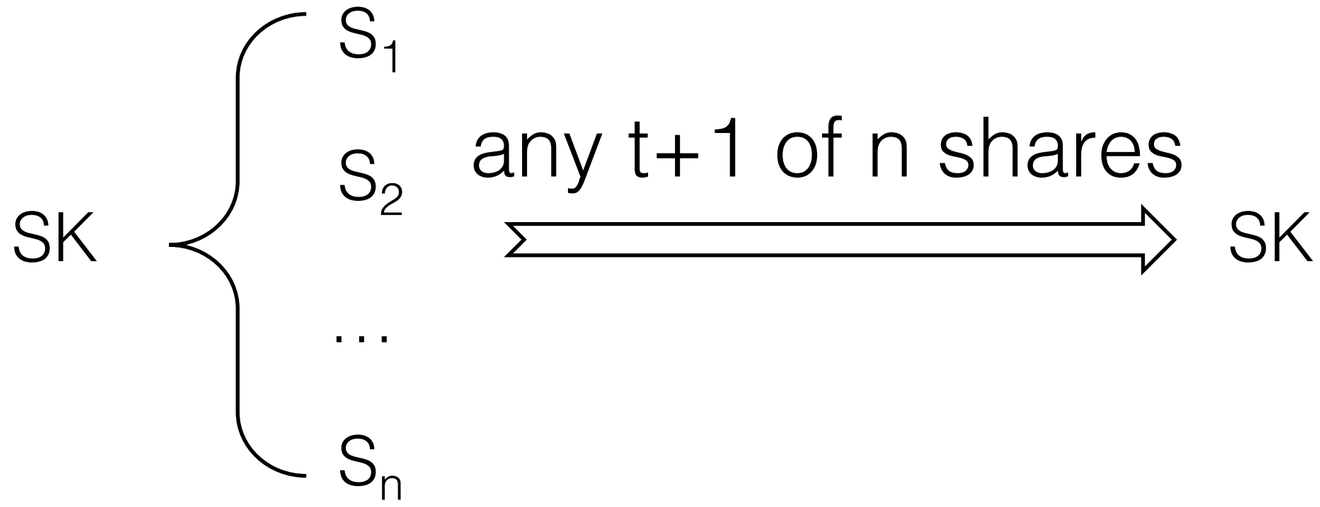
A fundamental problem at large…

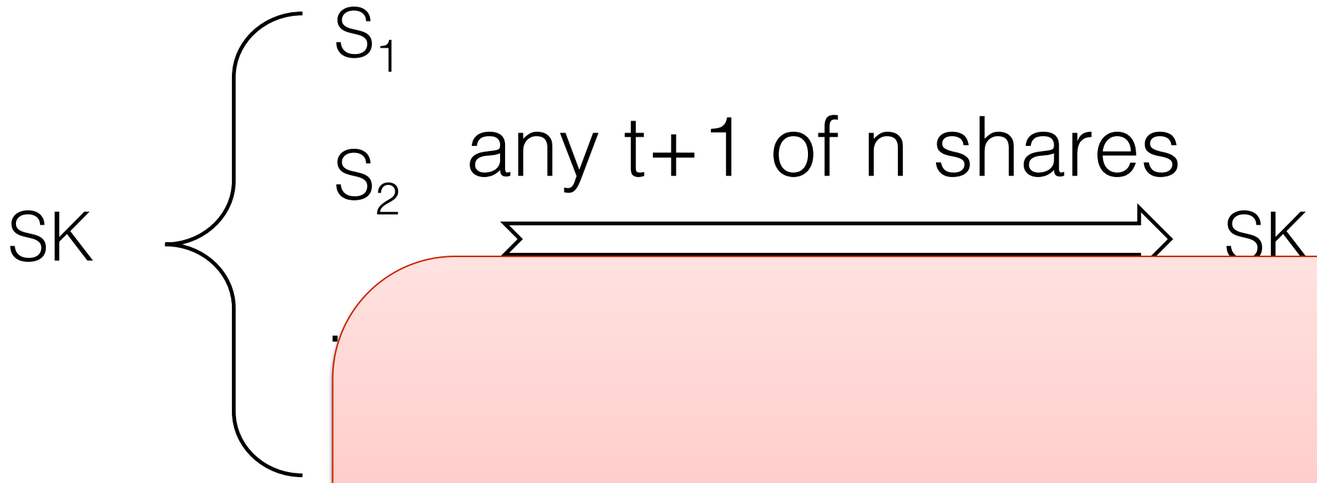
- *Centralized* secret management
  - At the heart of many (ironically) decentralized systems
  - E.g., use Coinbase to store Bitcoin keys



A classic solution (Shamir 1979)



- A committee of n nodes hold the secret collectively
- Harder to attack: must corrupt at least *t+1* nodes
- Threshold cryptography: (e.g.) sign without reassembling SK

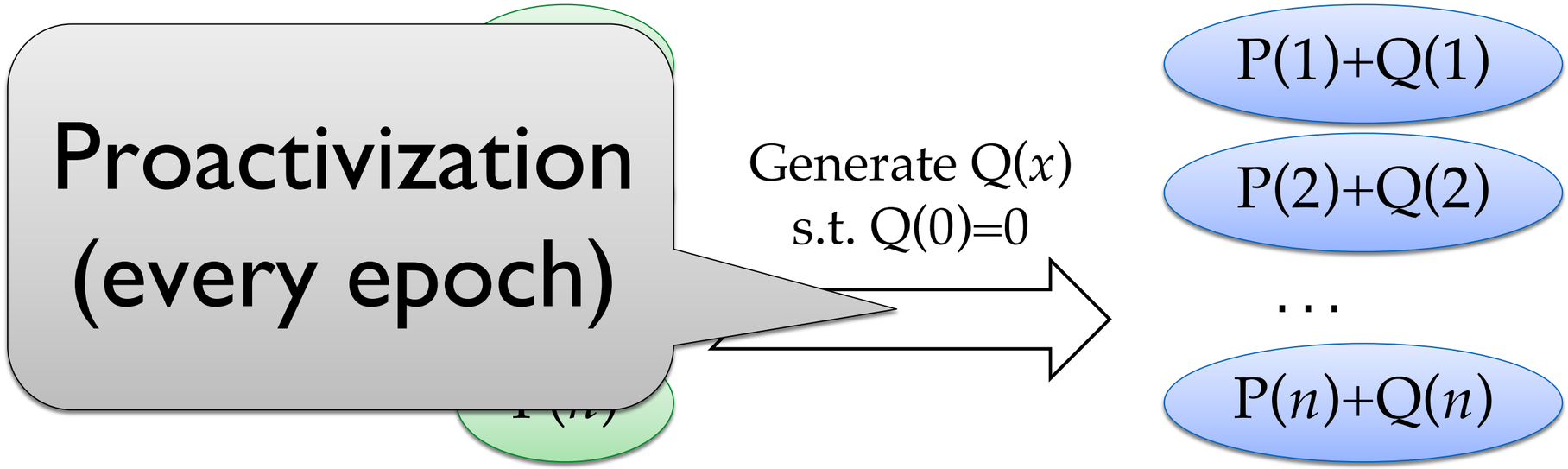


- Choo
  P(0)=
- Share
- Any *t+1* shares can interpolate P(*x*) and SK

- Mobile Adversary
- Requires a static committee

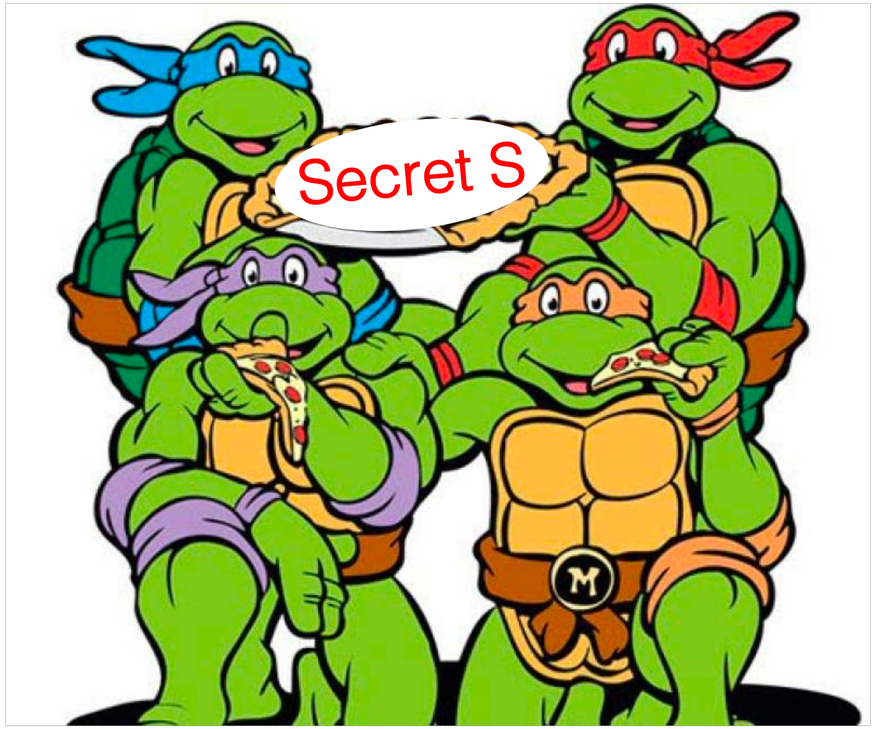
Mobile adversaries

- Overtime adv may eventually compromise t+1 nodes
- Limit the corruption rate (instead of the absolute number)



Green shares and blue ones are incompatible!

Node churn



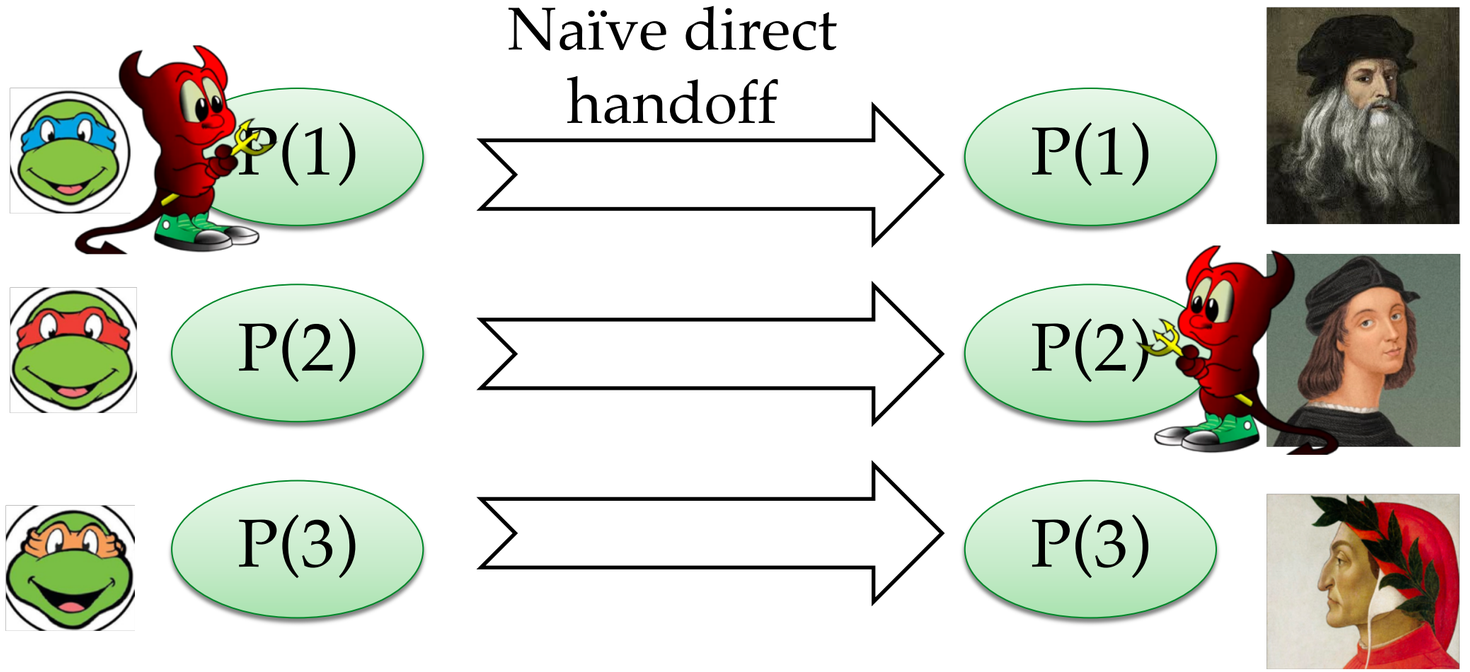
committee at time T

Next epoch

Committee at time T+1

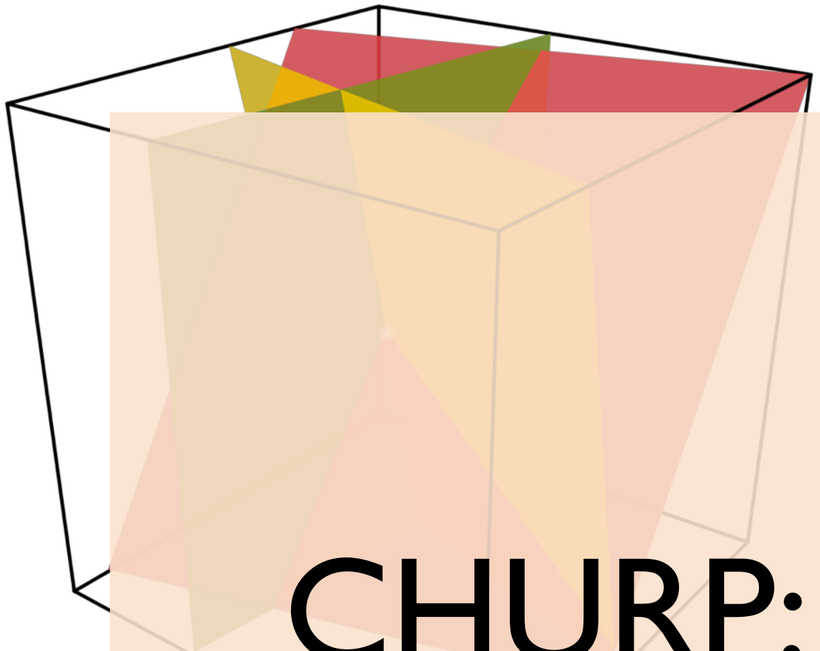
Secret sharing no longer works

- A mobile adversary can corrupts *t* nodes in each committee
  - 2t shares give away the secret!
- E.g., a naïve (1, 3)-SS:



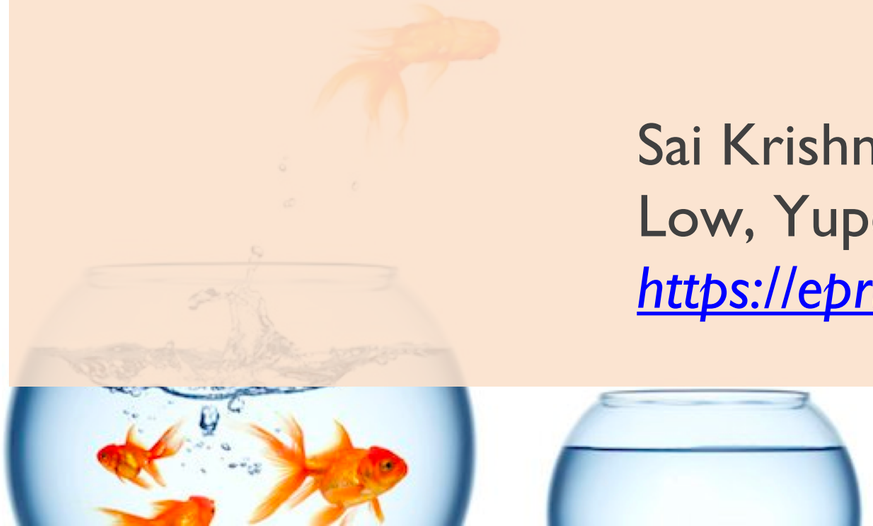
Needs a practical solution

- Previous works used a “blinding” trick
  - $O(n^4)$ complexity
  - Unfortunately, doesn’t scale to large systems
  - 5.3GB / epoch, for a committee of 100 nodes,



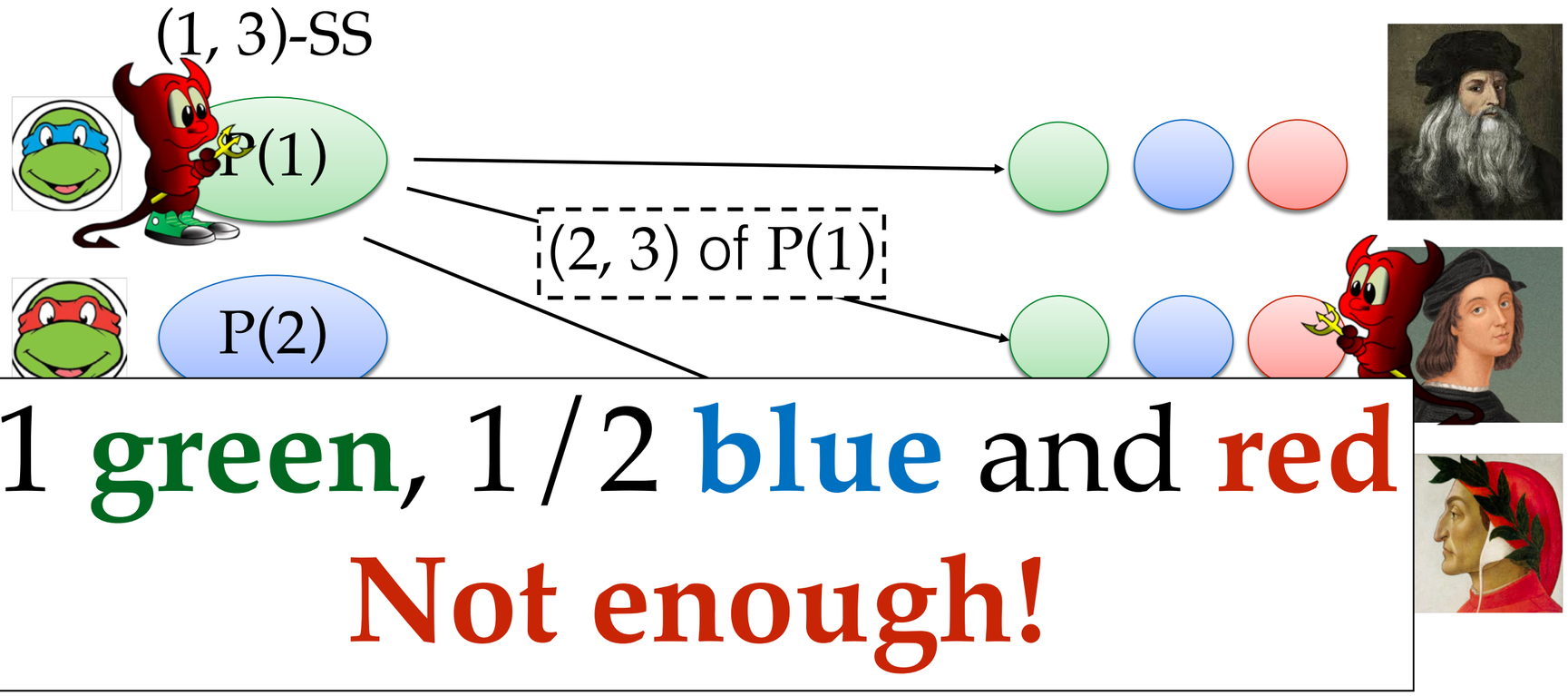
CHURP: Dynamic-Committee Proactive Secret Sharing

Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels and Dawn Song. *In submission.*
https://eprint.iacr.org/2019/017



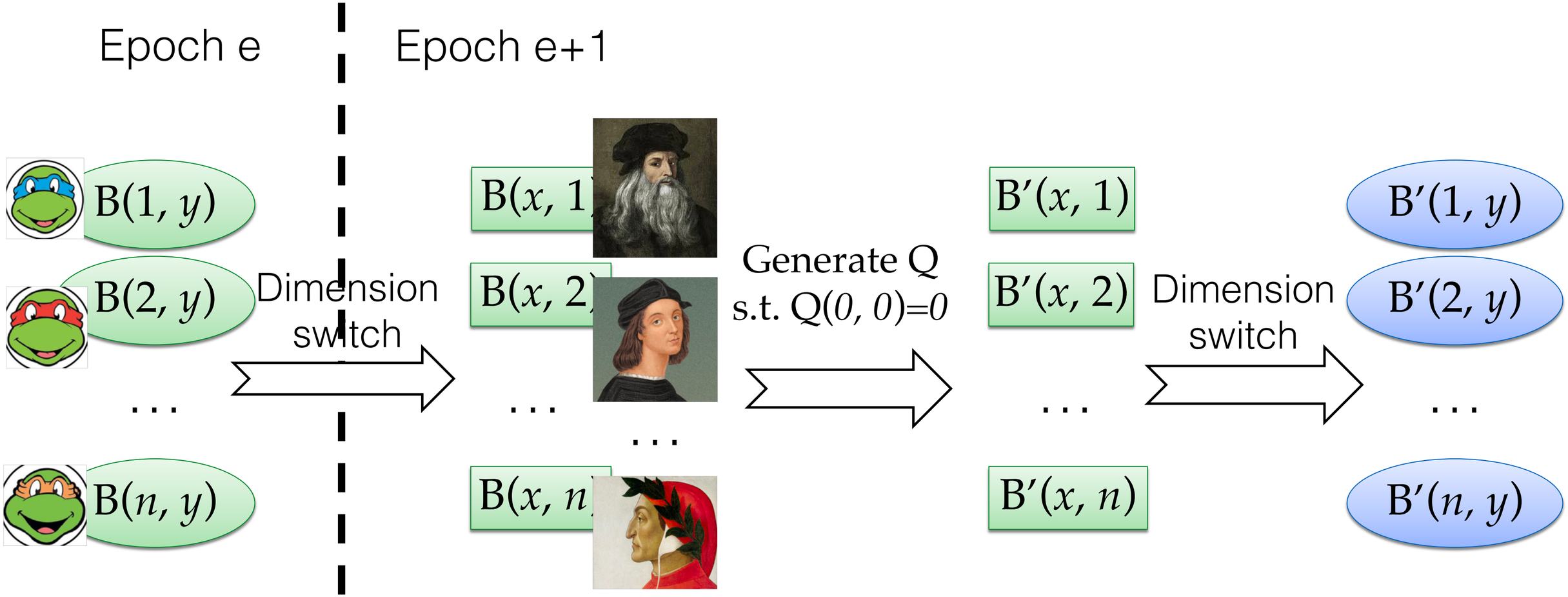
CHURP's main idea

- Making 2t shares insufficient to recover SK
- "dilute" the share > proactivize > "concentrate"



Do it efficiently (Technical details)

- Use a bivariate $(t, 2t)$-deg poly $B(x, y)$ such that $B(0, 0)$=SK
- ***Full*** shares (2t degree polys): $B(i, y)$
- ***Reduced*** shares (t-degree polys): $B(x, i)$
- The secret SK can be recovered from
  - t ***full*** shares, OR
  - 2t ***reduced*** shares
- "Dimension-switch": dilute by switching to the 2nd dimension



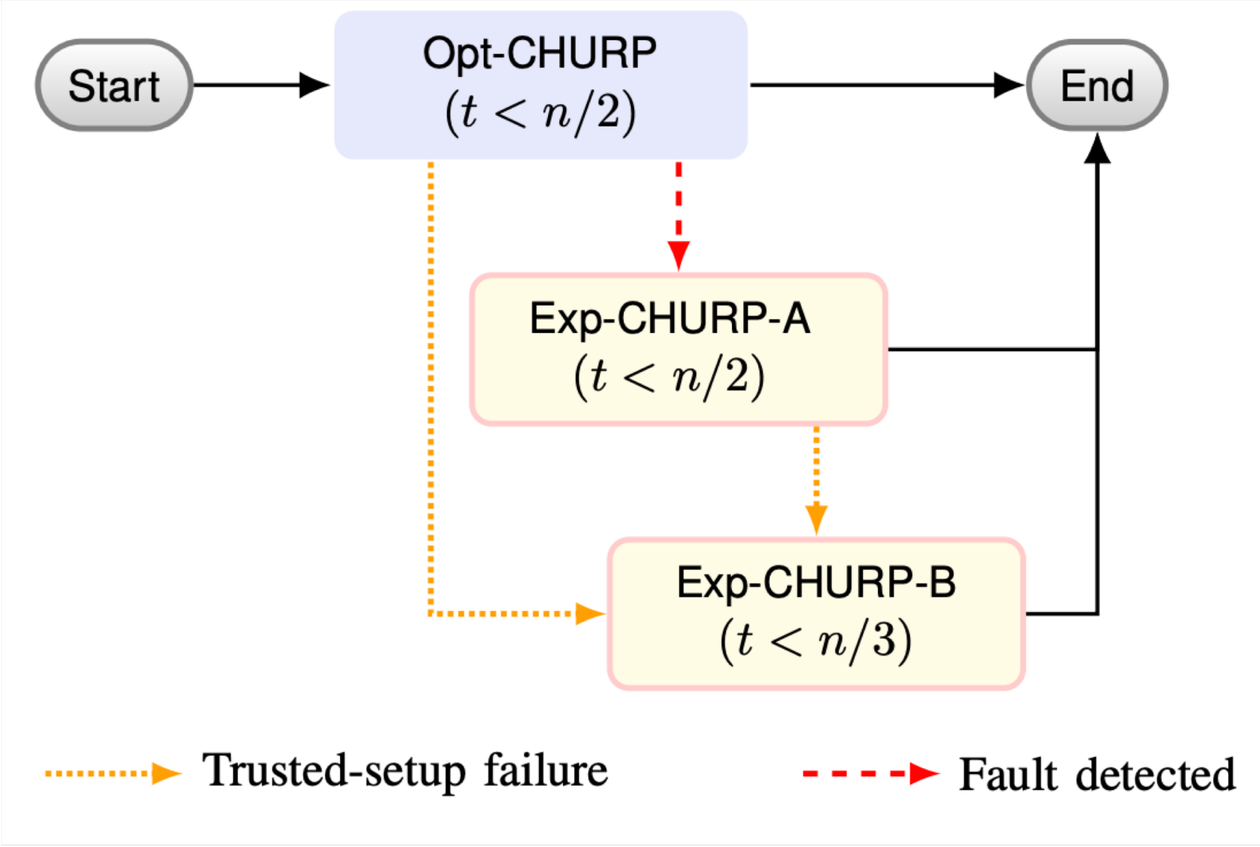
"Dilute the secret" "proactivze" "Concentrate again"

Efficient Proactivization

- Generate Q such that Q(x, y)
- A naïve solution takes O(n^3)
- > Don’t need Q(x, y), just Q(x, j) is enough
- 2t nodes generate s_i [takes O(n^2)]
- Set $Q_j(x, j) = R_j(x)$ and share [another O(n^2)]

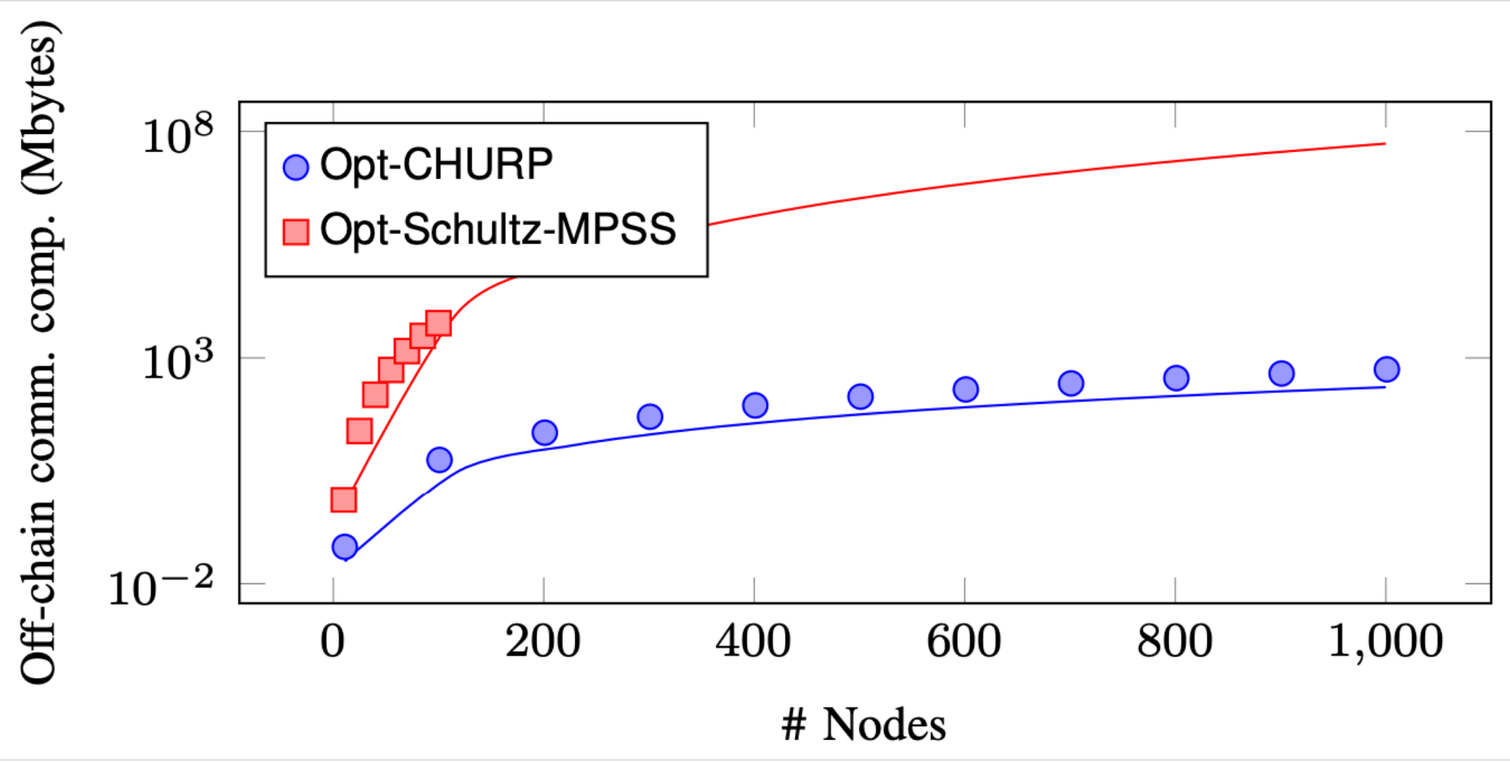
CHURP: a tiered protocol

- Highly efficient optimistic path ($O(n^2)$)
- Slower but more robust pessimistic path ($O(n^3)$)
- Can also hedge against trusted setup failure!
  - By checking meta-invariants



Performance

- Communication complexity comparison



For 100 nodes:

- Previously: 5.3GB
- CHURP: 2.3MB
- 2300x improvement!

Lots of applications

- Efficient secret sharing with node churn
- Applications
  - Enable decentralized confidential smart contracts (e.g. Ekiden)
  - Decentralized Identity (DIDs)
  - decentralized secure multiparty computation (MPC)