Setup-Free Committee Sampling With Subquadratic Communication

Ran Cohen, Poulami Das and **Tal Moran**

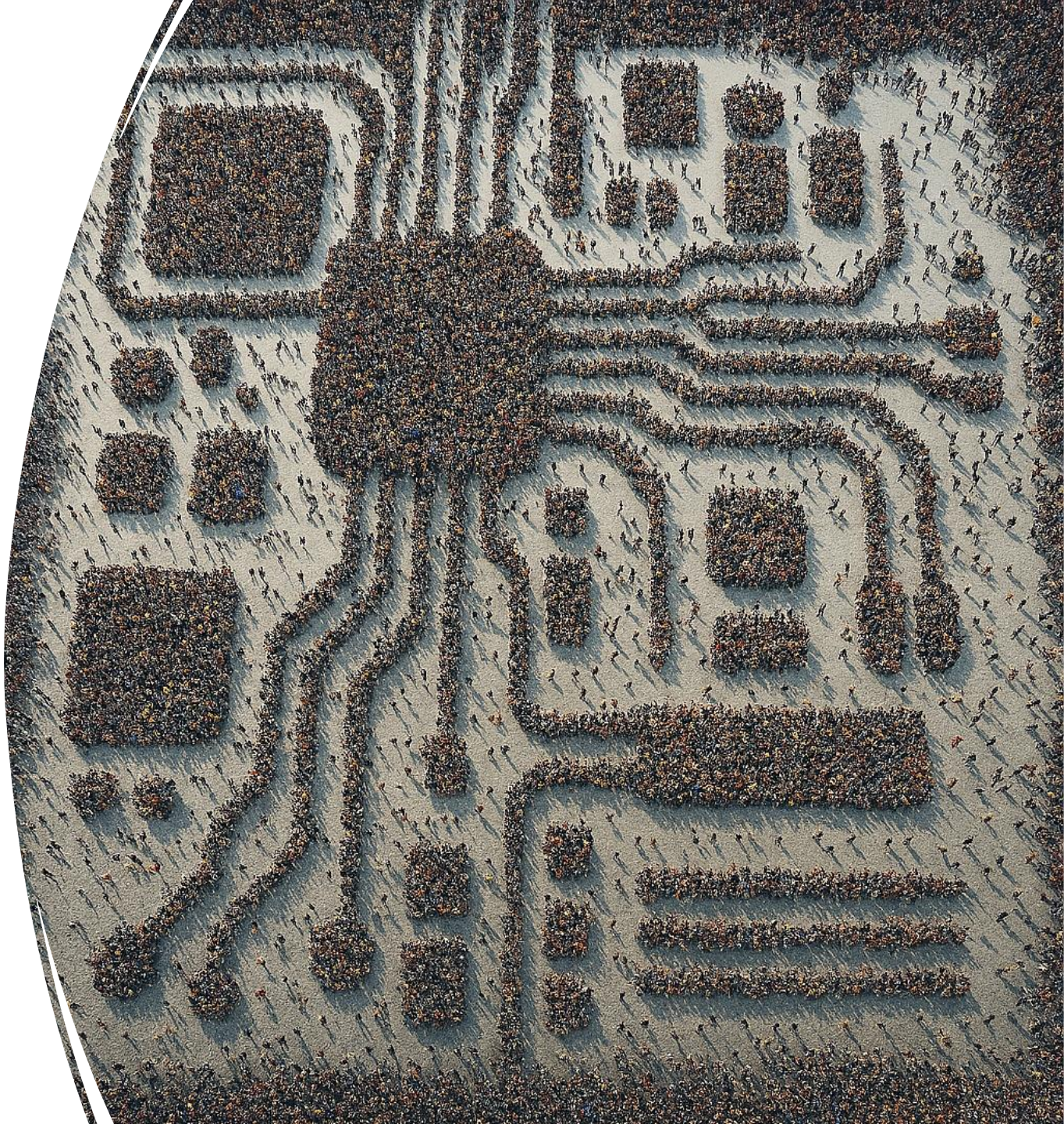
Reichman University

Least Authority
PRIVACY MATTERS

Reichman University

Secure Computation at Scale

- Our goal: MPC for many 1000s of parties
- Many potential use cases:
  - **Generating trusted setup**
  - Privacy-preserving machine learning
  - Secure computation for IOT mesh networks
  - …
- But all-to-all (n^2) comm. infeasible for large n

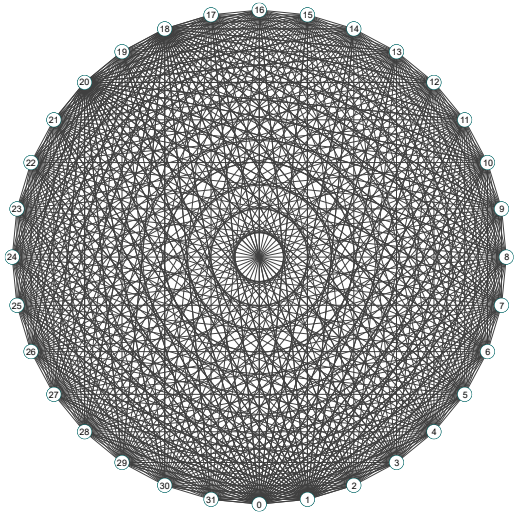


Scaling Solutions

Theory

Elect Committees

- Small size $m \ll n$
- Committee runs protocol
- Comm. is $O(m^2 \cdot n)$



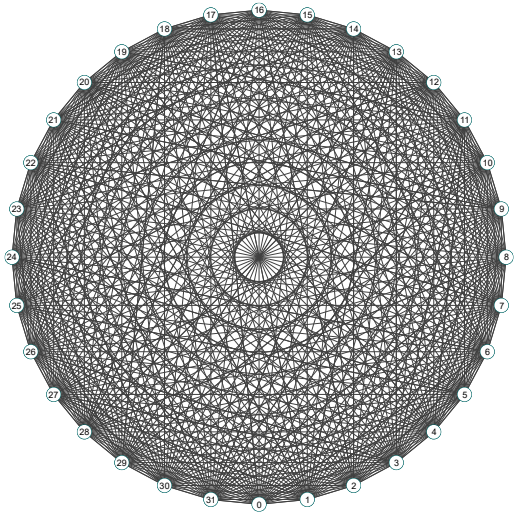
Honest nodes only know their neighbors

Don't need direct comm. with committee

Practice

"Gossip networks"

- Use sparse comm. graph $G = (V, E)$
- Parties forward messages
- Comm. is $O(m \cdot |E|)$



Who are the participants?

- Traditional protocols have fixed set of parties
  - Everyone knows who else is participating
- Less practical in very large-scale settings
  - E.g., parties may not agree on identification authorities
- **Permisionless** Protocols:
  - Parties can join or leave network
  - May not know who or how many other participants
- Easy to spoof identities
  - Sybil attacks: can generate huge number of fake identities

Replacing Identity With Resource Proofs

- Permissionless protocols replace id with resource proofs
  - CPU Work
  - Storage
  - Proofs of Useful Work
  - …
- Messages from other parties must include resource proof.
- For concreteness in this talk:
  - Resource is PoW:
  - Given challenge ch, find nonce r such that $H(ch||r) < p$
    - H is random oracle
    - $1/p$ is expected difficulty.

Our Contributions

- Committee election **without setup**
  - Build on core idea of Andrychowicz and Dziembowski [Crypto15]
- Define generalized resource proofs
  - Support wide variety of resources (not just PoW)
- Communication-efficient protocol $o(n^2)$
  - Gives *graded consensus* on committee
  - Can be bootstrapped to full consensus in honest-majority setting

	Comm.	Construct	Trust Assump.	Resources
AD15	$\Omega(n^2)$	Beacon	Honest Majority	Hash-based PoW
This work	$o(n^2)$	Committee	None	Generic RP

Electing a committee: Sortition

- Given prob. p, parties can provably "self-select"!
- High-level construction:
  - Each party has VRF (vk, sk) key pair
  - VRFs are all evaluated on a common input x
  - Party vk is selected if $f_{vk}(x) < p$
- Simple attack if adversary can choose keys after learning x
- Requires *trusted setup*.
  - E.g.: Trusted PKI
    - Trusted party samples keys for parties
- Other setup assumptions possible
  - E.g.: Common reference string
- Generating setup is hard at scale!
  - Using committee election to solve is circular

Main Protocol Ideas

- Use cryptographic sortition to elect committee
- Generate provably “fresh” randomness for VRF nonce
- Resource proofs *also* require fresh random challenges
  - Prevent “pre-mining”
- Main technical tool: efficient distributed timestamping
  - Used to prove freshness and order of events

Efficient Distributed Timestamping

- Two types of timestamp:
  - Ex ante timestamping:
    - prove a value *was* known *before* protocol end
  - Ex post timestamping:
    - Prove a value was *not* known until *after* protocol start
- Committee-election from timestamping?
  1. Ex post timestamp to generate challenges
     - Party i gets their own fresh challenge ch_i
  2. Parties generate VRF keys and resource proofs
  3. Ex ante timestamp for keys and resource proofs
  4. Use sortition to select committee
     - Party i selected if $f_{vk_i}(ch_i) < p$

Timestamp Generation
- everyone participates
- Efficient comm. $\tilde{O}(n)$
- **No agreement!**

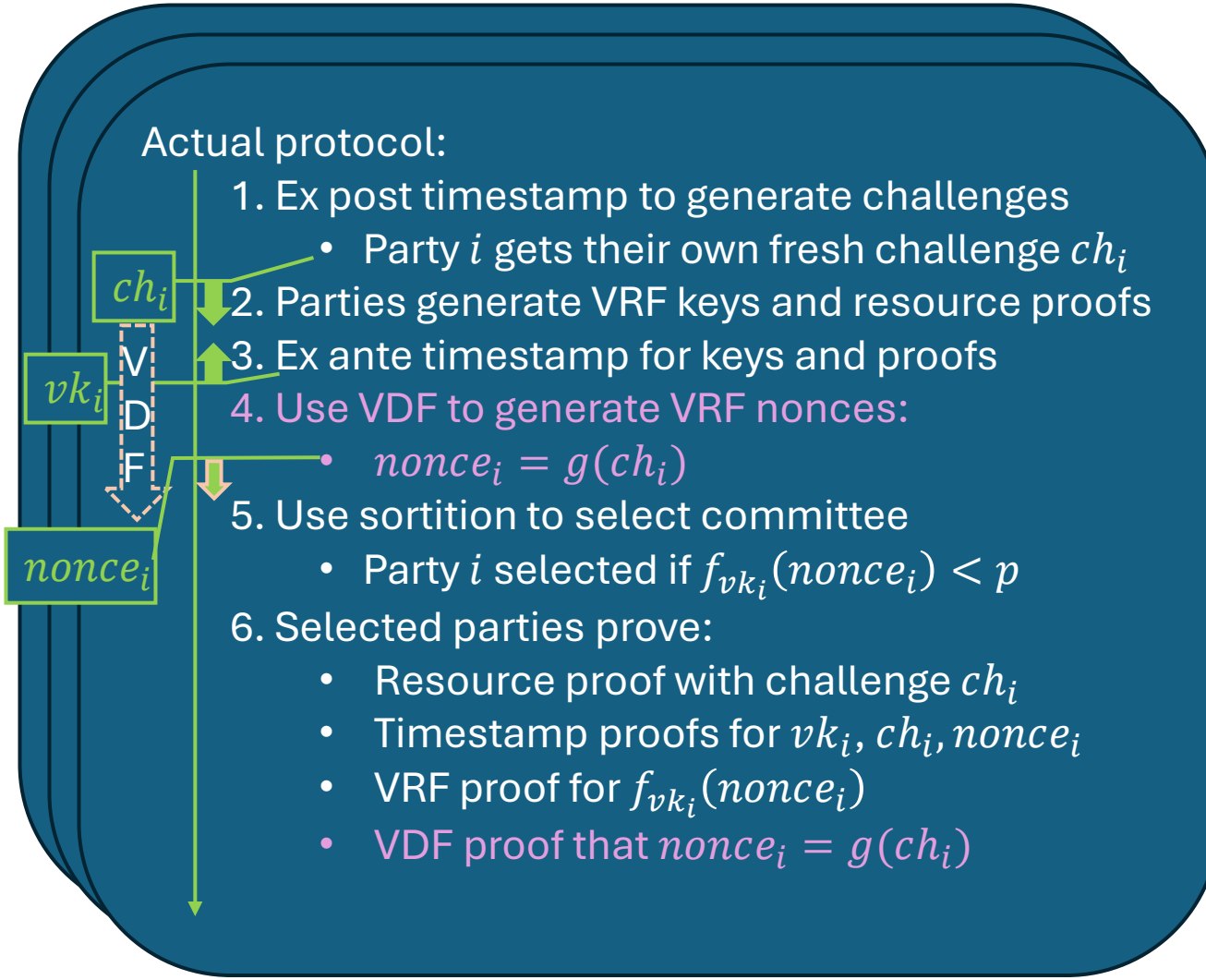
Timestamp Proof Verification
- Small number of provers $k = o(n)$
- Comm. $\tilde{O}(n \cdot k)$
- **Graded agreement on proofs**

Prevent pre-mining

Limits mining time

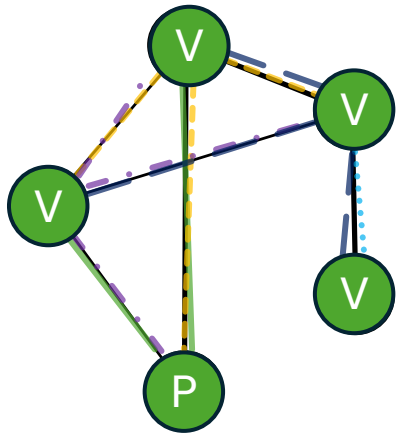
Committee Election From Timestamping

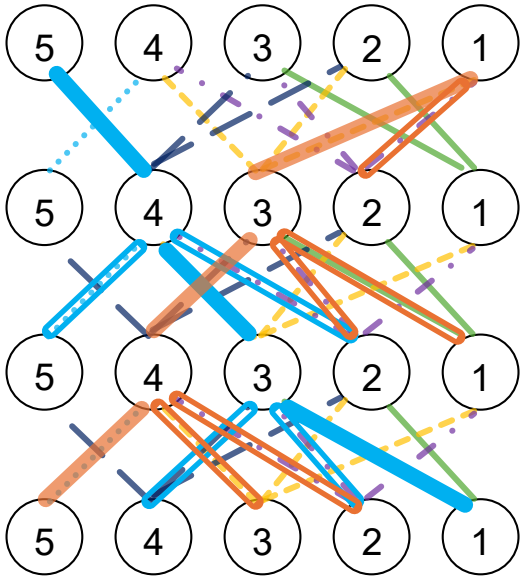
- Problem #1:
  - Challenge is generated before VRF key
- Problem #2:
  - Adv. Can generate many nonces with same resource proof
    - Only reveals “winning” nonce!
    - Ex post timestamp doesn’t prevent this – all nonces are fresh.
- Extra ingredient:
  - Verifiable Delay Function (VDF)
    - “moderately hard”
    - Lower bound on time to compute $g(x)$.
    - Efficient proof that $y = g(x)$



Gossip-based Timestamping

- Main tool: Merkle DAG
- Distributed computation of generalized Merkle Tree
- Timestamp Generation:
  - In each round, hash all incoming messages
  - Send hash to all neighbors
  - Store all messages





Ex Ante Timestamp:

- First message is hash of input
- Hash of final messages is commitment
- Proof is Merkle path from prover's input to verifier's commitment

Ex Post Timestamp:

- First message is random
- Hash of final messages is challenge
- Proof is Merkle path from verifier to prover's challenge

(Some) Open Questions

- Is a VDF necessary?

- Dealing with dynamic gossip graphs?
  - Our protocol requires parties to rerun Merkle-DAG *with the same communication graph*
  - Deployed gossip implementations change over time!

- Async comm.?
  - Our protocol strongly requires a synchronous network.