

Building Giant Threshold Cryptosystems with Lightweight Cryptography

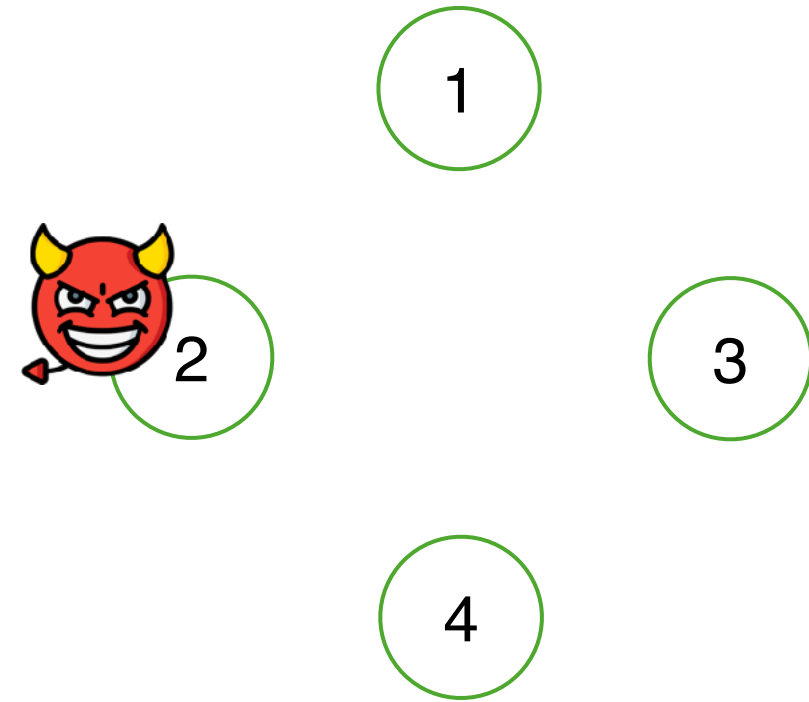
Aniket Kate

Purdue University / Supra Research

NIST Workshop on Multi-Party Threshold Schemes 2026



System Setting



System Setting

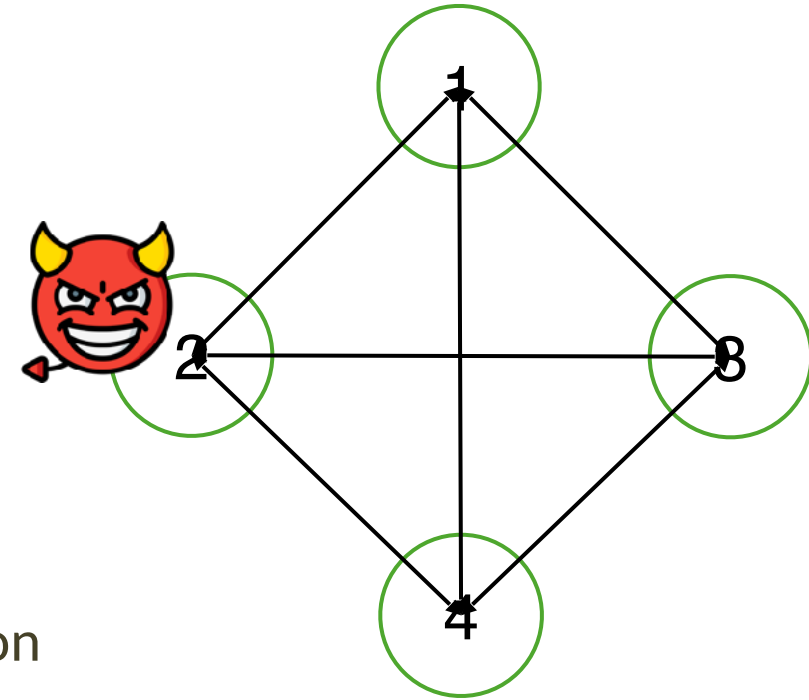
- Communication Model

- Secure point-to-point channels (e.g., SSL/TLS)
- Asynchronous message delivery
 - Messages can be arbitrarily delayed (no know bound on message delivery)



System Setting

- Communication Model
 - Secure point-to-point channels (e.g., SSL/TLS)
 - Asynchronous message delivery
 - Messages can be arbitrarily delayed (no know bound on message delivery)
- Decentralization Level
 - Number of nodes (n) in **several hundreds**
 - Optimal fault-tolerance of 33% malicious failures



Bounded Synchrony vs Asynchrony

Bounded Synchrony vs Asynchrony

Can the Internet be considered synchronous enough to build Broadcast channels?

Bounded Synchrony vs Asynchrony

Can the Internet be considered synchronous enough to build Broadcast channels?

Unless the latency per round in minutes is acceptable, the Internet can not be considered to be synchronous

Bounded Synchrony vs Asynchrony

Can the Internet be considered synchronous enough to build Broadcast channels?

Unless the latency per round in minutes is acceptable, the Internet can not be considered to be synchronous

Current MPC libraries will lose liveness if

- Synchrony bounds are broken, or
- One or more nodes behave maliciously

Security Notation

Security Notation

- Perfect/statistical security
 - Build the protocol completely using Reed-Solomon codes, or other codes
 - Drawback: additive communication overhead can be $O(n^5)$ or higher
 - ▶ For Asynchronous MPC with $n > 3t$, it is $O(n^{14})!$ [Crypto '24]

Security Notation

- Perfect/statistical security
 - Build the protocol completely using Reed-Solomon codes, or other codes
 - Drawback: additive communication overhead can be $O(n^5)$ or higher
 - ▶ For Asynchronous MPC with $n > 3t$, it is $O(n^{14})!$ [Crypto '24]
- Computational Security
 - Cryptographic assumptions such as DLog, Pairings, and LWE
 - Setups such as PKI, FHE, or Threshold Homomorphic Encryption
 - Positive: communication overhead can be $O(n^3)$ or lower
 - Drawback: Each cryptographic operation requires *1ms* to compute
 - ▶ For FHE (with verifiability), for decryption, it can take several seconds
 - ▶ For DumboMPC [Usenix Sec '25], computation complexity is $O(n^2)$ per gate

Traditional View-points

- Unconditionally secure threshold crypto is in the theory land
- Research Efforts
 - Reduce communication complexity
 - ▶ Typically, cubic, quadratic, or linear in #parties
 - Reduce the number of rounds
 - ▶ For FHE, it is not interactive until you decrypt
 - Computation overhead is not the first-order concern!
- Experiments are restricted to <50 nodes, and smaller circuits!

Can't we just use better machines to deal with computation cost?

Can't we just use better machines to deal with computation cost?

1. Prohibitive Cost

2. Limits participation

3. Better machines are also coming with better communication infrastructure that we are ignoring.

Computation overhead can't be ignored!

Computation overhead can't be ignored!

- HoneybadgerMPC & AsynchroMix [ACM CCS'19]

Computation overhead can't be ignored!

- HoneybadgerMPC & AsynchroMix [ACM CCS'19]
 - For MPC,
 - We preferred a non-GOD protocol for offline computation phase
 - As Discrete log-based computations were not scaling for $n > 50$

Computation overhead can't be ignored!

- HoneybadgerMPC & AsynchroMix [ACM CCS'19]
 - For MPC,
 - We preferred a non-GOD protocol for offline computation phase
 - As Discrete log-based computations were not scaling for $n > 50$
 - Anonymous Broadcast

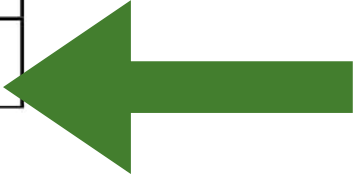
Protocol	Rounds	Comm. complexity	Compute
PowerMix	2	$O(n)$	$O(n + k^2)$
Switching Network	$\log^2 k$	$O(n \log^2 k)$	$O(n \log^2 k)$

k is the anonymity set

Computation overhead can't be ignored!

- HoneybadgerMPC & AsynchroMix [ACM CCS'19]
 - For MPC,
 - We preferred a non-GOD protocol for offline computation phase
 - As Discrete log-based computations were not scaling for $n > 50$
 - Anonymous Broadcast

Protocol	Rounds	Comm. complexity	Compute
PowerMix	2	$O(n)$	$O(n + k^2)$
Switching Network	$\log^2 k$	$O(n \log^2 k)$	$O(n \log^2 k)$



k is the anonymity set

How to deal with the computation *vs* communication tradeoff?

An Intermediate Solution!

Lightweight Cryptography



- Hash Functions as random oracle (RO), **and nothing more!**
- Secure Communication Links using symmetric-key cryptography
- Disadvantage: Lack of homomorphism

Operation	Time
Discrete-Log Exponentiation	70 μ seconds
Bilinear Pairing	600 μ seconds
Hash Computation	0.7 μ seconds
Hardware-accelerated MACs	0.02 μ seconds

Post-Quantum Security



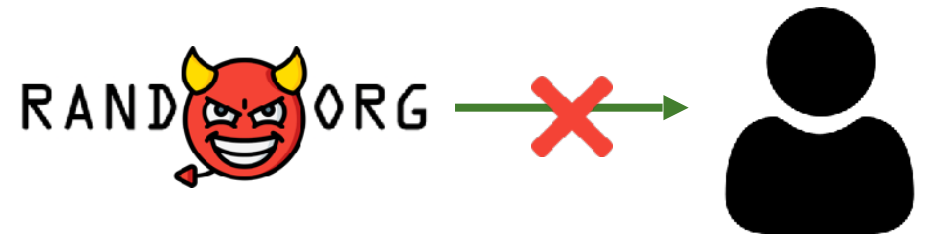
- Quantum Random Oracle Model (QROM)

HashRand: Asynchronous Random Beacon without Threshold Cryptographic Setup

Akhil Bandrupalli Adithya Bhat
Saurabh Bagchi Aniket Kate Michael K. Reiter

Random Beacons

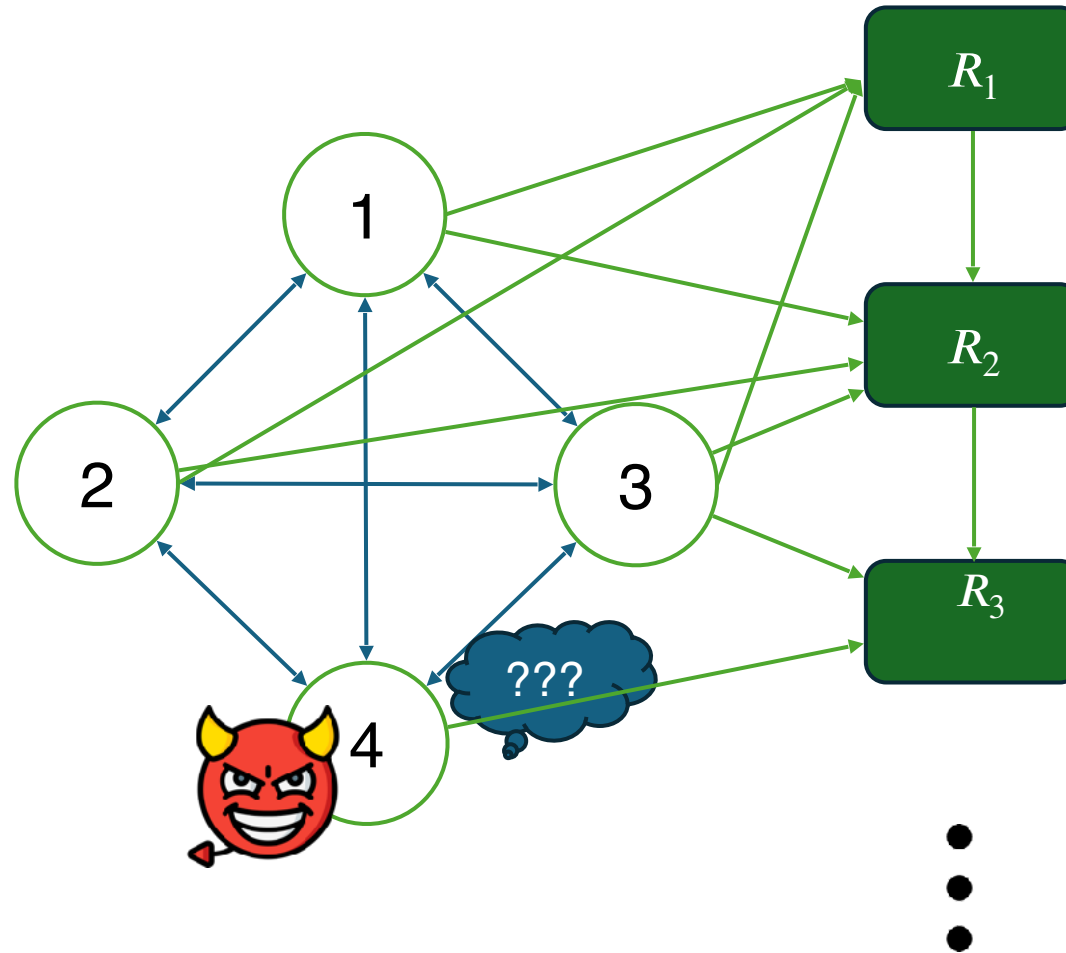
- A random beacon is a source of secure randomness emitting random numbers
- **Popular sources:** [random.org](https://www.random.org) and NIST random beacons
- Constitute a **single point of failure**



Distributed Random Beacons

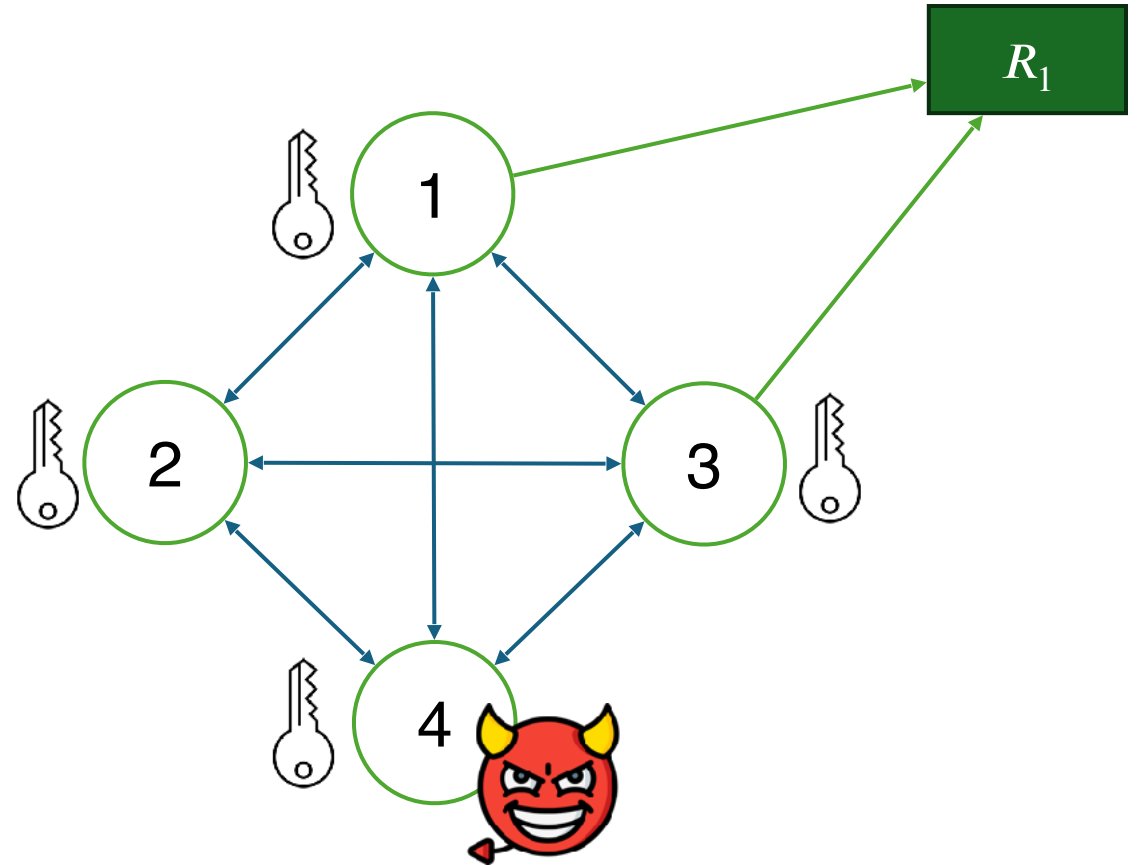
A beacon protocol has **three** properties.

1. Agreement
2. Liveness
3. Bias-Resistance and Unpredictability



Prior Random Beacon Protocols

- Protocols like DRand, Supra dVRF, Difinity dVRF use threshold unique signatures
- Require a threshold setup, which needs a Distributed Key Generation (DKG) protocol
- No post-quantum security!



Prior Beacon protocols

Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value

Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value

1

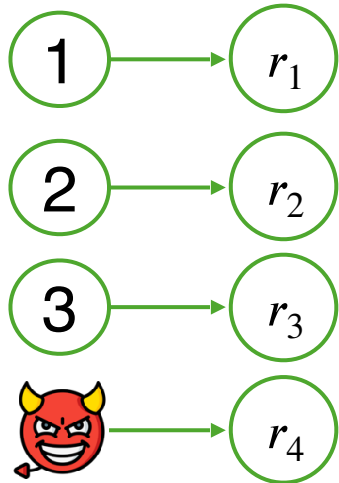
2

3



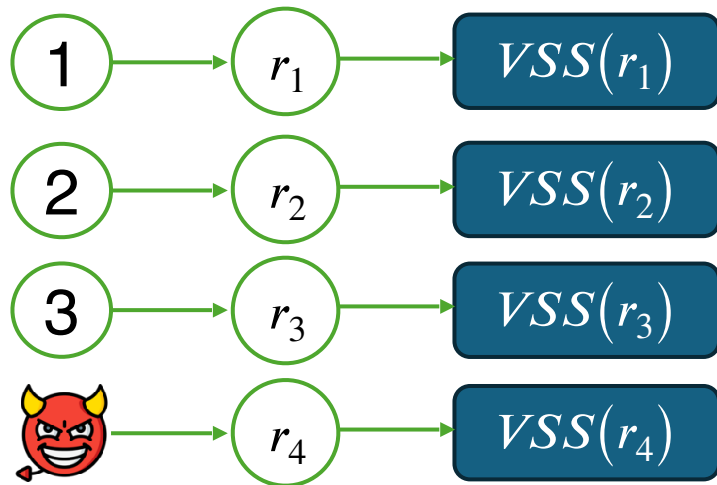
Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value



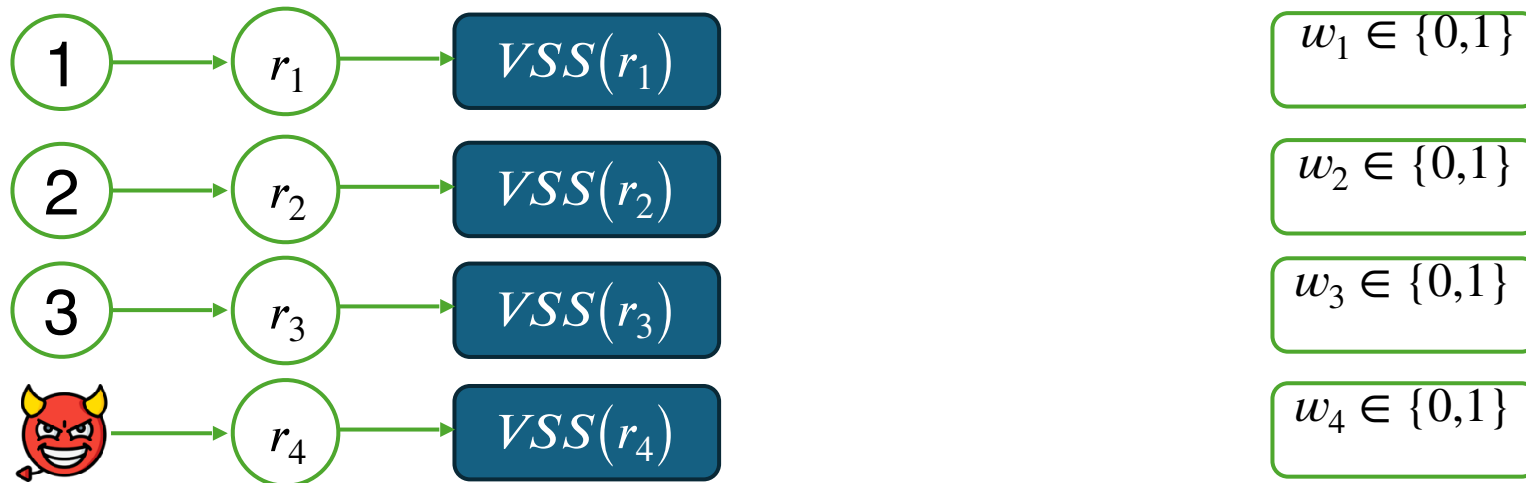
Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value



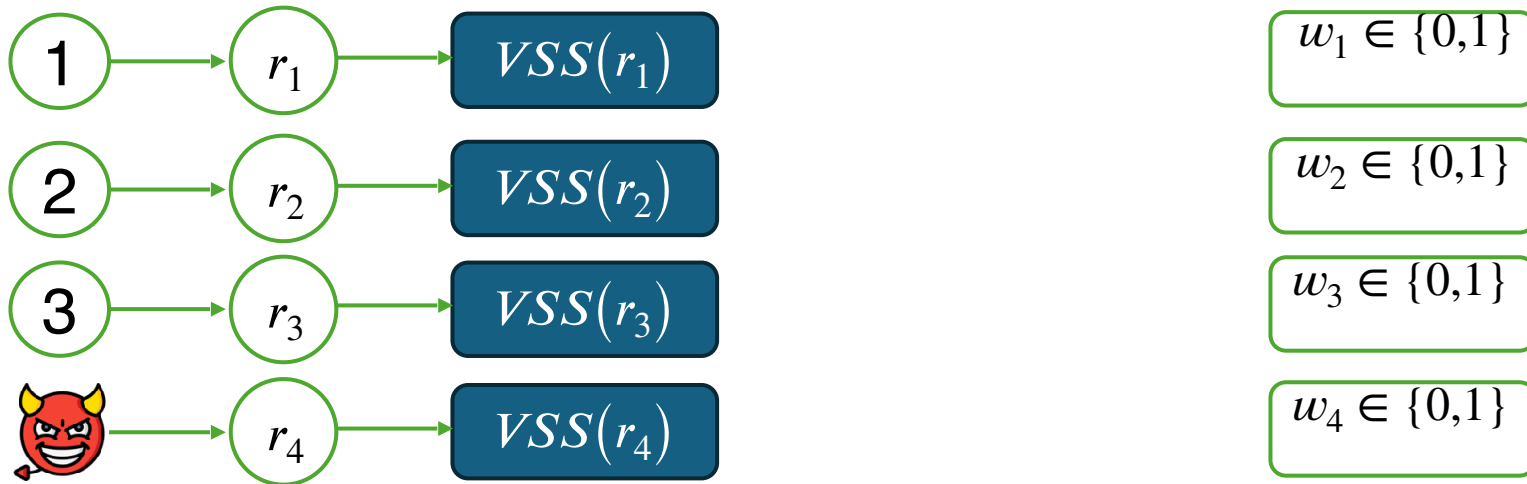
Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value



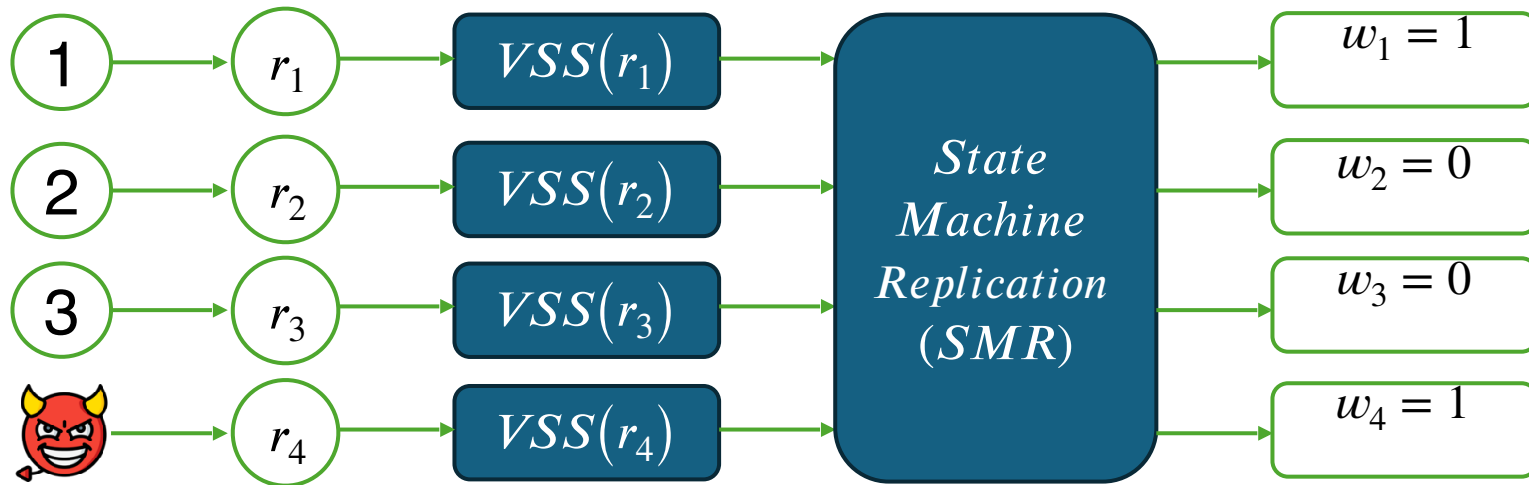
Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value
- Then, use state machine replications to agree on $t + 1$ nodes whose VSS instances succeeded



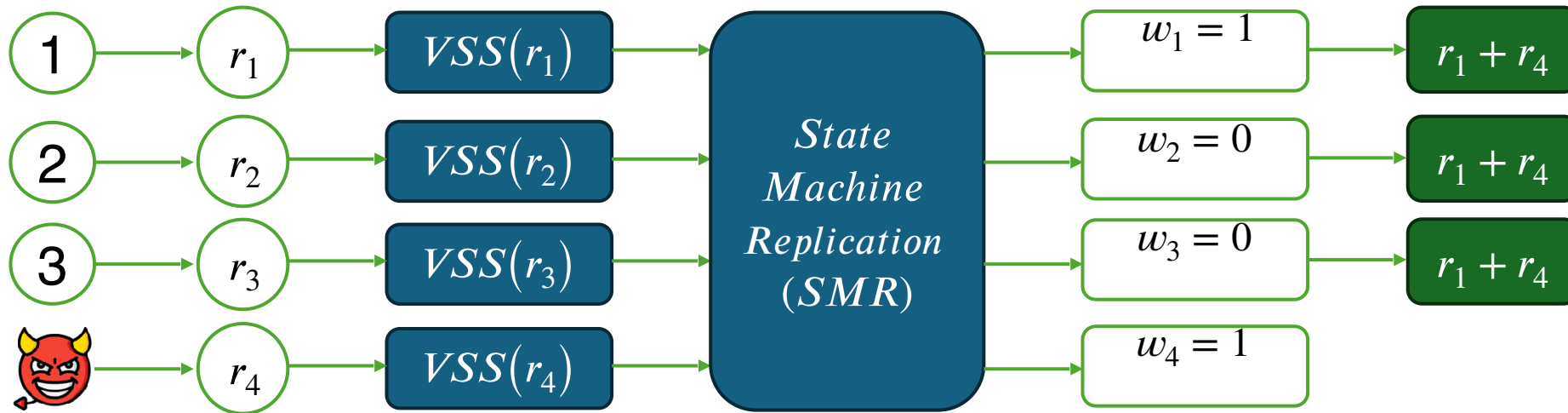
Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value
- Then, use state machine replications to agree on $t + 1$ nodes whose VSS instances succeeded



Prior Beacon protocols

- Protocols like OptRand [NDSS'23] and Spurt [S&P'22] use Verifiable secret sharing (VSS) to share a random value
- Then, use state machine replications to agree on $t + 1$ nodes whose VSS instances succeeded



Asynch. Verifiable Secret Sharing (AVSS)

- Computational AVSS
 - $O(n^2)$ communication
 - Can be reduced to $O(n)$ per secret through batching
 - Use expensive cryptographic computation
 - DLog, Polynomial commitments, Verifiable Multi-receiver encryptions
- Can we build AVSS with lightweight cryptography?

AVSS with Lightweight Cryptography

- First Construction [GRR, PODC 1998]
- Computational VSS Revisited [BKP, Asiacrypt 2011]
 - More of a feasibility result
 - $O(n^3L)$ communication for sharing L secrets
- Lightweight AVSS with Optimal Resilience [Smart & Shoup JCrypto 2024]
 - Worst case: $O(n^2L + \kappa n^3)$

Improving Computational Efficiency

Improving Computational Efficiency

- **Asynchronous weak VSS**: Honest nodes reconstruct a failure symbol \perp for a malicious dealer

[Dolev, Wang]

- Use a Hash function as a Random Oracle

Improving Computational Efficiency

- **Asynchronous weak VSS**: Honest nodes reconstruct a failure symbol \perp for a malicious dealer

[Dolev, Wang]

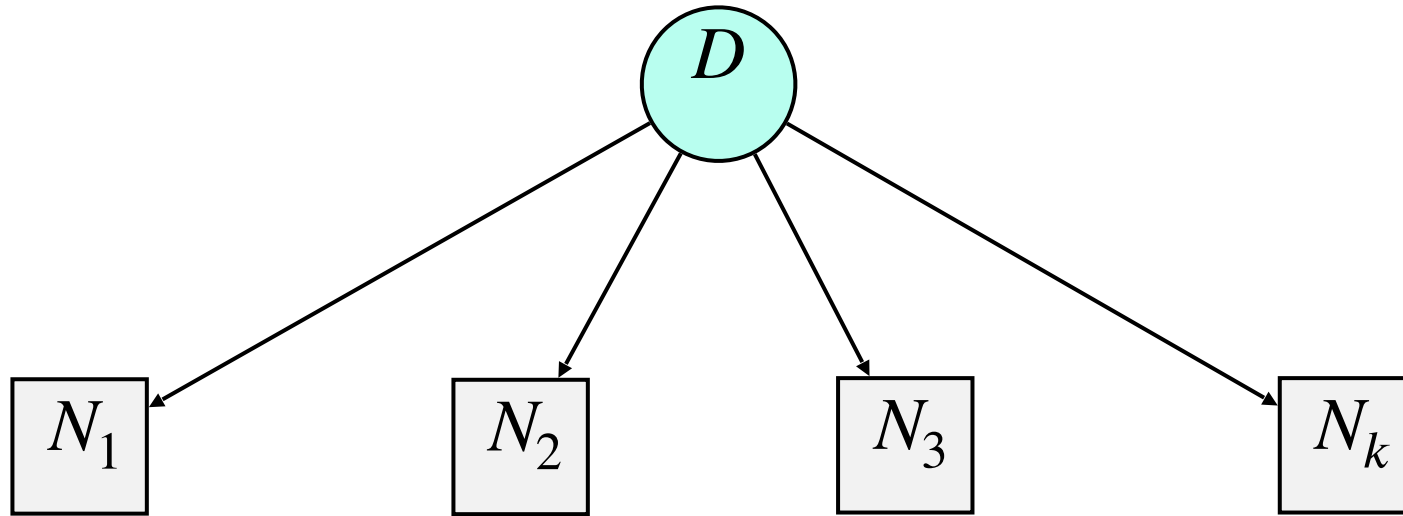
- Use a Hash function as a Random Oracle

- **Communication Complexity**:

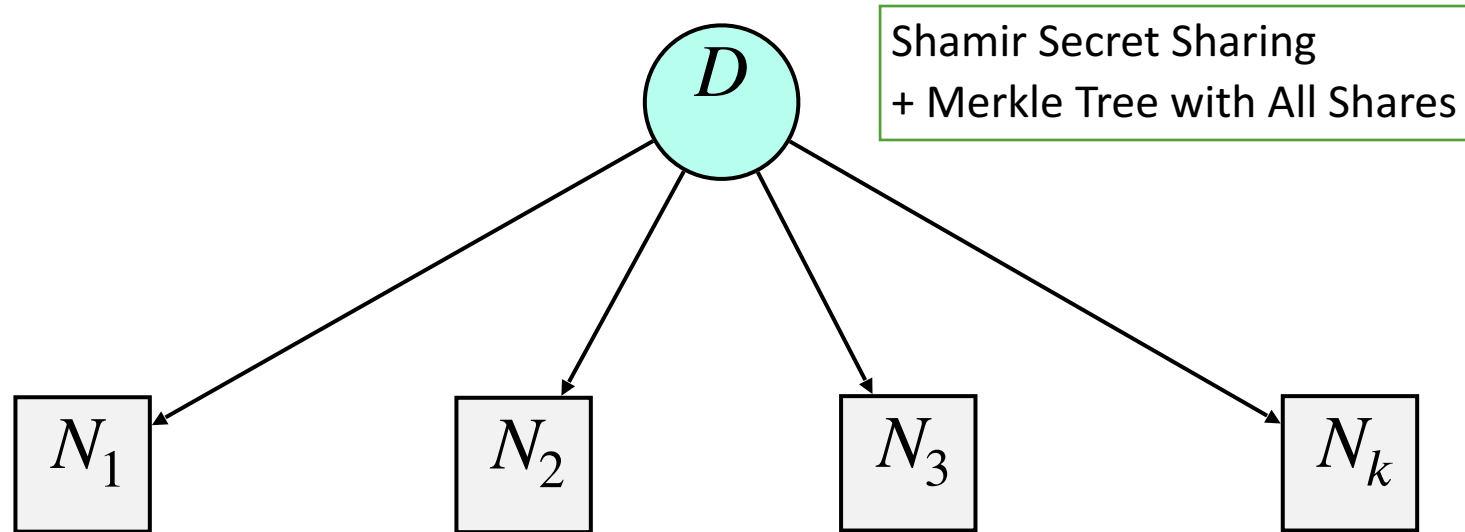
$O(\kappa L n \log(n) + \kappa n^2 \log(n))$ bits for sharing L secrets

Asynchronous weak VSS

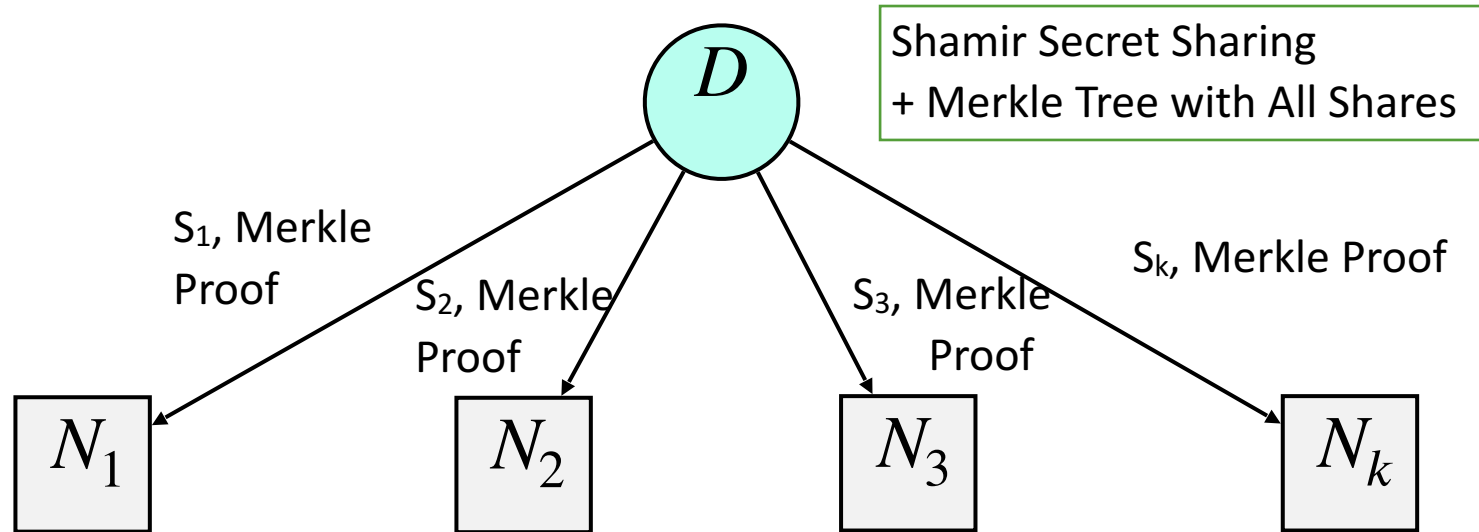
Asynchronous weak VSS



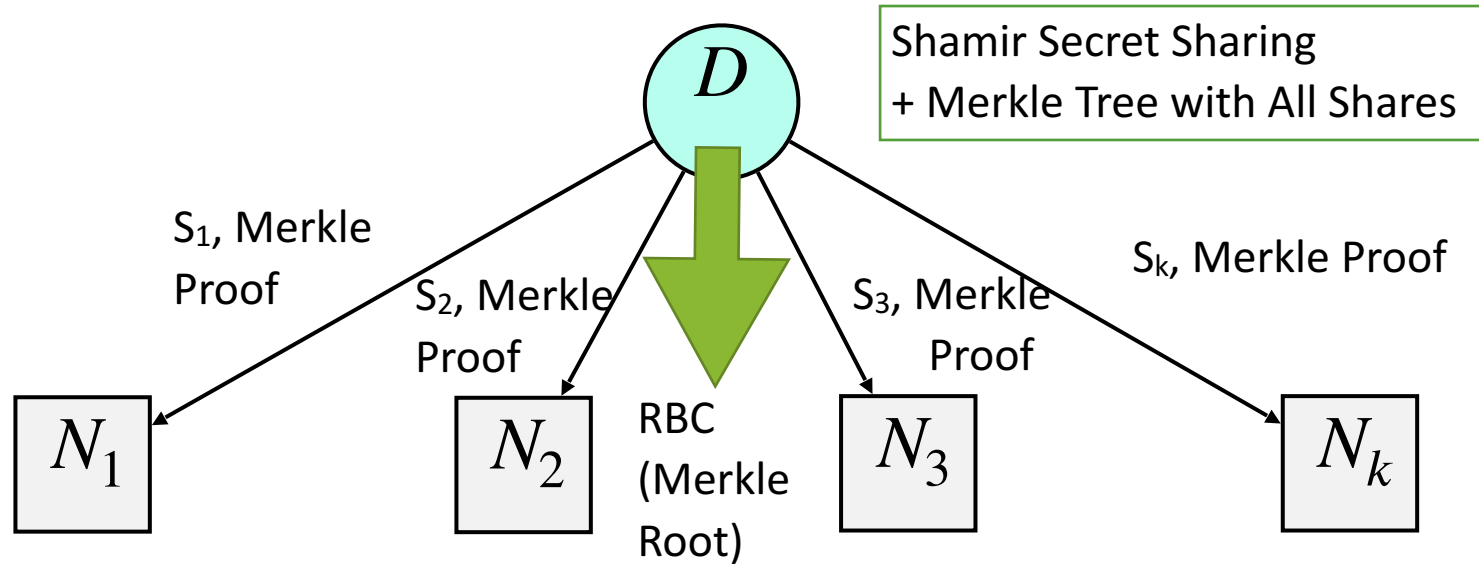
Asynchronous weak VSS



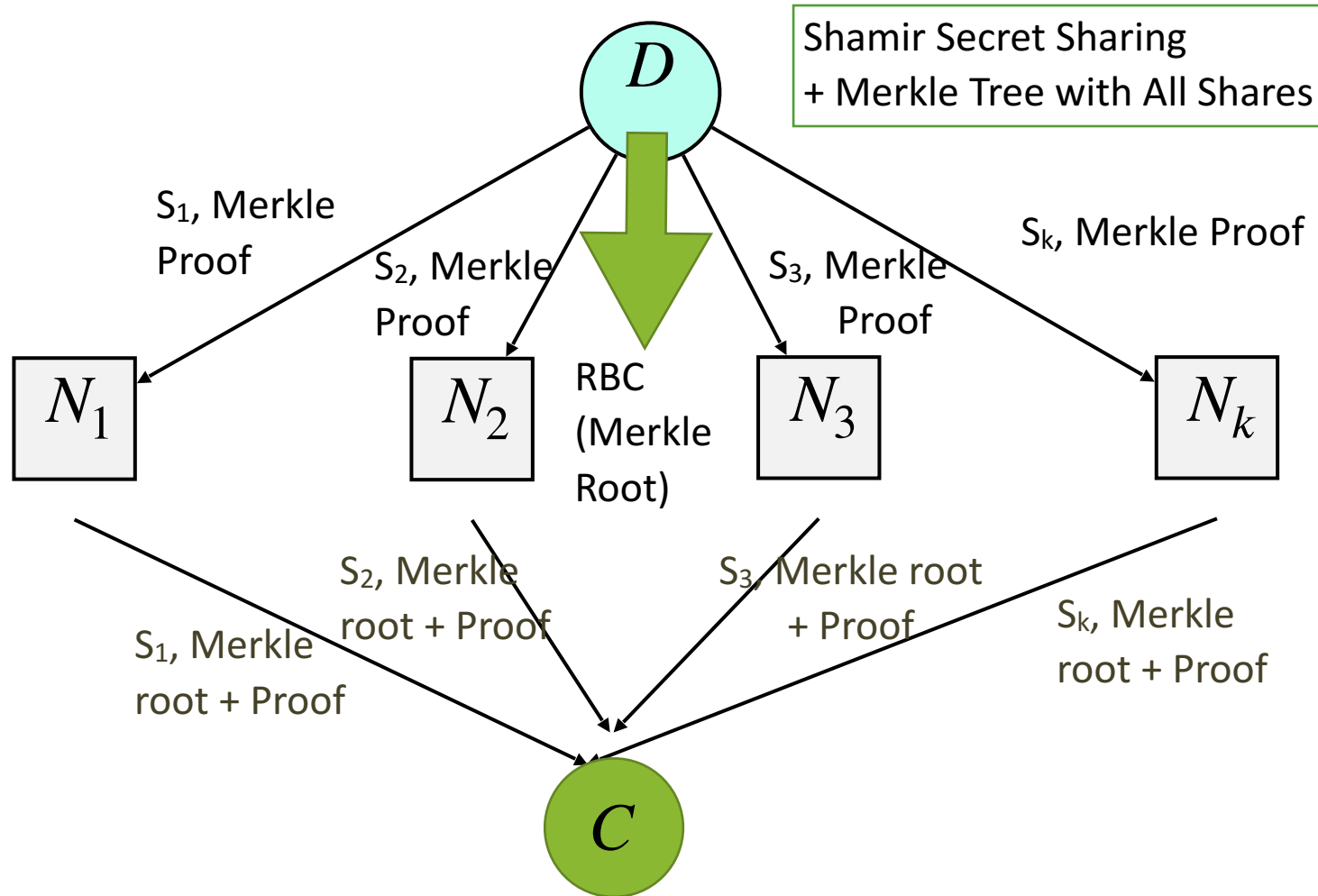
Asynchronous weak VSS



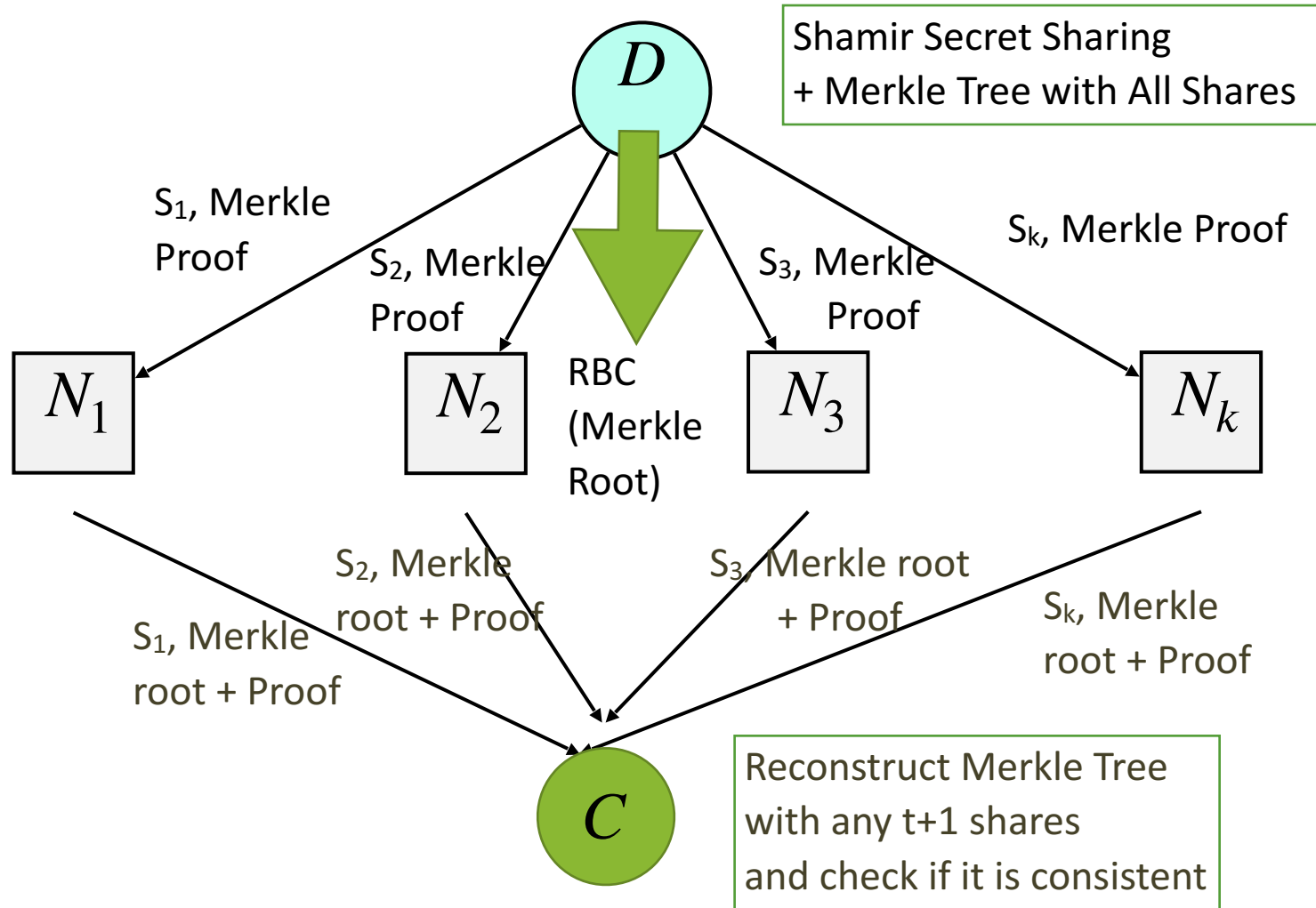
Asynchronous weak VSS



Asynchronous weak VSS

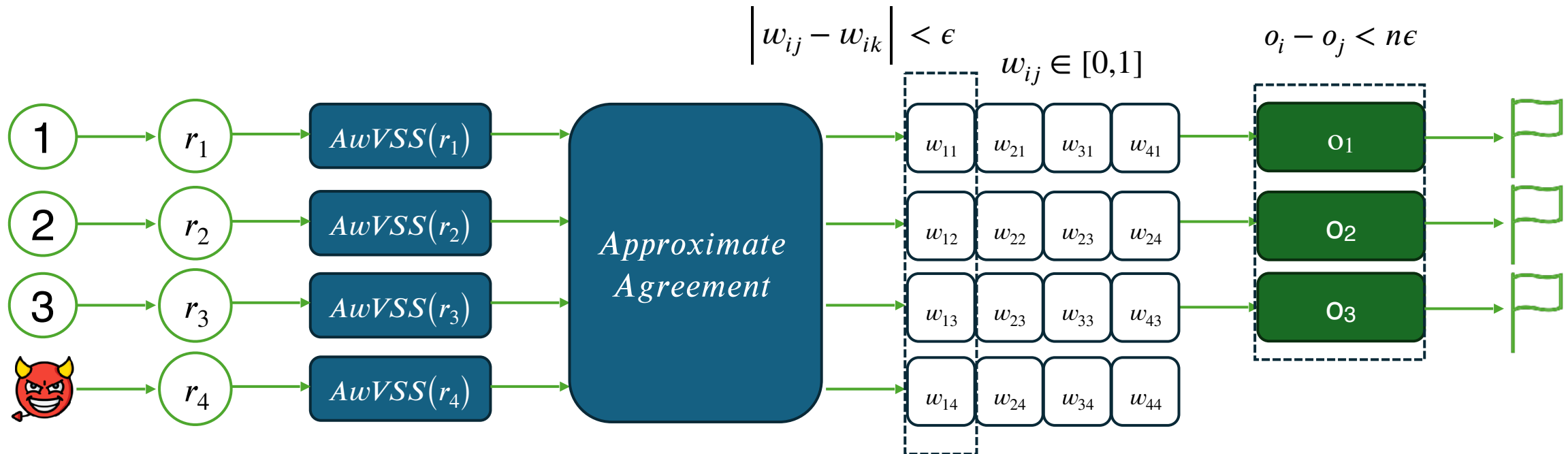


Asynchronous weak VSS



Addressing Beacon-SMR Circularity

- Use **Approximate Agreement** to agree on a weight for each node
- Round off output to a *checkpoint*. Results in *Monte-Carlo* agreement

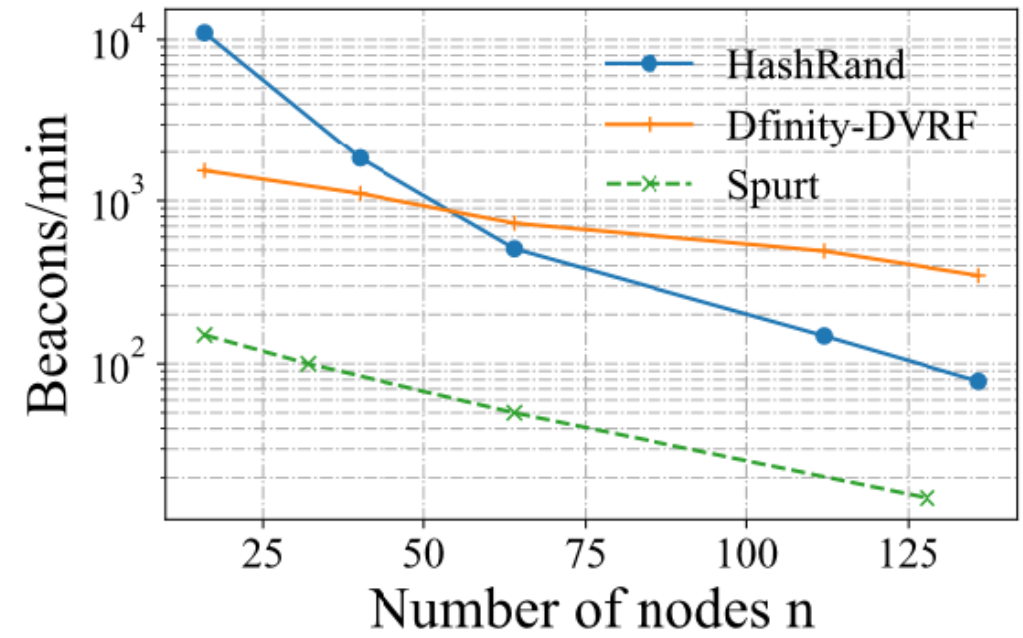


Evaluation

- We compare HashRand with two protocols
 - Spurt (IEEE S&P'22) based on Discrete-Log cryptography without a DKG setup
 - Dfinity-DVRF based on threshold BLS signatures with a DKG setup
- Experimental Setup:
Geo-distributed testbed on AWS, with $n = 136$ *t3a.medium* instances, with 2 CPU cores and 4 GB of RAM

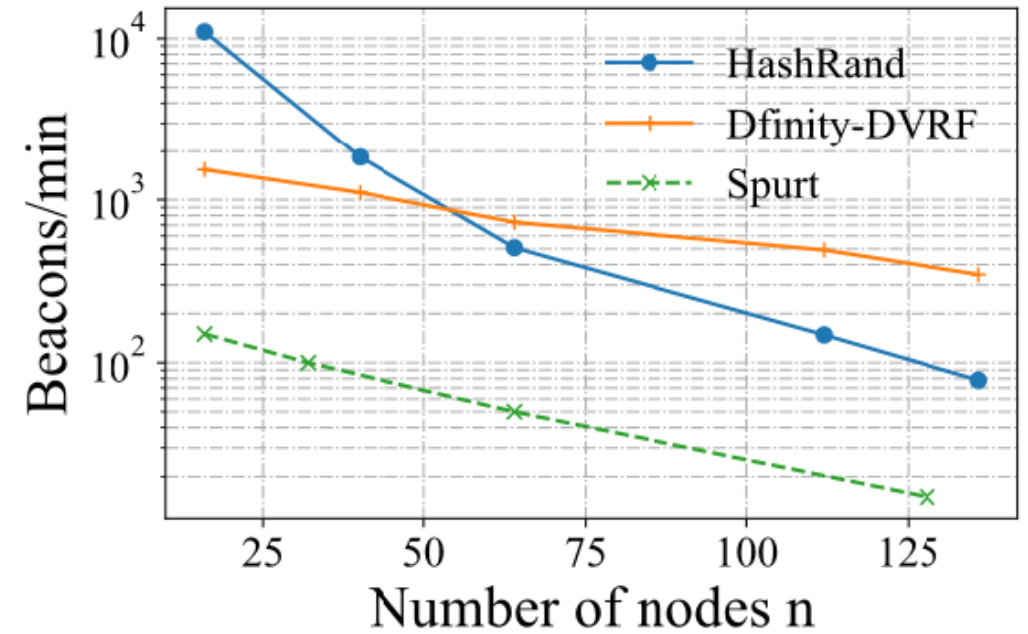
Results

Results



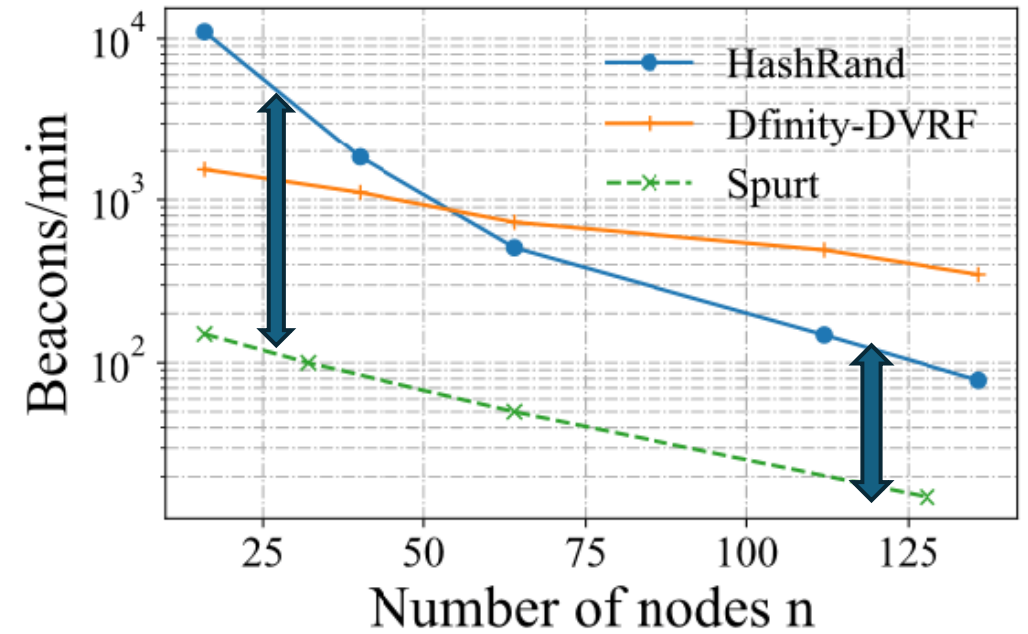
Results

- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency



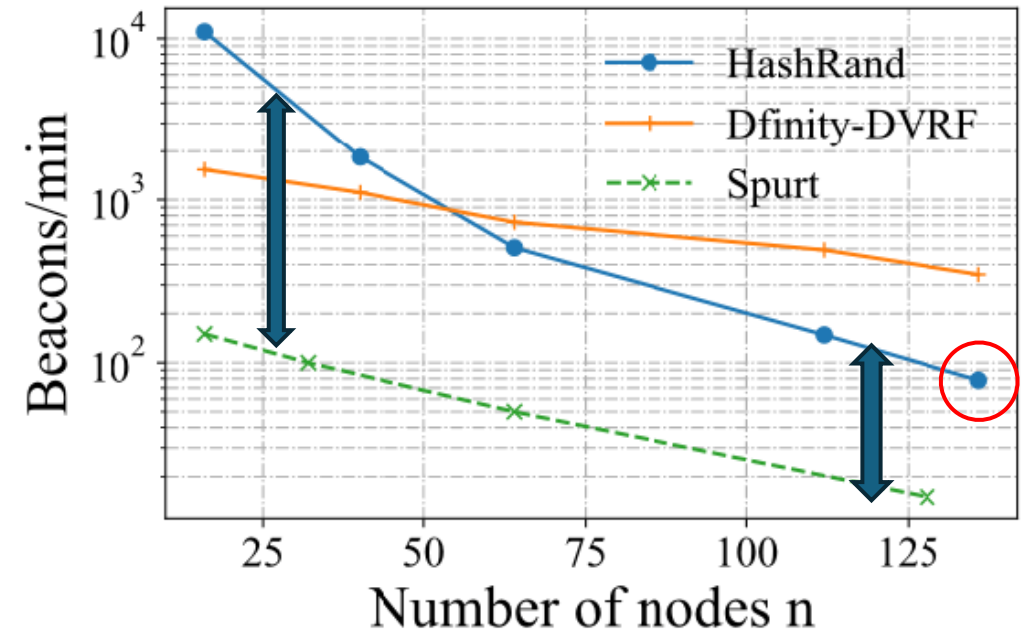
Results

- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency



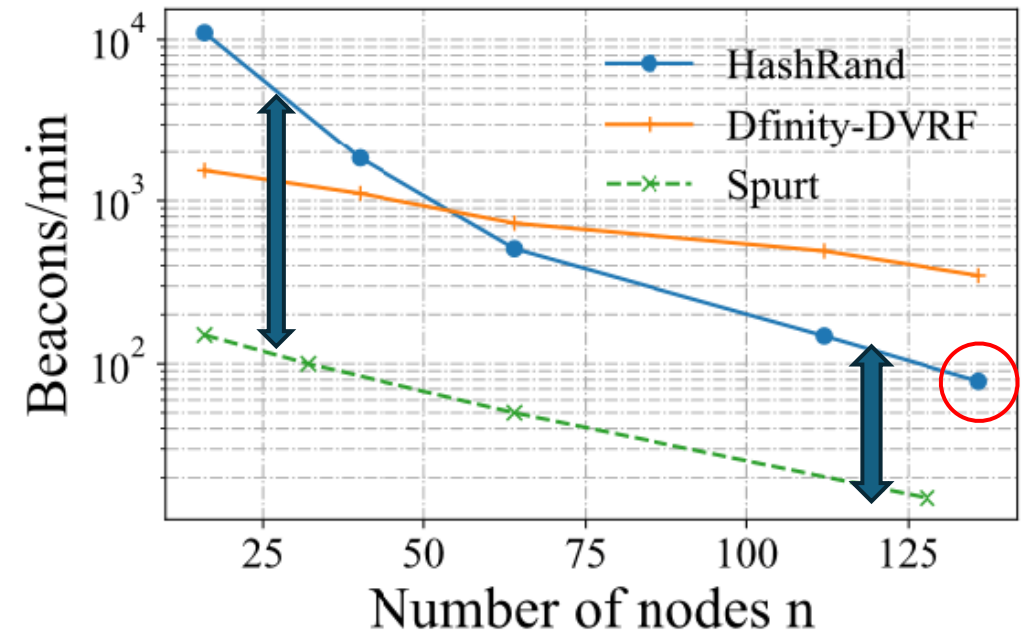
Results

- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency



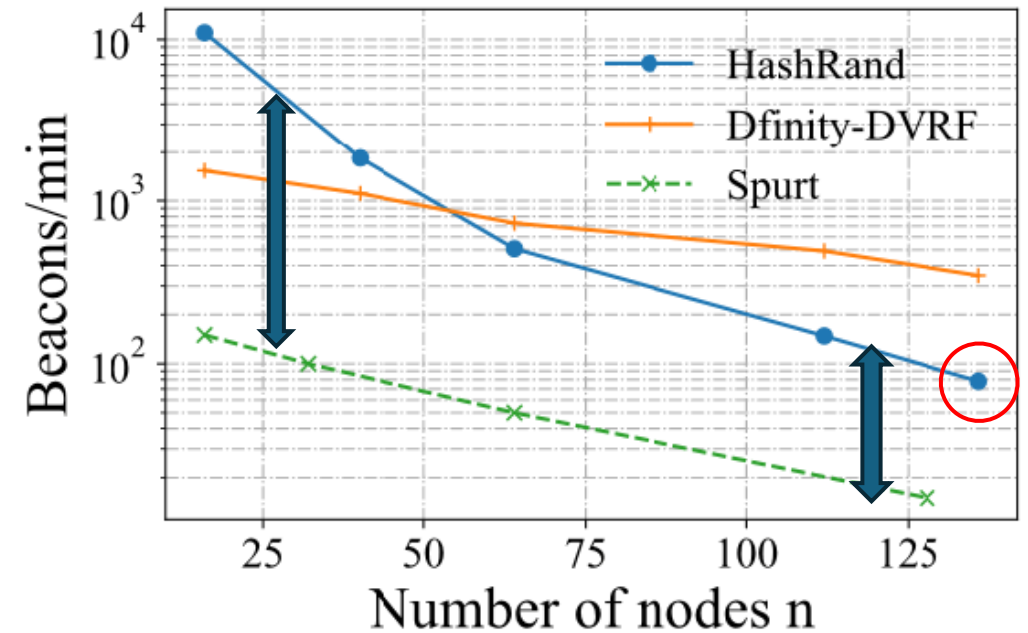
Results

- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency
- HashRand outputs more than 1 beacon per second at $n = 136$



Results

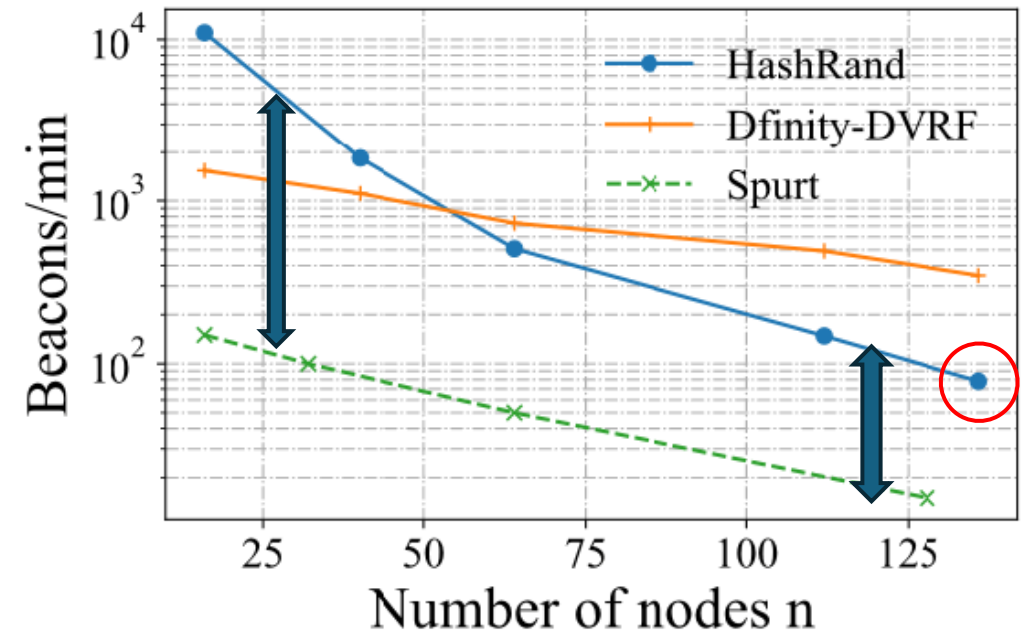
- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency
- HashRand outputs more than 1 beacon per second at $n = 136$



- HashRand is the best way to generate beacons until $n = 50$, even assuming a DKG setup

Results

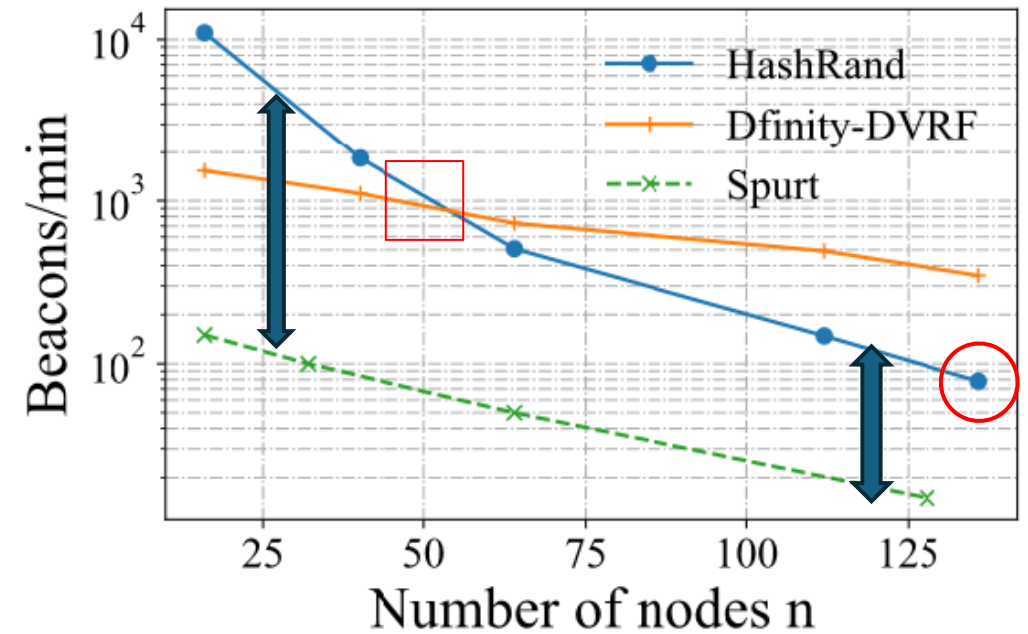
- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency
- HashRand outputs more than 1 beacon per second at $n = 136$



- HashRand is the best way to generate beacons until $n = 50$, even assuming a DKG setup
 - Along with post-quantum security

Results

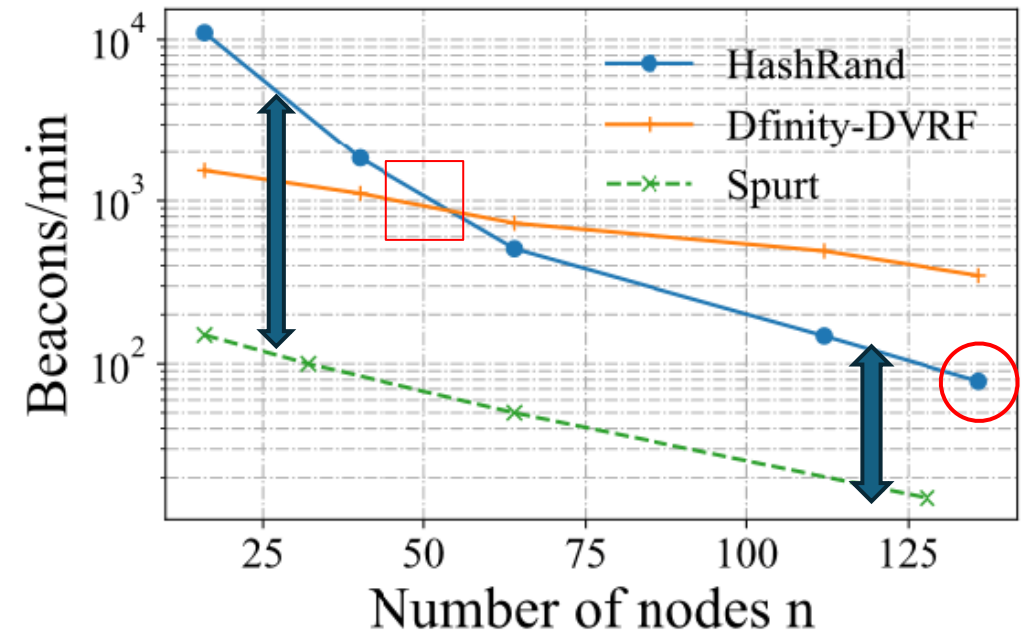
- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency
- HashRand outputs more than 1 beacon per second at $n = 136$



- HashRand is the best way to generate beacons until $n = 50$, even assuming a DKG setup
 - Along with post-quantum security

Results

- HashRand outperforms Spurt by a substantial margin, mainly because of computational efficiency
- HashRand outputs more than 1 beacon per second at $n = 136$
- Performance can further be improved by using Hardware-accelerated Hash functions



- HashRand is the best way to generate beacons until $n = 50$, even assuming a DKG setup
 - Along with post-quantum security

Asynchronous MPC (AMPC)

- Unconditional-secure AMPC
 - Arithmetic-circuit representation
 - Communication of $O(nC + n^{14})$ field elements for a circuit with C multiplication gates [GLS, Crypto'24]
- Can we build better AMPC with lightweight cryptography?

Concretely Efficient AMPC from Lightweight Cryptography

- AMPC protocol achieving GOD [Crypto 2025]
 - Communication: $O(Cn + n^4)$ field elements
 - Computation: $O(nC)$ hashes

Joint work with Akhil Bandrupalli, Xiaoyu Ji,
Chen-Da Liu-Zhang, Daniel Poellmann, and Yifan Song

Concretely Efficient AMPC from Lightweight Cryptography

- AMPC protocol achieving GOD [Crypto 2025]
 - Communication: $O(Cn + n^4)$ field elements
 - Computation: $O(nC)$ hashes
- Velox: AMPC protocol achieving fairness [ACM CCS 2025]
 - Communication: $O(Cn+n^3)$ field elements
 - Computation: $O(nC)$ hashes

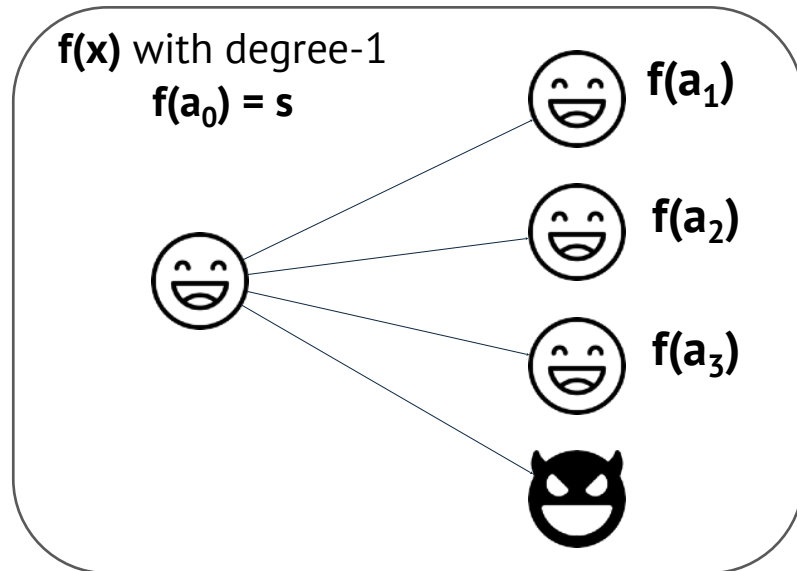
Joint work with Akhil Bandrupalli, Xiaoyu Ji,
Chen-Da Liu-Zhang, Daniel Poellmann, and Yifan Song

Asynchronous Complete Secret Sharing with Abort

- ACSS-Ab: When the dealer is malicious, at the end of sharing, up to t honest parties can output \perp
- Requires $O(nL + n^2)$ communication and $O(nL)$ lightweight cryptographic operations for L secrets
 - Standard ACSS requires $O(nL)$ public-key cryptography operations
- Build Asynchronous MPC with Fairness, where parties abort after identifying malicious behavior

Asynchronous Complete Secret Sharing with Abort

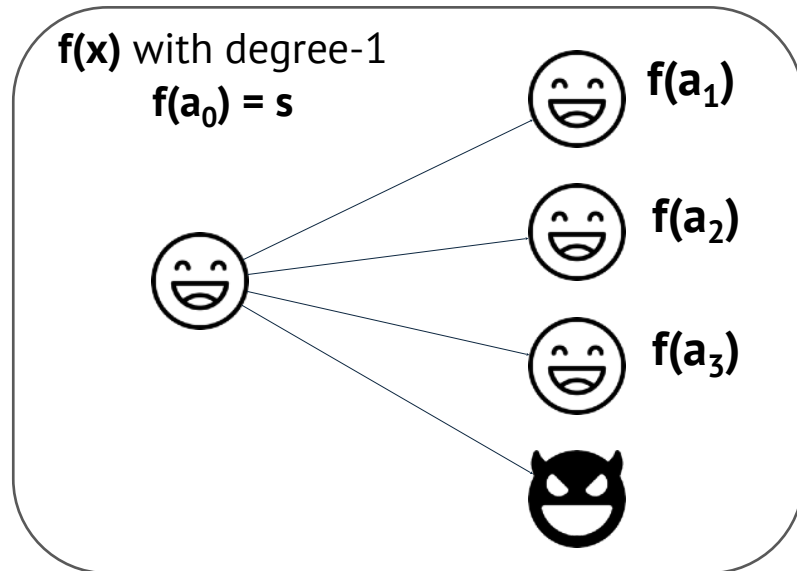
- ACSS-Ab: When the dealer is malicious, at the end of sharing, up to t honest parties can output \perp
- Requires $O(nL + n^2)$ communication and $O(nL)$ lightweight cryptographic operations for L secrets
 - Standard ACSS requires $O(nL)$ public-key cryptography operations
- Build Asynchronous MPC with Fairness, where parties abort after identifying malicious behavior



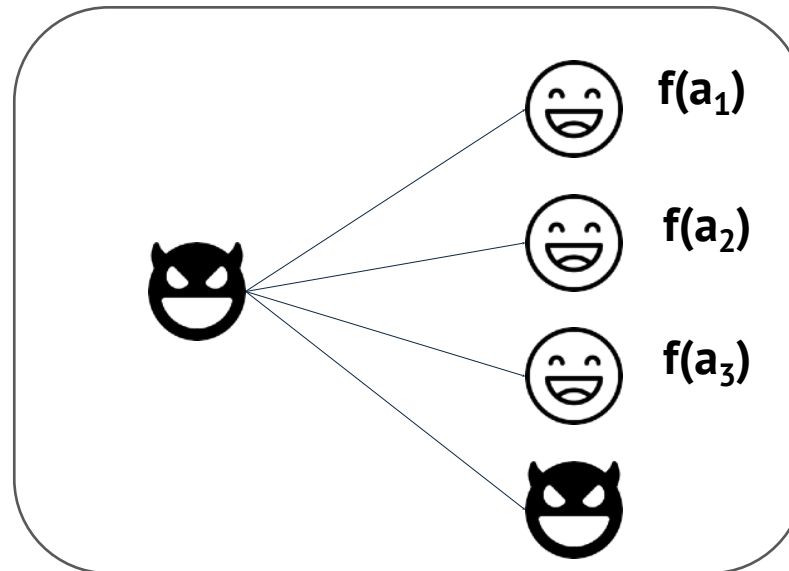
ACSS and ACSS-Ab with an Honest Dealer

Asynchronous Complete Secret Sharing with Abort

- ACSS-Ab: When the dealer is malicious, at the end of sharing, up to t honest parties can output \perp
- Requires $O(nL + n^2)$ communication and $O(nL)$ lightweight cryptographic operations for L secrets
 - Standard ACSS requires $O(nL)$ public-key cryptography operations
- Build Asynchronous MPC with Fairness, where parties abort after identifying malicious behavior



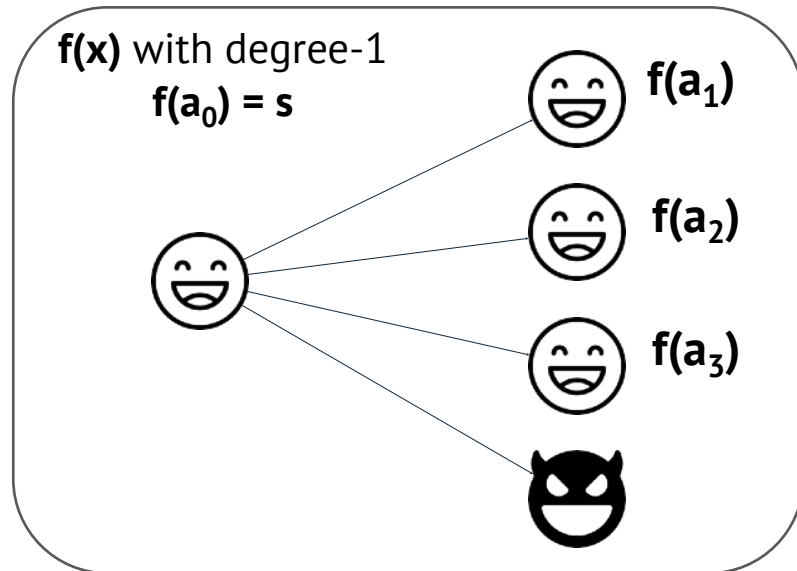
ACSS and ACSS-Ab with an Honest Dealer



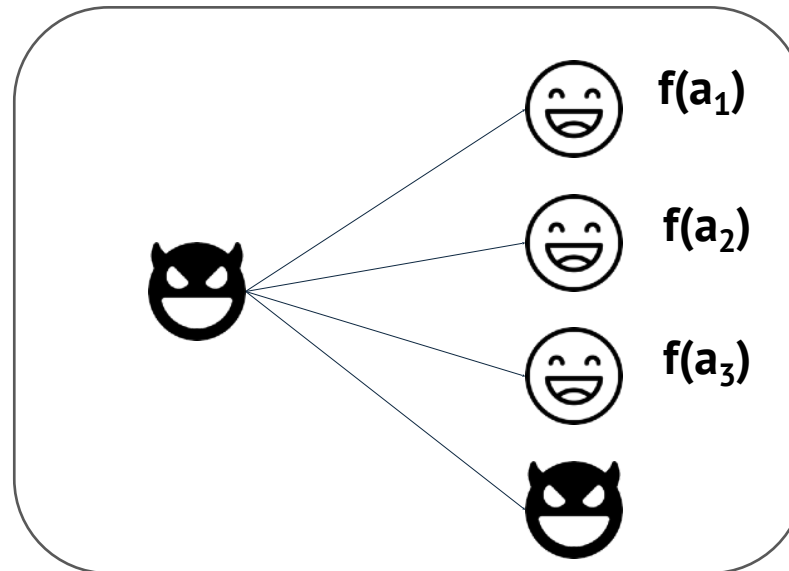
ACSS with a Malicious Dealer

Asynchronous Complete Secret Sharing with Abort

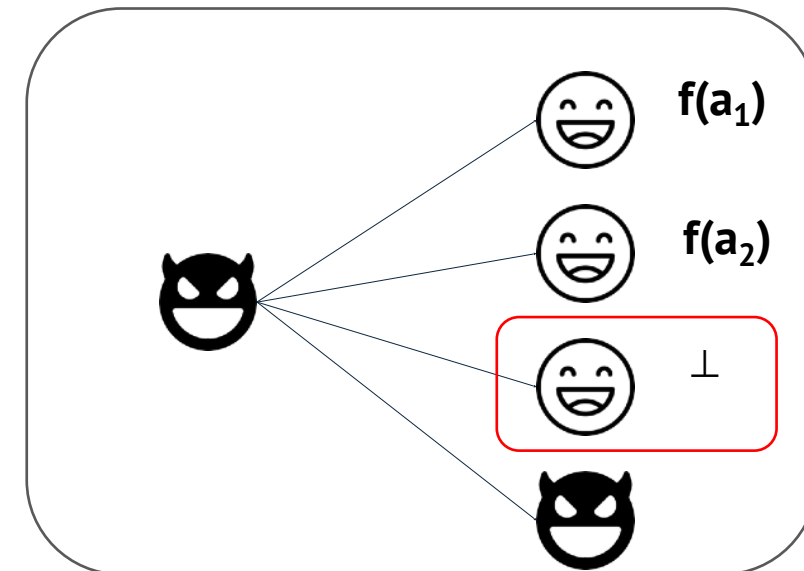
- ACSS-Ab: When the dealer is malicious, at the end of sharing, up to t honest parties can output \perp
- Requires $O(nL + \kappa n^2)$ communication and $O(nL)$ lightweight cryptographic operations for L secrets
 - Standard ACSS requires $O(nL)$ public-key cryptography operations
- Build Asynchronous MPC with Fairness, where parties abort after identifying malicious behavior



ACSS and ACSS-Ab with an Honest Dealer



ACSS with a Malicious Dealer

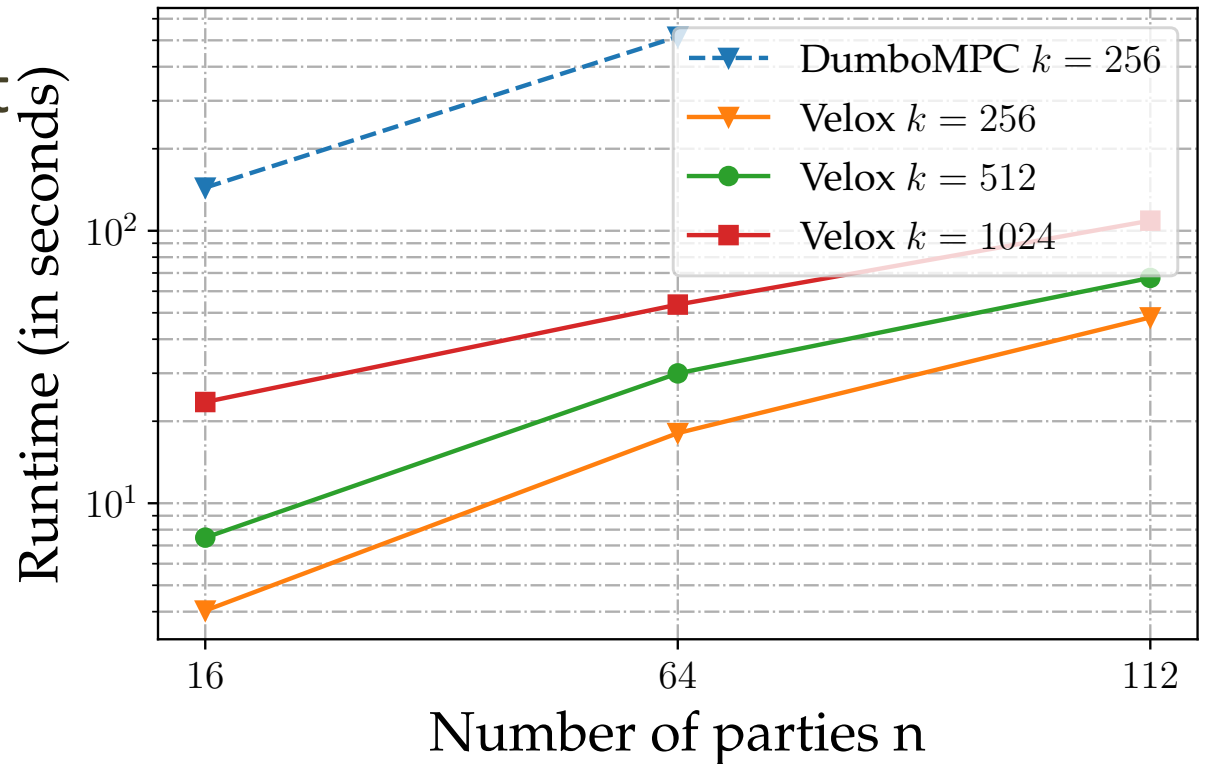


ACSS-Ab with a Malicious Dealer

Performance Analysis



- Comparison with DumboMPC [USENIX Sec'25]
- Application: Anonymous Broadcast of size k
- Setup: c5.large instances with 2 CPU cores and 4GB RAM
- Two orders of magnitude improvement!



Progress and Impact

- Lightweight Cryptography for Threshold crypto is a topic of immense recent interest
- More Examples:
 - Verifiable Secret Sharing [JoCrypto 2024, PiVer]
 - Agreement on Common Subset [Das et al, CCS 2024]
 - Asynchronous MPC [Momose, Eprint 2024/1717]
 - Dynamic Proactive Secret Sharing
 - [Eprint 2025/1631]
 - [Xing et al., Eprint 2025/1516]

Take Away

- Modern computing infrastructures offer high-bandwidth, low-latency communication capabilities
- We should consider communication/round overhead as our key adversary to conquer, and give a free hand to computation overhead
- Lightweight cryptography offers an apt tradeoff between computation and communication overhead, and we should exploit
- We discuss some illustrative examples, a distributed random beacon, and MPC with orders-of-magnitude performance advantages