Building Threshold Cryptosystems over a SMR/Blockchain channel

Aniket Kate

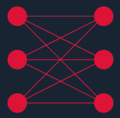
Purdue University / Supra Research

NIST Workshop on Multi-Party Threshold Schemes (MPTS) 2023
Presented on September 28, 2023

Threshold Cryptosystem with Honest Majorities

Properties: Robustness, Guaranteed Output
Delivery, Fairness, ...

Typical Communication Model



Synchronous Point-to-point Secure Links

Broadcast Channel

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

Based on Networking Research

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

Communication Model for the Internet

System Setting

- n parties and an f-limited adversary
- point-to-point links

Asynchrony

- For any message sent, the adversary can delay its delivery by any finite amount of time.
- there is no bound on the time to deliver a message but,
- each message must eventually be delivered.

Partial Synchrony

- **Assumption: There exists known finite time bound Δ and a special event GST (Global Stabilization Time).**
- **The adversary must cause the GST event to eventually happen after some unknown finite time.**
- **Any message sent at time x must be delivered by time Δ+max(x,GST).**

Byzantine Broadcast

Problem Setting

- n parties and an f-limited adversary
- A distinguished broadcaster p

With bounded synchrony

- **Agreement.**
  If two honest parties commit values v and v' respectively, then $v = v'$.
- **Validity.**
  If the broadcaster is honest, then all honest parties commit the broadcaster's value.
- **Termination.**
  All honest parties commit and terminate.

With partial synchrony

- Agreement.
  same as above.
- Validity.
  If the broadcaster is honest and GST = 0, then all honest replicas commit the broadcaster's value.
- Termination.
  All honest replicas commit and terminate after GST.

Threshold Cryptosystem beyond Synchrony

Lower Bound

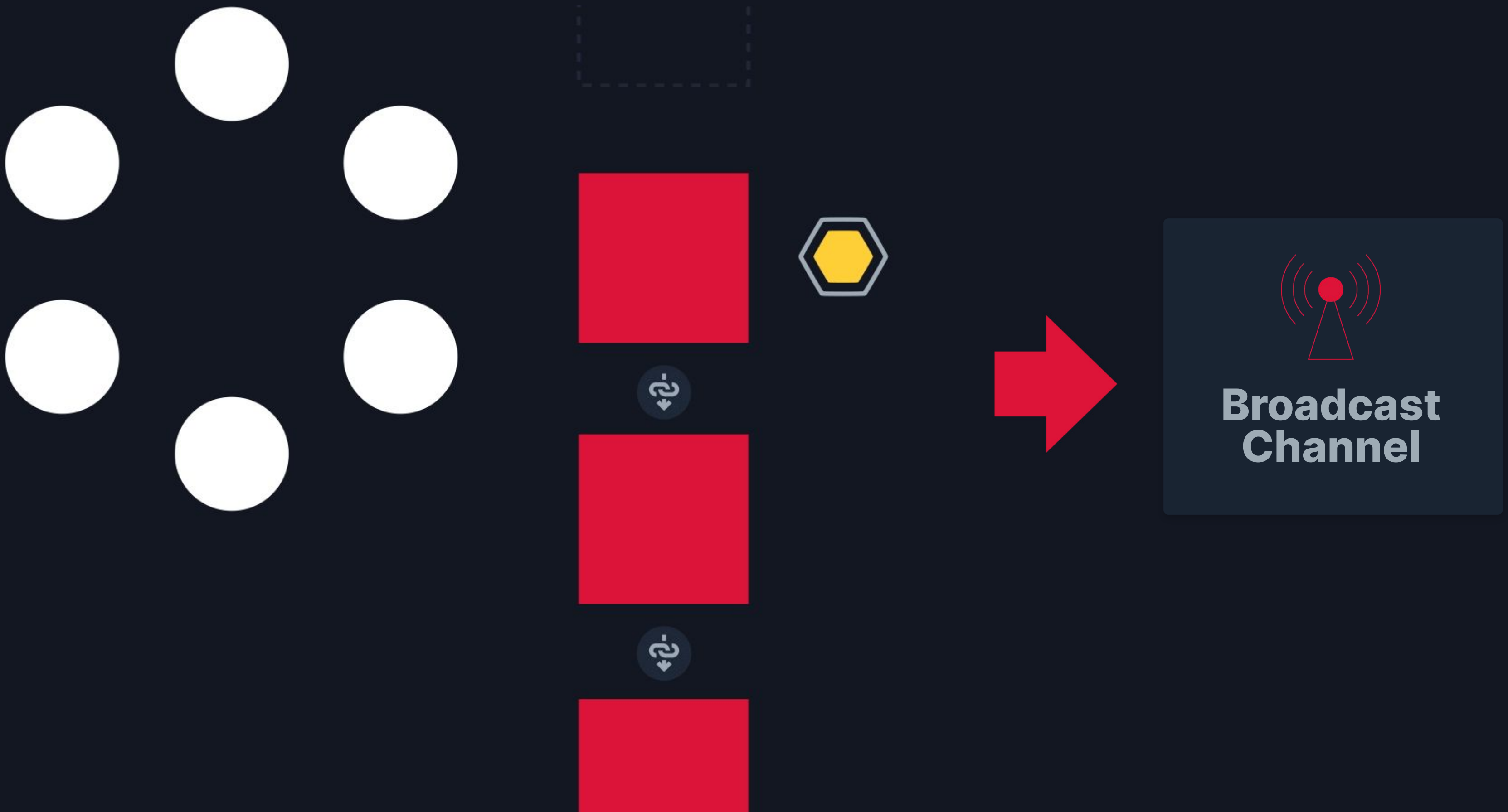
2/3

Super Honest Majority required

Increased complexity of development

- dealing with the asynchronous network with common coins
- dealing with timeouts, view-change, responsiveness in partial synchrony

Existing Threshold Crypto based Blockchain



State Machine Replication (SMR)

output: a transactions log = $[tx_0, tx_1, \ldots, tx_j]$

Safety:
If $[tx_0, tx_1, \ldots, tx_j]$ and $[tx'_0, tx'_1, \ldots, tx'_{j'}]$ are output by two honest nodes, then $tx_i = tx'_i$ for all $i \leq \min(j, j')$.

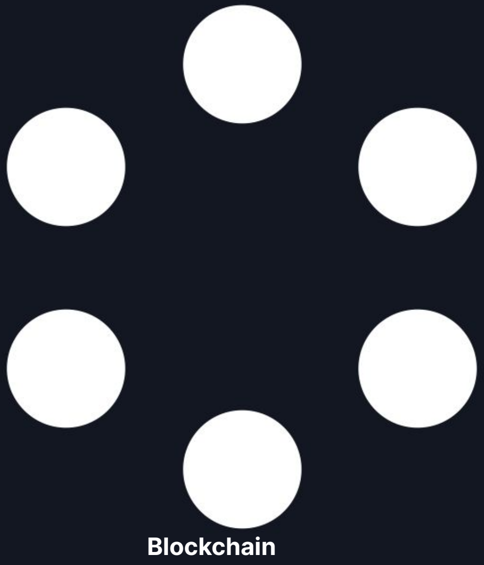
Liveness:
If a transaction tx is input to at least an honest node, then every honest replica eventually outputs a log containing tx.

Informally,

(i) Senders' messages appear on the blockchain eventually.

(ii) Different receivers observe messages at different points in time.

(iii) However, all the nodes eventually observe messages in the exact same total order.



State Machine Replication / Blockchain

Byzantine Broadcast

Problem Setting

- n parties and an f-limited adversary
- A distinguished broadcaster p

Agreement:
If two honest parties commit values v and v' respectively, then $v = v'$.

Validity:
If the broadcaster is honest, then all honest parties commit the broadcaster's value.

Termination:
All honest parties commit and terminate.

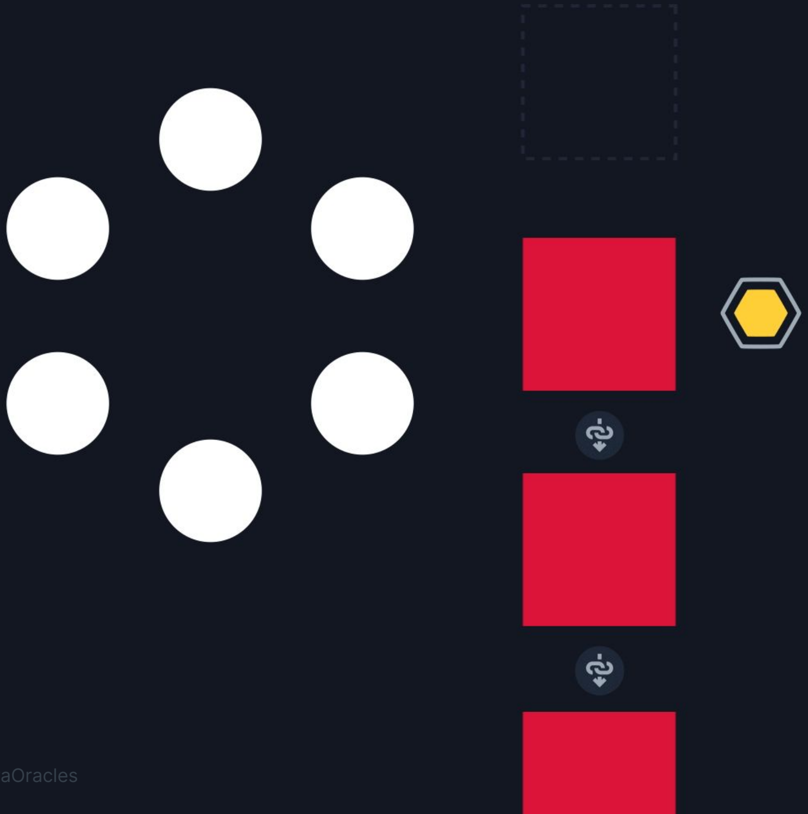
SMRs are not Broadcast Channels

	Employed Broadcast Channel	Efficient SMR / Blockchain
Message Delay	A fixed Δ	Only eventual guarantee
Receivers View	All messages by the end of the round	Only a prefix-order guarantee

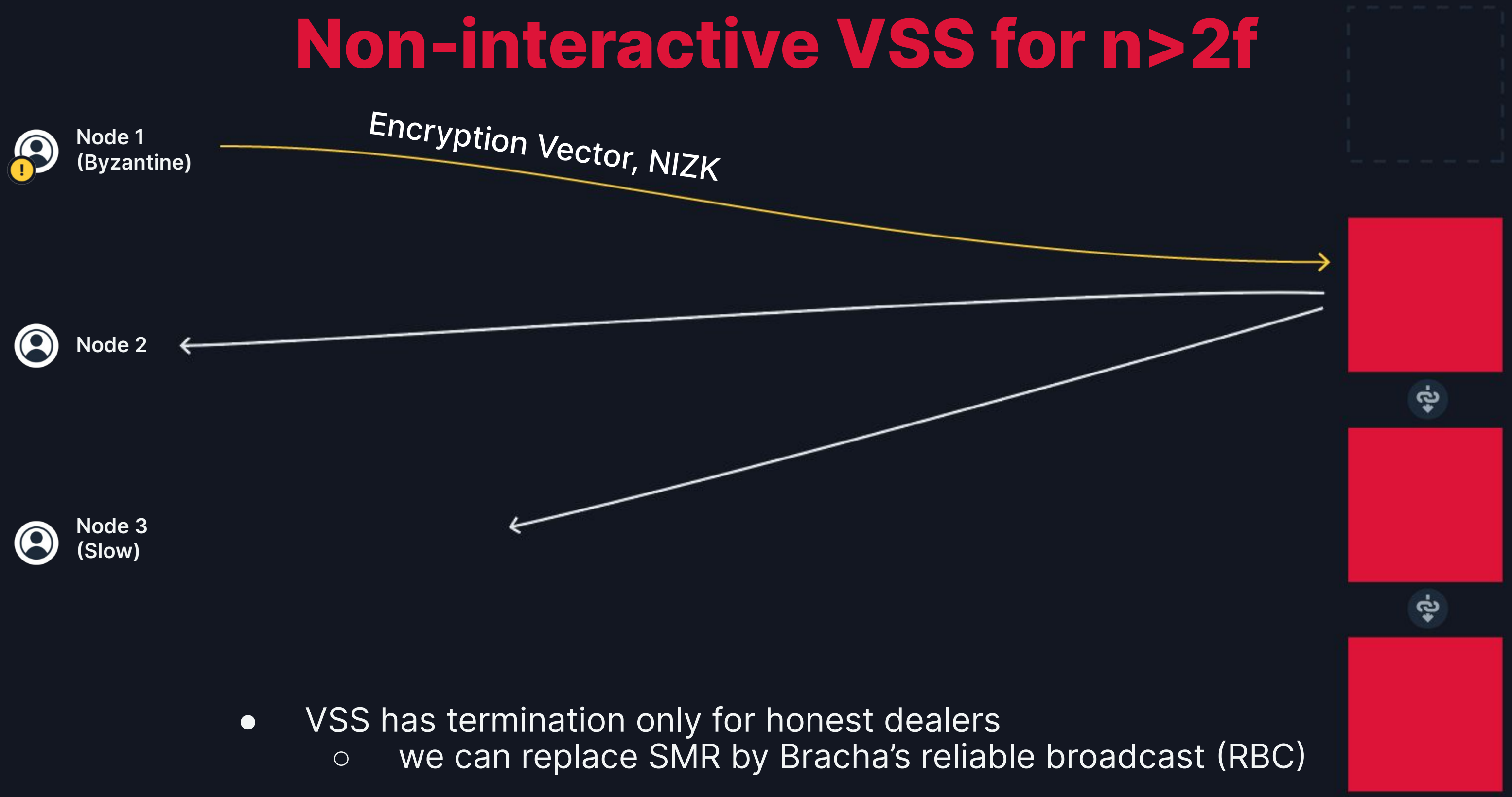
A Way Out

SMR-assisted Protocol Design

Making broadcast-based primitives to work in the environment with SMR



Non-interactive VSS for n>2f



- VSS has termination only for honest dealers
  - we can replace SMR by Bracha's reliable broadcast (RBC)
- Reasonable computational efficiency
- NI-VSS can only offer computational hiding property

Verifiable Secret Sharing (VSS)

VSS with Broadcast Channel

51%

Honest Nodes required

Even for unconditional hiding property

Asynchronous VSS

67%

Honest Nodes required

The bound holds for different AVSS versions

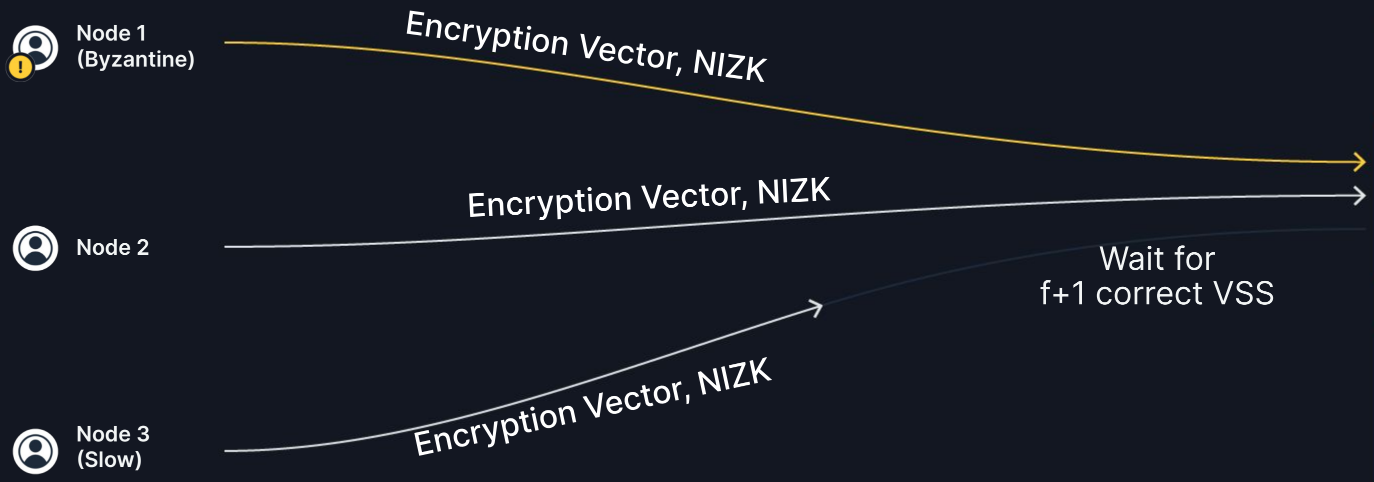
SMR/RBC-Assisted VSS

51%

Honest Nodes required

Unconditional hiding is *not* achievable

Building DKG for n>2f



- Agreement of common subset (ACS) comes free with SMR.
- SMR determines the network assumption for DKG.
- Robustness property [Jon Katz's talk] is possible with an additional asynchronous round.

Distributed Key Generation (DKG)

Partially Synchronous / Async DKG

67%

Honest Nodes required

Calculation of a shared public key for a random, unknown private key

SMR-assisted DKG

51%

Honest Nodes required

The protocol with a broadcast channel and a SMR channel are almost equivalent

Threshold Cryptography with SMR Channels

Threshold BLS Signatures / VRF

Purely asynchronous communication process over point-to-point links

Threshold ECDSA / EdDSA

We will need secure multi-party computation (MPC) capability

MPC over SMR Channels for n>2f

CDN MPC Framework [EUROCRYPT '01] based On Threshold Linearly Homomorphic Encryption

MPC over SMR Channels for n>2f

Threshold Linear Homomorphic Encryption Setup

- public key pk
- secret key is shared among the parties

Example Setups

Paillier Encryption, Class-group Encryption, Exponentiated ElGamal Encryption

Input (m) Processing:

- Compute ciphertext c=Enc(pk,m)

Secure Scalar Operations: (local or on SMR)

Given Enc(a) and Enc(b)

- Compute Enc(a). Enc(b)^x = Enc(a+bx)

Secure Multiplication:

Given Enc(a) and Enc(b)

- Publish Enc(d_i) and Enc(b.d_i)
  (wait for f+1 tuples)
- Compute and threshold decrypt Enc(a+ Σd_i)
- Publish decrypted share & compute (a+ Σd_i)
- Compute secure product as
  (a+ Σd_i).E(b) - ΣEnc(b.d_i) = E(a.b)

Conclusion and Unresolved Issues

Proposed VSS/DKG/MPC with SMR Channel

51%

Honest Nodes required

Computational Hiding
Key Overhead: Encryption + NIZK

Achieving Unconditional Hiding

67%

Honest Nodes required

Converting Feldman/Pedersen VSS to work on SMR

Several Open Problems

1. Proving lower-bound n>3f for unconditional hiding (i.e. using secure and authenticated channels)
2. MPC for n>3f with unconditional hiding towards avoiding encryptions and NIZK

...

Thanks!

Twitter/X: https://twitter.com/aniketpkate
Email: aniket@purdue.edu