
# Signet

## Scalable Network-Driven Proof of Notification for Blockchain Systems

**Elham Ehsani** · Marc Wyss · Jonghoon Kwon

Marc Frei · Yih-Chun Hu · Adrian Perrig · Alberto Sonnino

Mysten Labs

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

ICDCS 2026

# Motivation: Blockchain Notification Ecosystem

**Polling (Traditional)**

- ✗ Light client repeatedly queries full node
- ✗ Burdens full nodes with high read volumes
- ✗ Tragedy of the commons: no compensation
- ✗ Misses events between polls

**Push / Subscription (Target)**

- ✓ Full node pushes events when they occur
- ✓ Timely: critical for DeFi & latency-sensitive apps
- ✓ Paid subscription
- ✓ Requires verifiable proof of notification

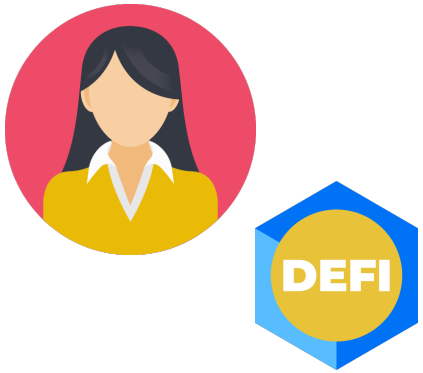
| Notifier | Recipient |
| --- | --- |
| Full blockchain node | Light client / wallet |

# Motivation: use case

High-stakes, ultra-low-latency
- a dedicated full node

mid-value, low-latency
- Signet

Low-value, less latency-sensitive
- Occasional polling



# Design Goals

**For a practical proof-of-notification protocol:**

- **Accountability:** verifiable, unforgeable, and non-repudiable cryptographic proofs.
- **High Throughput:** gigabit-scale workloads.
- **Low Latency:** negligible latency (µs) to the critical path.
- **High Scalability:** Scale with participants.
- **Compatibility and incremental deployment**.

# Key Insight: Network as the witness

- Autonomous Systems (ASes) are already trusted with forwarding.
- Leverage ASes as impartial witnesses.
- Witnesses are on the forwarding path: no extra round-trips, no mediated delivery

**However,**

- Not every AS is trusted by both parties; how to route via the mutually trusted ASes?
- How to generate proofs at line rate without becoming a bottleneck?

# Key Building Blocks

**Path-Aware Networking, e.g., SCION**

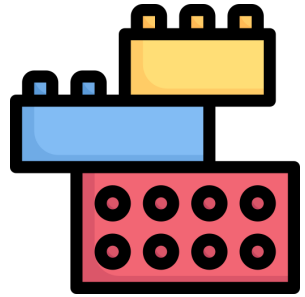
Select paths through mutually trusted ASes.

**Broadcast Authentication, e.g., TESLA**

Symmetric MACs: lightweight, no public-key bottleneck.

**On-chain Anchoring**

Proofs committed before key release: non-repudiation.



# Signet Components

## Notifier

**Full blockchain node**

- Monitors on-chain events
- Locks deposit on-chain
- Sends push notifications

## Witness AS

**Border router on path**

- Observes notifications
- Returns proof to notifier

## Recipient

**Light client / wallet**

- Pays subscription fee
- Initiates challenges

## Smart Contract

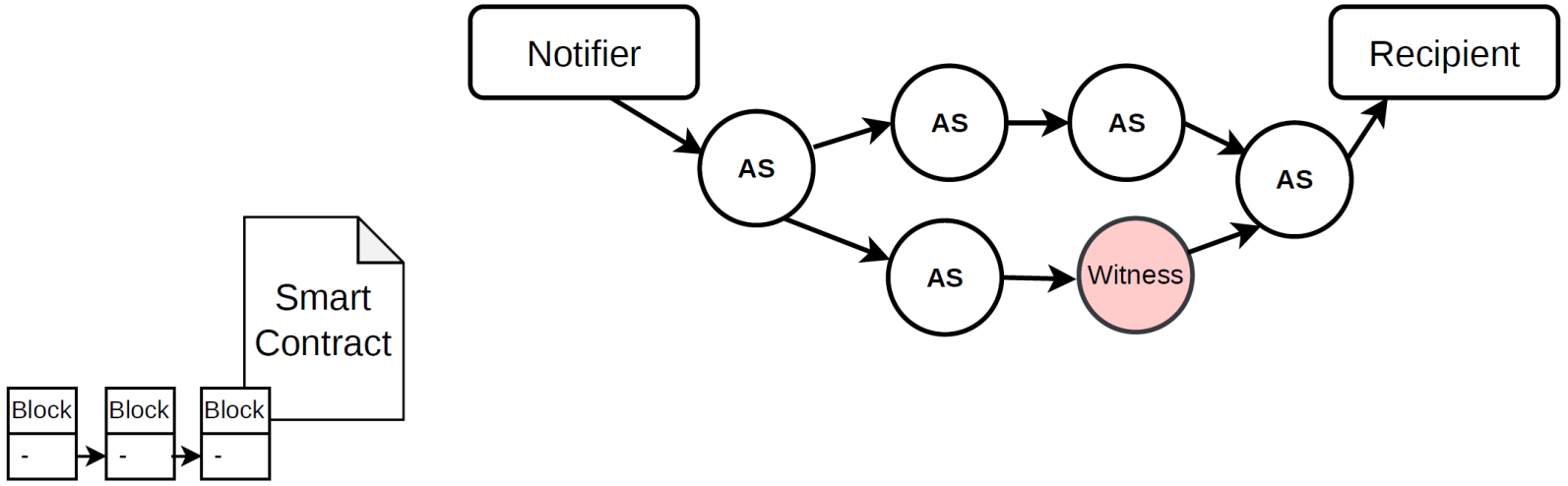
**On-chain**

- Stores SLAs & deposits
- Resolves disputes

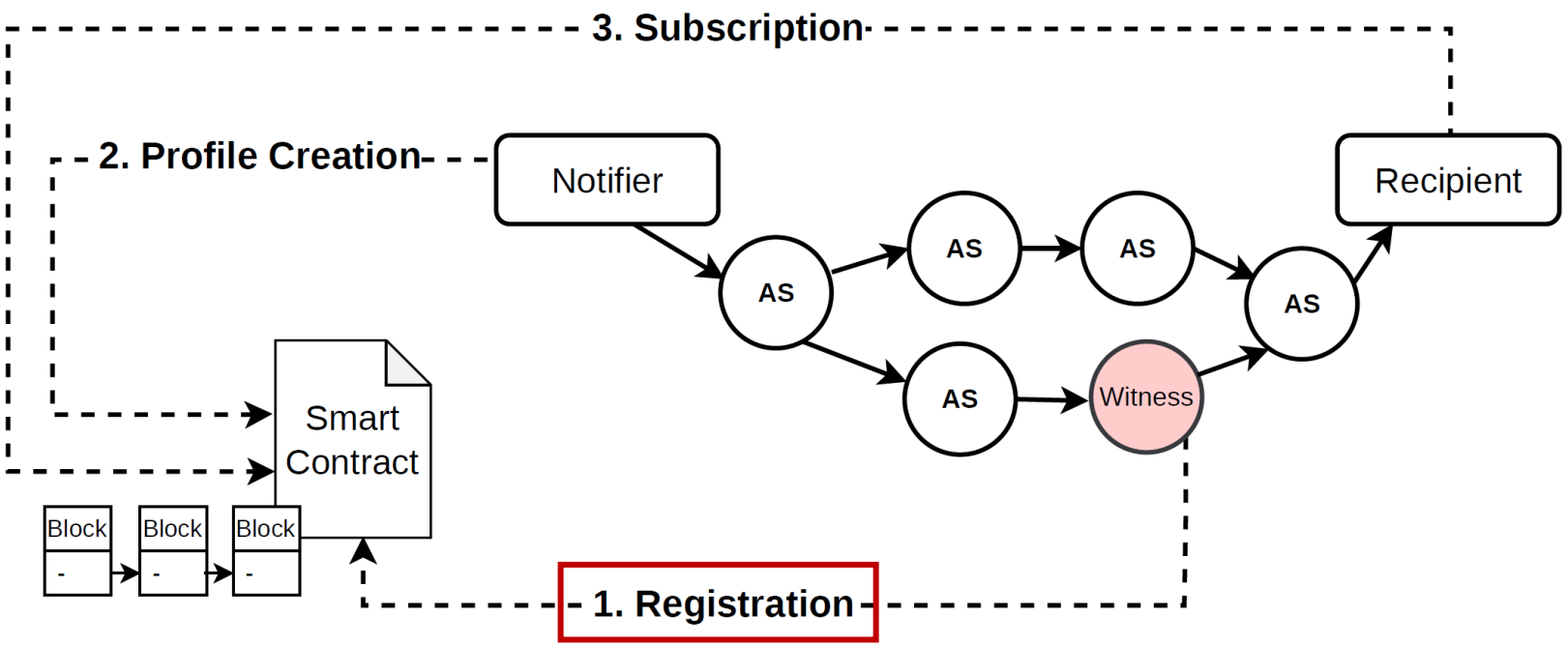
# TESLA: Efficient Broadcast Authentication

- Time is divided into intervals: each has a secret key $K_i$, disclosed only after a delay.
- Delayed disclosure = delayed verification, but lightweight and fast.
- Limitation: no inherent non-repudiation: anyone can forge MACs after key disclosure.
- Fix: commit messages on-chain before key disclosure.

# Protocol Flow

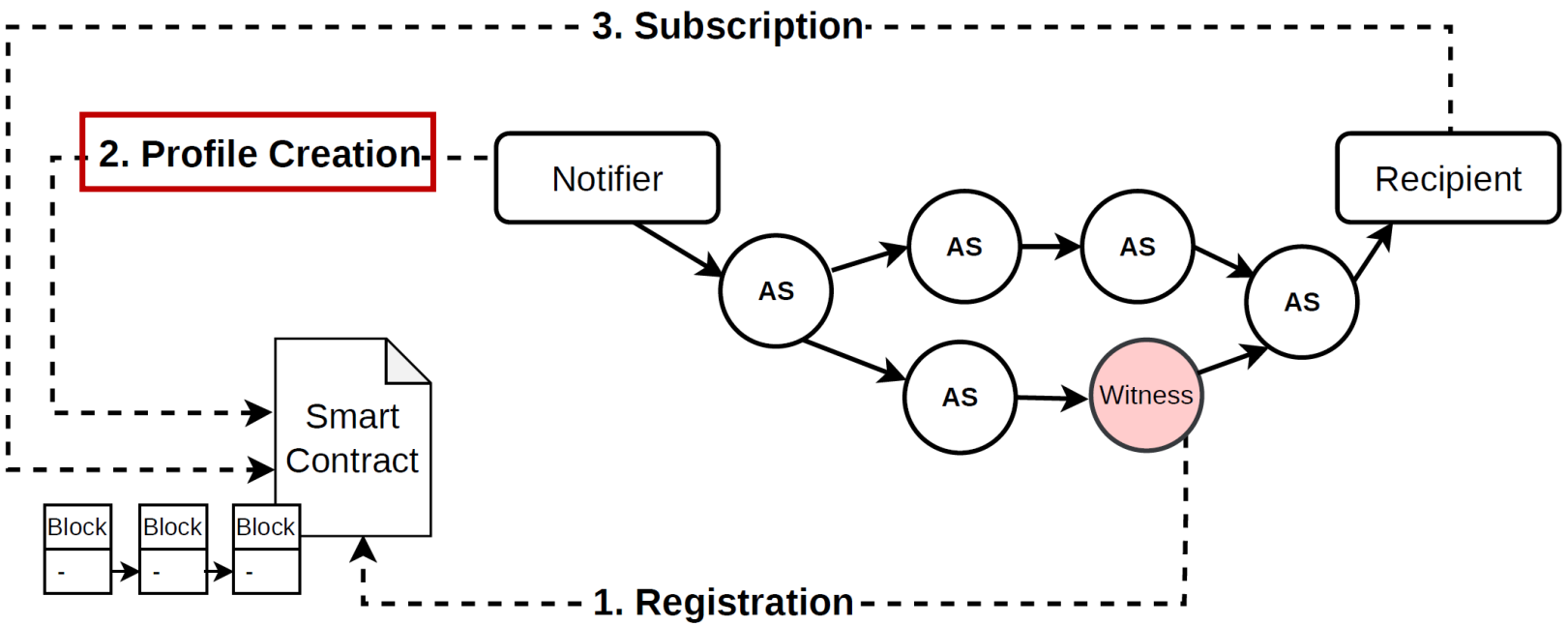


# Protocol Flow



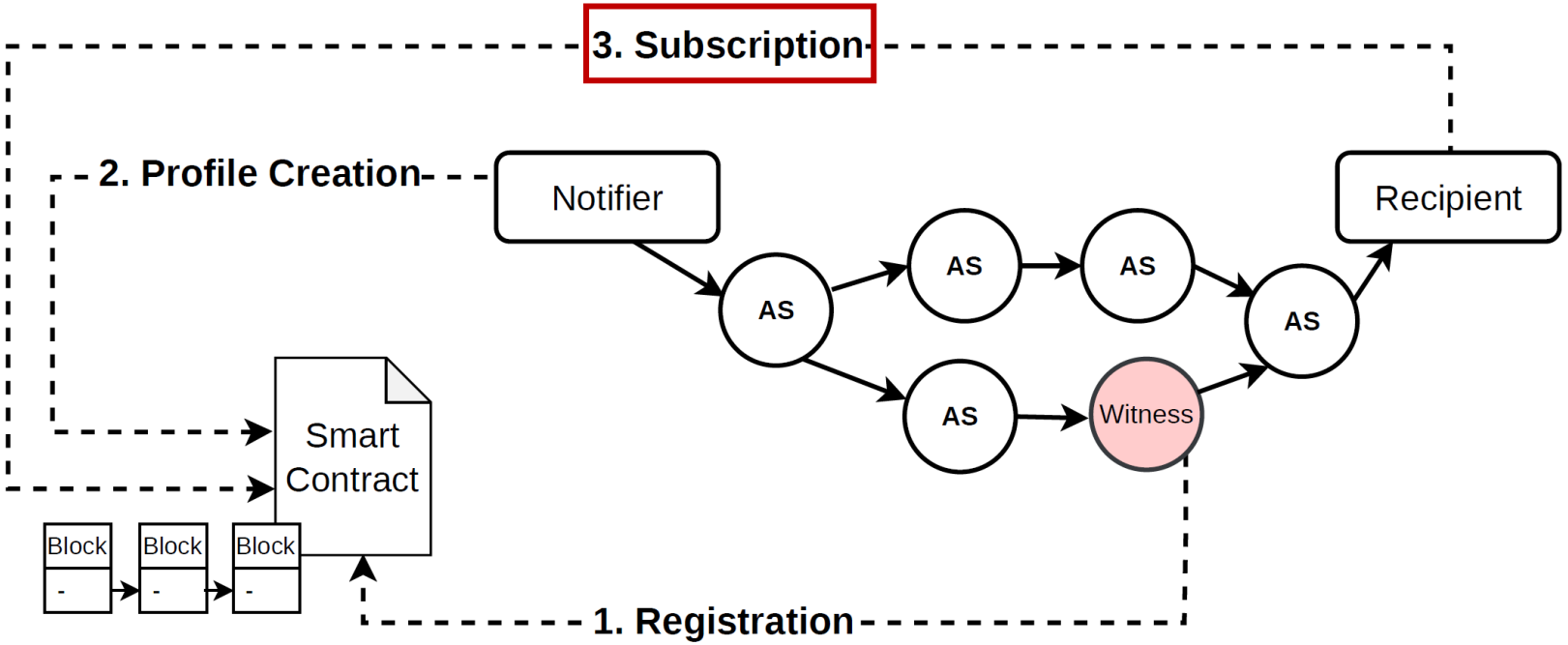

1. Witnesses register TESLA keys on-chain.

# Protocol Flow



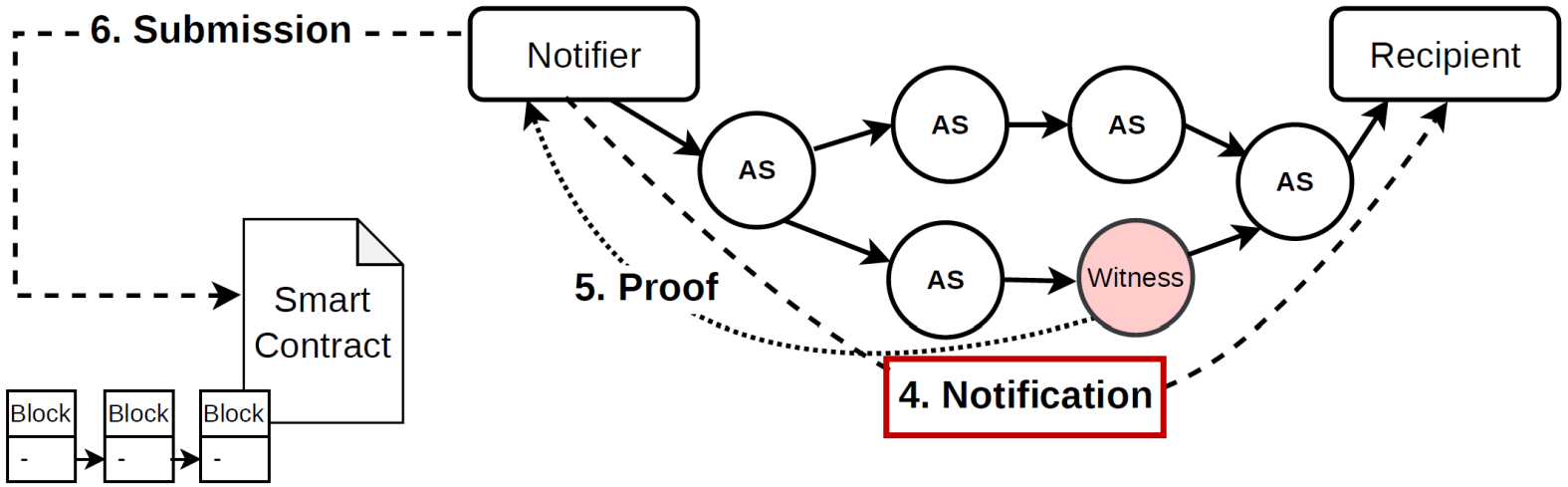

2. Notifier creates a profile with SLA and deposit.

# Protocol Flow



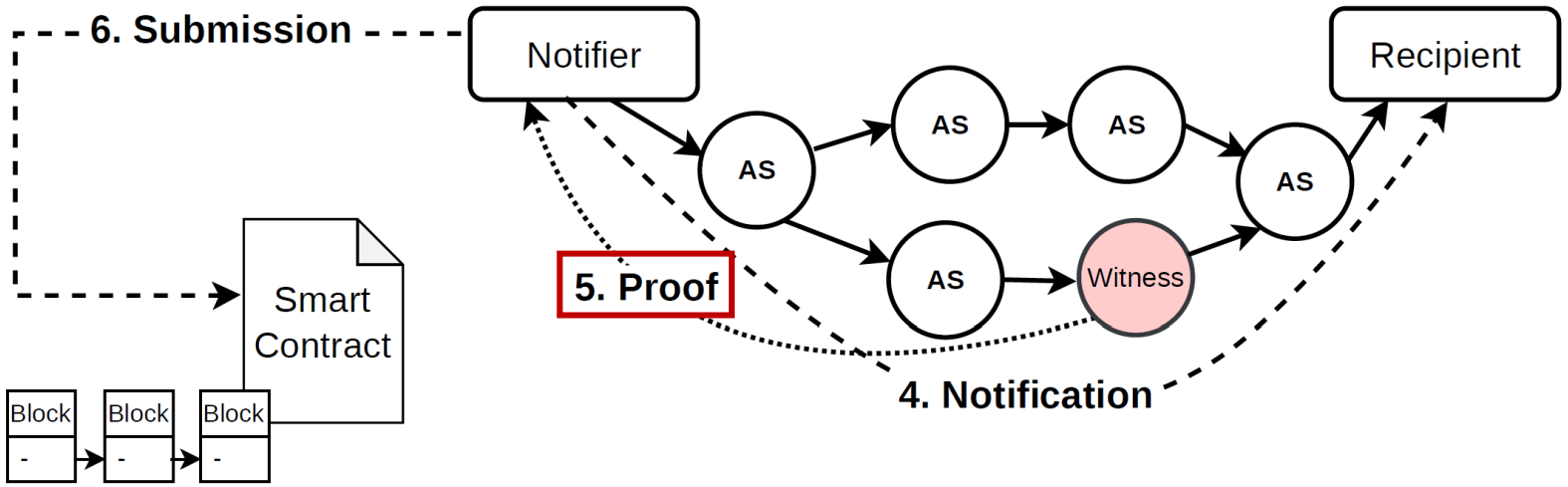

3. Recipient subscribes and selects trusted witnesses.

# Protocol Flow



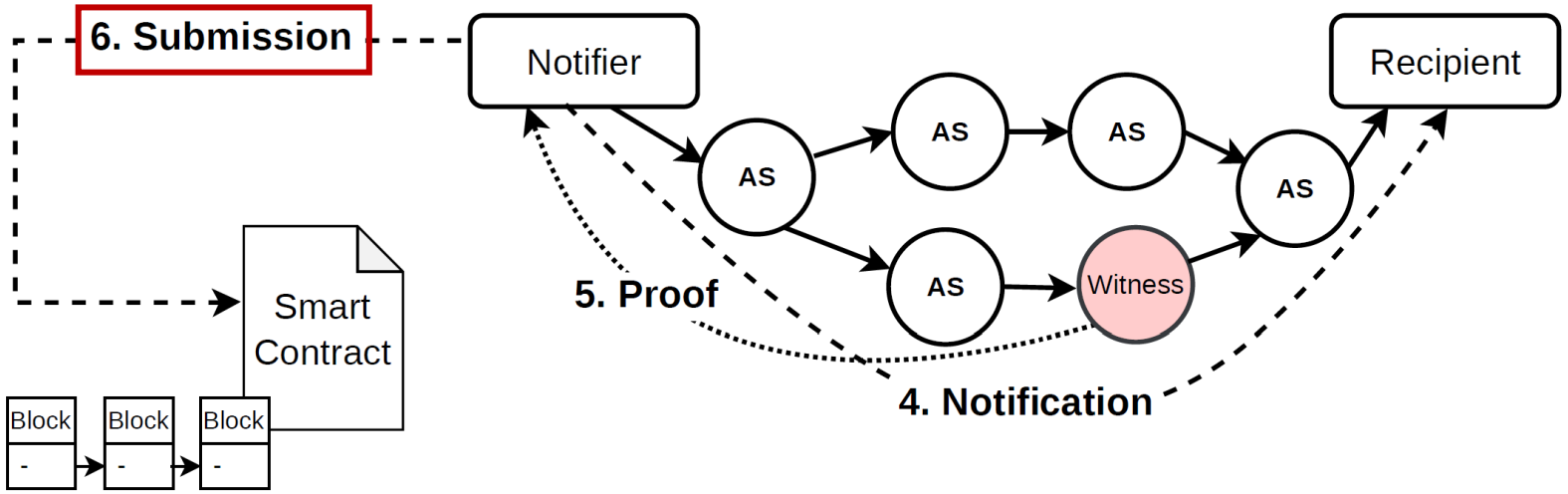

4. Notifier sends notification along a path through a witness AS.

# Protocol Flow




5. Witness creates a MAC-based proof and returns it to the notifier.

# Protocol Flow




6. Notifier batches proofs into a Merkle tree and commits the root on-chain.

# Proof Generation with TESLA Authenticator

**During interval i**

AS computes MAC($K_i$, m, ts)
Key $K_i$ is still secret.

**Before disclosure**

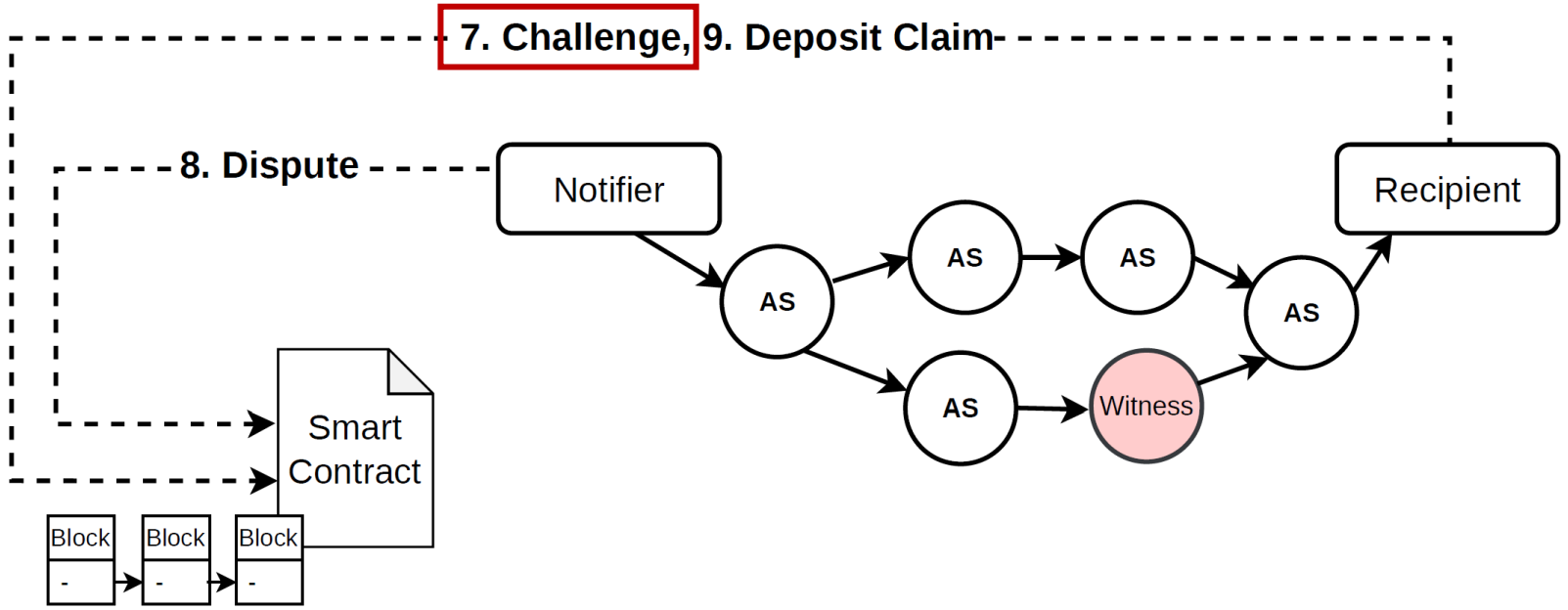
The notifier commits the proof on-chain.

**After key release**

Anyone can verify MAC authenticity and freshness.

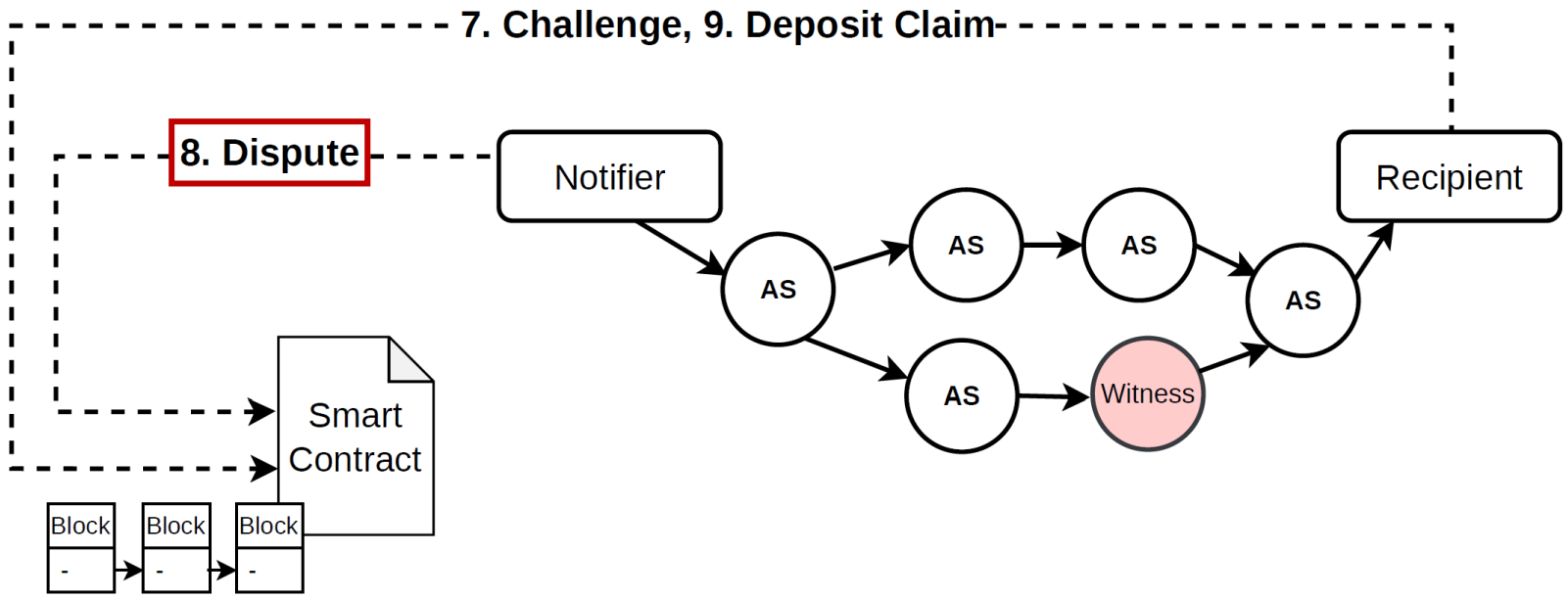
*No public-key operations on the critical path, Symmetric MACs instead → wire-speed proof generation*

# Protocol Flow



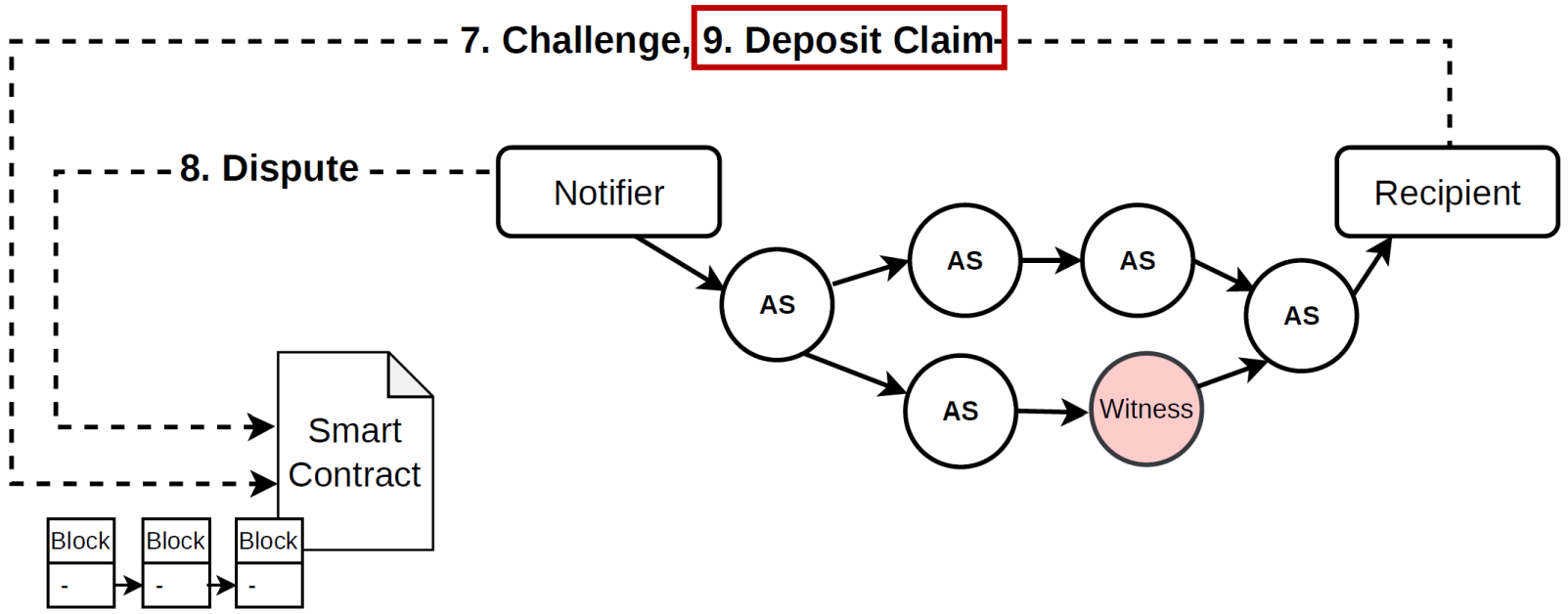

7. If notification is missed, recipient raises a challenge.

# Protocol Flow




8. Notifier disputes with the Merkle proof.

# Protocol Flow




9. Smart contract verifies and either drops the challenge or transfers the deposit.

# Security Analysis: Properties

**I** — **No Byzantine recipient can claim the deposit of an honest notifier.**

- False challenge: smart contract rejects non-existent events; valid proof MAC disputes any real event.
- Impersonation: source authentication blocks fabricated proofs from fake witnesses.

**II** — **An honest recipient can always claim the deposit of a misbehaving Byzantine notifier.**

- Omitted notification: notifier cannot forge a MAC without sending; deposit lost.
- Delayed notification: observation timestamp, non-repudiable via key anchoring.

**III** — **A faulty witness is detectable by notifiers.**

- After key disclosure, notifier verifies all MACs of that interval: invalid proofs identified.
- Misbehaving witnesses are detected and excluded.

**IV** — **A false notification is detectable by recipients.**

- Recipient queries indexer for blockchain inclusion proof of the event.
- Double notifications harmless: recipients not charged per notification.

# Evaluation: Data Plane (Router Throughput)

*Setup: Intel Xeon 2.1 GHz · Spirent SPT-N4U · 40 Gbps bidirectional · DPDK · SCION/EPIC router*

| | |
|---|---|
| **1.6M** pps | 16B packets<br>1 core |
| **17.4M** pps | 16B packets<br>16 cores |
| **0.6µs** overhead | per packet (16B) |
| **1.4µs** overhead | per packet (300B) |
| **Linear** scaling | throughput ∝ cores |

*Proof generation (hashing + timestamping + HMAC-SHA2*

# Evaluation: Smart Contract

- Implemented in Sui Move.*
- Deployed on Sui testnet.
- Each contract call costs less than 0.007 SUI (<$0.03).

* Open source: github.com/asonnino/signet-contract

# Conclusion

- Signet, a novel protocol for verifiable proof of notification.
- Mutually trusted Transit ASes as impartial witnesses.
- Integrating TESLA with AS witnessing enables:
  - ✓ Efficient proof generation by ASes.
  - ✓ Asynchronous on-chain verification.
  - ✓ Alignment with the trust assumptions of modern blockchain systems.
- Evaluation results:
  - ✓ negligible latency (μs).
  - ✓ >1M packets/sec per core.

# Thank you for your attention!

Elham Ehsani, ETH Zurich, NetSec

eehsani@ethz.ch