Enabling Secure and Scalable Data Analytics on Edge Devices

Arun Joseph

Department of Computer Science & Automation

Indian Institute of Science (IISc), Bangalore

Ph.D. Thesis Defense · 2026

Publications

This dissertation is based on three peer-reviewed publications

C1 — PERQS

A Contributory Public-Event Recording and Querying System

Arun Joseph, Nikita Yadav, Vinod Ganapathy, Dushyant Behl

ACM SoSR / SEC '23

C2 — Aramid

Data Protection in Permissioned Blockchains using Privilege Separation

Arun Joseph, Nikita Yadav, Vinod Ganapathy, Dushyant Behl, Praveen Jayachandran

COMSNETS '23

C3 — SoK (MPC vs TEE)

Evaluation of Methods for Privacy-Preserving Edge Video Analytics

Arun Joseph, Vinod Ganapathy

ICISS '25, Indore, India — December 2025

Contributions at a Glance

PERQS

- Contributory CCTV network on a permissioned blockchain
- PERQL query language (extends FRAMEQL) + fuzzy consensus
- Novel time-travel consensus for blind spots & camera faults
- Tamper-evident video via on-chain hash commitments

Aramid

- Privilege-separated Hyperledger Fabric (per-channel peers)
- Trusted proxy < 5% of peer+orderer LOC → smaller TCB
- Defends 8 concrete cross-channel / data-leak attacks
- ≈ 0.56% avg throughput cost; scales better with channels

MPC vs. TEE (SoK)

- Systematic evaluation: 14 MPC protocols + 3 TEE platforms
- 5 real video-analytics case studies on CityFlowV2
- 3-party replicated secret sharing (ring) = best MPC
- Decision boundary: TEE for heavy ML, MPC for trust-min.

01

PERQS

Public Event Recording and Querying System

A contributory, consensus-driven CCTV querying framework

Motivation: Cities Are Saturated with CCTV

But the footage is siloed across thousands of independent owners

- **CCTV cameras are ubiquitous, yet owned by many independent public and private entities**
  - – Delhi, India: 436,600 cameras — 1,446 per square mile, 26.7 per 1,000 people
  - – London, UK: 127,373+ public (942,562 incl. private) — a person is captured up to ~70×/day
- **Pooling this footage could greatly enhance public safety…**
  - – Accident investigation, hit-and-run tracing, forensic reconstruction across jurisdictions
- **…but the streams are owned by many entities and cannot be queried as one resource**

436,600

cameras in Delhi

1,446

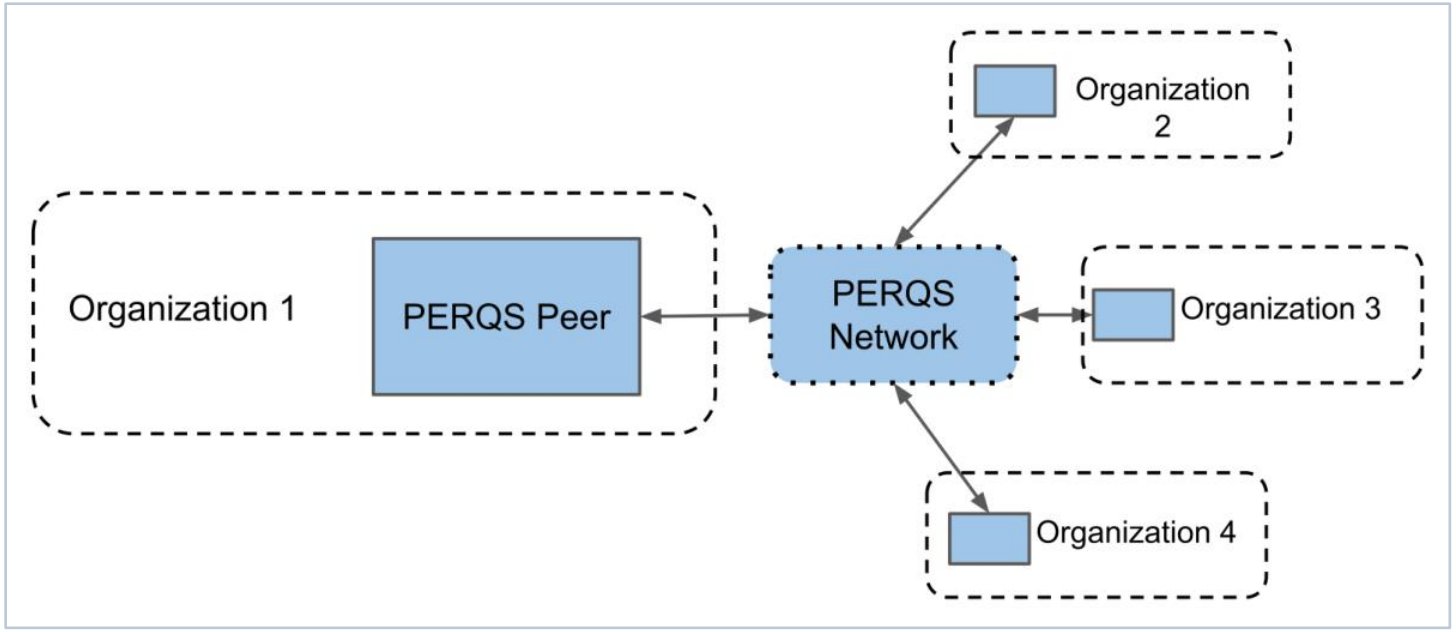
cameras / sq mile

~70×

captures / person / day (London)

PERQS: Uniting Public, Private & Individual Cameras

A single contributory network spanning mutually-distrusting owners



Contributory public-event recording & querying network

- **Participants**
  - – Public: traffic police, public buildings, parks, govt. institutions
  - – Private: companies, schools, residence associations, homes
- **Goal**
  - – Query an event across ALL feeds as if one resource
  - – Broader coverage when incidents span multiple locations
  - – Keep existing recording/storage; minimal deployment overhead

Four Challenges of a Contributory CCTV Network

1 Untrustworthy Video Feed

Technical errors or malicious activity → damaged / edited footage that is unreliable as evidence.

2 Privacy of Video Feed

Owners are reluctant to share raw feeds; risk of unauthorized access and misuse of footage.

3 Variable Quality of Feed

Heterogeneous cameras: differing resolution, viewing angles, aspect ratios, color calibration.

4 Availability / Blind Spots

Absence of cameras in some areas → surveillance blind spots; malfunctions create footage gaps.

How PERQS Addresses the Four Challenges

Tamper-Resistant Feed

solves: Untrustworthy feed

Video-hash commitment on the blockchain + verification before analysis → any retroactive edit is detectable.

Private Video Storage

solves: Privacy

Raw video never leaves the owner's domain; PERQS shares only query-result tables for consensus.

Consensus-Based Analysis

solves: Variable quality

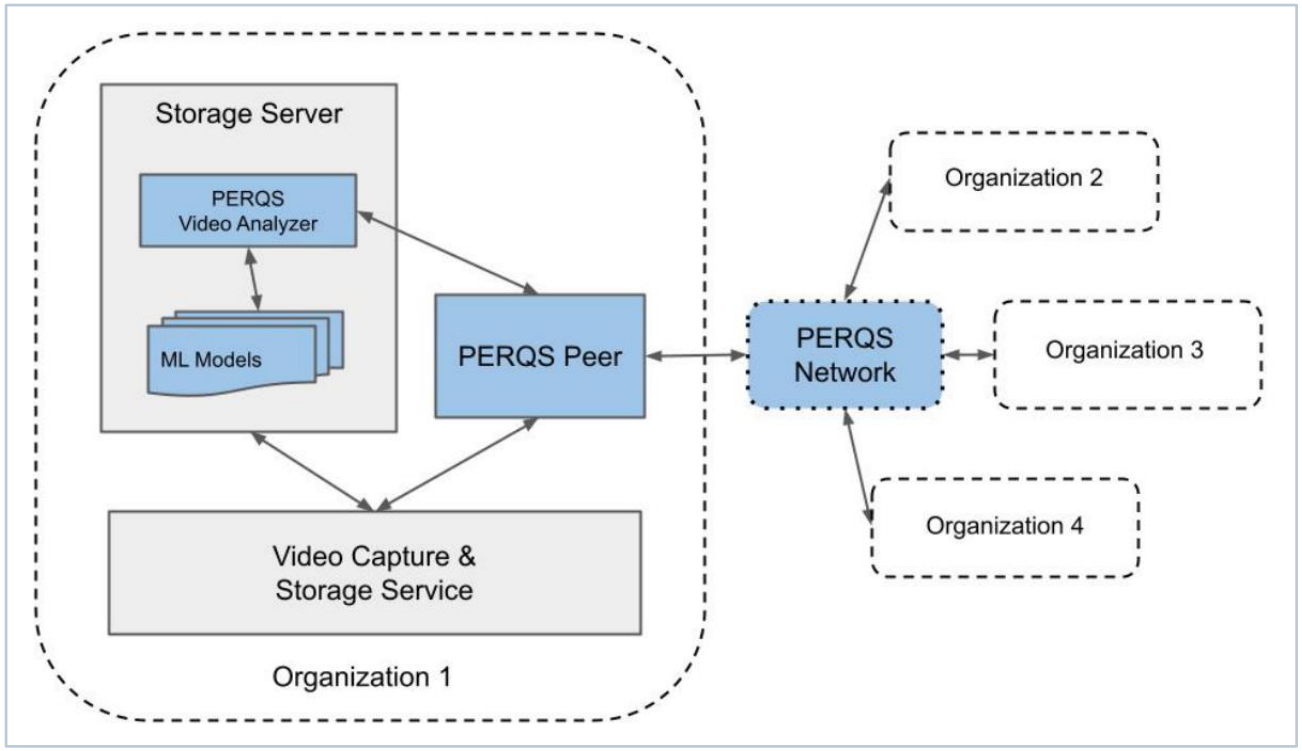
Multiple cameras analyze the same event; faults, damage or malice of individuals don't change the output.

Time-Travel Consensus

solves: Blind spots / faults

Uses footage from other cameras across space & time to fill gaps — travels back/forward to corroborate.

PERQS Architecture: Four Components



PERQS components & their interactions

Video Capture & Storage Service

Manages an organization's CCTV cameras and handles local video storage.

PERQS Network

Permissioned blockchain connecting all participants; membership is verified.

PERQS Client (Peer)

Commits video hashes, decodes & dispatches queries, and reaches consensus on results.

PERQS Video Analyzer

Runs the ML/vision model on local feeds; decodes & executes queries, returns a result table.

PERQL: The PERQS Query Language

A declarative extension of FRAMEQL for distributed, consensus-based video querying

CREATE MODEL

Register a new ML / vision model (e.g. an object detector) with the network.

ALTER MODEL

Update an existing model and propagate the change to all participating organizations.

SELECT

Extract information from the collective video database; narrows feeds by GPS + time, runs the model, returns a result table.

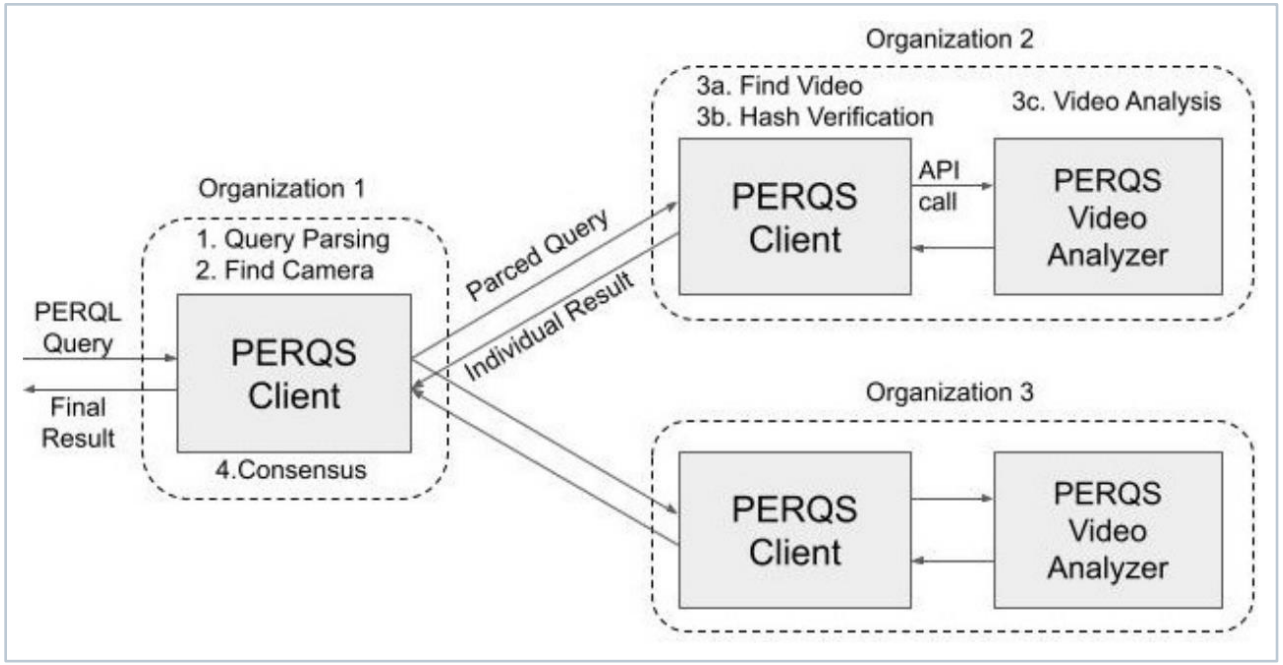
PERQL — example SELECT

```
SELECT timeAppeared, vehicleType, vehicleColor
    FROM vehicleDetector
    AT 42.525678, -90.723601
    WHERE vehicleType="truck",
          vehicleColor="#0000FF"
    TIME 2023-01-25 11:30:00.005
    TO 2023-01-25 11:35:00.005

--  fields FROM model WITH params
--  AT gps WHERE conditions TIME t0 TO t1
```

Query Execution: Four Stages

The life-cycle of a single PERQL query



Stages of a single query execution

1 · Query Parsing

Decode keywords and parameters from the query string.

2 · Find the Camera

Camera-finder oracle returns camera IDs for the given GPS location & range.

3 · Owner Query Execution

Owner locates video by time → verifies hash vs. on-chain commitment → if match, calls the analyzer API → returns a result table.

4 · Consensus

Combine all returned tables using fuzzy consensus rules & policies (majority / at-least-n).

Fuzzy Consensus Evaluation

Reconciling non-exact matches across heterogeneous cameras & models

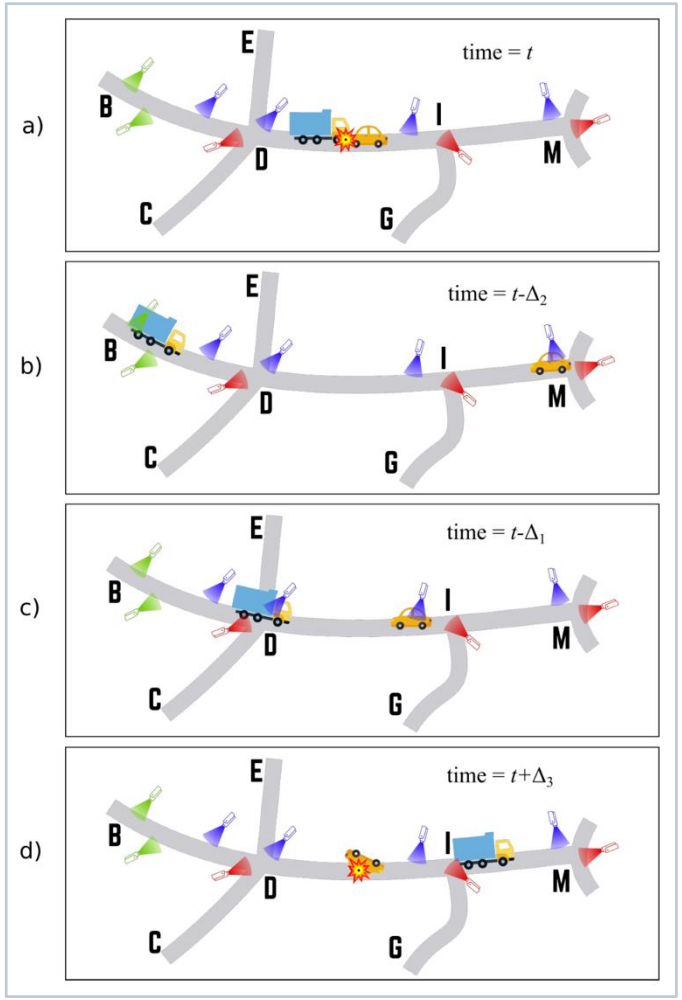
#	Data Type	Operation	Consensus Rule
1	String	Equals	Majority, or at least n
2	Set of strings	Intersection / Union	Strings present with majority (or n of results)
3	Range (time/int/float)	Overlapping range	Floor or ceil of the overlapping interval
4	Number (float/int)	Equals	±5% treated as equal (x% configurable)
5	Set of numbers	Set intersection	Majority; ±5% treated as equal
6	Color (hex string)	Equals	±5% per RGB channel treated as equal
7	Boolean	Equals	Majority, or at least n

- **No strict bitwise equality**
  - – Configurable, channel-wise tolerances absorb natural variation
- **Policies**
  - – majority · at least n cameras · at least n organizations
  - – Defaults can be overridden per query
- **Minimum t cameras must participate to reach a collectively verified decision**
- New data types → add analyzers / policies incrementally

Time-Travel Consensus: Motivation

When majority consensus is impossible — no / too few cameras, or malicious owners

- **Majority consensus needs overlapping honest observations at one location**
- **It fails when:**
  - there are no cameras at the location (blind spot)
  - there are insufficient cameras for a quorum
  - camera owners are malicious / faulty
- **Key insight**
  - Real-world events are rarely isolated — they are causally linked to activity earlier/later at neighboring locations
- **Time-travel consensus "travels through time & space":**
  - trace related vehicles backward (pre-event) and forward (hit-and-run) across nearby cameras
  - reformulate a failed direct query into an indirect inference over corroborating evidence



(a) accident (b) $t-\Delta_2$ (c) $t-\Delta_1$ (d) hit-and-run, $t+\Delta_3$

Time-Travel Consensus: Functional Query Adaptation

Instead of detecting the event, detect its causally-linked traces

Direct query FAILS

Accident occurs in a blind spot → the collision itself was never captured → a standard accident-detection model returns nothing.

Reformulate → infer

- Vehicle speed prior to estimated collision time
- Vehicle direction & trajectory
- Vehicle presence at adjacent junctions

Three core principles

Spatiotemporal Context

Events are interconnected across time & geography; a collision is preceded/followed by observable behavior elsewhere.

Indirect Evidence Aggregation

Consensus from aggregated supporting evidence at nearby cameras — not direct visual confirmation.

Plug-and-Play Models

Speed / trajectory / Re-ID models are modular evidence sources; better models drop in without changing consensus logic.

Time-Travel Consensus: Operational Procedure

Triggered when the camera-finder oracle returns fewer cameras than the policy requires

1 Expand spatial radius R

Identify additional nearby cameras that may have captured causally-related events.

2 Compute temporal offset Δ

$\Delta \approx f(R)$: expected vehicle travel time within radius R.

3 Reformulate the query

Transform a failed direct query into auxiliary queries: speed, trajectory, presence.

4 Execute over windows

Run modified queries on new feeds at $(t-\Delta)\pm\theta$ and $(t+\Delta)\pm\theta$, where θ is the search interval.

5 Aggregate & correlate

Search for consistent patterns: overspeeding, matching trajectories, temporal alignment.

6 Establish consensus

If enough independent cameras give mutually-consistent evidence, the policy is satisfied.

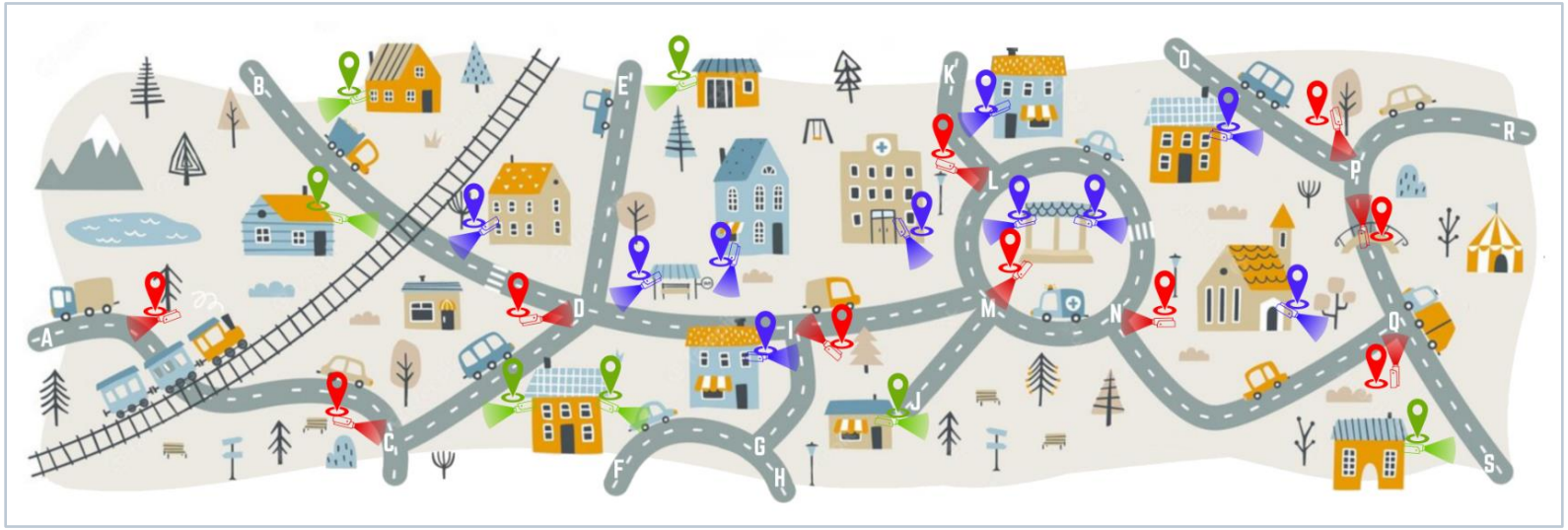
R and Δ are set empirically at deployment (small R in dense areas, larger R when sparse); R expands iteratively if too few cameras are found.

Time-Travel Consensus: When It Does Not Apply

- **Some events produce no observable causal traces at nearby cameras → indirect reconstruction impossible**
- **Cameras within expanded radius R may be irrelevant and provide no supporting evidence**
- **Depends on the availability of appropriate ML models**
  - If only an accident-detection model exists and no camera saw the collision, consensus cannot be reached
  - Needs speed estimation, direction inference, number-plate detection, vehicle tracking, color detection, Re-ID…
- **Inference precision is bounded by the performance of these vision models**
- **Effectiveness is bounded by spatiotemporal continuity and camera density**

Experimental Setup

A realistic city-scale deployment on real traffic footage



Example city map: red = traffic-police cameras; junctions labeled

- Three case studies exercise majority consensus, time-travel consensus, and Byzantine robustness
- Metrics: Accuracy, Precision, Recall, F1, plus end-to-end latency breakdown

Dataset

AICITY21 / CityFlowV2 — real traffic surveillance

Scale

46 cameras · 880 annotated vehicles · 215.03 min of video

Focus

8 junctions with overlapping multi-camera coverage

Analytics

YOLOv3 object detector as the vehicleDetector model

Platform

Hyperledger Fabric v2.2 on Kubernetes

Case Study 1 — Majority Consensus

Find a blue truck that passed through junction 1 around 11:30 AM

Five time-synced camera views of junction 1 (CityFlowV2)

PERQL query

```
SELECT timeAppeared, vehicleType,
       vehicleColor
    FROM vehicleDetector
    AT 42.525678, -90.723601
    WHERE vehicleType="truck",
          vehicleColor="#0000FF"
    TIME 2023-01-25 11:30:00.005
    TO   2023-01-25 11:35:00.005
```

- Five overlapping views → simple majority policy: a vehicle appears in the final table only if a majority of cameras saw it
- Fuzzy color matching: the same blue truck yields different hex per camera (#012AB4, #103BB8, #002CAD…) due to lighting & sensor calibration → ±5% per RGB channel considered equal

Case Study 1 — Consensus Result

Five per-camera tables reconciled into one final table

Source	timeAppeared	type	color (hex)
Camera 1	11:30:04.90–07.10	truck	#012AB4
Camera 2	11:30:04.30–06.70	truck	#103BB8
Camera 3	11:30:05.10–17.30	truck	#002CAD
Camera 4	11:30:01.80–06.80	truck	#003AB3
Camera 5	11:30:04.80–09.20	truck	#001261
★ Consensus	11:30:05.10–06.70	truck	#032CA2

- **First appearance of the blue truck**
- Each camera reports a slightly different colour and time window
- Time-sync oracle aligns the tables; fuzzy consensus reconciles colour (±5% RGB) and overlapping time ranges
- **Result: the truck is confirmed by the honest majority with a consensus colour of #032CA2**

Case Study 2 — Time-Travel Consensus

Find a cyclist passing through junction 8 — which lacks overlapping camera views

Cycle captured by cameras 10,11,12,13,15 at different times

Three cycles at cam 15, 10:01:06.10

query

```
SELECT timeAppeared,
  vehicleType
      FROM vehicleDetector
      AT 42.52.., -90.72..
      WHERE type="cycle"
      TIME 10:00 TO 10:05
```

- Only 2 of 6 nearby cameras see the junction; one misses the cycles entirely (turned before entering view)
- **Remaining cameras record the cycle a few seconds before / after the junction → merged via time-travel to reach a majority**

Case Study 3 — Byzantine Robustness

Find a white truck at junction IV — with one malicious and one faulty camera

White truck at junction IV, cameras 16–20 (cam 19 malicious)

- **5 cameras, each a different organization**
- **Camera 19 (Org 4) = malicious:**
  - – reports a car instead of a truck
  - – alters detected colour information
- **One further camera = faulty:**
  - – low resolution, poor lighting → incomplete/incorrect attributes
- **Despite 1 malicious + 1 faulty, the 3 honest cameras correctly detect the white truck**
- **Consensus reached by honest majority → Byzantine tolerance confirmed**

Evaluation: Consensus Improves Accuracy

Individual cameras vs. PERQS consensus

Configuration	Accuracy	Precision	Recall	F1
Camera 1 (alone)	70.0%	79.4%	70.6%	0.748
Camera 2 (alone)	72.5%	77.4%	75.2%	0.763
Camera 3 (alone)	65.0%	73.1%	69.7%	0.714
PERQS — 3 Cameras	82.5%	84.6%	80.7%	0.826
PERQS — All Cameras	92.5%	90.6%	88.1%	0.893

- **Single cameras top out at 65–72.5% accuracy; consensus across 3 cameras reaches 82.5%**
- **With all cameras participating, accuracy climbs to 92.5% and F1 to 0.893**
- 28 of 40 queries resolved by majority; 12 required time-travel consensus

Evaluation: Robustness Under Faults & Attacks

Varying Camera 1 across fault / adversarial conditions

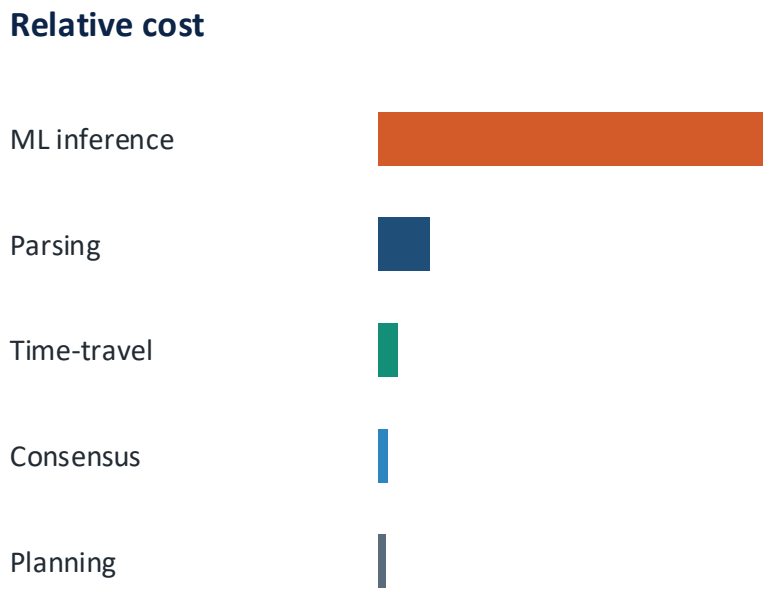
Condition on Camera 1	C1 Acc.	3-Cam Acc.	3-Cam F1	All-Cam Acc.	All-Cam F1
No fault	70.0%	82.5%	0.826	92.5%	0.893
Camera fault	55.0%	77.5%	0.777	90.0%	0.874
YOLOv3 ↓ confidence	42.5%	85.0%	0.853	92.5%	0.896
YOLOv3 ↑ confidence	45.0%	75.0%	0.828	87.5%	0.880
50% incorrect reports	32.5%	70.0%	0.761	85.0%	0.836
100% incorrect reports	0.0%	57.5%	0.651	77.5%	0.806

- **Even when Camera 1 reports 100% incorrect values (its own accuracy → 0%), PERQS all-cameras holds 77.5% accuracy / 0.806 F1**
- **A single faulty camera barely dents the collective result — honest cameras dominate consensus**
- Lowering YOLOv3 confidence can even help: more true positives survive aggregation while false positives are filtered out

Performance: Where Does the Time Go?

End-to-end latency breakdown — ML inference dominates

Stage	Latency	Note
Query planning	31.5 ms	camera-finder lookup
Query parsing	230.5 ms	decode keywords/params
Video analytics (ML)	1.72 s	per second of video — dominant cost
Majority consensus	47.15 ms	reconcile result tables
Time-travel consensus	89.44 ms	expanded search + aggregation



- **Machine-learning inference is the overwhelming cost (1.72 s per second of video)**
- **Both consensus mechanisms are cheap (< 90 ms) — the security layer adds negligible overhead**

PERQS vs. Existing Systems

Why relational DBs and centralized video engines don't solve this

Relational DBs (PostgreSQL, DuckDB)

- Assume outputs already materialized into trusted tables
- Assume centralized, trusted data ownership
- No collaborative cross-organization verification
- Fuzzy reconciliation needs complex UDFs + orchestration

Centralized Video Engines (FrameQL)

- Optimize queries over one trusted central repository
- Treat detector outputs as ground truth
- No notion of noisy / malicious observations
- No distributed ownership of feeds

PERQS

- Raw feeds stay distributed across untrusted owners
- Collectively-verified results via consensus
- Fuzzy consensus built into the query lifecycle
- Integrity verification integrated end-to-end

PERQS— Summary

- **PERQS turns siloed CCTV into a trustless, query-driven ecosystem**
- Blockchain hash commitments make historical footage tamper-evident without sharing raw video
- PERQL + fuzzy consensus reconcile heterogeneous, noisy camera outputs into verified results
- **Time-travel consensus recovers answers in blind spots by aggregating causal, cross-camera evidence**
- **Consensus lifts accuracy from ~65–72% (single camera) to 92.5% (all cameras)**
- Robust to Byzantine actors: 77.5% accuracy even when a camera reports 100% wrong data
- Security overhead is negligible — ML inference dominates end-to-end cost

02

Aramid

Data Protection in Permissioned Blockchains via Privilege Separation

Hardening the infrastructure beneath PERQS

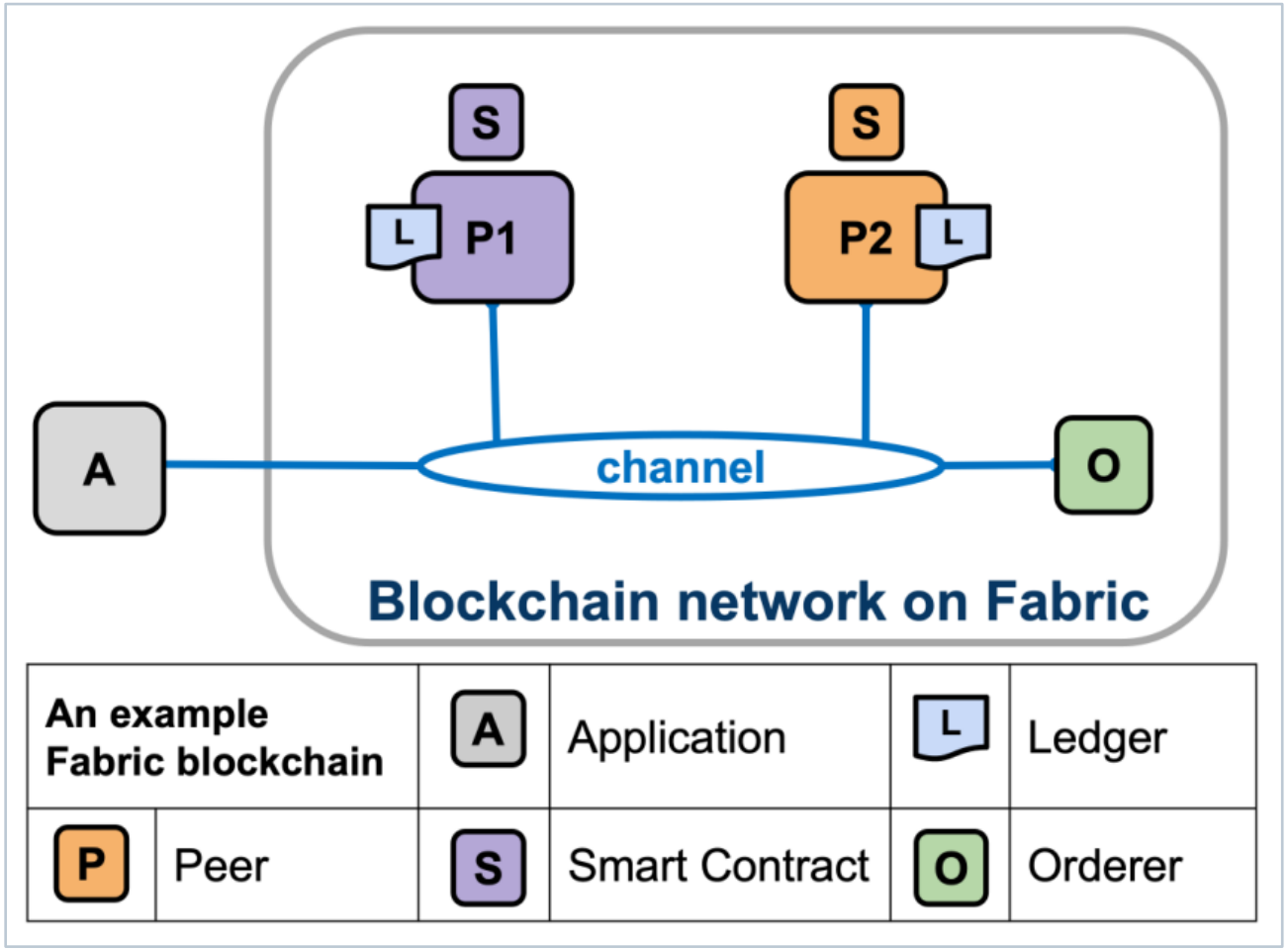
Why Aramid? The Fabric Attack Surface

Consensus & commitments are useless if the infrastructure leaks data

- **PERQS relies on Hyperledger Fabric for video-hash verification**
- **Fabric tolerates faulty/malicious peers for integrity — but offers NO data-protection guarantee**
  - A single compromised peer (exploit or insider) can leak confidential channel data
- **In Fabric, one peer process can belong to MANY channels**
  - → it can access — and leak — ledgers private to all of them
  - → cross-channel leaks, membership disclosure, business-logic exposure
- Application-layer encryption negates smart-contract computation; ZKPs/MPC add heavy overhead
- **Root problem: too large an attack surface — too many trusted components & users**

Aramid's Core Idea: Privilege Separation

One multi-channel peer → many single-channel peers behind a trusted proxy



Fabric peer (multi-channel) vs. Aramid (one peer instance per channel)

- **Each Aramid peer belongs to exactly one channel and holds one ledger**
  - – Runs in its own Docker container / fresh pod
  - – Its smart contracts run as separate external chaincode pods it alone talks to
- **A trusted proxy fronts all peer instances**
  - – Inspects & routes messages by channel — preserves transparency to legacy apps
  - – Proxy + peers look like one 'peer' to the Fabric network
- **Trust shifts FROM many peers TO one small proxy**
- Same design applied to orderer nodes (one orderer per channel)

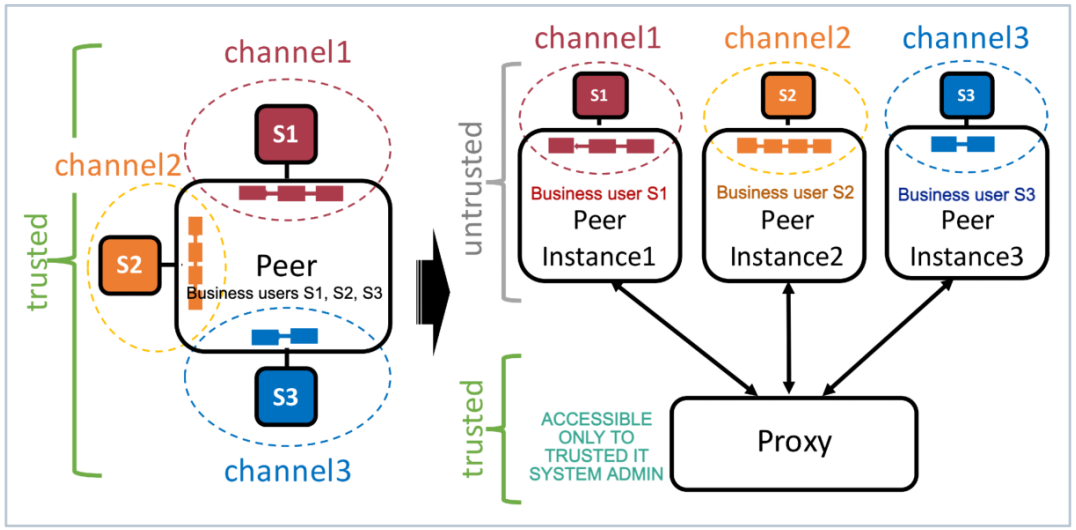
Aramid: Two Kinds of Enforced Policies

1 · Fabric-specific (trusted proxy)

- Components exchanging channel messages must be on the same channel
- Proxy inspects all in/out messages; silent-drops policy violations
- Orderer proxy: blocks go only to channel members (per header)

2 · Generic network (Istio + K8s)

- Peer talks only to its proxy, ledger, and smart contracts
- Smart contract & ledger talk only to their hosting peer
- All network communication encrypted with TLS by default



Client ⇄ trusted proxy ⇄ per-channel peer instances in Aramid

Shrinking the Trusted Computing Base (TCB)

The proxy is under 5% of the code it replaces in the trust boundary

Component	LOC (Go)	Vendor modules	In whose TCB?
Peer	141,031	18	Fabric (untrusted in Aramid)
Orderer	55,501	16	Fabric (untrusted in Aramid)
Trusted proxy	8,541	13	Aramid

- **In Fabric, the huge peer (141K LOC) + orderer (55K LOC) are all trusted**
- **In Aramid, peers & orderers are UNTRUSTED — trust is confined to the 8,541-LOC proxy (< 5%)**
- **Smaller, simpler TCB → far smaller attack surface and easier to audit**
- Caveat: Aramid reduces attack surface — it does not stop out-of-band leaks by legitimate insiders

Eight Data-Leak Attacks — All Defended

Fabric's leakage avenues and how Aramid closes each

#	Attack in Fabric	Aramid defense
1	Send data to entities outside the blockchain	Network policies: peer talks only to proxy + chaincode pods
2	Send unencrypted data	Service mesh enforces TLS on every connection
3	Intra-container data leaks	Unprivileged containers, read-only FS, limited host access
4	Inter-channel data leaks	Separate peer instance per channel (by construction)
5	Leak channel membership info	Peer knows only its own channel; proxy handles membership
6	Leak smart-contract business logic	External chaincode pods talk only to their peer
7	Leaks via non-channel (gossip) messages	Proxy handles/answers gossip on behalf of peers
8	Leak private-data-collection data	Proxy checks gossip content + collection policy first

Aramid: Security at Almost No Cost

Throughput & latency impact of the added proxy (avg. of 10 runs)

~1,670 tps

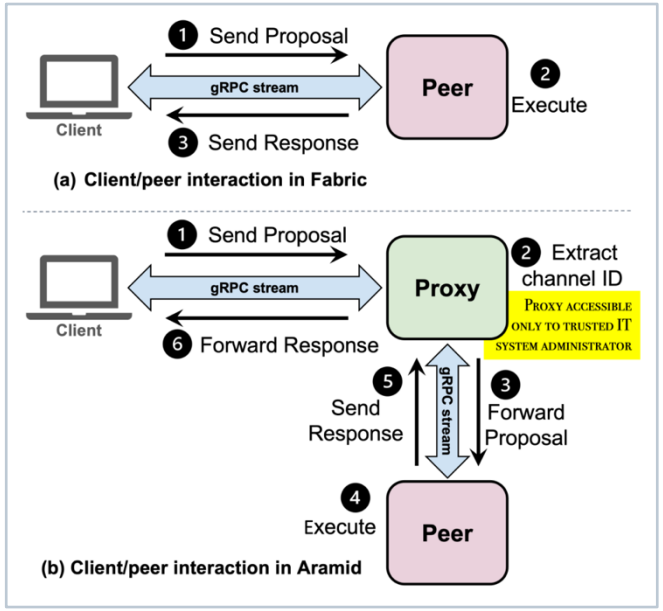
Saturation throughput — identical for Fabric & Aramid (single channel)

0.56%

Average throughput reduction vs. Fabric

2,100 vs 1,950 tps

With 6 channels, Aramid BEATS Fabric (per-channel peers)



Aramid transaction flow

- **The proxy is the only added element on the data path — overhead is < 1% on average**
- **With more channels, single-channel peers avoid Fabric's multi-channel bookkeeping**
- **Net result: stronger data protection with practically unchanged performance**

03

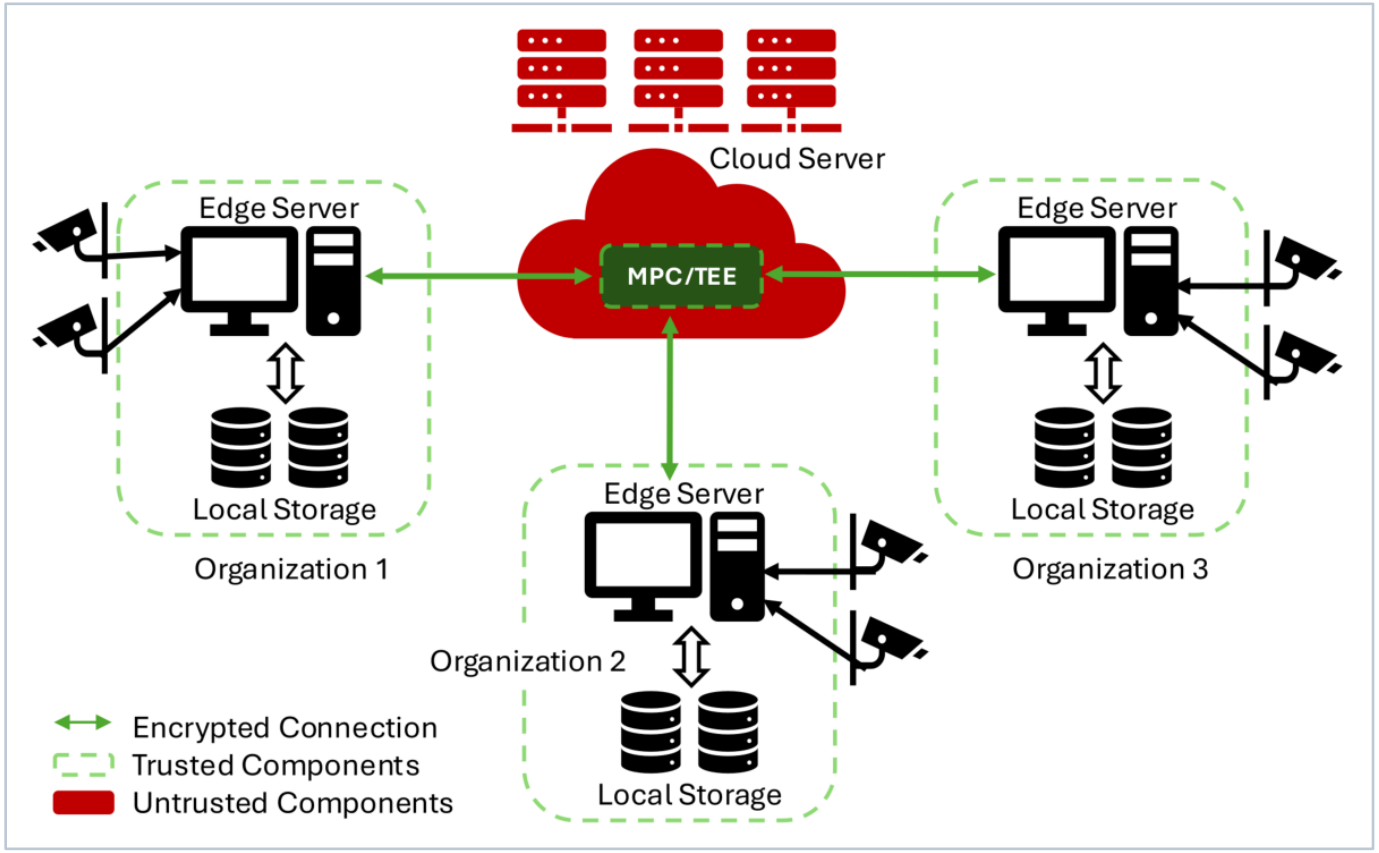
MPC vs. TEE

SoK: Methods for Privacy-Preserving Edge Video Analytics

Secure JOINT analytics when raw video cannot be shared

From Querying to JOINT Analytics

PERQS shares result tables — but some tasks need to compute over combined raw feeds

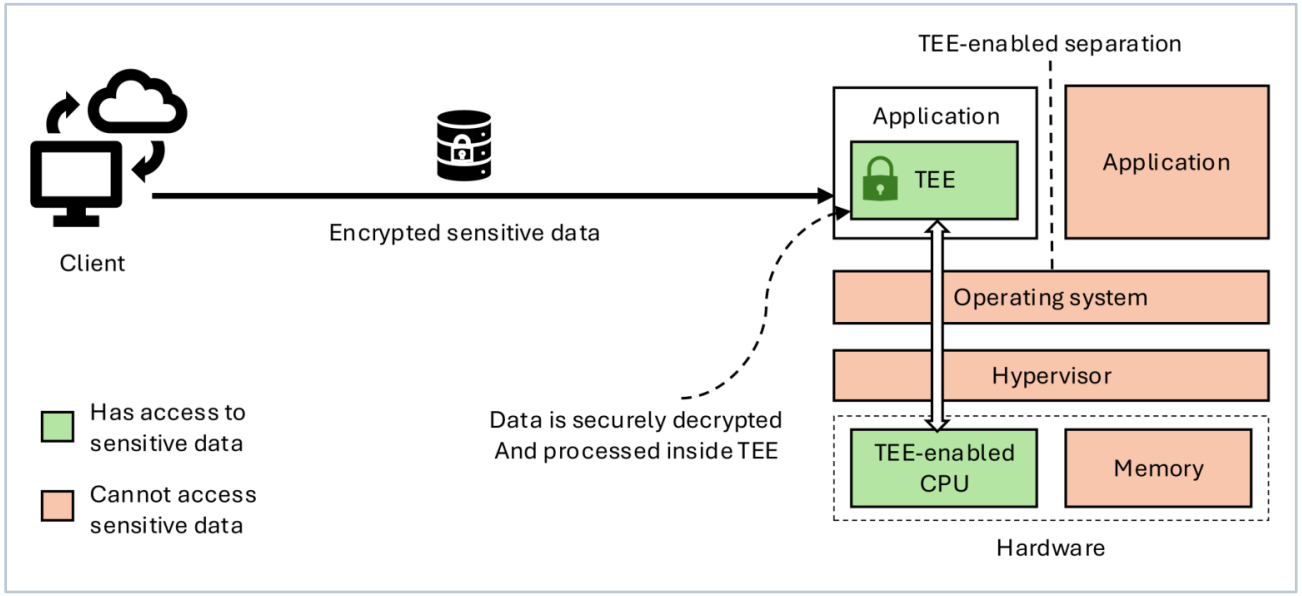


Extending PERQS with TEE / MPC for joint video analytics

- **PERQS does not natively support joint video analytics across organizations**
- **Some tasks require computing over multiple parties' raw feeds together:**
  - vehicle re-identification, scene similarity, joint recognition, 3D reconstruction, vehicle counting
- **Raw video is private → the joint computation must be secure**
- **Two families of techniques:**
  - MPC — cryptographic, no trusted hardware
  - TEE — hardware enclaves, near-native speed
- **This SoK asks: which one, when?**

Trusted Execution Environments (TEE)

Secure hardware enclaves — near-native performance, hardware trust



TEE-based joint analytics on a cloud server

Near-native performance, but security tied to hardware/firmware correctness — vulnerable to side-channel & speculative-execution attacks (Spectre/Meltdown).

Intel SGX

Secure enclaves for small sensitive workloads; strong isolation from OS.

AMD SEV

Virtualization-based; encrypts VM memory from hypervisor/other VMs.

AWS Nitro

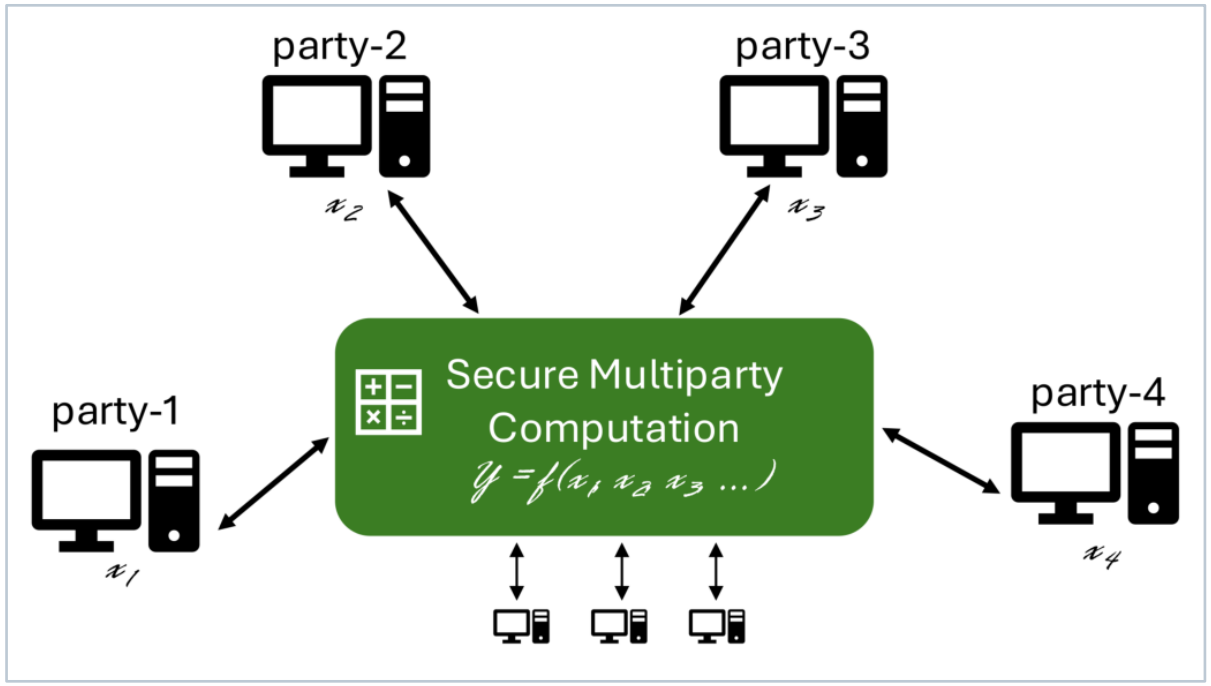
Container-based enclave, isolated from host EC2 — no external net/storage.

ARM TrustZone

System-wide secure vs. normal worlds within the processor.

Secure Multiparty Computation (MPC)

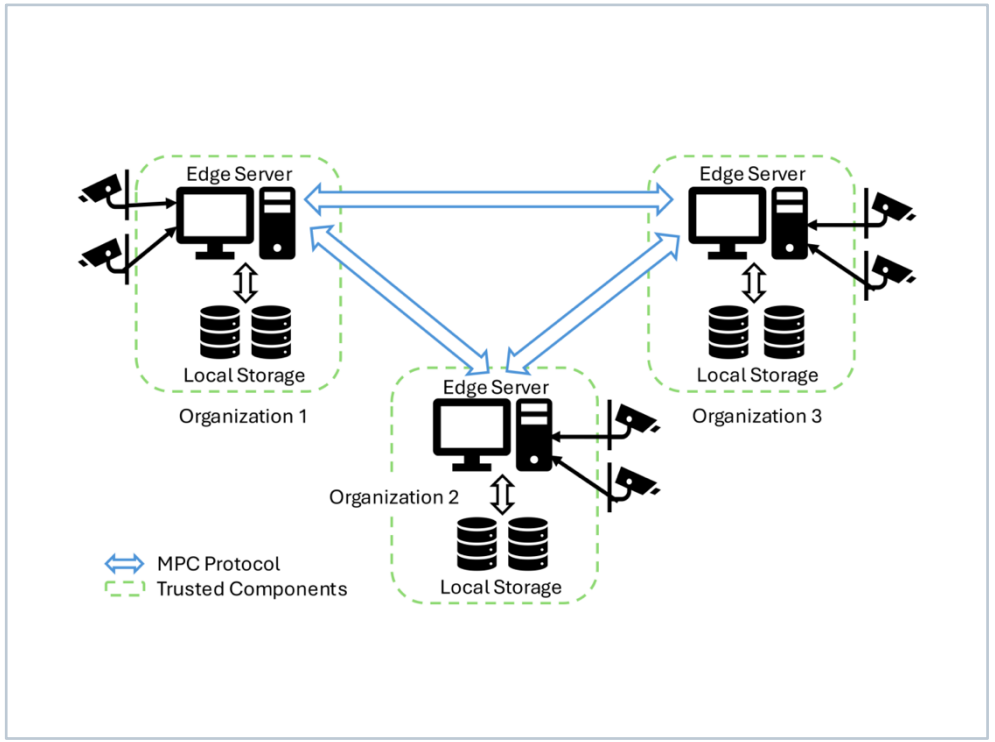
Joint computation without revealing private inputs — no trusted hardware



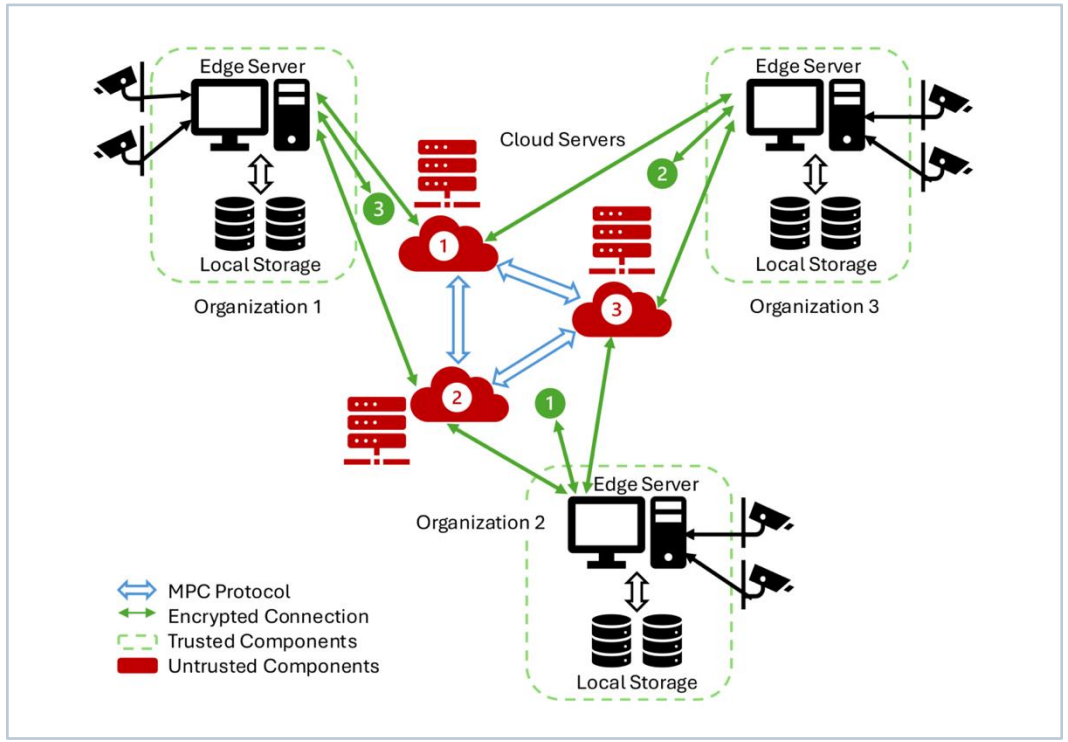
MPC: parties jointly compute on secret-shared inputs

- **Secure but resource-intensive; no reliance on a central trusted authority**
- **Design choices (MP-SPDZ, 14 protocols):**
  - Adversary: semi-honest (passive) vs. malicious (active)
  - Domain: arithmetic vs. Boolean
  - Technique: secret sharing vs. garbled circuits
  - Parties: 2 / 3 / 4-party
  - In-house vs. outsourced (edge shares → cloud servers)

MPC Deployment: In-House vs. Outsourced



In-house MPC: edge devices compute directly



Outsourced MPC: shares sent to cloud servers

- **Edge devices (DVRs, smart cameras) are resource-constrained**
- **Outsourced model shifts computation to independent cloud servers — edge only shares secret inputs → far less edge load**

Choosing the Best MPC Protocol

Image-processing macro-benchmarks — execution time 3-party ring wins

Workload	semi2k (2p)	ring* (3p)	rep-field* (3p)	yao (GC)	mascot (mal.)
Matrix ops	0.03 s	0.03 s	0.02 s	0.11 s	1.64 s
Thresholding	1.36 s	0.08 s	0.43 s	1.75 s	569.3 s
Histogram	1.93 s	0.07 s	0.456 s	2.12 s	488.4 s
Sobel	27.2 s	4.23 s	4.36 s	NA	568.3 s
3×3 Convolution	13.9 s	4.43 s	4.12 s	NA	354.3 s
Thinning	4.96 s	0.11 s	0.58 s	16.8 s	610.6 s

- **3-party replicated secret sharing (ring) = fastest & lowest communication across all workloads**
- **Secret sharing ≫ garbled circuits for arithmetic-heavy tasks; malicious-secure protocols are 100s× slower (Sobel > 100 GB)**
- **More parties → more time & communication; ring chosen for the TEE comparison**

Five Joint Video-Analytics Case Studies

1 · Vehicle Re-ID

Match a vehicle across cameras via feature vectors + cosine similarity.

2 · Scene Similarity

Compare scenes via feature vectors + Euclidean distance.

3 · Joint Recognition

Stitch complementary camera views, then recognize objects.

4 · 3D Reconstruction

Fuse multi-angle video into a 3D scene, then query it.

5 · Vehicle Counting

Count locally, then de-duplicate across cameras via similarity.

Evaluated on CityFlowV2

1,3,5 extract features locally (lightweight); 3,4 require heavy collaborative DNN inference.

MPC vs. TEE: Head-to-Head Performance

Execution time & communication per case study

Case Study	TEE (SGX)	TEE (Nitro)	MPC (MP-SPDZ)
1 · Vehicle Re-ID	0.002 s / 98 kB	0.001 s / 98 kB	0.006 s / 218 kB
2 · Scene Similarity	0.003 s / 123 kB	0.0012 s / 123 kB	0.015 s / 0.36 MB
3 · Joint Recognition	NA	4.1 s / 37.3 MB	347.5 s / 5.08 GB
4 · 3D Reconstruction	NA	7.3 s / 34.3 MB	621.4 s / 8.56 GB
5 · Vehicle Counting	0.025 s / 1.3 MB	0.017 s / 1.3 MB	0.033 s / 2.1 MB

- **Lightweight tasks (1,2,5): MPC only 3–6× slower than TEE, ~2× communication — MPC viable**
- **Heavy ML tasks (3,4): MPC is 70–90× slower and needs GIGABYTES of communication — infeasible**
- **Intel SGX can't run the ML models at all (no syscalls, 128 MB EPC) → 'NA' for cases 3 & 4**

Implementation Realities

What actually constrains each approach in practice

Intel SGX limits

- No system calls inside the enclave → can't use PyTorch / TensorFlow
- 128 MB EPC memory limit → modern ML models don't fit
- Officially discontinued on consumer platforms
- Could NOT run Python ML inference in SGX for this study

AWS Nitro & MPC

- Nitro: container-based, Docker images → flexible & developer-friendly
- Nitro benefits from EC2 scalability; ran the heavy ML workloads
- MP-SPDZ: modular, well-documented, arithmetic + Boolean
- But ML in MPC needs manual layer/activation definitions → error-prone

MPC vs. TEE: The Decision Boundary

Choose TEE when…

- Performance is paramount
- Compute-intensive DNN inference / multi-frame aggregation
- 70–90× faster on heavy ML; only MBs transferred
- Real-time collaborative ML is required
- Simple for developers (esp. Nitro)

Choose MPC when…

- Trust minimization is the priority
- No reliance on hardware vendors / firmware
- Lightweight similarity tasks (feature vectors)
- Transparent, flexible, hardware-independent
- Tunable threat model (semi-honest ↔ malicious)

Summary

Contributions at a Glance

PERQS

- Contributory CCTV network on a permissioned blockchain
- PERQL query language (extends FRAMEQL) + fuzzy consensus
- Novel time-travel consensus for blind spots & camera faults
- Tamper-evident video via on-chain hash commitments

Aramid

- Privilege-separated Hyperledger Fabric (per-channel peers)
- Trusted proxy < 5% of peer+orderer LOC → smaller TCB
- Defends 8 concrete cross-channel / data-leak attacks
- ≈ 0.56% avg throughput cost; scales better with channels

MPC vs. TEE (SoK)

- Systematic evaluation: 14 MPC protocols + 3 TEE platforms
- 5 real video-analytics case studies on CityFlowV2
- 3-party replicated secret sharing (ring) = best MPC
- Decision boundary: TEE for heavy ML, MPC for trust-min.

Future Directions

Hybrid MPC + TEE frameworks

Dynamically pick the secure-execution model per workload; e.g. MPC pre-processing at the edge, DNN inference inside TEE enclaves.

Richer PERQL

Temporal, semantic, relational-join & multi-modal queries — trace objects across junctions; fuse video, LiDAR.

Real-world deployment & user studies

City-scale / campus pilots — operational challenges, regulatory compliance, transparency, accountability, fairness.

Publications

Recap of the three papers underpinning this dissertation

C1 — PERQS

A Contributory Public-Event Recording and Querying System

Arun Joseph, Nikita Yadav, Vinod Ganapathy, Dushyant Behl

ACM SoSR / SEC '23

C2 — Aramid

Data Protection in Permissioned Blockchains using Privilege Separation

Arun Joseph, Nikita Yadav, Vinod Ganapathy, Dushyant Behl, Praveen Jayachandran

COMSNETS '23

C3 — SoK (MPC vs TEE)

Evaluation of Methods for Privacy-Preserving Edge Video Analytics

Arun Joseph, Vinod Ganapathy

ICISS '25, Indore, India — December 2025

Thank You · Questions?

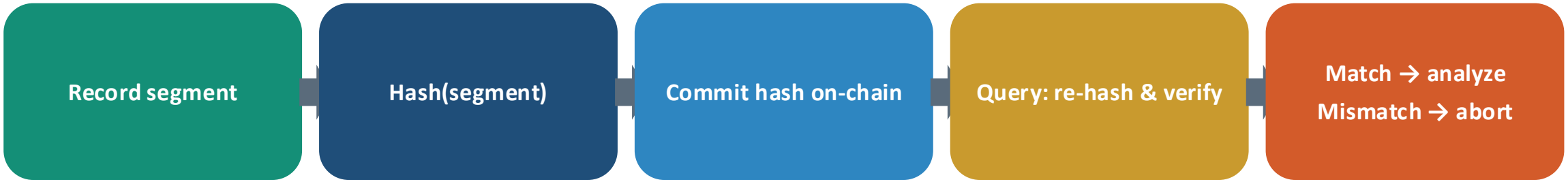
Arun Joseph

Department of Computer Science & Automation, IISc Bangalore

arunj@iisc.ac.in · +91 8547 493 535

Additional Slides

Tamper-Evidence via On-Chain Hash Commitment



- **The blockchain stores only the hash commitment — never the video itself**
- Before any analysis, the client re-hashes the located segment and checks it against the immutable on-chain value
- A hash mismatch aborts the query: retroactive alteration of stored footage becomes provably detectable
- Preserves privacy (raw video stays local) while establishing tamper-evident historical records

PERQS: Threat Model & Assumptions

Zero-trust at the participant level in a permissioned setting

Threats considered

- Integrity of recorded data: corruption via hardware / transmission / storage faults or deliberate tampering
- Malicious organizations: falsified footage or manipulated analytics to evade liability
- Byzantine behavior: selective / conflicting responses, collusion to bias consensus
- Primary threat: retroactive alteration of historical video to conceal events

Assumptions & guarantees

- Permissioned blockchain → authenticated membership + tamper-evident log
- Malicious participants ≤ tolerance threshold of the consensus protocol
- Standard crypto primitives (hashing, identity) remain secure
- Privacy: raw feeds stay local; only necessary outputs shared
- PERQS does not prevent edits — it makes them detectable via on-chain hash verification