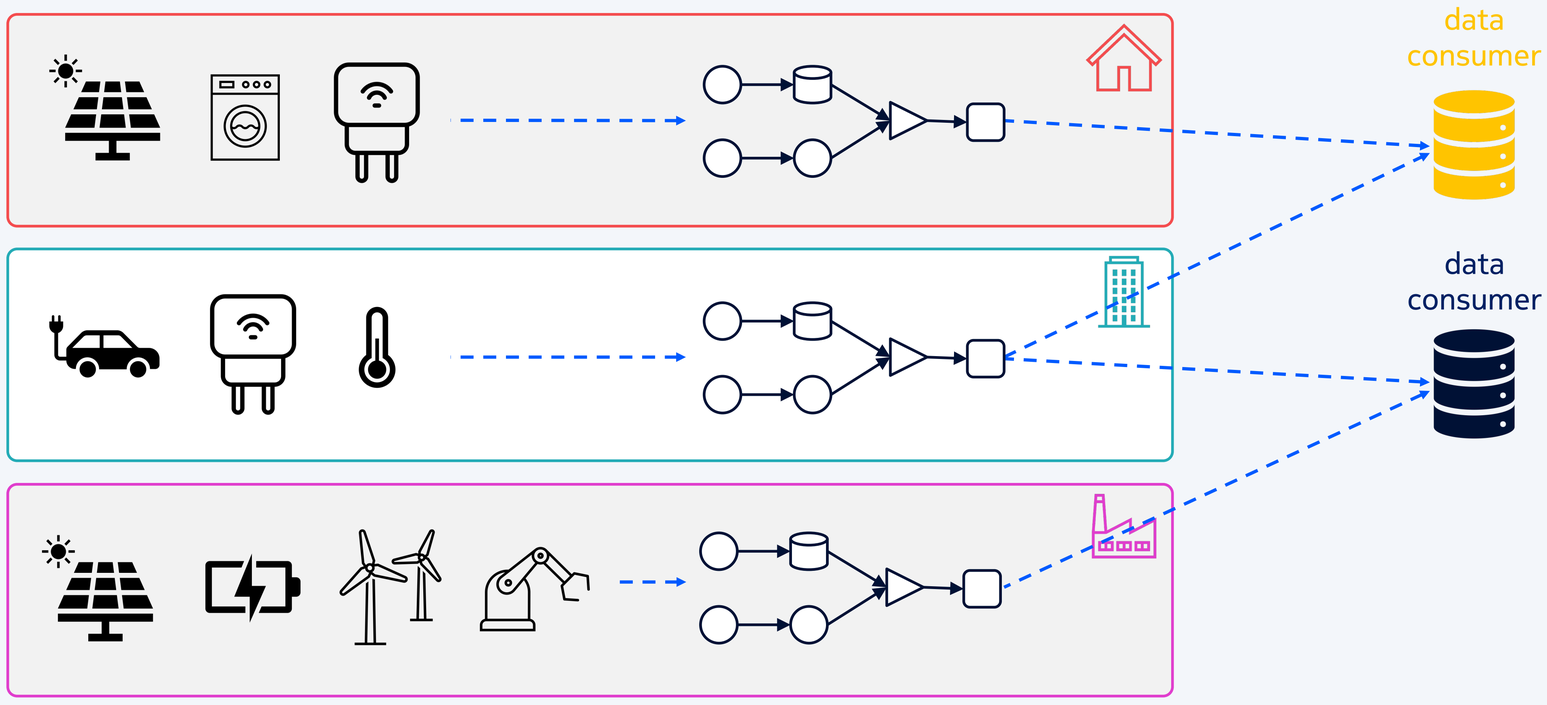
zkStream: a Framework for Trustworthy Stream Processing using Zero-Knowledge Proofs

NOKIA BELL LABS

Janwillem.Swalens@nokia-bell-labs.com
Lode.Hoste@nokia-bell-labs.com
Emad.Heydari_Beni@nokia-bell-labs.com
Lieven.Trappeniers@nokia-bell-labs.com

In IoT applications, vast numbers of sensors produce data streams.
Geo-distributed processors then execute a streaming application collaboratively:



Sensors

Installed on premise of processor
Contain public-private key pair

Sensor **signs** its output.

```
σ = sign(hash(value ++ salt), secretKey)
```

Processors

Data owner
Prover

Each operator, **in Zero-Knowledge Proof**:
1. Commits to sensor inputs
2. Commits to other inputs
3. Performs actual computation
4. Outputs result (hashed if intermediate)

```
const u32 N = 100;

def main(Key pubKey, Metadata[N] metadata, private u64[N] vals,
         private Signature[N] sigs) -> u64 {
  // 1. Verify signatures
  for u32 i in 0..N {
    assert(verifySig(pubKey, metadata[i], vals[i], sigs[i]));
  }
  // 2. Other inputs: elided
  // 3. Calculate average
  u64 mut sum = 0;
  for u32 i in 0..N {
    sum = sum + vals[i];
  }
  // 4. Return result
  return sum / cast(N);
}
```

Using the ZoKrates DSL for ZKPs

Consumers

Receives results
Provides algorithm
Verifier

Consumer must:
1. Verify all proofs
2. Verify commitments on intermediate values
3. Verify signatures

This leads to questions like:

- Is each party **computing honestly**?
- **Where is the data coming from** and where is it going?
- Can we prevent leaking **confidential or privacy-sensitive** data?

→ We can provide these guarantees automatically using **signatures** and **Zero-Knowledge Proofs**.

Trust guarantees

We provide the following guarantees:

- ✓ **Confidentiality**: Data owner does not leak sensor data nor intermediate results.
- ✓ **Computational integrity**: Data consumer knows algorithm ran as specified.
- ✓ **Provenance**: Inputs are guaranteed to come from trusted sensors.

What is a ZKP?

A **Zero-Knowledge Proof** is a technique that allows a prover (here, the processor) to convince a verifier (here, the data consumer) of the fact that a statement is true, without revealing anything else.

A ZKP can be used to prove that a **computation was performed correctly**.

Moreover, some inputs to the ZKP can be **kept secret** (`private`).

Execution characteristics

zkStream supports typical streaming processing features:

- Pipeline parallelism
- Caching
- Distribution
- Stateful operators

Optimizations

A naive implementation is too slow.
We implemented two optimizations:

1. **Lazy signature verification**
   Outsource signature verification to consumer. This means it can be done outside ZKP.
2. **Compact multi-signatures**
   Aggregate signatures into one. This reduces verification time and data transfer.

Gadgets

We created 'ZK-friendly' implementations of common aggregation operations used in streaming:

COUNT	TOP N	PERCENTILE
COUNT DISTINCT	BOTTOM N	FIRST
COLLECT DISTINCT	TOP DISTINCT N	LAST
SUM	BOTTOM DISTINCT N	LEAD
MAX	RANK	LAG
MIN	DENSE RANK	ANY
AVERAGE	PERCENT RANK	EVERY
VARIANCE	CUME DIST	BITWISE AND
STD DEV	ROW NUMBER	BITWISE OR
MEDIAN	NTILE	BITWISE XOR

E.g. median: sort outside ZK and only check inside.

https://github.com/Nokia-Bell-Labs/zkstream/tree/master/zkgadgets

Evaluation

We evaluated our approach using 6 benchmarks. The optimizations lead to 6.5–15x speed-up, showing that our approach is feasible for real use cases.

Specs: 2× Intel Xeon Gold 5318Y CPU (2× 24 cores/48 threads, 2.10 GHz), 256 GB RAM. ZoKrates 0.8.7.

Why not use TEEs?

We are running in a fully untrusted environment (e.g. a residence, wide area network, or remote industrial site) where the attacker has **physical access** to the system.
TEEs are often used in the cloud, in data centers with controlled physical access.

- TEEs have been subject to **many (side-channel) attacks**.
  → This is exacerbated in a fully untrusted environment with physical access.
- Patches require **firmware updates** and sometimes even **hardware upgrades**.
  → Tricky in our context: requires end user to update firmware or even supplying and installing new hardware.

We aim to explore ZKPs as an alternative.

- ZKPs are **software-only** and based on cryptography, not requiring trust in HW. This is interesting.
- **TCB (Trusted Computing Base) is smaller** for ZKP.

Paper:
Janwillem Swalens, Lode Hoste, Emad Heydari Beni, and Lieven Trappeniers. 2024. **zkStream: a Framework for Trustworthy Stream Processing**. In Proceedings of the 25th International Middleware Conference (MIDDLEWARE '24).

Framework and benchmark code:
https://github.com/Nokia-Bell-Labs/zkstream