

zkStream

a Framework for
Trustworthy Stream Processing

Middleware 2024 (extended version)

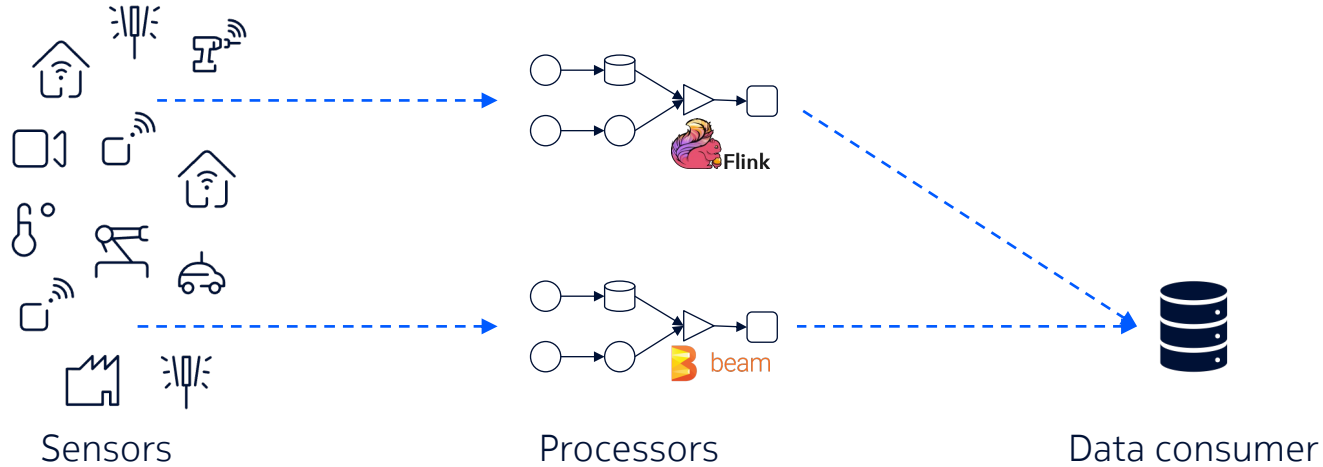
Janwillem Swalens, Lode Hoste,
Emad Heydari Beni, Lieven Trappeniers

The logo for Nokia Bell Labs, featuring the words "NOKIA", "BELL", and "LABS" stacked vertically in a white, sans-serif font. The logo is positioned inside a large, stylized circular graphic that consists of a white outer ring and a dark blue inner circle, set against a green-to-blue gradient background.

NOKIA
BELL
LABS

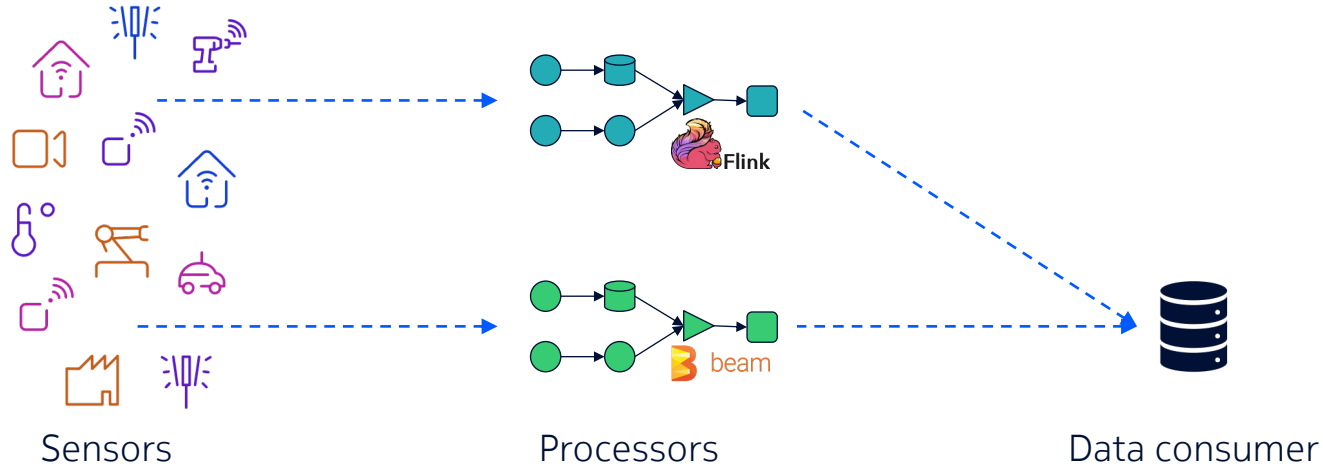
Distributed stream processing

leads to collaboration between many parties



Distributed stream processing

leads to collaboration between many parties



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

How can we automatically provide these trust guarantees?

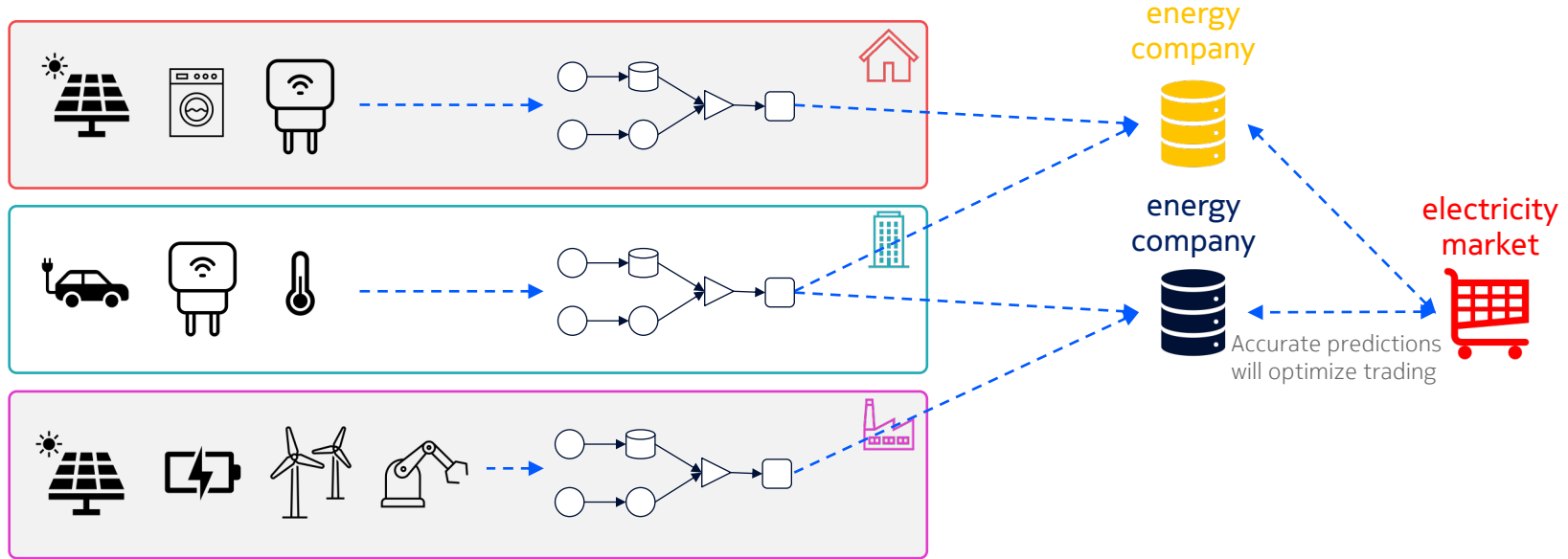
We studied three use cases

1. Electricity prediction (based on DEBS Challenge)
2. Energy orchestration (based on real use case)
3. NEXMark benchmark suite: 4 SQL-like queries for an online auctioning system

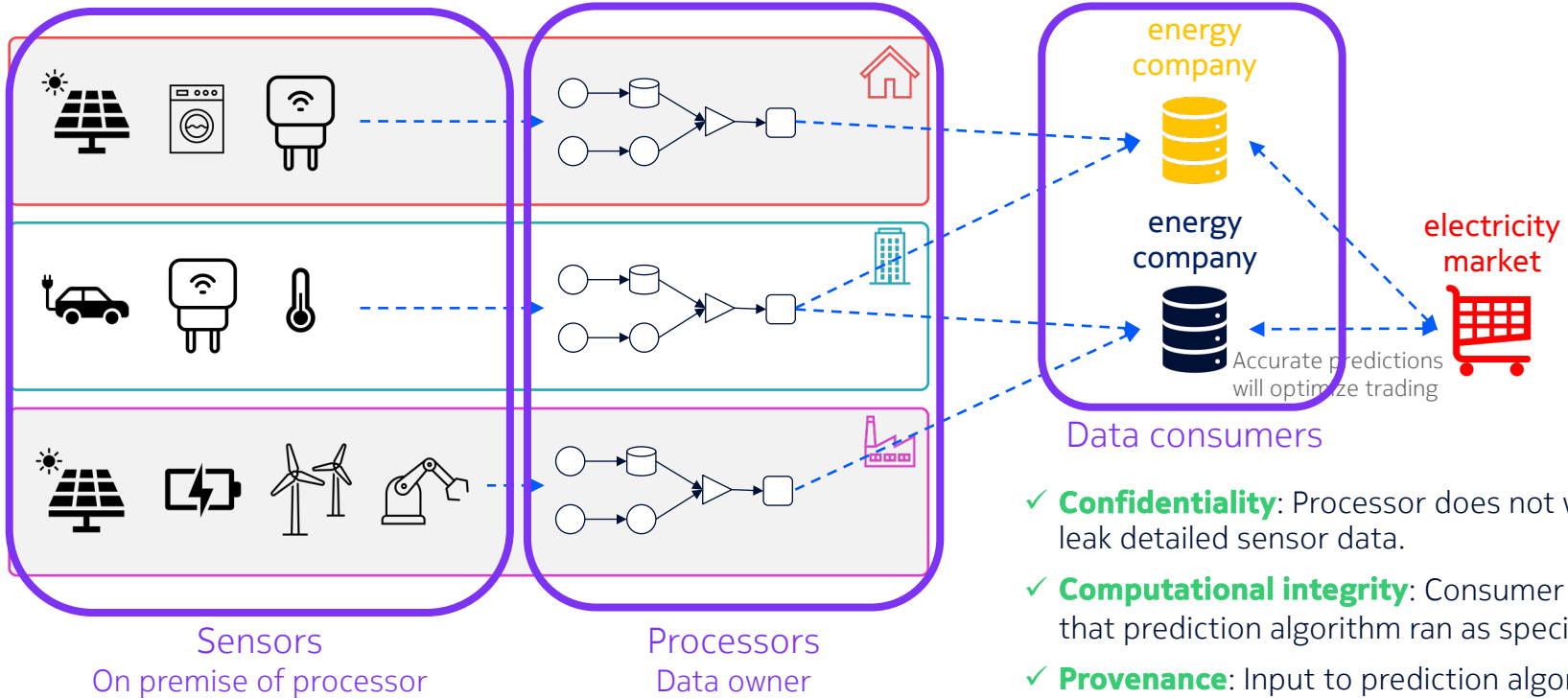
We studied three use cases

1. Electricity prediction (based on DEBS Challenge)
2. Energy orchestration (based on real use case)
3. NEXMark benchmark suite: 4 SQL-like queries for an online auctioning system

Example: electricity prediction as a distributed stream processing system



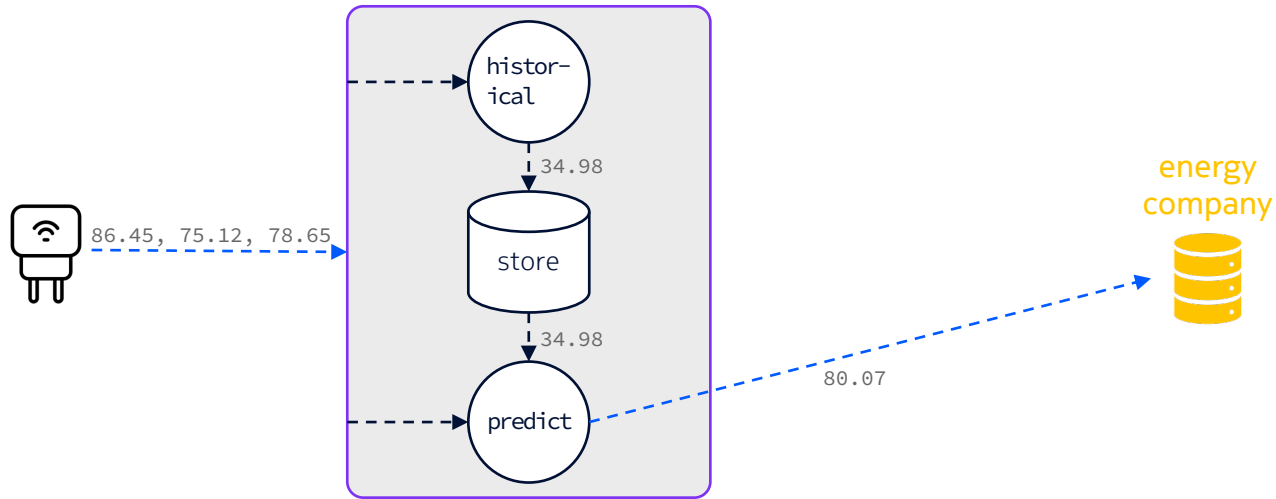
Example: electricity prediction as a distributed stream processing system



- ✓ **Confidentiality:** Processor does not want to leak detailed sensor data.
- ✓ **Computational integrity:** Consumer knows that prediction algorithm ran as specified.
- ✓ **Provenance:** Input to prediction algorithm must come from trusted sensors.

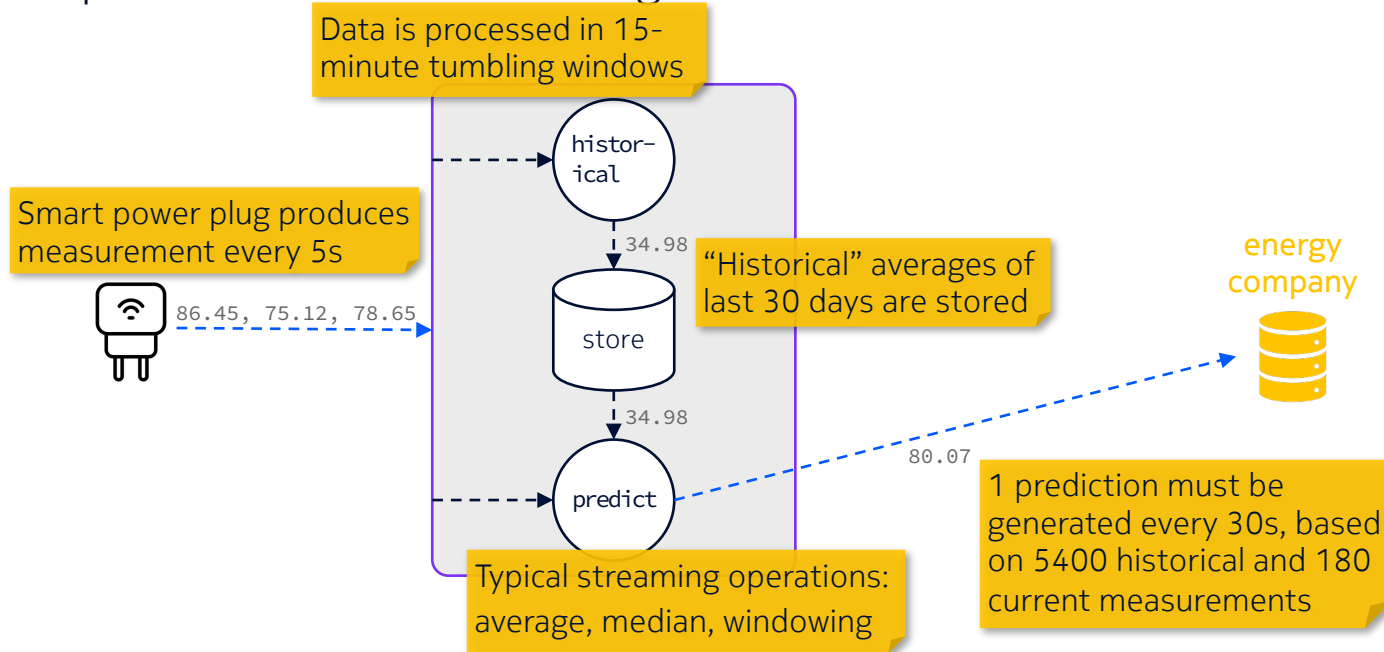
Example: electricity prediction

Specification of DEBS Challenge “Smart homes”



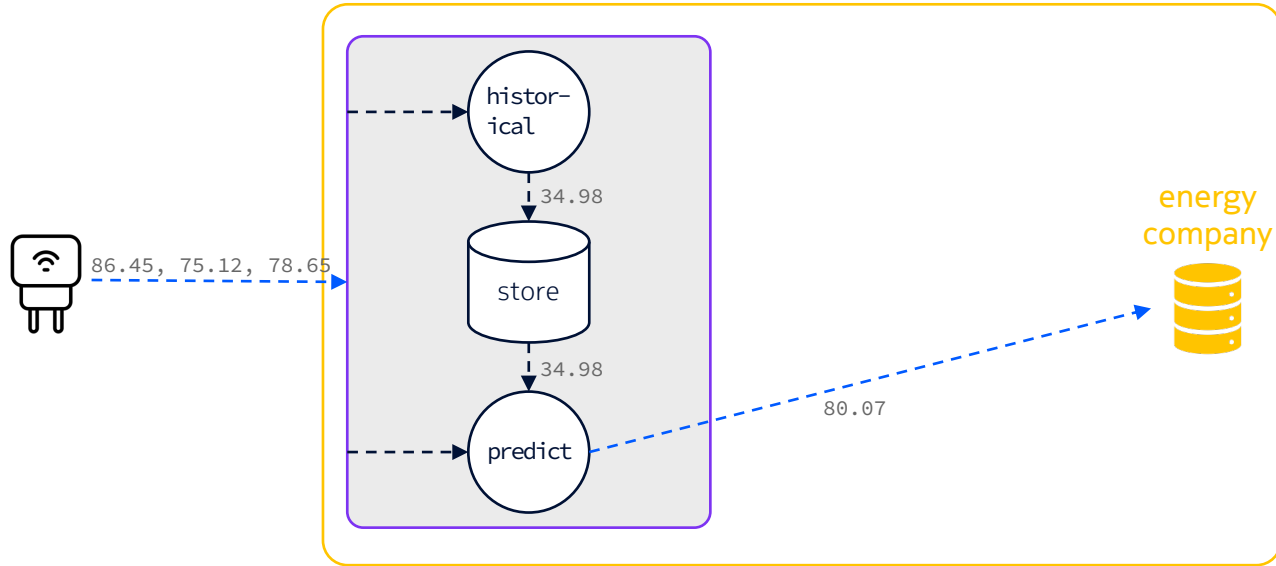
Example: electricity prediction

Specification of DEBS Challenge “Smart homes”



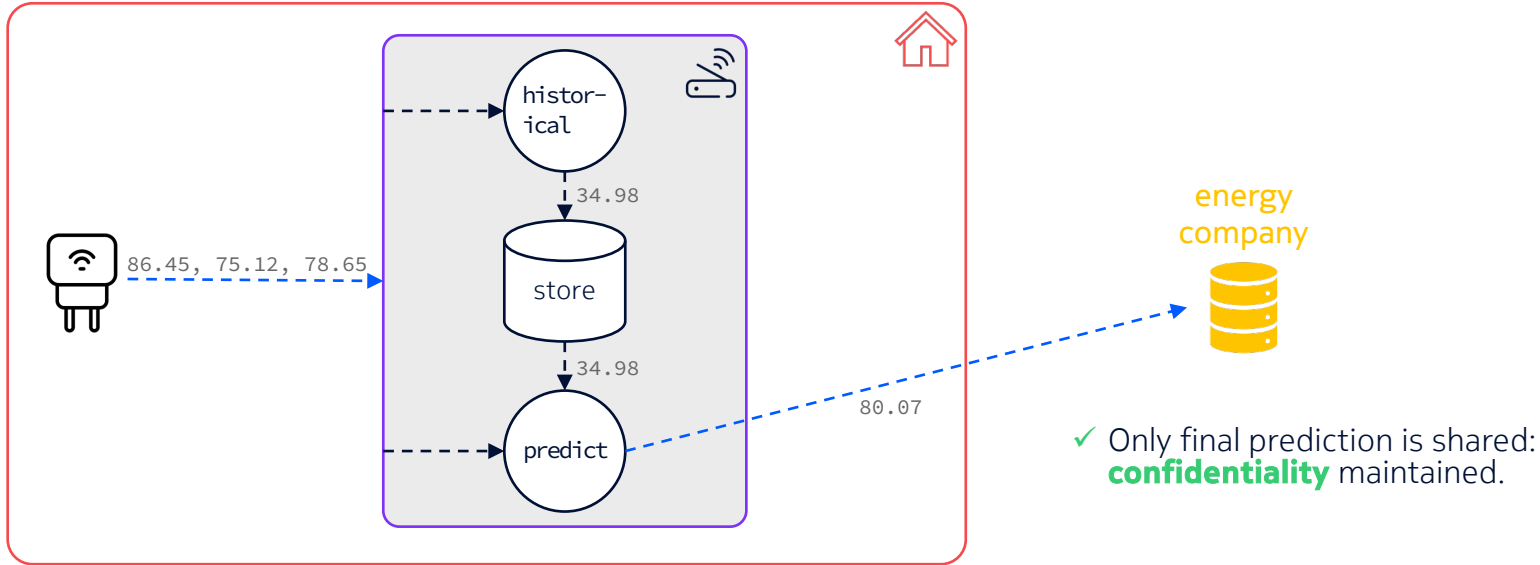
Typical streaming applications

process data in the cloud



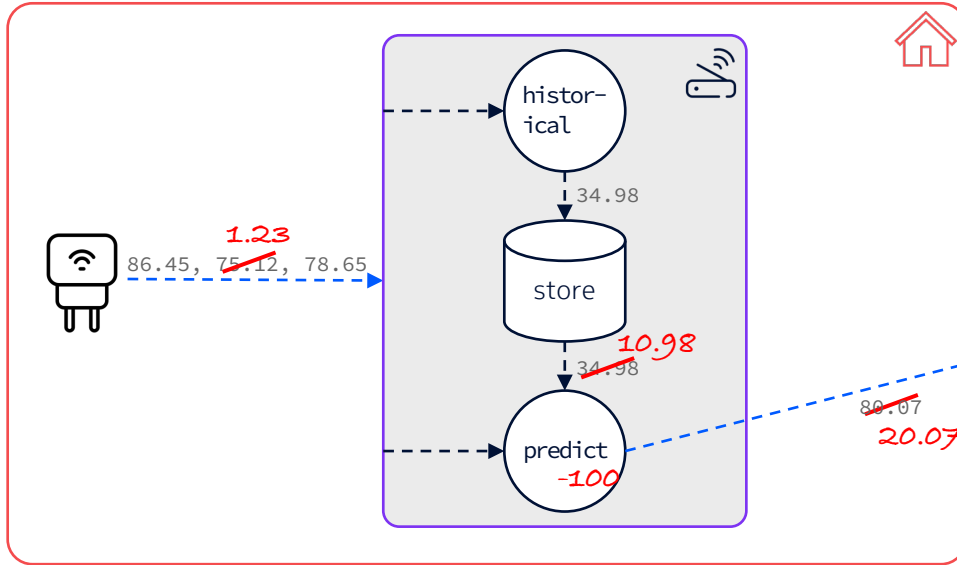
X All data is shared with energy company, but this breaks **confidentiality**.

Alternative: process the data on the edge



Alternative: process the data on the edge

This is an **untrusted** setting!



✓ Only final prediction is shared:
confidentiality maintained.

✗ Processor can tamper with **inputs**,

This breaks **computational integrity** and
provenance.

zkStream: a framework for trustworthy stream processing

1. **Signatures** on sensor data
 2. **Zero-Knowledge Proofs** for operator code
- in a streaming architecture

zkStream: a framework for trustworthy stream processing

1. Signatures

We assume sensors contain a public-private key pair.



86.45, 75.12, 78.65

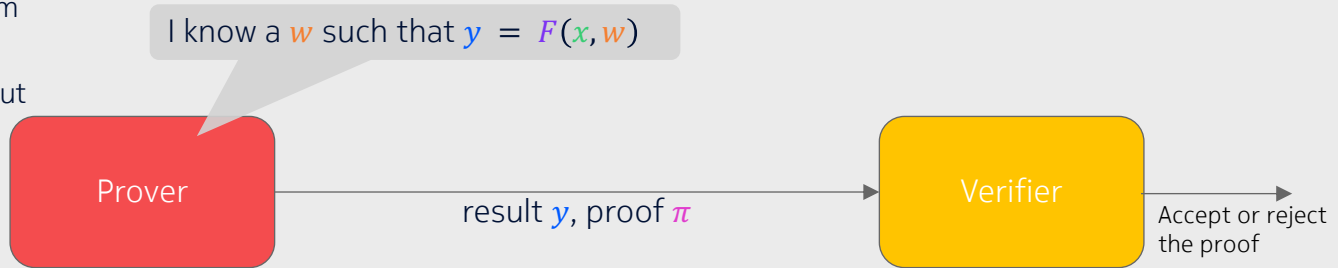
Sensor signs its output:
 $\sigma = \text{sign}(\text{hash}(\text{value}), \text{secret_key})$

To verify a sensor reading:
 $\text{verify}(\sigma, \text{hash}(\text{value}), \text{public_key})$

Zero-Knowledge Proofs to prove correct computation

A prover can convince a verifier that a statement is true, without revealing anything besides the fact that the statement is true.

F = pre-agreed program
 x = public input
 w = private (secret) input
 y = output



Zero-Knowledge Proofs to prove correct computation

Example: prove average

Program (ZoKrates)

```
const u32 N = 100;
def main(private u64[N] a) -> u64 {
  u64 mut sum = 0;
  for u32 i in 0..N {
    sum = sum + a[i];
  }
  return sum / cast(N);
}
```

compile

Set of constraints (R1CS)

```
((-1) * ~one + 1 * _1) * (1 * _4) == 1 * _3
(1 * ~one + (-1) * _3) * ((-1) * ~one + 1 * _1) == 0
(1 * ~one) * (1 * ~one) == 1 * _3
((-1) * ~one + 1 * _0) * (1 * _7) == 1 * _6
(1 * ~one + (-1) * _6) * ((-1) * ~one + 1 * _0) == 0
(1 * ~one) * (1 * ~one) == 1 * _6
...
(1 * ~one) * (1 * _6672) == 1 * ~out_0
```

6769 constraints

Witness

```
~out_0 = 50
~one = 1
_0 = 1
_1 = 0
...
_6864 = 1
_6865 = 0
```

generate prove

Proof

```
{"scheme": "g16",
 "curve": "bn128",
 "proof": {
  "a": ["0x1def914a6975111f9175dcfbc536b64122c1f74ee8742ee894e03971240995f",
        "0x23e36195308359a848127ab17c9752ae934cea958b442049ee41bfcf960d8a63"],
  "b": [{"0x0831e687bf60b170f76f8cfe8b4459c3712e15a266a0a5bcee4c5f8524dd46"},
        {"0x0caa6c13cf6535a6aa1cabb7337f66f27bd9ef2057744d00dbf6ad948348f9"},
        {"0x001d233eb8f4bb507fd8ab77337f66f27bd9ef2057744d00dbf6ad948348f9"},
        {"0x1aaada0ade26b6f5baeedbeaab0b711a2075c6853af8b49306b474c37fd0cdf"}],
  "c": [{"0x0ed1e3845fb7d175a60c3b58912727466b80d9c7859d26980c77ee81ab683c44"},
        {"0x2a619be66e620ba4985ae0a64ffc0b796fb694391cf9cf58fd2ee1a55d322c"}],
  "inputs": ["0x0000000000000000000000000000000000000000000000000000000000000000"]}
```

verify



- Limited set of (efficient) operations

(e.g. floating points operations need to be emulated; both branches of if are evaluated)

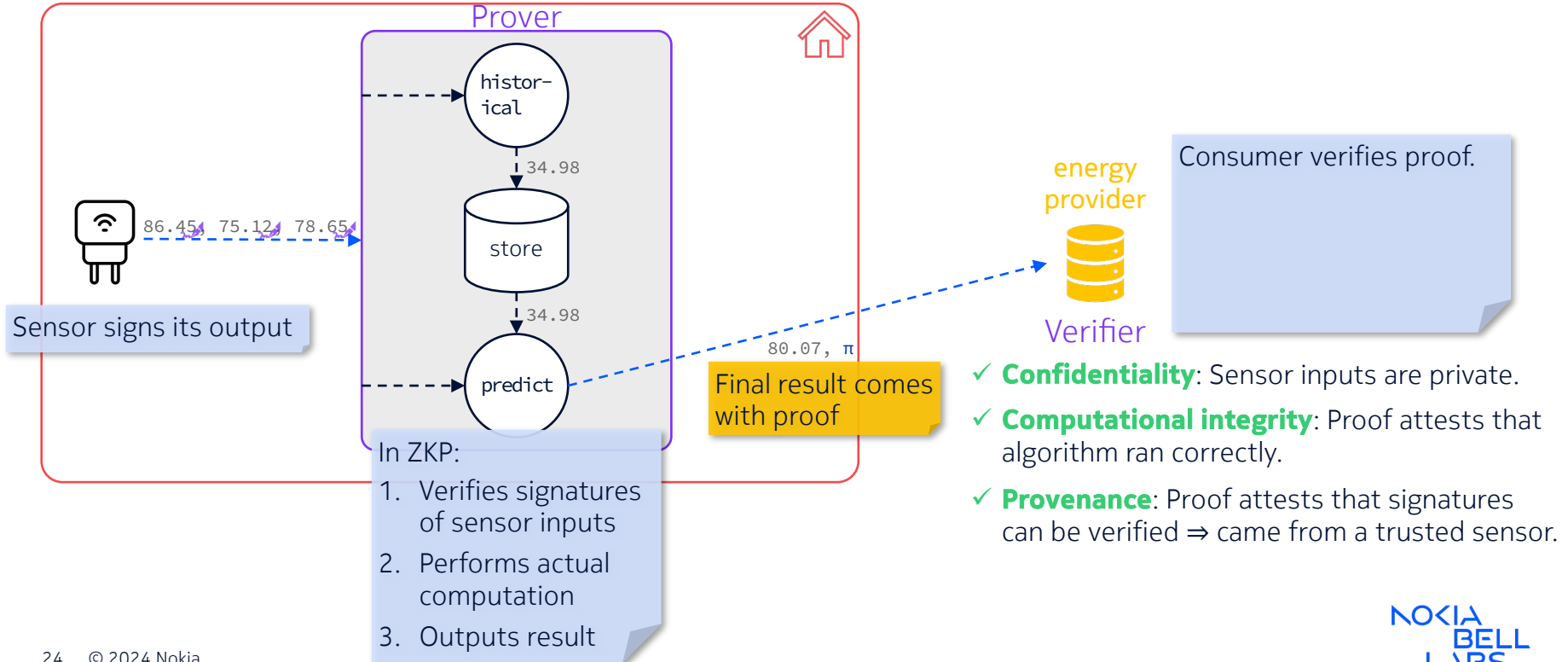
E.g. average of 1000 numbers: < 1 sec

median of 1000 numbers: > 20 minutes (+ 200 GB RAM to generate proof)

→ requires custom 'ZK-friendly' implementation

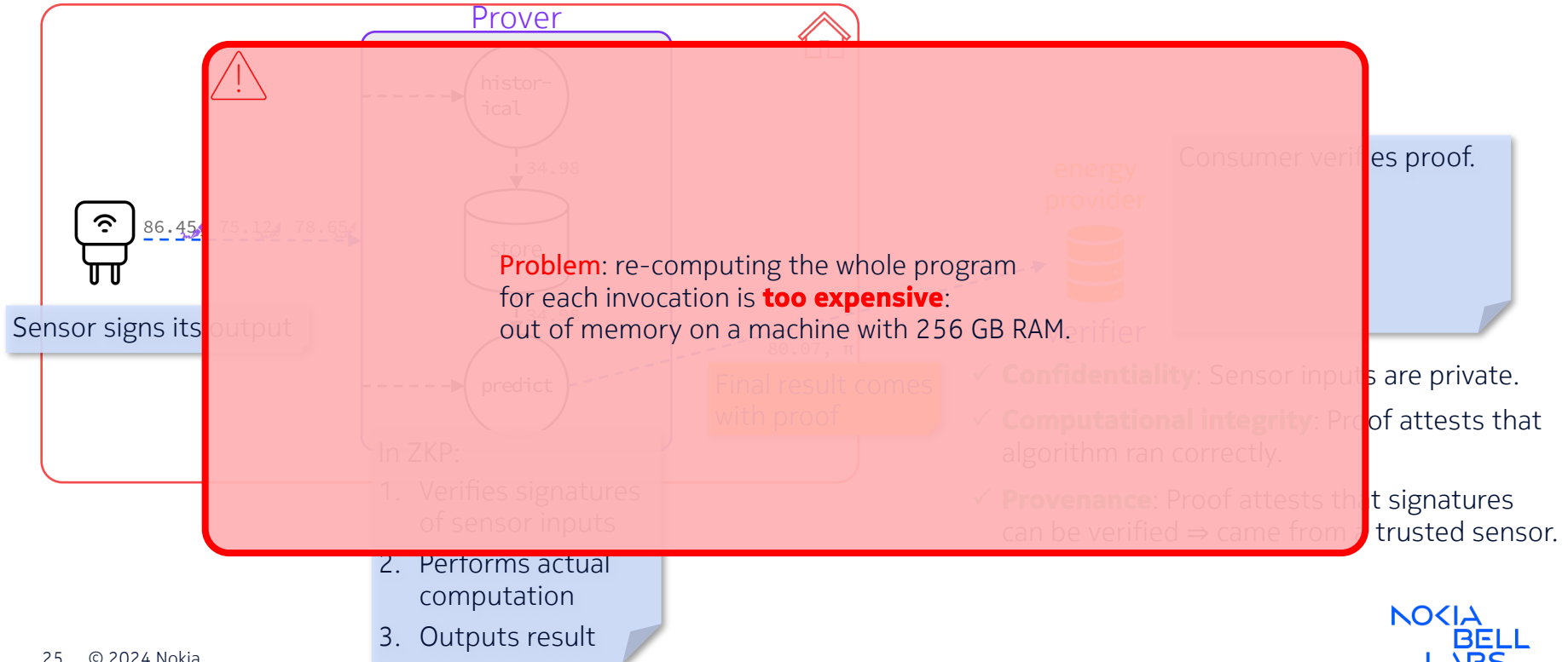
zkStream: a framework for trustworthy stream processing

2. Zero-Knowledge Proofs



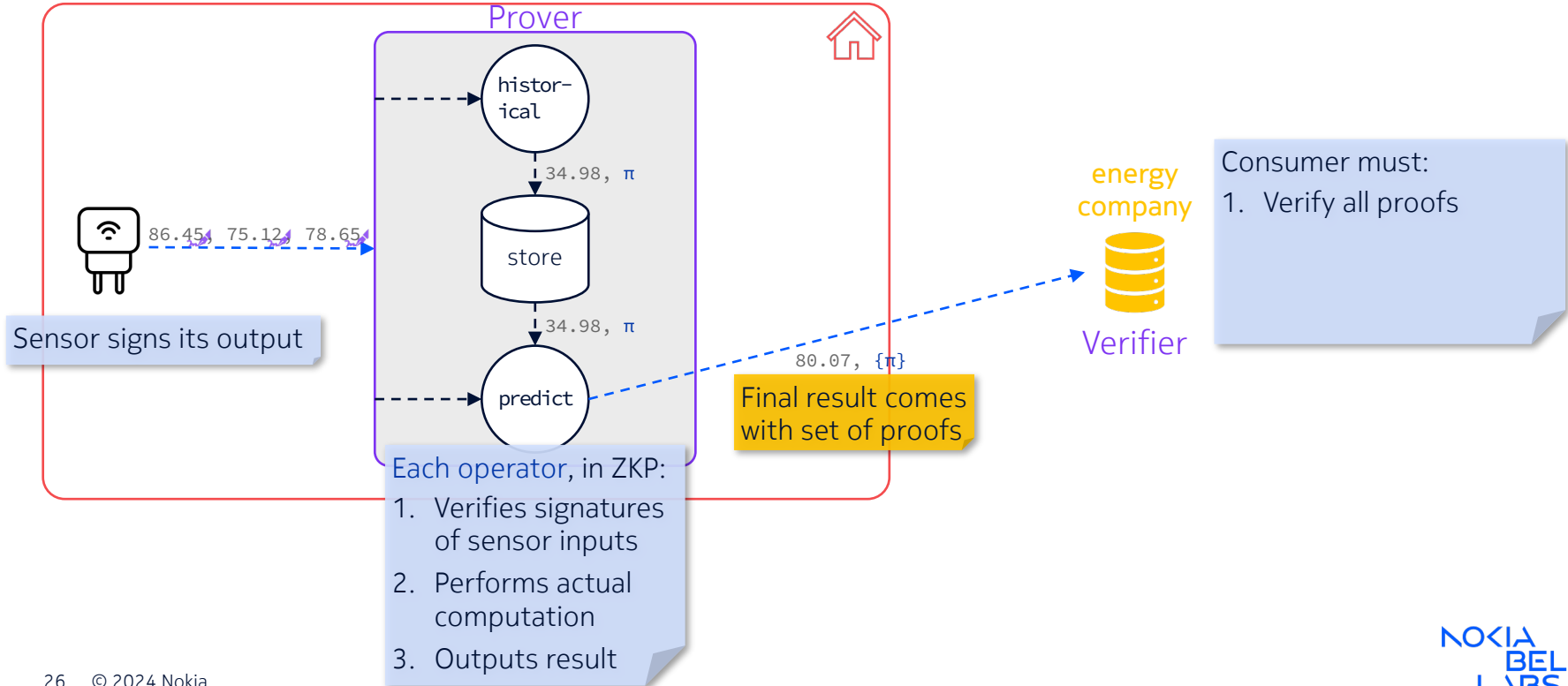
zkStream: a framework for trustworthy stream processing

2. Zero-Knowledge Proofs



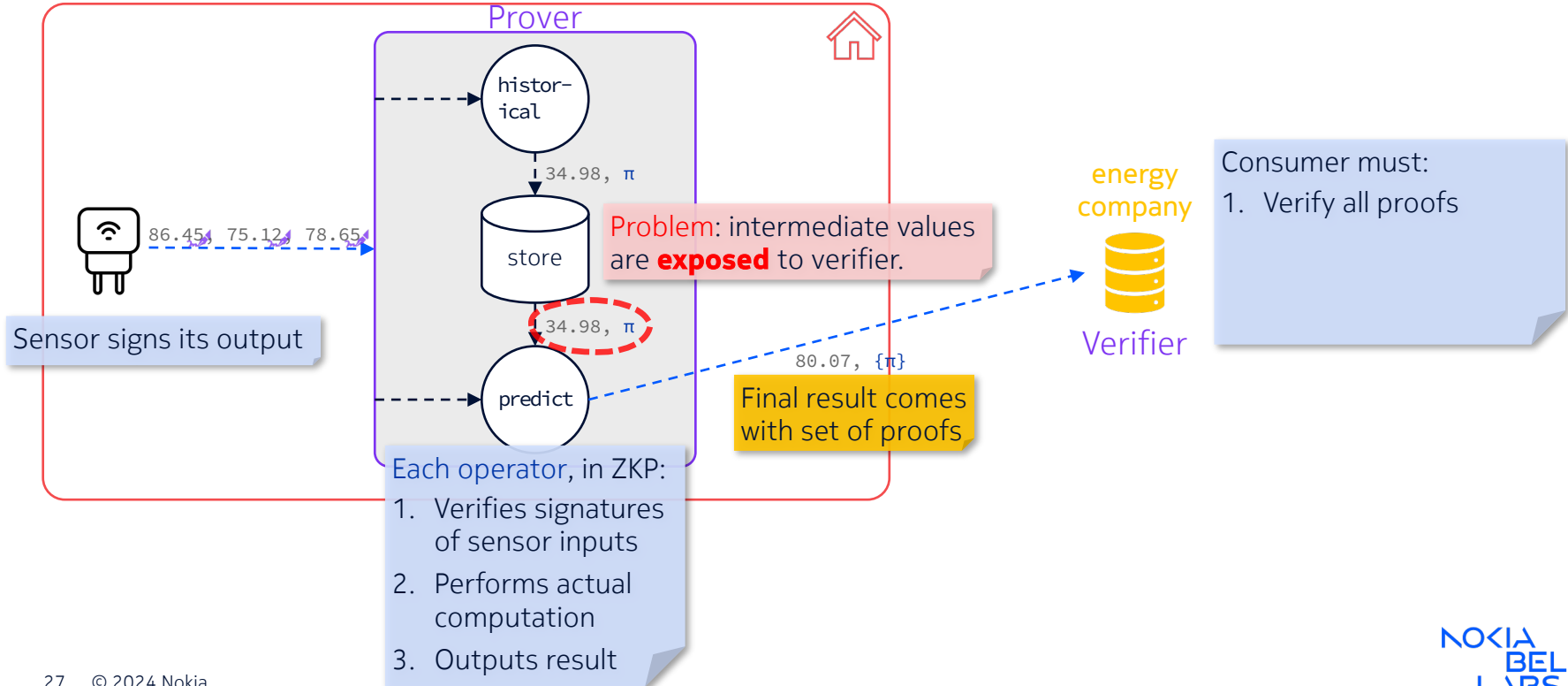
zkStream: a framework for trustworthy stream processing

2. Zero-Knowledge Proofs



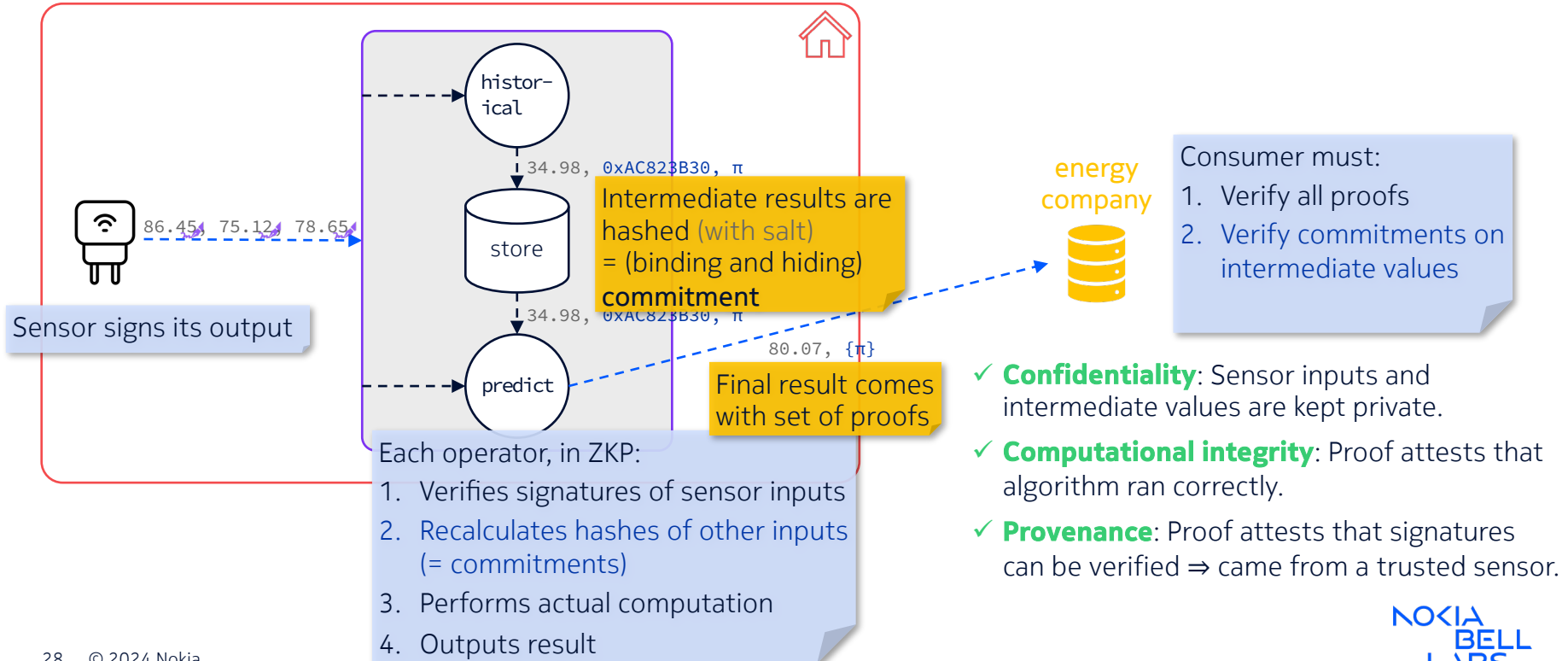
zkStream: a framework for trustworthy stream processing

2. Zero-Knowledge Proofs



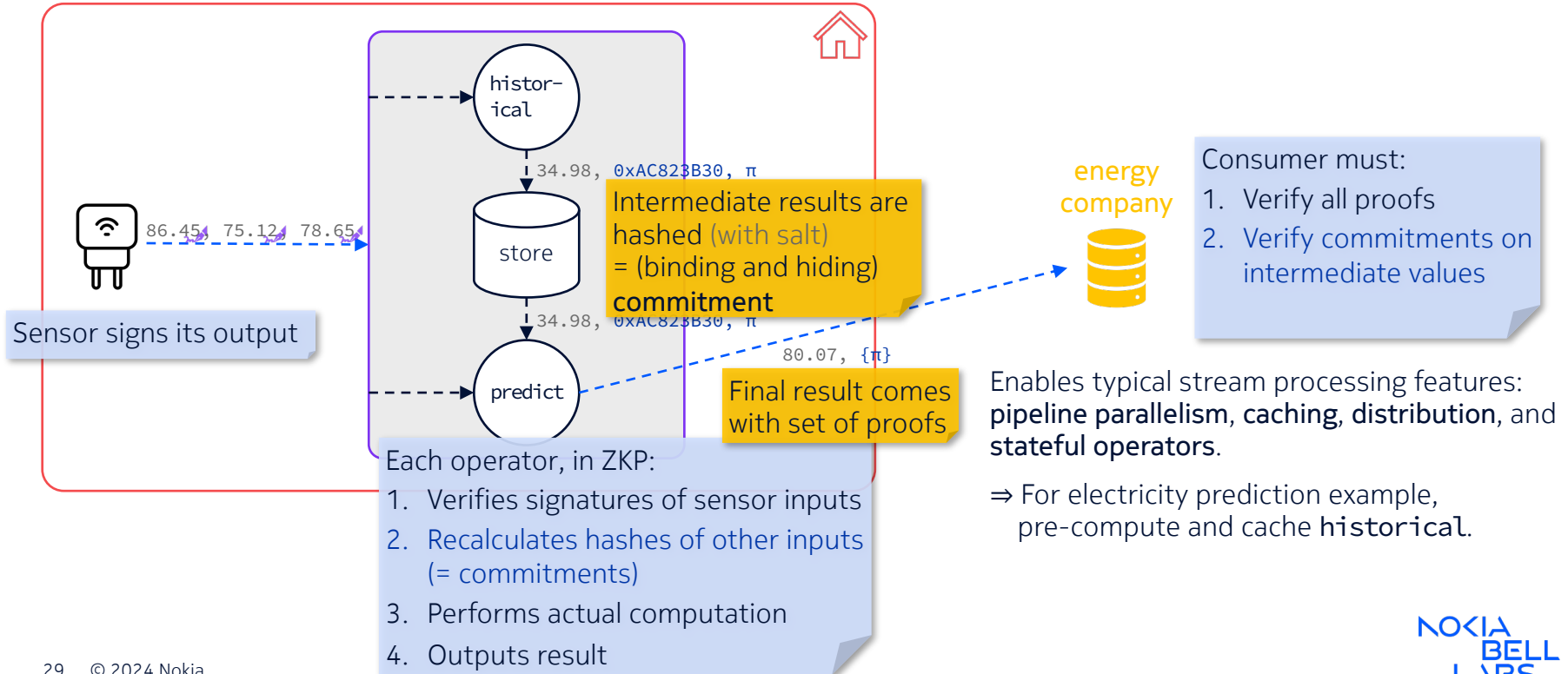
zkStream: a framework for trustworthy stream processing

3. Streaming architecture with “confidential proof chaining”



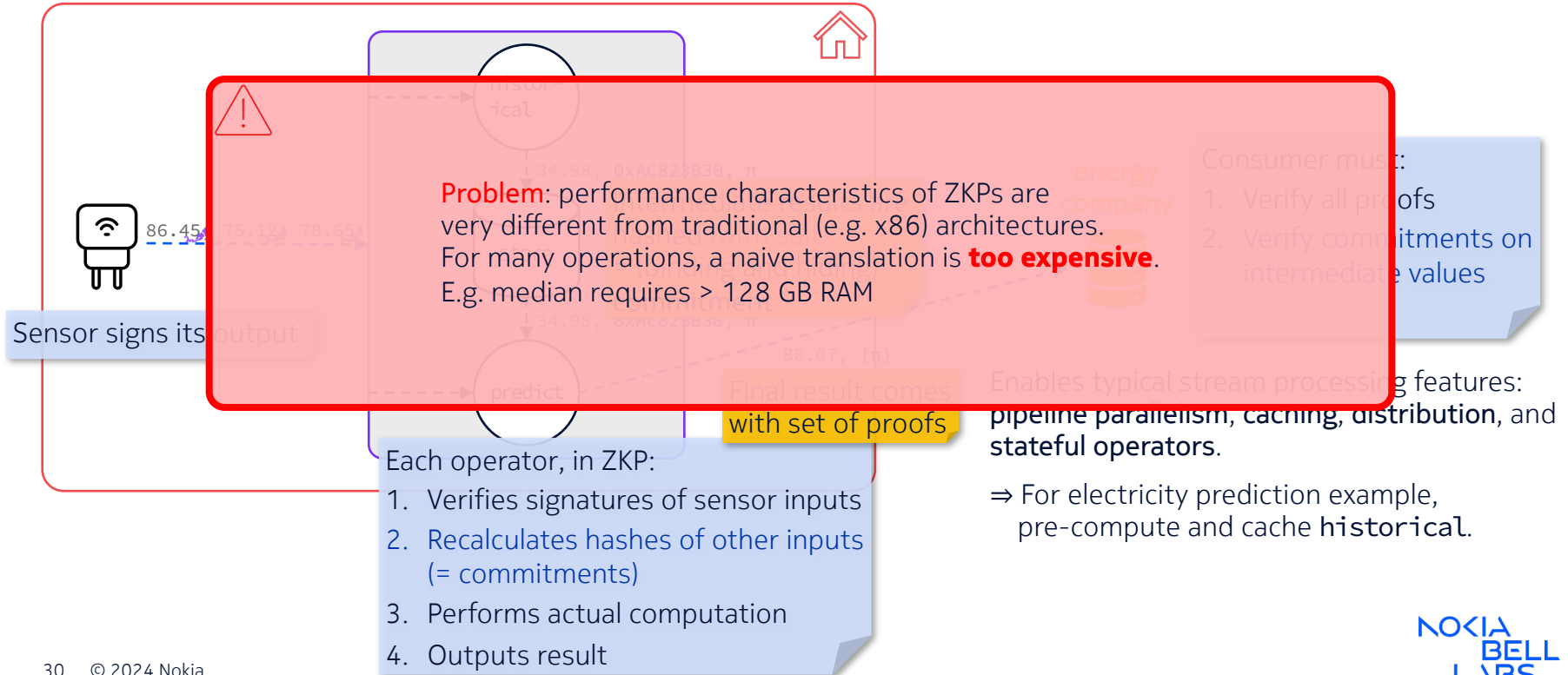
zkStream: a framework for trustworthy stream processing

3. Streaming architecture with “confidential proof chaining”



zkStream: a framework for trustworthy stream processing

3. Streaming architecture with “confidential proof chaining”



zkGadgets: ‘ZK-friendly’ aggregation operations

A toolkit for verifiable stream processing

A “gadget” is a small, specialized, and **efficient** proof **component** that can be **composed** and **re-used** as part of a larger proof. [Campanelli2019]

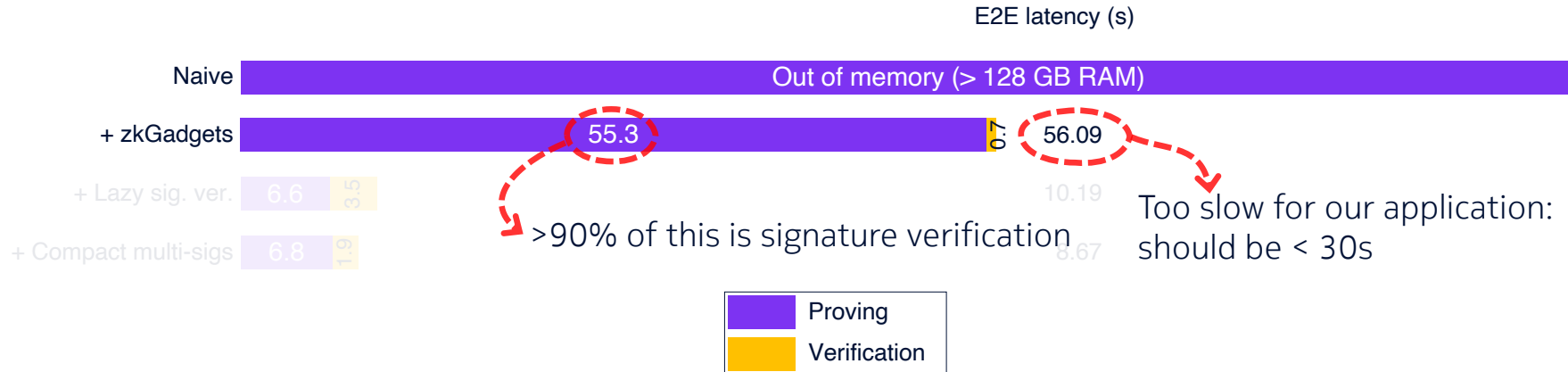
- ▶ ZK-friendly implementations of existing algorithms, e.g. matmul, SHA256 hash
- ▶ New ZK-friendly algorithms, e.g. Poseidon hash

We built a **toolkit of ZK-friendly aggregation operations for streaming.**

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

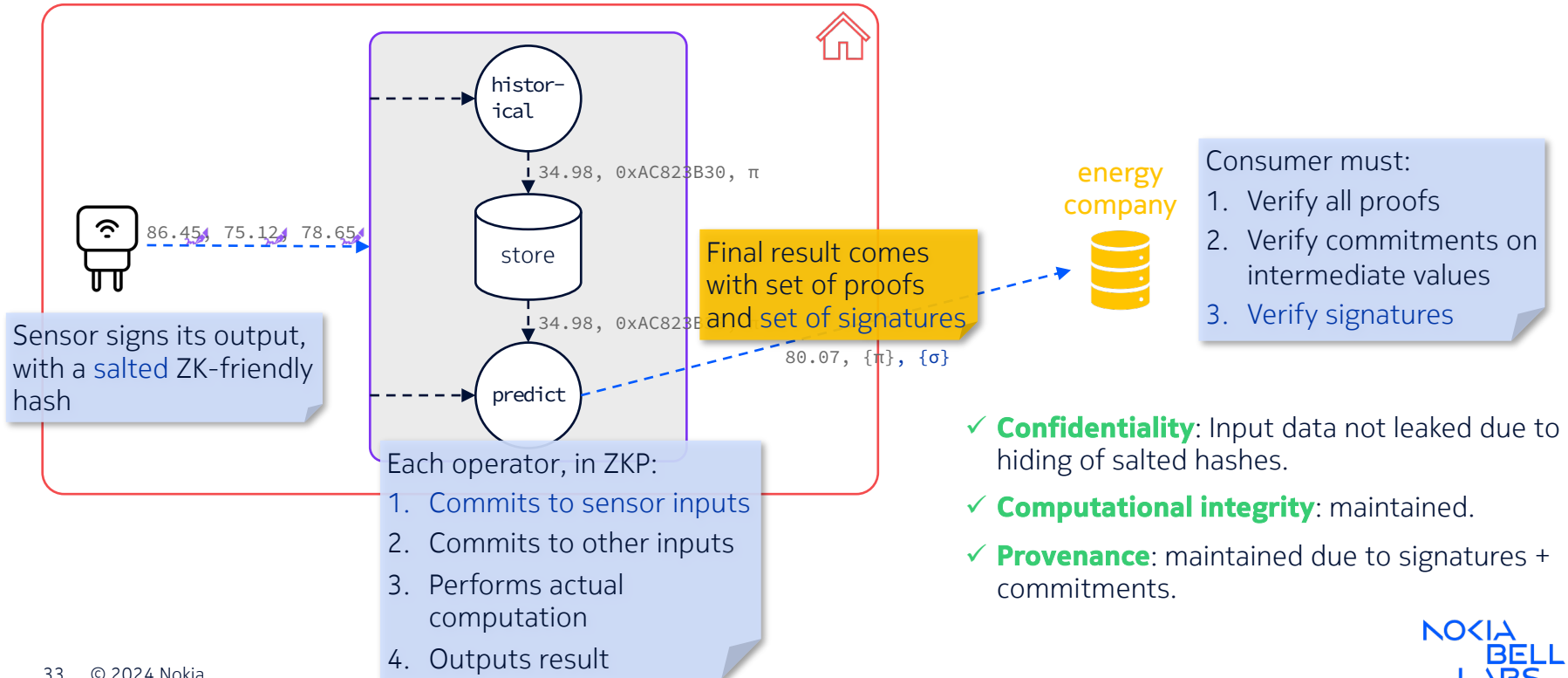
<https://github.com/Nokia-Bell-Labs/zkstream/tree/master/zkgadgets>

Performance analysis

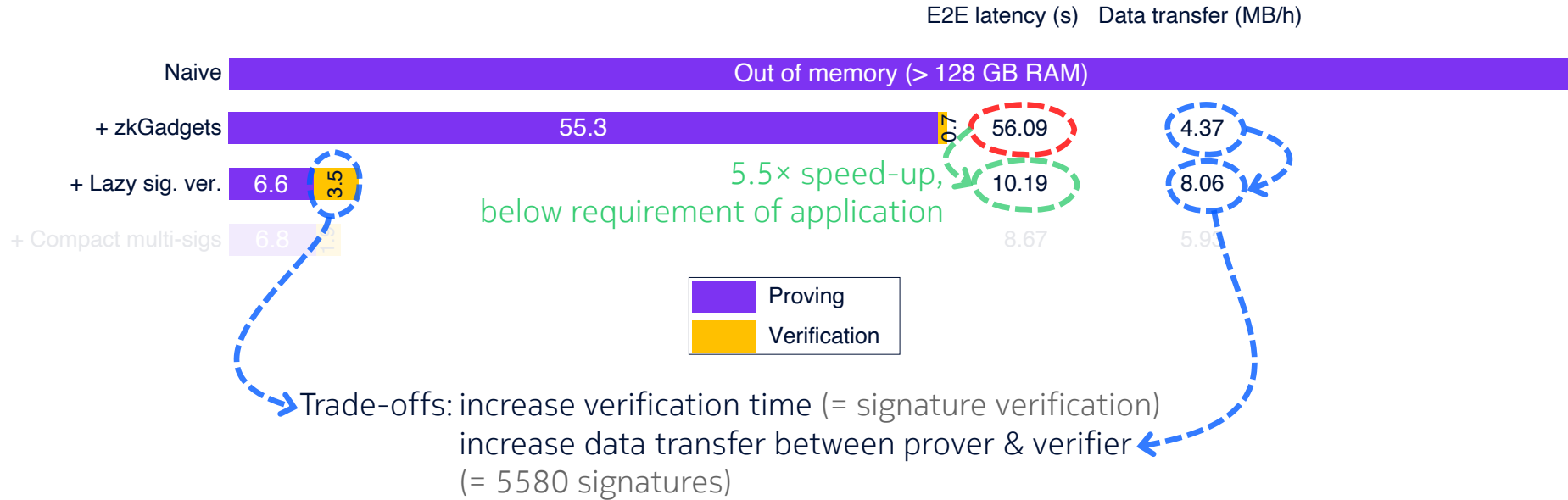


Optimizing zkStream: 1. Lazy signature verification

Outsource signature verification to verifier

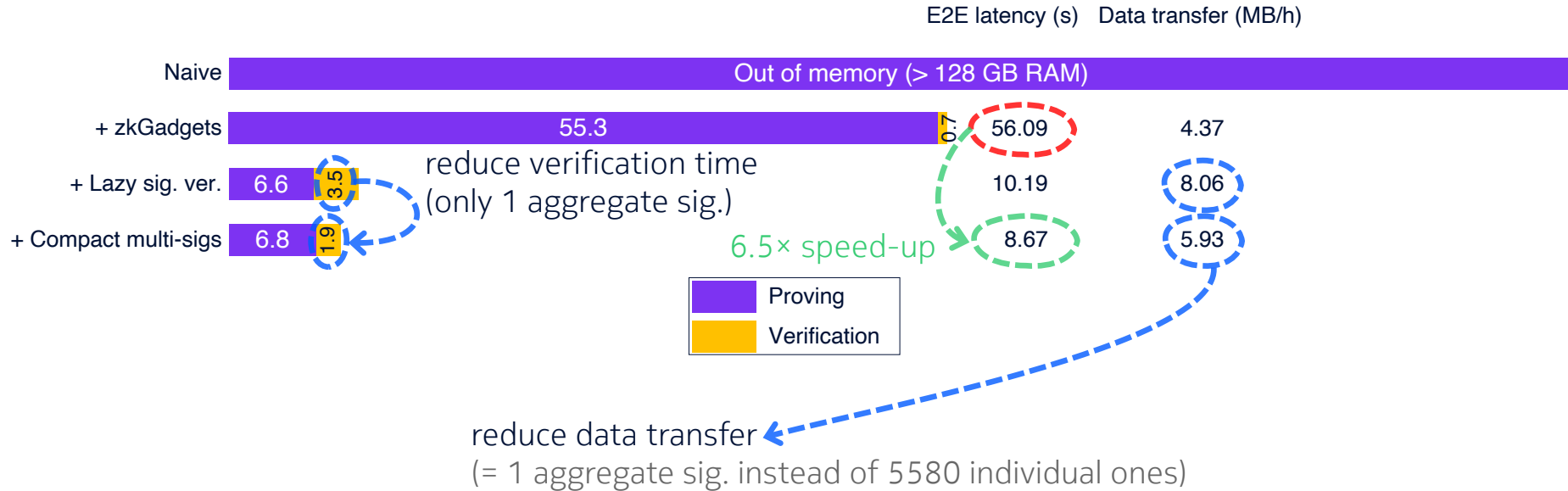


Optimizing zkStream: 1. Lazy signature verification



Optimizing zkStream: 2. Compact multi-signatures

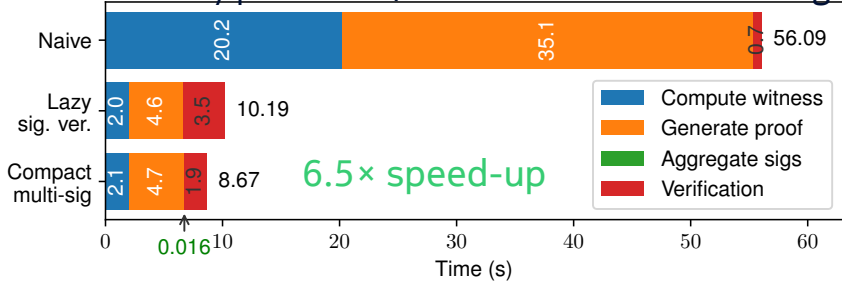
- 💡 Signature verification outside ZKP
- ⇒ no need for 'ZK-friendly' signature scheme
- ⇒ use aggregate signature scheme (BLS)



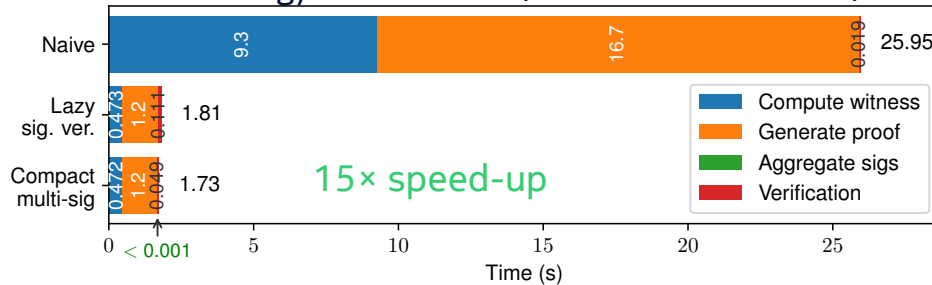
Evaluation



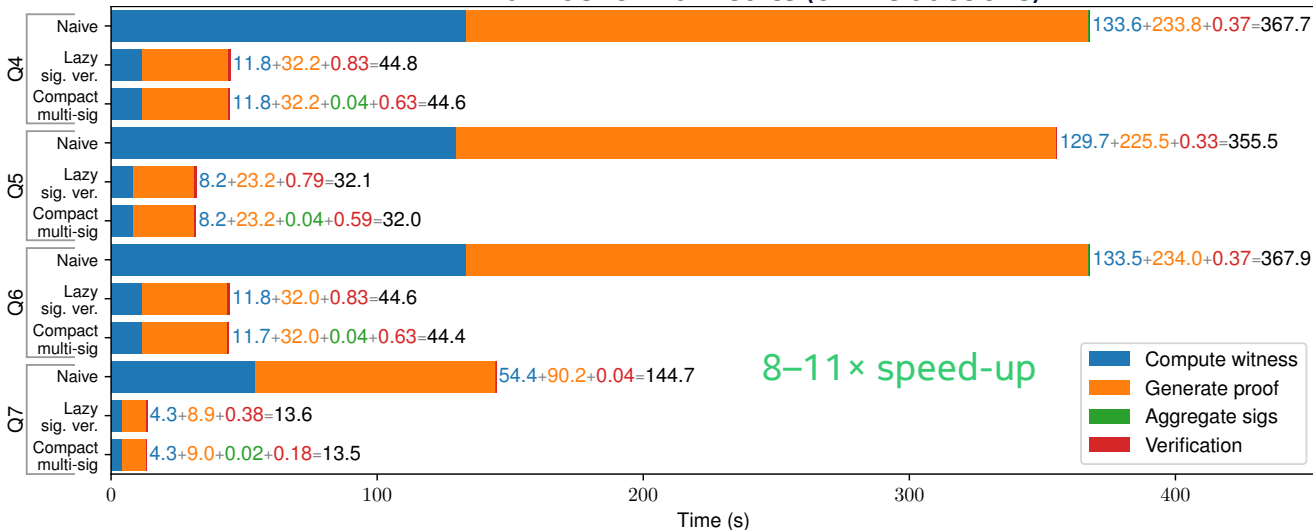
Electricity prediction (based on DEBS 2014 Challenge)



Energy orchestration (based on real use case)



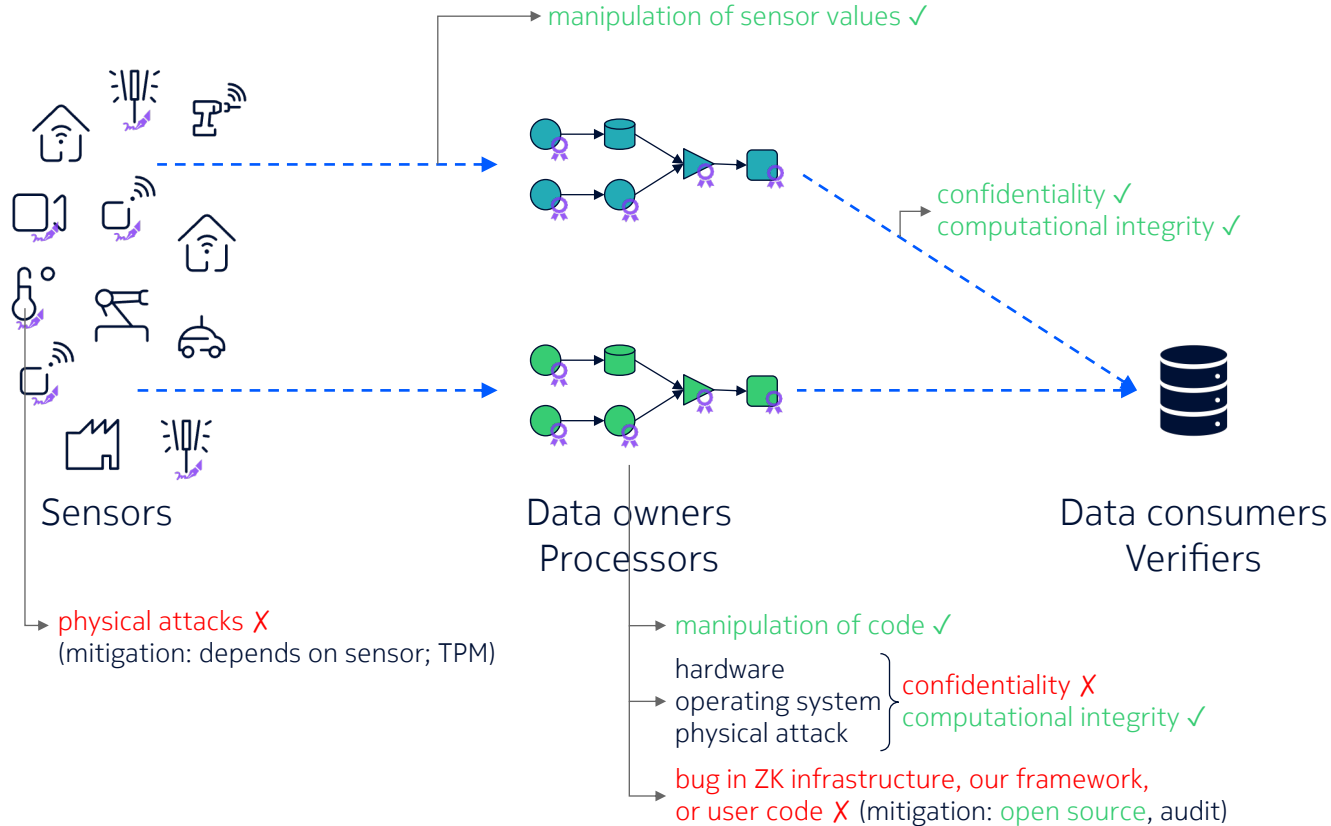
NEXMark benchmark suite (online auctions)



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



Security analysis

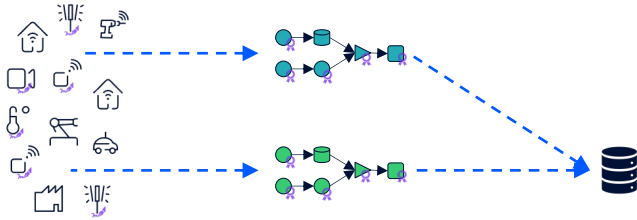


zkStream: a framework for trustworthy stream processing

A **trustworthy** stream processing framework, relying on two ingredients:

- **Signatures** on sensor data
- **Zero-Knowledge Proofs** for operator code

in an **architecture** that supports typical **streaming features** and with optimized **gadgets** for aggregation operations.



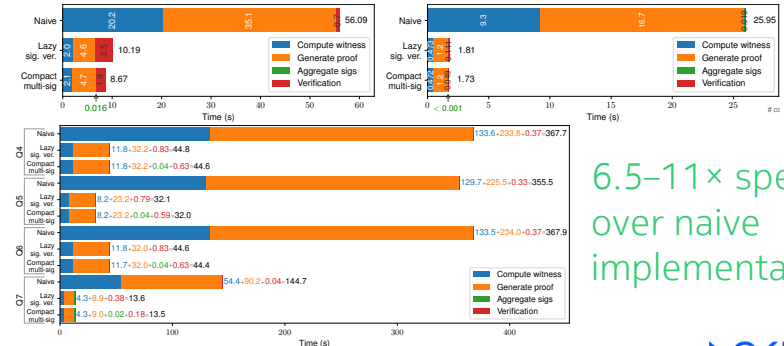
Optimizations:

1. Lazy signature verification
2. Compact multi-signatures

to make this feasible for practical applications.

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.



6.5–11× speed-up over naive implementation

NOKIA
BELL
LABS

Assumptions / Threat model

Sensors

- Limited processing power
- On premise of processor
- They function reliably
- They can contain secrets (i.e. signing keys), possibly protected using Trusted Platform Modules (TPM)

Processors

- Own the sensor data
- Want confidentiality
- Fully untrusted environment: tamper with data, act maliciously, physical attacks

(We assume a safe channel between processor and consumer, e.g. SSL)

Consumers

- Want correctness of output stream
- Should not learn raw sensor readings

Why not use TEEs?

Zero-Knowledge Proofs vs. Trusted Execution Environments

We are running in a fully untrusted environment, e.g. household, 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 **exploits**.
→ exacerbated in a **fully untrusted environment with physical access**.
- **Patches** require firmware updates and sometimes even hardware upgrades.
→ tricky in our context: would require **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. (But maybe less well tested?)

Isn't the performance overhead still huge?

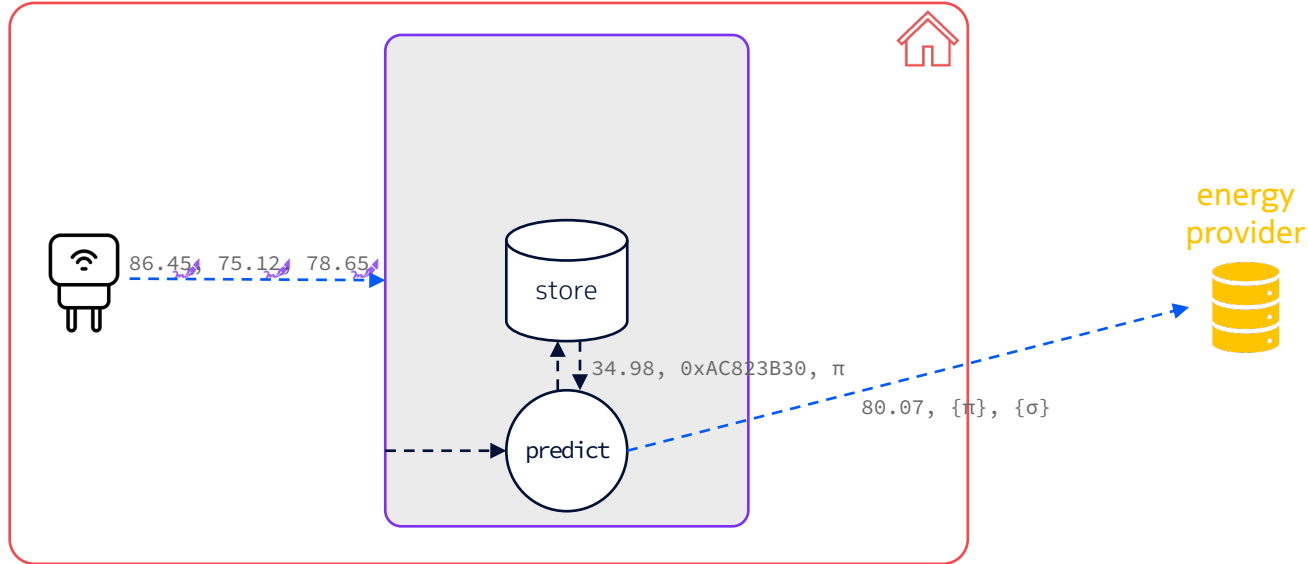
ZKPs are still a lot slower than native execution.

Some answers:

- Trend is that ZKPs are becoming faster (+ less memory hungry)
(new proving systems, proving systems optimized for e.g. binary, HW acceleration)
- Optimizations of algorithms
(e.g. 'zkML')
- Still need to be aware of ZK context

I believe it is valuable to research these applications of ZK now as they become faster.

How to support stateful operators



We already support a store, in which hidden values can be stored.

To implement a stateful operator, you can store the state in the store and re-use it on the next invocation.

(Potentially, could use incremental/recursive proving systems like Nova to avoid increasing proof size.)

Copyright and confidentiality

The contents of this document are proprietary and confidential property of Nokia. This document is provided subject to confidentiality obligations of the applicable agreement(s).

This document is intended for use by Nokia's customers and collaborators only for the purpose for which this document is submitted by Nokia. No part of this document may be reproduced or made available to the public or to any third party in any form or means without the prior written permission of Nokia. This document is to be used by properly trained professional personnel. Any use of the contents in this document is limited strictly to the use(s) specifically created in the applicable agreement(s) under which the document is submitted. The user of this document may voluntarily provide suggestions, comments or other feedback to Nokia in respect of the contents of this document ("Feedback").

Such Feedback may be used in Nokia products and related specifications or other documentation. Accordingly, if the user of this document gives Nokia Feedback on the contents of this document, Nokia may freely use, disclose, reproduce, license, distribute and otherwise commercialize the feedback in any Nokia product, technology, service, specification or other documentation.

Nokia operates a policy of ongoing development. Nokia reserves the right to make changes and improvements to any of the products and/or services described in this document or withdraw this document at any time without prior notice.

The contents of this document are provided "as is". Except as required by applicable law, no warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular

purpose, are made in relation to the accuracy, reliability or contents of this document. NOKIA SHALL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENT or for any loss of data or income or any special, incidental, consequential, indirect or direct damages howsoever caused, that might arise from the use of this document or any contents of this document.

This document and the product(s) it describes are protected by copyright according to the applicable laws.

Nokia is a registered trademark of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.