Confidential and Verifiable Telemetry using Zero-Knowledge Proofs (talk abstract)

Janwillem Swalens, Paarijaat Aditya, Lode Hoste, Emad Heydari Beni*
Nokia Bell Labs

9–10 May 2026

Zero-Knowledge Proofs (ZKP) have been successfully applied to introduce confidentiality and verifiability between untrusted parties in several domains, including decentralized finance, privacy-preserving identity management, and verifiable ML. In this talk, we will use ZKPs for a novel industrial application: to exchange telemetry data between a supplier and a customer, guaranteeing 'two-way' confidentiality of sensitive data (from customer to supplier and vice versa) and adherence to compliance policies. We will discuss our experiments and findings on a concrete use case.

In a typical telemetry scenario, as shown in Figure 1, one company manufactures and supplies devices that are installed at a customer site. The device is owned and operated by the customer, on their premises; but its firmware is provided by the supplier. The device gathers telemetry data: detailed data about its operations, including operational states, events, and logs. By sending this data back to the supplier, it can be used for fault detection and resolution, detecting and resolving security incidents, usage monitoring, predictive maintenance, future product improvements, etc.

In this set-up, the two companies have some confidentiality requirements before they are willing to exchange data. From the point of view of the customer, the telemetry data can contain confidential information: business-sensitive data like production set-ups, outputs, performance metrics, usage patterns, or personally identifiable information. Hence, the customer wants to define a **compliance policy** on acceptable data use and transformations. For example, the customer may demand that some data elements should be kept confidential, or can only be shared after anonymization or aggregation. From the point of view of the supplier, in some cases, they also desire confidentiality: detailed log files generated by the devices contain implementation details that the supplier may want to keep private. This can include administrative settings, details about the supply chain, granular internal metrics, or parameters that are considered a competitive advantage. Hence, the supplier wants to **limit access** to some portions of the telemetry data to selected parties. Data exchanged between the customer and the supplier must therefore satisfy two confidentiality properties: the compliance policy (from the customer towards the supplier) and the limited access requirement (from supplier towards the customer).

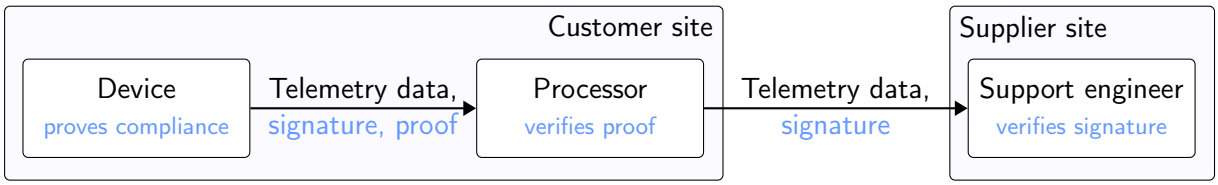

Figure 1: Typical setup

*janwillem.swalens@nokia-bell-labs.com, paarijaat.aditya@nokia-bell-labs.com, lode.hoste@nokia-bell-labs.com, emad.heydari_beni@nokia-bell-labs.com

As an example, network vendors are suppliers of networking equipment like routers, which are installed at its customers' sites, i.e. telecom companies. When an incident occurs, the router gathers telemetry data that is ultimately shared with a support engineer at the network vendor to resolve the issue. The telemetry data package contains outputs of commands reporting the router's state, its connections with other devices, logs of several subsystems, etc. We examined a specific case in which the package has a size of approximately 50 MB of text (uncompressed), corresponding to ∼1 million lines, with the output of more than 5000 commands. Telecom companies are hesitant to allow this data package to be sent to the vendor, as it contains sensitive information such as IP and MAC addresses, serial numbers of connected devices (including those of competitors), or information about their network topology. Hence, they adopt compliance policies, with varying degrees of strictness. Simultaneously, the network vendor is hesitant to share full log files with the telecom companies, as they contain internal implementation details of features that are considered a competitive advantage. This includes boot configurations, optimized configuration parameters, and metrics of internal subcomponents. In conclusion, there is a need for a solution that maintains, for both parties, confidentiality, authenticity, and integrity, to make collaboration possible.

In this talk, we will outline an approach to tackle this problem, extending our previous work [1]. We introduce a protocol that relies on (1) *signatures* to guarantee authenticity of the original data, (2) *symmetric encryption* to limit access to some portions of the data, and (3) *zero-knowledge proofs* to prove compliance to the policies and to prove integrity when transforming the original data. As shown in Figure 1 (in blue), the device generates a ZKP that proves that the telemetry data complies to the customer's policy, and the customer verifies this proof before forwarding the data to the supplier.

In the case described above, the router is a machine with 32 GB RAM and has a time budget of up to 30 minutes during which we can generate a ZKP while normal operations continue. However, even with those generous budget constraints, a naive implementation of our protocol does not lead to the desired performance. This is due to several costly operations that need to be proven (depending on the specific context):

1. The input of the ZKP consists of diagnostic information, e.g. the output of commands like `ifconfig`, `netstat`, `dmesg`, a CSV file, or JSON. This data needs to be parsed and validated. This requires string handling and dealing with data of variable length, which is challenging in many ZKP systems, especially circuit-based proving systems.
2. Our protocol relies on hashes and signatures for authenticity, and encryption to limit access to some data elements. Hashing, signature verification, and encryption are all costly operations.
3. In some cases the output is compressed in a ZIP file, which must be decompressed before its data can be read.

We will discuss several approaches and trade-offs we explored to tackle these performance issues:

- **Optimizations to our protocol**, such as outsourcing signature verification from prover to verifier and aggregating signatures (previously described in [1]).
- The development of **specialized techniques for expensive data operations**. This includes the development of 'gadgets' to efficiently handle common aggregation operations[1], 'Merkleization' of large data structures, and a technique to only decompress the relevant parts of a ZIP file while maintaining authenticity.
- An **empirical comparison of ZKP systems**: we implemented several programs using Circom, ZoKrates, and several zkVMs (RISC-Zero, SP1, Nexus, Jolt, Ligero, OpenVM)

[1]Open-sourced at `https://github.com/Nokia-Bell-Labs/zkstream/tree/master/zkgadgets`

and compared performance, expressivity, and developer effort. For example, we found that zkVMs are more suitable than circuit-based systems for string handling and parsing, and require lower developer effort, but suffer for expensive cryptographic operations like hashing, encryption, or signature verification.

Finally, we will discuss some open challenges. First, the need for further performance improvements for hashing, encrypting, or otherwise processing large amounts of data. Second, the need for compatibility with existing data formats (e.g. JSON, free-format logs) and limitations when parsing and validating these. Third, a desire for standardization and battle-testing to ensure correctness.

References

[1] Janwillem Swalens, Lode Hoste, Emad Heydari Beni, and Lieven Trappeniers. zkStream: a Framework for Trustworthy Stream Processing. In *Proceedings of the 25th International Middleware Conference*, Middleware '24, pages 252–265, New York, NY, USA, 2024. Association for Computing Machinery.