Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 769–780
DOI: 10.15439/2023F3882
ISSN 2300-5963 ACSIS, Vol. 35

An Experimental Framework for Secure and Reliable Data Streams Distribution in Federated IoT Environments

Jakub Sychowiec
Cybernetics Faculty
Military University of Technology
Warsaw, Poland
jakub.sychowiec@wat.edu.pl

Zbigniew Zieliński
Cybernetics Faculty
Military University of Technology
Warsaw, Poland
zbigniew.zielinski@wat.edu.pl

***Abstract*—An increasing number of Internet of Things (IoT) applications are based on a federated environment. Examples include the creation of federations of NATO countries and non-NATO entities participating in missions (Federated Mission Networking) or the interaction of civilian services and the military when providing Humanitarian Assistance And Disaster Relief. Federations are often formed on an ad hoc basis, with the primary goal of combining forces in a federated mission environment at any time, on short notice, and with optimization of the resources involved. One of the leading security challenges in a federated environment of separate IoT administrative domains is effective identity and access management, which is the basis for establishing a relationship of trust and secure communication between IoT devices belonging to different partners. When carrying out missions involving the military and ensuring security, meeting requirements for immediate interoperability is important. In the paper, an attempt has been made to develop a system architecture framework for secure and reliable data streams distribution in a multi-organizational federation environment, where data authentication is based on IoT device identity (fingerprint). Moreover, a hardware-software IoT gateway has been proposed for the verification process and the integration of Hyperledger Fabric's distributed ledger technology, the Apache Kafka message broker, and data-processing microservices implemented using the Kafka Streams API library. The performance tests conducted confirm the suitability of the developed system framework for processing and distributing audio-video data in a federation IoT environment. Also, a high-level security and reliability assessment was conducted in the paper.**

***Index Terms*—Internet of Things, Blockchain, Distributed Ledgers, Device Authentication**

I. Introduction

WITH technological advances in mobile radio networks, particularly concerning the implementation of 5G technology, and the development and general availability of many electronic devices equipped with a radio interface, we are seeing an increase in industrial-scale applications of the Internet of Things (IoT), occurring in both the civilian and military spheres. Examples of such applications include smart transportation, smart power grids, smart cities, or the Internet of Battle Things (IoBT) [1]. The consequence is the generation of enormous amounts of data by various IoT devices. One of the main problems attracting the attention of many researchers today is the acquisition, analysis, and fusion of these data, their secure and reliable distribution, and context-dependent information sharing. The problem is compounded in these applications when some institutions or organizations using IoT form a federation to enable different parties to cooperate. A prerequisite for effective cooperation between partners in a federation is sharing certain resources belonging to different participants and exchanging information.

One example is the creation of a federation formed of NATO countries and non-NATO mission actors (Federated Mission Networking) [2], where each actor retains control over its capabilities and operations while accepting and meeting the requirements outlined in pre-negotiated and agreed-upon arrangements, such as security policy. The main idea is to join forces in a federated mission environment at any time (zero-day interoperability), on short notice, and with optimization of the resources involved. The expected result is better command and full control of operations and decision-making through improved information sharing. Another example is the interaction of civilian services and the military, which form a federation when providing humanitarian assistance in eliminating natural disasters (HADR - Humanitarian Assistance And Disaster Relief). There are many situations in which federation partners need to exchange information, for example, about the location of each other's troops, detected threats, etc. IoT in a federation environment enhances the ability to get an accurate real-time [3] picture of the situation during an operation, e.g., by deploying mobile IoT devices such as unmanned aerial vehicles (UAVs). Hence, IoT devices operated by different federation partners must securely communicate with each other. To ensure the timely transmission of situational awareness data, the UAV may need to use a partner's resources within the communication range. In this case and many others, it is necessary to establish a trust relationship through mutual authentication of devices belonging to different federation partners, such as between a specific UAV and the partner's data distribution system.

In the case of information exchange in federated environments, where troops belonging to different NATO countries participate, or the military participates jointly with civilian services in HADR operation in an urbanized area, it is most often assumed that 5G mobile radio networks will be used as the main communication medium. To obtain a precise picture of the situation of the so-called situational awareness, there is often a need to transmit image data to other partners forming a federation. In such an environment, audiovisual data streams generated by IoT devices or city surveillance cameras (CCTV) must be considered trustworthy before they can be properly processed and transmitted to selected command posts. The primary way to confirm the reliability of such data is to authenticate the IoT devices that generate it. The presented needs for ensuring the reliability and security of data exchange bring challenges, the solution of which determines the implementation of IoT. The basic problem remains: *how to carry out the acquisition and fusion of data from various sources with different levels of reliability, and operate in computing environments with varying degrees of trust securely and reliably?* To solve it, it is necessary to know the answer to the subquestions in the first place:

- *Identity management gap* - How to manage the identity of devices? How to identify devices?
- *Security gap* - How to securely distribute data among participants in a federated environment? (Taking into account the priorities assigned to devices).
- *Network integration and interoperability gap* - How to organize interconnections, especially between unclassified systems (civilian systems) and military systems?
- *Resilience and centralization gap* - How to ensure data availability in constrained (partially isolated) environments?

Taking into consideration the aforementioned requirements for federated IoT environments and presented sub-questions, it is necessary to use and integrate multiple technologies, e.g., a data authentication mechanism where a unique identity image (fingerprint) is used, distributed ledger, 5G technology, and data distribution and processing systems. A point worth noting is that data distribution systems are often based on the MQTT protocol [4], [5]. The main disadvantage of this type of solution is the need for additional components to acquire and distribute streaming data, such as video data from daylight or infrared cameras.

This paper proposes a framework architecture for secure and reliable data streams distribution in a multi-organizational federation environment, where data authentication is based on IoT device identity. Moreover, a hardware-software IoT gateway has been proposed for the verification process and the integration of Hyperledger Fabric's distributed ledger technology [6], the Apache Kafka message broker, and data-processing microservices implemented using the Kafka Streams API library [7].

The remainder of the article is structured as follows: Section II provides an overview of the related research work that formed the basis for our solution. Section III describes the proposed framework architecture and its main elements, along with the security mechanisms used to enhance confidentiality, integrity, availability, and accountability for data in-transit and at-rest. The main operations for our experimental framework were described in Section IV. The test environment, workloads scenarios, and preliminary benchmarks with results were presented in Section V. The section also includes a high-level security risk assessment considering several security and reliability threats. Section VI presents conclusions and planned future work.

II. Background and Motivation (Related Work)

In this section, we will present related works that have had the greatest impact on the proposed framework architecture for secure and reliable data streams distribution in Federated IoT Environments. These works address the basic problems related to:

- securing data processed by IoT devices with the usage of blockchain technology;
- unique IoT device identification based on the distinctive features (fingerprints);
- the integration of heterogeneous military and civilian systems based on IoT devices, where the requirement for zero-day interoperability must be ensured.

Additionally, at the end of this section, we have briefly discussed our solution against the analyzed works.

A. Blockchain integration with the Internet of Things

The literature presents numerous attempts to integrate the Internet of Things and blockchain (distributed ledger) technology. The work [8] describes the challenges and benefits of integrating blockchain with the Internet of Things and its impact on the security of processed data. Similarly, in the works [9], [10], where a proposal for a 4-tier structural model of Blockchain and the Internet of Things (BIoT) is presented. Guo et al. [11] proposed a mechanism for authenticating IoT devices in different domains, where cooperating distributed ledgers operating in the master-slave mode were used for data exchange. Xu et al. [12] presented the DIoTA framework based on a private Hyperledger Fabric blockchain, which was used to protect the authenticity of data processed by IoT devices. The work [13] proposed an access control mechanism for devices, which used the Ethereum public blockchain placed in the Fog Layer and public key infrastructure based on elliptic curves.

B. Unique IoT device identification using fingerprint methods

Apart from classification methods for identifying a group or type of similar IoT devices [14], an interesting area of research is fingerprint techniques [15], [16], which aim to identify a unique image of a device identity through the appropriate selection of its distinctive features. The fundamental premise of fingerprint methods is the occurrence of manufacturing errors and configuration distinctions, which implies the nonexistence of two identical devices. Subsequently, the main challenge associated with fingerprinting techniques is the selection of non-ephemeral parameters that make it possible to distinguish devices uniquely. Generally, three main fingerprint methods can be identified for IoT devices as a result of distinction:

1) hardware and software features of the device;
2) characteristics of generated network traffic;
3) characteristics of generated radio signals.

The authors of the LAAFFI framework [17] presented a protocol designed to authenticate devices in federated environments based on unique hardware and software parameters extracted from a given IoT device. Concerning distinctive radio features, Sanogo et al. [18] evaluated the Power Spectral Density parameter. The work [19] indicates a proposal to use neural networks to identify devices based on the Physical Unclonable Function in combination with radio features: frequency offset, in-phase (I) and quadrature (Q) imbalance, and channel distortion. Charyyev et al. [20] proposed the LSIF fingerprint technique, where the Nilsimsa hash function was used to determine a unique IoT device network flow. In contrast, the work of [21] demonstrated the Inter-Arrival Time (IAT) differences between successively received data packets as a unique identification parameter.

C. Zero-day interoperability ensuring for heterogeneous military and civilian systems

Meeting the requirement for zero-day interoperability is a significant challenge for NATO coalition countries. Consequently, attempts are being made to integrate data exchange systems belonging to various partners to create an environment called Federated Mission Networking (FMN). For mentioned environment, NATO countries have regularly defined and revised requirements for years [2] and established research groups to identify the optimal solution for coalition data processing systems. Jansen et al. [4] presented an experimental environment consisting of four organizations between which data is distributed in two configurations. The first configuration uses two MQTT broker types (Mosquitto, VerneMQ), while the second configuration is broker-less and disseminates MQTT messages via broadcast and UDP protocol. Suri et al. [5] made an analysis and performance evaluation for eight data exchange systems used within mobile tactical networks. For the research conducted, the superiority of the DisService protocol over solutions such as Redis and RabbitMQ was demonstrated. Additionally, the work [22] proposes a data exchange system for IoT devices based on the MQTT protocol, where data is encrypted using elliptic curves. Moreover, Yang et. al [23] presented a system architecture designed for anonymized data exchange between participants using the Federation-as-a-Service (FaaS) cloud service model. The proposed system architecture was based on the Hyperledger Fabric ledger.

D. Discussion

For most of the reviewed publications, a trusted third-party infrastructure and a private distributed ledger were used to enhance the security of processed data. Compared to the work of Guo et al. [11] (master-slave chain) and Xu et al. [12] (DIoTA framework), our proposed solution is based on a single global instance of the distributed ledger. At the same time, we can freely transfer devices between organizations that are part of the federation, where these devices can use elements of another organization's infrastructure for secure data exchange. The works of [4], and [5] only address the issue of efficient data exchange and do not consider how to secure data streams. Additionally, these works did not consider the evaluation of Kafka, which also enables the handling of MQTT protocol messages. In our work, we proposed using the Kafka broker and stream processing microservices for data distribution.

Moreover, within the proposed framework, we took into account interoperability between military and civilian systems and the limited nature of such environments. Hence, for the proposed system, we have considered the recommendations made by the NATO IST-150 working group [4], which studied disconnected, intermittent, and limited (DIL) tactical networks. Our system uses a publish-subscribe model and Commercial Off-The-Shelf elements that are generally available. Subsequently, we have minimized operational costs, which is essential for ensuring immediate interoperability.

In addition, in our work, we have separated the key used to secure the communication channel for IoT devices from the key used for the data authenticity protection mechanism. Unlike the DIoTA framework, where an HMAC-based commitment scheme and randomly generated keys are used to authenticate messages, we proposed to use the device's unique distinctive features. Consequently, we proposed using a hybrid identity image defined by a combination of several fingerprint methods, primarily based on the parameters of the generated radio signals.

So far, in the publications that we analyzed, we have not noted a solution that, for the problem of secure data exchange, integrates elements of the Hyperledger Fabric blockchain, Kafka broker, and stream-processing microservices.

III. Proposed framework

This section proposes an experimental framework architecture for secure and reliable data (message) stream distribution in a multi-organizational federation environment. Figure 1 shows an example of the system structure for a federation formed from two organizations (Org1, Org2). The Apache Kafka message brokers acquire, merge, and replicate data generated by producers (publishers) and make data available to consumers (subscribers). At the same time, the proposed system enables verification of message streams based on device identity, which is stored redundantly in a distributed Hyperledger Fabric blockchain. Moreover, a hardware-software IoT gateway has been proposed for the verification process. Through which microservices using the stream processing library Kafka Streams API can communicate with the distributed ledger. A crucial aspect of the system architecture is the ability to freely transfer devices between organizations and utilize their

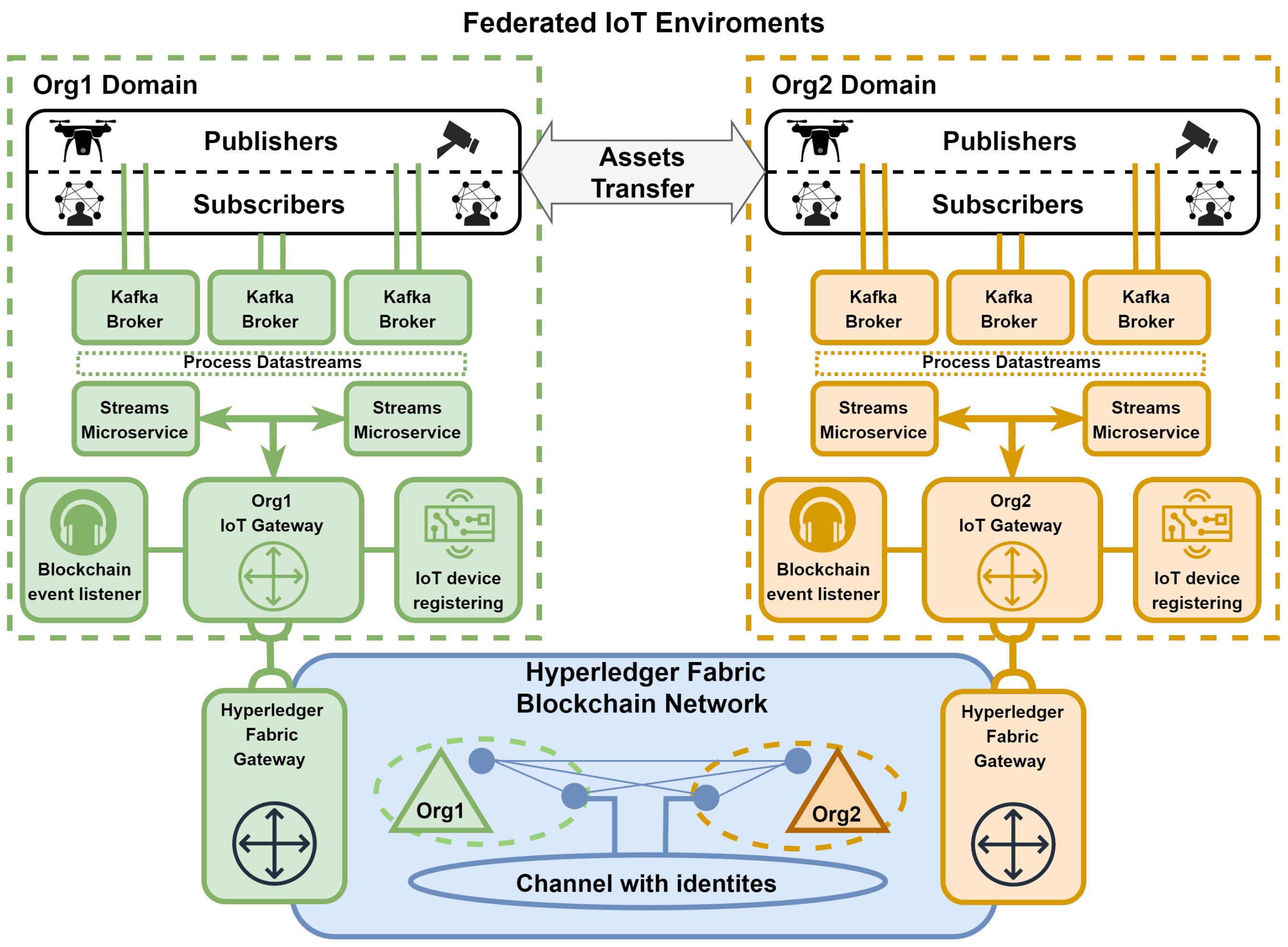


Fig. 1. Proposed framework general overview

Kafka brokers for secure data exchange. In addition, data can be exchanged in any scheme in accordance with the predefined policy, e.g., one-to-many. Furthermore, through the IoT gateway, it is possible to listen for events related to transactions of registering the identity of new devices.

Figure 2 illustrates in detail the proposed solution, where the messages generated by the producers are tagged (sealed) with their identity and then sent using the available communication medium to the broker on a specific topic (e.g., cctv-1-in). Messages sealed in this way are read from the broker by microservices and undergo a verification process. The microservice queries the Hyperledger Fabric blockchain for an image of the device identity for comparison with the identity extracted from the message. The IoT gateway handles all communication with the distributed ledger via an interface to Hyperledger Fabric Gateway services running on the ledger nodes. Successfully verified messages are saved on a dedicated topic (e.g., cctv-1-out) and provided to consumers. Messages which fail verification are discarded by the microservice or written to a previously designated topic to identify faulty (malicious) devices. As pointed out, a device identity image is used to verify the message. The identity is determined in the registration phase by using hybrid fingerprint techniques with a focus on the specificity of the generated radio signals. For a new identity image to be registered in the distributed ledger, the Hyperledger Fabric chaincode is called, which handles the transaction of adding a new identity. Successful registration of the device in the ledger is achieved by obtaining consensus among the organizations belonging to the federation. At the same time, the addition of a new identity implies the generation of an event by the blockchain, which can be handled by dedicated event listening applications (Blockchain Event Listener). In the context of the described solution, these applications were used in terms of reducing the delays associated with the processing of sealed messages. For this purpose, the event listening mechanism was integrated with the Kafka broker and microservices that use local data stores. Also, a dedicated topic (e.g., device-identity) is used for this operation. As a result of the proposed operation, the device identity image can be read from two stores: the on-chain store, called *world state* for Hyperledger Fabric, or the local off-chain store.

In the following headings, the main components of the experimental system and hardware-software IoT gateway are described, along with the reasons for the selection of the proposed elements. Additionally, the description of the various components includes their built-in security mechanisms that enhance confidentiality, integrity, availability, as well as accountability for data in-transit and at-rest.

A. Hyperledger Fabric blockchain

As part of the proposed system architecture, the Hyperledger Fabric solution was chosen. The work [8] provides a performance comparison of various distributed ledger technologies and consensus protocols. In the context of integration with the Internet of Things, mainly Hyperledger Fabric and Ethereum solutions were pointed out as legitimate due to the overall results obtained in experimental studies. Hyperledger Fabric achieved a 10 000 tps transaction throughput [8], for Ethereum throughput was lower. However, only the Proof of Work consensus protocol was benchmarked, and Proof of Stake that is currently used for Ethereum was not included in tests.

Hyperledger Fabric technology is a permissioned blockchain that uses the Practical Byzantine Fault Tolerance (PBFT) consensus protocol. For protocols of this type, all parties must know each other. As a consequence, the Fabric ledger uses public key infrastructure (certificates). The execution of complex business logic (e.g., device registry) is possible by calling multilingual chaincode (Go, Java, Node.js). Chaincode implements a group of smart contracts (transaction steps) and defines an endorsement policy, i.e. which organizations must authorize the transaction.

B. Hyperledger Fabric Gateway

The IoT gateway handles communication with the distributed ledger through an interface to the Hyperledger Fabric Gateway services [6], which allows for:

- performing queries to the world state store and reading the identity from it;
- registering, updating, and revoking IoT device identities from the ledger by calling chaincode;
- handling events generated as a result of approved transactions and blocks.

Moreover, a dynamic mode is proposed to use for the connection profile. This profile uses the ledger nodes' built-in mechanism to identify changes in the network topology on an ongoing basis. As a result, microservices will be able to operate reliably despite the failure of some nodes. Also valuable is the checkpointing mechanism, which makes it possible to resume event listening without losing events due to connection losses.

C. Device Fingerprint

As a part of the registration phase, the image of the IoT device identity will be defined, which will be stored within the device, the Hyperledger Fabric blockchain, and optionally in the local off-chain data store. The identity image will be used as a signing key for messages sent to the Kafka broker. The exact procedures of key management are out of the scope of this article. Consequently, only a general procedure for the mentioned key is presented.

In the registration phase, the device administrator places the device in RF Shielded chamber, which suppresses possible interference affecting the radio waves emitted by the device. Then, using dedicated software and measurement equipment, the device's distinctive features are subjected to a series of tests to define a unique identity image. In this study, a hybrid approach combining several fingerprinting methods is proposed, which is mainly based on the parameters of the generated radio signals. The rationale for this choice is:

- limitations arising from the heterogeneity of the environment and the need to maintain the mobility of IoT devices;
- devices' vulnerability to extreme environmental factors (e.g., temperature, humidity);
- autonomy from the protocols used in the network.

After defining the identity image, the next step is to add it to the distributed ledger. To do this, the chaincode is called, which handles the transaction of adding a new device. Then, an identity image is uploaded to the device. The whole procedure is performed through a secure communication channel with the distributed ledger.

Referring to the format of messages sent to the Kafka broker, Figure 3 shows the general structure for a single message that the broker supports. *Msg_key* and *Msg_value* are a sequence of bytes (binary stream) and represent the payload of a message. The broker or producer can specify the timestamp. Also, the producer can apply message compression or add metadata. The broker assigns the partition number and offset.

The described structure of Kafka messages makes it possible for the broker to accept and handle any format of data. The producer using a data serialization mechanism is the one who determines how to convert the data format of a given protocol (e.g., MQTT) into a bytes representation. Whereas the recipient, through deserialization, defines how to structure the byte string from the broker.

Due to the described Kafka message format and serialization mechanism, it is proposed to use dedicated software running on an IoT device to protect the message depending on its purpose and the required level of security (e.g., classified information). Using this software, it will be possible to:

- authenticating messages;
- signing messages with the identity image and its distinctive characteristics;
- encrypting the message;

Furthermore, this software will be implemented taking into account the parameters of the minimum classes of resource-limited devices, which are defined by RFC 7228 [24].

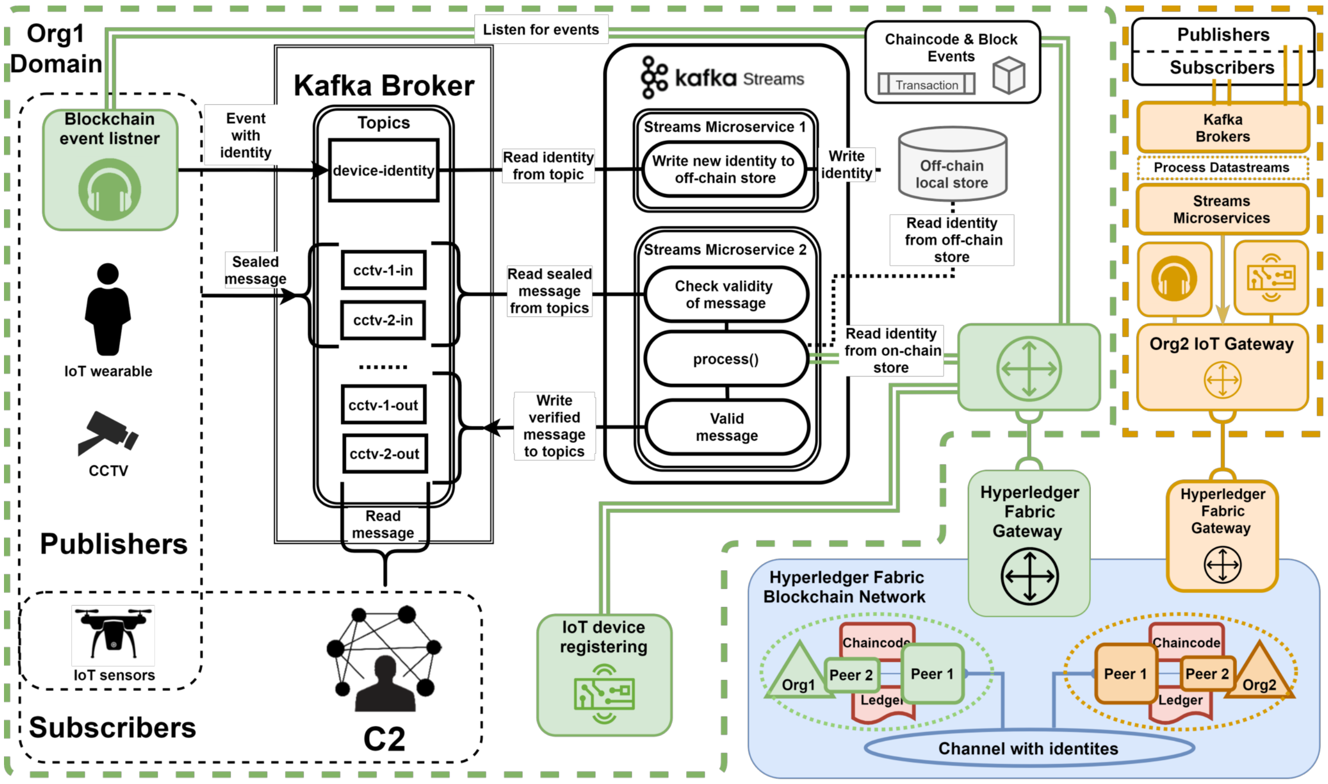
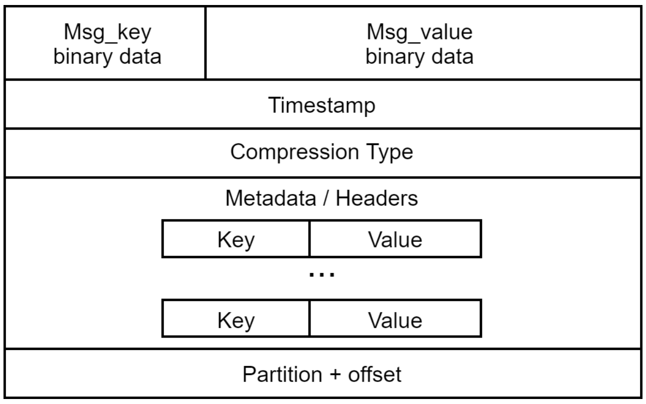


Fig. 2. Proposed framework detailed overview



Msg_key binary data	Msg_value binary data
Timestamp	
Compression Type	
Metadata / Headers	
Key	Value
…	
Key	Value
Partition + offset	

Fig. 3. Kafka message structure

D. Apache Kafka

Considering the multiplicity of data sources (devices) and the need to process messages generated by real-time systems, an Apache Kafka solution has been proposed to enable the streaming processing of data records (messages). For example, Kul et al. [25] presented a framework that uses Kafka and neural networks for tracking (tracking) vehicles, where the dataset was represented as data streams from city surveillance cameras.

Apache Kafka is based on a producer-broker-consumer (publish-subscribe) model and the classification of messages based on their topics. Due to the built-in synchronization mechanism and distributed data (registry) replication between brokers, it is possible to maintain the availability and reliability of data records. In addition, the mechanism of serialization and compression (e.g., lz4, gzip) of data records makes the proposed solution independent of the data format and the protocols used in the network (which is important for heterogeneous environments).

E. Kafka Streams API library

Performing complex operations on data records individually (stream processing) or groups of records (batch processing) requires the selection of an appropriate framework/library. Karimov et al. [26] and Poel et al. [27] evaluated solutions for processing data records. In both works, the Apache Flink framework exceeded in the overall grade other solutions: Kafka Streams, Spark Streaming, and Structured Streaming.

However, in the context of the proposed system architecture, the Kafka Streams API library was chosen, which uses the built-in primitives of Apache Kafka technology like failover and fault-tolerance. Moreover, the library uses a semantic guarantee pattern in which each record (message) is processed exactly once end-to-end. As a result, despite the failure of one of the stream processors (microservice), records will not be lost or double-processed. Spark and Structured Streaming were rejected because these technologies use a micro-batching processing technique, where aggregated data records are processed within defined time windows (threshold). Also, the Apache Flink framework was rejected since it requires a separate processing cluster, which influences operational costs for maintaining the entire infrastructure.

F. On-chain and Off-chain database

It is essential to define and distinguish two categories of data stores within the proposed framework architecture: *on-chain* and *off-chain* stores. World state and transaction log belongs to the on-chain category and refers to the Hyperledger Fabric solution. The world state is a database that determines the current state of the ledger. The transaction log is an exclusively incremental store that acts as a change data capture mechanism where approved and rejected transactions are stored. In contrast, the off-chain category refers to local data stores for applications and microservices that use the Streams API library. For the proposed framework within the on-chain category, the registered identities of IoT devices will be stored. And off-chain stores will serve as additional identity storage to reduce possible delays in the message verification process.

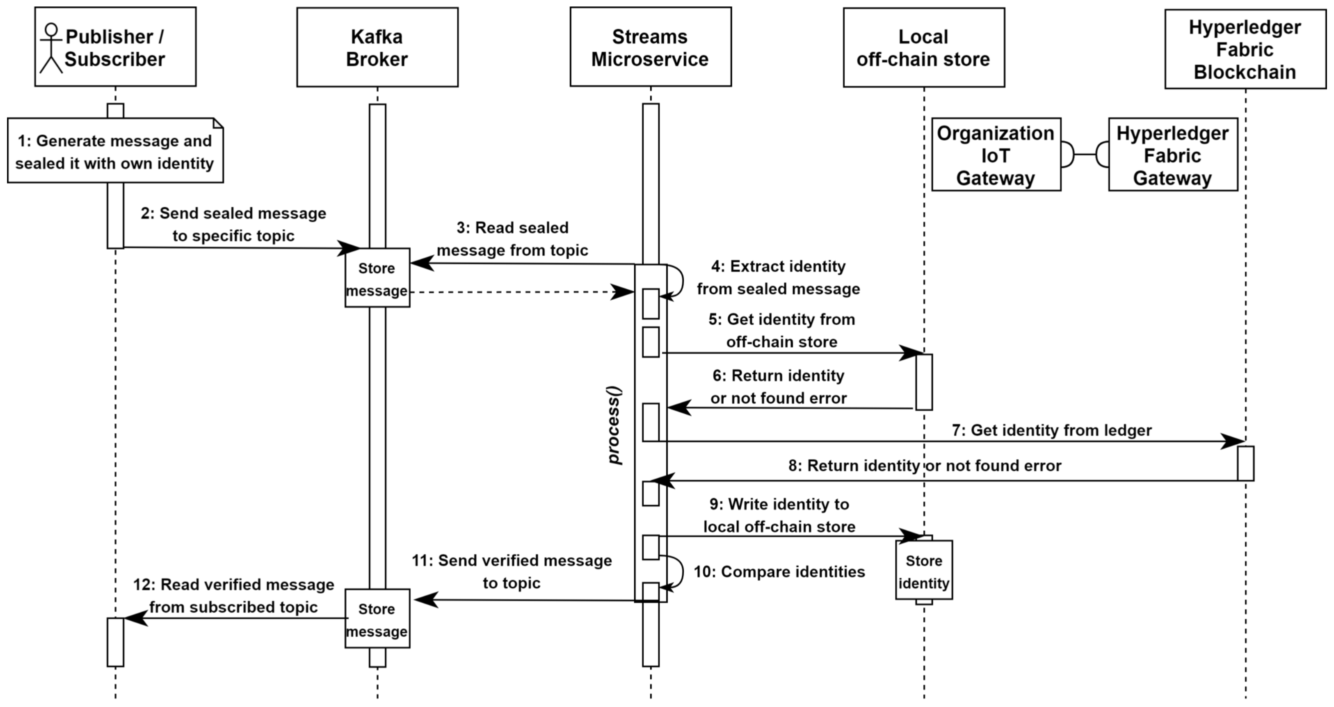
IV. Framework Basic Operations

In this part of the article, the main operations for our experimental framework were presented, taking into account relationships between system elements and message flow.

A. Verification of message streams

In order to verify messages using the distributed ledger, a custom stream processing logic was proposed. Also, a hardware-software IoT gateway was used through which microservices communicate with the Hyperledger Fabric solution. As an optional element, the usage of local off-chain data stores was included. Figure 4 shows a high-level sequence diagram where the steps and message flow are marked for data stream verification operations:

- Step 1: producer (publisher) generates a message and seals it with its own identity image that was beforehand uploaded to the device and the Hyperledger Fabric blockchain during the registration phase;
- Step 2: sealed messages are sent to the Apache Kafka broker to the specific topic using the available and secure communication channel;
- Step 3: microservices sequentially reads the messages from the topic to verify them;
- Step 4: streams microservice *process()* method carries out verification of the message, where the identity image is extracted from the message;
- Step 5: a query to the local off-chain store is made to retrieve the device identity;
- Step 6: the local data store returns the appropriate identity or an error related to its absence;
- Step 7: if identity is retrieved, step 10 is executed. Otherwise, the identity not found error results in a query for the device identity to the distributed ledger, which is executed via an interface to Fabric Gateway services;
- Step 8: the distributed ledger returns the appropriate identity or an error related to its absence;
- Step 9: identity obtained from the ledger is added to the local data store;



Publisher / Subscriber
Kafka Broker
Streams Microservice
Local off-chain store
Hyperledger Fabric Blockchain
Organization IoT Gateway
Hyperledger Fabric Gateway
1: Generate message and sealed it with own identity
2: Send sealed message to specific topic
3: Read sealed message from topic
Store message
4: Extract identity from sealed message
5: Get identity from off-chain store
6: Return identity or not found error
process()
7: Get identity from ledger
8: Return identity or not found error
9: Write identity to local off-chain store
10: Compare identities
Store identity
11: Send verified message to topic
12: Read verified message from subscribed topic
Store message

Fig. 4. Sequence diagram for verification of message streams

- Step 10: identities are compared with each other;
- Step 11: as a result of a successful identity comparison, the message is saved to the Kafka topic. The message that does not pass verification is discarded or saved to a previously designated topic to identify faulty (malicious) devices;
- Step 12: depending on the subscribed topics, the shared messages can be read sequentially by the consumer.

B. Adding identities through an event listener

The operation of adding (updating) identity to the local off-chain data store is optional and was proposed because of the possibility of reducing time delays for message verification. Figure 5 shows a high-level sequence diagram for the described operation, where:

- Step 1: the identity of the IoT device is defined;
- Step 2: chaincode, which handles the transaction of adding the new identity to the distributed ledger, is invoked;
- Step 3: the transaction is executed after obtaining approvals of organizations specified by the endorsement policy;
- Step 4: blockchain event listener application listens for events emitted by the distributed ledger;
- Step 5: for an approved and executed transaction, an event related to the registration of a new identity image is emitted;
- Step 6: when a specific event is received by the application (Blockchain Event Listener), the identity image is extracted from the event payload;
- Step 7: the identity is uploaded to a dedicated Kafka topic;
- Step 8: the streaming processing microservice sequentially reads the identities from the dedicated topic;
- Step 9: identity is added to the local off-chain data store, which can be used by the other streaming processor microservices.

Optionally, the application or microservice can invoke a synchronization query to the Hyperledger Fabric blockchain to compare the integrity of the master identity image from the ledger with the one extracted from the event payload or written by the microservice to the local data store.

V. Framework Evaluation

Performance benchmarking of streaming data processing systems is an extensive challenge that arises from the problem of the global notion of time. This section describes preliminary benchmarks for our framework, where we evaluated the processing-time latency for microservices that were implemented using the Kafka Streams API. The main purpose of the tests was to confirm whether our framework is feasible to process message streams, especially audiovisual streams. Even if our processing logic is dependent on performing identity read operations from the distributed ledger.

A. Setup

The various components of our experimental framework were deployed using the Amazon Web Services (AWS) cloud environment. Figure 6 shows the test environment, which includes:

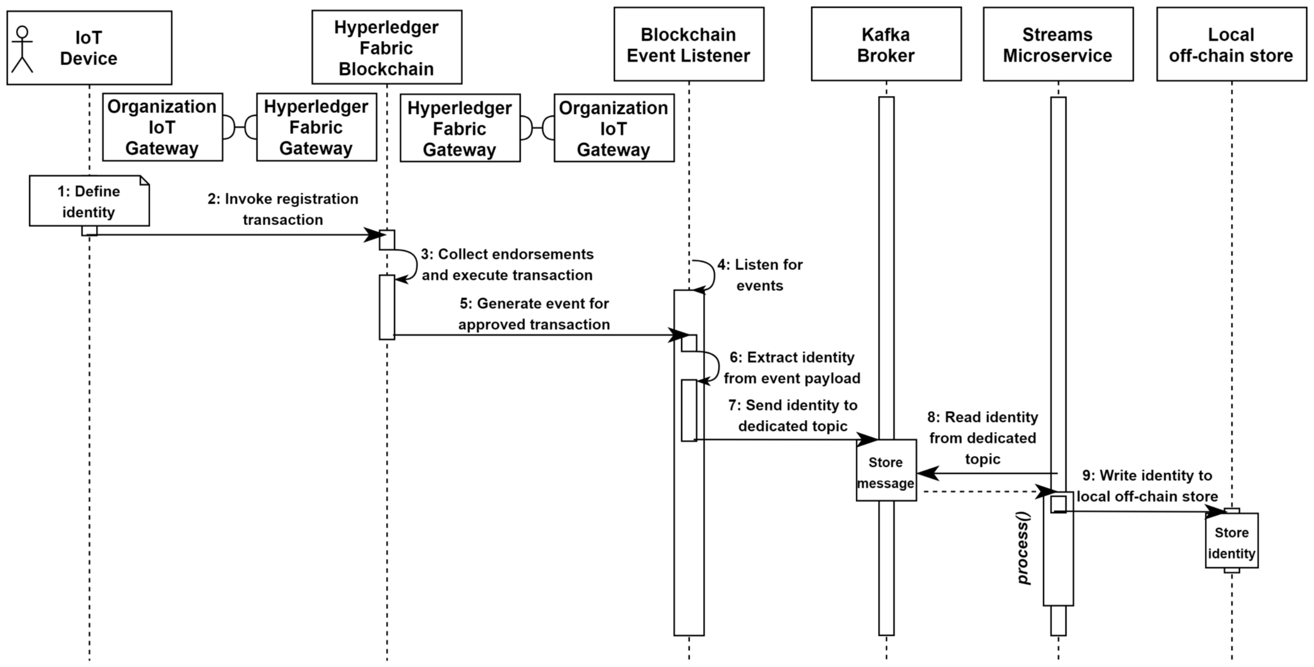

IoT Device
Hyperledger Fabric Blockchain
Blockchain Event Listener
Kafka Broker
Streams Microservice
Local off-chain store
Organization IoT Gateway
Hyperledger Fabric Gateway
Hyperledger Fabric Gateway
Organization IoT Gateway
1: Define identity
2: Invoke registration transaction
3: Collect endorsements and execute transaction
4: Listen for events
5: Generate event for approved transaction
6: Extract identity from event payload
7: Send identity to dedicated topic
8: Read identity from dedicated topic
Store message
9: Write identity to local off-chain store
process()
Store identity

Fig. 5. Sequence diagram for adding identities through an event listener

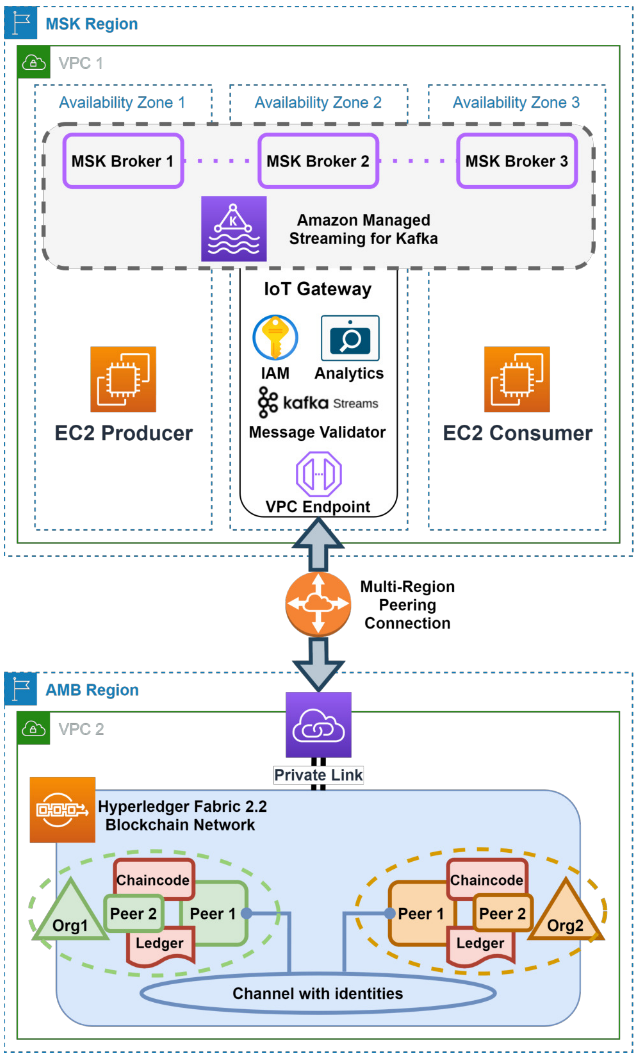


Fig. 6. Test environment overview

- two AWS regions to simulate geographical distances: MSK Region that belongs to Org1, and AMB Region for Org1 and Org2. The Multi-Region link was set using VPC Peering Connection;
- a single Amazon Managed Streaming for Apache Kafka version 2.8.1, deployed in the MSK Region isolated (availability) zones, where three kafka.t3.small brokers (vCPU: 2, Memory: 2GiB, Network Bandwidth: 5Gbps) were set. Each broker has a default configuration with a single partition and a replication factor of 3;
- a single Amazon Managed Blockchain Starter Edition for Hyperledger Fabric version 2.2, deployed in the AMB Region, where a single channel for identities was created within the blockchain network. Also, each member (Org1, Org2) of the channel has two nodes (peers) running of type bc.t3.small (vCPU: 2, Memory: 2GiB, Network Bandwidth: 5Gbps);
- two Amazon Elastic Compute Cloud (EC2) virtual machines of type t2.micro (vCPU: 1, Memory: 1GiB), deployed in the MSK Region in two isolated zones to simulate message producer and consumer;
- a single EC2 instance (t2.micro) deployed in the MSK Region, running as a microservice that verifies messages;
- a single AWS PrivateLink interface (VPC Endpoint) that enables the communication between the microservice and elements of the Hyperledger Fabric.

The AWS cloud due to its pay-as-you-go model and pluggable architecture for Commercial Off-The-Shelf services: Apache Kafka (Amazon Managed Streaming) and Hyperledger Fabric (Amazon Managed Blockchain), enables efficient deployment of our framework. Simultaneously minimizing the operational costs associated with the provisioning, configuration, and maintenance of its various components. As a consequence, our framework is suitable for federated environments for which it is required to ensure zero-day interoperability.

B. Processing scenarios

In conducting performance studies (benchmarks) of streaming data processing systems, it is necessary to consider three main metrics [26], [27]: latency, throughput, and the usage of hardware-software resources (CPU, RAM). Furthermore, the overall performance evaluation can be affected by the input parameters (e.g., system configuration) and processing scenarios (workloads) [25]. In the context of the proposed framework, several parameters are listed below:

- parallelization of stream processors (microservices);
- the kind (e.g., join, windowed aggregation) and type of operations (e.g., stateless, stateful);
- configuration for Kafka brokers: number of brokers, partitions, and replication factor [25];
- number of organizations that joined a federated environment;
- the number of nodes of the distributed ledger, and registered devices (identity count);
- the selected programming language for microservices and chaincodes (e.g., Java, Go, Node.js).

Generally, latency defines as the interval of time it takes for a system under test (SUT) to process a message, calculated from the moment the input message is read from the source until the output message is written by SUT. Hence, it is important to distinguish the latency metric [26] into its two types: *event-time latency* and *processing-time latency*. The first mentioned refers to the interval between a timestamp assigned to the input message by the source (e.g., broker) and the time the SUT generates an output message. The second one refers to the interval calculated between the time when an input message is ingested (read) by the SUT, and the time the SUT generates an output message. In this paper, we only prepared and conducted two workload scenarios to test the performance of the proposed processing logic for microservices, where the processing-time latency was measured:

1) Scenario I: involved verifying the input (sealed) message by performing a comparison operation between

TABLE I
PROCESSING-TIME LATENCY METRICS (IN MILLISECONDS)

Identity Count	Avg	Avg Dev	Min	Max	Pop Std Dev	Percentiles [p=0.9; p=0.95; p=0.99]
10000	38.3 (0.76)	3.0 (0.00)	32.7 (0.56)	133.8 (7.80)	5.5 (0.70)	[44.3 (0.76); 47.7 (1.24); 56.7 (2.16)]
20000	39.5 (1.20)	4.0 (0.00)	32.2 (1.28)	143.7 (9.90)	6.6 (0.60)	[46.7 (1.90); 50.7 (1.70); 61.4 (2.00)]
35000	39.9 (1.12)	3.7 (0.42)	32.6 (1.08)	131.2 (5.64)	5.9 (0.54)	[46.5 (1.10); 50.5 (1.30); 60.2 (2.16)]
50000	38.2 (1.28)	3.3 (0.42)	31.7 (1.3)	130.8 (7.76)	5.5 (0.60)	[44.6 (1.72); 47.8 (2.00); 55.7 (2.70)]
100000	38.6 (0.72)	3.3 (0.42)	32.0 (0.40)	139.3 (5.16)	5.5 (0.50)	[44.7 (0.56); 47.8 (0.64); 55.8 (1.64)]

TABLE II
TIME OF DATASTREAM (1000 MSG) RETRIEVAL BY CONSUMER (IN SECONDS)

	Identity Count				
	10000	20000	35000	50000	100000
Scenario I	38.86 (0.64)	39.95 (1.39)	40.19 (0.98)	38.59 (1.22)	39.15 (0.61)
Scenario II	20.42 (0.77)	21.03 (1.20)	21.15 (1.06)	20.39 (1.12)	20.06 (0.69)

the extracted device identity, with the identity stored in the distributed ledger.

2) Scenario II: involved verifying the sealed message by performing a comparison operation between the extracted device identity, with the identity stored in the off-chain data store. For this scenario, all device identities from the distributed ledger were also stored in the off-chain data store.

For all scenarios, the burst at startup technique was applied [27], where each input message was beforehand generated and sealed with a pseudo-randomly device identity. Once a certain number of input messages were generated, a single instance of the microservice responsible for verifying them was invoked. At the same time, within the scenarios, every second message pointed to an identity that was not registered in the distributed ledger. This approach was designed to minimize the impact of optimization (caching) mechanisms.

C. Discussion

Table 1 presents results for workload scenario I, where we determined the processing-time metric:

- average latency (Avg) and average absolute deviations of data points from their mean value (Avg Dev);
- minimum (Min) and maximum (Max) latency;
- standard deviation based on the entire population (Pop Std Dev);
- quantiles of order: p90, p95, p99.

Table 2 presents the average times for consumers to read message streams at the same time when SUT was processing it.

Both tables present the averaged results along with the average absolute deviation shown in brackets calculated for 10 repetitions of each processing scenario. Also, the input parameters for both scenarios were: a number of registered identities (identity count) in the distributed ledger and a fixed number of 1 000 messages. The optimal number of beforehand generated messages was determined through empirical tests for mentioned scenarios. During this, we noted slight increases in the accuracy of the measurements in comparison to 10 000 or 100 000 messages.

Regarding the results (Tab. 1), changing the number of registered identities did not affect the processing-time latency associated with the verification of a single message. An average delay of ~39ms was measured. The minimum delay was 31ms. In contrast, the average deviations for quantiles of the order p90 and p99 do not consecutively exceed ~2ms and ~3ms. For 100 000 registered identities, quantiles of p90 latencies were below ~45ms, and for p99 below ~56ms.

The results shown in Table 2 for workload scenario I are promising as the average time for consumers to read message streams was ~39 seconds. In the context of audiovisual streams, the measured time (1 000/39) represents ~25 frames per second (~25fps). The work of [28] demonstrated that CCTV cameras with a minimum 8fps frame rate are required to correctly identify objects on video. In addition, the results for workload scenario II confirmed the rightness of local off-chain data store usage as a mirco-caching mechanism for message verifying. The usage of mentioned data store almost doubly reduced the reading time of the message stream. An important point for scenario II is that for every second message, the identity did not exist in both data stores.

The collected results were also compared with the results in the works of [25], [26], and [27]. Hence, the rationale of the proposed processing logic for microservices was confirmed. Additionally, the obtained low average deviations indicate the stability of the proposed framework deployed with the AWS Cloud. This characteristic is important in the context of further performance studies (e.g., maximum, sustainable throughput).

D. Security and reliability risk assessment

We have conducted a high-level security risk assessment considering several security and reliability threats across the Application, Network, and Perception layers of the IoT system.

Application Layer:

1) *Storage attack* - the attack consists of changing device identity features. To prevent this attack, access to the device should be properly secured to prevent the change of data stored in it. In our framework, if the device identity is changed, the device will not be able to authenticate itself. It is almost impossible to change data in a distributed ledger without the knowledge and consent of the organization that owns the IoT device.
2) *Malicious insider attack* - the attack consists of the use of credentials by an authorized person. In the framework, access to data stored in the Hyperledger Fabric is possible only by an authenticated and authorized entity that uses an appropriate private key and a valid X.509 certificate. Each access attempt is logged. Resistance to this attack can be enhanced by using Security Information and Event Management (SIEM).
3) *Distributed ledger node failures* - in our framework, we propose each organization has a minimum of two nodes. Since the data in the blockchain is replicated, the failure of a single node does not affect the operation of the entire network.
4) *Kafka cluster (brokers) failure* - in our framework, it is possible to maintain the availability of data generated by IoT devices by using built-in synchronization mechanism and setting an appropriate replication factor.
5) *Denial of Service* – in our framework, the number of ledger nodes, Kafka brokers, microservice, and IoT gateways could be increased to handle more requests. Moreover, using SIEM we can identify specific properties of requests involved in DoS to detect a source of overload and reject all malicious requests at the gateway level.

Network Layer:

1) *Eavesdropping* - the attack involves eavesdropping on transmissions and obtaining messages (credentials). In our framework, we have separated the key used to secure the communication channel for IoT devices from the key used for the data authenticity protection mechanism. Only the registration phase is critical and must be carried out in a protected, trusted environment.
2) *Man-in-the-Middle* - the attack consists in changing the messages sent between the IoT device and the verifying microservices. Any change to the message will prevent it from being verified due to the data authenticity protection mechanism. Moreover, invalid messages can be logged to identify faulty (malicious) devices.

Perception Layer:

1) *Device capture* - the attacker can access the IoT device and generate messages sealed with its identity. In this situation, it is assumed that for such a device, its behavioral pattern (distinctive features) will change. As a consequence, it will be possible to use analytics tools (SIEM) to detect these changes, mainly related to network fingerprints. Moreover, when a compromised device is detected, it can be immediately marked and revoked from the distributed ledger. Also, hardware modules such as TPM can be used to increase device resilience against capture and manipulation.
2) *Malicious device* - the attack involves adding a fake IoT device to the network. In our structure, the process of registering a device takes place once in a protected environment. Therefore, we assume that the process will be coordinated by an authorized person. Therefore, it is not possible to register a fake device. If a device is not registered, it will not be authenticated and messages from such a device will be rejected, and consequently the device will be detected and blocked.
3) *Device tampering* - the attack consists in changing software or hardware components of the IoT device. In our framework, any changes to a unique device fingerprint would generate numerous failed verification attempts.
4) *Sybil attack* - The attack consists in having a multi-identity device by the IoT device. In our framework, this situation is prevented via a secure registration process.
5) *Side-channel (timing) attacks* - attacks consist in obtaining the key by analyzing the implementation of the protocol (e.g., current power consumption, time dependencies). The framework could be susceptible to a timing attack when the device will use unique data to seal messages. In this situation, it is possible to predict from where these data are read, but not the values of these data, therefore we believe that this attack is rather difficult to perform in practice.

VI. Conclusion and Future Work

One of the still unresolved problems is the acquisition, analysis, and fusion of enormous amounts of data generated by various IoT devices, and their secure and reliable distribution. In order to fill the gap, we proposed an experimental framework architecture for secure and reliable message stream distribution in a multi-organizational federation environment.

Deploying our framework within AWS Cloud infrastructure showed that it is suitable for environments where immediate interoperability is required. Moreover, preliminary performance benchmarking and obtained results (~25fps) confirmed the rationale for the usage of our solution to process audio-visual streams. Also, obtained low processing-time latency average deviations indicate the stability of the proposed system. This characteristic is important in the context of further performance studies, like event-time latency and maximum throughput.

Our framework has the potential to serve as the backbone for multiple applications. For instance, it could be incorporated into UAV detection and neutralization systems. This would allow both civilian and military organizations to have full control over air-defense activities, where IoT devices that are part of the smart city and military infrastructure can securely disseminate data about UAV location via our solution. Another possible scenario could involve setting up an ad-hoc system to coordinate international operations aimed at providing humanitarian assistance in eliminating natural disasters (known as HADR - Humanitarian Assistance And Disaster Relief). Our system can be of great help in this case, as it allows for the reliable exchange of data from CCTV cameras and health devices such as SOS wristbands. This would reduce response time for those in need of assistance and lead to better decision-making through improved information sharing.

Future work will focus on the development of a detailed design of a protocol for secure communication of IoT devices with a distributed registry, along with a protocol to enable message sealing that utilizes the identity of the IoT device. Additionally, we intend to conduct a comprehensive evaluation of a security and reliability risk, and we will implement dedicated software to seal messages with device identity.

References

[1] M. Manso et al., "Connecting the Battlespace: C2 and IoT Technical Interoperability in Tactical Federated Environments". MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM), 2022, pp. 1045-1052, DOI: 10.1109/MILCOM55135.2022.10017950.

[2] F. Johnsen, M. Hauge, "Interoperable, adaptable, information exchange in NATO coalition operations", Journal of Military Studies. 11, 2022, pp.49-62, DOI: 10.2478/jms-2022-0005.

[3] H. Kopetz, W. Steiner, "Real-Time Systems: Design Principles for Distributed Embedded Applications", Springer, 2022, DOI: 10.1007/978-3-031-11992-7_13.

[4] N. Jansen et al., "NATO Core Services profiling for Hybrid Tactical Networks — Results and Recommendations," 2021 International Conference on Military Communication and Information Systems (ICMCIS), The Hague, Netherlands, 2021, pp. 1-8, DOI: 10.1109/ICMCIS52405.2021.9486415.

[5] N. Suri et al., "Experimental Evaluation of Group Communications Protocols for Tactical Data Dissemination", MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 2018, pp. 133-139, DOI: 10.1109/MILCOM.2018.8599749.

[6] Hyperledger Fabric documentation. Accessed: May. 22, 2023. [Online]. Available: https://hyperledger-fabric.readthedocs.io/

[7] Apache Kafka documentation. Accessed: May. 22, 2023. [Online]. Available: https://kafka.apache.org/

[8] Xu Wang et al., "Survey on blockchain for Internet of Things", Computer Communications 136, 2019, pp. 10-29, DOI: 10.1016/j.comcom.2019.01.006

[9] L. Ramasamy et al., "A Survey on blockchain for industrial Internet of Things", Alexandria Engineering Journal. 61., 2021, pp. 6001-6022, DOI: 10.1016/j.aej.2021.11.023.

[10] O. Alfandi et al., "A survey on boosting IoT security and privacy through blockchain", Cluster Computing. 24, 2021, pp. 37-55, DOI: 10.1007/s10586-020-03137-8.

[11] S. Guo et al., "Master-slave chain based trusted cross-domain authentication mechanism in IoT", Journal of Network and Computer Applications. 172, 2020, DOI: 10.1016/j.jnca.2020.102812.

[12] L. Xu et al., "DIoTA: Decentralized-Ledger-Based Framework for Data Authenticity Protection in IoT Systems", IEEE Network. 34, 2020, pp. 38-46, DOI: 10.1109/MNET.001.1900136.

[13] U. Khalid et al., "A decentralized lightweight blockchain-based authentication mechanism for IoT systems", Cluster Computing 23, 2020, pp. 2067–2087, DOI: 10.1007/s10586-020-03058-6

[14] A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics", IEEE Transactions on Mobile Computing. 18, 2019, pp. 1745-1759, DOI: 10.1109/TMC.2018.2866249.

[15] Q. Xu et al., "Device Fingerprinting in Wireless Networks: Challenges and Opportunities", IEEE Communications Surveys & Tutorials. 18, 2016, pp. 94-104, DOI: 10.1109/COMST.2015.2476338.

[16] A. Jagannath et al., "A Comprehensive Survey on Radio Frequency (RF) Fingerprinting: Traditional Approaches, Deep Learning, and Open Challenges", 2022, DOI: 10.36227/techrxiv.17711444.

[17] M. Jarosz et al., "Formal verification of security properties of the Lightweight Authentication and Key Exchange Protocol for Federated IoT devices," 17th Conference on Computer Science and Intelligence Systems (FedCSIS), Sofia, Bulgaria, 2022, pp. 617-625, DOI: 10.15439/2022F169.

[18] L. Sanogo et al., "Intrusion Detection System for IoT: Analysis of PSD Robustness", Sensors. 23., 2023, pp. 2353, DOI: 10.3390/s23042353.

[19] B. Chatterjee et al., "RF-PUF: Enhancing IoT Security Through Authentication of Wireless Nodes Using In-Situ Machine Learning", IEEE Internet of Things Journal. 6, 2019, pp. 388-398, DOI: 10.1109/JIOT.2018.2849324.

[20] B. Charyyev and M. H. Gunes, "IoT Traffic Flow Identification using Locality Sensitive Hashes", ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, DOI: 10.1109/ICC40277.2020.9148743.

[21] C. Neumann et al., "An Empirical Study of Passive 802.11 Device Fingerprinting", 32nd International Conference on Distributed Computing Systems Workshops, Macau, China, 2012, pp. 593-602, DOI: 10.1109/ICDCSW.2012.8.

[22] F. De Rango et al., "Energy-aware dynamic Internet of Things security system based on Elliptic Curve Cryptography and Message Queue Telemetry Transport protocol for mitigating Replay attacks", Pervasive and Mobile Computing. 61, 2019, pp. 101105, DOI: 10.1016/j.pmcj.2019.101105.

[23] M. Yang et al., "Differentially Private Data Sharing in a Cloud Federation with Blockchain", IEEE Cloud Computing. 5, 2018, pp. 69-79, DOI: 10.1109/MCC.2018.064181122.

[24] RFC 7228: Terminology for Constrained-Node Networks Accessed: May. 22, 2023. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7228.

[25] S. Kul et al., "Event-Based Microservices With Apache Kafka Streams: A Real-Time Vehicle Detection System Based on Type, Color, and Speed Attributes", IEEE Access. 9, 2021, pp. 83137-83148, DOI: 10.1109/ACCESS.2021.3085736.

[26] J. Karimov et al., "Benchmarking Distributed Stream Data Processing Systems" IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 2018, pp. 1507-1518, DOI: 10.1109/ICDE.2018.00169.

[27] G. van Dongen, D. Van den Poel, "Evaluation of Stream Processing Frameworks", IEEE Transactions on Parallel and Distributed Systems. 31, 2020, pp. 1845-1858, DOI: 10.1109/TPDS.2020.2978480.

[28] H. Keval, A. Sasse, "To catch a thief - You need at least 8 frames per second", Proceedings of the 16th ACM international conference on Multimedia, 2008, pp. 941-944, DOI: 10.1145/1459359.1459527.