SOFIE - Secure Open Federation for Internet Everywhere

779984

DELIVERABLE D4.5

Final Architecture, System, and Pilots Evaluation Report

Project title:	SOFIE – Secure Open Federation for Internet Everywhere
Contract Number:	H2020-IOT-2017-3-779984
Duration	1.1.2018 – 31.12.2020
Date of preparation:	23.12.2020
Authors:	Vasilios A. Siris, Spyros Voulgaris, Nikos Fotiou, Yiannis Thomas, Iakovos Pittaras, George C. Polyzos (AUEB-RC), Tommi Elo, Dmitrij Lagutin, Yki Kortesniemi, Lei Wu, Ektor Arzoglou, Veria Hoseini, Sampsa Ruutu (AALTO)
Responsible person:	Vasilios A. Siris (AUEB-RC), vsiris@aueb.gr
Target Dissemination Level:	Public
Status of the Document:	Completed
Version	1.00
Project web-site:	https://www.sofie-iot.eu/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779984.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Program Interface
AS	Authorization Server
BLE	Bluetooth Low Energy
BP	Business Platform
CAMG	Context-Aware Mobile Gaming
CLD	Causal Loop Diagram
CS	Charging Station
DEDE	Decentralized Energy Data Exchange
DEFM	Decentralized Energy Flexibility Marketplace
DER	Distributed Energy Resource
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DNS	Domain Name System
DR	Demand Response
DSO	Distribution System Operator
eIDAS	Electronic Identification, Authentication and Trust Services
ER	Electricity Retailer
ERC	Ethereum Request for Comment
ETH	Ether – Ethereum coin
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
EVM	Ethereum Virtual Machine
FM	Fleet Manager
FSC	Food Supply Chain
GDPR	General Data Protection Regulation
GW	Gateway
Gwei	Gigawei – equivalent to one nanoether or 10^{-9} ETH
IAA	Identify, Authentication, and Authorization
ICT	Information and Communication Technology
ID	Identifier
ILG	Interledger Gateway
IoT	Internet of Things
JSON	JavaScript Object Notation
JWT	JSON Web Token
KPI	Key Performance Indicator
MAC	Message Authentication Code
MP	MarketPlace
MS	MileStone
OAuth	Open Authorization
P2P	Peer-to-Peer
PaD	Provisioning and Discovery

PDS	Privacy and Data Sovereignty
PoI	Point of Interest
PoP	Proof of Possession
PoW	Proof of Work
PV	PhotoVoltaic
RAXX	Requirement number XX of SOFIE Architecture
RFXX	Requirement number XX of SOFIE Framework Component
RBAC	Role Based Access Control
RES	Renewable Energy Sources
RP	Reference Platform
RPF	Reverse Power Flow
SD	System Dynamics
SoC	State-of-Charge
SM	Supermarket
SR	Semantic Representation
TC	Test Case
TSO	Transmission System Operator
TR	Transportation
UC	Use Case
UML	Unified Modelling Language
URI	Uniform Resource Identifier
V2G	Vehicular to Grid
VC	Verifiable Credential
VP	Verifiable Presentation
WH	Warehouse
WoT	Web of Things
WP	Work Package
ZKP	Zero Knowledge Proof

1 Introduction

This deliverable contains the final results from the evaluation work in WP4. Since the previous (second) evaluation deliverable D4.4 was submitted (April 2020), the deliverables containing the end-to-end platform validation (D5.3, submitted July 2020) and final version of the federation architecture (D2.6, submitted October 2020) have been submitted. The latter, deliverable D2.6, contains the final high-level view of the SOFIE components and federation adapters, and the interactions between them. Deliverable D5.3 contains a revised description of the system software architecture and end-to-end validation results for the four SOFIE pilots and an additional use case (Secure Marketplace for Access to Ubiquitous Goods, SMAUG) which serves as a reference use case encompassing all SOFIE federation components.

Aside from additional evaluation results for a subset of the SOFIE components (Section 3), a key focus of the current deliverable is the joint analysis of the testbed evaluation results based on the emulated pilot scenarios and the results from the actual pilot assessment (Section 4). Note that the submission of this deliverable coincides with the submission of deliverables D2.7 containing the final version of the federation framework and D5.4 containing the final validation and replication guidelines. The work and preparation of these two deliverables has been scheduled a couple of weeks prior to the preparation of the current deliverable D4.5, which allows the results from these two deliverables to be considered in the results presented in the current deliverable.

As mentioned above, a key focus of this deliverable is the joint analysis of testbed and pilot results. This joint analysis involves three directions: First, measurements of system performance KPIs obtained from both the testbed emulation environment and the actual pilots are presented side-by-side and compared. Note that the goal is not necessarily to replicate the measurements for all KPIs on both the testbed and the pilot platforms, since the former has the flexibility to implement a wider range of scenarios and a larger scale (e.g., in terms of number of entities). The second direction involves generalizing the pilot scenarios to include alternative options that are not necessarily considered in the SOFIE pilots. This allows us to assess the various tradeoffs of many potential alternative design decisions and compare these alternatives with the actual pilot scenarios. Finally, the third direction is to utilize traces and statistics from the actual pilots to conduct realistic evaluation experiments.

Note that the previous deliverable D4.4 contains, in addition to the validation strategy, interim validation results for the framework components and the pilots. Subsequent validation results for the framework components are contained in deliverable D2.7 (Federation Framework, final version, scheduled December 2020), while subsequent pilot validation results are contained in D5.4 (Final Validation & Replication Guidelines, scheduled December 2020).

The evaluation results reported in Sections 3, 4, and 5 of this deliverable use the tools identified in D4.1 (Validation and Evaluation Plan) and were performed in the testbed environments described in D4.2 (Testbed and Emulation Environment Design and Setup), which have been modified or adapted as described in each specific scenario.

1.1 Goals of this deliverable

The goals for this final architecture, system, and pilot evaluation deliverable are the following:

- to provide the final evaluation of the SOFIE approach, architecture, systems, and components, extending and complementing the results of the first two evaluation cycles,
- to provide new results for the component evaluation and for the evaluation of pilot emulation scenarios, revised and adapted based on input from the final design of the federation architecture and federation framework, and the end-to-end pilot platform validation,
- to extend the pilot scenarios to include alternative options in order to assess the various tradeoffs of many potential design decisions, to assess the scalability of the SOFIE

approach, and to conduct realistic evaluation experiments utilizing traces and statistics from the actual pilots,
- to jointly analyse the testbed emulation results and the pilot results, in order to ensure consistency and explain any differences, and assist in the selection of the specific use cases and corresponding results that are most appropriate for promoting the SOFIE approach,
- to identify the main evaluation results and performance gains that will be used to promote the SOFIE approach and establish foundations for its impact on technology and business.

1.2 Methodologies and evaluation approach

The methodologies employed for evaluation are many and diverse, from simple presentation of arguments and qualitative evaluation, through modelling, analytical evaluation and simulation, to implementation and measurements in real components and systems. The current deliverable, as the previous two deliverables D4.3 and D4.4, follows the detailed validation and evaluation planning that includes responsibilities, tools, and methods contained in the revised D4.1 (Validation and Evaluation Plan – initial submission September 2018, revised submission December 2019). Additionally, the evaluation results reported in this deliverable were performed in the testbed environments described in deliverable D4.2 (Testbed and Emulation Environment Design and Setup).

Because the pilots have a central position in the SOFIE project, an important evaluation direction was undertaken using each pilot as a starting point, considering the actual system and evaluating it in a specific application context. This provides a concrete systems and applications context in which we consider the SOFIE approach and evaluate it. However, WP4 aims to have a wider scope. It has chosen as one key approach for evaluation the emulation and/or simulation of the use cases considered in the pilots, but in a more general context, considering and evaluating various possible solutions and their parameters, going beyond what is possible within the actual pilots. On the other hand, in order to achieve this breadth, it proceeded to model and abstract out various aspects of the pilots, as will be explicitly described in the following sections. Among others, such an approach is necessary to assess the scalability of SOFIE's components and approach.

Since the first and second evaluation deliverables D4.3 and D4.4, the evaluation and validation approach has been adjusted. Specifically, the final version of the federation architecture in D2.6 and the revised description of the system software architecture and end-to-end validation results for the four SOFIE pilots contained in D5.3 have been taken into account in the evaluation approach used to obtain the results in this deliverable. In particular, the mapping of pilot use cases to their corresponding application domains and the context of each pilot have been considered in adapting and extending the emulation scenarios studied in this deliverable.

A new direction pursued in this deliverable is the joint analysis of testbed emulation results and pilot results. In addition to presenting side-by-side and discussing the emulation and pilot results for some system performance KPIs, we also present new emulation results that exploit traces and statistics from the actual pilots, for the assessment of realistic scenarios, albeit at a larger scale that cannot be achieved solely by the pilot environments.

As in the previous two evaluation deliverables D4.3 and D4.4, the results are presented in terms of the defined KPIs, which initially appeared in deliverable D2.2 (Federation Architecture, 1st version – August 2018) and were further refined and extended in the revised deliverable D4.3 (First Architecture and System Evaluation Report - initial submission June 2019, revised submission December 2019). The KPIs are also used in the joint analysis of testbed emulation results and pilot results.

1.3 Structure and overview of this deliverable

Below we present the structure of this deliverable, highlighting the new results compared to the previous (second) evaluation deliverable D4.4, submitted in April 2020.

Section 2 summarizes the high-level architecture evaluation contained in the previous deliverable, which focused more on the style and desirable properties of the architecture, rather than on specific architecture components and performance results. In the current deliverable, we focus in particular on three features: openness, federation, and privacy; these are directly related to new evaluation results that are presented and discussed in subsequent sections. Also, in Section 2.2.3 we present and discuss the final results achieved for the architecture KPIs. The system performance KPIs pertaining to the component and pilot emulation scenarios are evaluated in Sections 3 and 4, respectively.

Section 3 focuses on component evaluation. New results are presented for the Interledger, Privacy and Data Sovereignty (PDS), the Identification, Authentication, and Authorization (IAA), and the Marketplace components. Specifically, the Interledger component is evaluated in two new scenarios related to game asset transfer and fund transfer. For the PDS component, the new results involve the evaluation of its local differential privacy mechanism that adds noise to data, while maintaining the aggregate statistics. For the IAA component we evaluate W3C compliant verifiable credentials. Finally, for the Marketplace component we investigate the multiple winner selection functionality, which extends the basic functionality of the Marketplace component. This functionality is further assessed in the large-scale deployment scenarios in Section 4.

Section 4 focuses on evaluating the pilot-inspired use cases by including alternative options not necessarily considered in the SOFIE pilots. Moreover, this section contains the joint analysis of evaluation and pilot results. In addition to presenting side-by-side and discussing the results in terms of the system performance KPIs, demonstrating the complementarity and extensions of the evaluation results, this section also contains new emulation results that exploit traces and statistics from the actual pilots. Specifically, for the Food Supply Chain (FSC) we investigate a new scenario which closely emulates the model employed in the pilot. Together with the scenarios investigated in D4.4, the scenarios investigated form a wide range of clearly defined baselines, whose evaluation identifies their relative strengths, weakness, and tradeoffs among the ledger transaction cost, transaction delay, and privacy. For the Decentralized Energy Flexibility Marketplace (DEFM) we investigate large-scale deployments of the service, which include a large number of Electric Vehicles (EVs) and Charging Stations (CSs), in addition to realistic volumes of Reverse Power Flow (RPF) caused by the green-energy production units. To perform the evaluation, we have developed a sophisticated emulator-tool that models the main components of the pilot with sufficient detail; this includes bid generation, multiple winner selection, and scheduling of EVs. For the Decentralized Energy Data Exchange (DEDE), we investigate the local differential privacy mechanism utilizing smart meter traces from the pilot. Finally, for the Context-Aware Reality Mobile Gaming[1] pilot we consider a new scenario involving an open advertising ecosystem for DLT-assisted mobile gaming, which highlights one of the main features of the SOFIE architecture and framework: openness.

Section 5 focuses on the business platform evaluation. Specifically, this section evaluates how the DLT-based federation using the SOFIE framework affects the business value of SOFIE's application areas and pilots. SOFIE's framework provides the necessary technical features and capabilities to support DLT-based federations, but the benefits of the corresponding federation arrangements require the understanding of business-side issues. DLTs can reduce risks, while

[1] The Context-Aware Mobile Gaming (CAMG) pilot was called Mixed Reality Mobile Gaming (MRMG) in previous deliverables.

also bringing down the detection time of benefits and decreasing collateral damage from one party's fixes to others.

Finally, this deliverable concludes in Section 6 with a summary and the identification of possible directions that can be investigated beyond the SOFIE project, building on the results and exploiting the insight obtained through the evaluation work performed in its three-year duration.

2 Architecture evaluation and KPIs

In the previous two evaluation deliverables D4.3 and D4.4 we discuss and bring out key aspects of the SOFIE architecture that are critical for the SOFIE approach to IoT system federation and open business platform success. In Section 2.1 below we identify the main features of the SOFIE architecture. These features have been discussed in detail in the previous deliverable D4.4, hence we do not repeat them here. Rather, in this deliverable we summarize the description on how all the features are achieved. The main new results that are presented in subsequent sections of the current deliverable concern three features: openness, federation, and privacy.

Next, in Section 2.2 we present the architecture KPIs and the system performance KPIs grouped by pilot. The final achieved values of the architecture KPIs are discussed in Section 2.2.3 and shown in Table 8. The achieved values of the system performance KPIs are contained in subsequent sections for each emulated pilot scenario, where we also present the joint analysis of emulation and pilot results: Section 4.1.4 (Table 17) for FSC, Section 4.2.7 (Table 25) for DEFM, Section 4.3.3 (Table 28) for DEDE, and Section 4.4.5 (Table 30) for MRMG.

2.1 Architecture features

The key features of the SOFIE architecture are the following:

- Decentralization
- Multiple ledgers and interledger technology
- Availability
- Transparency
- Trust and accountability
- Security
- Privacy
- Federation
- Open business platforms

These features have been discussed in detail in the previous deliverable D4.4 (Second Architecture and System Evaluation Report, April 2020). The following table summarizes how each of the aforementioned features are addressed in the SOFIE architecture. For some features, namely open business platforms, federation, and privacy, we provide links to later subsections of the current deliverable containing new results.

Table 1: Architecture features.

Feature	How feature is addressed
Decentralization	SOFIE is decentralized by design, pertaining to the collaboration of distinct business entities with private data silos. SOFIE's main target is to enable interaction in such a decentralized set of private silos. At an architectural level, decentralization refers to the segregation of SOFIE's architecture into multiple self-contained components, which are subsequently combined to serve SOFIE applications. It also refers to the use of multiple distinct ledgers to support SOFIE applications, distributing the load of ledger operations to multiple entities to increase scalability and throughput. Finally, at a low, implementation level, decentralization refers to the main technology behind SOFIE, blockchains, which are inherently decentralized; this is even more so when considering the interoperation of multiple diverse blockchains through the use of interledger technology.

Multiple ledgers and interledger technology	The use of multiple ledgers in SOFIE is dictated by two goals. First, in a multi-party business scenario, the use of multiple ledgers may reflect more accurately the interaction between different parties. For example, one blockchain could serve the interaction between parties A, B, and C, while a separate blockchain could serve parties X, Y, and Z. Second, different blockchains have different technical properties, such as transaction cost, block generation speed, smart contract capabilities, etc. Combining different blockchains offers SOFIE applications more flexibility in fulfilling specific requirements using best-of-both-worlds features. In our tests we conclude that there is no single universally best option, therefore we compare a wide range of architectures involving multiple ledgers to be able to assess the best option for each case. New results presented in Section 3.1 of the current deliverable involve asset transfer between a gaming asset ledger and a marketplace ledger and fund transfer between different entities that have accounts on different ledgers.
Availability	Availability is an inherent feature of blockchains. As a blockchain is maintained by many nodes distributed across diverse geographic locations and administrative jurisdictions, the probability of all nodes crashing or becoming unresponsive simultaneously becomes extremely small. This is further strengthened when multiple distinct ledgers are used in parallel.
Transparency	Transparency is enforced via the use of blockchains. For example, in the DEFM pilot, the Marketplace component is responsible for guaranteeing transparency regarding current energy prices among electric vehicles (EVs), charging stations (CSs) and distribution system operators (DSOs). This is demanded by pilot requirements and is tested in our corresponding evaluation scenarios.
Trust and Accountability	Trust and accountability are enforced in SOFIE via blockchains. For example, in the FSC pilot, recording box handovers in blockchains guarantees trust between trading parties, while parties responsible for inappropriate handling of produce can be held accountable based on ground-truth records. Accountability is also a key requirement of other scenarios and pilot use cases, such as the IoT resource access scenarios investigated in the previous evaluation deliverable D4.3 and D4.4, and the DEFM and DEDE pilot scenarios.
Security	Security in SOFIE is managed by the IAA and PDS components, which encompass Hyperledger Indy, a popular Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) implementation, JSON web tokens, and OAuth2.0. This deliverable further expands on the evaluation of these components, in terms of authorization using W3C's compliant verifiable credentials (Section 3.3).
Privacy	The use of blockchains can often raise severe privacy concerns stemming from their public-access paradigm. To address such concerns and to offer options that allow business entities to maintain their privacy standards while still using our proposed solutions, we look closely at privacy trade-offs within the context of the FSC pilot in Section 4.1.5. Additionally, we provide an evaluation of the local differential privacy mechanism of the PDS component in Section 3.2.
Federation	Federation is an inherent notion in SOFIE, as the building of trust, security, and collaboration across many different business entities in private silos (platforms) is central to the design of the project. In the case of SOFIE, more

	specifically, federation refers to the seamless collaboration across many entities with distinct administrations in such a way that the final result appears as a single, well-integrated platform. SOFIE's framework provides the necessary technical features and capabilities to support DLT-based federations, but the benefits of the corresponding federation arrangements require an understanding of business-side issues. In Section 5 we discuss how DLTs can reduce risks, while also bring down the detection time of benefits and decrease collateral damage from one party's fixes to others.
Open business platforms	Openness is an intrinsic feature of the SOFIE project. The architecture, framework, and components proposed in SOFIE are open, with clearly defined operations and interfaces. Notably, SOFIE APIs are not hardcoded, but can be customized via SOFIE adapters. Moreover, the SOFIE architecture, framework, and components, enable open business platforms, in the sense that these platforms can be open for all to join by simply conforming to the SOFIE architecture and using the SOFIE framework. New results contained in the current deliverable on SOFIE's ability to support open business platforms are contained in Section 4.4. Specifically, Section 4.4.6 describes a new scenario involving an open advertising ecosystem for DLT – assisted mobile gaming, where a new advertising entity can join the ecosystem, without any intervention from middlemen.

2.2 Architecture and System Performance KPIs

This section contains the architecture and system performance KPIs, together with their targets, and the final values achieved for the architecture KPIs. The results for the system performance KPIs are presented in Section 4.

2.2.1 Architecture KPIs

The KPIs for the evaluation of the SOFIE architecture were initially defined in Deliverable D2.2 (Annex 1), and subsequently refined in the previous WP4 deliverables D4.3 and D4.4. This section reports the final values of the architecture evaluation KPIs, based on the emulated scenarios considered in the previous deliverables D4.3 and D4.4, the new results from the emulated testbed scenarios contained in the current deliverable, and the joint analysis of emulated testbed results and pilot results contained in the current deliverable. The system performance KPIs for the pilot scenarios will be investigated in Section 4.

The KPIs are shown collectively in the table below. For each KPI, the table indicates the metric for measuring the KPI, the method of verification or measurement and the target value. The final results of the architecture KPIs, based on all the evaluation and pilot results, is shown later in Table 8 (Section 2.2.3).

The KPIs related to system performance are shown after the next table, in separate tables for each pilot.

Table 2: KPIs with targets.

KPI	Goal	Description	Metric	Method of verification	Target
1	IoT operability	Prove operability of the implementation with IoT silos	Number of IoT silos	Detection of data flow in silos during implementation use case	5
2	IoT inter-operability	Prove interoperability across multiple IoT	Number of IoT silo pairs	Implementation use case accesses data or actuates	3

		silos of the reference architecture		operations in different IoT silos	
3	Ledger use	Validate SOFIE implementation capability with multiple ledgers	Number of distributed ledgers	Ledgers have detectable data passing through SOFIE implementations	5
4	Interledger use	Validate SOFIE implementation operating across multiple ledgers	Number of distributed ledger pairs	Implementation use case shown to result in operations across multiple ledgers	3
5	Ledger independence	Demonstrate capability of developing applications using ledgers, where a sufficient abstraction can be provided to applications to allow them to be targeted simultaneously to multiple ledger technologies	Number of Business Platforms (BP) samples classified into success or partial success	Demonstrate that a BP sample can be deployed on two ledgers with only configuration changes, and the BP sample users are able to use either one with only configuration item changes	3
6	Privacy designed in as a fundamental requirement	Demonstrate GDPR compliance where relevant	Number of operational GDPR features referenced and supported.	Final specifications have clear references to features implementing named GDPR requirements. Relevant pilot specifications also refer to the needed features	4
7	Device owner payments across ledgers	Ability of silo owners to send and receive payments or other value transfers	Number of ledger pairs supporting value transfer	Observation of value transfer as part of a use case in an implementation	2
8	Data sovereignty	Ability of data owners to reject or allow access, possibly for a specific time interval, to their data Each datum has an accompanying authorization list, which the data owner can modify	Number of pilot use cases utilizing data owner data sovereignty features where data owner is from a different silo than the storage silo	Count the number of use cases	3
9	User responsiveness	Apparent responsiveness of system for end users	Number of seconds within which user gets response for an action initiated by the user	Measuring from the onset of user action until the user gets a response by the system (to the user interface he or she is using)	See system performance KPI table.

10	System performance	Overall system performance reflecting the diverse needs and requirements of different use cases	Acceptable system performance for users and pilots	Qualitative evaluation of system metrics.	See system performance KPI table.

2.2.2 System performance KPIs

The general system performance KPIs together with their method of measurement and targets are shown in the table below. Later tables will adapt this table to present pilot scenario specific KPIs.

Table 3: System performance KPIs.

KPI	Name	Description	Metric	Method of measurement	Target
10.1	Ledger execution cost	Cost for executing operations on a ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost for all operations that a transaction involves	As low as possible
10.2	Configuration time	Time for configuration to complete	Time units (e.g., seconds)	Measure time between start of configuration until completion of configuration	<15 s
10.3	Response time or latency (or transaction delay)	Time for the system to respond to a request or to execute a transaction	Time units (e.g., seconds)	Measure time between instant system receives a request or transaction until the instant that the system responds	<5 s (if human involved) <1 s (if no human involved)
10.4	Throughput	Maximum number of transactions per time unit that the system can support	Number of transactions per time unit	Measure number of transactions per time unit that can be supported while the QoS (e.g., in terms of maximum response time) is satisfied	Domain specific
10.5	Scalability – cost	Increase of cost as load (e.g., number of transactions per time unit, number of nodes) increases	Ratio of delta cost over delta of load (number of transactions/nodes)	Measure cost for different loads	Linear or sublinear
10.6	Scalability – time	Increase of response time as load (e.g., number of transactions per time unit, number of nodes) increases	Ratio of delta time over delta of load (number of transactions/nodes)	Measure response time for different loads	Linear or sublinear

Next, we present the KPIs for the pilot emulation scenarios. These KPIs have been defined in the previous evaluation deliverable D4.4. Note that pilot evaluation results based on the system performance KPIs are presented in deliverable D5.4 (Final Validation & Replication Guidelines, scheduled December 2020).

The KPIs for the FSC scenarios are presented in Table 4, below. Smart contract execution on public ledgers, such as Ethereum, incurs a cost. Thus, our first KPI refers to this cost, demanding that it is kept as low as possible. Apart from some monetary cost, each interaction with a ledger also incurs a time cost. Given that Ethereum blocks are being generated every 15 seconds *on average* (thus, they could occasionally take much longer), and that a transaction submitted to public Ethereum is not guaranteed to be included in the exact next block, we believe that a target of 1 minute for handover transactions and 0.5 minutes for internal state transitions, constitutes a reasonable target for the anticipated system. Besides cost and timing limits for individual transactions, the FSC scenario is mainly concerned with *throughput*, that is, the number of products that can be processed through the chain per time unit. Having received input from a large association of producers in Greece, we know that 6000 boxes are produced per day during peak harvesting season. Thus, we set this number as a target for box processing throughput. Then follows the issue of scalability, which comes in two flavours, namely time and cost scalability. Scalability refers to the effect of the volume of box processing on the actual time delay and cost individual boxes incur. Time and cost per individual box should not increase by the volume of box transfers, which would render the system non-scalable. This is expressed by the two scalability requirements demanding that time and cost for transactions involving a number of boxes grow at most linearly with the number of boxes. Finally, a KPI is defined for the time it takes to retrieve all data needed for auditing, that is, to resolve a potential dispute. Given that multiple blockchains might have to be accessed to retrieve all relevant data, this value should not be higher than 1 minute.

Table 4: System performance KPIs for FSC scenarios.

KPI	Name	Description	Metric	Method of measurement	Target
KPI_FSC_1	Ledger execution cost in public ledger	Cost for executing operations on a ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost per box	As low as possible
KPI_FSC_2	Handover time	Time to register data to blockchain during a handover between two stages	Time unit (e.g., seconds)	Measure the total time required for blockchain-related operations during a handover of a box between two stages	<1 min
KPI_FSC_3	Internal state transition time	Time to register data to blockchain during a box's state transition occurring internally within a single stage	Time units (e.g., seconds)	Measure the total time required for blockchain-related operations during a state transition of a box within a single stage	<30 s
KPI_FSC_4	Throughput	Number of boxes that can be processed per time unit in any possible handover or internal state transition	Number of boxes per time unit	Measure the handover and state transition delays	> 6000 boxes per day
KPI_FSC_5	Scalability - time	Blockchain registration time for a handover or internal state transition, as a function of the	Derivative of the blockchain registration time with respect to	Measure handover and state transition blockchain registration time as a function of the number of boxes involved	Linear or sublinear

		number of boxes involved	the number of boxes involved		
KPI_FSC_6	Scalability - cost	Public blockchain costs associated with box handovers or internal state transitions, as a function of the number of boxes involved	Derivative of public ledger cost with respect to the number of boxes involved	Measure public blockchain cost for handovers and state transitions as a function of the number of boxes involved	Linear or sublinear
KPI_FSC_7	Response time for audit requests	The time it takes to respond to an audit request, by pulling out all data related to the box in question	Time units (e.g., seconds)	Measure the time it takes to pull out all records related to a given box, and to cross check them to identify potential issues	<1 min

The system performance KPIs for the DEFM scenarios are shown in the table below.

As in the previous scenario, Smart contract execution on public ledgers, such as Ethereum, incurs a cost. Thus, our first KPI refers to this cost, demanding that it is kept as low as possible. The next three KPIs concern response time, which previous studies have identified as an important factor for energy trading, e.g., see [Haa+18]. Moreover, previous studies suggest that the maximum value for the latency is of the order of minutes [Smart16]. The throughput and scalability of an energy marketplace system is a significant metric that characterizes the capability of the system to handle energy transactions. Similar to the other pilot scenarios, the scalability of the system should be linear or sublinear.

Table 5: System performance KPIs for DEFM scenarios.

KPI	Name	Description	Metric	Method of measurement	Target
KPI_DEFM_1	Ledger execution cost	Cost for executing operations on a ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost for all operations involved	As low as possible
KPI_DEFM_2	Response time for requests, offers, and charging event notifications	Latency of placing flexibility requests and offers on the marketplace	Minutes	Measure the time between the issuance of transaction by respective party until the transaction is recorded on the marketplace	<5 min
KPI_DEFM_3	Response time for determining the winner of the auction	Latency of determining and notifying the winner of the marketplace auction	Minutes	Measure the time between the deadline of bids and offers until the winner of the auction has been determined and notified	<5 min
KPI_DEFM_4	Response time for verifying the winning bid and compensati	Latency of verifying the winning bid and compensating (or fining) the winner	Minutes	Measure the time between sufficient charging events have been recorded on the marketplace, until the events have been verified and the winner has been properly	<5 min

	ng (or finding) the winner			compensated. If recorded charging events did not satisfy the requirement of the bid during its timeframe, measure time between the end of the bid's deadline until the verification of the failure of the bid and finding the winner of the bid.	
KPI_DEFM_5	Throughput	Number of transactions (bids, offers, selections of winning bid, charging event notifications, bid verifications, etc.)	Number of transactions per time unit (hour)	Measure number of transactions per time unit (hour) that can be supported while the QoS (e.g. in terms of maximum response time) is satisfied	>100 per hour
KPI_DEFM_6	Scalability – time	Increase of response time as load (e.g., number of transactions per time unit, number of nodes) increases	Ratio of delta time over delta of load (number of transactions/nodes)	Measure response time for different loads	Linear or sublinear

The system performance KPIs for the DEDE scenarios are shown in the table below.

As in the previous scenarios, Smart contract execution on public ledgers, such as Ethereum, incurs a cost. Thus, our first KPI refers to this cost, demanding that it is kept as low as possible. The response time metrics reflect the importance of latency in energy related marketplaces and data exchange platforms, e.g., see [SysFI19]. The scalability of data exchange systems is a significant metric that characterizes the capability of the system to handle data exchange transactions. Similarly to the other pilot scenarios, the scalability of the system should be linear or sublinear.

Table 6: System performance KPIs for DEDE scenarios.

KPI	Name	Description	Metric	Method of measurement	Target
KPI_DEDE_1	Cost for computing discounts	Cost for executing discount operations on a ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost for all operations involved	As low as possible
KPI_DEDE_2	Cost for recording hashes	Cost for recording hashes on a ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost for recording hashes	As low as possible
KPI_DEDE_3	Response time for access requests	Time for the system to respond to metering data access requests	Time units (e.g., seconds)	Measure time between instant system receives a request until the instant that the system responds	<5 s
KPI_DEDE_4	Response time for DID operations	Time for performing read/write operations on the identity ledger (Hyperledger Indy)	Time units (e.g., seconds)	Measure time between instant system receives a request until the instant that the system responds	<5 s

KPI_DEDE_5	Response time for KSI Blockchain signatures	Time for retrieving KSI Blockchain signature	Time units (e.g., seconds)	Measure time between instant system receives a request until the instant that the system responds	<2 s
KPI_DEDE_6	Processing time of requests in adapter	Time for the processing incoming requests in adapter - includes audit log entry, verifying credentials, setting up secure channel	Time units (e.g. seconds)	Measure time between instant system receives a request until the instant that the system responds	<5 s
KPI_DEDE_7	Response time for audit logs	Time for the system to respond to audit log requests	Time units (e.g., seconds)	Measure time between instant system receives a request until the instant that the system responds	<15 s
KPI_DEDE_8	Scalability – cost	Increase of cost as load (number of discount computations or hash recordings per time unit) increases	Ratio of delta cost over delta of load (number of discount computations or hash recordings per time unit)	Measure cost for different loads	Linear or sublinear
KPI_DEDE_9	Scalability – time	Increase of response time as load (e.g. number of transactions per time unit, number of nodes) increases	Ratio of delta time over delta of load (number of transactions/node)	Measure response time for different loads	Linear or sublinear

For a quantitative performance evaluation of the emulated CAMG pilot, various measurable metrics are required. First, transactions on a public blockchain incur a transaction cost, which in Ethereum is expressed as the cost of gas for executing transactions on the Ethereum Virtual Machine (EVM). The desirable target for this metric is to have as low a cost as possible. Furthermore, the most common performance metric of any system is the response time required by the system to execute read and write requests. In our case, where the gaming system utilizes blockchains, the response time metric corresponds to the time that the system performs read and write transactions. Kalra et al. [KSD18], in the evaluation of their system, present the latency of various multiplayer FPS games. The average latency for these games is 250 milliseconds. Moreover, Cai et al. [Cai+18] state that the desirable latency of any blockchain-based system, even for games that utilize blockchains, is 2 to 3 seconds. Thus, for a blockchain-based mobile game, the latency should be 3 seconds for write requests and 1 second for read requests, respectively.

Other performance metrics that are important in mobile gaming include the time that an IoT device needs to detect the player arriving at a particular location. The average time for an Android smartphone to detect a beacon is 5 seconds. So, we believe that a reasonable target for this metric is 4 seconds. Finally, the throughput and scalability of the mobile gaming system is a significant metric, since it characterizes the capability of the system to handle many users and games. We believe that the scalability of the system should be linear or sublinear in order for the system to support many users and transactions.

The aforementioned metrics along with their targets constitute the KPIs for the CAMG [2] scenarios and are shown in the following table.

Table 7: System performance KPIs for the CAMG scenarios.

KPI	Name	Description	Metric	Method of measurement	Target
KPI_CAMG_1	Public ledger execution cost	Cost for executing operation on a public ledger	Ledger execution cost units (e.g., gas in Ethereum)	Measure the total execution cost for all operations that a transaction involves	As low as possible
KPI_CAMG_2	Response time for write requests	Time for the system to respond to game state altering transactions, such as challenge creation & completion, skipping tasks and buying in-game items	Time units (e.g., seconds)	Measure time between instant system receives a request or transaction until the instant that the system responds	< 3 s
KPI_CAMG_3	Response time for read requests	Time for the system to respond to non-altering requests such as getting player's currencies and items	Time units (e.g., seconds)	Measure time between instant system receives a request or transaction until the instant that the system responds	< 1 s
KPI_CAMG_4	BLE beacon detection time	The time player has to wait between walking into the correct location and receiving the context-dependent task	Time units (e.g., seconds)	Measure average time between the instant player walks into the correct location and the client detects the beacon	< 4 s
KPI_CAMG_5	Throughput	Maximum number of transactions per time unit that the system can support	Number of transactions per time unit	Measure transactions per time unit	> 222 read and > 133 write transactions per second
KPI_CAMG_6	Scalability – cost	Increase of cost as number of challenges or active users increases	Ratio of delta cost over delta of challenges or active users	Measure cost for different numbers of challenges or active users	Linear or sublinear
KPI_CAMG_7	Scalability – time	Increase of response time as number of challenges or active users or increases	Ratio of delta time over delta of challenges or active users	Measure response time for different numbers of challenges or active users	Linear or sublinear

2.2.3 Status of architecture KPIs

In this section we present and discuss the final values of the architecture KPIs shown in Table 2. Specifically, the final values for the architecture KPIs are shown in Table 8. The user response and overall system performance KPIs will be presented in later sections.

[2] The Context-Aware Mobile Gaming (CAMG) pilot was called Mixed Reality Mobile Gaming (MRMG) in previous deliverables.

Table 8: Status of the SOFIE architecture KPIs.

KPI	Goal	Metric	Target	Number in D4.4	Achieved number
1	IoT operability	Number of IoT silos	5	5	6
2	IoT interoperability	Number of IoT silo pairs	3	-	8[3]
3	Ledger use	Number of distributed ledgers	5	5	7
4	Interledger use	Number of distributed ledger pairs	3	2	4
5	Ledger independence	Number of BP samples classified into success or partial success	3	2	3
6	Privacy designed in as a fundamental requirement	Number of operational GDPR features referenced and supported	5	5	5
7	Device owner payments across ledgers	Number of ledger pairs supporting value transfer	2	1	2
8	Data sovereignty	Number of pilot use cases utilizing data owner data sovereignty features and data owner is from a different silo than the storage silo	3	4	4[4]
9	User responsiveness	Number of seconds user gets response for an action initiated by the user			Testbed measurements reported in D4.4. Joint analysis of testbed and pilot measurements in Sections 4.1.4, 4.2.7, 4.3.3, 4.4.5
10	System performance	Acceptable system performance for users and pilots			Testbed measurements reported in D4.4. Joint analysis of testbed and pilot measurements in Sections 4.1.4, 4.2.7, 4.3.3, 4.4.5

The additional results produced after the submission of the previous evaluation deliverable D4.4 (submitted April 2020) have changed (improved) the architecture KPIs referring to IoT operability (KPI #1), IoT interoperability (KPI #2), number of distributed ledgers (KPI #3), number of distributed ledger pairs (KPI #4), ledger independence (KPI #5), and number of ledger pairs supporting value transfer (KPI #7). Moreover, additional results for the user responsiveness and

[3] This KPI is addressed by the WP5 pilot validation work. Deliverables 5.3 (End-to-end Platform Validation) and D5.4 (Final Validation and Replication Guidelines) contain more details. The evaluation results in the current and previous WP4 deliverables consider emulation/simulation scenarios, hence they do not consider this metric.
[4] This number refers to emulated pilot scenarios presented in the previous deliverable D4.3, D4.4, and the current deliverable.

general system performance jointly analysing the emulation and pilot results are presented in Section 4 of this deliverable.

Next, we discuss in detail the achieved number for each architecture KPI.

KPI #1: IoT operability

The objective of the KPI on IoT operability is to prove the applicability of the SOFIE federation architecture and its components to existing IoT silos. The corresponding metric is the number of IoT silos where the architecture has been applied. The current version of the architecture and a subset of its components have been applied and evaluated to the following six scenarios/silos:

- IoT resource access
- FSC
- DEFM
- DEDE
- CAMG
- Secure Marketplace for Access to Ubiquitous Goods (SMAUG)

Each scenario utilizes different features of the architecture and its components, such as authentication and authorization, recording of data or hashes and execution of smart contracts in private/permissioned and public DLTs.

In Section 4 of the current deliverable, we present new evaluation results for the four pilot scenarios. The previous deliverables D4.3 and D4.4 contain additional results for the four pilot scenarios based on emulation. Results for the IoT resource access scenarios are presented in the previous deliverables D4.3 and D4.4. Finally, the SMAUG use case is presented in deliverable D5.3 and validation results are reported in D5.4.

Hence, the achieved number for this KPI is 6.

KPI #2: IoT interoperability

The KPI on IoT interoperability focuses on the application of the architecture and its components to allow communication between different silos. For example, these silos can involve the IoT platforms of different entities, such as the smart farming platform, the transportation platform, and the logistics platform in the FSC scenario. The corresponding metric that represents this KPI is the number of IoT silo (platforms) pairs that exchange data through the SOFIE architecture. The interoperability of the IoT platforms will necessarily consider the SR component. The evaluation results for the pilot scenarios that are reported in Section 4, and those in the previous deliverables D4.3 and D4.4, emulate the various entities (IoT platforms) and focus on cross-ledger interactions. They do not consider the interoperability of the platforms at the semantic level. Based on this, the evaluation scenarios do not contribute to this KPI. Rather, the IoT platform interoperability KPI is addressed by the SOFIE pilot evaluation work. Specifically, as reported in deliverable D2.5 (Federation Framework, 2nd version), D5.2 (Initial Platform Validation), and D5.4 (Final Validation & Replication Guidelines). Specifically,

- the FSC pilot has demonstrated the interoperability of three IoT platforms, namely the smart farming IoT platform, the transportation IoT platform, and the logistics IoT platform, hence two IoT platform pairs,
- the DEFM pilot has demonstrated the interoperability of the electric vehicle and supply equipment platforms,
- the DEDE pilot has demonstrated the interoperability of a national data hub platform with smart meter platforms,
- the CAMG pilot has demonstrated the interoperability of the gaming, IoT beacon services, and asset platforms, hence two IoT platform pairs, and

- deliverable D5.4 presents two cross-pilot scenarios: cross-pilot data exchange and cross-pilot reward exchange.

The above yields a total of 8 IoT platform pairs.

KPI #3: Number of ledgers used

The third KPI refers to the number of ledgers used. The DLTs that have been used in the evaluation experiments reported in the previous evaluation deliverables D4.3 and D4.4, in the current deliverable, along with the pilots defined in WP5, include the following ledgers:

- Rinkeby[5] public Ethereum test network
- Ropsten public Ethereum test network
- Private Ethereum network (Testbed)
- Hyperledger Fabric (Testbed)
- Hyperledger Indy (Testbed)
- Sovrin testing network
- KSI blockchain

Deliverable D4.4 contains evaluation results for the first four DLTs. The PDS and IAA code related to DIDs and VCs has been tested and works with Sovrin. The interaction with Sovrin is orthogonal (since Sovrin is a registry and all calculations take place locally, in the entities considered in our evaluation scenarios) and does not affect the performance results. Finally, KSI is utilized in the DEDE pilot and the FSC pilot and discussed in the current deliverable in relation to the privacy properties for the FSC pilot.

Based on the above, the number achieved for the ledger use KPI is 7.

KPI #4: Number of ledger pairs

The fourth KPI refers to the number of ledger pairs investigated. The evaluation experiments reported in the previous deliverable D4.3 focus on the interoperation of the Rinkeby and Ropsten public Ethereum testnets with a private Ethereum network, while D4.4 contains results investigating the cross-ledger interaction of public Ethereum testnets and a private Ethereum network for the food supply chain and mobile gaming cases. D4.4 also reports evaluation experiments considering the interoperation of the public Ethereum testnets with a Hyperledger Fabric permissioned ledger. The PDS component can generate an access token based on the verification of a client's credentials, using Hyperledger Indy. Then this access token can be recorded in an Ethereum smart contract and used by the IAA component. Additionally, for the Food Supply Chain scenario, in the current deliverable we assess the use of KSI together with a consortium ledger (private Ethereum) and public ledger (public Ethereum). Moreover, as discussed in deliverables D3.5 (Final Business Platform Integration Report) and D5.4 (Final Validation & Replication Guidelines), the SMAUG use case considers a marketplace ledger (private Ethereum), and authorization ledger (private Ethereum), and Hyperledger Indy.

Based on the above, the scenarios considered in the emulation and pilot investigations have demonstrated the interaction between the following ledger pairs: 1) public Ethereum and private Ethereum and 2) Ethereum (both public and private) and Hyperledger Fabric, 3) Ethereum and Hyperledger Indy, and 4) Ethereum (private and public) and KSI. Hence, the number achieved for the interledger KPI is 4.

KPI #5: Ledger independence

The goal of the ledger independence KPI is to demonstrate the capability of developing applications using sufficient abstractions that allow the applications to run over different ledger technologies.

[5] https://www.rinkeby.io

The experiments in the previous deliverables D4.3 and D4.4 demonstrate the implementation of functions and services related to IoT resource access both in a private Ethereum network and a Hyperledger Fabric permissioned ledger. Specifically, the results in D4.4 utilize VCs as an authorization grant compatible with the OAuth 2.0 authorization framework. The proposed approach should be compatible with any ledger technology that follows W3C specifications; we have verified our constructions in Hyperledger Indy and in Sovrin's testing network. Additional results in D4.4 investigate the use of authorization tokens backed by Ethereum ERC-721 tokens. The results in deliverable D4.3 considered off-chain authorization tokens. Based on the above, our experimental results have demonstrated that subcomponents of the IoT resource access application, and both the PDS and the IAA components can be deployed on different ledgers (Ethereum, Hyperledger Fabric, and Hyperledger Indy). Moreover, the pilot results in the joint analysis with emulation results for the food supply chain consider the KSI blockchain, used in conjunction with a public ledger (Ethereum) and a consortium ledger (private Ethereum instance).These ledger alternatives are considered in different scenarios, pertaining to the emulation scenarios and the actual pilot implementation, that have clearly defined baselines and whose evaluation identifies their relative strengths, weaknesses, and tradeoffs, as discussed in the current deliverable.

The use of different DLTs for implementing various services is demonstrated by the results for CAMG in D4.4, where various functions that include key gaming functions, advertisement functions, and token and reward functions can be implemented on both an Ethereum network (private or public) and Hyperledger Fabric. Finally, the Secure Marketplace for Access to Ubiquitous Goods (SMAUG) use case has utilized and validated all six SOFIE federation components.

Based on the above, the number achieved for the ledger independence KPI is 3 (IoT resource access, food supply chain, and mobile gaming scenarios).

KPI #6: Privacy

The privacy KPI concerns the compliance of the SOFIE architecture with the GDPR and its metric is the number of operational GDPR features referenced and supported. There are various features of the SOFIE architecture that are related to privacy and GDPR. The relevant GDPR articles based on the GPDR checklist[6] are identified in parentheses. Firstly, as discussed and investigated in Section 3, the SOFIE architecture does not record personal data to immutable ledgers. This is necessary to support the "right to be forgotten" (GDPR Article 17 – Right to erasure ('right to be forgotten')). Instead, the immutability of data recorded in local databases is ensured by recording hashes of the data in public ledgers. Secondly, in the various scenarios only the minimum set of data is stored in a public ledger, in order to ensure the correct operation and functionality that pertains to the specific scenario. Also, based on DIDs the SOFIE architecture can support pseudonymisation (GDPR Article 25 – Data protection by design and by default, GDPR Article 32 – Security of processing).

SOFIE's applications do not process data without having permissions granted by users (consent), e.g., in the CAMG scenario when the user installs the app, a pop-up screen is displayed asking for permission to access storage, location, etc. Furthermore, in the IoT resource access use case, consent is provided through access tokens, which can be revoked or are valid for a specific time duration, and whenever VCs are used as authorization grants, a user can select which claims of a VC can be revealed (GDPR Article 7.3 – Conditions for consent). Authorizations are recorded in an immutable manner on a DLT, which allows verification in a non-repudiated way that the user (owner of data) provided consent (GDPR Article 7.1 – Conditions for consent, Article 6 – Lawfulness of processing). Moreover, through

[6] GDPR checklist for data controllers: https://gdpr.eu/checklist/

the PDS and IAA components, IoT resource owners can provide access to clients (GDPR Article 20 – Right to data portability).

Based on the above discussion, the following GDPR features are referenced and supported by the SOFIE architecture: 1) 'right to be forgotten', 2) pseudonymisation, 3) user selects which claims can be revealed, 4) authorizations immutably recorded on a DLT serve as proof of user consent, and 5) users can provide access to their data. This gives a total of 5 operational GDPR features referenced and supported by the SOFIE architecture.

KPI #7: Number of ledger pairs supporting value transfer

Whereas KPI 4 on "Interledger use" focuses on the interoperability, in general, between different ledgers, KPI 7 concerns the transfer or, more accurately, the exchange of value, between different ledgers. An example of such an exchange of value is discussed in detail in the previous deliverable D4.3, which involved the exchange of a payment token, stored in a public Ethereum blockchain, with an access token stored in a private Ethereum blockchain. This exchange is performed using functionality of the Interledger component. The exchange of value between Ethereum and Hyperledger Fabric utilizing the Interledger component is illustrated in the IoT resource access scenarios reported in D4.4.

Based on the above, our results have demonstrated value transfer between two ledger pairs: 1) public Ethereum and private Ethereum and 2) Ethereum (both public and private) and Hyperledger Fabric.

KPI #8: Data sovereignty

The data sovereignty KPI is related to the ability of data owners to reject or allow access, possibly for a specific time interval, to their data. This KPI can be verified with the number of pilot use cases utilizing data owner data sovereignty features, where the owner can be in a different silo than the storage silo.

All scenarios presented in the previous deliverables D4.3 and D4.4 can leverage the PDS component of the SOFIE architecture to achieve data sovereignty. Solutions related to IoT resource access with specific functionality and features are investigated in detail in the previous deliverables D4.3 and D4.4. The number achieved for this KPI is 4 (same as in the previous deliverable D4.4), based on the emulated scenarios considered.

Finally, we discuss the relation of the architecture KPIs with the KPIs corresponding to the SOFIE's project objectives from the Description of Work. The KPI related to the first objective (O1) involves federating at least 5 IoT platforms. This corresponds to the first architecture KPI in Table 8, for which the achieved number is 6. The second objective (O2) KPI involves the support for simultaneous use of at least 3 ledger technologies. Based on the results discussed above for the interledger KPI (architecture KPI #4 in Table 8), the SOFIE architecture supports four ledger technologies: Ethereum, Hyperledger Fabric, Hyperledger Indy, and KSI. The third objective (O3) KPI involves transactions involving at least three ledgers. The Food Supply Chain pilot and scenarios investigated in the current deliverable and the SMAUG use case consider interactions between three ledgers. Specifically, the Food Supply Chain scenarios consider hierarchies involving consortium ledger (private Ethereum instances), a public ledger (public Ethereum), and KSI, while the SMAUG use case considers a marketplace ledger (private Ethereum), an authorization ledger (private Ethereum), and Hyperledger Indy.

3 Component evaluation

This section presents evaluation results for SOFIE's framework components. The results presented in this section focus on the internal (basic) functionality of the components. Evaluation results that consider the functionality of the components within the pilots are presented in Section 4. Compared to the previous evaluation deliverable D4.4, the current deliverable contains new results for the four components: Interledger, Privacy and Data Sovereignty, Identification, Authentication, and Authorization, and Marketplace. Specifically, the Interledger component is assessed in two new scenarios that involve game asset transfer and transfer of funds. For the Privacy and Data Sovereignty (PDS) component, we evaluate the privacy module, which implements local differential privacy mechanisms. For the Identification, Authentication, and Authorization (IAA) component we evaluate W3C's compliant verifiable credentials (VCs). These differ from the VCs as implemented by Hyperledger Indy that were evaluated in the previous deliverable D4.4 in that the latter exclusively utilize Zero-Knowledge Proofs (ZKP), which results in larger digital signature sizes and higher computation overhead. Finally, for the Marketplace component we investigate multiple winner selection, which extends the basic functionality of the Markeplace component.

Furthermore, in this section we also present the requirements of each component, defined in deliverable D2.4, and the related evaluation/emulation scenarios and/or how these requirements are met. The previous deliverable D4.4 linked the requirements to the corresponding evaluation/emulation scenarios contained in D4.4. Finally, we note that the evaluation results presented in the previous deliverables D4.3 and D4.4 and in the current deliverable consider simulation and emulation scenarios that have been implemented in corresponding testbeds. These results are distinct from the component validation results contained in D2.7 (Federation Framework, final version) and the pilot validation results contained in D5.4 (Final Validation & Replication Guidelines), since the evaluation work in WP4 has a wider scope than the pilots, seeking to evaluate many potential alternatives going beyond what is possible within the pilots.

3.1 Interledger

The main purpose of the SOFIE Interledger component is to enable transactions between actors and devices belonging to different (isolated) IoT platforms or silos. Each IoT silo either utilizes or is connected to one or more DLTs. The Interledger component then enables interaction between these DLTs. The evaluation of the previous deliverables presents and analyses the transaction costs using the Interledger component across Ethereum and Hyperledger Fabric ledgers. The current deliverable adds to that aspect by comparing the costs and delay of conducting the same load of inter-ledger transaction with the SOFIE Interledger component and doing it manually on two separate ledgers. In addition to the quantitative analysis, this section also evaluates the functional aspect on how the automatic operation is performed without user interaction and atomicity is achieved.

The Interledger component can run one or more Interledger instances in parallel. Each instance provides a unidirectional transaction with clear roles: one ledger acts as the Initiator that triggers the transaction, and the other ledger(s) acts as the Responder(s) that reacts to the trigger. The Initiator can also send a data payload to the Responder(s), but the Responder(s) can only reply with a success/fail status, so the Interledger functions as a unidirectional data transfer from the Initiator to the Responders. Different types of ledgers can act as the Initiator and/or Responder provided that a corresponding DLT adapter is used. The Interledger Core then acts as the bridge in the center to connect the Initiator and Responder adapters.

3.1.1 Experiment setup

The previous evaluation deliverables D4.3 and D4.4 considered the authorization use case for constrained IoT resources, where users deposit some funds in order to receive a decryption key for accessing an IoT resource. An instance of the component that implements the interledger functionality is connected to a Hyperledger Fabric permissioned blockchain and to a public Ethereum testnet. The end-to-end delay is evaluated in D4.4 for the transactions on the public ledger (Ethereum) and Hyperledger Fabric, which are interconnected with the Interledger component. Interaction with other components and functionalities related to decentralized authorization in constrained IoT environments are also considered.

For a more extensive quantitative evaluation of the costs and performance of the SOFIE Interledger component, its use for asset transfer is assessed in the experiment below related to a gaming system. Some applications rely on multiple ledgers to utilise assets in different ways, but the asset is allowed to be active in only one ledger at a time. An example is a game system, where one ledger is used to maintain all the assets available in the game, another is used as the marketplace to trade the assets between the gamers — and an asset being transferred cannot be used in the game and vice versa.

As shown in Figure 1, the SOFIE Interledger component can be utilised to build a protocol to ensure that the above rule is satisfied. Here, a gamer wants to trade an asset which is currently active in the Game Asset Ledger. First, the gamer calls a smart contract function from the Game Asset Ledger that initiates the transfer process by changing the status of the asset to *Transfer Out* so that the asset can no longer be used in the game, and then emitting an event. The Interledger component in turn calls the receive function on the Marketplace Ledger that activates the asset in that ledger by changing its status to *Here*. Finally, the Interledger component calls a function from the Game Asset Ledger to deactivate the assets there by changing the status to *Not Here*. The same process takes place in the reverse direction when the asset is returned to the Game Asset Ledger.

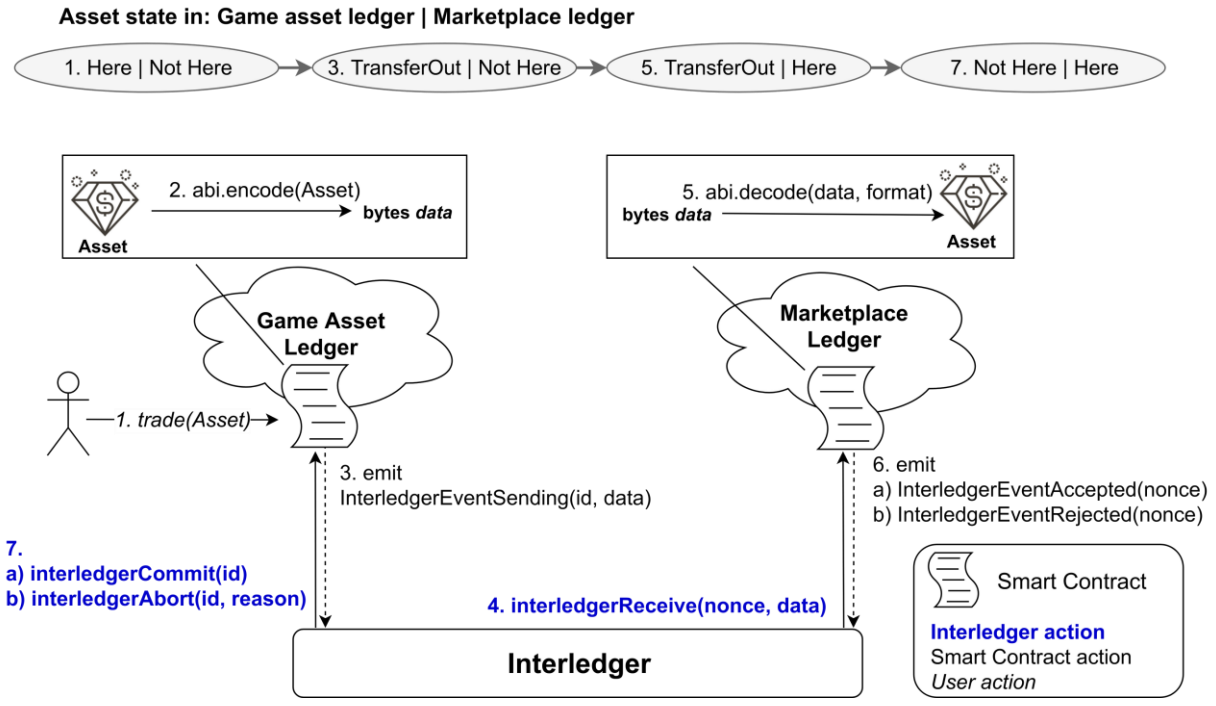

Figure 1: Game asset transfer using the SOFIE Interledger component.

The other experiment, namely the HTLC-based fund transfer, is conducted to evaluate whether atomicity can be achieved by using the latest SOFIE Interledger component. A Hash Time-Locked Contract (HTLC) can be utilised to ensure that linked transactions on different ledgers happen atomically, i.e., they either both complete successfully or both fail. Notice that the previous evaluation in D4.3 and D4.4 utilize the HTLC for transfer of assets as well. As shown in Figure 2, once Alice reveals the secret to transfer money to her account, Interledger automatically triggers the related money transfer to Bob's account thus saving Bob the effort of monitoring when Alice actually reveals the secret. A key difference between the game asset transfer scenario and the fund transfer scenario is that the first involves transferring assets to a different ledger, while remaining to the same owner (gamer).

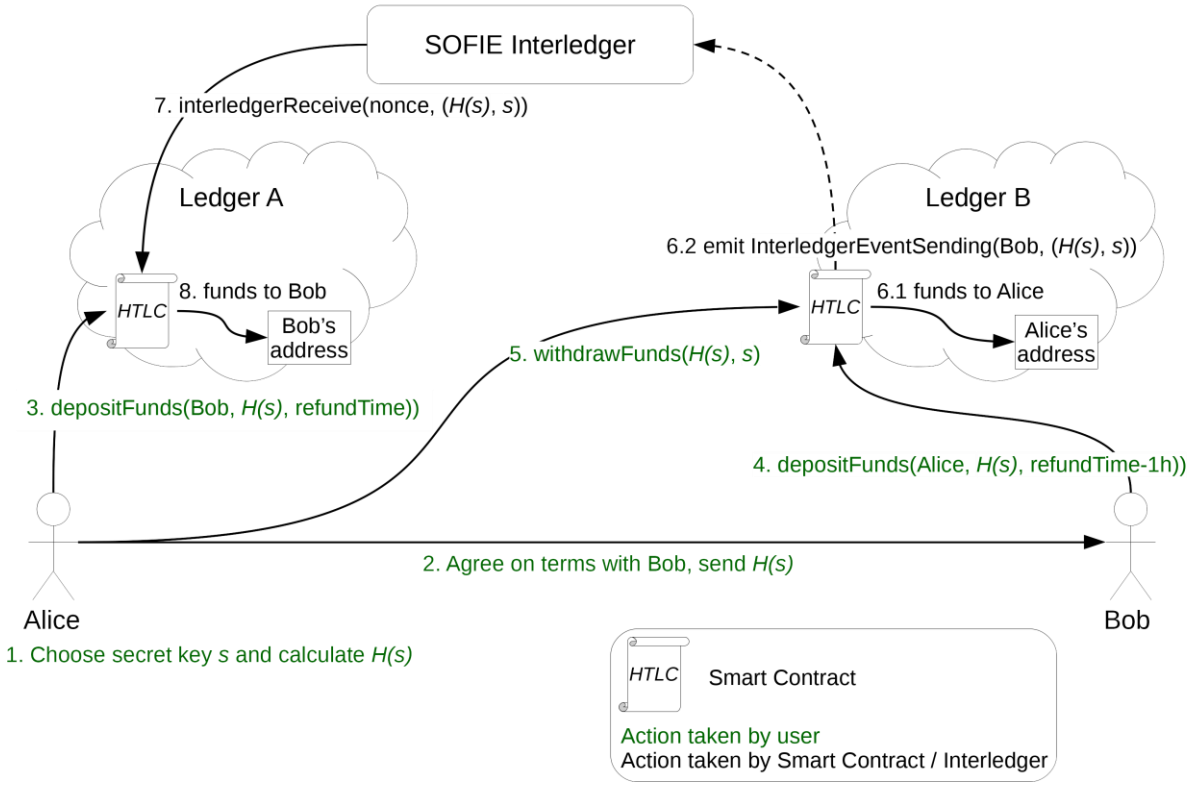

Figure 2: Interledger automates the HTLC-utilising transfer of funds.

3.1.2 Results

This section describes the results shown in the Game asset transfer and HTLC atomic transfer use cases, to show how the Interledger component enables the automatic transaction across ledgers, without the need for active user interaction, and how the atomicity of such a transaction can be achieved. The results of the previous evaluation about the authorization use case for constrained IoT resources can be found in the corresponding section in D4.3.

To evaluate the quantitative impact of utilising the Interledger component between two ledgers, several experiments were conducted based on the game asset transfer use case described in the previous section. The asset transfer can alternatively be performed using the SOFIE Interledger (through events), or manually (without the Interledger component and the related events) by actively checking the state of assets in both ledgers.

Table 9: Experimental results for deployment of a single GameToken smart contract.

Action	For manual transfer		For transfer with Interledger	
	Cost gas / euro	Time s	Cost gas / euro	Time S
Contract deployment	3601.3k / 18.01	2.28	3829.4 / 19.15	2.30

Table 10: Experimental results for asset transfer actions.

Action on asset	For manual transfer			For transfer with Interledger		
	Initiator gas / euro	Responder gas / euro	Time s	Initiator gas / euro	Responder gas / euro	Time s
Creation	164.6k/0.82	116.3k/0.58	1.85	164.6k/0.82	116.3k/0.58	1.89
Transfer	48.9K/0.24	48.3k/0.24	3.02	51.0k/0.26	50.3k/0.25	3.07

The costs in terms of Ethereum gas / euros and the performance measured in delay, for both the deployment and the asset transfer are given in Table 9 and Table 10. Here it is assumed that a gas fee of 10 GWei is taken in all the operations, as in the previous deliverables, and a recent average conversion rate of 500 Euro/ETH is used for the calculation. The deployment cost for the Interledger-compatible GameToken contract increased by 6% compared to the basic GameToken contract and the time overhead is less than 1%, so overall the overhead of deploying the Interledger-compatible smart contracts is negligible. Also, it should be noted that the smart contract only has to be deployed once and it can then be used for managing all the assets, so the costs and overhead can be amortised over the lifetime of all the assets. As can be seen from Table 10, regardless of the communication paradigm between ledgers (i.e., either direct or via the Interledger component) the cost of asset creation is identical, however, there is only a 2% time penalty with the Interledger-compatible smart contract.

It is worth noting that the direct transfers implementation was set up only for the purpose of performance analysis. In practice, it is not a feasible solution for such use cases as it assumes the solution is aware of when the asset transfer takes place and then actively checks the ledger status on both sides of the whole process.

The table below relates the evaluation scenarios to the corresponding Interledger component requirements identified in deliverable D2.4 and describes how each requirement is achieved.

Table 11: Requirements for the SOFIE Interledger component.

Req. ID	Requirement Description	Priority	Evaluation/Emulation Scenarios and how the requirement is achieved
Interledger			
RF01	User interaction is not required for interledger operations.	MUST	In the game asset transfer experiment, no user action is required to trigger the transfer. It is completed automatically by utilizing the SOFIE Interledger component.
RF02	There should be support for atomic interledger operations.	SHOULD	The HTLC fund transfer between two agents is done in atomic manner, which means the success or failure of such transaction is consistent across ledgers.

As shown in the game asset transfer experiment, the SOFIE Interledger component meets requirement RF01, since in the way that the whole process of game asset transfer can be automatically conducted, no manual user interaction is needed to activate or deactivate the asset state on different ledgers.

The other experiment of HTLC fund transfer from Alice to Bob clearly shows that atomic operations are supported by the SOFIE Interledger component, where partial failure of a particular transaction will not cause inconsistent state across ledger. In this manner, both of the requirements listed for the SOFIE Interledger component are evaluated and satisfied.

3.2 Privacy and Data Sovereignty

The Privacy and Data Sovereignty (PDS) component is composed of two modules: the Data Sovereignty module and the Privacy module. The Data Sovereignty module implements an OAuth2.0 authorization server, and its evaluation was performed in D4.4. In this deliverable we are focusing on the Privacy module of the PDS component.

The Privacy module enables the creation of "privacy preserving surveys". These are surveys that allow users to add "noise" to their responses, using local differential privacy mechanisms. The addition of the noise prevents 3rd parties from learning meaningful information about specific users, but at the same time aggregated statistics can be extracted. The accuracy of the extracted statistics depends on the number of responses.

Due to the local differential privacy mechanism used, the PDS component supports only “multiple-choice” questions. For example, supposedly an entity is interested in learning the “average age of all AUEB students”: it should construct a survey that includes a single question (“What is your age?”) and *n* possible responses (e.g., “A:18, B:19, C:20, …K: >28”). Then each student would select the correct answer, apply the local-differential algorithm provided by PDS, and submit her response. No matter the number of the responses, the age of a single student is never revealed to any third party.

3.2.1 Experiment setup

The privacy module of PDS uses the basic one-time RAPPOR[7] algorithm to implement local differential privacy. A drawback of this algorithm is that it requires a big number of responders so as to extract accurate results. In order to evaluate the accuracy of the extracted results, we perform the following experiment. We invoke a number of PDS instances (the number of instances is used as a variable in our experiments) and we ask them to respond to a survey that includes 20 possible choices. Each instance selects its response using a normal distribution with mean 11 and deviation 2. The following figure shows the distribution of the responses. The horizontal axis is the choice value, and the vertical axis is the percentage of instances that responded with that choice. For example, 21% of the PDS instances selected the value 11, 16% the value 15, and so forth.

[7] U. Erlingsson, V. Pihur, and A. Korolova, “RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response,” in Proc. of ACM SIGSAC Conference on Computer and Communications Security, 2014

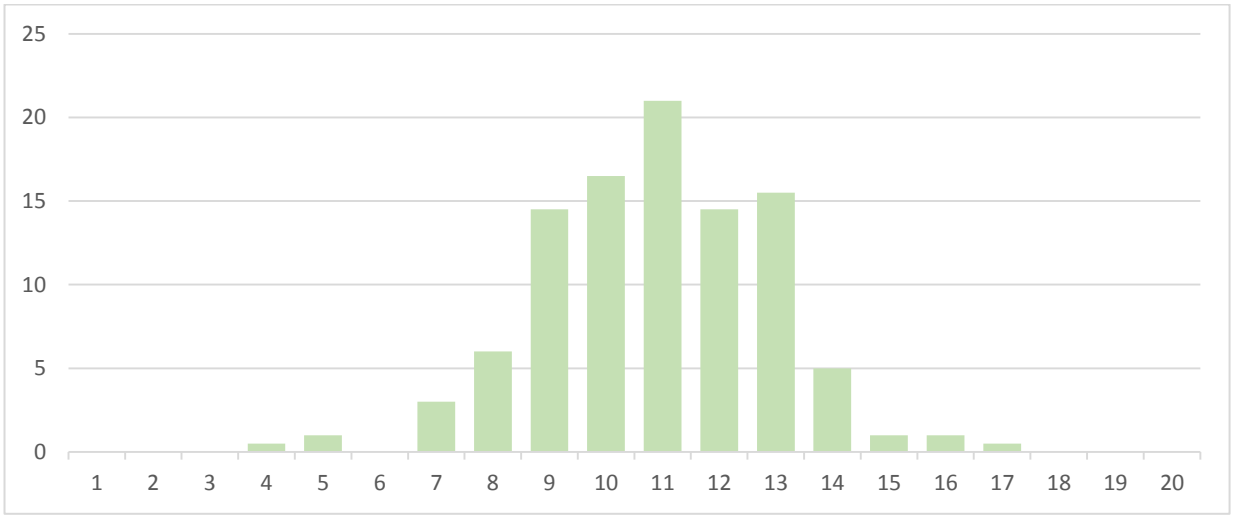

Figure 3: Real distribution of the responses for differential privacy experiments.

Then each PDS instance applies the RAPPOR algorithm and submits the “noisy” outcome to a centralized server.

3.2.2 Results

We now plot the real distribution of the responses (green bar), the extracted distribution based on the noisy responses (black bar), and their difference (black line): the closer to the horizontal axis the black line is, the more accurate the results are. The following diagrams concern experiments with 200, 600, 1000, and 1000 PDS instances. It should be noted that the accuracy of the extracted results is not affected neither by the number of choices, nor by the distribution of the real responses.

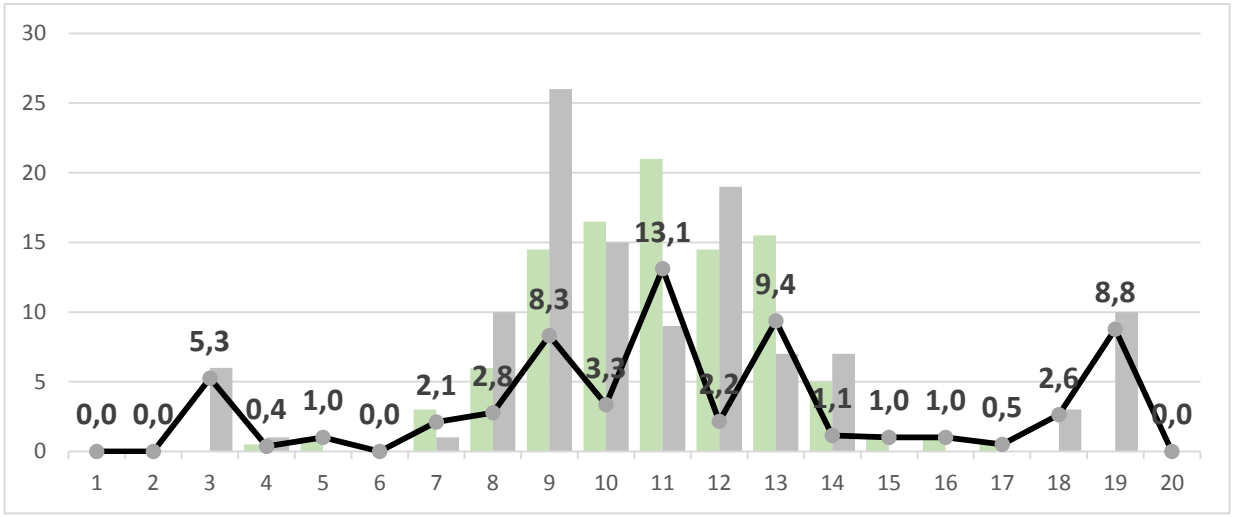

Figure 4: Distribution of responses for 200 PDS instances.

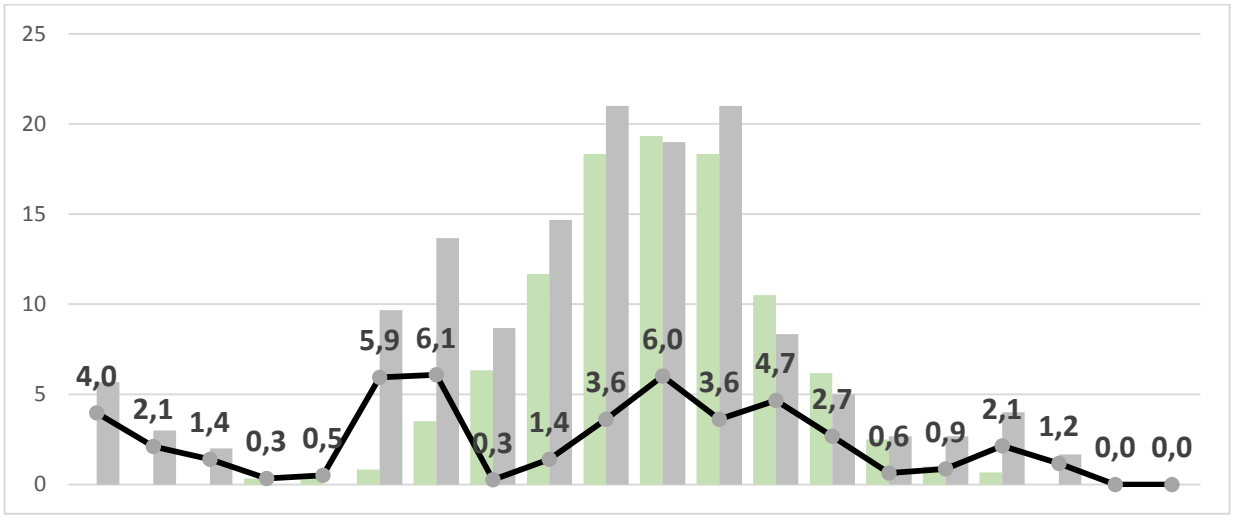

Figure 5: Distribution of responses for 600 PDS instances.

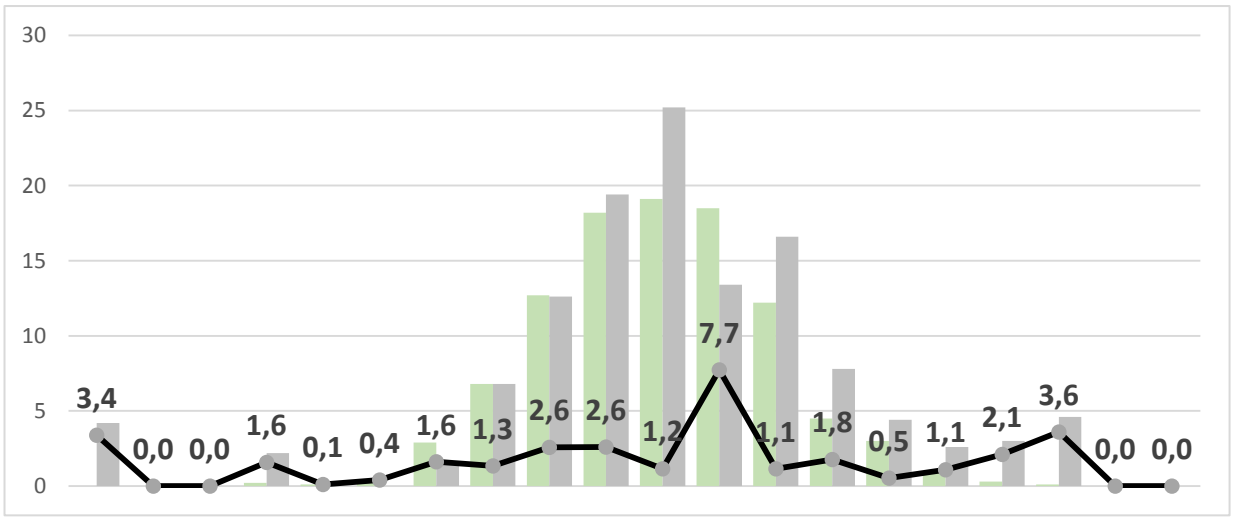

Figure 6: Distribution of responses for 1000 PDS instances.

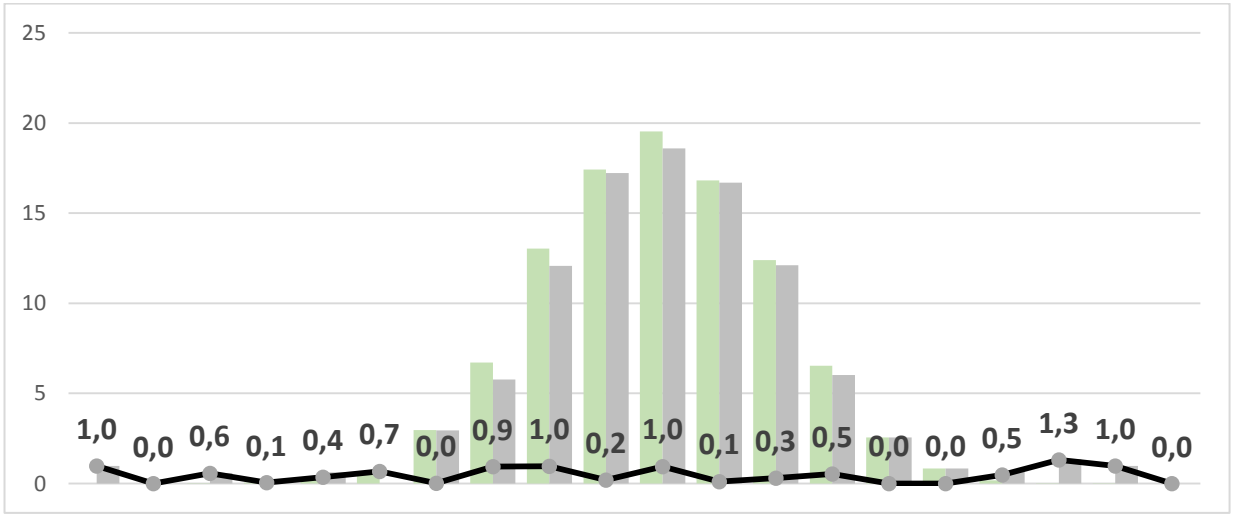

Figure 7: Distribution of responses for 10000 PDS instances.

The table below relates the evaluation scenarios to the corresponding PDS component requirements identified in deliverable D2.4 and describes how each requirement is achieved.

Table 12: Requirements for the SOFIE PDS component.

Req. ID	Requirement Description	Priority	Evaluation/Emulation Scenarios and how the requirement is achieved
PDS			
RF10	SOFIE must follow the data minimisation principle for personal data and only request or process what is necessary for the situation and purpose.	MUST	Evaluated in D4.4
RF11	Processing of individual's personal data is justified by a valid legal basis, e.g., a valid consent from the individual.	MUST	Evaluated in D4.4
RF12	Consent to process personal data must be revocable at any time.	MUST	Evaluated in D4.4
RF13	SOFIE must allow organisations and actors to manage (create, update, delete) their own data privacy policies.	MUST	Evaluated in D4.4
RF14	SOFIE should support user privacy even when aggregate statistics are made public (e.g., using differential privacy mechanisms).	SHOULD	Evaluated in this section. The local differential privacy mechanism is implemented inside the PDS component.

3.3 Identification, Authentication, and Authorization

In previous deliverables we evaluated how the Identification, Authentication and Authorization (IAA) component can be used to authenticate users using Decentralized Identifiers (DIDs) (evaluated in D4.3), as well as using JSON Web Tokens (JWTs), which are generated by the PDS component (evaluated in D4.4). In this deliverable we evaluate user authorization using W3C's compliant verifiable credentials (VCs). In D4.4 we evaluated VCs as implemented by Hyperledger Indy (in the context of PDS). Although W3C-VCs and Indy-VCs share the same goals, they have key differences. Indy-VCs are using Zero-Knowledge Proofs (ZKP) exclusively. Although ZKP enhance user privacy they have some drawbacks. Firstly, their size is bigger compared to traditional digital signatures. Secondly, they introduce a bigger computational overhead. Thirdly, ZKP verification requires access to the VC "schema": with Indy this schema is retrieved from a blockchain, hence VC verification requires a look up to an Indy-based blockchain.

Details on how W3C-VC based authorization is implemented are provided in D2.7. Here we evaluate the communication and computation overhead of this mechanism.

3.3.1 Experiment setup

Our evaluation scenario assumes an IAA instance acting as a HTTP proxy. The IAA instance is configured with rules for verifying VCs: all authorized requests are forwarded to a protected HTTP resource.

VCs are JSON-encoded and include a list of URLs for which a client is authorized. The VC verification process includes the following steps:

- The digital signature of the token is verified. Digital signatures are generated using Ed25519.
- The second step verifies that the token contains the required claims. This functionality is implemented using JSOPath, a JSON query language.
- The third step verifies that the user is the legitimate holder of the credential. This is achieved by verifying a digital signature using the public key of the user, which is also included in the VC.

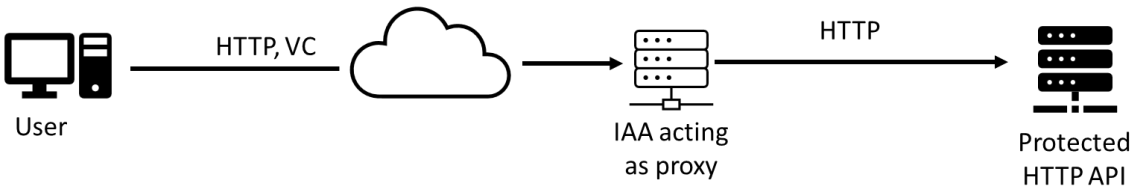

Figure 8: W3C VC based authorization entities and their interaction.

3.3.2 Results

We have implemented the IAA component in Python3. All related cryptographic operations are implemented using JWCrypto[8]. JSONPath functionality is provided by the jsonpath-ng library[9].

VCs are included in HTTP headers, for this reason they are encoded using the Base64URL format.[10] The following listing includes a typical VC that authorizes a user to access a single URL. The Base64URL encoding of this VC is 1252 bytes.

[8] JWCrypto home page, available at https://jwcrypto.readthedocs.io last accessed 11/2020
[9] Jsonpath-ng home page, available at https://pypi.org/project/jsonpath-ng/ last accessed 11/2020
[10] https://base64.guru/standards/base64url

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://mm.aueb.gr/contexts/access_control/v1"
  ],
  "id": "https://www.sofie-iot.eu/credentials/examples/1",
  "type": ["VerifiableCredential"],
  "issuer": "did:nacl:46IigI__T3vZF0izvYNQBF0xaYuYPYH...EHU",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:nacl:J7VMbVcUt040x0yR8tqEP40sIild4NNgkZXKrCtfCqA",
    "type": ["AllowedURLs"],
    "acl": [
      {
        "url": "http://sofie-iot.eu/device1",
        "methods": ["GET", "POST"]
      },
      {
        "url": "http://sofie-iot.eu/device2",
        "methods": ["GET"]
      }
    ]
  },
  "proof": {
    "@context": "https://w3id.org/security/v2",
    "type": "Ed25519Signature2018",
    "created": "2020-12-23T13:10:05Z",
    "verificationMethod": "did:nacl:46IigI__T3vZF0...izvYNQaYu",
```


Figure 9: W3C VC based authorization headers.

In order to prove VC ownership, we adapt OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)[11]. From a high-level perspective, to prove VC ownership, a user generates a signature that includes the requested HTTP URI and a nonce. The signature is also included as an HTTP header (referred to as DPoP). Therefore, VC ownership verification does not require any additional communication. The size of a DPoP is 511 bytes.

Overall, IAA has to verify two digital signatures, one for verifying VC integrity (VC proof) and one for verifying VC ownership (DPoP). Both signatures are generated using Ed25519 but with different input. In an Xubuntu 20.04-based PC with intel i5 and 4GB of RAM, VC Proof verification requires 112 milliseconds whereas DPoP verification requires 4.5 milliseconds.

[11] D. Fett et al. "OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP)," available at https://tools.ietf.org/html/draft-ietf-oauth-dpop-02 last accessed 11/2020

Table 13: Requirements for the SOFIE IAA component.

Req. ID	Requirement Description	Priority	Evaluation/Emulation Scenarios and how the requirement is achieved
IAA			
RF03	Resource owners must be able to delegate the authentication and authorization tasks for their resources.	MUST	Evaluated in D4.4.
RF04	The IAA component must provide users the capability to revoke authorizations.	MUST	Evaluated in D4.4
RF05	The IAA component must allow individuals to control their personal information and digital identities (e.g., support self-sovereign identity technology).	MUST	In D4.4 we evaluated the use of DID and in this section we evaluated the use of W3C compliant Verifiable Credentials (VCs).
RF06	The IAA component must support secure, tamper-proof, and verifiable logging of transactions and events.	MUST	Evaluated in D4.4
RF07	The IAA component must support Role Based Access Control (RBAC).	MUST	Roles can be defined in a VC and then IAA can be configured with role-based policies. This functionality is evaluated in this section.

3.4 Marketplace

The goal of the SOFIE Marketplace (MP) component is to enable the trade of different types of assets in an automated, decentralized, and flexible way. The actors (buyers and sellers) can carry out trades by placing bids and offers using the MP component, which utilizes Ethereum smart contracts.

The implementation of the MP component exposes an API that offers two actions: a request and an offer. Through the *request* action an actor declares that an asset is available for sale, thus creating an auction, while through the *offer* action an actor declares his interest in buying an asset, thus placing a bid. These two actions are highly parameterizable, thus supporting the needs of diverse services.

The third fundamental action of the MP component is the *decide* action, which is the capability of the MP to select the best offer among a set of offers, thus declaring the "winner" of the request. Similar to the previous actions, the decide action allows the implementation of different policies for different services. In this section, we explore the decide action of the MP component that is exploited in the emulation of the Decentralized Energy Flexibility Marketplace Pilot, that is described in Section 4.2. In brief, the explored MP component supports the declaration of multiple winner-offers (when a single offer cannot satisfy the requirements of a request), in an effort to boost the effectiveness of the service. This feature increases the complexity of the decision-making process which, now, requires sophisticated methods to resolve conflicting winner-offers, that are discussed in detail in Section 4.2.3.

The performance results presented in this section consider two metrics: the response time, which is the time until a transaction is mined, and the execution cost. We expect the execution cost of smart functions to be higher compared to the cost measured in D4.4, since the current setup includes a more proficient and, in turn, resource-expensive decision function compared to the previous setup.

3.4.1 Experiment setup

The prototype implementation of the MP component introduces a series of Solidity-based smart contracts that are deployed in an Ethereum node. We have tested the prototype implementation of the MP component in a local Ethereum node that is built with the Ganache blockchain. With respect to the parameters of Ganache, we fixed the average block mining time to 15 s and set the block gas limit to 10,000,000 gas units. Both these values reflect the corresponding values in the public Ethereum main net. We deployed the smart contract and invoked the available functions through a JavaScript script that utilizes the web3.js Ethereum API.

3.4.2 Results

We assess the performance of the MP component in two directions: the response time, which is the time until a transaction is mined, and the execution cost, since transactions in Ethereum incur a cost. Regarding the response time, all functions that result in blockchain transactions, are mined within one mining period, which is 15 seconds, hence their expected response time is 15 seconds.

The actions of our system that involve the invocation of the smart contract functions incur some computational overhead. The following table shows the cost of deploying the smart contract in the Ethereum network, as well as the cost of operations performed by our system measured in gas units.

Table 14: Cost of MP component's smart contract operations.

Operation	Cost measured in gas
Contract Creation	3232k
Submit Request	256k
Submit Offer	285k
Decide Request	1714k
Close Request	92k
Delete Request	102k

The above results can be considered as the minimum cost for operating a marketplace-based service. Depending on the requirements of the service, e.g., the amount of data that an offer includes, the cost is expected to be higher. The execution costs of a realistic service are discussed in greater depth in Section 4.2.6.1, where we assess the execution cost of the emulated decentralized energy flexibility marketplace pilot.

Table 15 relates the evaluation scenarios to the corresponding MP component requirements identified in deliverable D2.4 and describes how each requirement is achieved. Given that the requirements are delivered by the Ethereum node, which our setup continues to exploit, the following table is identical with Table 23 of D4.4.

Table 15: Requirements for the SOFIE MP component

Req. ID	Requirement Description	Priority	Evaluation/Emulation Scenarios and how the requirement is achieved
Marketplace			
RF20	The marketplace must log the configuration of all trading actions (including offers, bids, parameters of resources, transactions etc.).	MUST	The pilot emulation scenario is investigated in Section 4.2. The marketplace component is implemented in Ethereum smart contracts. Hence, trading actions are recorded in the Ethereum blockchain, making them traceable. This is also discussed in deliverable D5.2.
RF21	The marketplace must provide actors the capability to post/claim offers and sell/negotiate/exchange/buy resources and digital objects.	MUST	The marketplace component offers two interfaces: Request Maker for sellers to create, manage and conclude auctions, and Offer Maker for buyers to participate and bid in auctions. These functions are illustrated in the emulation scenario of Section 4.2. The extended functionality that allows multiple winners enhances its capabilities.
RF22	The marketplace must support transparent trading of resources, i.e. the bids/offers matching process and the payments must be transparent.	MUST	The marketplace component utilizes Ethereum smart contracts to record bid/offers and payments, thus supporting transparency. This functionality is illustrated in Section 4.2.
RF23	The marketplace must provide evidence once trades have been completed and resources have been properly delivered to the buyers.	MUST	Ethereum transaction receipts, that describe the state of the blockchain after a transaction took place, can be used as evidence of a trade. This functionality is illustrated in Section 4.2.
RF24	The marketplace should allow integration of payment technologies.	SHOULD	Currently, the emulation of the marketplace component, as described in Section 4.2, supports transferring ETH coins and ERC20 tokens. The prototype implementation of the component does not disallow the integration of other payment technologies.

As we can observe from the above table, the requirements for the MP component can be met, since it is based on Ethereum, which, as any public blockchain technology, natively offers the majority of the required features.

4 Evaluation scenarios and joint analysis of testbed & pilot results

This section considers, as in the previous two deliverables D4.3 and D4.4, the SOFIE pilots and generalizes them into pilot-inspired scenarios by including alternatives not selected in the SOFIE pilots, while also abstracting various aspects to an appropriate degree, so that they can be emulated and/or simulated. Our focus is on identifying and quantifying the various tradeoffs of many potential alternative design decisions and the impact of various system parameters on the resulting performance.

The new results in this section include the joint analysis of emulation results and pilot results. In addition to presenting side-by-side and discussing the emulation and pilot results for some system performance KPIs (presented in Section 2.2), we also present new emulation results that exploit traces and statistics from the actual pilots. The main new results in the current deliverable compared to the previous deliverable D4.4 are the following: For the Food Supply Chain (FSC) we investigate a new scenario which most closely emulates the model employed in the pilot emulation. Together with the scenarios investigated in D4.4, the scenarios form a wide range of clearly defined baselines, whose evaluation identifies their relative strengths, weakness, and tradeoffs among ledger transaction cost, transactions delay, and privacy. For the Decentralized Energy Flexibility Marketplace, we investigate large-scale deployments of the service, which include a large number of EVs and CSs, in addition to realistic volumes of RPF caused by the green-energy production units. For the Decentralized Energy Data Exchange, we investigate the local differential privacy mechanism utilizing smart meter traces from the pilot. Finally, for the Context-Aware Mobile Gaming pilot we consider a new scenario involving an open advertising ecosystem for DLT-assisted mobile gaming, which highlights one of the main features of the SOFIE architecture and framework, openness.

4.1 Food Supply Chain

The FSC pilot explored the use of DLTs in designing supply-chain systems with guaranteed reliability and tamper-proof provenance and tracing data. The pilot's focus is on a FSC transferring agricultural products from producers to supermarkets, and its aim is to provide the following features:

- Traceability of agricultural products from the producer to the consumer
- Traceability of transport and storage conditions
- Resolution of disputes through audit procedure in case of customer complaints

This deliverable presents the results produced through the final evaluation and compares them to the respective results acquired through the pilot implementation. The pilot implementation focuses on a specific architecture, comprising one *consortium ledger*, one *public ledger*, and one *timestamping service* (*KSI*). The evaluation conducted in WP4 and reported in the current and the previous evaluation deliverables D4.3 and D4.4, spans a wider spectrum of design architectures, of which the pilot implementation architecture constitutes a single instance, which allows the comparison of alternative ledger organizations.

In addition to evaluation results presented in previous deliverables (D4.3 and D4.4), this deliverable also examines trade-offs concerning the audit procedure. These include privacy trade-offs, as well as trade-offs stemming from the frequency of logging sensor data and anchoring data (see next section).

4.1.1 Overview

The FSC pilot studies the end-to-end design of a food supply chain, focusing on tamper-proof provenance and tracing data leveraging DLTs. The pilot assumes a supply chain consisting of

five stages, namely the *farm* (Table-Grapes Field, TGF) where agricultural products grow, a transport company (TRA) transferring products from the farm to the depot, a depot (Storage & Distribution Center, SDC) collecting, storing, and dispatching boxes of agricultural products, a second transport company (TRB) transferring products from the depot to the supermarket, and a supermarket (SM). Table 16 lists the stages of the food supply chain.

Table 16: The stages of the FSC pilot use case.

Stage no.	Stage name	Abbreviation	Role
1	Farm (Table Grapes Field)	TGF	Grows table grapes and packs them into boxes
2	Transport A	TRA	Transfers boxes from TGF to SDC
3	Depot (Storage & Distribution Centre)	SDC	Collects, stores, and dispatches boxes
4	Transport B	TRB	Transfers boxes from SDC to SM
5	Supermarket	SM	Displays boxes and sells them to consumers

In the FSC pilot, agricultural products are transported end-to-end in *smart boxes* (or, simply, *boxes*). That is, products are packaged into boxes by the producer and they remain in these boxes throughout the entire transfer until they reach the consumer. Each box is equipped with an RFID tag, which is scanned and registered when the box is handed over by one stage of the supply chain to another.

The following two sections detail the models adopted by the evaluation (WP4) and the pilot implementation (WP5), respectively.

4.1.2 Model used in evaluation

The evaluation adopts the model illustrated in Figure 10 to assess various parameters and architecture scenarios of the FSC. Each stage of the FSC is equipped with a number of sensors, periodically reporting data concerning the conditions at this stage.

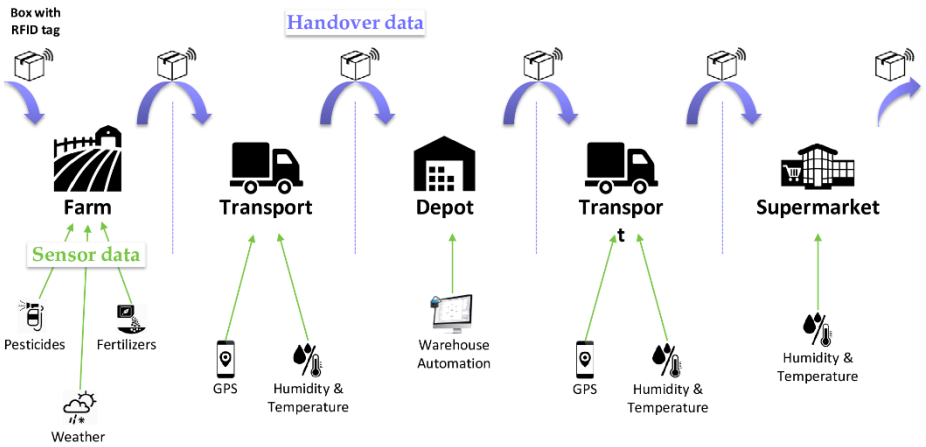

Figure 10: FSC model used in Evaluation.

As stated in deliverable D4.4, data collected, recorded, and processed in our FSC model can be split into three types:

- *Handover data* constitutes the first type. It involves all data related to passing around boxes across different stages in the FSC. Specifically, it includes a box' entry in the first stage to start a new transport session, its consecutive handovers between adjacent stages, and its exit from the FSC when the session is complete. Metadata of all aforementioned actions are also part of this data type, such as which employees were involved in each action, at what time each action happened, and some box state at that point, such as the box' weight. Handover data are recorded on demand, as soon as a handover action takes place.
- *Sensor data* forms the second type. It refers to all environmental and location data collected by each individual stage concerning the conditions that may affect produce quality. Sensor data are recorded periodically.
- *Anchoring metadata* constitutes the third type. Anchoring per se does not directly represent a state or action taking place in our model. Instead, it refers to a series of block hashes of any private ledger(s) employed in our model, which is expected to be stored in a public ledger to provide strong immutability guarantees.

To evaluate the FSC pilot we have considered four different architecture scenarios. Two of them use a single public ledger, while the other two employ hierarchical designs involving a combination of public and private ledgers.

More specifically, our evaluation has considered the five architecture scenarios discussed below. Scenarios 1 to 4 have been discussed and evaluated in the previous deliverable D4.4, while Scenario 0 is a new scenario which is considered in the current deliverable, and as we discuss later is closer to the pilot implementation.

Scenario 0 – Implementation Emulation: This scenario constitutes a special point in the evaluation space, serving as the scenario that most closely emulates the model employed in the pilot implementation. This scenario (Figure 11) comprises two ledgers: a shared ledger, corresponding to the Pilot Implementation's *consortium ledger*, and a public ledger. The former is an Ethereum instance run privately by all consortium members, while the latter is a public Ethereum instance, namely Ropsten. Both handover and sensor data are stored on the shared ledger. However, in order to provide high immutability guarantees, upon completion of a box session, a digest of all data concerning that box (i.e., a hash of its handovers and all relevant sensor data) is stored on the public ledger.

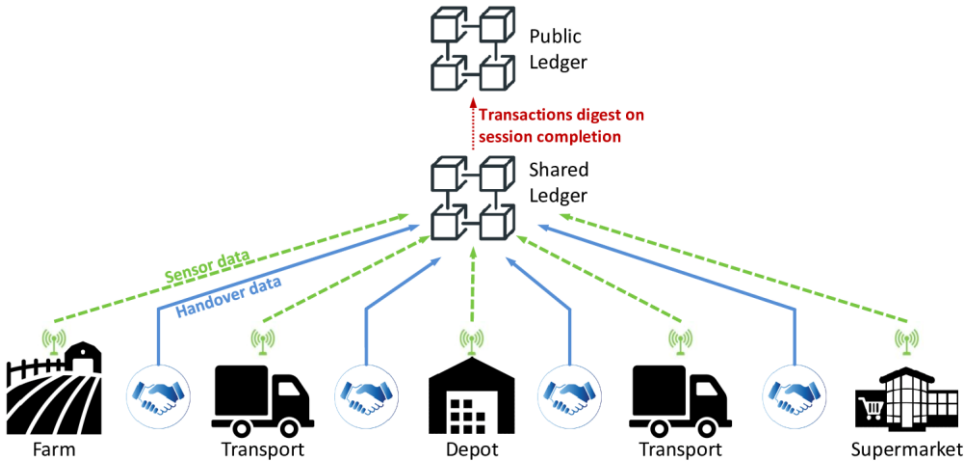

Figure 11: Scenario 0 – Public ledger: All sensor data (dashed green lines) and handover data (solid blue lines) are registered in a public ledger.

Scenario 1 – Public ledger: In this scenario (Figure 12), both handover data and sensor readings, are directly stored on a public ledger. This is a straightforward architecture, inherently guaranteeing immutability, transparency, and trust among chain members. However, as we will see in our evaluation results, the large volume of data to be stored on the public ledger place an enormous burden in terms of cost and increase delay.

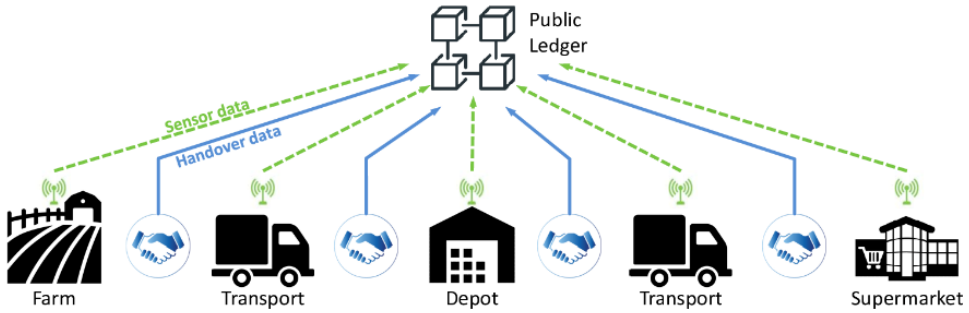

Figure 12: Scenario 1 – Public ledger: All sensor data (dashed green lines) and handover data (solid blue lines) are registered in a public ledger.

Scenario 2 – Single shared ledger: Architecturally, this scenario (Figure 13) is identical to the previous one, other than using a shared private ledger (run collaboratively by all chain members) in place of a public ledger, to avoid high costs and delays. A public ledger, however, is still needed to add strong immutability guarantees to the data stored on the private ledger. More specifically, the private ledger's latest block hash is periodically stored on the public ledger to strengthen the former's immutability, a process referred to as anchoring. Note that the difference between this scenario and our reference Scenario 0 lies on what data is being stored on the public ledger to provide high immutability guarantees. This scenario periodically stores block hashes of the shared ledger, while Scenario 0 stores a separate hash digest per box session; as we investigate below, this affects the public ledger costs.

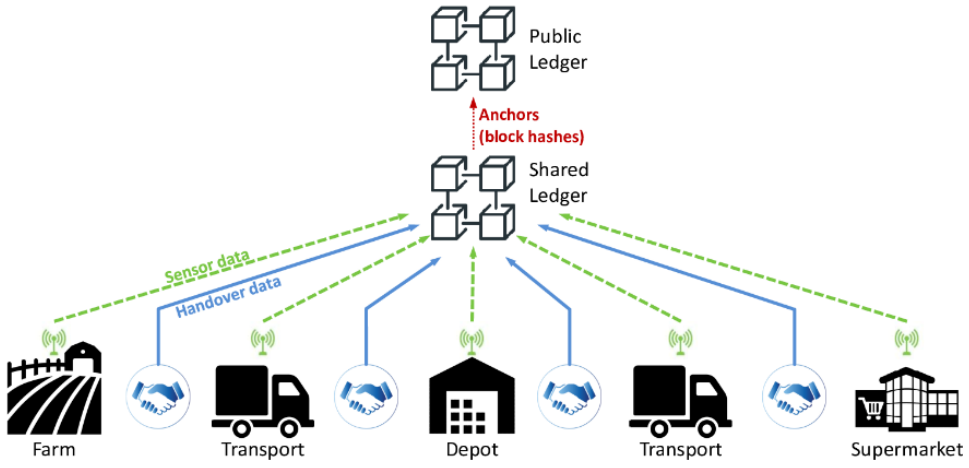

Figure 13: Scenario 2 – Single shared ledger: All sensor and handover data are registered in a shared ledger operated by the entire consortium.

Scenario 3 – One private ledger per pair: This scenario (Figure 14) employs multiple private ledgers, one per pair of adjacent chain stages. It improves on Scenario 2 with respect to data privacy, as well as in overall throughput, as we show in the evaluation results. Anchoring to a public ledger is necessary here too, to guarantee immutability.

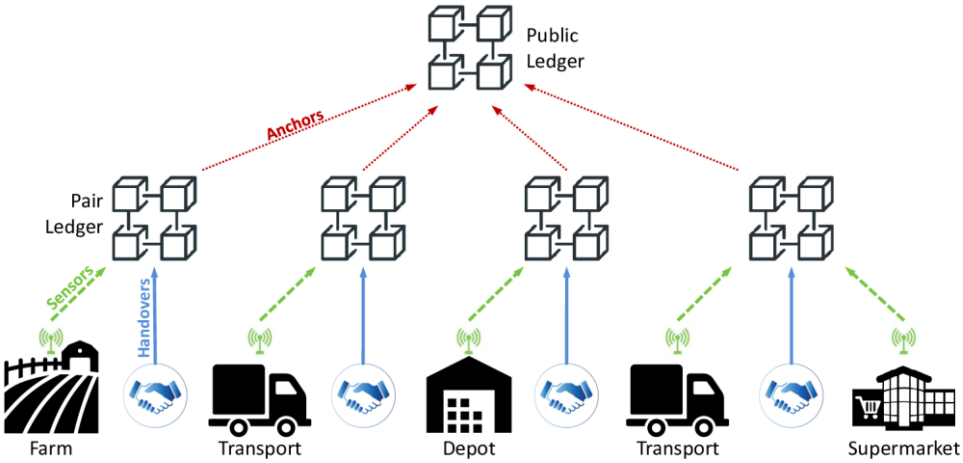

Figure 14: Scenario 3 – One private ledger per pair: Each pair of consecutive stages maintain a separate ledger for recording box handovers between themselves.

Scenario 4 – Private storage: This scenario (Figure 15) maximizes privacy with respect to sensitive data, by having each business entity storing all their data in private storages. These storages need not be ledgers (although they could be). They can be local databases, cloud storage, or even local permissioned ledgers. In the absence of a ledger to store mutually approved handover transactions between adjacent stages, handover records should be signed by both stages involved, and stored individually by both. In order to guarantee immutability of private storages, each stage is responsible to implement anchoring for their private storage by periodically storing a hash digest (Merkle tree root or alternative cryptographic tool of their choice) of their contents.

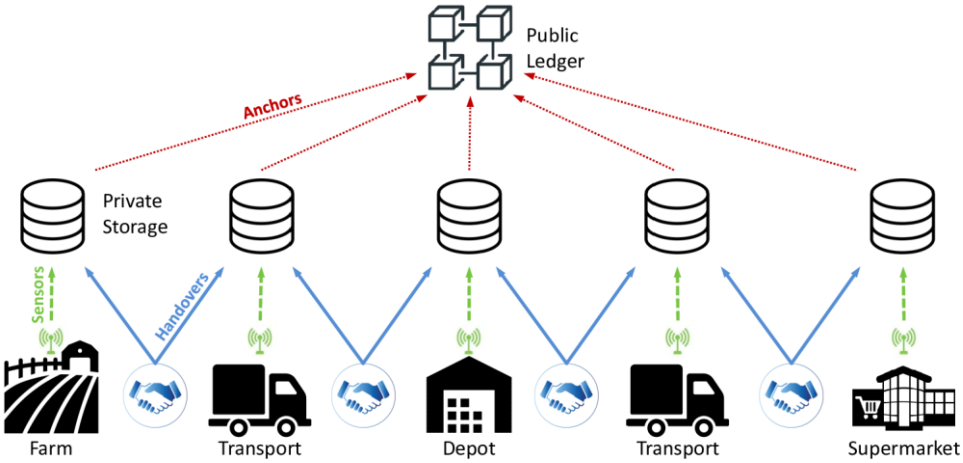

Figure 15: Scenario 4 – Private storage: Each stage maintains their own private storage.

Clearly, an unlimited number of architecture designs can be devised, either as combinations of the above, or by introducing completely new schemes. However, the selected scenarios form a wide range of clearly defined baselines, whose evaluation identifies their relative strengths and weaknesses, and help us make educated decisions in using them as is or in combinations.

4.1.3 Model used in pilot implementation

The pilot implementation assumes a FSC consisting of the same five stages as for the evaluation. However, it adopts a slightly different model. Figure 16 illustrates the model used by the pilot implementation.

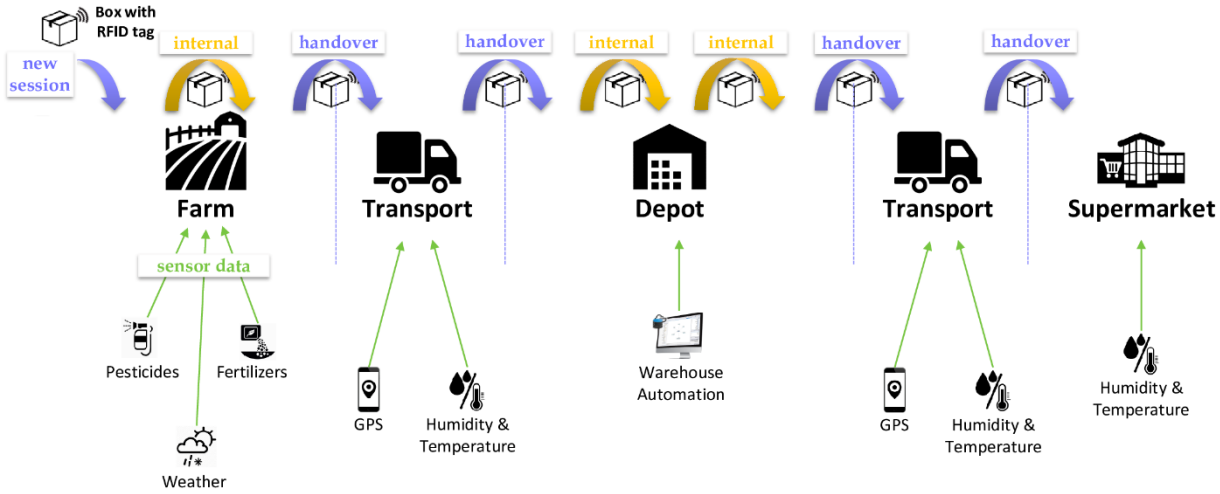

Figure 16: FSC model used in the pilot implementation.

Although the pilot implementation model is largely the same as the evaluation model, it differs in the following ways:

- **Number of handovers:** The most notable difference between this model and the one considered in evaluation lies in the number of transactions across the FSC. The pilot implementation model assumes some *internal state transitions* (marked in yellow), which record box state changes taking place within a FSC stage. One such internal state transition takes place in the first stage (Farm), recording that the producer has filled-in and labelled a previously empty box (but it is not yet handed over to the transporter). Another two internal state transitions take place in the third stage (Depot), to record that the received products have been placed in some storage space, and that the products have been packaged and are ready for further shipping, respectively.
- **Session end:** Another difference between the two models concerns the way a box session is considered complete. The pilot implementation records a box' state up until it reaches the fifth stage (Supermarket). Then, the box' session is considered complete. In contrast, the evaluation model records an additional Exit transaction, marking the end of a box' session.
  Consequently, in the pilot implementation each box and related metadata are recorded in a total of eight transactions throughout the FSC (new session + 3 internal state transitions + 4 handovers), as opposed to six (entry + 4 handovers + exit) in the evaluation.
- **Sensor data storage:** Another important difference between the two models lies in the way sensor data are being recorded. In the evaluation, sensor data from each stage are being recorded *periodically* to the corresponding ledgers as standalone data, i.e., independently of any box-related data (handovers, etc.). Retrieving the sensor readings associated with a given box session entails the retrieval of the respective sensor readings for the specific time periods for which that box was at each stage.
  The pilot implementation follows a different approach. No periodic recording of sensor data takes place. Instead, sensor data concerning a box during its stay at a given stage are embedded in the handover transaction marking its move to the next stage. This

simplifies the retrieval of sensor data; however, it uses up more storage space in the ledgers, assuming that boxes come at a rate higher than the frequency of sensor logging.

- Finally, there are a number of smaller differences in most data structures, such as the lengths of employee IDs, stage IDs, box IDs, session IDs, etc. The evaluation generally resorts to 32-bit IDs, while the pilot implementation opts for 256-bit IDs. Such differences result in different ledger storage costs associated with transactions.

The differences between the pilot implementation and the evaluation reflect their different focus. The pilot implementation aimed at designing and building a single application for a real-world FSC, interfacing with sensors and employees at all stages of the chain and offering a web application providing complete functionality. This entailed including a complete data scheme to represent a number of potential platforms, sensor types, and handover metadata. It also required a complete implementation, including all interfaces to sensor and box tracking systems, hooks for a fully functional web application, and deployment on a number of machines.

In contrast, the evaluation focused on devising a number of alternative potential architectures for a FSC, on implementing them and on providing insights supporting the use of one or another depending on specific performance or cost requirements. This was based on a detailed comparative evaluation of their respective implementations. In order to keep the focus on the effects of different architectures on the performance metrics, the evaluation had to assume a more homogeneous and generic FSC model, considering, for example, that all stages get involved in the same number of handovers and all sensors produce equal-length readings.

Figure 17 depicts the architecture considered by the pilot implementation, as described above. As mentioned in Section 4.1.2, above, the pilot implementation architecture is akin to that of evaluation Scenario 0 (Figure 11), which is used as a reference scenario in our evaluation. Their only architectural difference is the additional use of KSI in the pilot implementation to timestamp the end of each session.

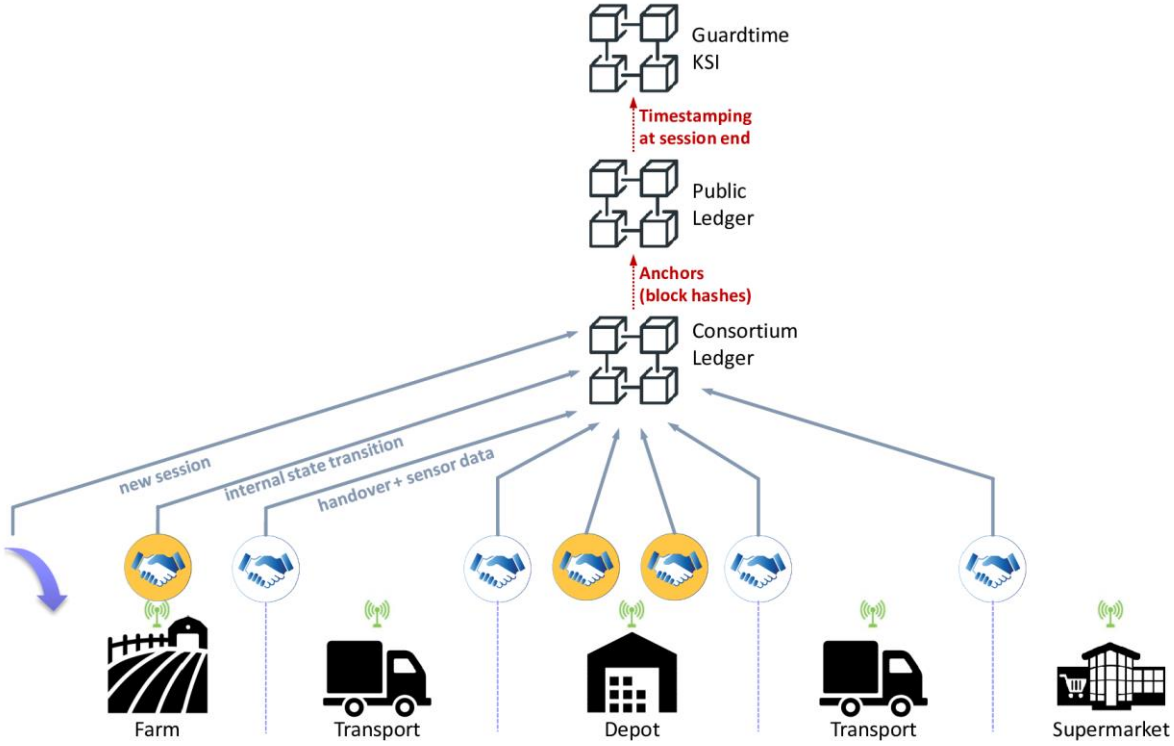

Figure 17: All handover (white) and internal state transition (yellow) data are registered in a shared ledger operated by the entire consortium. A public ledger is used for anchoring, while KSI is used for timestamping session completions.

4.1.4 Joint analysis of emulation and pilot results

In the following sections we present the results retrieved both through the evaluation work (detailed in D4.4, Section 6.1), as well as through the pilot implementation (detailed in D5.4).

All smart contracts, both for evaluation as well as for pilot implementation, were implemented in Solidity, which is the mainstream Ethereum language, and were deployed on Ethereum. More specifically, local instances of Ganache were used as local Ethereum instances, while the Ethereum Ropsten testnet was used as a public ledger.

We made some configuration decisions to reduce the parameter space and to allow for a fair comparison. With respect to the parameters of our local Ethereum instances, we fixed the average block mining time to 15 seconds, and we set the block gas limit to 10,000,000 gas units. Both values reflect the respective values in the public Ethereum main net.

All cost estimations concerning transactions on public ledgers were based on average prices of the public Ethereum main net. More specifically, we have assumed a cost of *10 Gwei* (i.e., 10 nanoether, or 10^{-8} ETH) per gas unit, and a price of €200 per ETH. That is, we have assumed a cost of €2*10^{-6} per gas unit.

Finally, we fixed both the anchoring period and the sensor logging period to 5 minutes, for all stages in all evaluation scenarios.

4.1.4.1 Public ledger cost per box

As explained in D4.4, in the evaluation Scenarios 1, 2, and 3, the total gas cost associated with each box' end-to-end route through the FSC is 340K, 340K, and 360K gas units, respectively. Of these, only Scenario 1's costs translate into actual monetary value (in ETH, and indirectly in EUR), as Scenario 1 stores handovers directly on the public Ethereum. These 340,000 gas units translate into 0.0034 ETH, or about **€0.68 per box**. For a total of 6,000 boxes, which is the typical number of boxes anticipated to be processed through the FSC per day, this corresponds to **€4080 per day**.

Scenarios 2 and 3 store handovers on private Ethereum instances where gas can be paid for in Ethers collected by very lightweight mining or through an ether faucet. Scenario 4 incurs **no gas cost**, as handover and sensor data are recorded on business entities' proprietary storage.

Scenario 0 and the pilot implementation, on the other hand, use a private Ethereum instance, the *consortium ledger*, to store handover data, therefore they do not incur any monetary costs at this level. However, they do record a hash digest of all handovers concerning a box upon its arrival at the end of the FSC. That is, there is one transaction made on the public ledger per box session completion. This transaction incurs a cost of 207,220 gas units, which translates into 0.00207220 ETH or about **€0.42 per box**. For a total of 6,000 boxes per day, this corresponds to **€2486 per day**.

4.1.4.2 Public ledger cost for periodic operations

Scenario 1 of the evaluation performs periodic sensor logging on the public ledger. The associated cost is 250K gas units, or **€0.50 per log**. Assuming a logging period of 5 minutes, this corresponds to **€144 per day**.

Scenarios 2, 3, and 4 of the evaluation do not involve any sensor logging on the public ledger, as they log all sensor readings on private ledgers. However, as explained in D4.4, in order to improve immutability guarantees they need to resort to periodic anchoring of the current private ledger(s) block hash(es) on the public ledger. This incurs a cost of 50K, 200K, and 250K gas units per anchoring, which correspond to **€0.10**, **€0.40**, and **€0.50 per anchor**, respectively. For a 5-minute anchoring period these correspond to **€28**, **€115**, and **€144 per day**, respectively.

Finally, Scenario 0 and the pilot implementation are not subject to any periodic logging or anchoring operations, as all sensor readings and public ledger anchoring are performed on a per-box basis.

4.1.4.3 Handover and internal state transition time

Evaluation Scenarios 0, 2, and 3, as well as the pilot implementation, record handovers on private Ethereum instances, where blocks are generated once per 15 seconds. As the capacity in terms of the number of transactions per block can be adjusted as needed on private instances, no transaction competition is anticipated, therefore transactions are expected to be registered in the next generated block, that is, within the next **15 seconds** since their submission. The same applies for internal state transitions for the pilot implementation only, as such transactions are not applicable for the evaluation.

Evaluation Scenario 1 submits such transactions on the public Ethereum, therefore there is no guarantee on how many blocks it will take to get them endorsed on the blockchain. The delay depends on the gas price offered by the issuer of the transaction: the higher the gas price, the higher the priority by which Ethereum miners will handle the transaction.

Finally, the time to record a handover is not applicable to evaluation Scenario 4, where handovers are stored on proprietary storage of individual business entities, which may very well be commodity databases.

4.1.4.4 Box throughput

Research results presented in D4.4 conclude that evaluation Scenarios 1, 2, and 3 can process end-to-end a maximum of **111**, **133**, and **285 boxes per minute**.

Newer experiments showed that Scenario 0's throughput matches that of Scenario 2, that is, it can process a maximum of **133 boxes per minute**. This comes as no surprise, as for both Scenarios 0 and 2 it is the shared ledger that forms the bottleneck, which stores precisely the same amount of information in both cases.

For evaluation Scenario 4 there is no applicable limit on the respective throughput capability, as handovers are stored on arbitrarily fast proprietary storage systems.

Finally, the pilot implementation can only process **30 boxes per minute**. This is due to the fact that in the pilot implementation handovers and internal state transitions contain a lot more data in comparison to the evaluation scenarios, as they include larger ID fields and they embed all sensor data associated with each box. It is also due to the pilot implementation's longer line of transactions across the FSC, namely involving 8 transactions per box as opposed to 6 for the evaluation.

4.1.4.5 Time scalability

All experiments for Evaluation Scenarios 0, 1, 2, and 3 indicate that the time required to register *h* handovers is **linearly proportional** to *h*. More specifically, if a block can fit *k* handovers, registering *h* handovers will require $\lceil h/k \rceil$ blocks; it will, therefore, take $\lceil h/k \rceil \cdot 15$ seconds to complete, on average.

The same applies for the pilot implementation, that is, its time complexity is **linear** with respect to the number of handovers being processed.

Evaluation Scenario 4 constitutes the only exception, as the time to register *h* handovers depends on the individual entities' proprietary storage of arbitrary throughput.

4.1.4.6 Cost scalability

With respect to transaction costs on the public ledger, evaluation Scenarios 0 and 1, as well as the pilot implementation, incur a constant cost per box, as they both register a fixed amount of

data per box on the public ledger. Therefore, they both exhibit a **linear cost function** with respect to the number of processed boxes.

In contrast, evaluation Scenarios 2, 3, and 4 do not store any box-specific data on the public ledger. All handover and sensor data are stored on private ledgers only, while the public ledger is used exclusively for periodic sensor logging and anchoring, which are independent of the number of boxes being processed. Therefore, these scenarios exhibit a **constant cost function** with respect to the number of processed boxes.

4.1.4.7 Response time for audit requests

Auditing constitutes a vital functionality of the FSC pilot. It allows an auditing authority to collect all data related to a box in case its contents reach the customer at a state that does not comply with the required quality standards.

In all Evaluation Scenarios as well as in the Pilot Implementation, audits involve the execution of exclusively read-only smart contract functions on the respective ledger(s). As such, the time to complete these calls depends only on the local processing power and I/O bandwidth of the Ethereum node executing them, which is expected to be negligible in comparison to typical ledger actions such as registering a new transaction.

4.1.4.8 Joint analysis KPIs

In Table 17 we summarize the joint-analysis results obtained for evaluation Scenario 0 (our reference scenario) and for the pilot implementation, organized by the defined performance KPIs and presented side-by-side to allow for comparison. We note that, as expected, the two scenarios demonstrate similar performance, with the sole difference being related to their throughput, due to the different number of transactions and data stored per transaction for the two cases.

Table 17: Comparison of the Food Supply Chain KPIs for evaluation Scenario 0 and pilot implementation.

KPI	Description	Evaluation Scenario 0	Pilot Implementation
KPI_FSC_1: Public ledger execution cost	Total cost on public ledger	€0.42 per box, €2486 per day	€0.42 per box, €2486 per day
KPI_FSC_2: Handover time	Time to register a handover	≤ 15 sec	≤ 15 sec
KPI_FSC_3: Internal state transition time	Time to register an internal st. tr.	N/A	≤ 15 sec
KPI_FSC_4: Throughput	Boxes processed per time unit	133 box/min	30 box/min
KPI_FSC_5: Time scalability	Handover time vs #boxes	linear	linear
KPI_FSC_6: Cost scalability	Public ledger cost vs #boxes	linear	linear
KPI_FSC_7: Audit requests response time	Time to execute an audit request	negligible (local action)	negligible (local action)

Finally, Table 18 summarizes the results across all evaluation scenarios. The results of Scenario 0, presented in the previous table, are repeated here for convenience.

Table 18: Comparison of all Evaluation Scenarios, regarding the Food Supply Chain KPIs.

KPI	Description	Evaluation Scenario 0	Evaluation Scenario 1	Evaluation Scenario 2	Evaluation Scenario 3	Evaluation Scenario 4
KPI_FSC_1: Public ledger execution cost	Total cost on public ledger	€0.42 per box, €2486 per day	€0.68 per box, €4224 per day	N/A per box, €28 per day	N/A per box, €115 per day	N/A per box, €144 per day
KPI_FSC_2: Handover time	Time to register a handover	≤ 15 sec	unbounded, but typically ≤ 15 sec	≤ 15 sec	≤ 15 sec	negligible
KPI_FSC_3: Internal state transition time	Time to register an internal st. tr.	N/A	N/A	N/A	N/A	N/A
KPI_FSC_4: Throughput	Boxes processed per time unit	133 box/min	111 box/min	133 box/min	285 box/min	unlimited
KPI_FSC_5: Time scalability	Handover time vs #boxes	linear	linear	linear	linear	N/A
KPI_FSC_6: Cost scalability	Public ledger cost vs #boxes	linear	linear	constant	constant	constant
KPI_FSC_7: Audit requests response time	Time to execute an audit request	negligible (local action)	negligible (local action)	negligible (local action)	negligible (local action)	negligible (local action)

4.1.5 Privacy trade-offs

In our modelling both for the evaluation as well as for the pilot implementation, we have so far assumed that the ledgers store *actual raw data*, including handover metadata and sensor readings. This provides transparency for the FSC pilot, as all relevant data will be readily available to auditors should a problem arise.

In certain cases, however, this may be undesirable due to privacy concerns. As the ledgers assumed in our scenarios are either public or at least accessible by several actors, data written on them is essentially not private anymore. Certain business entities might not want their competitors to have access to the volume of products they handle, the companies they are partnering with, or the specific sensor readings coming from their infrastructure.

In cases where storing raw data in public ledgers violates the business entities' privacy rules, two main alternatives may be proposed. The first alternative is to store just *hashes* of data on the ledger, while keeping the actual data on private local storage. Upon request from a certified auditor, the business entity would be obliged to disclose any data requested. Subsequently, the auditor would use the hashes stored on the ledger to verify the integrity and authenticity of the data. The main advantage of this alternative is that it respects the business entities' privacy. The main disadvantage, though, is that auditing cannot be a local operation anymore, as it requires the cooperation of the entity being checked. Furthermore, it may raise an availability issue, in case of accidental or intentional raw data loss.

The second alternative seeks to address the availability issue inherent in the aforementioned option. In this proposal, entities have to store all *actual data* on the ledger, however in *encrypted form*. This defeats the availability risk, to the degree that a private ledger is operated by a

number of distinct parties, without exposing any entity's sensitive trade data to its competitors. In the face of an audit, however, the business entity would be obliged to disclose the respective keys to a certified auditor. The encryption scheme should be as granular as possible, allowing a business entity to grant access to precisely the specific data requested and no more. To that end, a distinct key should be associated with each data item encrypted on the ledger, and it should be the business entity's responsibility to maintain all keys safe and to make them available to certified auditors on demand. Although the aforementioned availability risk may still seem to be a threat (e.g., in the case of an entity losing or hiding their series of keys), this can be easily addressed by demanding the use of a standardized key generation scheme, such as Hierarchical Deterministic Wallets (HD Wallets), which provide an infinite series of keys based on a single secret seed.

4.1.6 Sensor logging and anchoring trade-offs

Two periodic operations involved in our evaluation scenarios, namely the periodic logging of sensors and periodic anchoring, present two new trade-offs regarding the respective frequencies.

Sensor logging rate should be high enough to reliably record a representative state of the environment. For instance, it might make little practical difference if a refrigerator's temperature is being recorded every 1, 2, or 5 minutes; however, if it is recorded only once a day, the quality assurance these measurements are supposed to provide is rather questionable. On the other hand, pushing sensor logging frequency higher than necessary, only increases the cost of ledger storage without providing any practical benefits.

An alternative policy that aims at being the best of both worlds is to apply *adaptive* sensor logging. In this policy, a sensor's value is not recorded on the ledger unless its difference to the previously recorded value has exceeded a predefined threshold. Furthermore, the environmental sensors' typically smooth value deviation can be leveraged to compress data, for instance by recording delta values rather than actual values.

The second periodic operation concerns anchoring. By anchoring we mean copying a private ledger's block hash on a public ledger, to increase the former's immutability guarantees. The intuition is that, should an entity controlling the private ledger decide to "rewrite history", their fake blocks' hashes will not match the respective hashes already stored on the public ledger (which is assumed to be immune to history rewriting due to its humongous user base and block generation difficulty).

Anchoring too rarely can prove problematic. The time between two consecutive anchored blocks, that is, the anchoring period, effectively constitutes a potential attack window for entities with significant mining power on the private ledger. On the other hand, anchoring too frequently (e.g., anchoring every single block of a private ledger that generates blocks every 15 seconds) can use up too much storage on the public ledger, resulting in unnecessarily high anchoring costs. Finding the right balance regarding the anchoring period should be done by taking into consideration the potential attacks and motivations for attacks, as well as the risk of such an attack.

Related to the above discussion regarding the anchoring frequency is the analysis in deliverable D 4.4 that assessed the tradeoff between the recording frequency of hashes of data from the decentralized data energy exchange pilot and the opportunity cost from the ability to modify the data from the time the last hash was recorded on the public chain until the time the next hash will be recorded.

4.2 Decentralised Energy Flexibility Marketplace

The Decentralized Energy Flexibility Marketplace (DEFM) pilot aims to balance the load on a real energy network, namely the distribution grid of the city of Terni, located in central Italy, by charging Electric Vehicles (EVs).

In the following, we present the results of the second phase of the evaluation of pilot-inspired emulation scenarios, which explores the large-scale deployment of the service.

4.2.1 Overview

In this evaluation, we shed light upon the performance of the DEFM service in a realistic large-scale setup. We consider a large number of EVs and Charging Stations (CSs) as well as α realistic volume of Reverse Power Flow (RPF) caused by the green-energy production units. In particular, we exploit real traces that were produced by the actual pilot implementation and deployment. We discuss the exploitation of the traces in greater depth later in this section.

The explored scenario focuses on the typical energy flexibility campaign where the DSO registers an offer in the energy flexibility marketplace and the Fleet Managers (FMs) submit their bids accordingly, to participate in the DSO's auction (Demand Response - DR - campaign) with the aim of obtaining the economic bonus for providing flexibility, as discussed in deliverable D5.2. The scenario does not involve independent EV users in the trading process, hence all considered EVs belong to one fleet. Such an assumption explores the service in a more "controlled" setup, where the EVs in the experiment are deterministically directed by the related FM, e.g., assuming fleets of delivery service or a courier service or large repair companies. This case is particularly interesting, since it is the first to introduce a hierarchical energy trading paradigm based on the blockchain, it simplifies the scheduling of the EVs and is expected to enhance its efficiency; we assume that the EVs are required to follow the directions of the FMs, which avoids the need to consider the possibility that an EV (user) denies the scheduling directive. In case the forecasted fleet size of the "winner" FM that placed the best offer is not large enough to consume the required energy, the scenario supports the declaration of multiple "winner FMs" in an effort to enhance energy consumption. In the latter case, the bids of the FMs are treated as complementary/additive bids, hence a single energy campaign offer can have multiple winners.

Regarding EV scheduling, there are two popular scheduling approaches: the *day ahead* scheduling and the *daily valley* scheduling. In the first approach, the DSO forecasts the excessive amount of energy production for the next day and creates incentives for EV users to augment their energy reduction while, in the second approach, the EVs are scheduled in order to shave energy production peeks that appear at certain timeslots within a day. Our investigation considers the first scheduling approach with the DSO creating a campaign where the FMs participate in order to enhance energy consumption but is also compatible with the second approach since the EVs should be scheduled in order to consume a certain amount of energy within the 24-hour deadline. Compared to traditional EV scheduling policies, our scenario poses a simpler problem for two reasons: first the energy consumption requirement must be satisfied instead of optimized and, second, the FM, who is omniscient about the CSs and its fleet (FMs knows where, when and for how long the EVs are idle), can direct each EV deterministically.

4.2.2 Emulation setup

The implementation of the service consists of two parts: a blockchain-based part and an IoT-based part. In the blockchain-based part, we consider the smart contracts and the blockchain node that implement the energy flexibility marketplace; we exploit the prototype implementation of the SOFIE marketplace smart contracts and a Ganache Ethereum node to evaluate this part (grey modules in Figure 18). Focusing on the networking aspect of the service, the IoT-based part includes the actors of the service (or the actors' devices) that interact with the blockchain-

based part in order to add and get offers, bids and charging events; we emulate these actors through a web3 script that orchestrates the calls to the smart contract functions, thus emulating the participation of multiple actors, such as a DSO, many FMs with numerous EVs and numerous CSs (green module in Figure 18).

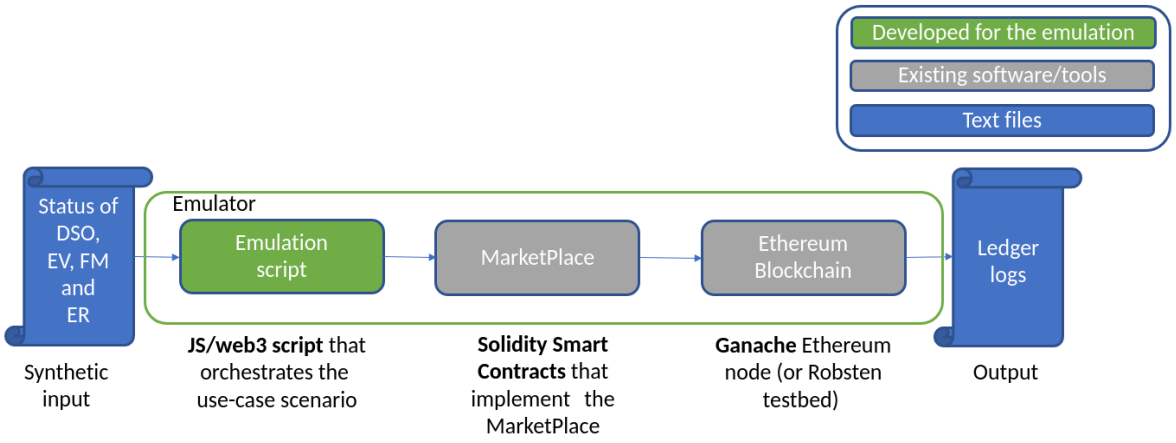

Figure 18: DEFM pilot emulation setup.

Figure 18 depicts an overview of the emulation setup. The setup consists of three core parts: the input, the emulator, and the output; these are discussed in detail below.

4.2.2.1 Input

The input constitutes the experiment variables that are "fed" to the emulator in order to build specific use-case scenarios. Typically, the input includes:

- DSO status: specifies an energy flexibility campaign, it indicates the volume of energy, the location (e.g., list of CSs), the temporal constraints and the maximum number of tokens to be used for providing incentives
  - Energy forecast: we exploit the traces of daily RPF provided by the real Pilot implementation. Given that the daily RPF forecast model presents many performance spikes, in each emulation run, the DSO campaign contains the mean RPF value of 200 random daily measurements.
  - Tokens: a relationship between tokens and energy units, e.g., 0.002 Euro/watt
    - The specific value can be derived by the overall energy consumption of the fleet.
  - District map: Location of CSs and EVs
    - We exploit the district map shown in Figure 19: Emulation district provided as Input., which indicates the position of 3CSs and roughly 200 EVs.
- FMs status: list of EVs that belong to the FM's fleet
  - EV fleet size: the number of EVs that each FM directs
    - We experiment with different Fleet sizes from 25 to 75 EVs per FM.
    - We randomly assign the EVs of the district map to FM(s).
- EV status: location, autonomy, energy until full charge,
  - Location: we create three EVs according to the district map
  - We exploit the real traces of the 6 cars from the Pilot, which periodically record the state of each EV, resulting in more than 150.000 real EV status records.

- The status records indicate the autonomy (Km), the battery state of charge (%) and the battery capacity (KWatt h) among others.
- For each of the EVs of the district map we randomly assign a state from these traces.

- CS status: the location, charging rate, number of charging slots
  - Location: we create three CSs according to the district map
  - Charging rate: in order to estimate the average charging rate of a CS, we exploit real traces produced by the Pilot deployment which record roughly 900 charging sessions
  - Slots: we experiment with different slots sizes from 1 to 50 slots per CS

For the graphic illustration of the input, we refer to the following figure which depicts an overview of an emulated district. The input indicates the geographic distribution of the CSs and the EVs at a certain time slot. The CS placement is fixed, while the EVs placement indicates their location when they are idle (if EVs are idle more than one timeslot in the next 24hours, the longest timeslot is considered), hence we can assume that this is the map of the district in the next 24 hours.

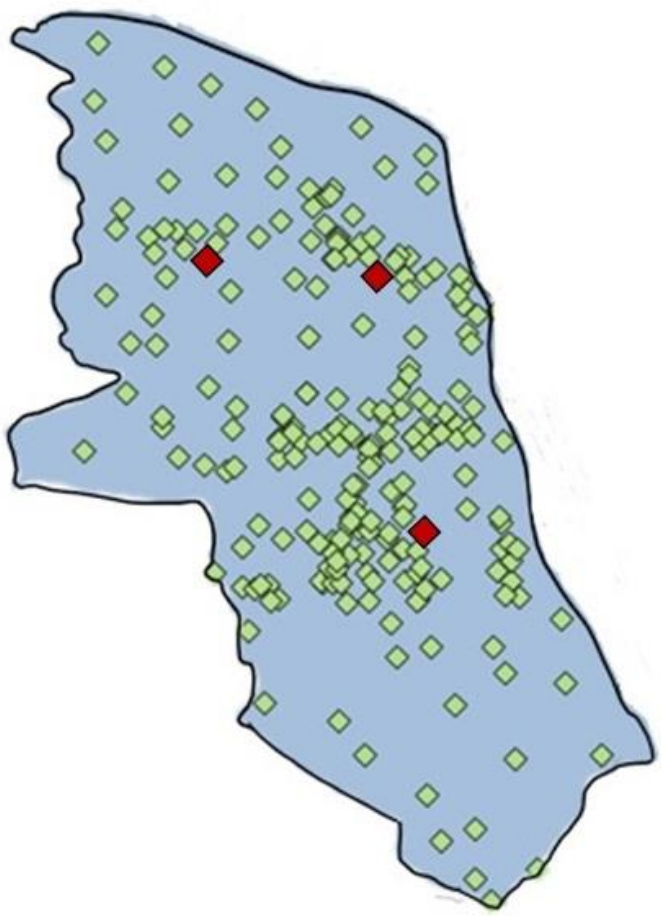

Figure 19: Emulation district provided as Input.

4.2.2.2 Emulator

The emulator consists of the web3 script, the smart contracts and the Ethereum node. We focus on the web3 script which is the only emulation-specific part; the other two being an extended

implementation of the SOFIE's marketplace that supports multiple campaign winners and the Ganache Ethereum node.

The web3 script provides the logic of the emulation; put succinctly, it parses the input and makes all the appropriate calls to the marketplace, thus emulating the roles of the DSO, the FMs and the EVs visiting the CSs. In addition, the script undertakes the important task of generating the EV charging schedule for each FM before the latter places its bid.

The message diagram that describes the interaction of the script and the marketplace is depicted in the following figure.

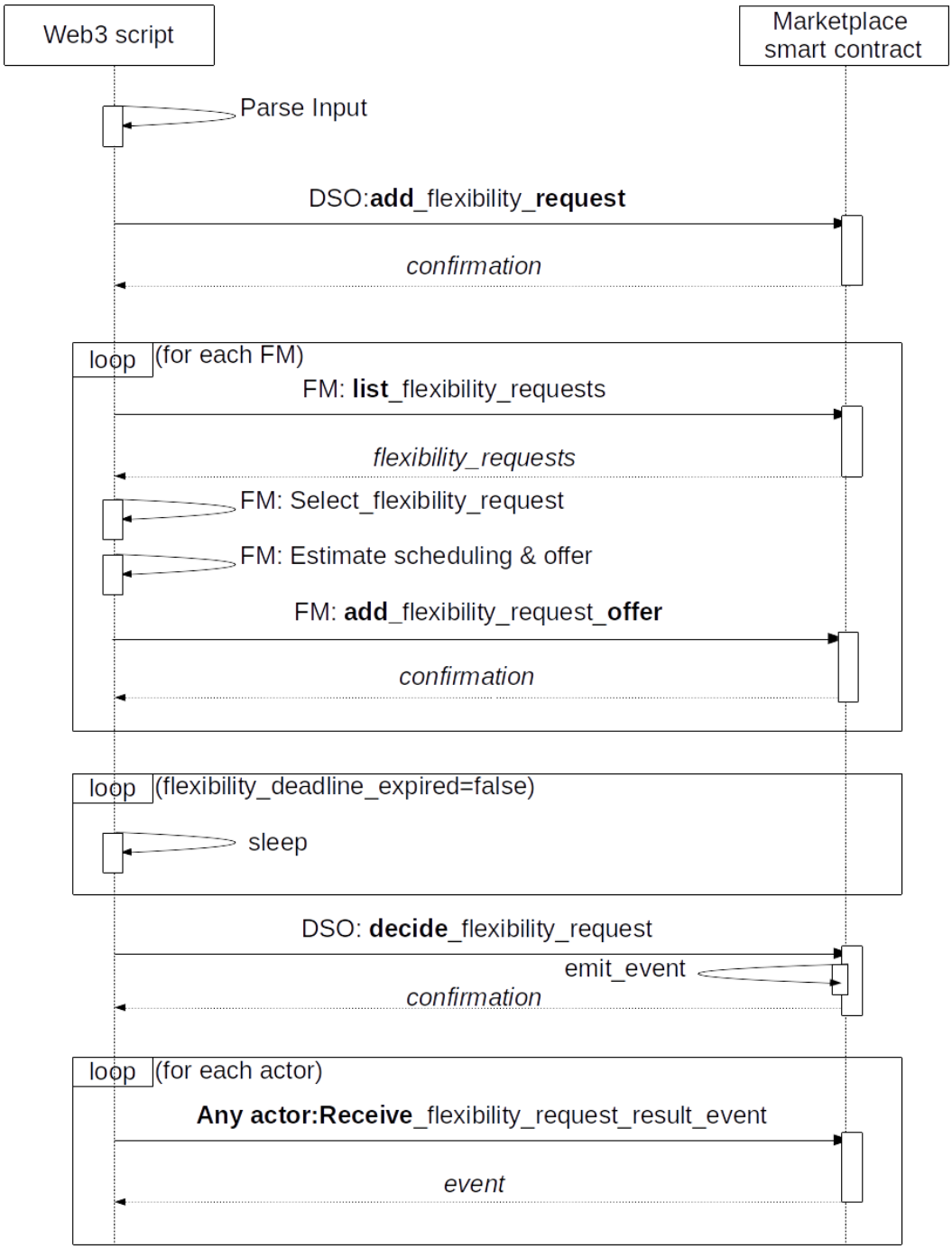

Figure 20: Emulation script and marketplace message diagram.

Initially, the script parses the input files that describe the status of the actors of the scenario. Then, the flexibility campaign (also referred to as 'request' or 'flexibility request') is added to the marketplace, emulating the operation of the DSO. The DSO message has the following structure, which is depicted as a collection of different attributes:

{requestId{R} energy{E}, tokens{T}, District{D}, bid_timer{bt}, campaign_timer{ct}}

The semantic representation of the attributes follows the pattern *type{value}*, hence the previous example symbolizes a flexibility request with "R" unique request id, "E" energy units, "T" number of tokens, "D" unique district id, and "bt" and "ct" deadline for accepting bids and consuming the energy, respectively.

Then, the script emulates the operation of the FMs. For each FM indicated in the input it executes three functions: first, it lists the available campaigns in the marketplace (in our case, there is only one offer); second, it generates the EV charging schedule that addresses the requirements of the campaign; and third, it adds the bid (also referred to as offer) to the marketplace. If the generated schedule does not meet the requirements no bid is added. The bid indicates the ID of the flexibility request that it concerns, the unique ID of the offer, the district that it concerns and the "charging schedule"; the latter part is a complex primitive, hence we discuss it in depth in Section 4.2.3. The FM bid message has the following structure:

{request{R}, offer{O}, District{D}, sched{..}}

Notice that the energy to be consumed is derived by the EV schedule (attribute "sched{..}").

Having finished the bidding process, the script (optionally) waits until the bidding timer expires and then emulates the operation of the DSO by requesting from the marketplace to estimate (also referred to as decide) the best bid (or the best combination of bids, if such a policy is required). Any involved actor can be updated about the campaign outcome through correlated blockchain events or can call the smart contract function that lists the decision of the request/campaign. The structure of the event or message is the following:

{offer{0}, Bid1{tokens{T1}, sched{..}}}, .., BidN{tokens{TN}, sched{}}}}

4.2.2.3 Output

The output consists of two plain text log-files that include the logs of the Blockchain node and the logs of the web3 script. Through these logs we can infer the following metrics:

- Blockchain performance: mean cost and latency of executing each smart contract function (similarly, the overall service cost can be estimated)
- Service performance: % energy that is consumed, #FMs (and EV users) participating in the campaign, #FMs (and EV users) that won the campaign, #EVs that were charged

4.2.3 Generating bids

The generation of bids is a complex problem with no "one design fits all" solution. Some attributes of the bids, such as the consumed amount of energy and involved CSs, are correlated to the scheduling of the EVs, which we discuss in the following section. Here, we discuss the method to estimate the consumed amount of energy and we elaborate on the price, in terms of tokens, that the FM receives in order to consume the energy of the flexibility campaign.

After receiving the flexibility request, the FM starts the generation of the bid. The most important part of a bid is the amount of energy that the FM's fleet can consume. We estimate this amount by summing the "charging dynamic" metric of all the EVs, which is a CS-specific metric that

consists of two values: the energy that the vehicle needs until it is fully charged and the energy that the vehicle will consume until it reaches a particular CS.

Regarding the tokens, in the baseline scenario that we explore in our evaluation, we assume a linear relation between the tokens (T) and energy (E), T = *x* E, where the coefficient x is defined by the offer. For instance, if the DSO offers 100 tokens for consuming 10 units of energy, then the coefficient *x* is equal to 10 and a bid for consuming 5 units of energy should be paid 50 tokens.

In more sophisticated scenarios this relation may not be linear. For instance, in order to boost energy consumption, the DSO can increase the number of tokens faster than the increase of energy consumption, thus rendering higher energy bids more financially advantageous. The opposite strategy can be used to boost the engagement of smaller clients, such as autonomous EV users, by making smaller bids cheaper. Overall, we consider this an interesting topic for future research.

4.2.4 Deciding bids – Multiple winner selection

We have considered the support of multiple winner FMs in order to boost energy consumption. Apparently, the issue of conflicting scheduling arises in this case, since multiple FMs may try to exploit a CS simultaneously. We suggest that the conflict is resolved by the "decide" smart contract function which elects the winners. In particular, if the second-best bid leads to overlapping scheduling, the smart contract can reject it or partially accept it, e.g., accept only the non-overlapping scheduled timeslots. As expected, the more fine-grained the solution (resolving conflicts at timeslot level), the greater the Blockchain-related cost to submit the required data and estimate the conflicts. In contrast, the more coarse-grained the solution (rejecting overlapping bids), the less efficient the algorithm will be. Trying to combine the best of both worlds, we propose the following scheme.

In addition to the required tokens, the FMs include in their bids the list of CSs, that are included in the charging schedule, and the amount of energy to be consumed at each CS. In order to reduce the overhead of indicating all CSs, the bid can include the CS cluster, a simplification that is realistic since CSs are usually clustered, e.g., parking lots. Thereafter, the smart contract can prioritize the exploitation of a conflicting schedule by granting the use of the CS to the bid that plans to consume the most energy.

Consider the following example where a DSO sends an offer (DSO1) to pay up to 200 tokens the FMs that will consume cumulatively 20 units of energy through three charging station clusters (CS1-3) of a district (D1) that each can offer up to 7 units of energy in the next 24hours. Hence:

DSO1: {request{DSO1}, tokens{200}, energy{20}, District{D1}}

CS1-2:{energy{7}}

Then, we assume that the marketplace receives the following three FM bids (Bid1-3):

Bid1: {request{DSO1}, offer{bid1}, District{D1}, sched{5@CS1 + 3@CS2}}

Bid2: {request{DSO1}, offer{bid2}, District{D1}, sched{5@CS2 + 5@CS3}}

Bid3: {request{DSO1}, offer{bid3}, District{D1}, sched{3@CS2 + 3@CS3}}

Where "X@CSY" means that X energy units will be consumed at CS with id Y.

The metric that determines the best bid tries to maximize energy consumption hence, the smart contract "decides" (as winners) the best bid per charging station. If multiple bids consume the

most energy at a given CS, then the FIFO policy is considered in order to declare a winner. Thereupon, in our example, the decision of the campaign DCD will be:

DCD: {request(DSO1), sched{(CS1, bid1),(CS2,bid2),(CS3,bid2)}}

Notice that the tokens that each bid will earn are not specified in the decision, however we assume that the overall tokens of the request will be shared among the winner-bids proportionally to the amount of energy they are granted by the decision.

Although we acknowledge that our method is not optimal, we consider it acceptable for this evaluation. Given the processing cost of Blockchain, a suboptimal but cheap solution can be preferable to an expensive optimal one.

4.2.5 Scheduling EVs

Scheduling of EVs at different CSs at different times is a rather complex process, which is usually solved through heuristics and greedy algorithms. In this work we develop our own greedy algorithm which, although not being optimal, fits best to the smart contract operation and the input traces that we base the evaluation upon.

The FM schedules its fleet to the available slots meeting four constraints. First, the CS is reachable by the EV given the latter's autonomy; second, the EV is not fully charged; third, the FM's schedule at each CS does not surpass the daily production of energy of the CS; fourth, the FM's schedule at all CSs does not surpass the amount of energy of the campaign. Regarding the first constraint, the distance between a CS and an EV is given by the Euclidean distance of the two elements in the district map.

At the same time, the scheduling policy tries to maximize the number of EVs scheduled at one CS in order to assist the decision process, that we discussed in the previous subsection. The scheduling process follows 3 steps. At step 1, the FM estimates the "reachability map" of CSs by linking EVs with CSs that are within reach. At step 2, based on the reachability map, it estimates the amount of energy that the fleet could consume if only one CS was exploited, and sorts the CSs according to that metric. The final step is a recursive process, where the CS with the highest amount of energy consumption is selected, the EVs scheduled at that CS are removed from the reachability maps of the other CSs and step 2 is executed again, until there are no more CSs or EVs to schedule.

4.2.6 Emulation results

In order to assess the KPIs, we deployed the smart contracts that implement the emulated marketplace in a local Ethereum node, which is set up through the Ganache software tool (Linux 64-bit, 4 cores @ 3.40 GHz, 4GB RAM). We exploit a local Ethereum node since the emulation scenario requires numerous user accounts, thus making it difficult to work a public ledger.

4.2.6.1 Blockchain performance

We first assess the performance of the smart contracts by exploring the "response time", the "execution cost" related with calling each function and the average "throughput".

We measure response time by subtracting the time that the smart contract function was called from the time the response of the function is received. We measure execution cost through the Ganache logging system, that provides detailed logs about the transactions. First, we measure the execution cost of the smart contract functions that write to the blockchain. The functions are invoked through a local script that exploits the Ethereum JavaScript API web3.js. In the list below, we present the results per function. The careful reader may notice that we also present the response latency of each function (in the parenthesis); we configured Ganache to auto-mine transactions (only for this experiment) in order to tentatively examine the processing overhead per request.

- smart contract deployment: 3.232k gas
- add_flexibility_request: 256k gas (100ms)
- add_flexibility_offer: 285k (200ms)
  decide_flexibility_request_multiple: 1714k (900ms)

The results are similar to the results of the evaluation presented in D4.4, which the exception of the cost of the decision function. In this evaluation session, we experiment with a significantly more complicated method to decide multiple winner-FMs, thus resulting in a significantly larger execution cost and latency. Although we do not consider the cost of the function critical, we propose an alternative design where the decision process is performed off-band by the DSO, who then submits the result to the blockchain. Given that all the data (request and offers) are publicly available through the Blockchain, any actor can simply verify the operation of DSO, thus offering sufficient level of security.

Finally, regarding throughput, the extensive cost of the decision function reduces the number of transactions that can be mined to 4 (every 15s), however the mining latency remains linear.

4.2.6.2 Service performance

We now investigate the effectiveness of the service in large-scale scenarios, focusing on the following two metrics:

- The reduction reverse power flow reduction (%): the ratio of the cumulative amount of energy of the decision to the amount of energy of the flexibility request
- The scheduled EVs (number and %): The cumulative number EVs and the ratio of the cumulative number of EVs that are scheduled to the cumulative number of EVs in the FMs' fleets.

In our *basic* setup, we assume that 3 FMs (with 50 randomly selected EVs each) participate in the campaign, trying to consume the amount of energy indicated by the DSO, which is the average of 200 random daily RPF measurements, through the 3 CS (with 5 slots each) that are depicted in Figure 19: Emulation district provided as Input.. The status of each EV is based on the Pilot traces and the location of EVs at the map of Figure 19: Emulation district provided as Input.. In the following, we present the average values of the performance metrics since each experiment setup was executed 10 times.

Throughout the evaluation we explore the impact of the following parameters on the performance of the service:

- Fleet size: we expect that larger fleet sizes will enhance energy consumption (as far as the CSs are not saturated)
- Number of slots: we expect that a larger number of charging slots at CSs will allow the consumption of larger amounts of energy
- District size: we expect that larger districts (with the same number of CSs) will present less power consumption since the distance between EVs and CSs increases proportionally, thus disallowing the scheduling of EVs with lower autonomy
- Number of FMs: we expect that more FMs will be able to consume a larger amount of energy, since they collectively present a larger fleet.
- RPF volume: we explore the amount of RPF that the service can consume given the basic setup (and the real traces).

Table 19: Service performance for different fleet sizes.

	25 EVs per FM	50 EVs per FM	75 EVs per FM
Scheduled EVs (%)	13 (37.4)	65 (43.7)	101 (45.2)
RPF reduction %	7	15.5	22.1

In Table 19 we present the result of the experiments with different fleet sizes. We first observe that the percentage of scheduled EVs is not affected by the fleet size and is significantly low (roughly 40%). The traces report that roughly 30% of EVs (status records) have fully charged batteries, thus reducing the exploited EVs metric; the remaining 30% is reduced during conflict resolution in the decision process as we discuss later (Table 22). The fact that the traces are randomly assigned to the EVs (following the uniform distribution) explains why this metric is not affected by the fleet size. Next, we observe that the RPF reduction is proportional to the fleet size, which validates that the effectiveness of the service depends on the large number of EVs. Unfortunately, we could not investigate the fleet size that maximizes reduction, since the district map indicates roughly 200 EV positions, hence 75 EVs for 3 FMs is our emulation limit.

Table 20: Service performance for different District sizes.

	10 x District size	**1 x District size**	**0.5 X District size**	**0.1 x District size**
Scheduled EVs (%)	28 (18.2)	65 (43.7)	53 (35)	52 (34.2)
RPF reduction %	9.3	15.5	13.4	12.8

In Table 20 we present the result of the experiments with different district sizes by scaling the map. We validate that the percentage of scheduled EVs is affected by the district size since larger maps reduce the number of scheduled EVs and, in turn, the RPF reduction. In larger maps, the Euclidean distance between EVs and CSs is less likely to be covered by the EVs' autonomy, hence the scheduling is significantly less rich. We also discover that the gains are slightly penalized when shrinking the district. This occurs because all EVs can reach all CSs in the map, hence the FMs submit similar scheduling plans which, in turn, increase the possibility of scheduling conflicts.

Table 21: Service performance for different numbers of slots per CS.

	1 slot	**5 slots**	**50 slots**
Scheduled EVs (%)	38 (25.1)	65 (43.7)	60 (40)
RPF reduction %	9.8	15.5	14.4

In Table 21 we present the result of the experiments with different number of charging slots per CSs. We validate that the percentage of scheduled EVs is affected by the number of charging slots per CSs since fewer slots (1 slot) reduce the number of exploited EVs and, in turn, the RPF reduction. On the other hand, we find no apparent gains in growing the number of slots beyond 5, since the performance of the service is not increased when using 50 slots per CS.

Table 22: Service performance for different numbers of FMs.

	1 FM	**2 FMs**	**3 FMs**	**4 FMs**
Scheduled EVs (%)	31 (62.4)	47 (47)	66 (43.3)	71 (35.1)
RPF reduction %	7.1	12.3	15.5	6.6

In Table 22 we present the result of the experiments with different number of FMs operating in the district. The results show the lack of optimality of our simple scheduling process which fails to schedule a significant amount of EVs as the number of FMs (and the corresponding bids) increase. Namely, EV exploitation is decreased by roughly 20% when considering two winner-FMs in an effort to resolve scheduling conflicts. Nevertheless, the RPF reduction, which is our primary goal, is successfully addressed since the reduction increases as more FMs get involved and the cumulative fleet size increases. Notice, that more than 3 FMs are not handled efficiently by our algorithm which decides one bid per CS, hence one FM (and its fleet) will be always left out in this setup which includes 3 CSs.

Table 23: Service performance for different RPF volumes.

	1 x RPF	**0.5 x RPF**	**0.2 x RPF**	**0.1 x RPF**
Scheduled EVs (%)	66 (43.3)	64 (42.4)	65 (43.1)	58 (38.5)
RPF reduction %	15,5	31.1	74.4	147.7

In Table 23 we present the result of the experiments with different RPF volumes. The results show no linkage between RPF and scheduled EVs (in the range of investigation), but also reveal a linear relation between RPF and RPF reduction. The results imply that, in our experiments, the energy consumption is not limited by the RPF. An interesting observation is that consumption can overcome the RPF indicated in the flexibility request although each FM bid meets this requirement. The occurs because the decision function of the smart contract does not check this condition and can exceed the limit by combining the (constrained) bids. However, this is a feature of our implementation and could be edited easily.

Table 24: Service performance for 200 EVs and different number of FMs.

	1 FM (200EV per fleet)	**2 FMs (100EVs per fleet)**	**3 FMs (66 EVs per fleet)**	**4 FMs (50 EVs per fleet)**
Scheduled EVs (%)	124 (61.5)	104 (51.5)	82 (40.5)	65 (32.3)
RPF reduction %	28.6	23.2	17.1	16.5

Finally, in Table 24 we present the results of the experiments with 200 EVs cumulative fleet size and different numbers of FMs (the fleet size of the FMs reduces as the number of FMs increases). The results reveal that the increasing number of FMs with fixed cumulative fleet size can be detrimental to the performance of the service, because more bids lead to more scheduling conflicts. The most effective approach to enhance the performance is to increase the number of charging EVs, if the fleet sizes can be aggregated then exploiting more FMs is useful, otherwise is can penalize the performance of the service.

4.2.7 Joint analysis of emulation and pilot results

In the following table we compare the evaluation results of the Pilot emulation and the real Pilot implementation based on the defined KPIs for the decentralized energy flexibility service. We observe that most of the KPIs are assessed by both evaluation sessions and reveal that the results are similar (and consistent), thus validating the integrity of the evaluation approaches. Notice, that the Pilot emulation and the real Pilot implementation are based on a similar design of the MP component and the service deployment but exploit two different implementations, hence the possibility to present similar results due to implementation specifics is quite small. In

addition, the emulation supports more sophisticated MP operations, such as multiple winners of one campaign, which are applicable only to large scale setups that a real Pilot cannot support.

The KPIs are categorized in two general categories: the service-related and the blockchain-related. The former assess the performance of the service, thus investigating the goals of the energy flexibility application, while the latter asses the performance of the MP component which runs as the Ethereum nodes. The real Pilot implementation assesses all the KPIs, but the Pilot emulation is not applicable when measuring metrics related to the hardware of the IoT devices, since those devices are emulated. On the other hand, the Pilot emulation can assess the performance of the service in full deployment, when hundreds of EVs can be controlled by numerous FMs; this case is very difficult to assess through real implementation, since the market of EVs has not reached its full deployment yet. Overall, the two evaluation approaches are complementary tools towards the comprehensive performance assessment of the service.

Table 25: System performance KPIs for the DEFM emulation scenarios and pilot.

KPI	Emulation results	Pilot results	Comparison remarks
RPF reduction Amount of RP	0-1MWh/day (depending on setup)	13.7 kWh/ day,	Consistent. The emulation presented the same results with the Pilot for the same setup.
Power losses reduction	Up to 75% (>100% when RPF was scaled -- Table 23)	up to 75%	Consistent. The emulation presented the same maximum reduction with the Pilot for the same setup.
Voltage under the limits Voltage waveforms	-	maximum and minimum voltages are 0.98p.u. and 1.06 p.u., resp.	Not applicable to/measured in the Pilot emulation
Green energy consumption	0-1MWh/day (depending on setup)	13.7 kWh/day	Consistent. The emulation presented the same results with the Pilot for the same setup.
DR campaign money saved	-	0,13 €/kWh.	Not investigated in the emulation but would be identical considering the same price rates.
Ledger execution cost	Roughly 280k gas for write functions -- 1700k gas for deciding multiple winners	Roughly 290k gas for write functions	Consistent. The cost of deciding multiple winners is only applicable for the emulation.
Response time (for requests and offers/ for determining winner/ for verifying winner)	< 15s	< 15s	Consistent. The actions are completed in one mining period.
Throughput	> 100 per hour	> 100	Consistent
Scalability - time	Linear	Linear	Same as previous

4.2.8 Conclusions

In this evaluation session we explored the performance of the service under realistic conditions in a large-scale deployment. We exploited real traces produced by the Pilot implementation (and deployment) and we built a sophisticated emulator-tool that emulates the different service parts of the service with sufficient detail. The comparison of the emulation and the Pilot results verify the accuracy of the emulation.

Despite the non-optimal scheduling algorithm that the emulator considers, the results reveal that the efficiency of the service is enhanced when many bids from different FMs can win the flexibility campaign. By allowing multiple winners, the number of scheduled EVs increases significantly and the energy consumption increases roughly proportionally. However, this decision comes with the cost of resolving scheduling conflicts which decreases the rate of exploited EVs. We detect a security-performance tradeoff, where the solution adopted in this work is to use a light and not optimal blockchain-based solution, while an alternative would be to perform the conflict resolution off-band by a trusted third party, which would find a more efficient solution at the cost of decentralization and trust. Overall, the results yield great potential in promoting green energy, and that constitutes a great motivation for future work.

4.3 Decentralized Energy Data Exchange

The aim of the Decentralized Energy Data Exchange (DEDE) pilot is to enable trust between parties who exchange energy meter readings. The work related to this pilot reported in the previous deliverable D4.4 consists of two directions. The first direction involves mapping the pilot scenarios to PDS and IAA scenarios presented in different sections of this deliverable. The second direction focuses on the verification of smart meter data, which is a function highlighted in one of the pilot scenarios. The latter work involves developing and evaluating a model that captures the cost tradeoffs related to the frequency with which hashes of the smart meter measurements are recorded on a public blockchain.

For the DEDE scenario, D4.4 contains evaluation scenarios for the PDS and IAA components, which implement the authentication and authorization functionality required by the pilot scenario. These scenarios consider the usage of VCs to support privacy (for the PDS component) and OAuth 2.0 access tokens supported by Ethereum ERC-721 tokens (for the IAA component). We also extended the evaluation results that illustrate the tradeoffs involving the hash recording frequency and how they depend on various system parameters (transaction and verification costs, rate at which data is produced, and rate of verification requests

The new results contained in the current deliverable focus on evaluating the local differential privacy mechanism of the PDS component, utilizing traces from the actual DEDE pilot.

4.3.1 Overview

The four actors of the pilot are the following:

- Smart meter system operator: this is the entity responsible for some part of the energy grid. This entity is also the smart meter data collector (granted by data owner) and provides access to the smart meter data to third parties (Data consumers - energy service providers), after the request of the data owner.
- Smart meter data owner: this is the entity who is legally bound to the smart metering point and has the right to allow access to its smart meter data to third parties (Data consumers - energy service providers).
- Data consumer - energy service provider: this entity is responsible for providing the energy service to the end-user (data owner) and is the main user of the smart meter data to which the smart meter data operator, after the request of the smart meter data owner, provides access to.

- Auditor: this entity has an auditing role in energy grid operations and handles disputes between parties.

The aim of the pilot can be expressed in the following two high level scenarios identified in deliverable D5.2 (Initial Platform Validation):

- Data exchange scenario, which covers the sequence from identification, authorization to granting, requesting access, and exchanging smart meter data
- Data exchange verification scenario, which includes audit logging, maintaining tamper-proof evidence in case of disputes, and verification of the integrity of smart meter data.

4.3.2 PDS local differential privacy evaluation

As discussed in Section 3.2 the accuracy of the extracted result is affected by the number of responses. Hence, in a real-world deployment a data consumer would pay for the provided service only if a pre-defined number of responses is collected. On the other hand, supposedly a data consumer agrees to pay for a service if *X* responses are collected, but eventually *X - n* are collected, where *n* is a small number. In that case the consumer may be able to extract meaningful statistics, but it does not have to pay. Therefore, from a system operator and data provider perspective, a data consumer should not be able to extract any statistics before paying for the provided services. Finally, a data consumer should not be able to tell if a data owner is eligible to respond to a query. For example, suppose a consumer is interested in learning the average energy consumption of households located in Athens, Greece: from a privacy preservation perspective a data consumer should not be able to tell which data providers are located in this city, instead, it should rely on the service provider to "filter" the collected responses. In order to achieve the desired functionality, we adopt a smart contract-based approach. We assume that the system operator and the data providers share a secret key (PSK). Moreover, we assume that the system operator knows the *current* Ethereum address of each provider. We implement the following protocols, using the notation included in the next table.

Table 26: Protocols implemented for local differential privacy.

E(k,m)	**Symmetric encryption of message *m*, using the key *k***
H(m)	Hash of message *m*
HMAC(k,m)	Keyed MAC of message *m* using key *k*

Survey setup

With this protocol, a system operator and a data consumer agree on a smart contract that includes: (i) the question that data providers should respond to, (ii) filtering rules, (iii) the number of responses that should be collected, (iv) an amount of digital currency each responder will receive, and (v) a service fee. Additionally, the system operator and the data consumer agree on a nonce *n*: all data providers (and the system operator) can derive an encryption key *sk = HMAC(psk,n).* Then, the system operator sends *H(sk)* to the data consumer. Finally, *n* is included in the smart contract.

Response commit

Any provider wishing to participate in the survey, prepares a "noisy" response *R* using local differential privacy. It derives *sk* using the nonce *n* included in the smart contact and generates a ciphertext *C=E(sk,R).* Then it records in the smart contract *H(C),* as well as *H(H(C),H(sk)).* Since a data consumer knows both H(C) (from the smart contract), as well as H(sk) (from the

service operator) it can also calculate the second hash and verify that the data provider derived the correct key.

Response filtering

The system operator indicates in the smart contract which of the providers that responded abide by the agreed filtering criteria. If the number of the (valid) responders is bigger than the number of the responses that should be collected, then the system operator indicates which of the responders will be compensated. It also reveals *H(sk).* All approved data providers send to the consumer the encrypted, noisy responses (i.e., *C=E(sk,R)* calculated with the *Response Commit* protocol)

Fair exchange

The data consumer verifies the integrity of the received encrypted responses with the hash stored in the smart contract. When it receives the agreed number of responses it deposits to the smart contract the appropriate amount of currency. Then the system operator reveals *sk*; if the hash of *sk* matches the one provided with the response filtering, the contract transfers the deposit to the appropriate entities.

We implement a smart contract that provides the above functionality. In the following table we report the consumption in gas for each function

Table 27: Gas consumption for fair exchange functionality.

Function	Gas consumption
Contract creation	660 809
Commit Response	74 537
Record filtering result and H(sk) reveal	63 309
Deposit commit	23 642
sk reveal and deposit transfer	36 269

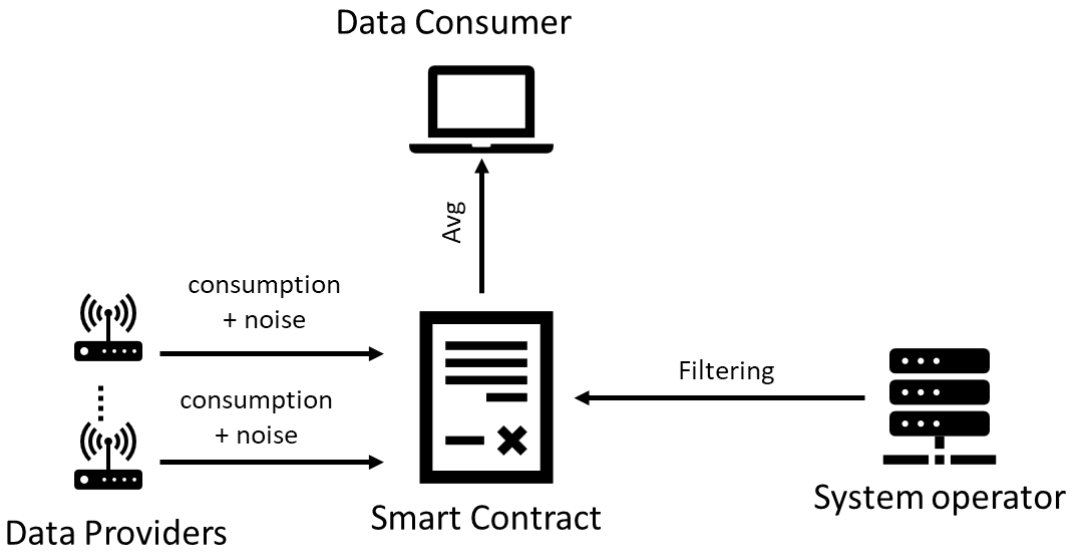

Figure 21: Evaluation setup.

In order to evaluate the efficiency of local differential privacy we used real network measurements provided by Guardtime. In particular, we used over 60.000 measurements of hourly consumption measured by smart meters. From these measurements we constructed 4000 measurements of daily consumption. We then executed our local differential privacy algorithm offering data providers 100 choices, ranging from 0 to 99 kw. The results are presented in the following diagram.

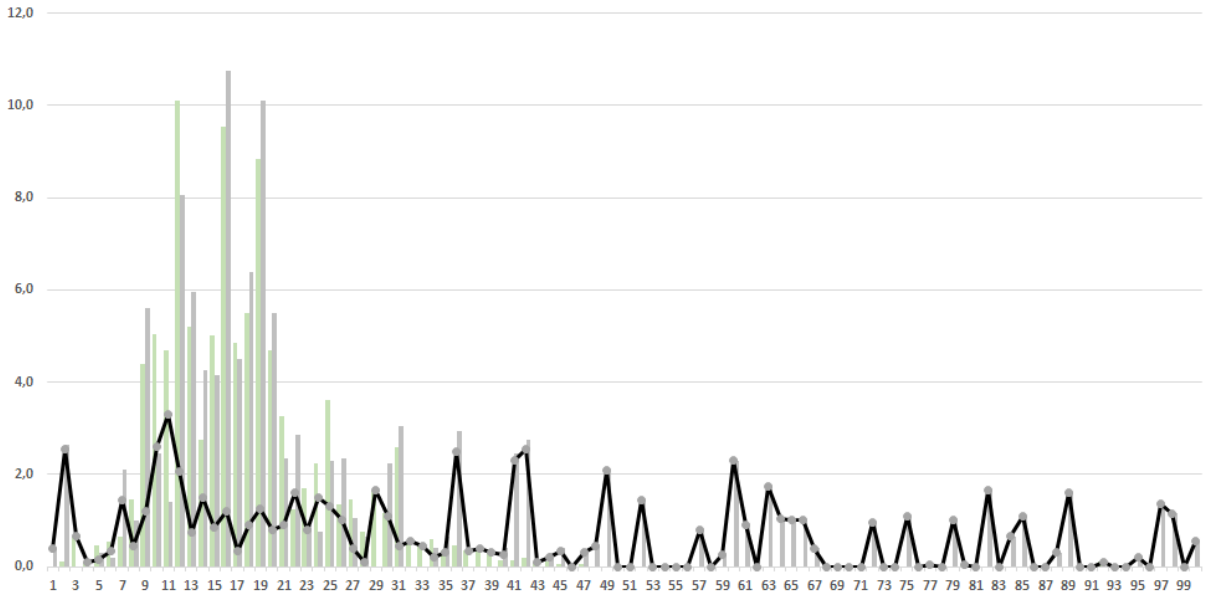

Figure 22: Differential privacy evaluation using real traces.

The green columns correspond to real measurements, the grey columns to the estimated measurements, and the black line to their difference. Although the black line is most of the time below 2, the error of the estimated measurements is more noticeable (compared to our measurements presented in section 3.2). The reason for that is that the real measurements are more scattered and hence closer to zero. The following figure illustrates the same experiment but with 8000 measurements. In order to produce this number of measurements we created synthetic results by extrapolating from the real ones.

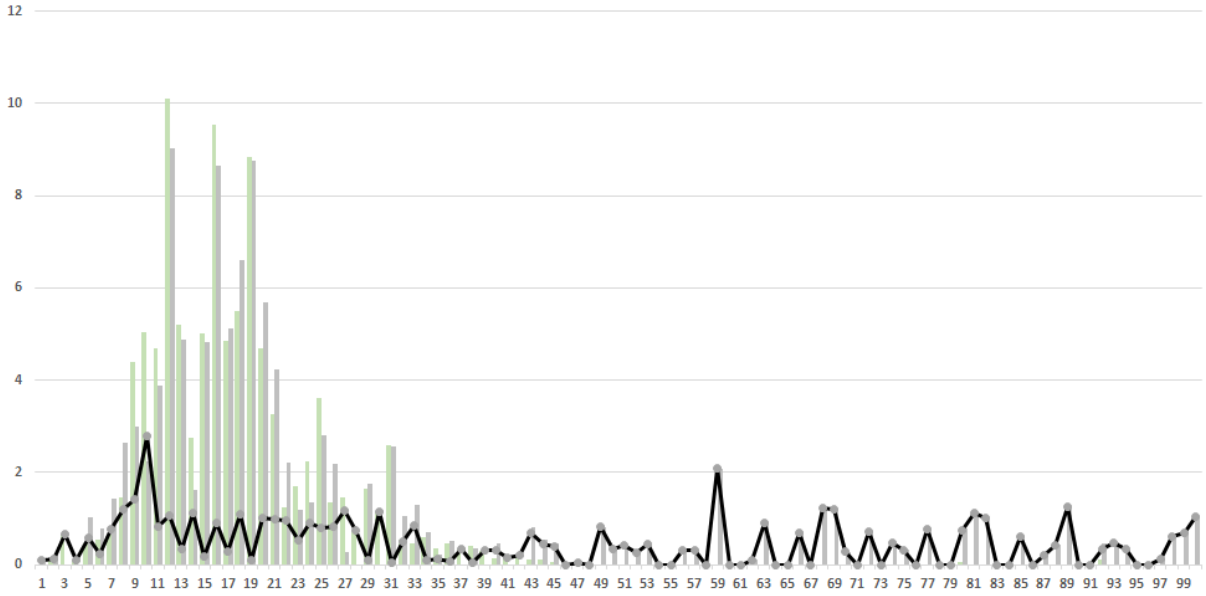

Figure 23: Differential privacy evaluation using real and synthetic traces.

4.3.3 Joint analysis of emulation and pilot results

The table below shows the evaluation results from the DEDE emulation scenarios and the pilot results. Some of the KPIs, namely KPI_DEDE_1, KPI_DEDE_8, and KPI_DEDE_9 where evaluated only in the emulation environment because they involved functionality and services that extended those of the pilot (KPI_DEDE_1) or referred to the scalability that was assessed only in the evaluation (KPI_DEDE_8 and KPI_DEDE_9).

The KPI regarding the cost for recording hashes (KPI_DEDE_2) involves the cost for recording hashes on a public DLT (for the emulation scenarios) or the KSI timestamping service (for the pilot). Also, KPI_DEDE_5 involves the response time for KSI, which was investigated in the pilot.

The KPIs regarding the response time for access requests (KPI_DEDE_3) and the response time for DID operations (KPI_DEDE_4) involves logical operations that are part of the time taken for request processing by the pilot federation adapter (KPI_DEDE_6).

Finally, the response time for audit logs (KPI_DEDE_7) was measured in the pilot. Because this KPI involves local read operations, the results do not differ in the testbed environment.

Table 28: System performance KPIs for the DEDE emulation scenarios and pilot.

KPI	Name	Target	Emulation scenarios	Pilot
KPI_DEDE_1	Cost for computing discounts	As low as possible	Results presented in D4.3 (Section 5.3.1). When only hashes are recorded on the public blockchain cost can be more than 80% lower compared to having smart contracts handle discounts. Moreover, the cost for recording hashes is proportional to the hash recording frequency (see also D4.4, Section 4.3.4).	Assessed only in emulation scenarios which extend the pilot scenarios
KPI_DEDE_2	Cost for recording hashes	As low as possible	Cost for recording hashes on public DLT is proportional to the hash recording frequency. The tradeoff with the verification cost is investigated in D4.4, Section 4.3.4.	The pilot utilizes a private ledger and KSI, which has a fixed cost per timestamp. Hence, the total cost is linear to the timestamp recording frequency.
KPI_DEDE_3	Response time for access requests	<5 s	Responding to a resource access request for the first time requires 2 roundtrips, one lookup in the blockchain, one signature generation, and one signature verification (see D4.4, Section 4.3.2). All operations can be done in less than 1 s.	The response time for access requests is part of KPI_DEDE_6 below, which assesses DEDE federation adapter's response time.
KPI_DEDE_4	Response time for DID operations	<5 s	DID operations are evaluated in D4.3, Section 3.3.2 and they required less than 200 ms, even in constrained devices.	The response time for DID operations is part of KPI_DEDE_6 below, which assesses the DEDE federation adapter's response time.

KPI_DEDE_5	Response time for KSI Blockchain signatures	<2 s	Identical to pilot result	1.2 sec (average). Result from D5.4 (Final Validation & Replication Guidelines)
KPI_DEDE_6	Processing time of requests in adapter	<5 s	Not applicable since it refers to the pilot's federation adapter	0.1 sec (average). Result from D5.4 (Final Validation & Replication Guidelines)
KPI_DEDE_7	Response time for audit logs	<15 s	Not measured in emulation scenarios. Involves local read operations hence results do not differ with those of pilot	0.3 sec (average). Result from D5.4 (Final Validation & Replication Guidelines)
KPI_DEDE_8	Scalability – cost	linear or sublinear	Results presented in D4.3 (Section 5.3.1) showed linear dependence of transaction cost on the frequency of discounts computations and hash recording frequency. The tradeoff with the verification cost is investigated in D4.4 (Section 6.3.4). If the optimal hash frequency is considered, the public ledger transaction cost as a function of the data rate is sublinear for a linear verification cost and constant for a logarithmic verification cost (D4.4,Section 6.3.4)	Assessed only in emulation scenarios
KPI_DEDE_9	Scalability – time	Linear or sublinear	The computational cost and delay of IAA and PDS is linear to the number of clients. Nevertheless, with respect to a single client, the computational cost and delay are sublinear to the number of requests, since a VC (in the case of PDS) and an access token (in the case of IAA) can be re-used.	Assessed only in emulation scenarios

4.3.4 Conclusions

For the DEDE scenario the new results contained in this deliverable involve the joint evaluation of evaluation results that involve the IAA and PDS components that are utilized by the pilot scenarios investigated in the previous deliverable D4.4 and results from the DEDE pilot reported in D5.4 (Final Validation and Replication Guidelines). The latter results pertain to the response time for KSI signatures, for processing requests by the pilot's federation adapter, and for audit logs. The main conclusion from the joint analysis is that the performance is not influenced by other functionalities of the federation adapter concerning the communication of different platforms, and they are consistent with the evaluation results for the IAA and PDS components, which are within the defined targets. The next new result of this deliverable is the evaluation of the local differential privacy mechanisms of the PDS component utilizing smart meter traces from the pilot. Specifically, we have evaluated the transaction cost when a smart contract is responsible for the fair exchange of data and the corresponding compensation between the data

provider and the data consumer. Additionally, we evaluated the accuracy of the local differential privacy mechanism for real traces from the DEDE pilot. The accuracy of the local differential privacy mechanism is evaluated with synthetic measurements in Section 3.2.

4.4 Context-Aware Mobile Gaming

The Context-Aware Mobile Gaming (CAMG) pilot prototyped a variety of mobile games in order to explore how DLTs can be used to provide new exciting features for players and more general advantages for the whole gaming ecosystem, as discussed below. The first use case prototyped is a game that enables players to collect and trade assets with other players. The second use case is a scavenger hunt location-based game, and the third use case is the Blockmoji, which is an application that players can use to see the items they own and equip them on their virtual avatar. The goal of these games is to explore various use cases of blockchains within gaming, like payments based on cryptocurrency or real-world money, advertisements (including an open advertising ecosystem built on top of DLTs, which we describe and investigate Section 4.4.6), and IoT technologies (e.g., IoT beacons).

In this deliverable, we summarize our efforts in evaluating the CAMG pilot inspired use cases, through emulation and also juxtapose and discuss the evaluation results obtained by the pilot. In particular, we present all the results that we have gathered so far, from all the emulation scenarios we have developed for the scavenger hunt game and then we compare them with the results taken from the evaluation of the actual implementation of the game in the pilot.

Furthermore, in Section 4.4.6, we introduce one more emulation scenario, which utilizes Hyperledger Fabric and public Ethereum, in order to investigate an open advertising ecosystem for the scavenger hunt location-based game. This is in addition to the (open) federation investigations with System Dynamics in Section 5. Our study here uses a real Ethereum smart contract and Hyperledger Fabric chaincode deployed on real blockchains to investigate the essence of a DLT-enabled open ecosystem in a plausible pilot-inspired use-case.

4.4.1 Overview

The scavenger hunt location-based game, defined in Deliverable 5.1, and described in a more detailed manner in Deliverables 5.2 and 5.3, is a game, where players have to solve some riddles using received clues to reveal the next target location. By solving the riddles, the players get points, which they can exchange with rewards and assets. The flow of the main game is as follows: Initially, the player sees the available nearby hunts (based on GPS location). Then, she selects a hunt and receives the clues to the first location she must visit. Next, the player physically visits the area, where the IoT device is deployed. When she visits the location, she has to answer all the questions correctly. Solving the riddle (answering all the questions) will reveal the location of the next Point of Interest (PoI), in order to go there, collect her points, and download the next riddle. This procedure continues until the last IoT device is reached.

In addition to the main functionality of the game described above, the mobile game provides some additional features. A player will be able to skip any challenge she wants by viewing an advertisement, by paying in in-app tokens or in real-world money. Furthermore, if a player watches an advertisement, irrespective of skipping the challenge, she will get a reward given by the advertising company. Moreover, the player is able to get in-app tokens by paying or by viewing an advertisement. Finally, the player can, at any point of the game, redeem her points in order to get rewards given by the game company or by any other company participating in the ecosystem. The rewards can be assets (e.g., a sword or a shield) that can be used in the game or in any other games that use the same blockchain platform, thanks to the blockchain's properties (immutability, transparency, etc.). In the CAMG pilot the management of these assets is performed by the Blockmoji application.

For the IoT part of the game, IoT beacons are used to provide the proximity location of the player when she visits the appropriate location. The beacons communicate with the smartphone of the player using the Bluetooth Low Energy (BLE) protocol.

4.4.2 Emulation overview

As already mentioned in the previous Deliverables 4.3 and 4.4, we have developed an emulation environment in order to evaluate the performance of the scavenger hunt location-based game use case of the CAMG pilot. Our emulation environment supports public (Rinkeby public Ethereum test network) and private (private Ethereum and/or Hyperledger Fabric) DLTs, allowing us to experiment with different configurations and architectures with the two types of blockchain. In the previous deliverables, the emulation environment helped us to evaluate the system based on defined performance metrics (KPIs). In this deliverable, we introduce one more emulation scenario that investigates one of the core ideas of this project, namely openness (Section 4.4.6) in business models. Due to the emulation environment, we can easily extend one of the already defined emulation scenarios and develop a proof-of-concept implementation of an open advertising ecosystem for mobile gaming, backed by DLTs. On the other hand, with our emulation we cannot investigate business-oriented metrics, such as player satisfaction.

One of the main components of the emulation environment is the client application, which emulates the mobile gaming client, and is written in JavaScript. This application has a simple UI that both players and administrators (game administrator, ads administrator, etc.) interact with. The remaining components of the emulation environment are the three smart contracts, which emulate the game, the advertisements, and the tokens/rewards respectively. The smart contracts are deployed on public/private Ethereum or Hyperledger Fabric (depending on the emulation scenario). The IoT related functions (such as beacon discovery, beacon deployment, etc.) are not included in the emulation.

The game smart contract communicates directly with the client application and implements the functionality related to the challenges. Specifically, it records the challenges on the ledger and a mapping of players and challenges and whether a particular player has completed a challenge or not. Furthermore, it automatically calculates the points for each player and implements the functions for skipping a challenge and redeeming the rewards. The second smart contract is responsible for the advertisements, while the last smart contract creates and manages the in-app tokens (rewards).

4.4.3 Emulation scenarios

One of the advantages of using emulation is that we can experiment with various configurations and scenarios involving different DLTs setups. In Deliverable 4.4 we presented four different scenarios in order to evaluate the scavenger hunt location-based mobile game. In this deliverable, we introduce a 5th emulation scenario () that acts as a proof-of-concept implementation of an open advertising ecosystem for DLT-assisted mobile gaming. As we describe below, in order to develop the 5th emulation scenario, we slightly modified the architecture of the emulation environment and some of its components, in particular the ads smart contract. Below, we briefly present the four scenarios (for a more detailed description of the emulation scenarios refer to Deliverables 4.3 and 4.4).

4.4.3.1 Scenario 1 - Public blockchain

The first scenario considers a single (public) Ethereum blockchain, which implements all the gaming functions of the three aforementioned smart contracts. Thus, all smart contracts are deployed on the same (public) Ethereum network. The experiments in this scenario took place on the Rinkeby test network.

4.4.3.2 Scenario 2 - Two types of blockchains (public – private Ethereum)

The second scenario investigates the gains from utilizing two types of blockchain, a public blockchain and a private/permissioned blockchain. In this scenario, the first blockchain is a public Ethereum, while the second blockchain is a private instance of Ethereum. Thus, the smart contracts are deployed on different blockchain networks: the game smart contract is deployed on the private Ethereum blockchain, while the other two smart contracts are deployed on the public Ethereum and in the Rinkeby test network. The interconnection of these two ledgers is performed through an Interledger Gateway (ILG). The ILG "listens" for events on both Ethereum blockchains. Such events are generated each time a player invokes a function of the game's smart contract that needs to perform an action involving the in-app tokens or the advertisements.

4.4.3.3 Scenario 3 - Private blockchain

The third scenario considers a single ledger but differs from the first scenario in that the blockchain is a permissioned blockchain, namely Hyperledger Fabric, rather than the public Ethereum. As in the first scenario, there is no need for an ILG since smart contracts are deployed on the same ledger. However, there are some differences between these two scenarios, due to the different blockchain technology used. In particular, in this scenario a player cannot skip a challenge by paying in cryptocurrency, because Hyperledger Fabric does not support cryptocurrencies.

4.4.3.4 Scenario 4 - Two types of blockchains (public Ethereum – Hyperledger Fabric)

The fourth scenario utilizes two (different) blockchains instead of one. The first one is the public Ethereum, while the second one is Hyperledger Fabric. The game smart contract is deployed on Fabric, while the ads and tokens smart contracts are deployed on the public Ethereum (Rinkeby test network). In this scenario, we need an ILG for the communication between the two ledgers. The ILG implements both the Fabric SDK, as well as the communication with the Ethereum blockchain, using the web3 library.

4.4.4 Evaluation results

In deliverable D4.4, we presented and analysed the performance evaluation results, based on the defined KPIs, for the four emulation scenarios that we have developed. Furthermore, we showed for each emulation scenario, whether the defined target for each KPI is achieved. We summarize these results in the table below.

Table 29: System performance evaluation results for the CAMG emulation scenarios.

KPI	Name	Target	Scenario 1	Scenario 2	Scenario 3	Scenario 4 (Fabric & Ethereum)
KPI_CAMG_1	Public ledger execution cost	As low as possible	50164	36437	0	0 & 36437
KPI_CAMG_2	Response time for write requests	< 3 sec.	13.89 sec.	13.89 sec.	2.189 sec.	2.197 & 13.89
KPI_CAMG_3	Response time for read requests	< 1 sec.	1.105 sec.	1.105 sec.	0.024 sec.	0.025 & 1.124
KPI_CAMG_4	BLE beacon detection time	< 4 sec.	Not Applicable	Not Applicable	Not Applicable	Not Applicable
KPI_CAMG_5	Throughput	> 222 read and > 133 write transactions per second	4 write tr/sec	4 write tr/sec	200 write tr/sec	Same as scenario 3 for Fabric and

			> 222 read tr/sec	> 222 read tr/sec	> 222 read tr/sec	scenario 1 for Ethereum
KPI_CAMG_6	Cost Scalability	Linear or sublinear	Linear	Linear	Not applicable	Linear (for Ethereum)
KPI_CAMG_7	Time Scalability	Linear or sublinear	Step-wise function	Step-wise function	Step-wise function	Step-wise function

The results in the table above are the average of all the appropriate actions of all experiments. For example, for the 1st KPI, in the 1st emulation scenario, we have 9 actions (e.g., add a new challenge, begin a challenge, skip a challenge, etc.) that require gas. So, the result in this cell presents the average of all these actions. Furthermore, for the 3rd KPI, in the 1st emulation scenario we have 3 actions (query points, challenges and tokens) that send read requests, and for each one of these actions we performed 10 experiments. So, we calculated the average value for each of the actions and then the average value of these actions, which is presented in the appropriate cell in the table above. Moreover, for the 4th emulation scenario, which utilizes both Hyperledger Fabric and public Ethereum, we present the results for both blockchains.

In our emulation environment, we did not develop the IoT related functions, and that is why the row for the 4th KPI does not contain any results. Finally, for the throughput KPI, we have calculated that Hyperledger Fabric can support 117 sequentially read transactions per second. However, we can achieve the desired target, which is more than 222 transactions per second, if we send the transactions in parallel. For a more comprehensive analysis of the results, as well as of the experiments, refer to Deliverable 4.4.

From the above table, it is clear that the use of a public ledger (scenario 1) is expensive and demonstrates poor performance results, especially in terms of response times and throughput. Using a single private ledger (Scenario 3) is way better than using a single public ledger, in terms of cost and performance. However, we miss some important properties of the public ledger, such as transparency and openness. So, the best solution is to combine these two types of blockchains, the private ledger in order to implement all the gaming functionality that time is a crucial factor, while the public ledger in order to implement other actions, such as advertisement and asset specific actions, that require higher levels of trust, transparency and openness in order for the entities to be able to join the ecosystem easily. So, since Hyperledger Fabric demonstrates better performance results than a private instance of Ethereum, we end up that the 4th emulation scenario is better than the 2nd emulation scenario that utilizes public and private Ethereum.

4.4.5 Joint analysis of emulation and pilot results

This section compares the results produced from the emulation environment (reported in deliverable D4.4) to the results produced from the pilot actual implementation (deliverables D5.3 and D5.4). The emulation scenario that fits better in the actual implementation of the pilot is emulation scenario 4, which also considers the same ledgers and has pretty much the same architecture. So, in order for the comparison to be fair, we compare these two implementations, the actual pilot and the 4th emulation scenario. The comparison of the KPIs between these two implementations is shown in the table below.

Table 30: Joint analysis of emulation scenario 4 and pilot results.

KPI	Name	Emulation Scenario 4	Pilot
KPI_CAMG_1	Public ledger execution cost	0 in Fabric 36437 in Ethereum	-
KPI_CAMG_2	Response time for write requests	2.197 sec. (± 0.016) in Fabric 13.89 sec. in Ethereum	2.247 sec. in Fabric
KPI_CAMG_3	Response time for read requests	0.025 sec. (± 0.022) in Fabric 1.124 sec. in Ethereum (RPC) 0.045 sec. in Ethereum (local)	0.026 sec. in Fabric
KPI_CAMG_4	BLE beacon detection time	-	5.8 sec
KPI_CAMG_5	Throughput	117 seq. read tr/sec 200 write tr/sec	307 read tr/sec 128 write tr/sec
KPI_CAMG_6	Cost Scalability	Linear	Linear
KPI_CAMG_7	Time Scalability	Stepwise function (Macroscopic: Linear)	Linear

The first KPI involves the cost for executing operations in a public ledger. For the 4th emulation scenario, some actions (get token by viewing ads) require the invocation of a smart contract (ads or token smart contract) that is deployed in public Ethereum. So, these actions incur a transaction cost, which is 36437 gas on average. On the other hand, on the pilot implementation the gas is not measured, as it used a permissioned ledger for most of the actions. The second KPI refers to the time needed for the system to respond to game state altering transactions, namely write requests, while the third KPI refers to the time needed for the system to respond to read requests. For these two KPIs, the average 95% confidence interval is shown (in parenthesis) in the table above. As we observe from the table above, the results for these two KPIs are very close to each other, as we use the same private ledger, Hyperledger Fabric, in both implementations. However, there is a small difference between them, which occurs because in the pilot implementation, Hyperledger Fabric was deployed on AWS, while in the emulation environment, Fabric is deployed locally in a VirtualBox image.

The fourth KPI is the BLE beacon detection time. As we have already mentioned, our emulation environment does not emulate IoT related actions, thus we have not produced results for this specific KPI. However, in the pilot implementation, the detection time for BLE beacons has been measured and its average value is presented in the table.

The next KPI is about throughput, which is defined as the maximum number of transactions per time unit. In the emulation environment we have measured the throughput of Fabric, and we found that Fabric can support 200 write transactions per second and 117 sequential read transactions per second. In order to determine the throughput, we calculated how many transactions can fit in a block, and how much time is needed for a block to be added to the ledger. On the other hand, in the pilot implementation, we measured the throughput of the system by submitting multiple transactions to the blockchain at a fixed rate and at a composite rate. In the first case, the throughput is 191 write transactions per second and 351 read transactions per second, while in the second case, which simulates a real-life scenario, the throughput is 128 write transactions per second and 307 read transactions per second. The difference between the throughput found by the emulation and the one found by the pilot implementation, occurs because in the pilot implementation, they performed the test with a duration-based round. An initial 30-second normal workload is followed by a 30-second intensive workload, which is followed by 10 seconds of low workload and ending with another 30 seconds of normal workload. In the emulation environment, we did not consider the rate at which transactions are submitted on the blockchain, and that is why the emulation result is closer to the result acquired by the first experiment (fixed rate).

The remaining KPIs refer to scalability, cost scalability and time scalability, respectively. Cost scalability is investigated only with respect to actions that utilize public Ethereum, which incur a transaction cost, and is defined as the ratio of gas cost over the number of challenges. In both implementations the scalability is linear. Finally, time scalability is a stepwise function for the emulation environment, but macroscopically it becomes linear as in the pilot implementation.

4.4.6 An open advertising ecosystem for DLT-assisted mobile gaming

One of the core components of the scavenger hunt location-based game is the blockchain technology, which is mainly used to record and manage information, including state, on the ledger (e.g., a player arriving at an IoT beacon location, perform the points calculation and recording the current value for each player, trading assets, etc.). However, in addition to these actions and features (logging, immutability, etc.), blockchain technology can help in the creation of open, federated ecosystems, where many different organizations (e.g., advertising companies and game studios in the CAMG pilot case) can securely and trustily cooperate with each other, but without one organization having a veto power over membership, as is the case with most current platforms, e.g., the Apple App Store and Google Play.[12]

One such ecosystem is the system described in the CAMG pilot, where three different companies, the gaming company, the advertising company, and the PoI company, cooperate with each other in order to provide more features for the players. In particular, regarding the advertisements, the procedure that the CAMG pilot follows is the following: initially, the advertiser has to create an advertiser account, which needs the approval of the game administrator. Then, he can log in to the Web application and upload the video with the corresponding reward. When advertisements and their related information are approved by the game administrator, a smart contract is created and deployed in the blockchain for this specific advertisement.

It is clear from the above procedure that the ecosystem is in principle open to any advertising company. However, in this specific case, there is a need for approval from the game company. In order to further exploit the openness feature of SOFIE's architecture and to show that blockchains can indeed help in the creation of truly open ecosystems, we developed a 5th emulation scenario, which emulates a completely open advertising ecosystem for DLT-assisted mobile gaming.

The architecture of this scenario is presented in the following figure (Figure 24). The main components of the system are the client application, which is the same application as in the previous emulation scenarios, the Hyperledger Fabric and public Ethereum blockchains, the Interledger Gateway (ILG), the three smart contracts, and the advertising companies.

[12] Consider the recent removal of the Fortnite game from both Apple and Google platforms, a widely reported news story: https://www.theguardian.com/technology/2020/aug/13/fortnite-maker-lawsuit-apple-app-store-removal.

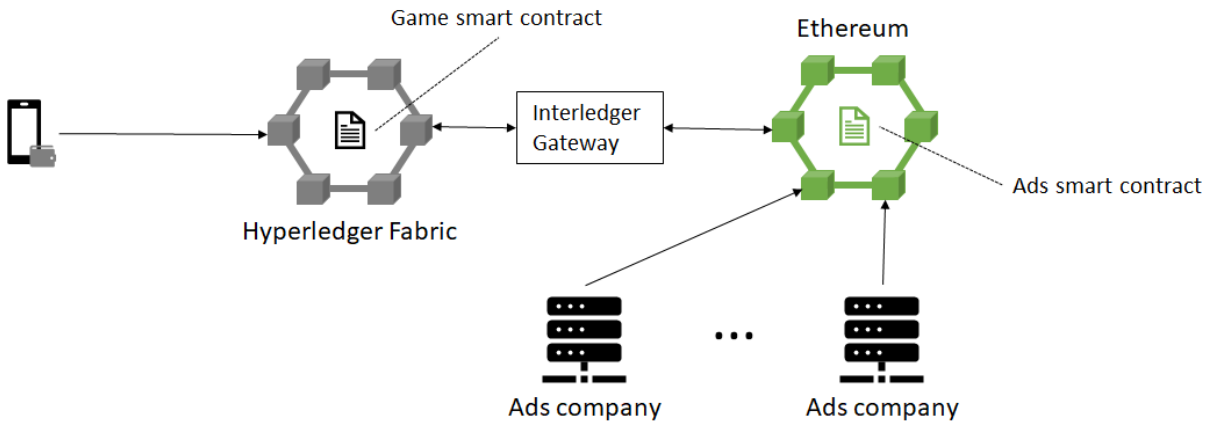

Figure 24: An open advertising ecosystem for DLT-assisted mobile gaming.

In this ecosystem, we utilize two different types of blockchains, Hyperledger Fabric and public Ethereum. As in emulation scenario 4, the game smart contract, which implements all the main functionality of the game (e.g., challenges, players, points) is deployed on Hyperledger Fabric, while the token smart contract, which creates and manages the tokens/rewards is deployed on public Ethereum. The ads smart contract, which is responsible for the advertisements, is also deployed on Ethereum as in the 4th emulation scenario.

In this scenario, with regards to advertisements, we have one smart contract entry for all the advertisements, as opposed to the previous emulation scenarios and the CAMG pilot, where we had one smart contract per advertisement. This smart contract is deployed on the public Ethereum by the game administrator. Anyone having an Ethereum wallet can supply an advertisement for the game by simply invoking this smart contract. The entity that invokes the smart contract, an advertising company in our case, must provide in the smart contract the URL of the advertisement, as well as compensation for the game provider (which is the owner of this smart contract). After that, the URL of the advertisement as well as the Ethereum address of the advertising company are stored in a list within the smart contract on Ethereum and an event is triggered. Eventually, the event will be caught by the ILG, which will forward the URL of the advertisement and the Ethereum address of the advertising company to Hyperledger Fabric, in order to also be stored in the game smart contract (chaincode). By storing the advertisements in Hyperledger Fabric too, we perform some sort of caching, because we improve performance by avoiding having the delay introduced by Ethereum (~2 seconds for read request, as measured in other emulation scenarios), when the player wants to "watch" the advertisement. (We do not completely avoid this delay, as typical in caching, but we endure it once, when it is not crucial for the game, and without disturbing the player.)

Furthermore, with the use of the ads smart contract, it is possible for the game company to implement policies that need to be respected when adding advertisements in the game and also for selecting which advertisement will be displayed when a player wants to "watch" an advertisement and make all this transparent and immutable. In this emulation scenario, we chose to implement a First Come First Serve (FCFS) policy, which can also be combined with some minimum compensation for the game provider. However, we can also imagine other, more elaborate, potentially more economically efficient policies. For example, the game company may perform auctions for advertisements or even consider context-awareness in the selection of the ads. Finally, in this emulation scenario, as in the other ones, the players that watch an advertisement can skip a challenge or acquire tokens/rewards.

Of course, there are new issues that arise from this open approach, which are beyond our investigation. Mainly whether inappropriate content is introduced through the ads and also whether open business platforms are beneficial, for the ecosystem participants, and whether

they are sustainable. Adoption and sustainability of such open, federated, ecosystems is further investigated in more abstract terms with System Dynamics in Section 5. On the other hand, clearly, Apple and Google have elected to exercise tight control over their store platforms, to some extent justifying it with a rationale of controlling content for appropriateness (and security), but they have also been able to extract sizeable commissions. These commissions are significant inefficiencies for these markets, which can be detrimental to wider innovation and a barrier to entry for very small new players. Such issues are often dealt with through regulation when they achieve significant visibility. But true evaluation of the business impact of these ideas is only possible through the market. We do not attempt further consideration of these two issues here. Instead we demonstrate the technical feasibility of DLT-assisted open platforms and evaluate the easiness of implementation and performance impact on game performance and evolution.

To implement the ecosystem described above, in our emulation environment, we modified the 4th emulation scenario that utilizes Hyperledger Fabric and public Ethereum. In particular, we developed a new ads smart contract, which has all the functionality that we described above. Furthermore, we changed the ILG in order to “listen” for the event generated by the new ads smart contract and transfer the URL of the advertisement and the Ethereum address of the advertising company to the game smart contract. All the other components (client application, token smart contract, ILG actions for tokens, etc.) of the emulation environment have remained the same, as in emulation scenario 4.

A more technical representation of this scenario is presented in the following figures that illustrate the sequence diagrams of the procedure. In Figure 25 we present the sequence diagram for the addition of a new advertisement. Initially, the advertiser (advertising company), which owns an Ethereum address, calls the appropriate function of the ads smart contract, which is deployed in public Ethereum. If the advertiser has included in the transaction the required compensation for the game company, the advertisement is stored in the blockchain. Then, an event is emitted and is eventually caught by the ILG, which sends a transaction on the game smart contract to also store the advertisement in the Hyperledger Fabric.

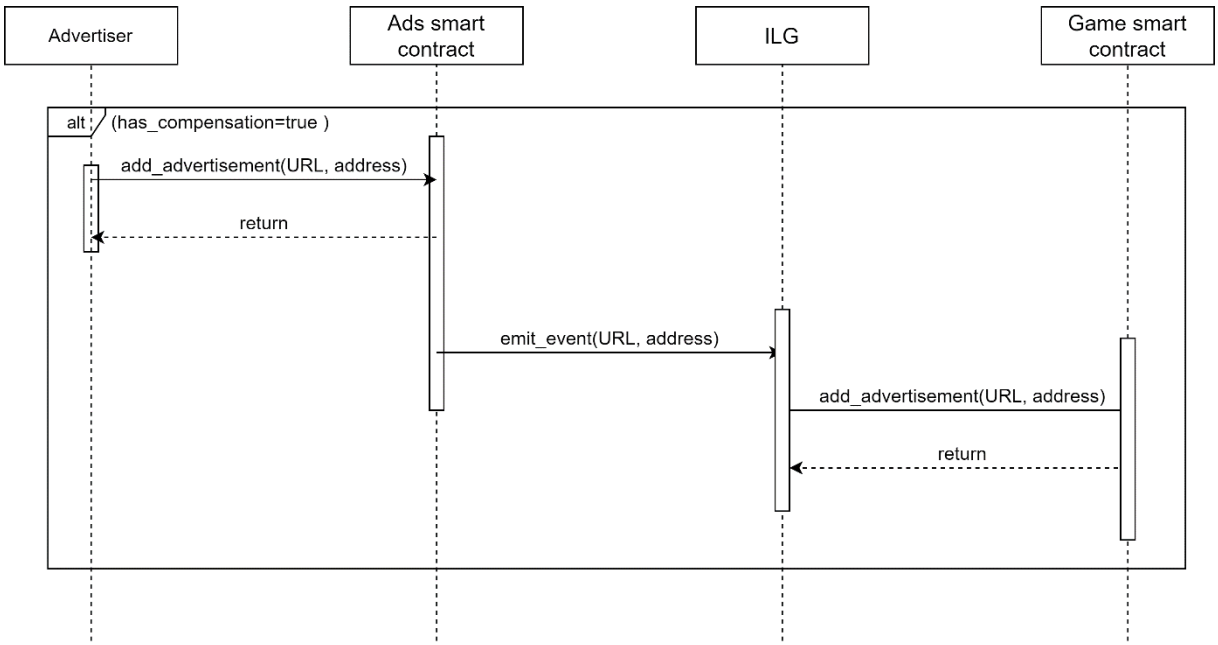

Figure 25: Sequence diagram of emulated scenario 5: Add advertisement.

The gas consumption for the advertisement smart contract in shown in Table 31. The transaction delay for the new add invocation is around 15 seconds, which is the average time for mining a new block on the Ethereum blockchain.

Table 31: Gas consumption for the advertisement smart contract.

Function	Gas consumption
Smart contract deployment	328 693
New add invocation	46 335

Figure 26 presents a sequence diagram that illustrates what happens when a player wants to "watch" an advertisement. The player has to call the appropriate function of the game smart contract in order to get the URL of the advertisement. We do not present here the time needed for these actions to be completed, since these actions are the same as in the other emulation scenarios, where we have already measured them (see KPI 2 and 3 for the 4th emulation scenario).

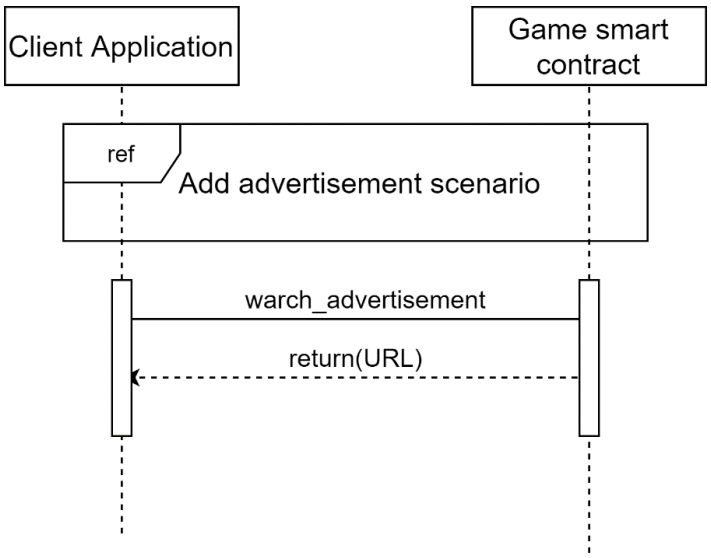

Figure 26: Sequence diagram of emulated scenario 5: Watch advertisement.

We developed this emulation scenario, in order to investigate the feasibility and potential gains and costs of such an ecosystem. First of all, this ecosystem is truly open, as there are no barriers to entry and anyone with an Ethereum address (which are freely available) can add an advertisement without any intervention from the game company, or other middlemen. Of course, the initiator of the ads smart contract, typically the gaming company, can introduce conditions to be checked automatically by the smart contract. One such expected condition is that the entity that wants to add an advertisement must also add compensation for the game provider. Note, however, that the form and terms of the compensation will be transparent and non-discriminatory. With the immutability of DLTs, these terms will remain fixed over time and this may be an issue. But one can easily imagine that the entity controlling the game could improve the deals by providing another smart contract (with better terms), which will be preferred by advertisers as soon as it becomes available. The reverse does not seem economically realistic (i.e., to worsen the terms—but this is consistent with economic history). Furthermore, except

from the openness feature, we have in this ecosystem all the benefits and the advantages that come with the use of blockchains, such as transparency, availability, logging and non-repudiation. With the use of the smart contract, we have automation, in the sense that when someone adds a new advertisement, compensation is automatically added to the smart contract owner's wallet. Moreover, with the use of smart contracts the game company can be sure that the policies that have been defined will be respected and that someone can add an advertisement only if they have provided the appropriate compensation.

On the other hand, with an open ecosystem, as also discussed above, we have one drawback. It is very difficult for anyone (game company, advertising company, game administrator, etc.) to control the content of an advertisement, since the video of the advertisement is not stored in the ledger, but only the URL of the video is stored in it. However, we can address this issue by having the gaming company define some rules and check whether these rules are respected. In this case, the ecosystem is not completely open, like the one presented before, but the gaming company has some sort of control, which in some cases is desirable or required. Furthermore, we can consider cases which have the same result but in a more open and decentralized way. We can introduce oracles that can communicate with the web server that has the actual video (or metadata) of the advertisement, which is outside the blockchain, and check if the defined rules are respected. These oracles would need to be trusted or some form of consensus among multiple oracles checking the same advertisement video would be necessary.

As we have already mentioned, when a user wants to watch an advertisement, he has to communicate with the game smart contract, in order to get the URL of the advertisement. With a design like that, it is very difficult for the game or advertising company to enforce or check whether the user watched the video of the advertisement. The solution may be that the game company sends back to the user the actual video and not the URL of the advertisement, but this is something that we cannot develop and investigate through emulation.

4.4.7 Conclusions

For the CAMG pilot and use case, this deliverable summarizes our evaluation efforts. In particular, it contains a summary of the results gathered through emulation for four scenarios that consider different DLT setups and architectures. Furthermore, it provides a comparison of the results between the emulation scenario closest to the pilot and the actual implementation and its evaluation in the pilot. The process has demonstrated that the results are consistent between our emulation approach and the pilot, but also that the two approaches should be considered complementary. The emulation approach allows much greater flexibility and many more scenarios to be researched, as well as the investigation of scalability. The prototypical implementation of the pilot is much more realistic and in particular more easily allows the investigation of issues of real IoT and mobile devices, including performance issues related to discovery and access to sensors and third-party systems, which is much harder to do through emulation. The pilot had also the form factors and the user-interface that allow some level of investigation of user acceptance, something that we chose to completely avoid in the emulation. On the other hand, the pilot is limited by the performance of actual and available devices and systems and makes it very difficult to examine many what-if questions.

Finally, this deliverable introduces a 5th emulation scenario, which implements and investigates a fully open advertising ecosystem and demonstrates that openness, a key feature of SOFIE, is technically feasible through the exploitation of open (public) DLTs. Furthermore, through the use of interledger technology (and the SOFIE ILG component specifically) and a design based on caching, the performance impact of the open ecosystem is unfelt by the gamer. On the other hand, for advertising companies, the open, direct participation in the gaming ecosystem, automatically through smart contracts, might be a significant advantage even if the gaming company traditionally controlling participation through registration is "open" and benevolent,

because it can decrease the time for starting the participation and through the transparency and constancy of the policy. If this leads to wider acceptance of the process by advertising companies, it could lead to benefits even for the gaming company and thus make it a win-win situation, which will facilitate its adoption and sustainability.

Finally, at a high-level, the results show the gains that can be achieved for the CAMG use case, in terms of cost and performance, when public and private ledgers are combined. A private ledger is better in terms of cost and performance and that is why it is used for the main gaming functionality, while a public ledger offers transparency and openness and that is why it is used for actions that require these properties, such as managing assets and advertisements.

5 DLT-based Federation and SOFIE business platforms

The purpose of this chapter is to evaluate how DLT-based federation using the SOFIE Framework affects the business value of the SOFIE application areas and pilot-inspired use cases. While the SOFIE Framework provides the necessary technical capability for DLT-based federations [LBBC19, Sir+19d], the business problem of why it would be beneficial for platforms to participate in such federal arrangements in the first place is not primarily a technical problem, and further investigation into the business-side of the platforms is required to better understand the impact of federation.

Federations consist of self-sovereign entities partially under central governance.[13] In the case of SOFIE, a federation is defined to be a seamless collaboration across many entities with distinct administrations in such a way that the final result appears as a single, well-integrated platform [Sofi20, Section 4], [LBBC19]. Thus, IoT Federations can be understood to be federations which combine different IoT platforms to facilitate cooperation while maintaining the sovereignty of the existing heterogeneous platforms.

Business platforms are multi-sided marketplaces in which two or more platform customer groups benefit from each other. Here, s*ame side network effects* bring up the value of a platform for each individual adopter (entity, company, group etc.) as more adopters enter that side of the market, while *cross-side network effects* bring up the value for individual adopters on one side as adoption increases on some other side [KaHR20]. Examples of business platforms are modern smartphone ecosystems with their operating systems (mainly iOS and Android) and the associated application stores and apps (which we have studied with System Dynamics and discuss in [AEN19]).

The value proposition of federations seems clear enough: with a single decision, it is possible to multiply the network effects [RuCK17] for all the parties involved and for their collection as a whole. However, federations also appear to face multiple practical problems: having a central governing party may facilitate the operations of the federation, but it also empowers the governing party, and if the goals and motives of that party significantly differ from the federation members', it may result in the governing party exploiting its position, including in the form of delays and costs for parties joining or transacting, insufficient transparency to the operations of the platform, and limits to the access of the platform for existing and new members.

The promise of the SOFIE approach is to utilise DLTs to counter these problems by supporting transparency with the ledgers, reducing costs and delays by automating actions with smart contracts and other related technologies, and opening the platform to all parties, while enabling accountability and even collective countermeasures against misbehaving members. Opening a platform to all parties in an automated way can be achieved through, e.g., "running" the federation with smart contracts in public, permissionless blockchains, in a way that any other entity can interact with the platform and its parties without requiring any initial permission or other registration before doing so. The core value of the SOFIE framework is derived from interledger technology, which allows effective utilisation of multiple DLTs in parallel, and decentralisation according to the capabilities of the individual ledgers and governance policies.

This chapter focuses on two important and distinct aspects for the federation to succeed: dynamics of collaboration in which member actions may have unintended negative effects for others, and the impact of network effects on the dynamics of platforms. These questions have been studied using System Dynamics as the analysis methodology. This chapter addresses the following research questions:

1. Why is the DLT-federation better than a federation in general?
2. How are concepts such as transparency and trust contributing to federation?

[13] https://en.wikipedia.org/wiki/Federation

These questions are answered using simulation on an archetypal cooperation model amended with the DLT effect and its associated simulation parameters in Section 5.2 and with the detailed risk tables of Section 5.4.

Table 32 compares DLT-based IoT federation (DLT federation) with more traditional federations and centralised organisations.

Table 32: Comparison of centralised organisation, traditional (judicial) federations, and DLT-federation on decision speed, decision method, and method of decision verification.

	Centralised	Federation with central governance	Federation without central governance	DLT federation
Decision speed for decisions on behalf of the collective	Medium	Medium	Slow	Fast[14]
Method of arriving at decision	Authority	Authority	Voting on every decision	Configurable / automated voting
Method of decision verification against member consensus	After the fact, if it is (still) part of governance	Election of authority, facilitated by (after the fact, "manual") transparency	Integrity of the voting system and its transparency	Continuous transparency, and possibly decentralised enforcement

The results of this section show how DLTs help cooperation to continue in federation arrangements, making them more stable and that the adoption of federation is accelerated by DLTs. This requires the governance policy applied through the DLTs to be sufficiently strong. Below, Table 33 describes the key components or parameters for the DLT effect (and federations in general) and Section 5.2.2 shows the modelling of the most important DLT effect components. The simulation results in Sections 5.2.3 and 5.3.3 further provide threshold values for qualitatively distinct behaviours of the DLT effect.

5.1 IoT Federations are based on openness

IoT business platforms consist of independent entities with their own administration and processing internally. To enable effective cooperation with these entities and to entice new members to the federation, the facilitation of free flow of information using universally accepted protocols among IoT business platforms is a key factor. Open Application Programming Interfaces (Open APIs[15]) allow that integration since the APIs are available publicly, typically without access control or usage fees.

Federation, however, is mainly a governance issue and requires some level of maturity of the platforms. Thus, a functioning federation requires that two or more parties have actually been integrated to it via an API of the one-to-many type i.e., of the federation-to-members type. It is clear that standardized open APIs are necessary but not sufficient for an IoT-federation.

5.2 The Dynamics of Collaboration in Federations

The success or failure of any business platform and the value produced for its users, are the result of interacting decisions by the platform operator and by members of the different classes of users: producers, consumers and third parties [BKFA12]. The collaboration dynamic of the

[14] DLT federations can rely on automated actions and accountability without human intervention via smart contracts while the solutions in the other columns typically involve human intervention in governance.
[15] https://en.wikipedia.org/wiki/Open_API#Open_API

parties forms a main characteristic of the success-failure dynamic of federations. In this section, the archetype Causal Loop Diagram (CLD) of accidental adversaries by Braun [Brau02] is used as a starting point to study the dynamics of federations. We develop this model into a stock and flow diagram of two entities and then further into a full-blown simulation model with *n* vectors representing *n* distinct units of governance, e.g., companies.

5.2.1 The Accidental Adversaries Archetype

The accidental adversaries archetype builds on *self-benefiting actors* of two systems. The corresponding generic problem archetype can be presented in its more general form as an out-of-control system—see Figure 27, left side [Wols03]. The upper system causes an effect to the lower system while fixing its control problem. When this is applied to our problem domain of federations, we get the CLD of the right side of Figure 27. Here, a selfish action to improve own success causes harm on the other party, i.e., when Firm 1 aims for its own success it causes (delayed) harm to (the success of) Firm 2 [Wols03].

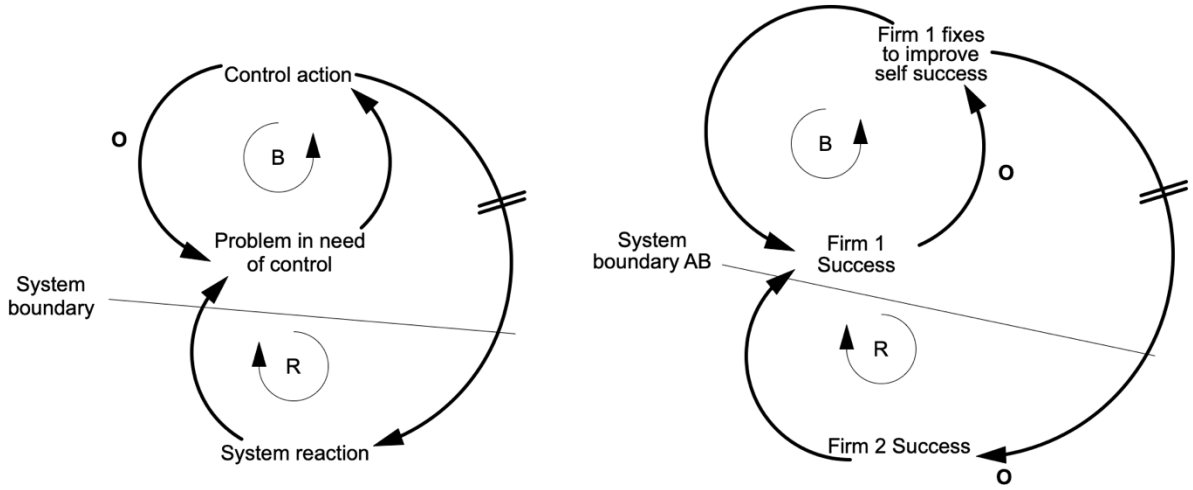

Figure 27: Left side: Generic out-of-control archetype; Right side: Semi-generic problem archetype.

Figure 28 then presents the semi-generic solution archetype [Wols03] corresponding to the semi-generic problem archetype of the right side of Figure 27. Here, unselfish action helps counteract the harm of selfish action. The red arrow denotes the action that benefits the other, i.e., altruism (unselfish action). This will counteract the harm for Firm 2's success and can uphold cooperation.

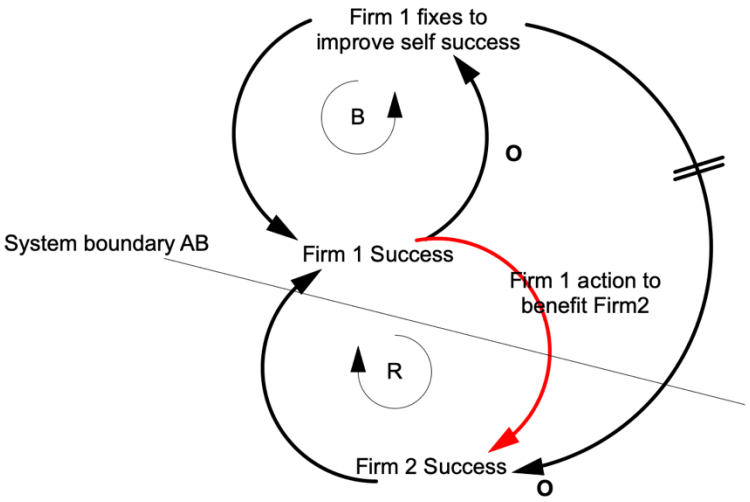

Figure 28: Semi-generic solution archetype: accidental adversaries.

Finally, Figure 29 shows the accidental adversaries solution archetype, which demonstrates how cooperation between two systems can continue. The loops B1 and B2 are the self-benefiting balancing loops of Firm 1 and Firm 2, aiming for a pre-set goal of success (e.g., monetary). At the same time, these actions produce collateral damage to the other party, which forms a Selfish Fixes Spiral loop, which is a race-to-the-bottom loop, denoted as R2. When R2 dominates, the behaviour of the model resembles the escalation archetype in reverse. Loop R1, collaboration, on the other hand, consists of benefitting the other reciprocally and selflessly, hence it is called Reciprocal Altruism, which is the key part of stabilising cooperation in the system.

When altruism is reciprocal, it is a significant component of cooperation and keeps the joint process, here federation, going [Brau02].

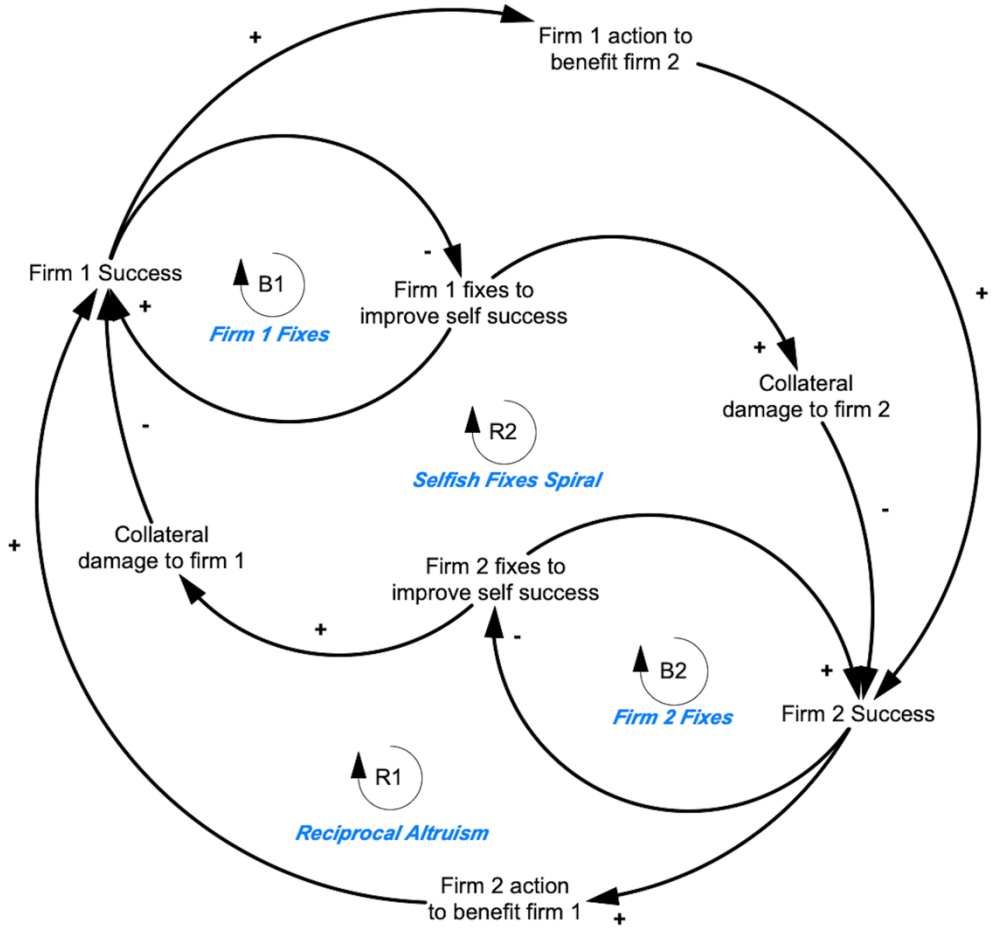

Figure 29: CLD of accidental adversaries solution archetype for two firms.

5.2.2 DLT effects on accidental adversaries archetype

The main benefits claimed by DLTs are the added transparency and integrity DLTs provide, which enable a joint view of reality for participants. DLTs can bring (federation) participants closer to a common knowledge, such as a joint view of reality [MoSh00].

The DLT transparency effect can be modelled so that it assumes all relevant information is available and verifiably correct, which can be modelled with the information security concepts of availability and integrity. Integrity is the continued property of information being unchanged from what creator of information produced, or, at the very least, detecting the changes efficiently.

Ideally information would be both authentic and it would have high integrity. DLTs are able to technically provide integrity, availability, and immutability and do so in a decentralized way. These properties can be used to implement the business and judicial level concepts of transparency, accountability, and non-repudiation, via technical and decentralized means, making these assurances very strong. Figure 30 presents a model, where the stabilising effect of transparency and integrity have been included. In the upper left corner, the *DLT transparency effect* is a multiplicative reduction on the time perception of benefits of the collaboration loop R2. In the lower left corner, the *DLT integrity effect* is a multiplicative reduction on the *Fixes effect on harm*.

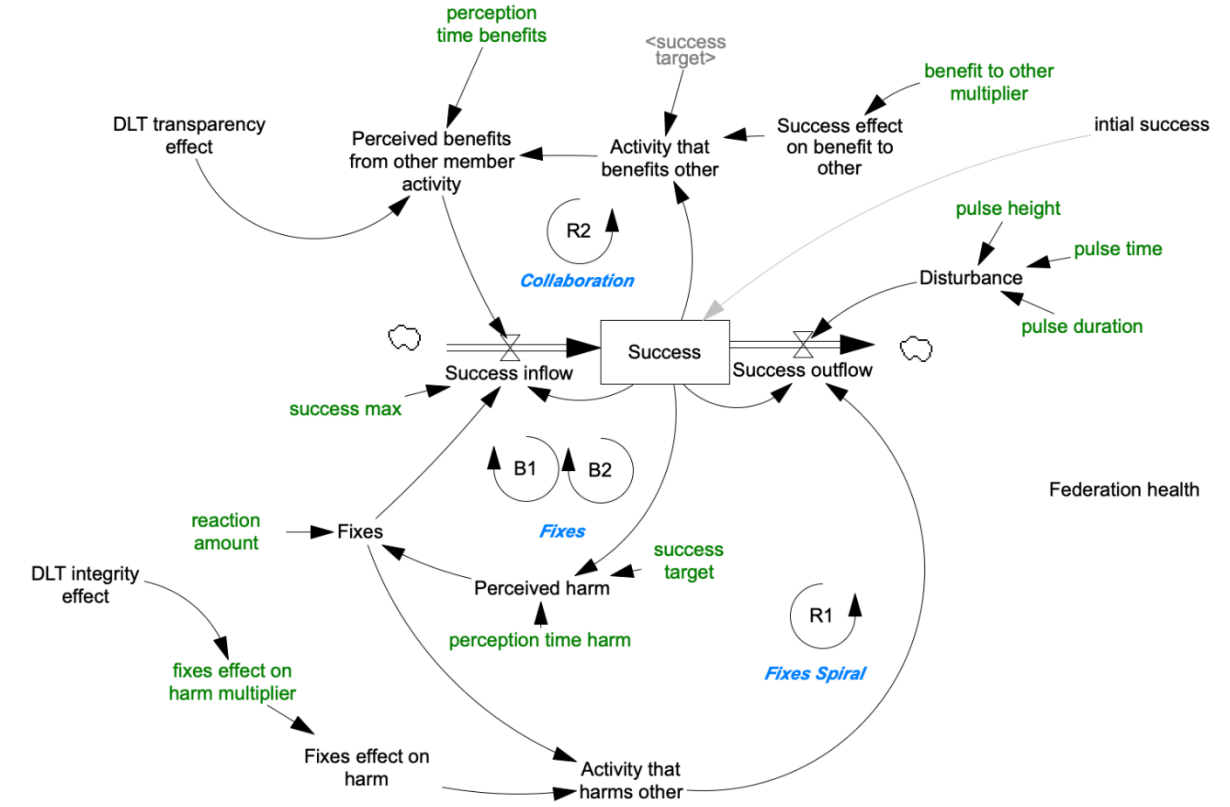

Figure 30: Simulation model with DLT effect components.

Figure 31 depicts simulation results for a sensitivity investigation with a random normal distribution of the *Success* stock when the *Fixes effect on harm multiplier* is varied. Federation stability is very sensitive to even minor variations of the collateral damage to other participants of the system. As can be deduced from the figure, an exogenous shock is applied (to firm 1) at time 5 (of magnitude 40) and lasts for 5 time units, stopping at time 10. We can see that *Success* is very sensitive to this parameter changing. Because of the shock and depending on the parameters and chance (i.e., for different runs) the federation may end-up either healthy and sustainable, or collapsing. The graph shows that in more than 75% of the cases (within the green region), the federation is sustained. However, there are cases beyond these 75% that may end up either sustained or collapsed. This indicates that the side-effect harm of selfish self-benefiting actors to other members is an important determination for the whole federation. DLT integrity, transparency and immutability properties can affect this multiplier.

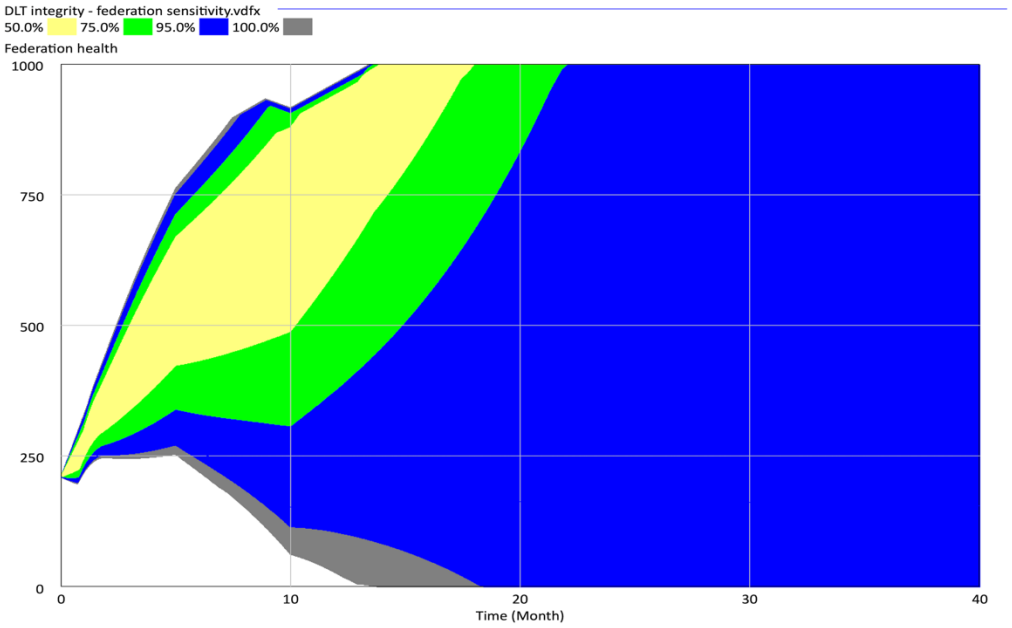

Figure 31: Sensitivity simulation investigation over stable federations.

Table 33 below presents the main DLT effect components, how they affect the platform and how the DLT implements them.

Table 33: DLT effect factors on the business platform [Sofi20, S.4].

DLT effect factor name	**Description of effect of factor to platform**	**Description of technical implementation in DLT supported IoT -federation**
Integrity	Prevents accidental or malicious modification of data.	Consensus algorithm guarantees multiple synchronized copies (of the “database”).
Transparency	Everyone sees each relevant transaction. Transactions are common knowledge to every member.	(DLT integrity and DLT availability) of (DLT integrity and DLT availability)[16] .
Availability	Information stays available against accidental or malicious Denial of Service attack by any party.	Massive number of multiple parallel copies which are guaranteed to be in sync via consensus algorithm.
Immutable full history	Reputation solidified.	DLT is write only. A Key component of the integrity of the “database.”

Table 34 below lists the beneficial governance effects which can be implemented with basic DLT factors and are part of the 2nd tier effects.

[16] This recursive infinite transparency mimics common knowledge [MoSh00].

Table 34: Second tier business and judicial level emergent ledger properties based on the (primary technical) DLT effect factors in Table 33

Second tier DLT effect factor	Description of effect of factor to platform	Description of technical implementation in DLT supported IoT-federation
Accountability	Information about conformance to joint rules is available to every member. Non-conformance is punished while conformance is rewarded.	Automatic recording of every business action to DLT. Automated execution of punishments and rewards via smart contracts.
Non-repudiation [ZhGo96]	Actions can be efficiently proven.	Non-repudiation can support accountability. Automated punishments and rewards are possible via smart contracts.
Decentralisation	Decision making authority is well distributed, and resists centralisation.	Strongly decoupled consensus algorithms from business governance.

5.2.3 Simulation model and general results

The Simulation model was developed from the two companies' semi-generic solution archetype presented in Section 5.2.1. 82Figure 32 below depicts this model version, which also adds a *DLT effect* parameter. The *DLT effect* is able to reduce the decrease in *Firm 1 success* resulting from *Firm 2 Fixes* loop, in which Firm 2 aims to improve its own state. The same is true for the decrease in *Firm 2 success* resulting from Firm 1's selfish fixing action: it too can be decreased via the *DLT effect*.

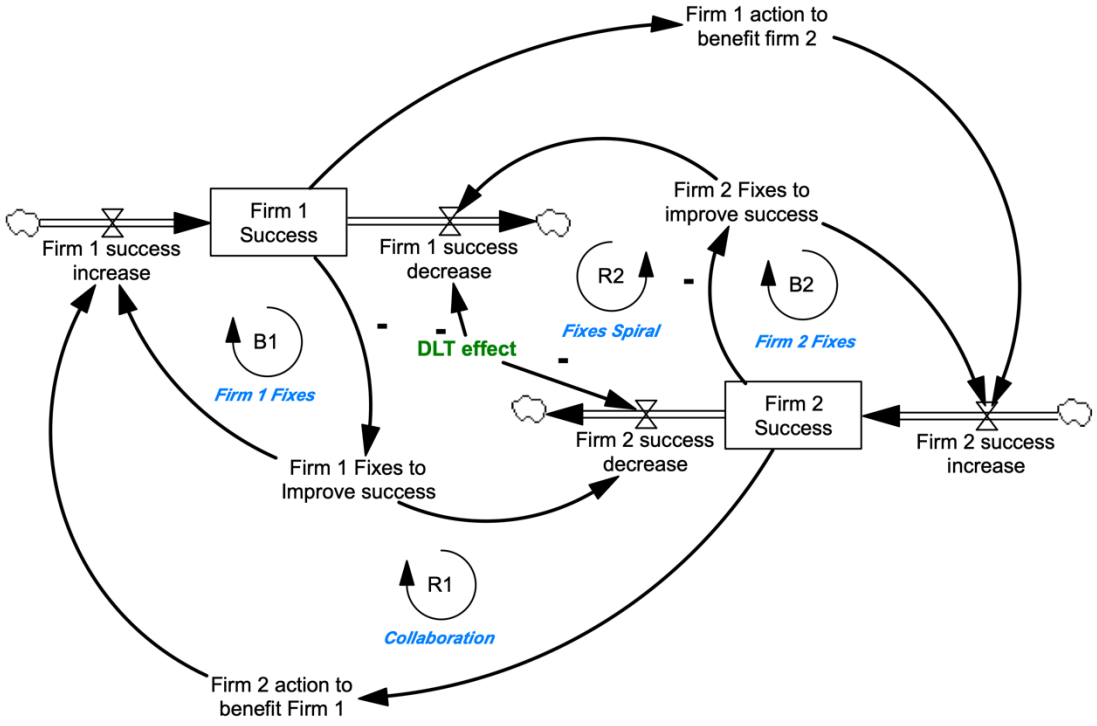

Figure 32: Stock and Flow model of the two firms accidental adversaries semi-generic solution archetype.

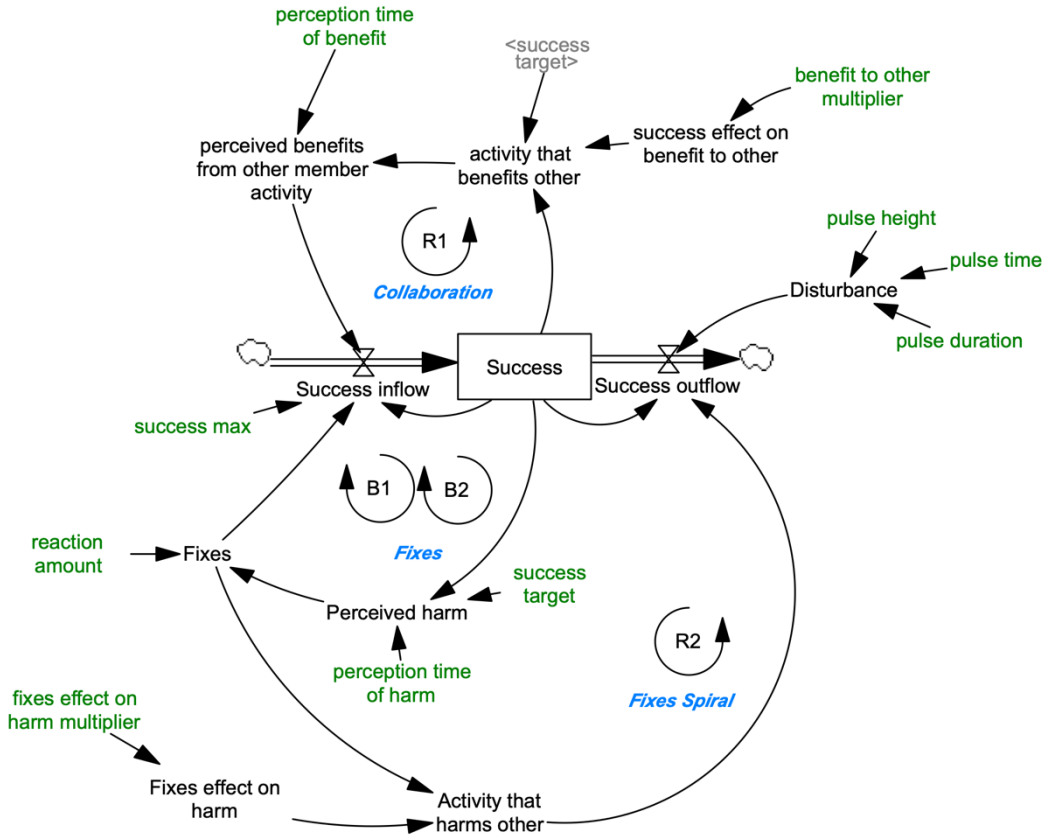

Figure 33: Semi-generic accidental adversaries archetype simulation model of n companies with cooperation and other control parameters (enables simulation of DLT effects on cooperation).

The simulation model of the stock and flow diagram depicted in Figure 33 allows for two companies to be modelled in terms of self benefiting fixes of feedback loops B1 and B2. They cause collateral damage in Fixes Spiral loop named R2. Collaboration loop R1 corresponds to the reciprocal altruism, R1 in Figure 33 above. The green parameters are model behaviour parameters affecting the dynamic behaviour of the simulation. The simulation results shown in Figure 34 were obtained with this model. The disturbance variable on the upper right corner models an external error shock that affects the system.

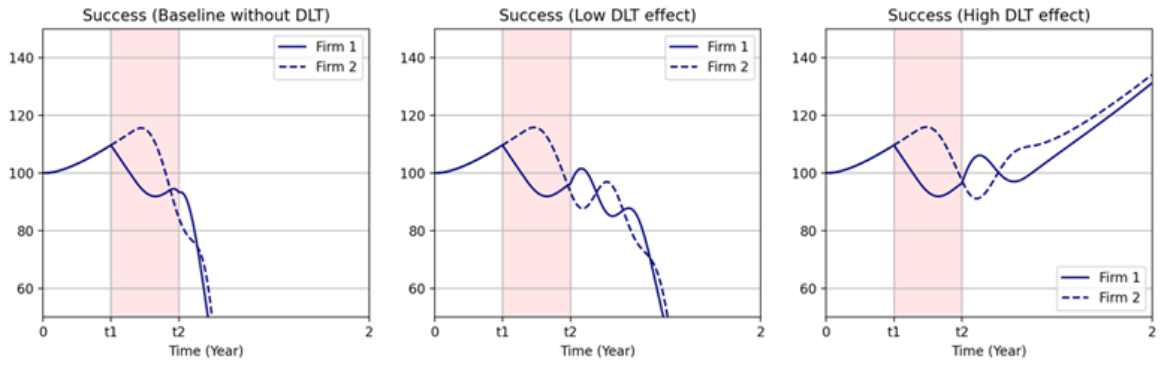

Figure 34: Simulation scenarios with no DLT, Low DLT effect, and High DLT effect; the red bar denotes the timing of an exogenous disturbance pulse.

Table 35 shows the baseline value for the key parameters of our semi-generic accidental adversaries archetype simulation model. The fourth column ("Adding the DLT has effect") describes how the DLT can affect the parameter—what is the direction and type of the change.

Table 35: Parameters of the accidental adversaries archetype simulation model in Figure 33

	Parameter	**Value** (baseline)	**Description**	**Adding DLT has effect**
1	benefit to other multiplier	3	How much each firm obtains value from other firm's collaborative actions	
2	fixes effect on harm multiplier	2	How much harm to other is caused by each fix.	Goes down if players are honest since harm in this model is unintentional collateral damage to the other party.
3	perception time of benefit	2 (months)	How quickly each firm perceives benefits from the other firm's actions	Delay goes down as DLT can automate perception triggers.
4	perception time of harm	2 (months)	How quickly each firm perceives harm (unintended effects) of the other firm's actions.	Delay goes down as DLT can automate perception triggers.
5	pulse duration	5 (months)	Duration of external disturbance	Average pulse duration goes down as per *transparency.*
6	pulse height	10	Magnitude of external disturbance	Average pulse height goes down as per *integrity.*
7	pulse time	5 (months)	Start time of external disturbance	Number of pulses on average goes down as per interaction being closer to common knowledge.
8	reaction amount	2	How much each firm reacts to deviations from success target by using fixes	
9	success max	200	Maximum success of each firm	
10	success target	100	Target for each firms' success	

The main result from the simulation is that the DLT in DLT-based federations can increase the likelihood that the collaboration within the federation continues. As shown in Figure 34 above, the effect of the DLT needs to be sufficiently high, otherwise the long-term behaviour does not differ from the case without a DLT.

The "DLT effect" consists of different factors affecting different parts of the dynamics of the cooperation model. Table 35 lists the components, Figure 30 shows the effective components in the simulation model, and Figure 31 provides a sensitivity analysis with respect to the "DLT effect" parameter.

5.3 Platform adoption model

This section applies the general platform adoption model, which can take into account same-side and cross-side network effects, to the DLT-based federation.

Digital platforms can be used to mediate transactions between different actor groups in an ecosystem. Depending on their role in the transactions, actors can be classified into different platform sides such as “end users,” “service providers,” “application developers,” etc., which together form a multi-sided market. In order to understand platform adoption in this setting, it is crucial to understand how network effects influence the value creation in a multi-sided market.

5.3.1 DLT effects on platform adoption

The inherent ability of DLTs to handle both data and payments reduces the complexity of data exchanges, including digital monetary transactions, which involve multiple players [ItFg19], [Woo18]. Without the framework of open DLT-federation, the cooperation and interoperability between separate IoT platforms requires modifying the legacy systems or building a separate integrated platform [TuZC16]. Moreover, utilising only standardized APIs to build cooperation via direct business transactions between partners has some downsides. In this case, players must bear the added risks alone. Also, they need to have access to resources and competences to safely negotiate an agreement. This makes the process hard to automate and quite costly, both in terms of time and resources [Remi20].

The effect of the SOFIE DLT-federation on either direct or cross-side effects is vital. In the case of direct network effects, the heterogeneous IoT data platforms require us to combine IoT data into rich operational intelligence, facilitating improved value creation. Thus, an IoT data platform entrant can be able to connect to other players or, if authorized, it can have access to the shared data resources in the form of a ledger by using the same protocol (e.g., the SOFIE framework). By reaching critical mass in DLT-federation, the adoption of the SOFIE framework would be nearly ubiquitous [Curr19]. Likewise, each additional IoT platform brings the new IoT data and new resources and adds value to the network. Accordingly, the new third parties joining the federation bring new services and resources to make data smarter by storing, analysing and deploying. The more heterogeneous IoT data platforms, the more third-party services and resources. Also, the more created information and IoT products, the more they attract new consumers with new needs and desires, which leads emerging new IoT data platforms to join the federation.

5.3.2 Simulation model and results

The platform model presented in Figure 35 is capable of presenting cross-side network effects from one market side to another. Direct network effects are added as dotted lines. In a cross-side network effect Potential Adopters transforming into Adopters on side A, add value to sides B and C. This will increase to the adoption on sides B and C, which is the actual cross-side network effect.

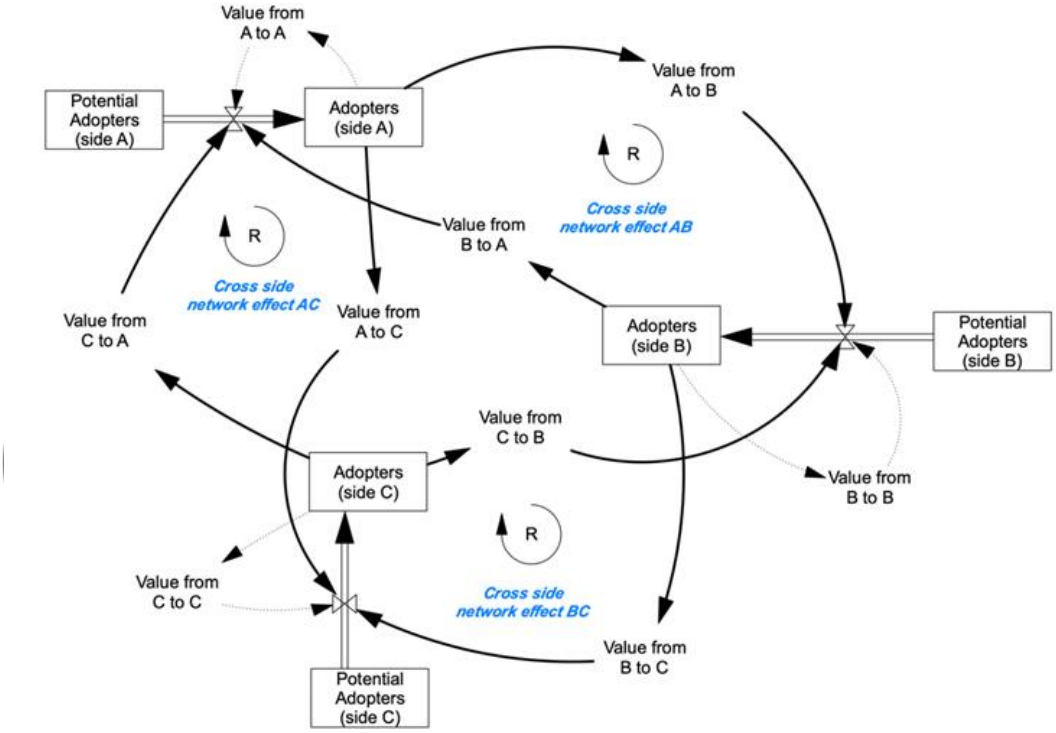

Figure 35: Overview of the platform model with three different platform sides and cross-side network effects.

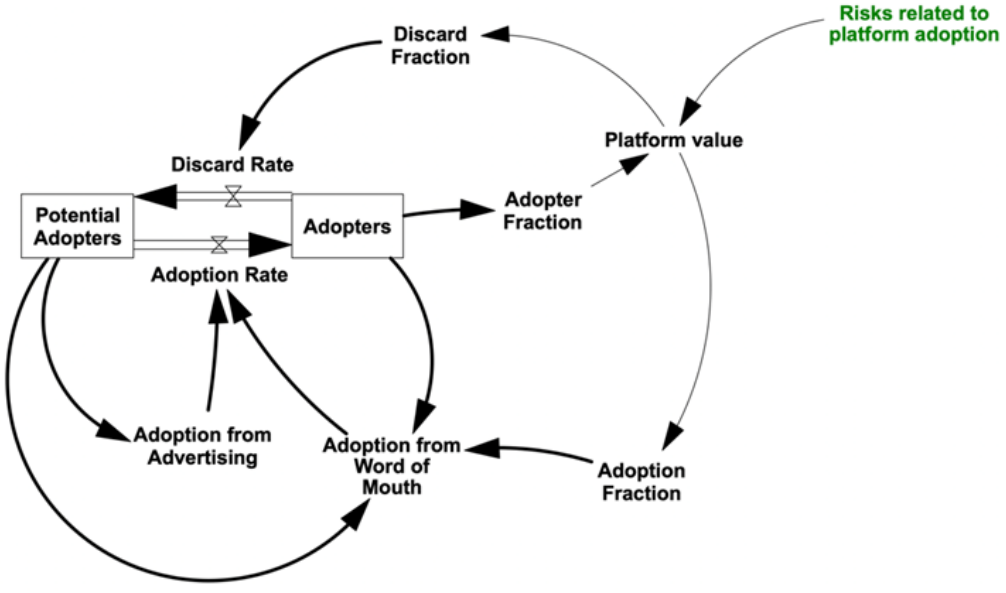

Figure 36: Platform simulation model diagram with n market sides.

The simulation model in Figure 36 is a modified version of the model by Ruutu, Casey and Kotovirta [RuCK17]. Here, the platform value is calculated based on the cross-side network effects as well as a risk factor (high risks reduce the value obtained from the platform).

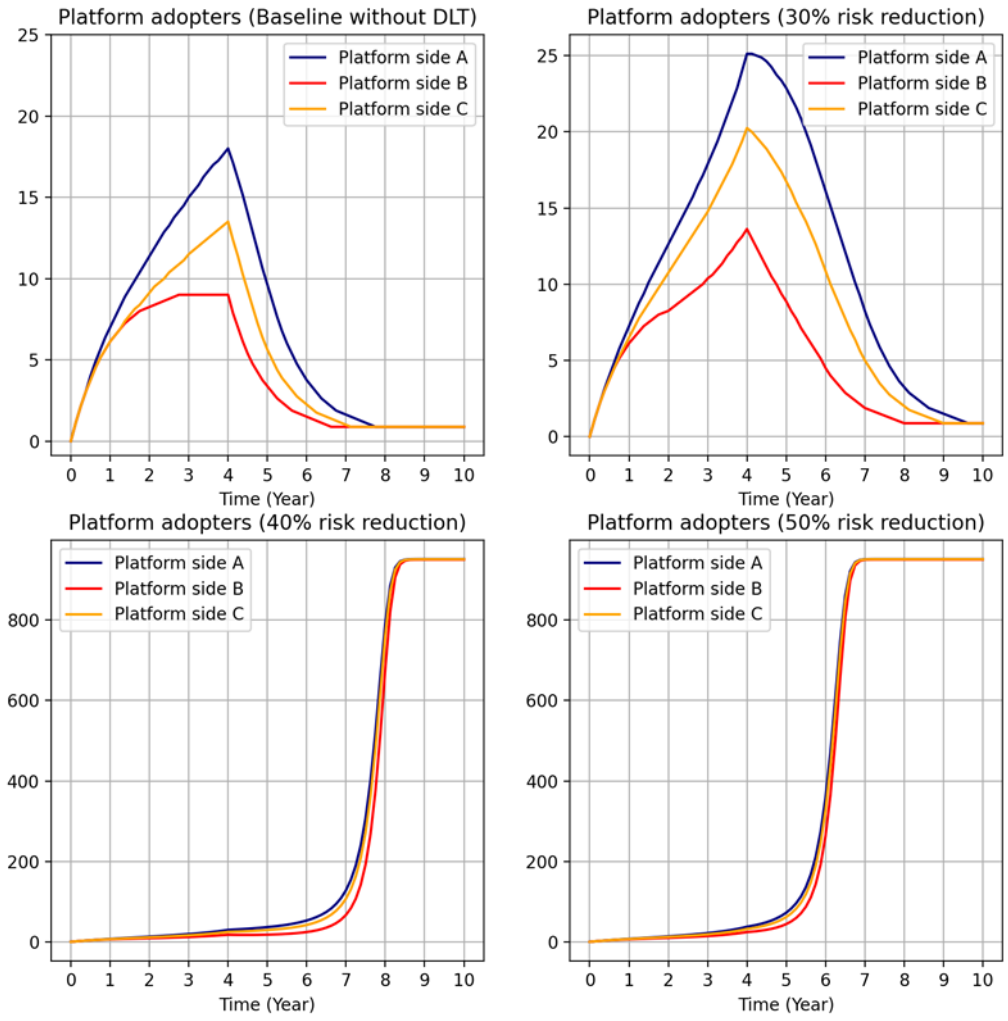

Figure 37: Platform model simulation results.

In the simulation scenarios, we examined how different values of the risk factor influence the adoption dynamics of the platform. In Figure 35, the adopters of three platform sides (A, B, C) are shown. In the simulations we assume that the value to the different platform sides varies slightly due to differences between the platform sides. Hence, the number of adopters on each side of the platform has slightly different values. (The Y axis is the numbers of adopters and the maximum possible in the simulation is 1000.) The *baseline* simulation scenario corresponds to a situation without a DLT. In this scenario, there are high risks associated with adopting the platform. Platform adoption grows initially, but after exogenous support ceases (at time=4), the number of adopters starts to decrease. This reflects a situation in which a critical mass of adopters is not achieved.

The same type of behaviour is observed in the simulation scenario with a low (30%) amount of risk reduction due to DLT, even though the number of adopters initially grows more than in the baseline scenario. However, when the risk reduction due to DLT is sufficiently high (40% and 50% in the scenarios presented in the Figure 37), a critical mass of adopters is achieved. The platform adoption follows an S-shaped curve, eventually saturated by the limited number of

potential adopters. To conclude, the generic results of the platform model, a critical mass (and sustained growth of adoption) can be achieved more easily with a DLT because DLT reduces risks and thus improves the value of the platform to the different actor groups.

5.3.3 Applying the platform model to the food chain use case

The generic platform adoption model can be used to examine in more detail the SOFIE Food Supply Chain use case. The cross-side network effect for consumers is calculated based on the number of suppliers, and the cross-side network effect for suppliers is calculated based on the number of consumers. Furthermore, the value to consumers depends on the cross-side network effect and the product quality. In the simulations, a multiplicative value function is assumed, meaning that the value to consumers is zero if either the cross-side network effect or the product quality is zero. In the baseline simulation scenario, product quality is set to 1. The effect of DLTs on product quality is assumed to be positive and two different scenarios with different values are examined (small increase: product quality = 1.1 and large increase: product quality = 1.5).

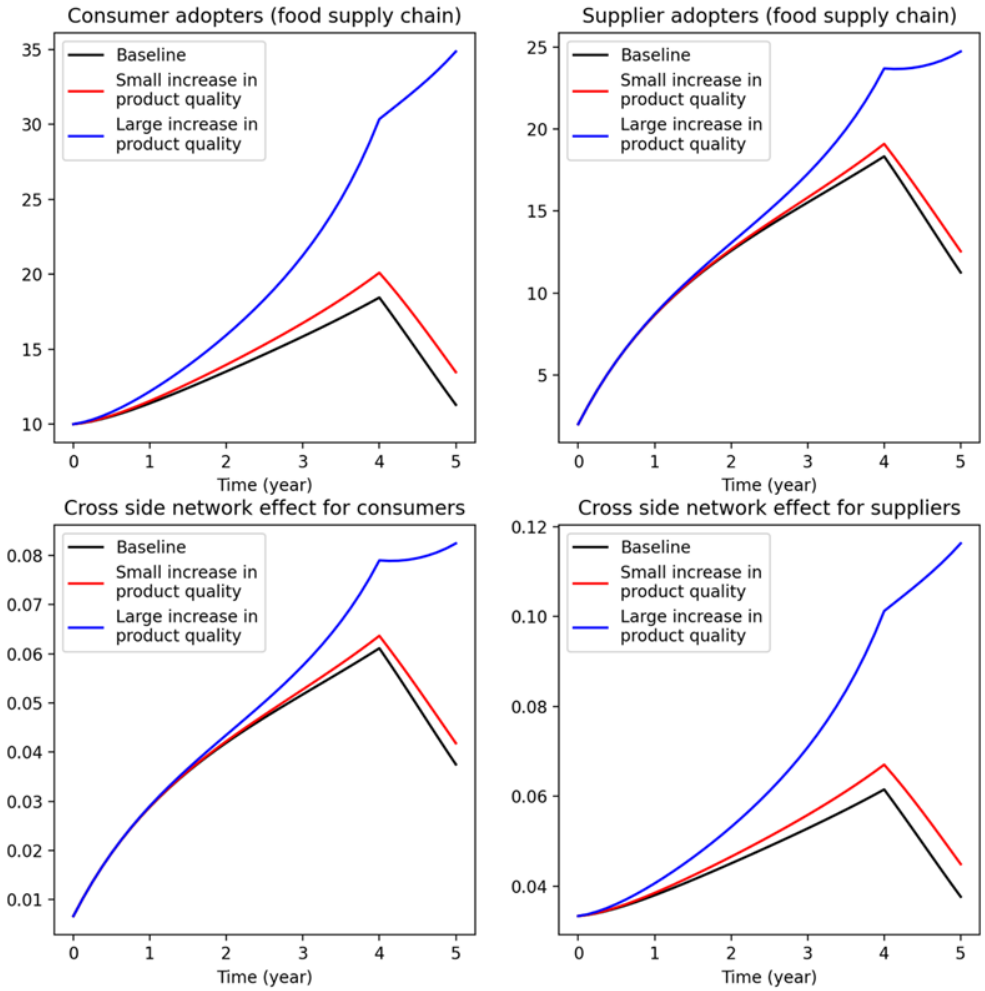

Figure 38: Adoption model simulation runs.

In Figure 38 above, the two topmost graphs show that large increases in product quality lead to a sustained increase of platform adoption of both consumers and adopters. The lower two pictures show a sustained cross-side network effect for both groups, which is not happening without a large quality increase.

5.4 Interpretation of results from dynamics of cooperation and platform adoption for SOFIE use cases

This section applies the generic results from the previous sections to the SOFIE use cases. Here, the effect of DLTs is primarily a risk control instrument. Table 36 below presents the generic effects of risk reduction of utilising DLTs in federations.

Table 36: The generic effects to risks of a DLT to federation (sustainability).

Component of DLT effect (enforced property)	**Effect to risk for the whole federation**	**Mechanism of effect**
Integrity	Reduce	Enables reliable reputation management
Transparency	Reduce	Enables widespread reputation management
Availability	Reduce	Main component of transparency
Accountability	Reduce risk in case of automation of accountability via smart contracts	Punishment in case of deviation, reward in case of compliance

5.4.1 Food Supply Chain

The results depicted in Figure 37 of our platform simulation model in Figure 35, Section 5.3, reveal that if risk reduction is sufficiently high, platform (federation) discarding among adopters is minor for both platform sides i.e., among consumers and suppliers (or among all constituent platforms in case of federation of platforms). Hence, the platform/federation is able to gather a critical mass. Gathering critical mass for the platforms and federation in the FSC pilot, is interpreted as a successful effect of the DLT.

This produces a risk reduction with respect to risk 1, product growing conditions (SynField IoT platform); risk 6, transferred products (Transportation IoT platform); and risk 4, storage conditions of products in warehouses (Aberon IoT platform). By successfully managing the aforementioned risks for the three platforms, the DLT is effective in preserving trust within and across platform actors (e.g., risk 5), supporting collaboration between suppliers (e.g., risk 8), verifying product quality and safety to consumers (e.g., risk 1), and securing exchanged information within and across platform actors (e.g., risk 5).

The main results of the cooperation dynamics in federations, depicted in the simulation runs of Figure 34, Section 5.2, indicate that in the presence of a sufficiently high DLT effect, cooperation is able to continue even after an exogenous shock. With the exogenous shock manifesting, others are harmed by the actions of one platform. For example, risk 1 in Table 37 may occur as a result of the transportation company making an error with temperature conditions.

With an early detection, this risk is prevented from affecting the consumers as the warehouse is able to become aware of the problem in time. Via the reputation gathered in the federation DLT, the federation members are able to help reduce the effects of such mistakes and improve the quality of the transportation. If the risk is realised all the way to the consumer, compensation and other corrective actions may be automated and efficiently targeted. This all requires that *efficient and effective cooperation* is facilitated and realised inside the federation as *activity benefitting the other* business directly and happens with enough intensity between the federation members. See R1 collaboration loop of Figure 29, Figure 32, and Figure 33 above.

Table 37: Risks of the pilot with DLT effect and member accountability.

	Risk description	Who is directly at risk?	Effect of risk being realised	DLT effect to risk	Member account-ability effect to risk
1	Consumers receiving low quality products	Consumers	Consumers have adverse health effects or are unhappy with products	Detect, reduce via reputation	Reduce risk via enforcing quality
2	Suppliers rejecting responsibility for low quality products	Consumers and super-markets	Consumers not compensated. Supermarkets not compensated.	Reduce via reputation and accountability	Reduce
3	Suppliers concealing ingredients and production process	Consumers	Consumers have health effects	Reduce via reputation Reduce via accountability	Reduce via contract enforcement
4	Shortage of products of (high) demand by consumers	Consumers	Consumers turn to alternative markets	More transparent cooperation reduces	Reduces via explicit SLAs.
5	Monetary fraud incidents (suppliers not being paid or being paid less than agreed; consumers paying more than agreed)	Consumers, suppliers	Consumers and supplier's financial loss	Reduce via reputation Reduce via accountability	Reduce via statutory liability
6	Consumers receiving different products than ordered	Consumers	Supplier financial loss	Reduce via accountability Reduce via reputation	Reduce via contract enforcement
7	Suppliers sending different products than ordered	Consumers	Delay, corrective action required	Reduce if orders also through DLT	Reduce & tool for contract enforcement
8	Suppliers claiming ownership of products offered by other suppliers	Suppliers, Consumers	Supplier financial loss Logistic confusion leading to non-availability Consumers mistaken about origin of products	Reduce via reputation Reduce via accountability	Reduce via statutory liability Reduce via SLA and other contractual liability
9	Confidentiality and privacy of orders for both consumers and suppliers	Consumers, suppliers	Privacy diminishing because of wide availability and transparency of reputation	Increases as a function of transparency	Reputation of the platform affected Statutory accountability

5.4.2 Context Aware Mobile Gaming

This pilot utilises both Ethereum and Hyperledger Fabric ledgers. The first ledger is a common public ledger and the assets there are controlled by the company siring the ecosystem. The Hyperledger instance is controlled by the consortium. The ecosystem needs to be initialised or sired by some party (investors). This is assumed to happen by the actions of the gaming company. The consortium and siring gaming company must stay in the federation together in order for the ecosystem to function meaningfully. The pilot has a function of being able to move digital assets from one ledger to another. This enables asset ownership of virtual gaming assets across games and via the consortium ledger also across gaming companies. Having a mix of virtual and physical assets participating in the game enables mixing virtual and physical gaming assets and their interactions.

The main cooperation dynamics are illustrated in Figure 34, Section 5.2.3, where it is demonstrated that with a sufficiently strong DLT effect, beacon manufacturers (cross-side market providers) stay within the federated business platform. This will control risks 3 and 4, defined in Table 38 below. Risk 5 is mostly a usability issue, but the DLT effect may be used to counter some of the non-connectivity produced by other reasons, such as fearing the information security and privacy risks of connecting the device. Risks 1 and 2 are mostly reputational and decrease the cooperation in the federation. Stronger DLT effects will reduce the impact.

The primary result from the platform simulation model in Figure 37, Section 5.3.3, reveals that if risk reduction is sufficiently high, platform discarding among beacons is minor for both platform sides, i.e., among both consumers and suppliers. Hence, the platform/federation is able to gather critical mass. Gathering critical mass for the constituent platforms is interpreted as a successful effect of the DLT. This produces a risk reduction with respect to risks in Table 38.

Table 38: Risks of the gaming pilot with DLT effect and member accountability effect.

	Risk description	Who is directly at risk?	Effect of risk being realised	DLT effect to risk	Member account-ability effect to risk
1	Federation founded, but not surviving	Investors	Initial investment lost, no federation	No effect on risk with closed DLT, fully open DLT would offer accelerated free reputation	-
2	Federation not attractive enough	Federation	No self-sustaining federation	Transparency and integrity increase, which reduces risk	-
3	Federation setup in a monopolistic value sharing model (rent seeking)	Players, cross-side market providers	Player and beacon provider revenue reduced	Transparency decreases the risk; immutability of rules has potential to decrease risk	Potentially reduced via transparent and fixed ecosystem rules
4	Beacon owning not attractive enough	Cross-side market providers	No profit for beacon providers	Transparency decreases the risk; immutability of rules has potential to decrease risk	Reduces via transparent and fixed ecosystem rules
5	Monetary losses due to players losing things, or theft	Players	Financial loss	Theft becomes more traceable, i.e., boost to accountability, lost items can be found	Reduction of theft, reversal of theft
6	Beacons not connected	Players	No profit for players	Transparency and integrity vital in helping build usability	Reduces if mandate to connect

5.4.3 Decentralized Energy Flexibility Marketplace

Electric vehicle (EV) fleet managers interested in demand response campaign benefits and electric power Distribution System Operators (DSOs) requesting flexibility to balance the grid, are able to interact through a decentralized marketplace. The marketplace is governed by a set of business rules. Part of these rules are smart contracts, which facilitate negotiations and collaboration between demand (EVs) and supply (DSOs) (from the energy provision perspective, reversed roles from the demand of flexibility perspective). Technically, the marketplace assists achieving flexibility management of distributed energy resources (suppliers and consumers).

The DLT-federation reduces potential fraud by the parties by recording offers (of desired and promised flexibility) immutably in ledgers (exact amount of energy, specific time intervals, specific locations). This leads to fair competition on the platform and FMs can deal freely with local producers through the secure network. Related risks are listed in Table 39. The immutability and transparency properties of DLTs improves trust among the parties and avoid favouritism (Table 39). In fact, the results of our platform simulation model in Figure 37, Section 5.3.3, reveal that the transparency and availability of the DLT-federation platform decrease the risks and make it acceptable among producers and consumers. The risk of monopoly is decreased by providing opportunities for local electricity producers to join the platform conveniently without paying membership fees (openness property) and the low cost of DLT infrastructure. In this decentralized marketplace, trade occurs through smart contracts (Table 39). The smart contract enforces all contractual obligations for consumers and producers

without any deviations and deletes the risk of tampering (accountability property). Also, the search cost and transaction cost are reduced in the business platform. Moreover, there is an incentives mechanism to make it more profitable to join the business platform for participants. All these facilitate the goal of the platform to achieve critical mass and sustainability.

Table 39: Risks of the pilot with DLT effect and member accountability effect.

	Risk description	Who is directly at risk?	Effect of risk being realised	DLT effect to risk	Member accountability effect to risk
1	Bid rigging by DSO	Producers, electricity retailers	Financial loss	Reduce the risk by immutability and transparency	Reduce the risk by accountability provided by DLT
2	Bid rigging by producer	Consumer	Financial loss	Reduce the risk by immutability and transparency	Reduced the risk by accountability in smart contract
3	Bid rigging by consumer	Producer	Financial loss	Reduce the risk by immutability and transparency	Reduced the risk by accountability in smart contract
4	Digital currency volatility	Consumers	Suffer loss	-	-
5	Risk of monopoly	Consumers, producers	Inflation, declining product quality, price fixing	Reduce by decentralized property	Reduce
6	Supplier avoids obeying contract	Consumers	Unreliable trade, suffer loss	Detect, reduce	Reduce, punishment rules
7	Consumer avoids obeying contract	DSOs	Reverse power flow	Detect, reduce	Reduce punishment rules

5.5 Conclusions for Business Platform Analysis with System Dynamics

DLTs can make the adoption of a federated platform faster because they reduce risks. They can also help sustain cooperation in federations because they help parties to perceive the benefits faster and because they decrease the collateral damage from one party's actions to improve its position to others. Adequate or wide participation in the federation and stable cooperation are the basic building blocks for a successful federation. These are facilitated by the DLT core properties of providing integrity, transparency, and non-repudiation. A threshold of risk reduction through the DLT effect needs to be reached in order for the benefits to manifest.

System Dynamics is a suitable tool to study such thresholds and analyse them qualitatively and quantitatively. In particular, the “accidental adversaries archetype” seems as an appropriate basis for such investigations, however, additional adoption oriented models, as we have used above, are also needed for a more complete investigation. Both need appropriate parameterizations, simulation runs with many parameters and sensitivity analysis for deeper exploration.

Based on an analysis of the pilot cases the best possibilities for risk reduction seem to be in the Food Supply Chain and the Context-Aware Mobile Gaming pilots. Both benefit from an open consortium, where trustworthy transparency would seem to be one of the important enabling factors for each business case as a whole. In the Food Supply Chain use case, transparency

would directly serve to make the quality of the food better. Furthermore, in this case specifically, the ability to create several levels of “transparency” (e.g., through separate channels) may be important to reduce risk number 9 from Table 37 (confidentiality of orders for both consumers and suppliers and in the end privacy for individuals). In the gaming use case, it would serve to make the consortium and its value sharing more transparent and more accountable, leading to higher adoption by both game players and partner companies in the ecosystem (e.g., advertisers and challenge contributors).

6 Conclusion

This deliverable contains the final results from WP4's evaluation work. In summary, the evaluation results contained in the current and in the previous two evaluation deliverables D4.3 and D4.4 include the following:

- D4.3 (First Architecture and System Evaluation Report, initial submission June 2019, revised December 2019): This deliverable contains architecture KPIs and pilot-specific system performance KPIs, along with their target values. Additionally, it contains the initial evaluation results for the SOFIE framework components, as well as a more detailed evaluation of IoT resource access solutions, which considers the problem of authentication and authorization in more depth and with more alternatives compared to the initial evaluation of the Identification, Authentication, and Authorization (IAA) and the Privacy and Data Sovereignty (PDS) components. This deliverable also contained the gains, in terms of reduced cost and transaction delay, and the tradeoffs involving transparency and trust, when multiple ledgers are interconnected through the Interledger component. Finally, the deliverable considers pilot inspired evaluation scenarios, abstracting the pilot use cases to consider the various tradeoffs of many potential alternative design decisions and their impact.
- D4.4 (Second Architecture and System Evaluation Report, submitted April 2020): This deliverable contains the results from the second evaluation cycle. This includes SOFIE framework component evaluation results, including the interconnection of Hyperledger Fabric and the public Ethereum testnet (Interledger), Indy-based Verifiable Credentials (PDS), and authorization based on JSON Web Tokens (IAA). Additionally, we investigate decentralized interledger gateway architectures. For the pilot emulation scenarios, we investigate the use of multiple ledgers and interledger technology, as well as using arbitrary private storages and storing hashes in a public ledger (Food Supply Chain), performance in terms of transaction cost, response time, and throughput for the Decentralised Energy Flexibility Marketplace scenarios, verifiable credentials and OAuth 2.0 access tokens based on Ethereum ERC-721, as well as the tradeoffs involving the hash recording frequency (Decentralized Energy Data Exchange), and scenarios utilizing public Ethereum and Hyperledger Fabric for implementing various functionalities of the Context-Aware Mobile Gaming scenario. Finally, this deliverable contains the business platform evaluation based on system dynamics, focusing on pilot-based use cases and the interaction among elements and forces.
- D4.5 (Final Architecture, System, and Pilots Evaluation Report, December 2020), the current deliverable: In addition to new results on the evaluation of SOFIE's framework components and pilot-based evaluation scenarios, this deliverable also contains the joint analysis of pilot and pilot-inspired use cases emulation results. In addition to presenting side-by-side and discussing the emulation and pilot results for some system performance KPIs, we also present new emulation results that exploit traces and statistics from the actual pilots for the assessment of realistic scenarios, but at a larger scale that cannot be achieved solely by the pilot environments. Specifically, the new results consider the privacy and sensor logging and anchoring trade-offs in the context of the Food Supply Chain pilot scenarios; a large-scale evaluation of the extended functionality of the marketplace component which allows multiple winners for improved effectiveness in absorbing Reverse Power Flows, utilizing traces from the Decentralized Energy Flexibility Marketplace pilot; smart meter traces for the evaluation of local differential privacy mechanisms, which are part of the Privacy and Data Sovereignty (PDS) component; and the evaluation of a new scenario involving an open advertising ecosystem for DLT-assisted mobile gaming. Finally, the business evaluation assessed how DLTs can reduce federation risks, while they can also bring down the detection time of benefits and decrease collateral damage from one party's actions to others.

The evaluation results have demonstrated the gains from combining different ledgers, with different properties and performance, and the tradeoffs in terms of performance, decentralised trust, transparency, and privacy. Furthermore, the evaluation has assessed, in a qualitative and quantitative manner, the benefits of features in the SOFIE architecture and framework to promote open business platforms and federation to enable seamless collaboration across multiple entities with a distinct administration.

In addition to achieving the evaluation targets and answering the corresponding evaluation questions set forth in the SOFIE's project description of work, WP4's evaluation work has also identified directions for further research, some of which have already started to be pursued by consortium partners. We identify a few of these directions below.

Future directions:

- Less on-chain and more off-chain: In addition to reducing the aggregate transaction cost and transaction delay, moving more functionality off-chain would enable asynchronous peer-to-peer transactions between entities that are disconnected from the Internet, hence do not require accessing a blockchain, including simply reading data from a blockchain. Domains that can benefit from such a capability include the further shift towards the Multiaccess Edge Computing (MEC) paradigm in beyond 5G systems in order to support ultra-reliable and low latency services in a decentralized and trusted manner.
- Application to new domains beyond those of the SOFIE pilots. The evaluation of SOFIE's components and mechanisms has allowed us to identify the fundamental services and interactions that are independent of the particular application domain. Investigating how SOFIE's solutions can be applied to different domains is a fruitful direction. Specifically, the application of decentralized and self-sovereign identifiers to enable authorized and trusted services can be applied to domains such as decentralized file sharing and asset tracking and configuration in manufacturing systems.
- Trusted and efficient energy flexibility: Our investigations of the energy flexibility marketplace scenarios has highlighted the importance of blockchains and smart contracts to achieve trust in an open business environment. This is even more central for the emerging peer-to-peer energy communities. However, the efficiency of such systems also depends on off-chain procedures and mechanisms. How to exploit such off-chain procedures to enhance efficiency, while not losing the transparency and trust offered by smart contracts executed on DLTs is a direction that can yield further performance gains, without compromising security and trust.
- DLT extensions for legacy systems: Having investigated solutions pertaining to the use of distributed ledger technology to provide a new level of trust, integrity, and immutability guarantees, and at the same time acknowledging the vast volume of deployed legacy software, it is of paramount importance to provide generic tools and components to extend legacy software with DLT functionality. Such tools could be provided as plug-ins and extensions to database systems as well as to proprietary data storage architectures, and could enhance them by diverse functionalities, including immutability guarantees, validation across diverse business entities, trusted timestamping services, and off-chain trusted transactions.

7 References

[AEN19] E. Arzoglou, T. Elo, and P. Nikander, “The Case of iOS and Android: Applying System Dynamics to Digital Business Platforms,” Proc. ICCS 2019. In: J. Rodrigues et al. (eds) *Computational Science – ICCS 2019*, Lecture Notes in Computer Science, Vol. 11540, Springer, Cham, 2019.

[AJM14] R. Azarderakhsh, K.U. Järvinen, and M. Mozaffari-Kermani, “Efficient Algorithm and Architecture for Elliptic Curve Cryptography for Extremely Constrained Secure Applications,” *IEEE Transactions on Circuits and Systems*, Vol. 61, No. 4, pp. 1144–1155, 2014.

[Ber06] D.J. Bernstein, “Curve25519: New Diffie-Hellman Speed Records,” Proc. 9th International Conference on Theory and Practice of Public-Key Cryptography (PKC 2006), pp. 207–228, New York, NY, USA, April 2006.

[Ber+12] D.J. Bernstein et al., “High-speed high-security signatures,” *Journal of Cryptographic Engineering*, Springer, Vol. 2, No. 2, pp. 77–89, 2012.

[BKFA12] W.E. Beyeler, A. Kelic, P. Finley, M. Aamir, A. Outkin, S. Conrad, M. Mitchell, V. Vargas, “Creating Interaction Environments: Defining a Two-Sided Market Model of the Development and Dominance of Platforms,” in *Social Computing, Behavioral - Cultural Modeling and Prediction*, S.J. Yang, A.M. Greenberg, M. Endsley (eds), Lecture Notes in Computer Science, Vol. 7227, Springer, 2012. https://doi.org/10.1007/978-3-642-29047-3_38

[Brau02] W. Braun, “The System Archetypes,” in *The Systems Modelling Workbook*, 2002. Available: https://www.albany.edu/faculty/gpr/PAD724/724WebArticles/sys_archetypes.pdf

[But16] V. Buterin, “Chain Interoperability,” R3 Report, September 2016. Available online: https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf

[Cai+18] W. Cai, Z. Wang, J.B. Ernst, Z. Hong, C. Feng and V.C.M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," *IEEE Access*, Vol. 6, pp. 53019-53033, 2018.

[Cle+14] R. de Clercq, L. Uhsadel, A. Van Herrewege, and I. Verbauwhede, “Ultra low-power implementation of ECC on the ARM Cortex-M0+,” Proc. 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, June 2014.

[Curr19] J. Currier, “The Network Effects Manual: 13 Different Network Effects (and counting),” NfX, available: https://www.nfx.com/post/network-effects-manual/, accessed on 24-11-2020.

[RBM09] M. de Reuver, H. Bouwman, and I. MacInnes, “Business model dynamics: a case survey,” *Journal of theoretical and applied electronic commerce research*, Vol. 4, No. 1, pp. 1–11, April 2009.

[Del+16] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, and B. Parno, “Cinderella: Turning Shabby X.509 Certificates into Elegant Anonymous Credentials with the Magic of Verifiable Computation,” Proc. IEEE Symposium on Security and Privacy (SP), May 2016.

[Dül+15] M. Düll et al., “High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers,” in *Designs, Codes and Cryptography*, Springer, Vol. 77, No. 2-3, pp. 493-514, 2015.

[Fot+16] N. Fotiou, T. Kotsonis, G.F. Marias, G.C. Polyzos, “Access control for the Internet of Things,” Proc. ESORICS International Workshop on Secure Internet of Things, pp. 29–38, 2016.

[Fot+19] N. Fotiou, I. Pittaras, V.A. Siris, S. Voulgaris, G.C. Polyzos, “Secure IoT access at scale using blockchains and smart contracts,” Proc. 8th IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IoT-SoS), Washington DC, USA, June 2019.

[Fot+20] N. Fotiou, I. Pittaras, V.A. Siris, S. Voulgaris, G.C. Polyzos, "OAuth 2.0 authorization using blockchain-based tokens," NDSS Workshop on Decentralized IoT Systems and Security (DISS), San Diego, CA, USA, February 2020.

[FSP18] N. Fotiou, V.A. Siris, G.C. Polyzos, "Interacting with the Internet of Things using Smart Contracts and Blockchain Technologies", Proc. Security, Privacy, and Anonymity in Computation, Communication, and Storage (SpaCCS 2018), Melbourne, Australia, December 2018.

[Ger+18] S. Gerdes et al., "An architecture for authorization in constrained environments," IETF Draft, October 2018.

[GH01] J.F. Gubrium and J.A. Holstein, *Handbook of interview research: Context and method*, Sage Publications, 2001.

[Glo18] Global Platform, "TEE System Architecture v1.2," December 2018. Available online: https://globalplatform.org/specs-library/tee-system-architecture-v1-2/.

[Hag09] A. Hagiu, "Two-Sided Platforms: Product Variety and Pricing Structures," *Journal of Economics & Management Strategy*, Vol. 18, No. 4, pp. 1011–1043, 2009. Available online: https://www.onlinelibrary.wiley.com/doi/full/10.1111/j.1530-9134.2009.00236.x/

[Haa+18] J. Haapola et al., "Peer-to-Peer Energy Trading and Grid Control Communications Solutions' Feasibility Assessment Based on Key Performance Indicators," Proc. 87th IEEE Vehicular Technology Conference (VTC Spring), 2018.

[Har+12] D. Hardt et al., "The OAuth 2.0 Authorization Framework," RFC 6749, Standards Track, IETF, October 2012.

[HS13] M. Hutter and P. Schwabe, "NaCl on 8-Bit AVR Microcontrollers," Proc. International Conference on Cryptology in Africa, published in *Progress in Cryptology – AFRICACRYPT 2013*, A. Youssef, A. Nitaj, and A.E. Hassanien (eds), Lecture Notes in Computer Science, Vol. 7918, Springer, 2013.

[ItFg19] ITU-T Focus Group on Application of Distributed Ledger Technology; (FG DLT): Technical Report FG DLT D2.1 Distributed ledger technology use cases, P 73, 2019

[JBS15a] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Standards Track, IETF, May 2015.

[Kor+19] Y. Kortesniemi, D. Lagutin, T. Elo, N. Fotiou, "Improving the Privacy of IoT with Decentralised Identifiers (DIDs)," *Journal of Computer Networks and Communications*, Hindawi, 2019.

[KaHR20] Karhu, Kimmo; Heiskala, Mikko; Ritala, Paavo: Beyond the N in Network Effects: Five Type of Network Externality Functions in Platform Markets. In: SSRN Electronic Journal (2020)

[Kov+19] M. Kovatsch et al., "Web of Things (WoT) Architecture," retrieved August 2019. Available at: https://www.w3.org/TR/wot-architecture/.

[KL08] J. Katz and A.Y. Lindell, "Aggregate message authentication codes," Proc. The Cryptographers' Track at the RSA conference on Topics in Cryptology (CT-RSA), 2008.

[KM00] R. Kaplinsky and M. Morris, "A handbook for value chain research," University of Sussex, Institute of Development Studies, 2000. Available online: http://www.fao.org/fileadmin/user_upload/fisheries/docs/Value_Chain_Handbool.pdf

[KSD18] S. Kalra, R. Sanghi, and M. Dhawan, "Blockchain-based real-time cheat prevention and robustness for multi-player online games," Proc. 14th International ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT), pp. 178–190, Heraklion, Greece, December 2018.

[Lag+19] D. Lagutin, Y. Kortesniemi, N. Fotiou, V.A. Siris, "Enabling Decentralized Identifiers and Verifiable Credentials for Constrained Internet-of-Things Devices using OAuth-based Delegation," NDSS Workshop on Decentralized IoT Systems and Security (DISS), San Diego, CA, USA, February 2019.

[LBBC19] D. Lagutin, F. Bellesini, T. Bragatto, A. Cavadenti, V. Croce, Y. Kortesniemi, H.C. Leligou, Y. Oikonomidis, G.C. Polyzos, G. Raveduto, F. Santori, P. Trakadas, M. Verber, "Secure Open Federation of IoT Platforms Through Interledger Technologies – The SOFIE Approach," Proc. European Conference on Networks and Communications (EuCNC), Valencia, Spain, June 2019.

[MoSh00] S. Morris, H.S. Shin, "Approximate Common Knowledge and Co-ordination: Recent Lessons from Game Theory," *Journal of Logic, Language and Information*, vol. 6, pp. 171–190, April 1997.

[Nak08] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," White Paper, October 2008. Available online: https://bitcoin.org/bitcoin.pdf

[NE19] P. Nikander and T. Elo, "Will the data markets necessarily fail? A position paper," Proc. 30th International Telecommunications Society (ITS) European Conference, Espoo, Finland, June 2019.

[PD16] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable off-chain instant payments," White Paper, January 2016.
Available online: https://lightning.network/lightning-network-paper.pdf

[Ree19] D. Reed et al., "Decentralized Identifiers (DIDs) v0.13: Data Model and Syntaxes for Decentralized Identifiers," Draft Community Group Report, W3C, June 2019. Available online: https://w3c-ccg.github.io/did-spec/

[Remi20] Remi, Robert: Can we put our trust in a decentralized marketplace? URL https://www.ericsson.com/en/blog/2020/3/decentralized-marketplace-cloud. - abgerufen am 2020-11-22. — Ericsson.com. — Last Modified: 2020-09-05T16:51:28+00:00

[RT06] J.-C. Rochet and J. Tirole, "Two-Sided Markets: A Progress Report," The RAND Journal of Economics, Vol. 37, No. 3, pp. 645-667, 2006.

[RuCK17] S. Ruutu, T. Casey, V. Kotovirta, "Development and Competition of Digital Service Platforms: A System Dynamics Approach," *Technological Forecasting and Social Change,* Vol. 117, pp. 119–130, April 2017.

[Sho01] V. Shoup, "A proposal for an ISO standard for public key encryption," Cryptology ePrint archive, Report 112, 2001. Available online: https://eprint.iacr.org/2001/112

[Smart16] Deliverable D3.2, "Key performance indicators for p2p energy trading communications," H2020 Project Peer to Peer Smart Energy Distribution Networks (P2P-SmarTest), 2016. Available online:
http://www.p2psmartest-h2020.eu/deliverables

[Sir+19a] V.A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, G.C. Polyzos, "Interledger Smart Contracts for Decentralized Authorization to Constrained Things," Proc. 2nd Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock 2019), in conjunction with IEEE INFOCOM 2019, Paris, France, April–May 2019.

[Sir+19b] V.A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, G.C. Polyzos, "OAuth 2.0 meets Blockchain for Authorization in Constrained IoT Environments," Proc. 5th IEEE World Forum on Internet of Things, Limerick, Ireland, 2019.

[Sir+19c] V.A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, G.C. Polyzos, "IoT Resource Access utilizing Blockchains and Trusted Execution Environments," Proc. Global IoT Summit, Aarhus, Denmark, 2019.

[Sir+19d] V.A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, G.C. Polyzos, "Interledger Approaches," *IEEE Access*, Vol. 7, pp. 89948-89966, 2019.

[Sir+20] V.A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, G.C. Polyzos, "Decentralized authorization in constrained IoT environments exploiting interledger mechanisms," *Computer Communications*, Vol. 152, pp. 243-251, February 2020.

[Sof20] Project Use Cases, SOFIE website. Accessed: 15.4.2020. Available online: https://www.sofie-iot.eu/about/project-use-cases

[Sofi20] SOFIE: SOFIE Deliverable D4.4 – Second Architecture and System Evaluation Report, 2020

[SMS07] B. Sunar, W.J. Martin, and D.R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on Computers*, Vol. 56, No. 1, pp. 109-119, January 2007.

[Spo+19] M. Sporny et al., "Verifiable Credentials Data Model 1.0: Expressing verifiable information on the Web," Draft Community Group Report, W3C, September 2019.

[Ste00] J.D. Sterman, *Business dynamics: systems thinking and modeling for a complex world*, McGraw-Hill Education, 2000.

[Sti+20] K. Still et al., "Platform Economy - Interactions & Boundary Resources: Checklist for Companies," Technical Report, Tampere University of Technology, 2017. Available online: https://tutcris.tut.fi/portal/files/13267284/INTERACTIONS_BOUNDARY_RESOURCES_CHECKLIST_2017_10_27.pdf

[SysFl19] Deliverable 10.1, "Report on the selection of KPIs for the demonstrations," H2020 Project EU-SysFlex, February 2019.

[TuZC16] Tu, Zhiying; Zacharewicz, Gregory; Chen, David: A federated approach to develop enterprise interoperability. In: Journal of Intelligent Manufacturing Bd. 27, Springer Science+Business Media, Van Godewijckstraat 30 Dordrecht 3311 GX Netherlands (2016), Nr. 1, S. 11–31

[VV15] V. Vogelsteller and B. Vitalik, "ERC-20 token standard," Tech. Rep., 2015. Available online: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

[Wik20a] "Federation," *Wikipedia*. Accessed: 15.4.2020. Available online: https://en.m.wikipedia.org/wiki/Federation

[Wik20b] "Metcalfe's law," *Wikipedia*. Accessed: 15.4.2020. Available online: https://en.m.wikipedia.org/wikihttps://en.m.wikipedia.org/wiki/Metcalfe%27s_law

[Wols03] E.F. Wolstenholme, "Towards the definition and use of a core set of archetypal structures in system dynamics," *System Dynamics Review,* Vol. 19, No. 1, pp. 7-26, February 2003.

[Woo18] D.G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," Ethereum Yellow Paper, December 2018. Available online: https://ethereum.github.io/yellowpaper/paper.pdf

[ZhGo96] Zhou, Jianying; Gollmann, Dieter: Observations on non-repudiation. In: Kim, K.; Matsumoto, T. (Hrsg.); Goos, G.; Hartmanis, J.; van Leeuwen, J. (Hrsg.): Advances in Cryptology — ASIACRYPT '96, Lecture Notes in Computer Science. Bd. 1163. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996 — ISBN 978-3-540-61872-0, S. 133–144