applied system innovation

MDPI

Article

A Novel Combination of Distributed Ledger Technologies on Internet of Things: Use Case on Precision Agriculture

Odysseas Lamtzidis [1],†, Dennis Pettas [2,†] and John Gialelis [1,3,*,†]

[1] Electrical and Computer Engineering Department, University of Patras, 26504 Patras, Greece; up1019749@upnet.gr
[2] Computer Engineering and Informatics Department, University of Patras, 26504 Patras, Greece; dennis.petta@gmail.com
[3] Industrial Systems Institute, ATHENA RC, 15125 Athens, Greece
* Correspondence: gialelis@isi.gr
† Current address: Electrical and Computer Engineering Department, University of Patras, Greece.

Received: 4 August 2019 ; Accepted: 12 September 2019; Published: 18 September 2019

Abstract: Internet-of-Things (IoT) is an enabling technology for numerous initiatives worldwide such as manufacturing, smart cities, precision agriculture, and eHealth. The massive field data aggregation of distributed administered IoT devices allows new insights and actionable information for dynamic intelligent decision-making. In such distributed environments, data integrity, referring to reliability and consistency, is deemed insufficient and requires immediate facilitation. In this article, we introduce a distributed ledger (DLT)-based system for ensuring IoT data integrity which securely processes the aggregated field data. Its uniqueness lies in the embedded use of IOTA's ledger, called "The Tangle", used to transmit and store the data. Our approach shifts from a cloud-centric IoT system, where the Super nodes simply aggregate and push data to the cloud, to a node-centric system, where each Super node owns the data pushed in a distributed and decentralized database (i.e., the Tangle). The backend serves as a consumer of data and a provider of additional resources, such as administration panel, analytics, data marketplace, etc. The proposed implementation is highly modularand constitutes a significant contribution to the Open Source communities, regarding blockchain and IoT.

Keywords: precision agriculture; blockchain; internet of things; distributed; data integrity; data monetization

1. Introduction

Internet of Things (IoT) is a rapidly evolving paradigm, ranging over multiple different vital domains such as manufacturing, smart cities, precision agriculture, and smart hospitals addressing the critical mission of data aggregation. Data integrity refers to the reliability and consistency of the data over its entire lifecycle, from sensor detection to cloud storage.

In this paper, a system which ensures the integrity of IoT aggregated field data is proposed and its corresponding alpha prototype implementation is demonstrated in the precision agriculture domain. The system's core is an innovative distributed ledger (DLT) implementation which securely process the aggregated field data and its uniqueness lies in the embedded use of IOTA's ledger, called "The Tangle", used to transmit and store the data. The combination of an immutable ledger of information (in our case, sensor data) and the use of cryptographic primitives, such as public key cryptography, transforms the IoT nodes from data collectors to data owners. The data is stored in an anonymous and secure fashion on the ledger, while the IoT node is the sole actor that can procure access to the data streams. This envisioned Edge-centric architecture where each IoT node is a autonomous unit in a "swarm"

of nodes that belong to the same stakeholder and perform the same "activities" is in tune with the latest industry advancements in the area of IoT, where progressively more activities are migrated from the cloud to the Edge layer. The system's Super Node (SN), aggregates the data from the sensors, packages them into transactions and pushes them in the IOTA network. Access control to the data stored in the Tangle is provided by the Masked Authenticated Messaging (MAM) which uses appropriate keys to manage encrypted data streams over the Tangle. MAM, on the one hand, empowers the users to have a fine-grained access control over sensor-owned data existing in the Tangle and on the other hand, a sensor data marketplace can be built on top of it, exploiting the monetization of sensor data while also acting as an incentive to further install IoT nodes. The system design was first conceptualized in terms of architecture and envisioned functionality in previous work [1], while this publication attempts to highlight an actual implementation. Regarding the implementation, it constitutes an alpha version of the system which showcases the most critical aspects of the architecture as depicted in Figure 1. Additionally, it is deployed in an actual use-case demonstrating promising results with respect to the introduction of DLT technologies in the IoT infrastructure. The use-case comprises IoT Nodes deployed in an actual precision agriculture farm, called IoT nodes, with the collected data send to a gateway node functioning as an Super Node which is responsible for aggregating all sensor data into a data log and broadcasting them over MAM.

In a nutshell, the proposed system highlights how a DLT implementation can enable new functionality in existing IoT systems, while solving critical issues in terms of privacy and security. Moreover, a prototype implementation is presented demonstrating its effectiveness while at the same time enriching the open source community build around the IOTA protocol initiative [2].

The rest of the paper is organised as follows: In Section 1 we introduce the IOTA protocol on top of which the system is built, in Section 2 we go through the state of the art regarding the interconnection of the domains of blockchain and the Internet of Things. In Section 3 the system architecture & implementation is introduced and described and in Sections 4 and 5 we discuss the results and the small-scale demonstration that was conducted. Finally, in Section 6 important issues on the use of IOTA are raised and in Section 7 we outline our conclusions.

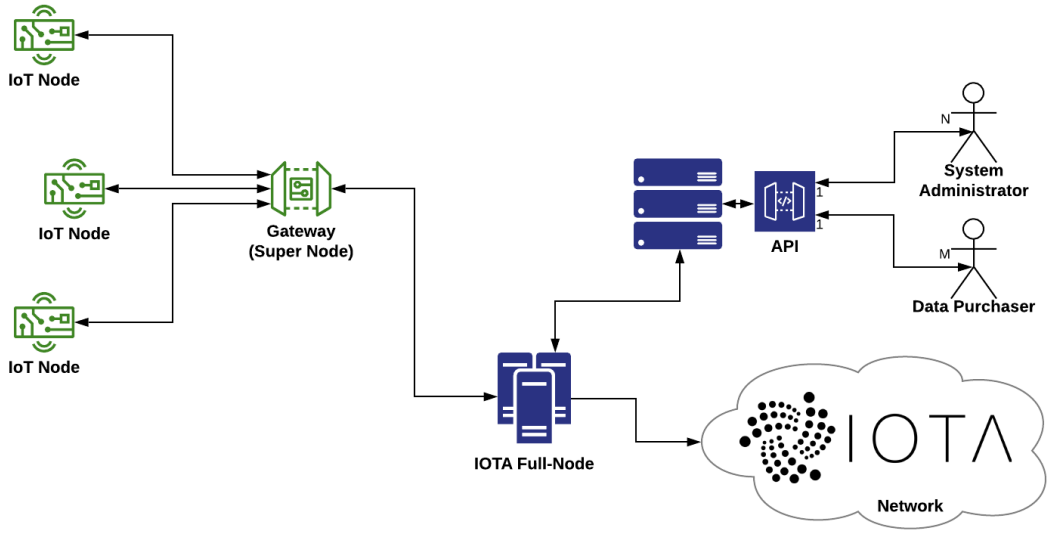


Figure 1. High Level Architecture.

1.1. IOTA Constituents

The IOTA protocol and cryptocurrency were introduced in early 2015, illustrating the use of a Directed Acyclic Graph (DAG) in lieu of a blockchain [3,4]. As Figure 2 depicts, IOTA's ledger, "The Tangle", stores all the transactions that are issued in the network by actors called IOTA Full Nodes

(FN). In a blockchain each block aggregates a large number of transactions and is connected to the previous block of usually the longest chain depending on the consensus rule. In IOTA, each transaction references two (2) other transactions in the DAG so in order to issue a transaction, an agent needs to verify two (2) previous transactions. Albeit each transaction references *directly* 2 transactions, it also references *indirectly* all the transactions that are referenced by those two, either *directly* or *indirectly*. The current implementation offers a reference algorithm [5] to the FN for the choice of the two transactions but there is no enforcement by the protocol. The protocol will give incentives to the FN to choose transactions that are not referenced already, called "Tips", in order for the graph to grow organically. On top of that, IOTA is built using ternary logic, thus using trytes instead of bytes for its various operations (e.g., hashing) [6]. Each user has a unique string formed from 81 trytes called a seed, from which the protocol generates all the user's private and public keys.

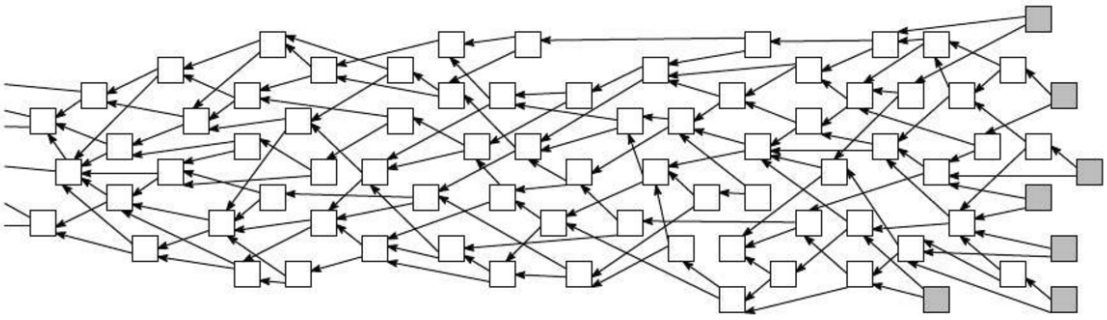


Figure 2. IOTA's ledger, "The Tangle". The graph's edges are the transactions while the vertices indicate the transactions that are referenced [5].

A transaction, as depicted in Figure 3, is the smallest unit of data in the IOTA protocol consisting of 2673 trytes (1589 bytes) and can be used to transfer both value (IOTA tokens) and data (1300 bytes). It is worth noting that as the IOTA network is enriched with new transactions, the size of the Tangle grows substantially. For this reason, the network is currently performing network synchronization or local snapshots at fixed time intervals, eliminating all zero transactions.

```
{
"hash":tx_hash,
"signatureMessageFragment": signature_fragment ,
"address": address,
"value":0,
"tag":"99999999999999999999999999",
"timestamp":1482522289,
"currentIndex":0,
"lastIndex":0,
"bundle": bundle_hash,
"trunkTransaction": tx1_hash,
"branchTransaction": tx2_hash,
"nonce": PoW_nonce
}
```

Figure 3. An indicative example of a transaction structure.

1.2. IOTA Functionality

As noticed in Figure 4, IOTA uses a seed in order to generate multiple private and public keys, as the signing scheme of IOTA is similar to the Winternitz type signature scheme [7]. Thus, each private key should be used only once to sign a transaction, as key re-use leads to forge-ability.

To issue a transaction, the FN needs to perform 3 distinct actions: (i) sign the transaction using a unique private key and store the signature in the transaction; (ii) use the reference algorithm to choose 2 transactions; (iii) perform Proof of Work (PoW); and finally (iv) broadcast the transaction in the network. It is worth noting that in IOTA, PoW is not used to achieve consensus but rather as a spam counter-measure. When a FN receives a transaction from the network, it verifies the validity of the above steps and moreover verifies the validity of all the transactions directly or indirectly referenced. This verification includes both the validity of the transaction structure as well as the absence of conflicts. This process is conducted each time by a different FN that has different view of the Tangle, since at a particular time no FN knows all the transactions that are currently being issued in the network due to lag. Consensus is achieved by collaboratively assessing the state of the Tangle as each FN verifies a specific subset of the whole Tangle. As these subsets overlap, different FN agree on their view of the Tangle and progressively the whole system achieves equilibrium. Thus, a transaction can have different levels of acceptance by the network, depending on how many different nodes have accepted it. The system will let the users to decide what acceptance level is adequate to consider a transaction legitimate and thus proceed with the exchange of goods or services. Currently, the aforementioned mechanism remains a concept in the IOTA white paper and network consensus is achieved using a centralized architecture. The IOTA foundation runs a private special Node called "The Coordinator", which issues regularly special transactions that are called “milestones”. Every transaction that is directly or indirectly referenced by a milestone is considered as accepted. This particular irregularity is also discussed in Section 6.

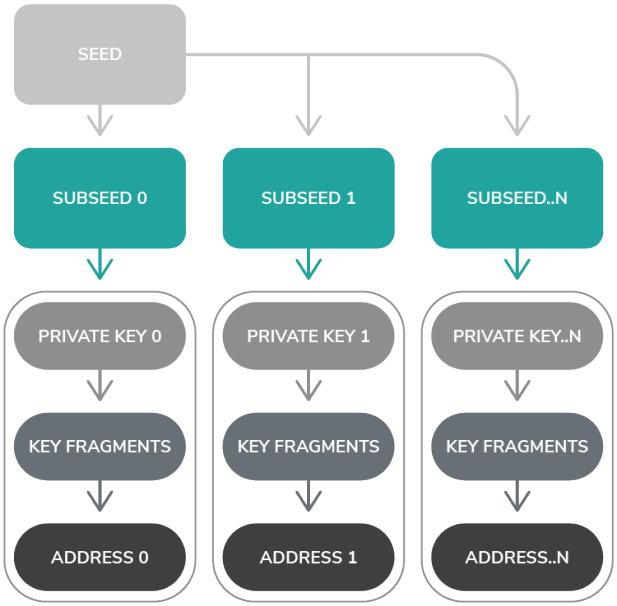


Figure 4. Generation of private and public keys from a single string called a Seed [6].

1.3. MAM Specification

IOTA is also the medium to transfer data in a fashion that ensures their integrity. Using the MAM protocol, a mechanism called channel broadcasters, creates channels to which other mechanisms called channel subscribers, subscribe. MAM’s latest specification is released by the ITsec lab of the Belarusian State University [8]. As illustrated in Figure 5, each channel is divided into endpoints from which messages are splitted into multiple packets and broadcasted. Each channel is identified by a public key, called *chid* (channel-id) and corresponds to a private key used to sign endpoints and/or messages.

Each endpoint is identified by a public key, called *epid* (endpoint-id), corresponding to a private key used to sign messages. The central endpoint is the endpoint whose *epid* = *chid*.

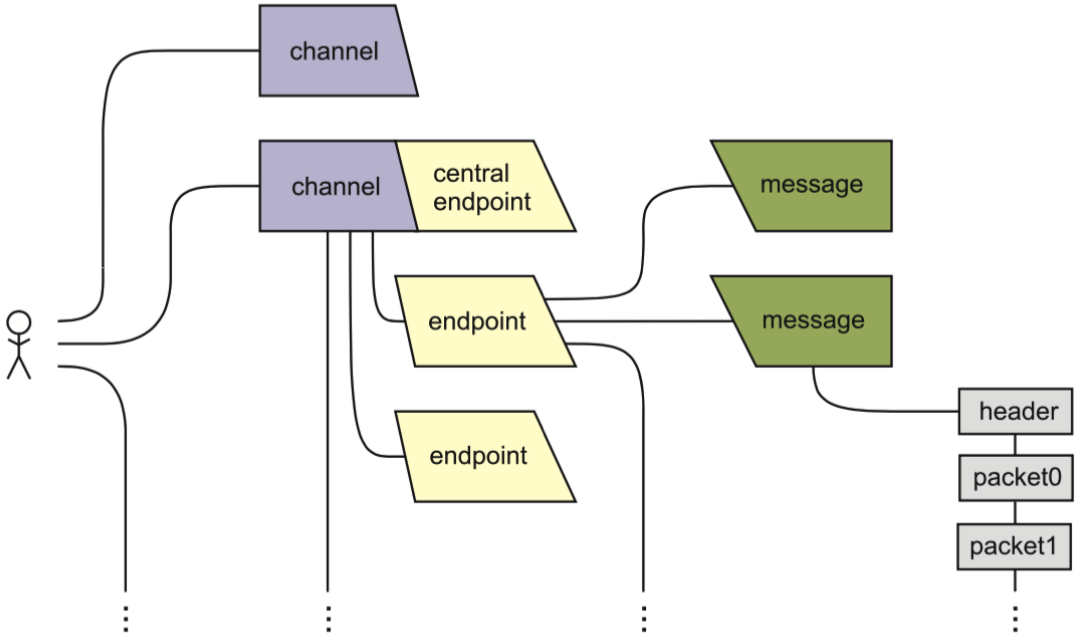


Figure 5. Concept diagram of the Masked Authenticated Messaging (MAM) functionality [8].

1.4. MAM Protocol

MAM's protocol consists of many different layers which are characterised by a specific state. In order to illustrate MAM's usage in the proposed design and implementation, it is pertinent to reference the *WOTS, MSS, NTRU, Protobuf3*, and *MAM2* layers. Winternitz One-Time Signatures (WOTS) layer generates private/public keys and signatures, it verifies a signature and it recovers a public key from a signature. As the layer's name suggests, the signing scheme uses the Winternitz signatures. Merklee-tree Signature Scheme (*MSS* layer) [9] is responsible for generating 2^d signatures of different messages, where d is asserted as $d \leq 20$. A Merklee tree is a binary tree where the leaves are the public keys of a corresponding *WOTS* instance, thus 2d instances, and each tree layer is hashed until the root which stands as the public key of the whole tree. *NTRU* supports the use of NTRU-style public key encryption [10]. *Protobuf3* supports the higher-level encoding, decoding and cryptographic processing of the data. In essence, *Protobuf3* is a language based on the Protocol Buffers Version 2 notation [11]. *MAM2* layer is responsible for the high-level operations of the protocol, like sending and receiving messages.

While the full specification and apt description of each algorithm and data structure can be found in the MAM specification, it is crucial to outline certain high-level algorithms. The creation of a data stream by an agent, starts with the creation of a proper channel. When creating a channel, the protocol creates a Merklee tree, where the Merklee Tree Root *MTR* = *chid*. The inputs of the *CreateChannel* algorithm are height d (as described above) and a channel name. The output is the *chid*. Having generated a channel, the agent can generate an endpoint by running the *CreateEndpoint* routine, where input is a height d, a channel name and an endpoint name. The output is the *epid*. Finally, in order to broadcast a message, the agent needs to procure a Header data structure comprised of a message_id, a type_id, a session key and a *KEY* (optional). The session key can be encrypted using the KEY (either using a pre-shared key or a *NTRU public key*). This is crucial as it grants access control to each message, where only the key owner (either having the pre-shared key or the *NTRU private key*)

will be able to decrypt the session key and access the data packets. A message, finally, has M ≥ 1 data packets, depending on the data size.

1.5. MAM in IOTA

IOTA serves as the transport layer of the MAM data layer protocol; thus, MAM messages are transported using the Tangle. MAM messages are split into fragments and each fragment is encapsulated into an IOTA transaction. Transactions are divided into 2 categories: Header transactions that encapsulate the Channel, the Endpoint and the Header data structures, and the Packet transactions that encapsulate data packets. Transaction objects have numerous metadata but two of them are relevant to the MAM protocol: (i) the tryte address field (81 trytes) = *chid* and (ii) the tryte metadata field (27 trytes) = *msgid* + (0 OR packet order). Both address field and metadata field are strings made of [X] trytes. The metadata_field is a contention of the message identifier and a "0" if it is a header message or the packet's number if it is a packet transaction. The latter is very important, because using the IRI's API, an agent can easily find all the header transactions registered to a specific address, read the session key and then again using the metadata, find and decrypt all the transactions with the specific *msgid*. The aforementioned data structures are presented below, in pseudocode. An IOTA transaction will either have the mam_header structure or the mam_packet structure encapsulated in the data field.

```
{
struct mam_endpoint {
trits name;
mam_mss mss;
};

struct mam_channel {
trits name;
trits msg_ord;
mam_endpoint_set endpoints;
trint endpoint_ord;
}
```

where **mam_mss** is a data structure that refers to a Merklee Tree which holds the signatures as described, **msg_ord** a trit that is incremented each time a message is added in the channel, **mam_endpoint_set** is a set of all the active endpoints in the channel and **endpoint_ord** is a trint that is incremented with every endpoint added to the channel

```
struct mam_header {
mam_channel channel;
mam_endpoint endpoint;
mam_channel new_channel:
mam_endpoint: new_endpoint;
mam_psk psk_keys;
mam_ntru ntru_keys;
mam_msg_id msg_id;
mam_msg_id_type type;
}
```

where **mam_channel** is a reference to the mam_channel data structure (it either references the current channel or a new one (fork), **mam_endpoint** a reference to the mam_endpoint data structure

(it either references the current endpoint or a new one), **mam_psk** a set of the relevant Pre-Shared Keys (PSKs) that are required to unlock the session_key, **mam_ntru** a set of the relevant NTRU public keys which belong to a private key that unlock the session_key, **mam_msg_id**: the id of the message and **mam_msg_id_type type**: a message that can either be a MAM message or a Public Key Certificate.

```
struct mam_packet {
context ctx;
mam_checksum checksum;
mam_payload payload;
}
```

where context references the current state of the MAM library (and its various layers), mam_checksum the checksum of the packet so as the reader can verify it's integrity and mam_payload the payload that encapsulates a segment of the streamed data.

The above data structures are used in the MAM2 reference implementation in C and are based on the MAM specification. The structures are different from those described in the specification as these are the actual data structures that are encapsulated into the IOTA transactions while the others are Protobuf3 models.

1.6. IOTA Challenges

Given the business requirements for data integrity and immutability of the gathered sensor data, the use of a DLT is an appealing solution to an existing problem. Moreover, the usage of cryptocurrencies empowers the system to easily perform automated M2M payments without intervention of the human factor. The implementation concerns the precision agriculture domain, a domain where data can be essential for certain activities, such as crop insurance. Ensuring the accuracy and consistency of the stored data to the detected data sets is critical and requires the exploration of experimental technologies, such as the DLT. Since, there is intention to use the ledger as an immutable storage of sensor data, albeit a temporary one, it is expected to have several transactions per SN. This is a restricting characteristic since most blockchain based projects have a high fee for each transaction, spiraling the upkeep cost of the system. The IOTA protocol, on the other hand, supports transactions with zero fees, demanding only lightweight computation (1–2 min on a PC).

Therefore, it is imperative to design and implement a system where each Super Node aggregates sensor data and issues a bundle at the FN that is hosted in the cloud. The SNs can select how many datasets will be encapsulated in each MAM message, varying on the granularity the system offers, in respect to granting access to specific data. In our case, the Super Node aggregates data from all the IoT Nodes that are placed in the field and forwards them to an Edge service on the cloud. Instead of running an Edge service on the SN, a simulated process was chosen in order to decouple sufficiently the two prototype implementations, the first being the prototype hardware infrastructure of the IoT Nodes and the Super Node and the second being the IOTA data and service layer. The Edge services process the aggregated data so each Node's data are encapsulated into a different MAM stream. This design choice allows the field owner to sell data streams from each Node. The implementation also includes a data marketplace interface, where an potential user can overview the stakeholders that use the system and make a choise to purchase access. Actually, the user purchases the necessary PSK in order to decrypt the messages found in the already known *chid* of the data stream. In the future, the implementation will support multiple authentication modes and workflows, such as the use of Private/Public key cryptography (NTRU). This is exceptionally important as it offers a wide range of access control to support multiple different use-cases and business requirements.

2. Related Work

2.1. Blockchain

The blockchain is considered one of the most groundbreaking innovations of the last decade, foreseeing a completely decentralized future where consensus and trust are built upon mathematical models. It is essentially a distributed database of records of transactions—a ledger—where all the nodes in the network participate actively to reach a consensus on the final status of the database, thus the database can be publicly auditable [12]. The notion of blockchain was introduced in 2008 by Satoshi Nakamoto [13]. It uses well known cryptographic systems, such as public key cryptography and PoW to create a system where value can be transferred without the need for intermediaries [14]. Bitcoin was the first open source project to implement this new vertical which created the notion of cryptocurrencies [15].

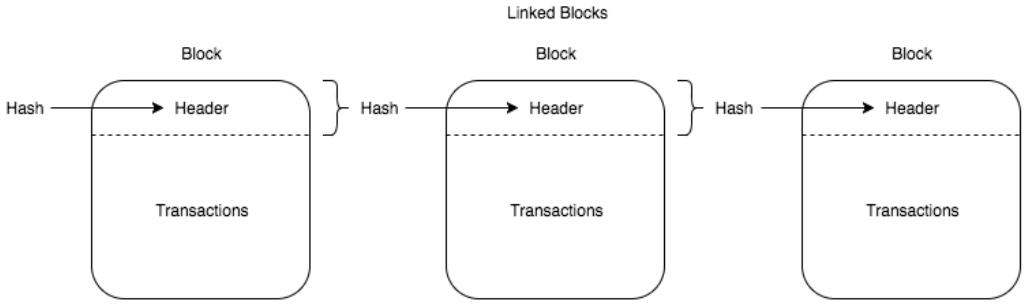


Figure 6. This diagram illustrates the general architecture of a blockchain, blocks of data that are interlinked by using the hash of the last block as the header of the next one [16].

As blockchain started to evolve, the core concepts of Distributed Ledger Technologies are applied to more domains, thus encouraging the emergence of new uses in Ethereum, beyond the transfer of value, such as "smart contracts" [17]. In Ethereum, "smart contracts" stands for a code executed in every node which "checks" certain conditions prior the transfer of funds. Once these conditions are verified it is impossible to breach or cancel the transfer. An interesting aspect of the blockchain technology is the scalability aspect regarding the transactions that the network can process in a secure way. The goal is that the network reaches a consensus on whether a transaction is valid or invalid in a timely manner, avoiding any possible misuse, such as double-spend. Although consensus is well established, blockchain has certain specific characteristics that challenged its status quo, regarding demand for permission-less and trustless structures [15]. For this reason, several consensus algorithms have been proposed for use in a blockchain network. Bitcoin kick-started the research field by using PoW as an integral part of the consensus [13]. In essence, each network node aggregates transactions from the network in a Peer to Peer (P2P) fashion and places them in blocks. In order to attach the new block to the last block of the chain, the node needs to become the first in the network to find the solution to a computational puzzle, in other words, perform PoW. This chain of blocks, the blockchain itself, is shown as a structure in Figure 6. The network agreed (consensus) that the longest chain, that with the highest accumulated computational power, is the correct one. Thus, an entity needs to control 51% of the computational power of the network in order to outperform each competitive node, effectively taking over the blockchain. Albeit theoretically possible, the cost to obtain 51% is deemed exorbitant and consequently this chain is considered as secure [18]. Moreover, there are projects using stake as a medium of consensus, where block producers are chosen by the network based on state of funds and network nodes' votes, such as EOS [19]. The blockchain technology has been notably applied in the industry sector in the form of "private blockchains", networks where a party needs permission to participate, thus rendering the nodes trustworthy [20]. The nodes can either belong to the same organisation or to a consortium of organisations, where a small number of nodes (in comparison with "public blockchains") are required to secure the network, increasing the TPS throughput of the system

considerably. Hyperledger fabric [21] is considered mature enough to be implemented in realistic business scenarios that demand private blockchains.

2.2. IoT End and Super Nodes

The open source initiative is capable of supporting the free use of open source-licenses. It is through this open source movement that the popularity of open-hardware has been triggered and consequently made available with different options for open hardware microcontrollers-based platforms. The question of which platform would be ideal to use, depends on the requirements of the system. These requirements comprise power consumption, cost and type of connectivity among others. An apposition of some of the most notable open-hardware platforms follows: Arduino Genuino 101: The module contains two tiny cores, an x86 (Quark) and a 32-bit ARC architecture core, both clocked at 32 MHz [22]. Orange Pi: Orange Pi 3 is a single-board computer with an Allwinner H6 system-on-a-chip (SoC) combined to a quad-core, Arm Cortex A53-based 64-bit processing cores running at 1.8 GHz and 1GB LPDDR3 memory and a Mali T720 graphics processor [23]. Raspberry Pi 3 B+: Raspberry Pi uses the Broadcom BCM2837B0 system-on-chip (SoC) and includes four high-performance Cortex-A53 (ARMv8) 64-bit processing cores running at 1.4 GHz with 32 kB Level 1 and 512 kB Level 2 cache memory, a VideoCore IV graphics processor, and is linked to a 1 GB LPDDR2 (900 MHz) memory module on the rear of the board [24]. From the above-mentioned open hardware platforms, we are utilizing Raspberry Pi 3 B+ jointly with our proposed End Node in order to formulate the Super Node.

An IoT system can function and transfer information only when the devices are safely connected with wires or wirelessly to a communication network. This connection is established by choosing the appropriate type of IoT protocol based on our system's specific requirements, like bandwidth and power consumption. The two major M2M (Machine to Machine)/IoT protocols are the Constrained Application Protocol (CoAP) [25], and the Message Queuing Telemetry Transport (MQTT) [26]. Both the MQTT and the CoAP are designed as a long-term vision that will enable their use in a lightweight environment. They both work well with low power and network constrained devices, so the choice depends on the application use case. In our particular use case, for reasons of simplicity and fast deployment Post Office Protocol (POP3) [27] was adopted.

Hardware-wise our proposed IoT node is built around a state-of-the-art microcontroller, a high-performance RF and mixed-signal system, Flash/EE memory and SRAM as thoroughly described in [28]. Its power supply system is based on an integrated boost regulator that converts DC power from a PV cell, charging a Lithium-ion storage element for unceasing operation. Moreover, the node provides connectors for various sensors and system debugging. Software-wise, there is no operating system but the firmware code in C language with an approximate size of 10 Kbytes including the radio and sensors control libraries. Its main modules are the Microcontroller and Radio Module, the Power Supply Module and the Input/Output Module. The Microcontroller and Radio Module uses Analog Devices ADuCRF101 microcontroller which is a fully integrated single chip data acquisition solution designed for low power wireless applications. It features a 12-bit ADC, a low power Cortex^{TM} M3 processor from ARM®, a 431 MHz to 464 MHz and 862 MHz to 928 MHz RF transceiver, and 128 KBFlash/EE memory packaged in a 9 mm × 9 mm LFCSP. The Power Supply Module is built around Analog Devices ADP5090 chip which is an integrated boost regulator that converts DC power from PV cells or TEGs. The ADP5090 provides efficient conversion of the harvested power from 16 μW to 200 mW. A 4 × 4 cm 2V monocrystalline PV panel has been used with 45 mA peak output. The PV panel charges a 3.7 V-650mAh Li-ion battery. The output from the PV panel and the battery drives the ADP190 (linear voltage regulator) which finally provides a 3.3 V stable output. The Input/Output Module provides connectors for various sensors, such as I2C interface, SPI, UART, Interrupt pins, A/D interface, and SWD for system debugging or firmware download.

In a typical deployment and depending on the appropriate configuration of the firmware, two types of IoT nodes are distinguished: the broadcasting IoT nodes stations called Sensor nodes and a single receiving IoT node station called centre node. The centre node is continuously in reception mode at the

pre-set frequency channel and its serial port is always connected to any device with a full TCP/IP stack (in our case a Raspberry Pi 3 B+), forming the Super node depicted in Figure 7. A Super node can serve concurrently multiple end nodes. In the unfortunate event of a collision, packets of measurements are lost but a small shift on the end nodes' clocks ensures that such a collision will not occur during the next broadcasting session.

Figure 7. The Super Node.

2.3. IoT Security and Privacy Challenges

Although new architectures are emerging in the field of IoT, i.e., the Edge computing paradigm, the most common architecture is the cloud-centric one [29]. The cloud is responsible for identifying and authenticating the various IoT devices, provisioning new ones, aggregating their data, analysing them and providing additional services to the stakeholders. The growing number of IoT devices combined with centralization, will almost certainly lead the process to bottleneck [30]. On top of that, IoT devices are exceptionally prone to malicious attacks, such as hacking, remote hijacking and Distributed Denial of Service (DDoS) [31]. In the case of a successful attack, the whole system could be in jeopardy, which could prove extremely detrimental in domains like manufacturing industry. Even a simple electric energy sensor could be used in order to provide burglars with sensitive information for the absence of the occupants of a house. According to a recent EU commission report, 72% of EU residents could be affected by this attack vector [32], as by 2020 about 72% of EU residents will own a smart energy meter. Finally, it should be noted that the stakeholders who own the server are granted full authority over the data, enabling them to censor, edit or even delete them. The recent event in Flint, Michigan, USA about lead contamination of tap water, emphasised the need for data integrity in the IoT sector. Investigators discovered that officials had tampered with water samples discarding two of them, in an effort to adjust the report's findings regarding the parts per billion (ppb) of lead in the water [33].

2.4. Blockchain @ IoT

A possible solution to the above mentioned challenges could be the implementation of blockchain protocols in the IoT architecture, from the sensors to the cloud layer. Using the blockchain as a medium to store data signatures (e.g., hashes), the stored data become immutable. In the unlikely event of data tampering or deletion, it can easily be revealed by simply checking the hashes and their corresponding data. Moreover, the blockchain can serve as an immutable log of events (e.g., device provisioning, data creation), securing the network from intrusions [34]. An illustration on how blockchain can be used in the domain of smart home to enhance security and privacy is depicted in [35]. In the design of Dorri et al.

a blockchain is used—with elimination of PoW—merely relying on a hierarchical structure and distributed trust to maintain the blockchain properties. The local blockchain is used to control the devices and their data and also to create an immutable history of the devices and their actions. IOTA was the first DLT project with IoT as the main application (hence the name IOTA). In theory, by differentiating from the blockchain technology with the use of a DAG, it can overcome the scalability issues mentioned above while offering adequate security during the transfer of value. It is one of the few DLT projects that uses post-quantum cryptography, meaning that the established protocols will be secure even after the appearance of the first quantum computers. In addition, the MAM protocol empowers the IoT system to easily incorporate IOTA as a secure medium for data transfer, storage or as an integrity layer.

This paper builds upon the state of the art, by incorporating several innovative designs for the future IoT system. The system features extreme battery efficiency, as well as low-cost devices. The IOTA layer, introduces a new approach for data delivery and storage, enforcing integrity nearly end-to-end. A low-cost system with emphasis to security has the potential to be featured in the sector of micro-insurance where farmers purchase crop insurance for small and remote farms [36]. The data integrity layer delivers security to both the farmer and the insurance provider, automating the refund process based on the data gathered from the sensors or even completely automating the process with the use of smart contracts with Qubic [37].

3. System Architecture and Implementation

3.1. System Architecture

The proposed system design and implementation comprises several subsystems, including a backend cloud system, an Edge service, IoT nodes, and specific IOTA subsystems. Our proposed architecture shifts from a cloud-centric IoT system, where the Super nodes simply aggregate and push data to the cloud, to a node-centric system, where each Super node owns the data pushed in a distributed and decentralized database (i.e., the Tangle). The backend serves as a consumer of data and a provider of additional resources, such as administration panel, analytics, data marketplace, etc. Specifically, a narrow down approach was followed commencing from a three-tier reference architecture proposed by Tranoris et al. [38], as shown in Figure 8, which was modified accordingly to meet our needs.

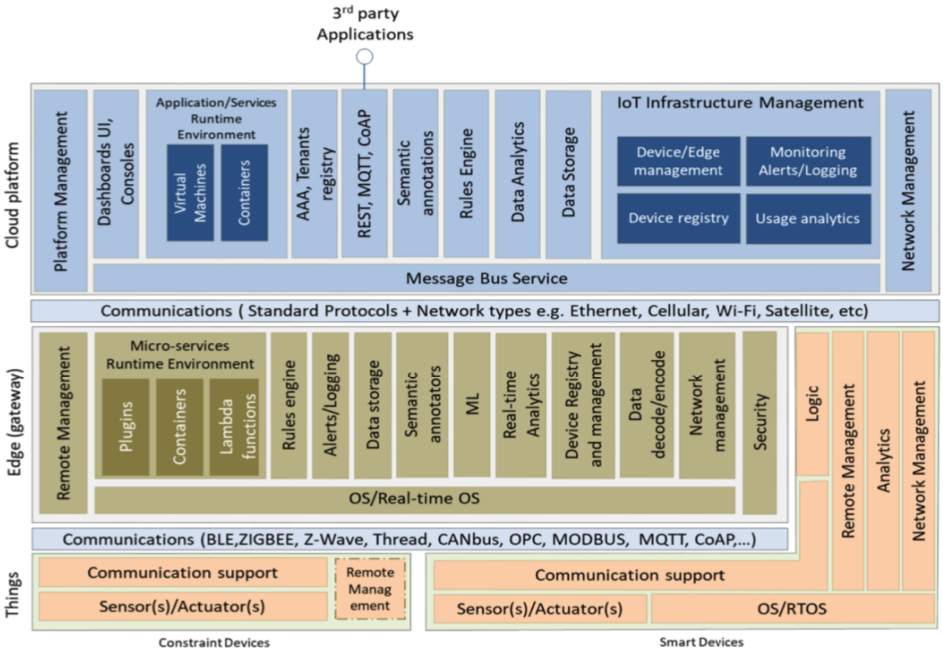


Figure 8. The three-tier Reference Architecture.

Figure 9 illustrates the activities diagram of a typical IoT application that was used as a template for the implementation of our prototype [38]. The diagram was further simplified in order to keep the complexity of the prototype low. The semantic notation submodule was removed as it was deemed unnecessary. The MAM submodule is an addition to the generalized IoT architecture, providing a service which is responsible for receiving the data and based on the metadata, creates the appropriate MAM messages. This submodule is also responsible for connecting to the IOTA Full Node. The final architecture that was adopted is shown in Figures 10 and 11.

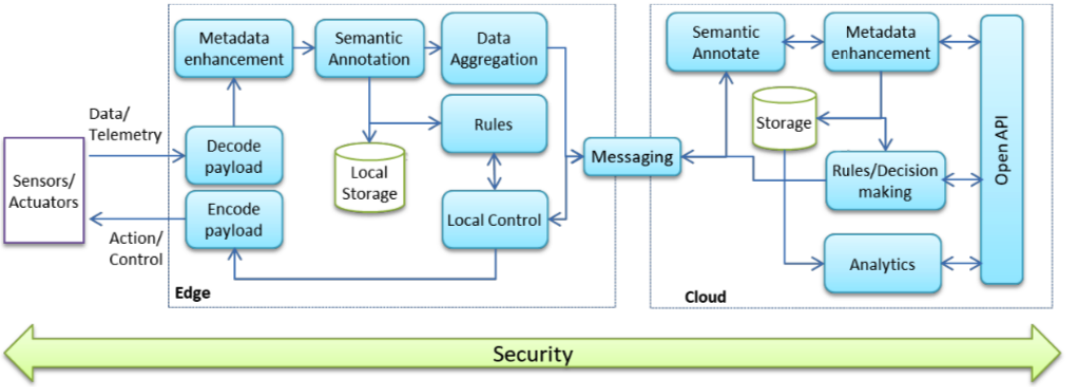


Figure 9. Subsystems Activities Overview.

Process wise, data originated from the sensors are decoded in the Edge through specific drivers. Afterwards, they are enriched with time stamp and node denotation metadata and saved. Local storage not only cancels latency but in conjunction with a rule engine, it could enable intelligence. Then the data are transmitted to the Cloud where they traverse through the same subsystems as shown in Figure 10.

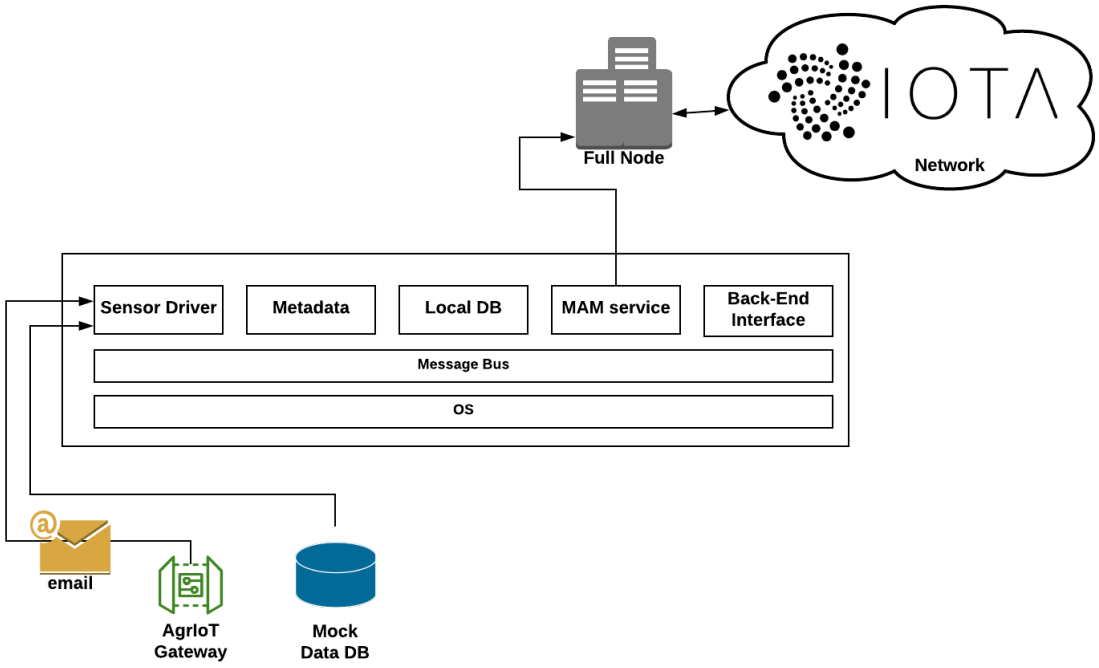


Figure 10. The Edge Super Node Module.

The cloud module also offers a data marketplace based on IOTA as well as a dashboard for the IoT system owner as Figure 11 depicts. Essentially, these are activities that are built on top of the Open API. Various sub-modules were discarded as they increased the complexity of the system without contributing to the implementation's purpose.

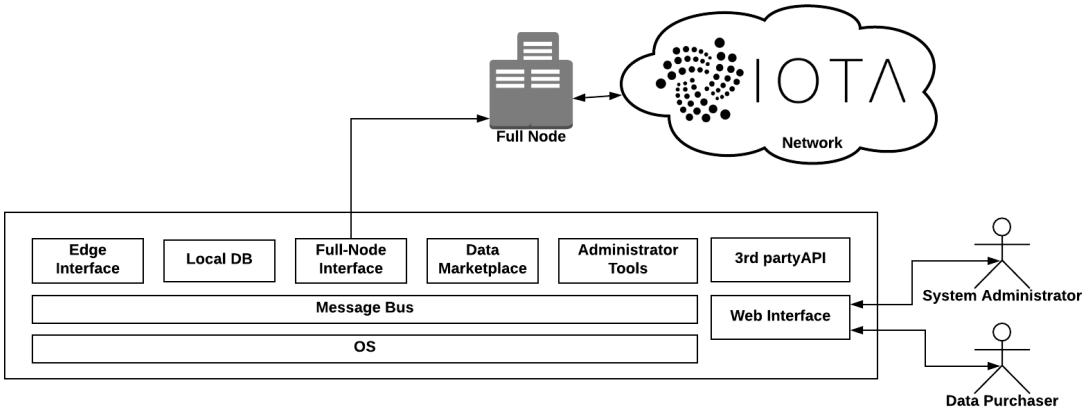


Figure 11. The Edge Super Node Module.

3.2. Envisioned Core Activities

The architecture is built around 3 key activities *setup phase*, *data log*, *data request* which encompass the key innovations that are illustrated in this design.

During the **setup phase**, IoT nodes create the proper MAM channels in order to start broadcasting in the Tangle. Using a unique seed, the IoT node runs the *createChannel(dimension)* function twice. Note that the higher the dimension of the Merklee Tree, the higher the requirement for memory. The first channel is called status_channel, it has no endpoints other than the central and it is used to publish status messages of 1300 bytes (1300 ASCII char) as follows:

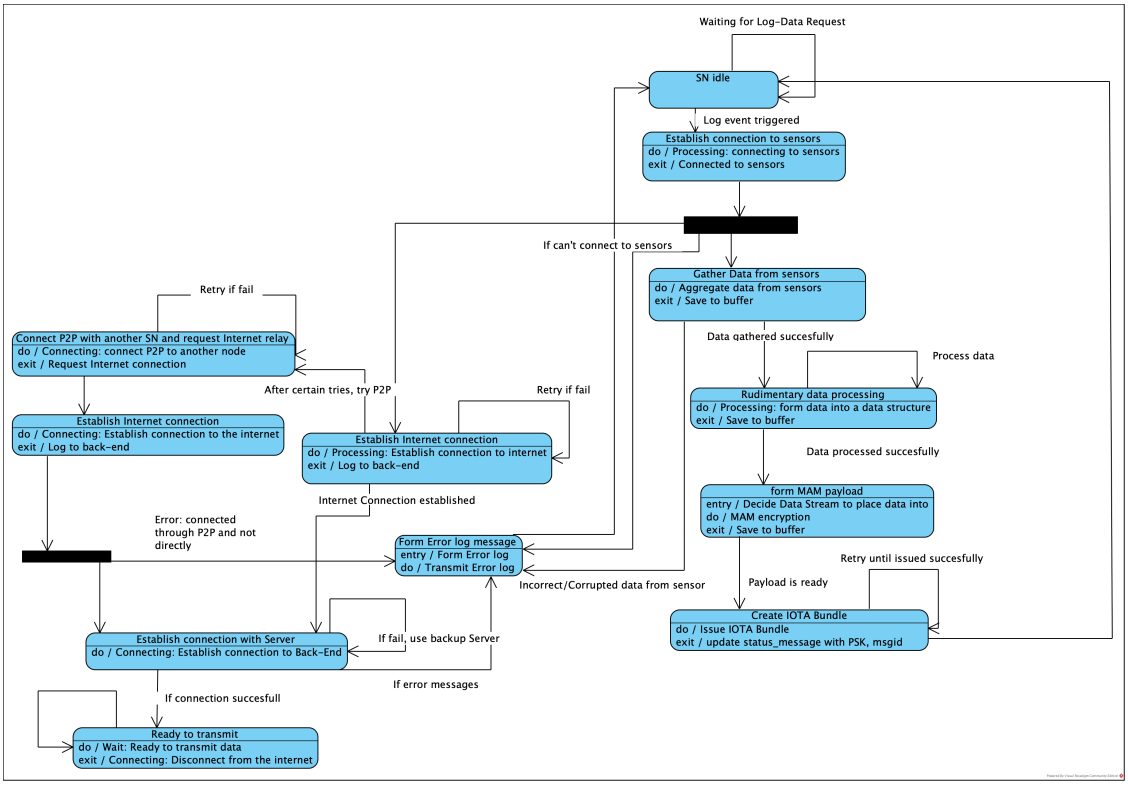
```
status_message: [string node_id, integer geo_loc_lat,
integer geo_loc_alt, integerbattery, string data_channel_chid,
string last_msgid, string chid_PSK, string data_description,
integer error_code].

where:
node_id: Unique id of the IoT node
geo_loc_lat: geographic location latitude
geo_loc_alt: geographic location altitude
battery: Battery level between [0, 100]
data_channel_chid: string of trytes representing channel_id
last_msgid: string of trytes representing the message_id of the last message
chid_PSK: string of the pre-shared key required to unlock the data stream
data_description: short string representing the data
(e.g.TEMPBAT|TEMP|PRES|HUMID|RAIN|TEMPSHT21|HUMSHT21 )
error_code: a unique integer that represent error messages (e.g., 103)
```

Each message is encrypted with an NTRU scheme, using a public_key that is shared by the backend system. According to the above described scheme, the IoT Node ID is publicly auditable, since it resided in the Tangle, but it is only readable by the parties that have the appropriate private key. In our case, only the backend can read it. As such, everyone can verify its integrity and correctness,

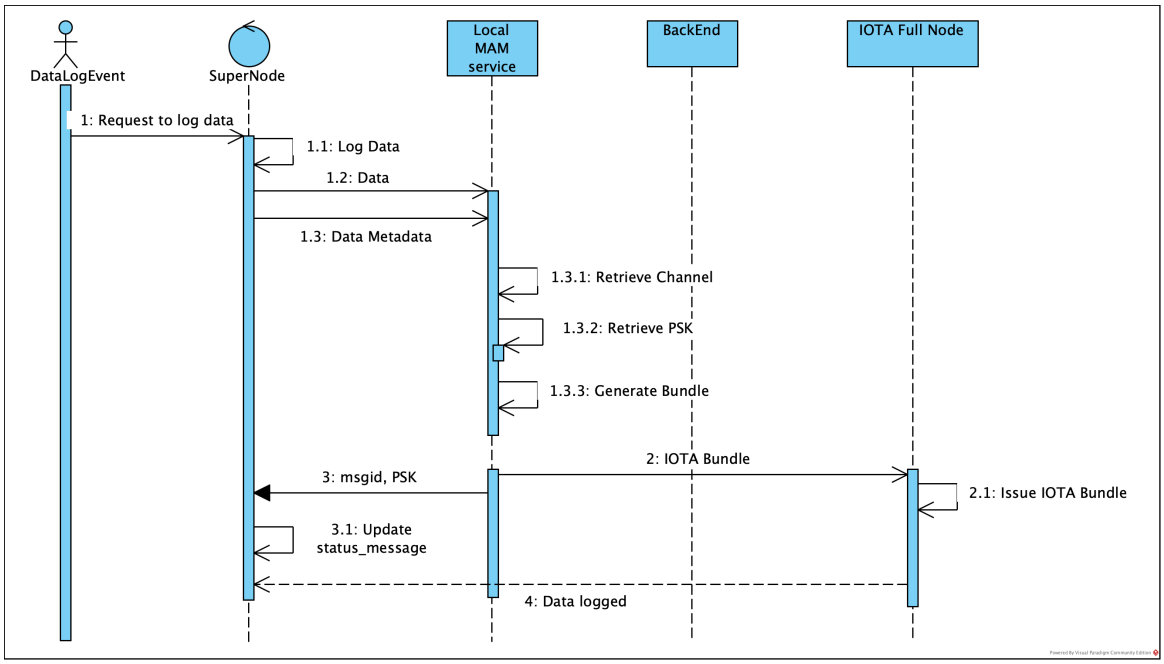
but only the backend can extract the sensitive information. When the channel exhausts the Merklee Tree, the last message points to a newly generated *status_chanel_chid*. It is important to underline that the IoT node is able to dynamically change the format of the offered data streams, by changing how the sensor data logs (SDLs) are encapsulated into different channels/endpoints and the frequency of the PSK change. All the changes are eventually mirrored in the status messages. In case a sensor needs to be added to the system, *setup phase* is initiated through the functions in the *setup_driver.js* and its data are stored through Mongoose in the MongoDB database.

Data Log Phase: The IoT node services are activated on event basis and read data from the sensors that are stored in the Tangle. Before that, the data are enriched with metadata (e.g., sensor id, type, etc.) and stored for local use (e.g., local decision-making). Then the MAM service, adhering to the business logic encapsulates the data into data streams. In case a sensor data log (SDL) is divided into multiple data streams, they are referred to as data log segments. Afterwards, the MAM service creates the proper MAM Bundle where the header and packet transactions are placed and hashed. The service then sends the MAM Bundle to an IOTA Full node in order to issue it to the network. Finally, the MAM service updates appropriately the *PSK* and `data_channel_chid` fields of the `status_message`.

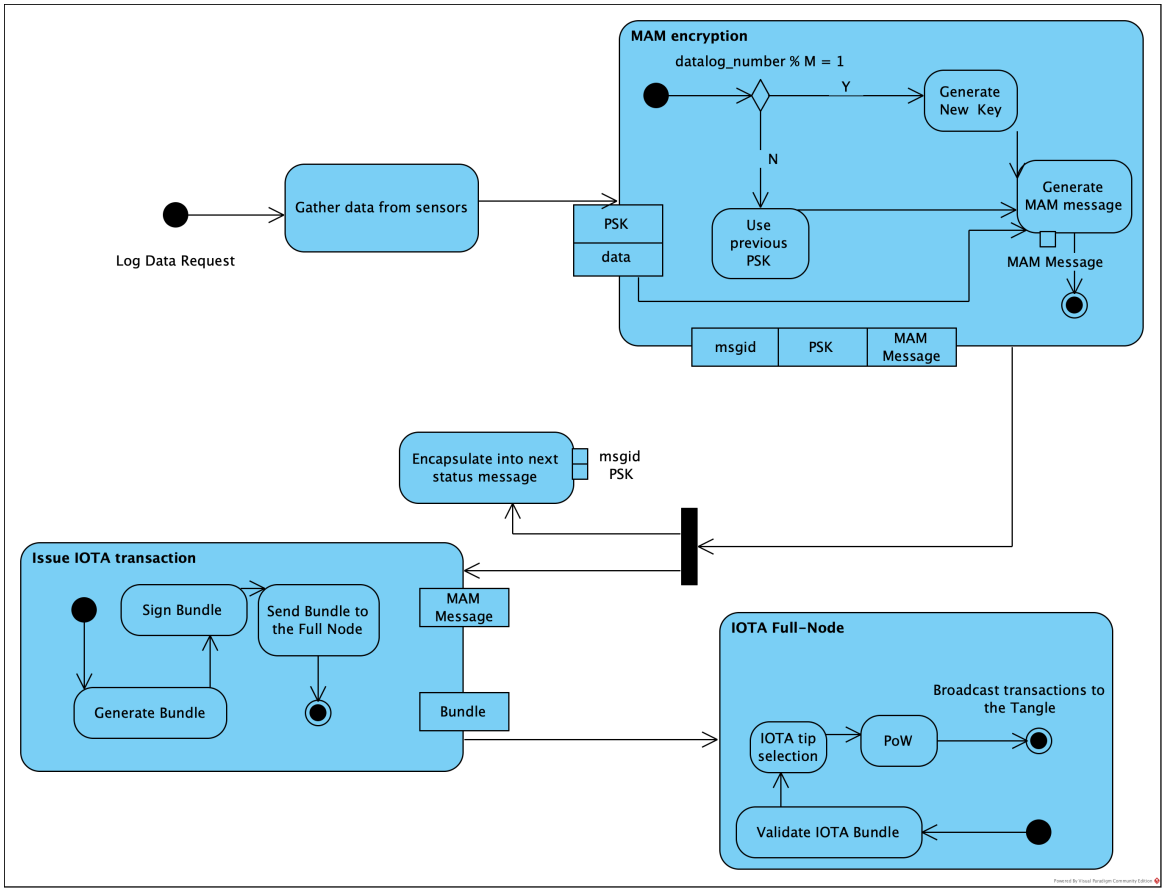
Each data stream can be viewed as a time series of SDLs $(SDL_1, SDL_2, \ldots SDL_n)$ at corresponding instances $(t_1, t_2, ..., t_n)$. Likewise, as *PSK* changes every m SDL (m ≥1), the SDL time series creates a *PSK* time series $(PSK_1, PSK_2, ..., PSK_y)$ and a data channel *chid* time series $(chid_1, chid_2, ..., chid_z)$. In the proposed data configuration, in n SDL there are m *PSK* (m≤n) and z *chid* (z≤m), since the *chid* changes every 2^d messages, where d is an arbitrary integer as mentioned in Section 1.4. All the above are illustrated in Figure 12. Figure 12a describes the state machine of the data log phase, Figure 12b illustrates the required activities, while Figure 12c showcases the sequence of the messages between the actors (users and services).



(**a**) State Diagram



(**b**) Sequence Diagram

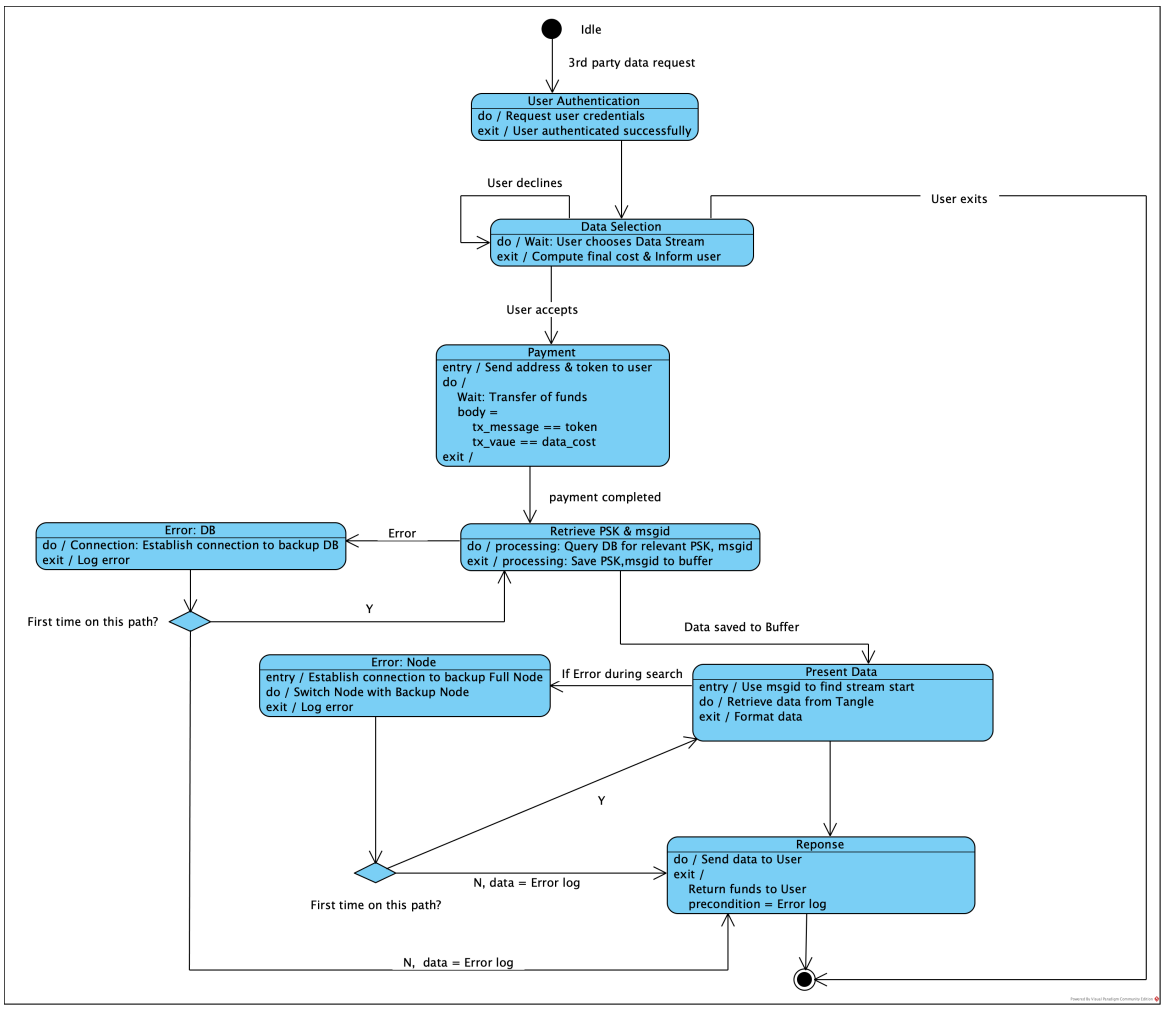


(**c**) Activities Diagram

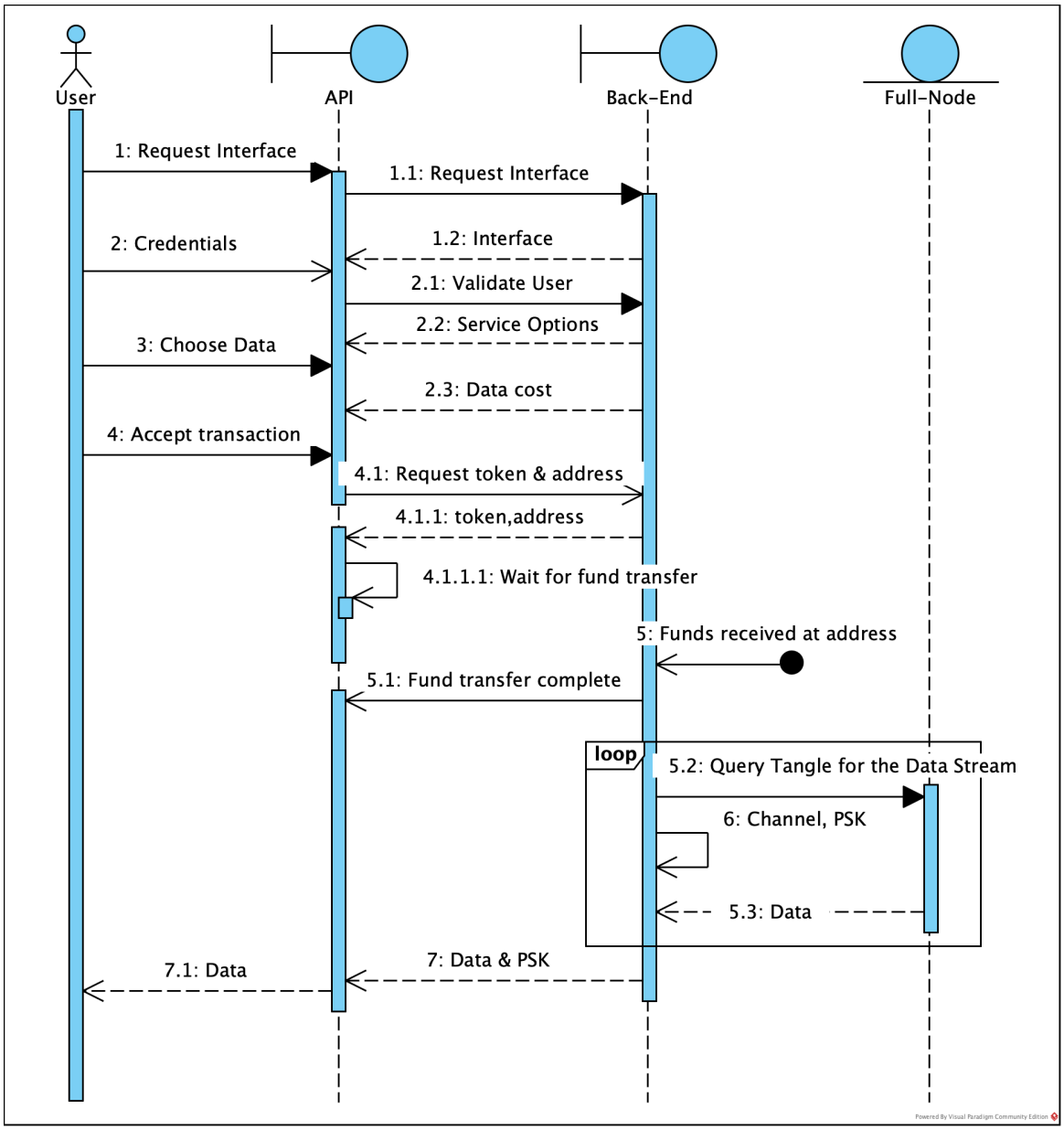
Figure 12. UML diagrams regarding the data log phase.

Data Request Activity: During the data request phase, a user accesses the marketplace and creates an account and applying appropriate filters (e.g., data types, geolocation, etc.) reveals data streams on

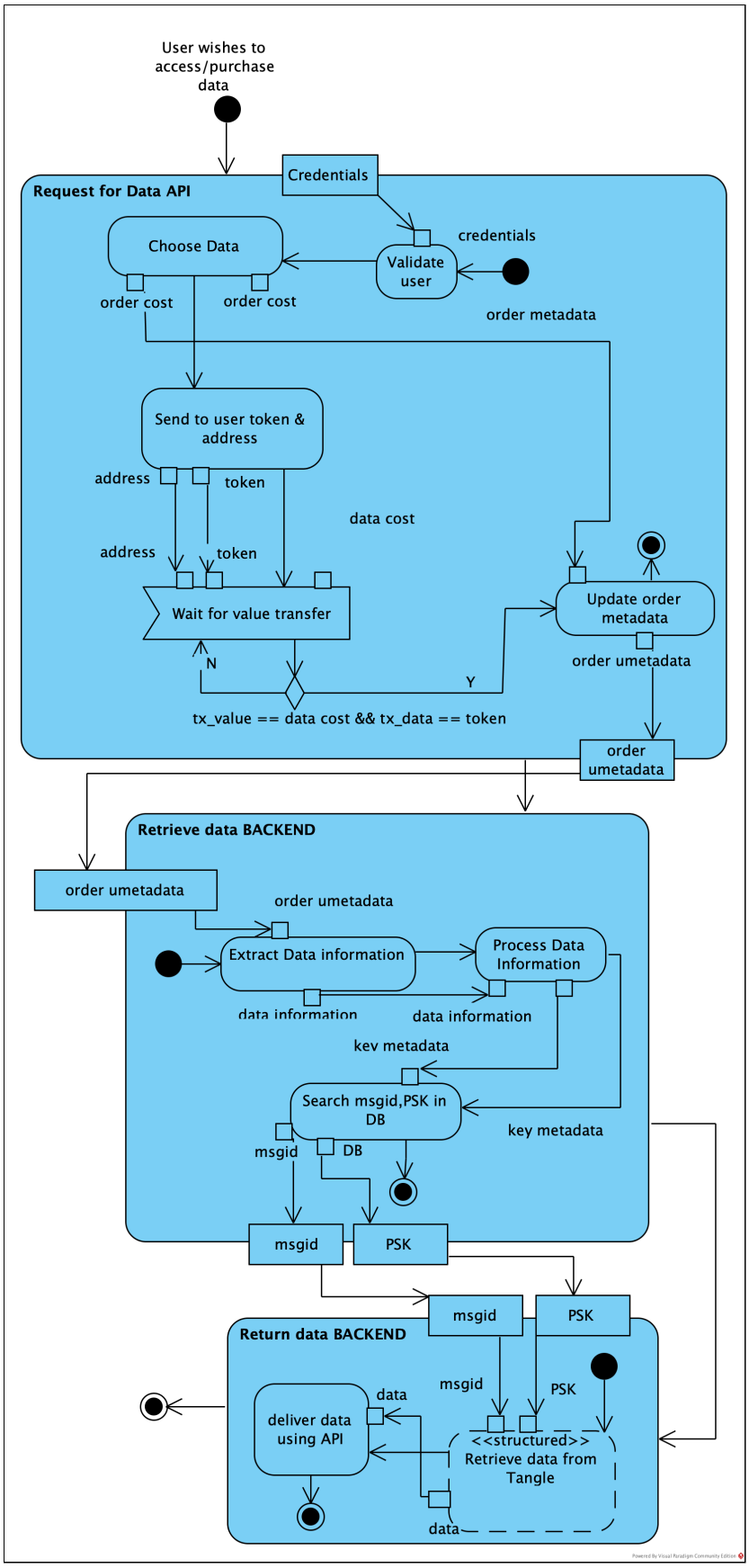
the map. The user can purchase data, in chunks of m SDL, selecting from specific streams, where m is the result of the PSK changes frequency for each data stream. The user can either purchase access of data already in the Tangle or pre-purchase access for future SDL by subscribing to a data stream for a specific amount of SDL chunks. After the user selects the appropriate data stream and accepts the exchange of tokens, the backend generates an IOTA address and an Oauth token by using a uniquely generated seed [39]. Consequently, the user acquires the IOTA address to deposit the requested amount and the Oauth token in the data_field. Upon the verification of the transfer, the backend searches the database for the appropriate *msgid* and *PSKs* combinations which are used to find the transactions in the Tangle, fetches the Bundles and decrypts the data. Data are served through the web interface to the end user as the corresponding diagrams depict. The above mentioned service is also offered through a REST API that can be used in lieu of the web interface. Figure 13a showcases the state machine of the data request phase, Figure 13b depicts the required activities, while Figure 13c illustrates the sequence of the messages between the actors (users and services).



(**a**) State Diagram



(**b**) Sequence Diagram



(**c**) Activities Diagram

Figure 13. UML diagrams regarding the data request activity.

3.3. Architecture Metrics

We can identify two distinct metrics that encapsulate the capabilities of the proposed architecture. Those two metrics are:

1. Iota Transaction throughput
2. PoW

Iota Transaction throughput Regarding the Iota Transaction throughput, that is, the number of transactions per second (TPS) that the network can support, we are referring to a metric that solely depends on the inherent characteristics of the protocol (IOTA) is important because it directly influences the average time that a transaction needs to be accepted by the network. The current implementation of the tangle supports a TPS of 4–5 [40] with an average confirmation time of 10 min. The Tangle is thus mature enough to support the described architecture as it offers an acceptable transaction time with no fees. Finally, this latency, although adequate for real-time transactions is bound to be decreased through improvements of the IOTA network architecture itself.

Proof of Work This metric is important because it highlights and ultimately defines the scalability of the proposed system. The PoW must be done for each transaction (in the current state of the network, and for the foreseeable future). PoW can either be conducted on the hardware, which is preferable as it increases the autonomy of the IoT device, or can be offloaded to a server to be offered as a service. Latest advancements in FPGA implementation [41] of the PoW algorithm has illustrated the possibility of adding a relatively cheap dedicated PoW core to the IoT hardware, which will be able to conduct PoW in an extremely efficient manner, both in terms of time and power consumption. By implementing an FPGA array in a cloud based PoW, the cost is reduced even more, proving the architecture to be extremely cost effective even with a large increase of IoT nodes.

Each transaction can encapsulate 1 to n data transmissions. By increasing the number of data log packets in each transaction, we decrease the required PoW (by lowering number of transactions needed to transmit the same number of information) but we also decrease the granularity of the data access (since access is granted on transaction basis). The current time needed to perform PoW is shown bellow:

- A median computing time of 90 s on Raspberry power 3:4× ARM Cortex-A53, 1.2 GHz [41]
- Natively on our IoT Hardware (IoT Node): Unfeasible, both in terms of time and power consumption. (M-cortex)
- On ×86 CPU (4×cores + GPU, upgrade ccurl implementation "dcurl" [42]): 9 s
- FPGA implementation: 70 ms

Each MAM message can encapsulate up to 1300 bytes of information, or 65 SDL, since each Sensor Data Log has a size of 20 bytes as dictated in the specification. In a scenario where each node emits an SDL every 10 min and there are 12,000 nodes that are used in various applications and are supported by the same backend server, the following metrics are found, in regards to data access granularity and PoW cost. In the scenario, it is assumed that each MAM message demands 3 distinct IOTA transactions to be fully transmitted (this can change in the future as MAM is still in development). The final number of transactions is enlarged by a factor of 10% in order to allow for unforeseen errors that may appear and may lead the IOTA node to re-perform PoW. Figure 14 shows on a per node basis, the amount of transactions issued per day versus the access control granularity, translated in number of SDLs that will be encapsulated into every MAM message. Figure 15 illustrates the linear increase of the daily cost to conduct PoW for the whole system, assuming that it is conducted in a centralised data center on FPGA arrays that consume approximately 8 W and the cost of 1 Kwh is $0.2.

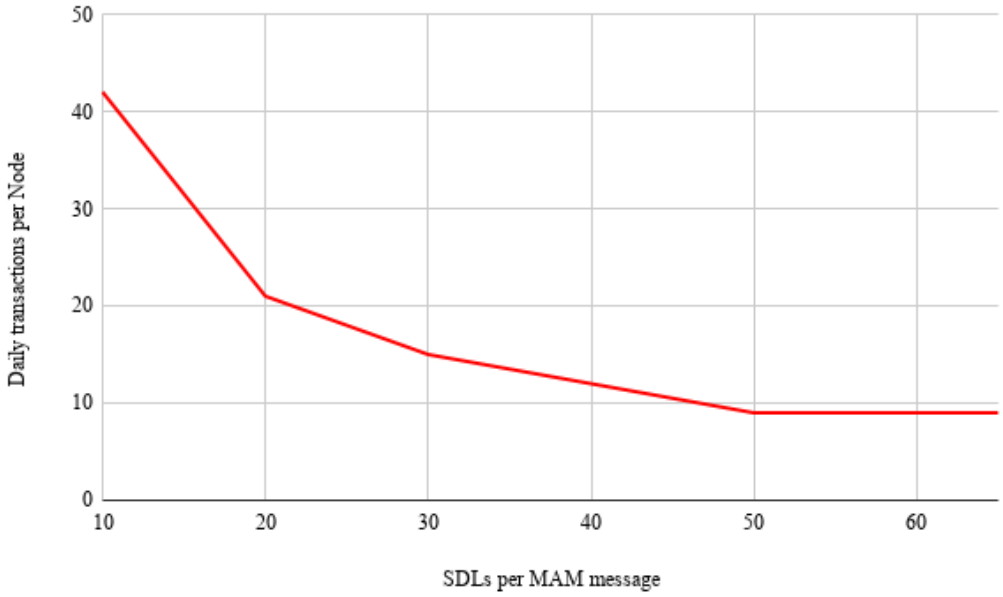


Figure 14. Number of Transactions vs sensor data logs (SDLs)/MAM message.

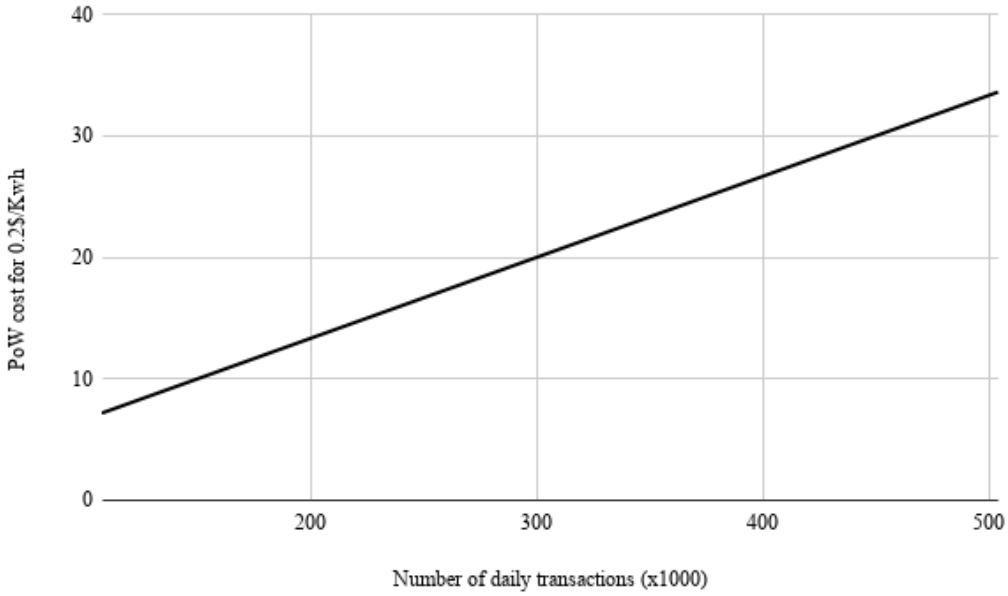


Figure 15. System scalability in terms of Proof of Work (PoW) cost. X-axis is the number of transactions in units of thousands.

4. Results

4.1. System Implementation

Three distinct modules are implemented, namely the IoT nodes, the Edge System and the Cloud system. For reasons of efficiency, the Edge System application logic is allocated into two distinct nodes, the Gateway Super node that serves as an aggregator and the IOTA powered Edge node that is hosted in the Cloud. The data from the Gateway Super node are eventually pushed to the Edge node through an email carrying along a log file with the sensors' data as shown in Figure 16. The implementation is mainly focused on the Cloud system as the IOTA powered Edge node simulates

most of the activities and pushes the data that it receives to the database, so in essence the Edge and Cloud systems communicate through the common database.

```
[00:00:04 5 0 397 7 106 10276 85 0 108 89 99 7a]
[00:01:59 4 0 403 7 111 10275 86 0 111 89 85 7a]
```

Figure 16. Log File Instance.

The Edge logic is ported in C to be able to run natively on the hardware, while the MAM sub-module is simulated since there is no stable MAM implementation.

IoT Nodes Module

The physical parts of the IoT node constitute the following blocks, as depicted in Figure 17

Figure 17. Physical Parts of the IoT node.

With respect to its performance the most notable aspects are:

- **Coverage Distance:** The maximum, transmission distance is a function of antenna power, frequency and transmission speed. For 868 MHz radio frequency, 1Kbps transmission speed and 10dbm antenna power, a 12.5 km line of sight has been achieved. For a 433 MHz frequency the distance is doubled. The transmitter and receiver use antennas of half wave 1.2 dBi peak gain. In case the topology requires it, IoT nodes are used as repeaters (with minor modifications to the firmware).
- **Energy Consumption** Hibernate mode is the dominant mode for the IoT node since the total consumption in this state is 6 μA. During broadcast, for output power of 10 dbm in the antenna, the consumption is 32 mA. Fifteen (15) levels of output power are available, while the transmission speed can be selected from a range of 1Kbps to 300 Kbps. Therefore, for a 20 bytes transmission packet every ten (10) min with maximum antenna power and 1 Kbps transmission rate, the average power consumption of the IoT node is approximately 22 μA and its 650 mAh battery could endure for approximately 3.5 years. The node's battery will be recharged through the photovoltaic panel which allows the battery to fully charge within 20 h of exposure to solar radiation.

Regarding data aggregation, the sensors depicted in Figure 1 acquire the corresponding parameters shown in Table 1.

Table 1. The aggregated data by the IoT Node.

Sensor and Scope
Humidity,Temperature, Pressure and Ambient H, T, P
Rain Amount and Rain
Leaf Wetness and Leaf
Soil Moisture and Soil
Wind Speed, Direction and Wind
Solar Radiation and Pyranometer

4.2. Edge Node and Cloud System Modules

In general, the modules are implemented with Javascript paired with a NOSQL database and serviced by an AWS scalable EC2 cloud. In order to achieve the specified scalability, modularity and expandability requirements, the prototype implementation [43] of the API and the Backend was built based on an adaptation of the M.E.A.N. (MongoDB, Express, Angular, Node) stack. MongoDB is a NoSQL database that allows flexible data modelling and intuitive control over the structure and the location of the data, while ensuring fast performance and seamless data migration. Express is a minimal and flexible Node.js web application framework that provides robust and well-maintained middleware and HTTP utility methods. Node.js is an asynchronous event-driven JavaScript runtime, that is designed to build scalable network applications. Node uses a deadlock free, streaming and low latency-oriented framework/design, which is ideal for the current implementation requirements of the proposed architecture. Angular.js is a JavaScript frontend framework, that is the state of the art in modern web-based applications. For the prototype version, the boilerplate for Angular is preserved, while the Jade templating engine which works in conjunction with Angular is implemented in order to further boost the scalability, expandability and modularity of the system. The project is hosted as a cloud service by Amazon Web Services (AWS). The prototyped API uses an Elastic Computer Cloud, a web service that provides secure, resizable computing capacity paired with a cloud-based MongoDB server, that handles the connectivity and the Data Streaming with the Edge-Service. Figure 18 presents all the aforementioned technologies and the way they are orchestrated.

- **Edge Module:** The edge Module provides, through an HTTP connection paired with an extra security key, the ability of the system to accept and update the sensor pool. The edge interface also maintains, when needed, a Mongoose powered connection to the system's Local Database with a unique and secure Open Authentication (Oauth) key pair combination that is kept internally and is defined during the system initialization. When a new sensor data log is introduced to the system, after it passes a sanitisation phase, it is imported through a *POST* request into the Database. It is then able to be served by the Backend interface to the user through the normal procedure. Every sensor that belongs to a field, is paired with that field using a corresponding unique field id and each field is showcased through the Google Maps API in the map on the front-end main view. Figure 19 showcases the Admin/User Authentication Layer, which through passport and PUG front end template engine ensures that the user and admin level actions are distinguished both visually and internally.
- **IOTA Full Node Interface:** The IOTA Full Node interface interacts with the IOTA network, through the IOTA DevNet servers that act as IOTA Full-Nodes (FN). When an order is placed by a customer through the Front-End interface, a unique order ID is created and an Oauth confirmation code is served to the user. The user can then place their payment in the corresponding address that is provided, with the confirmation code on the message of the transaction. Additionally, a pending order entry is created via a *post request* in the database and is showcased to the user. When an order is placed, the IOTA Full Node Interface module that checks for new transactions handles the inbound transaction and cross-checks the Oauth verification key. If the keys match,

the pending order is accepted and the corresponding data are handed to the user via My Data view on the front end. As Figure 20 depicts, the user can then download the data.

- **Administrator Tools:** The administrator tools provide a Backend insight for the system administrator to interact with the data streams that are provided in the platform. The availability of the data is set through the ""*UPDATE*" functions as shown in Figure 21, while the views are provided by Chart.js [44] as shown in Figure 22.
- **Cloud Technology:** The entirety of the backend server and the local NoSQL database is uploaded in an AWS EC2 cloud service and runs on a CENTOS Linux Virtual machine. This service is chosen since it provides the necessary functionality to serve as a proof of concept system, while it also has the capability to scale up in the future, keeping up with the market's needs. Any system updates are pushed through Git that is installed locally and the data uplink from the live sensors can be linked through a *post request* with a secret key ensuring that the source providing the data is verified.

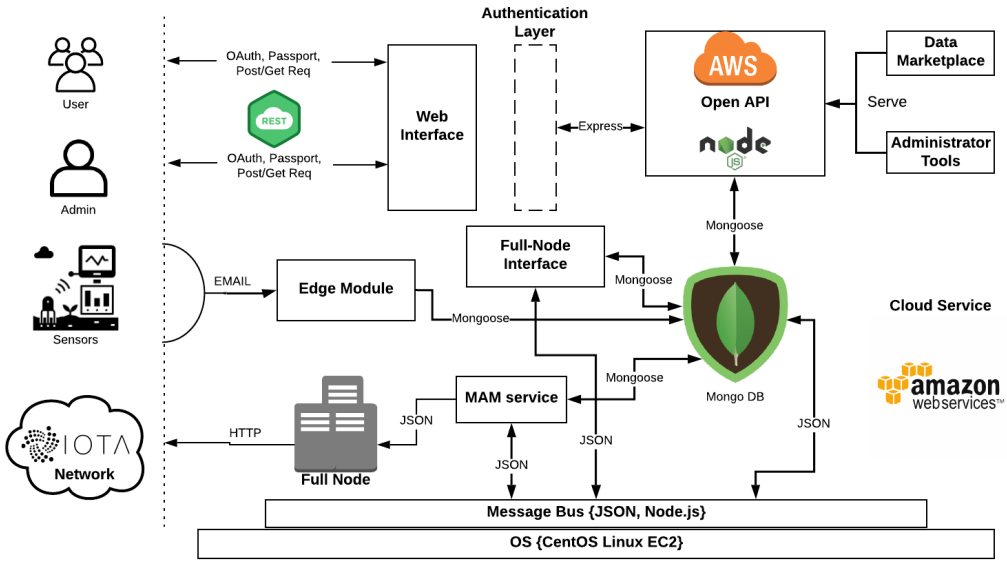


Figure 18. Edge Super Node and Cloud Subsystems.

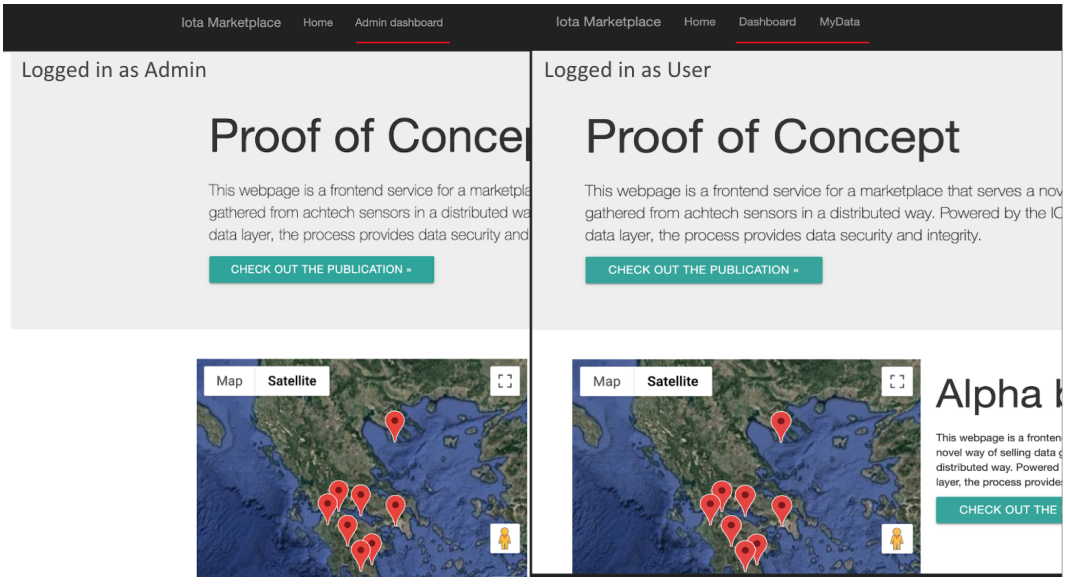


Figure 19. Edge Super Node and Cloud Subsystems.

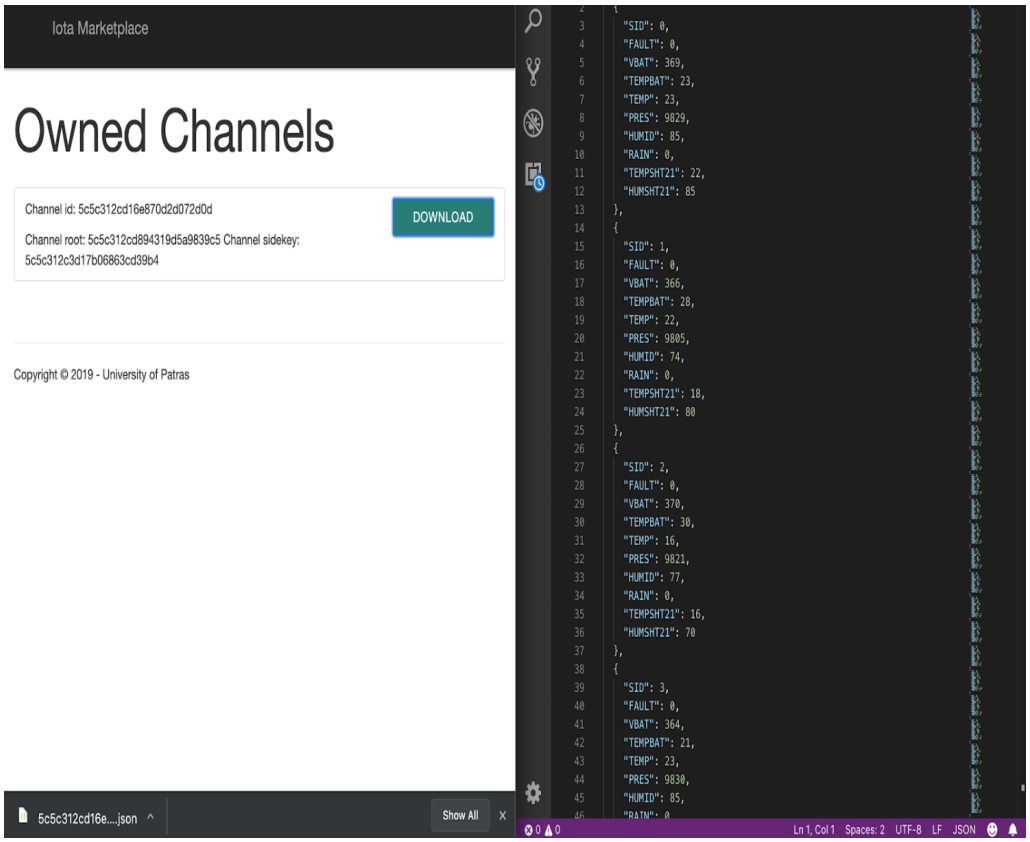


Figure 20. Data View.

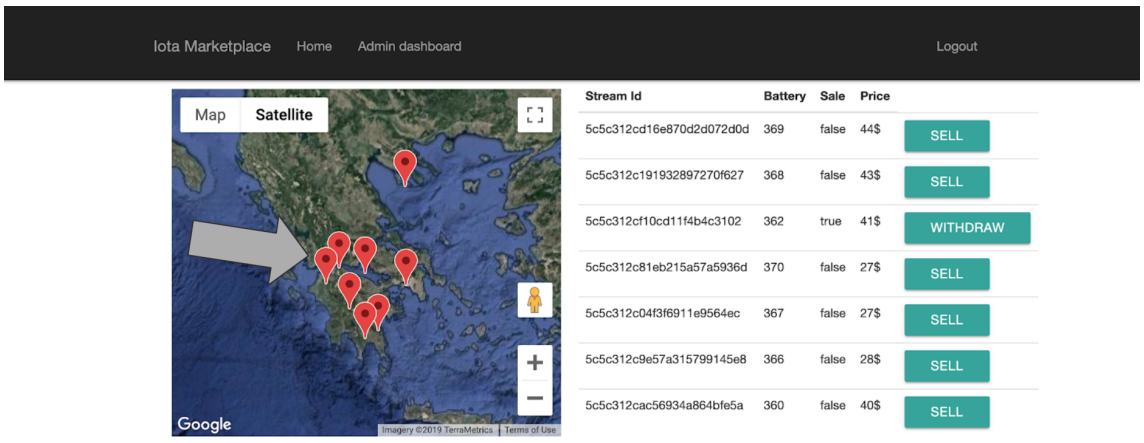


Figure 21. The system administrator can overview the data channels and select those to be available in the marketplace.

- **Sensor Driver:** When a sensor is imported into the system, the sensor driver service runs the driver initialization function in order to update the setup, the interface and register the sensor details into the database. During the initialization, the driver runs preliminary checks ensuring sensors's health and stability, with the `run_preliminary_routines()` function. It also uses the `check_statistics()` function in order to display sensor logging and performance information regarding all its vitals and other statistics regarding use, install date, etc. Finally, with the `import_sensor_to_db()` function, the sensor is incorporated into the system and marked available for use. Since the data are aggregated using an email client, the current sensor driver implementation is for reference only.

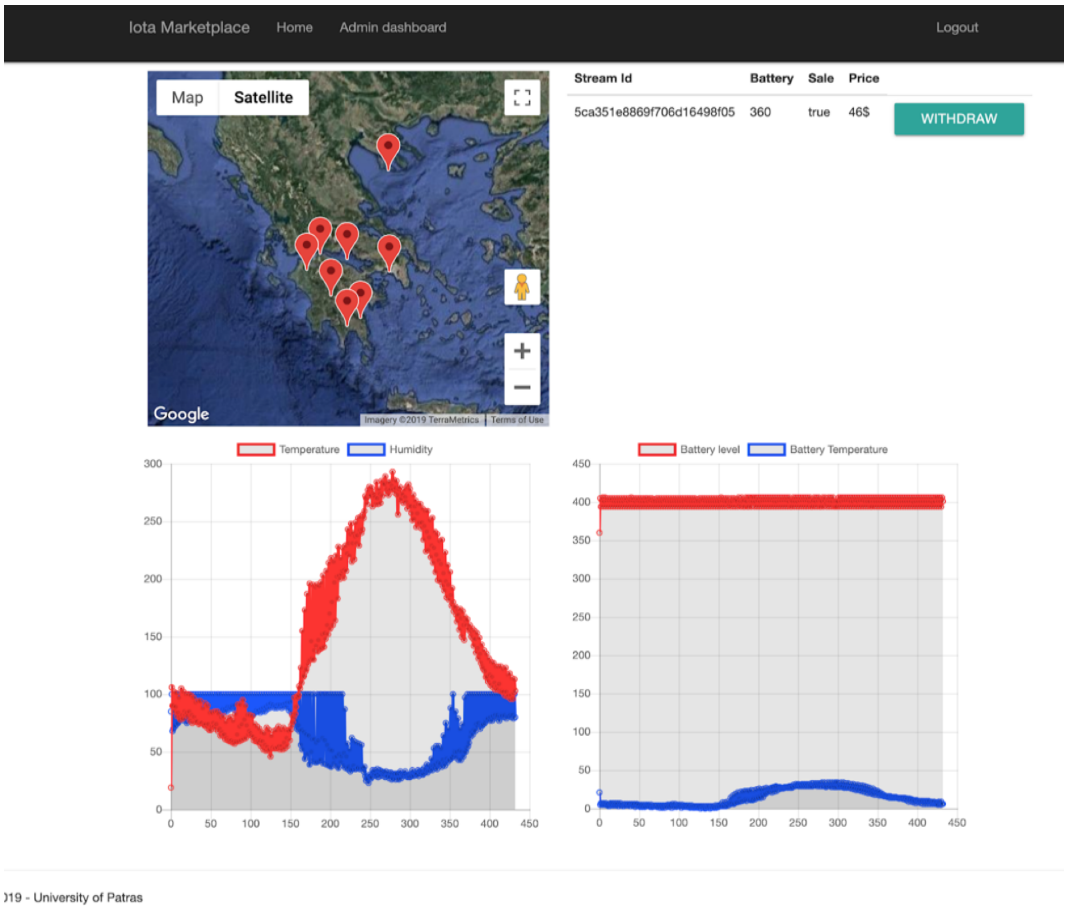


Figure 22. System administrator data views.

- **MAM Service:** This service provides a structure of functions that handles the data through the MAM data layer such as the initialization of the public chain (`_public_chain()`) or the addition of a channel to an existing chain `append_channel_to_chain()`. When the service interacts with the database by establishing a secure connection through the mongoose API. For the Merkle tree generation, the service shall utilize the `generate_merkle_tree()` function. To trace a corresponding channel in the database, a query is executed and the response is fed to the corresponding function `fetch_from_Tangle()`.
- **Database Service:** This service facilitates the connection with the MongoDB database and the handling of the data before their incorporation to the system. This service uses the `semantic_filtering()` function to filter, label and export in JSON format the sanitized data. There are also functions for data cleaning, aggregation, back up and maintenance. The data flow within the Node.Js powered backend server is facilitated through JSON files, thus maintaining the integrity of the structure of the data and the ease of use.
- **Web Interface** The user interface is generated by *PUG* engine which is a predecessor of JADE. PUG, in essence, enables the generation of dynamic and reusable HTML documents, while the incorporation of the modern Bootstrap CSS framework results into a minimalistic user experience that maintains the structural modularity required for a multi view/authentication-layer website. The map provided in the home page, is powered by Google development platform and the corresponding graphs in the administration panel are generated by parsing the available data through JQuery and showcasing through Chart.js, while with different authentication layers provided by `Passport.js` the user can see their corresponding available sub views. Any of the unavailable pages (such as the administration panel for the users) are protected with the same library. The front end interface is modular and expandable.

5. Verification and Small-Scale Demonstration

5.1. General

The activities envisioned for the proposed system are grouped into 2 main categories: (i) Edge Activities and (ii) IOTA powered Activities. The Edge Activities comprise (a) data aggregation; (b) metadata enrichment; (c) local rule-based decision making and (d) local control. Regarding the IOTA powered activities, the Iota Powered Nodes work in a collaboratively fashion, exchanging value for resources and/or services. IOTA serves as the infrastructure for the automated exchange of value in a publicly auditable fashion while enhancing trust between the parties. Clusters of nodes also cooperate, employing IOTA functionalities, such as the flash channels to transfer IOTA tokens offline using common transport layer protocols such as 802.15.4 and 802.15.1. or LoRa [45]. Indicatively, the IoT nodes could be further upgraded and exchange IOTA tokens in order to exchanges services. For example, they could improve the efficiency of their own rule-based system by exchanging data, or (given the proper infrastructure) procure and transfer electrical energy between stakeholders, or in case of internet failure, the request from their neighbors to act as gateways.

5.2. Implemented Activities in the Use Cause

As mentioned in Section 3.1, the use case prototype is built around 3 key activities: *setup phase*, *data log*, *data request* but setup phase is currently innactive as the system is implemented progressively, in a modular fashion. Below, a succinct overview of the implementation of each activity is given.

Data log During runtime a variety of data are stored in the MongoDb server. Live data are currently sent through an IGMAP interface to a Gmail account and then are scrapped through Python and the corresponding Gmail API. After the arrival of the data packets, a sensitization function formulates them into JSON and the Python script, using an HTTP *POST* request and a predefined authentication key and forwards the data alongside with the id of the sensor in the database. The MAM service is abstracted through a mock up series of functions that semantically complete the aforementioned behaviors in the `mam_service.js`.

Data request A variety of CRUD operations in the backend service as well as the MAM service were implemented, so that the user can access the data in the MongoDB. The POST/GET requests, during the interaction with the PUG/Jquery front end, occur in an async format provided by Node.js and are served statically by Express. When a end user registers to the system, a process is initiated in the user.js file, performing a *POST* request to the server, checking the validity of the data the user provided. In case the data are correct, a secure password is generated by the `genSalt()` function of bcrypt cryptography library that is then hashed through `bcrypt.hash()` and stored through Mongoose in the MongoDb database through a `newuser.save()` function. Finally, the page is refreshed in order to showcase the success of the registration. The new user is then granted user level status which enables the Dashboard and My Data tabs. When a user selects a field or a stream, a variety of *POST* and *GET* functions are initiated, and a filtered list of the requested data are served in the frontend by an async mongoose query. When a user buys a stream, the async function `createOrderKeys()` generates the authentication token through OAuth, providing it along with the rest of the required information. The user can then initiate a transfer of funds through the IOTA network in the given address in order to complete the order. This functionality can be achieved via an external service or via a module given by the IOTA foundation.

The backend maintains a service that checks the status of pending orders. When the appropriate funds are sent to the corresponding address, the authentication token of the message that lies in the transaction is parsed, and the same token is fetched through an `order.findOne()` function from the database. If those two tokens match, the stream's id is attached to the buyer, and the pending order is deleted. The buyer can then access and download the data through the MyData tab on the front end.

5.3. Application Use Case Description

The capabilities of the proposed system are demonstrated in the precision agriculture domain, since the deployment of numerous sensors enables the farmers to closely monitor their fields and substantially increase their yield and quality while lowering the production costs. In our use case, the proposed system is used to gather the required data from three different vineyard pilots as Figure 23 depicts. The physical installation of the IoT nodes with their sensors and the corresponding Iota Powered Nodes followed the plan indicated by domain experts (phytopathologists). Each IoT node with its sensors collects various sets of data covering a radius of 25 m with a minimum distance between two IoT nodes of 50 m. One Edge Node is deployed in each pilot. In pilot 1 the following parameters are collected: soil moisture, air temperature, air humidity, and atmospheric pressure. In pilot 2 the following parameters are collected: leaf wetness, air temperature, air humidity, and solar radiation. In pilot 3 the following parameters are collected: rain amount, air temperature, air humidity, and wind direction/speed. Each IoT node is in a hibernate state and every 10 min wakes up, collects the measurements from its sensors and emits them at a pre-agreed speed and frequency channel. Each IoT node has a unique 16bit ID and does not expect any acknowledgement from the receiving Edge Node assuming that the measurements have reached the destination. Each packet of measurements, with the correct CRC, constitute a text-based dataset of an email which is send to the database of the backend server through GPRS once a day. The above parameter data have been collected uninterruptedly in the backend server for approximately five (5) months. The collected information is valuable to the farmer and his agronomist as it offers a live view of the farm's micro-climate. The agronomist can monitor different meteorological phenomena, which are used to detect various diseases within vineyards such as Downy mildew, Black rot and Botrytis. Details regarding the use of the collected information by the agronomist and the parameters of the smart-agriculture pilot are not the goal of this work and are omitted.

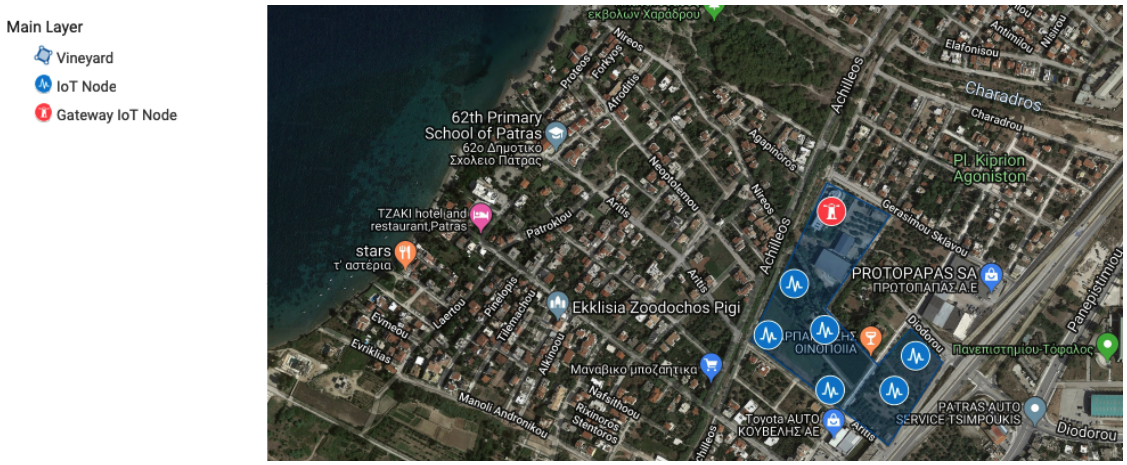


Figure 23. Precision Agriculture Use Case.

6. Discussion

The proposed implementation shows how a DLT technology can potentially serve as a transport and data layer protocol that enables data integrity. It showcases a decentralized IoT smart precision agriculture system, where each Super Node functions as an autonomous unit, aggregating data and participating in a M2M economy that exchanges services and data for value. IOTA, albeit its drawbacks, deems to be an excellent choice to serve as the infrastructure to support a decentralized transfer of value and data, in a manner that is publicly auditable.

IOTA is a pioneer in implementing a DAG structure, in lieu of blockchain, as the immutable ledger of the network and it is an novel approach on bringing such a system to consensus and eliminate the need for miners and for transaction fees. As the Tangle is not functioning in a truly

decentralized fashion, the IOTA foundation operates a special node called "The Coordinator" that is responsible for issuing milestone transactions on specific intervals [46]. Currently transactions are not approved by the network using a consensus algorithm, but their confirmation depends solely on whether they are referenced (directly or indirectly) by a milestone transaction. Although the IOTA foundation claims that this step is necessary to protect the network until it reaches a safe critical mass, critics make a point that "the Coordinator" could actually censor specific transactions by not referencing them [47]. Another important question raised is the question of cryptographic maturity of the project. IOTA introduced a large number of novelties, amongst them being the use of a ternary system instead of binary and the use of post-quantum cryptographic algorithms based on the Winternitz one-time signatures [7]. On top of that, IOTA firstly introduced its own ternary hashing algorithm, called Curl-P(Prototype). Cryptographic researchers, though, illustrated vulnerabilities in the cryptographic primitives which forced the project to replace the Curl-P function with the peer-reviewed Keccak-384 [48]. Finally, the ternary system makes it considerably harder to assess the protocol for vulnerabilities using well established tools and methodologies [48]. The IOTA foundation claims that most of the vulnerabilities regarding the cryptographic elements have been fixed and are undergoing improvement [49]. Regarding the network, the Coordinator acts as a safety mechanism for the time being, while research is conducted on the consensus algorithms.

Due to the use of quantum resilient cryptography, IOTA currently has a substantial footprint of 1589 bytes in contrast to Ethereum's 100–110 bytes [17]. In order to assist the IOTA Full Nodes handling the Tangle, the network performs a synchronised snapshot, where all zero-value transactions are deleted and all of the confirmed transactions are reduced to a database. From that point, the Tangle starts growing again until the next snapshot. This poses a considerable problem since all MAM transactions are zero-value and will be deleted at the next snapshot. These apply to IOTA Full Nodes but the IOTA foundation intends to release Permanent Nodes that will store the whole Tangle but will have much greater demands for computational power. An important aspect of future research will be the use of the Tangle as a publicly auditable and immutable "anchor" of integrity of data that will be stored off-Tangle in a database.

7. Conclusions

By using an innovative sensor node in an precision agriculture scenario, we were able to implement a simple sensor data marketplace and sensor data aggregation system which relies on and features data integrity and auditability. The proposed implementation is highly modular and an important contribution to the Open Source communities, both to those regarding precision agriculture as well as to those regarding blockchain. IOTA served as an ideal test-bed for the reference implementation, offering zero-fees which are ideal for the transaction throughput that this systems demands. This is due to the fact that the systems uses IOTA as the transport protocol of the sensor data.

This implementation showcases the architecture that we envision, where each IoT Node is able to interact with the Ledger, by logging its data directly on it, performing all the cryptographic functionalities. This will boost system's security, for its interaction with the public ledger will not depend on a trusted third party node, which can easily be manipulated and tamper with data. Future implementations will support multiple different DLT and a robust Edge Computing system that will serve as a smart aggregator of data from the IoT nodes.

The current implementation contributes to the Open Source Community by setting the infrastructure of an IoT suite with an embedded data marketplace. Depending on the field results, it is possible to either migrate the architecture to run on hardware nodes in a more lightweight fashion or migrate to an existing solution (e.g., EdgeX) and contribute to that Open Source project by creating the framework for a cryptocurrency based data marketplace.

Author Contributions: Conceptualization, O.L.; methodology, J.G.; software, D.P.; validation, O.L., J.G. and D.P.; formal analysis, O.L.; investigation, O.L. and D.P.; resources, O.L.; data curation, J.G., D.P. and O.L.;

writing–original draft preparation, O.L., J.G. and D.P.; writing–review and editing, O.L. and J.G.; visualization, O.L. and D.P.; supervision, O.L.; project administration, J.G.

Funding: This research has been co-financed by the European Union and Greek national funds namely the Regional Operational Program "Western Greece 2014-2020", under the Call "Regional research and innovation strategies for smart specialization (RIS3) in Microelectronics and Advanced Materials" (project: 5021449 entitled "Intelligent Services Based on the Internet of Things to Support Agriculture "AgrIoT").

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lamtzidis, O.; Gialelis, J. An IOTA Based Distributed Sensor Node System. In Proceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, UAE, 9–13 December 2018; pp. 1–6. [CrossRef]
2. The Next Generation of Distributed Ledger Technology|IOTA. Available online: https://www.iota.org/ (accessed on 28 June 2019).
3. Yonatan, S.; Lewenberg, Y.; Zohar, A. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *IACR Cryptol. ePrint Arch.* **2016**, *2016*, 1159.
4. Sompolinsky, Y.; Zohar, A. Phantom, Ghostdag. Available online: https://eprint.iacr.org/2018/104.pdf (accessed on 29 June 2019).
5. Serguei, P. The Tangle. Available online: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf (accessed on 29 June 2019).
6. Home|IOTA Documentation. Available online: https://docs.iota.org/ (accessed on 28 June 2019).
7. Johannes, B.; Dahmen, E.; Ereth, S.; Hülsing, A.; Rückert, M. On the security of the Winternitz one-time signature scheme. In Proceedings of the International Conference on Cryptology in Africa, Dakar, Senegal, 5–7 July 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 363–378.
8. MAM2 Specification. Available online: https://github.com/iotaledger/entangled/blob/develop/mam/spec.pdf (accessed on 29 June 2019).
9. Johannes, B.; García, L.C.C.; Dahmen, E.; Döring, M.; Klintsevich, E. CMSS—An improved Merkle signature scheme. In Proceedings of the International Conference on Cryptology in India, Kolkata, India, 11–13 December 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 349–363.
10. Jeffrey, H.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In Proceedings of the International Algorithmic Number Theory Symposium, Portland, OR, USA, 21–25 June 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
11. Protocol Buffers Version 2 Language Specification|Protocol Buffers|Google Developers. Available online: https://developers.google.com/protocol-buffers/docs/reference/proto2-spec (accessed on 28 June 2019).
12. Jesse, Y.; Ko, D.; Choi, S.; Park, S.; Smolander, K. Where is current research on blockchain technology?—A systematic review. *PLoS ONE* **2016**, *11*, E0163477.
13. Satoshi, N. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 29 June 2019).
14. Salomaa, A. *Public-Key Cryptography*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
15. Joseph, B.; Miller, A.; Clark, J.; Narayanan, A.; Kroll, J.A.; Felten, E.W. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 104–121.
16. Blockchain Architecture: The Basics|Pluralsight. Available online: https://www.pluralsight.com/guides/blockchain-architecture (accessed on 28 June 2019).
17. Buterin, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Available online: https://github.com/ethereum/wiki/wiki/White-Paper (accessed on 28 June 2019).
18. Li, X.; Jiang, P.; Chen, T.; Luo, X.; Wen, Q. A survey on the security of blockchain systems. *Future Gener. Comput. Syst.* **2017**, in press. [CrossRef]
19. EOSIO—Blockchain Software Architecture. Available online: https://eos.io/ (accessed on 29 June 2019).
20. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.

21. Hyperledger Fabric—Hyperledger. Available online: https://www.hyperledger.org/projects/fabric (accessed on 29 June 2019).
22. Genuino 101. Available online: https://store.arduino.cc/genuino-101 (accessed on 29 June 2019).
23. Orange Pi—Orangepi. Available online: http://www.orangepi.org/ (accessed on 29 June 2019).
24. Raspberry pi Home-page. Available online: https://www.raspberrypi.org/ (accessed on 29 June 2019).
25. CoAP—Constrained Application Protocol|Overview. Available online: http://coap.technology/ (accessed on 29 June 2019).
26. MQTT. Available online: http://mqtt.org/ (accessed on 29 June 2019).
27. RFC 1081—Post Office Protocol: Version 3. Available online: https://tools.ietf.org/html/rfc1081 (accessed on 29 June 2019).
28. Gialelis, J.; Gerasimos, T.; Maria, F.; Dimitrios, K. An Integrated Low Cost IoT Node based on Discrete Components for Customized Smart Applications; Use case on Precision Agriculture. In Proceedings of the 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 10–14 June 2019.
29. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future internet: The internet of things architecture, possible applications and key challenges. In Proceedings of the 2012 10th International Conference on Frontiers of Information Technology, Islamabad, India, 17–19 December 2012; pp. 257–260.
30. Internet of Things Forecast—Ericsson Mobility Report. Available online: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast (accessed on 29 June 2019).
31. Kshetri, N. Can blockchain strengthen the internet of things? *IT Prof.* **2017**, *19*, 68–72. [CrossRef]
32. Report from the Commission: Benchmarking Smart Metering Deployment in the EU-27 with a Focus on Electricity. Available online: https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters (accessed on 29 June 2019).
33. Library, C. Flint Water Crisis Fast Facts. Available online: https://edition.cnn.com/2016/03/04/us/flint-water-crisis-fast-facts/index.html (accessed on 29 June 2019).
34. Conoscenti, M.; Vetro, A.; Martin, J.C.D. Blockchain for the Internet of Things: A systematic literature review. In Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–6.
35. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623.
36. Micro-Insurance for Small Farmers—This Amazing Idea Is Changing Lives for Farmers in East Africa (Must Watch!)—Smallstarter Africa. Available online: https://www.smallstarter.com/get-inspired/micro-insurance-for-small-farmers-in-africa/ (accessed on 29 June 2019).
37. Qubic: Details on 03-06-2018. Available online: https://qubic.iota.org/ (accessed on 29 June 2019).
38. Tranoris, C. *Open Source Software Solutions Implementing a Reference IoT Architecture from the Things and Edge to the Cloud*; University of Patras: Patras, Greece, 2018. [CrossRef]
39. Access Tokens. Available online: https://auth0.com/docs/tokens/overview-access-tokens (accessed on 2 September 2019).
40. IOTA Tangle Explorer and Statistics—TheTangle.org. Available online: https://thetangle.org/ (accessed on 2 September 2019).
41. Thomas Pototschnig/Pidiver1.3. Available online: https://gitlab.com/microengineer18/pidiver1.3 (accessed on 2 September 2019).
42. DLTcollab/Dcurl. Available online: https://github.com/DLTcollab/dcurl (accessed on 2 September 2019).
43. Proof of Concept Codebase|Github. Available online: https://github.com/OdysLam/iota_paper_codebase (accessed on 29 June 2019).
44. Chart.js|Open Source HTML5 Charts for Your Website. Available online: https://www.chartjs.org/ (accessed on 29 June 2019).
45. Instant & Feeless—Flash Channels. Available online: https://blog.iota.org/instant-feeless-flash-channels-88572d9a4385 (accessed on 29 June 2019).
46. Coordinator-Part-1: The-Path-to-Coordicide. Available online: https://blog.iota.org/coordinator-part-1-the-path-to-coordicide-ee4148a8db08 (accessed on 29 June 2019).
47. Wall, E. IOTA is Centralized. Available online: https://medium.com/@ercwl/iota-is-centralized-6289246e7b4d (accessed on 29 June 2019).

48. Heilman, E.; Narula, N.; Tanzer, G.; Lovejoy, J.; Colavita, M.; Virza, M.; Dryja, T. Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 344.
49. Official IOTA Foundation Response to the Digital Currency Initiative at the MIT Media Lab. Available online: https://blog.iota.org/official-iota-foundation-response-to-the-digital-currency-initiative-at-the-mit-media-lab-part-4-11fdccc9eb6d (accessed on 29 June 2019).