Zero-Trust Hierarchical Management in IoT

Mayra Samaniego
Computer Science
University of Saskatchewan
Saskatoon, Canada
mayra.samaniego@usask.ca

Ralph Deters
Computer Science
University of Saskatchewan
Saskatoon, Canada
deters@cs.usask.ca

***Abstract*— Internet of Things (IoT) is experiencing exponential scalability. This scalability introduces new challenges regarding management of IoT networks. The question that emerges is how we can trust the constrained infrastructure that shortly is expected to be formed by millions of 'things.' The answer is not to trust. This research introduces Amatista, a blockchain-based middleware for management in IoT. Amatista presents a novel zero-trust hierarchical mining process that allows validating the infrastructure and transactions at different levels of trust. This research evaluates Amatista on Edison Arduino Boards.**

Keywords— Internet of Things, Blockchain, Hierarchical Mining, Zero-Trust Management, Edge Computing

I. INTRODUCTION

The Internet of Things (IoT) describes a network of embedded and pervasive sensors and actuators with heterogeneous and limited computing capabilities (things). Gartner [1] estimates that by 2020 there will be more than 20 billions of things connected to the Internet. This estimate has been observed by Howard [2]. IoT has experimented and exponential growth since 2015, involving more devices than people.

IoT applications offer new forms of communication and services for many areas. For example, IoT Cloud systems for health services [3], smart homes [4] or public services [5]. These systems have one characteristic in common. They work as unidirectional data-centric reading entities. These systems become consumers of data, not being aware of the validation of the constrained network that is producing the data. Cloud applications consider that the received data is reliable because they assume that the constrained network that sensed or processed it has the same trusted characteristics it had in the beginning when both layers (applications and things [6]) started working and collaborating.

The massive scalability of IoT networks introduces the need for new management controls regarding the infrastructure (physical or virtual resources that can be placed at the perception, edge, fog or cloud layer), verification of data provisioned by the infrastructure, and validation of transactions between resources that are part of the infrastructure at constrained and edge levels. Thus, we can have ubiquitous and autonomous IoT networks capable of validating their state and transactions.

Some works have implemented a zero-trust model in IoT. For instance, Saied et al. [7] present a trust management system against malicious attacks that works based on a context-aware approach. This solution requires a centralized validation authority. Our research shifts the focus and introduces distributed validation authorities that work based on a blockchain approach.

This research considers IoT a zero-trust environment because we cannot trust either the infrastructure or transactions executed by the infrastructure. To guarantee a reliable IoT network we have researched and developed a blockchain-based middleware that we call Amatista. Amatista follows a blockchain approach. However, Amatista goes beyond the traditional mining process [8] of blockchain [9] and presents a novel hierarchical mining approach, distributing the validation authority responsibilities not only horizontally but also vertically into different hierarchies of trust. Amatista handles the validation of the infrastructure and transactions at two levels of trust.

The rest of the paper is organized as follows. Section 2 studies trust management. We make a special focus on the IoT perspective. Section 3 explores the features of blockchain and how some of these features would be a good option for IoT. In section 4, we introduce Amatista for zero-trust hierarchical management in IoT. In section 5 we explain the architecture of Amatista. In section 6 we present the experiments and evaluations. Finally, conclusions and future work are presented in section 7.

II. TRUST MANAGEMENT

According to Eschenauer et al. [10], regarding network security, the idea of trust refers to a set of relations among entities (physical or virtual). There has to be evidence of created trust relations and evaluations of that evidence to accept a trust relation. A previously accepted trust relation can be used as evidence to accept new ones.

Some studies make emphasis on using trust management in distributed systems. Blaze et al. [11] implement trust management to improve the security of authorization mechanisms in distributed systems. Also, few studies have applied trust management for other purposes. For example, Chu et al. [12] apply the trust management concept to build a management system for Web applications. This system monitors all clients and servers trust decisions and controls that every action is performed under some policy.

Some studies have integrated a trust model in IoT as well. For example, Chen et al. [7] present an adaptive trust system for social IoT that uses validation identity and recommendation

978-1-5386-7244-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICIOT.2018.00019

information based on some trust concepts to deal with trust-related attacks like whitewashing or self-promoting. In this architecture, each node evaluates its trust compositions. There is no distributed agreement but a sharing of final evaluations.

A. Zero-Trust Model

The Zero Trust model [13] was introduced by the analyst firm Forrester Research in 2010 to confront new attack methods in information security. Behind this model there is the simple principle of "never trust, always verify." Thus, all data traffic generated must be untrusted, no matter if it has been generated from the internal or external network.

Zero-trust network security has already been used and implemented [14] by few companies. For example, the Cisco Application Centric Infrastructure (ACI) [15] Whitelist-Based Policy Model Supports Zero-Trust Security Architecture [16]. Cisco ACI does not trust new endpoints by default, but by checking a whitelist policy for connectivity. These policies (contracts) permit, deny, log, redirect, or instantiate traffic between two endpoints.

B. Zero-Trust Model in IoT

According to Yan et al.[17], a trust management framework for IoT networks has to cover the three main layers of IoT: perception, network and application [18]. The three layers should work as a whole to provide a reliable service. However, it is not enough that the three layers work in a reliable environment. A valid configuration of the infrastructure at each layer is a crucial element to provide reliable data.

From a things-centric IoT perspective [6], transactions (mostly data-centric readings) from services hosted in the Cloud are accepted at the constrained network without evidence of the current state of the requester node. It is assumed that a transaction that a specific user is trying to execute has always been valid and permitted. This trustworthiness means that as long as the user is part of the network, it is assumed that all transactions from it are good enough to avoid validation. However, there is the possibility that the permissions of users executing those transactions might be corrupted, which means that their transactions are corrupted as well. Uncommon transactions that do not make sense might be easily performed.

A zero-trust management model for IoT might help to guarantee that the credentials and configurations of every resource of the infrastructure are validated every time they wake up to participate in the network. Thus, cheating behaviour is prevented. Also, a zero-trust management model might help to guarantee that every message between resources is validated applying a security method (e.g. cryptography) to ensure that messages are not falsified. Finally, zero-trust management might help to ensure that the nature of every transaction is verified before executing it. Thus, unusual transactions are cached and discard.

However, a zero-trust management model alone would be an expensive option that might not be suitable to apply in IoT networks for the following reasons: lack of computational resources, lack of sufficient bandwidth, and need to preserve power networks where computing power is limited. We cannot have a central trust validation authority because we do not have enough resources at the edge level. We cannot validate every single transaction at one validation authority because the heterogeneity nature of communications in IoT would cause the network collapse.

This work evaluates the features of blockchain from a things-centric perspective. Thus, we can implement those features in a distributed manner towards edge devices and build a distributed middleware that fits the unique characteristics of IoT networks.

III. Blockchain for Management in IoT

Blockchain gained popularity when it was integrated by Bitcoin in 2009 [9]. Blockchain represents the public ledger that stores Bitcoin transactions in the form of blocks. Blocks are connected through a hash value forming a chain. A Blockchain is a peer-to-peer network, which allows the execution of direct transactions without any central verification authority.

Blockchain protocols have the following characteristics [19]:

Distributed Database: Blockchain distributes and synchronizes all transactions across the network. This approach encourages data sharing between organizations or between departments from the same organization. All nodes get a copy of the blockchain and are notified about new blocks.

Smart Contracts: A smart contract is a piece of code stored in the blockchain. Smart contracts can directly modify data, control the access privileges and trigger actions. Smart contracts define the type of transactions between participants. This enhances the security and reliability of transactions.

Consensus: Consensus is a mechanism to validate transactions. Transactions are validated by a consensus mechanism in which participants must invest computation. In Bitcoin, there is no consensus but a proof-of-work task based on cryptography hashes algorithms.

Immutability: After transactions are recorded in the blockchain, they cannot be changed. Thus, Blockchain provides provenance control to track assets.

The above-explained characteristics of blockchain might contribute to the efficient management of the IoT infrastructure at the edge level. For example, the Autonomous Decentralized Peer-to-Peer Telemetry (ADEPT) Proof of Concept (PoC) [20] is an example of the integration of blockchain in IoT for autonomic systems. ADEPT has a hybrid and decentralized architecture. ADEPT uses Ethereum [21] blockchain for autonomous device coordination functions such as storing the configuration of devices and authentication.

In our previous studies, we have integrated blockchain protocols at the Fog level to handle some management tasks close to the constrained network. For instance, blockchain to store the configuration of virtual resources [22][23]. Also, we have used blockchain as a mechanism to distribute software-defined components towards the edge network to support

multitenancy and some AI features [24][25]. Thus, we have made the constrained network more autonomous. However, until now, we have focused on the implementation of an entire blockchain technology at the Fog level. We have been limited by the lack of resources at the edge level.

Even though blockchain might be used to implement a zero-trust management model, the way in which blockchain protocols achieve their features would have to be modified for IoT networks. For example, Bitcoin, miners are powerful machines performing a proof-of-work task to demonstrate they are trustworthy nodes and get their transactions accepted as valid. In IoT, edge miners do not have the same computation capabilities than Bitcoin miners. Thus, the mining process has to be done differently. We could choose to implement any other consensus algorithm like a Practical Byzantine Fault Tolerance (PBFT) algorithm [26] or Round Robin (RR) scheduling [27], but the velocity of transactions would cause edge miners to collapse. Thus, an additional evaluation criterion has to be added. This research adds a context evaluation that would allow miners to validate transactions generated only by nodes under their context.

Blockchain provides a distributed mining process. However, this mining process is performed at one level (horizontally) by all nodes in the network. In IoT, devices mining every single transaction would cause the propagation process to fail. In the same manner that we add edge and Fog layers for data processing, it is necessary to add layers to mine transactions at different levels (vertically). This research adds a second layer to mine transactions. Thus, we get a horizontal and vertical mining process.

The immutability of transactions and the decentralized architecture of blockchain are the perfect combination to provide reliable software components in IoT. However, the high demand of computing power required to make the previous two features work, make its implementation in IoT a challenge. This research adds a publish/subscribe policy to segment the in-chain storage and data propagation.

According to Biswas and Giaffreda [28], a potential benefit for IoT is building software-defined ecosystems or virtual systems. This research extends that view and proposes that software-defined ecosystems that integrate concepts from already developed technologies such as blockchain can help to deal with the management of IoT networks, in this case, the infrastructure and transaction management based on trust. Following the ADEPT PoC idea of integrating blockchain to build hybrid and decentralized applications, this work seeks to develop a hierarchical and decentralized middleware for zero-trust management in IoT, Amatista.

Amatista implements some blockchain features separately at different layers. Thus, the mining process can be handled by software components hosted on different physical edge devices. This approach improves the performance of the constrained network.

IV. AMATISTA FOR ZERO-TRUST HIERARCHICAL MANAGEMENT IN IOT

This research introduces a blockchain-based middleware for managing the IoT infrastructure and transactions in a zero-trust manner. We call this middleware Amatista. Amatista integrates the key features of blockchain: distributed database, consensus, smart contracts and immutability, to guarantee a reliable constrained infrastructure and transactions. Amatista adapts these features to deal with the nature of IoT.

Amatista does not trust either the infrastructure or transactions. This blockchain-based approach eliminates the need for centralized validation nodes. Amatista not only validates participant resources but also the transactions produced by them. Thus, the flow is not only a data-centric reading but a resource-centric communication in which managed resources can be accessed without the presence of a central authority commonly used. Additionally, Amatista introduces a context parameter to the hierarchical mining process. Thus, the mining process is segmented.

A. Hierarchical Mining

The emblematic characteristic of Amatista is the hierarchical mining process. This research defines hierarchical mining IoT as the process of validating the physical or virtual infrastructure and transactions at various levels of trust by different resources working in a specific context.

Amatista provides this hierarchical mining by integrating distributed validation authorities, which we call miners (following the Bitcoin nomenclature), placed at the edge level. The hierarchical mining is performed before data is sent to observer parties.

Trust Composition

First-Level Mining

The first-level mining involves trusting the identity and the context of the resources that send the transactions. Amatista does not trust resources directly, but the provenance information stored in chain previously signed and validated locally. For instance, this resource can be an observed constrained component updating its state, a third party resource from the Cloud or Fog network trying to access to the constrained network or a virtual resource that is collaborating in the internal network.

At this level, the mining process validates the infrastructure locally where smart contracts are triggered when observed resources change their state. The resulting data is provisioned to miners in an encrypted key/value manner. Data of the mined blocks are stored in-chain locally. Also, provenance off-chain data is stored in a key/value manner in a Fog resource.

Second-Level Mining

The second-level mining involves trusting transactions. Amatista does not trust the user that is sending the transaction, but the consensus mechanism that peers achieve to guarantee those transactions. At this level, the mining process is achieved

under a specific context. Miners validate blocks that are signed by first-level miners that belong to the same context.

Each miner validates transactions collaboratively and propagates the results to the other miners. Once miners achieve consensus, they update the state of the transactions to an approved or valid state. Thus, Amatista makes edge resources become distributed validation authorities.

The manner in which Amatista works achieves two levels of mining. The traditional horizontal consensus introduced by blockchain protocols plus a vertical validation hierarchy. Hierarchical mining enhances security in IoT networks as the legitimacy of resources and transactions are evaluated by distributed authorities.

B. *Segmented Data Provisioning*

Amatista provides a segmented data provisioning service by implementing a publish/subscribe policy. Similar to the Hyperledger Fabric concept of Chain Channels [29] that isolates data to specific receivers, Amatista allows that any virtual or physical resource can subscribe to receive updated data from specific resources. In Hyperledger Fabric, data is redirected to specific groups, not to individual users. In Amatista, each resource or user can have their database of observed resources. This approach ensures that resources receive specific data no matter the context they belong. Also, small blocks of data are moved faster towards the specific subscribed edge nodes improving the performance of the network.

V. ARCHITECTURE OF AMATISTA

The architecture of Amatista is presented in Figure 1. This is a blockchain-based middleware that implements the mining process in a hierarchical manner. Amatista follows a REST design. Amatista has been designed to address the lack of engagement with constrained networks by representing each miner as an independent full-state resource that can be engaged through a URL. The high-level abstraction of Amatista is presented in the upcoming subsections.

First-Level Miner

First-level miners have four components. First, a CoAP interface to engage sensors. This resource observes (CoAP observing pattern [30]) sensors and receives updates of transactions any time their state has changed.

Second, a validator component to validate the context of the node sending the transaction. This mining component works locally and validates that the resource identification matches with the correct one obtained from the in-chain provenance storage. This component works with a local key/value table and a Fog storage resource. Smart contracts are also configured here to manage the results of the mining process, such as provisioning those results to subscribed observers, sending notifications to higher miners or sending alarms to other observer parties.

Third, a block fabric component. This component builds blocks of transactions from validated nodes. Once nodes have been trusted, blocks of transactions are built and sent to the second-level mining. Blocks are encrypted and signed by the miner using a private key. The corresponding miner's public key is used in the second level of trust to decrypt the block.

Finally, a key/value table to store necessary metadata of built blocks. This component helps to improve latency when different parties query the same data. A light version of the table with fewer registries is stored at the edge resource, and a complete hash table is stored in a Fog level, close to the edge. Each miner has a copy of the most recent metadata of the table.

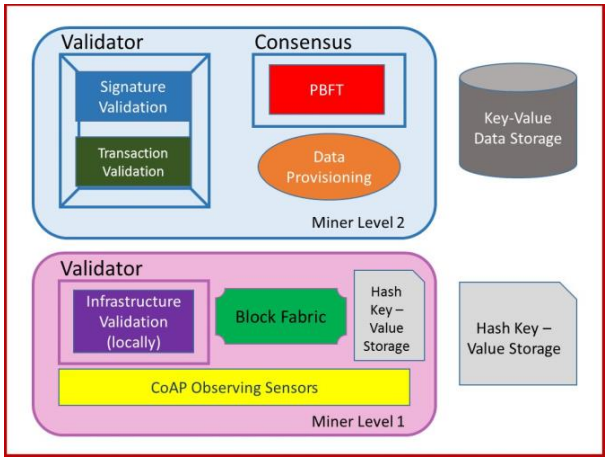

Figure 1.The high-level architecture of Amatista

Second-Level Miner

A second-level miner has four components. First, a validator component to validate the legitimacy of every block received. This resource evaluates the identity of the user that signs the block. This component uses asymmetric digital signatures and public/private mechanisms to trust first-level miners.

Second, a consensus component. This component implements a consensus algorithm to agree on all transactions before they are executed and stored. The consensus algorithm implemented by Amatista is the Practical Byzantine Fault Tolerance (PBFT) [26]. In the common blockchain approach, all miners participate in the consensus algorithm. However, in Amatista, miners at this level of trust validate only blocks received from first-level miners under the same context. This component verifies the validity of each transaction in the received block. This is whether the transaction makes sense to the context in which the device is currently working. This approach optimizes the utilization of computing resources.

Third, a data provisioning component. This component works under a publish/subscribe policy and distributes data to specific subscribers. Even though a miner is not part of the consensus process; it can subscribe to observe the results of the consensus.

Finally, Fog storage, which stores data off-chain and is not part of the edge miner but works close to it.

A. Technology & Communication

Amatista has been developed using Go Language [31]. Each component of Amatista has been modelled as a full-state resource that can be accessed through an URL.

The communication between miners and sensors is based on the Representational State Transfer (REST) design concept [33]. REST focuses on managing the state of resources, which allows getting dynamic interactions with sensors. Constrained components can be represented as full state resources, and their interaction can be mapped to CRUD operations. We have found particularly interesting the fact of using REST to model hierarchies of trust.

Amatista implements the Constrained Application Protocol (CoAP) [34] to communicate first-level miners with the perception layer. CoAP follows the REST approach exchanging representations of resources. Like in HTTP, the most used methods in CoAP are GET, POST, PUT, DELETE.

VI. Evaluation of Amatista

This section presents the evaluation of Amatista. An environment with two layers has been set up (Figure 2). First, a Things network. As IoT sensors, we have used Intel Edison System on a Chip (SoC) modules [32] plugged on Arduino Boards [33] (Table I). Sensors have been programmed using GOBOT [34].

Second, an edge network. Edge miners have also been deployed on Edison SoC modules but plugged on Spark modules [35]. Two Edison modules work at the first-level mining, and three modules work at the second-level mining.

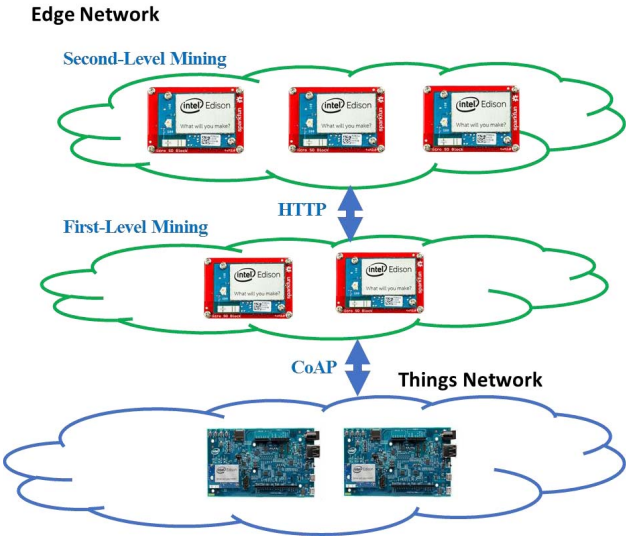

Figure 2. Experiment setup

TABLE I. Specifications of Edison SoC [32]

Edison System on a Chip	
Operating System	Linux Yocto
CPU	500 MHz dual-core, dual threaded Intel Atom and a 100 MHz 32-bit Intel Quark microcontroller
RAM	4GB LPDDR2 SDRAM

A. First-Level Miners

We have evaluated first-level miners under three delay intervals (0, 100, and 200ms). Figures 3 shows the time it takes to complete the first-level mining. In the Graphs, the X-axis represents every transaction sent from sensors. For this experiment, sensors sent 1000 transactions. Figure 4 shows an example of a JSON transaction from a sensor. Each transaction has been encrypted using the Advanced Encryption Standard (AES) [36] with a key of 16 bytes. The payload size of each request is 106 bytes.

The Y-axis represents the time that Amatista takes to do the first-level mining. The graphs show the performance of miners under the three delay intervals. The results obtained show that miners have a constant performance pattern. The initial high picks are presented due to the discovery process of first-level and second-level miners. The other picks observed in the graphs are presented because the block factory starts every 50ms, which affects the processing time.

The mining process includes the context validation of each transaction, which is 0.01ms average. The delay intervals in which transaction are sent from devices do not affect the execution time of the context validation. Thus, any change does not make a significant difference. The context validation is a local execution, that is why we get such low response times.

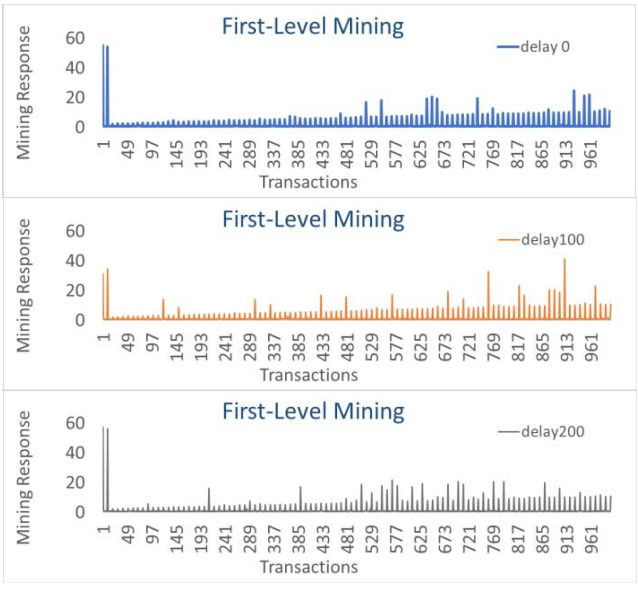

Figure 3. Evaluation of the First-Level Mining Process.

```
{
    "id":"ES-386",
    "t":50,
    "d":"2017-09-05 23:01:32.565410835"
}
```


Figure 4. Example of an unencrypted sensor transaction

Figure 5 shows the structure of a block built in the block factory. Amatista represents each block as a JSON structure in which all transactions (or list of transactions) are encrypted using AES encryption with a key of 16 bytes. With no delay in the generation of transactions from devices, the size of the message of the block was between 267 bytes and 1516. With 100ms delay the size was between 397 and 1759 bytes. With 200ms delay the size was between 268 and 1288 bytes. These ranges are caused because there are some HTTP requests from devices that get time-out responses, which makes the in-chain transactions in the first-level miner grow.

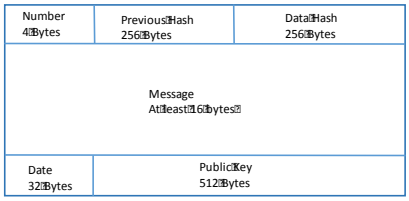

Figure 5. Structure of a block built in Amatista

The average processing time of the first-level mining is 1.63ms with no delay, 1.70ms with 100ms delay, and 1.76ms with 200ms delay. These results show that as the delay interval between transactions increases, so does the processing time in the miner. This is because the delay interval causes smaller messages at the block factory but more blocks with a higher demand of the encryption process.

With a 0ms delay, the number of blocks that the block factory built was 30. With a 100ms delay, the number of blocks was 170. With a 200ms delay, the number of blocks was 380. This is because as the delay interval increases, then the block-factory spends less time building blocks. Some of the blocks were sent with an empty message because the factory activates every 50 ms. We will include a specific factory rule in future implementations of Amatista.

B. Second-Level Miners

Figure 6 shows the time it takes for the three miners to achieve consensus. The X-axis represents the number of blocks received from the first-level mining. The Y-axis represents the consensus processing time. The graphs show that the time to achieve consensus follow the same pattern, it increases when more blocks are received. This is mainly because the in-memory chain of processed blocks affects the processing time of the device.

With no delay, there was an average of 20 transactions detected as unusual. With a delay of 100ms, there was an average of 58 transactions detected as unusual. With a delay of 200ms, there was an average of 80 transactions detected as unusual. Those transactions were not processed. For this experiment, a transaction was considered as unusual when the temperature reading exceeds a high constant value. Transactions were randomly injected with these fake values.

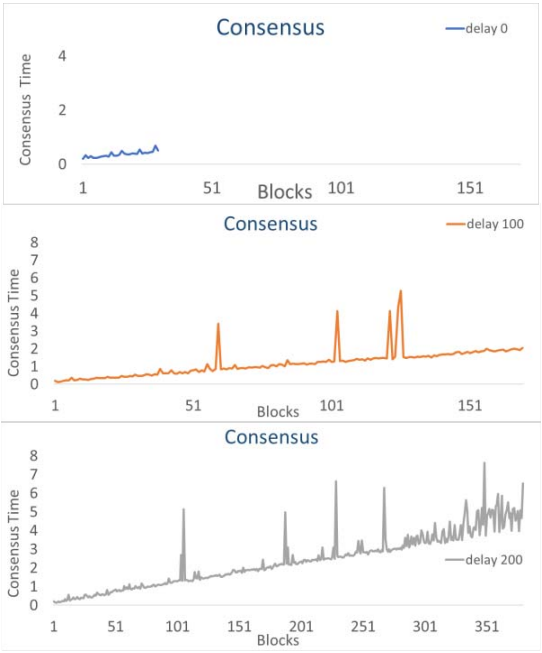

Figure 6. Evaluation of the Consensus Process.

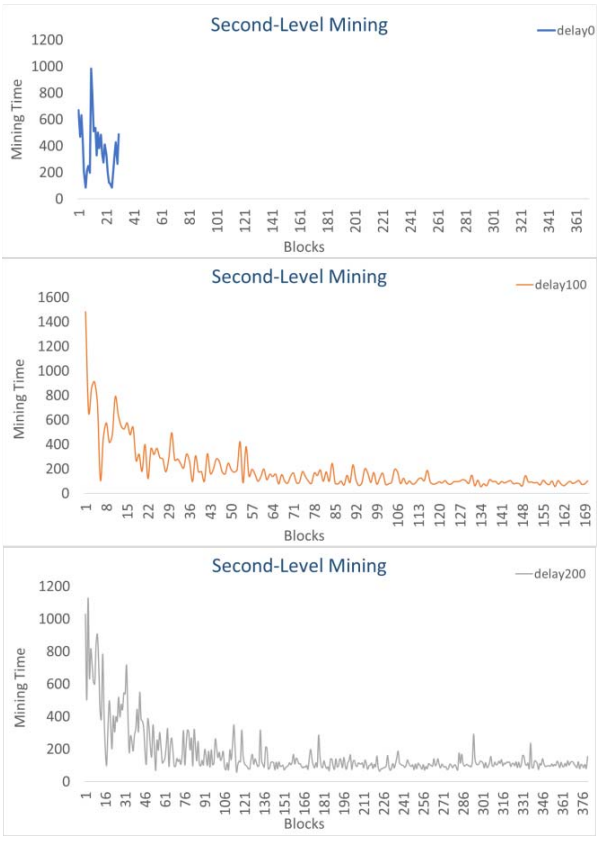

Figure 7. Evaluation of the Second-Level Mining Process.

Figure 7 shows the time it takes to complete the second-level mining. In the graph, the X-axis represents the blocks received from the factory in the first-level under different delay intervals. The Y-axis represents the time that the entire mining process takes. The results obtained confirm that with a 0 delay, only 30 blocks were processed by second-level miners. With a 100 ms delay, 170 blocks were processed. Finally, with a 200ms delay, 380 blocks were processed. Additionally, the graph shows some high picks in the beginning. This is because when the mining starts, there is no local in-chain data about the nodes that sign the transactions. Thus, before propagating the consensus result, each miner must send reading transactions to the chain. Also, in the beginning, the discovery process of miners is performed, which adds processing time as well. The response time stabilizes after the block 60 when the delay is 100ms and after the block 100 when the delay is 200ms. The average processing time of the second-level mining is 375.99ms with no delay, 193.18ms with 100ms delay, and 167.17ms with 200ms delay.

It is necessary to highlight that all the data presented in the figures mentioned above already includes the time for encryption and decryption at each mining level and even at the things layer. The above-presented evaluations cover from the time individual transactions are produced in sensors until the hierarchical mining process has been completed, and data has been propagated. Amatista presents a satisfactory performance considering all the hierarchical processing and encryption/decryption of data that occurs. Our previous studies of Bluemix IBM denote responses times up to 2500 ms from a Cloud implementation. Also, our previous studies of Multichain blockchain denote response times of up to 1000 ms [37] from a Fog implementation.

The experiments presented above do not include the time to engage the storages at the Fog level.

A subsequence simulation work is being developed to test Amatista in a large scale IoT deployment. The next step is simulating a large-scale attack scenario.

VII. CONCLUSIONS

This research introduced Amatista, a blockchain-based middleware to handle the infrastructure and transactions management of IoT networks in a zero-trust manner. Amatista addresses two challenges that emerge due to the massive scalability of IoT networks:

- Introduce zero-trust hierarchical mining to validate the infrastructure and transactions at two levels of trust, horizontal and vertical, at the edge level. Thus, the mining task is distributed towards edge components. This approach eliminates the limitation of constrained computing capabilities.
- Improve the performance of the networks by integrating a context parameter and a publish/subscribe policy that allows segmenting the mining process and the provisioning of data.

By integrating blockchain features to Amatista, the IoT focus is shifted away from centralized architectures to hierarchies of authorities.

The results of the evaluations showed that Amatista is capable of validating not only the infrastructure but also transactions in a hierarchical manner. Amatista demonstrated a satisfactory performance when validating transactions produced by sensors and blocks built in the block factory.

Overall, this research made the following contributions to IoT systems:

1) Designed and developed Amatista.
2) Introduced a novel hierarchical mining concept, which integrates distributed validation authorities for IoT at two levels of trust.
3) Introduced a context parameter in the mining process. Miners can mine nodes under a specific context.
4) Introduced a publish/subscribe provisioning of data. Nodes can select which results to observe.

Future studies can take Amatista as the foundation to implement other hierarchies of validation, consensus algorithms and more specialized smart contracts for IoT.

REFERENCES

[1] J. Rivera and R. Van der Muelen, "Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020," *Gartner*, 2013. [Online]. Available: https://www.gartner.com/doc/2625419/forecast-internet-things-worldwide-. [Accessed: 26-Nov-2015].

[2] P. N. Howard, "Sketching out the Internet of Things trendline." [Online]. Available: https://www.brookings.edu/blog/techtank/2015/06/09/sketching-out-the-internet-of-things-trendline/. [Accessed: 05-Aug-2017].

[3] Z. Pang *et al.*, "Design of a terminal solution for integration of in-home health care devices and services towards the Internet-of-Things," *Enterp. Inf. Syst.*, vol. 9, no. 1, pp. 86–116, 2015.

[4] M. Darianian and M. P. Michael, "Smart home mobile RFID-based internet-of-things systems and services," *Proc. - 2008 Int. Conf. Adv. Comput. Theory Eng. ICACTE 2008*, pp. 116–120, 2008.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[7] Y. Ben Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the Internet of Things: A context-aware and multi-service approach," *Comput. Secur.*, vol. 39, no. PART B, pp. 351–365, 2013.

[8] "Mining - Bitcoin Wiki." [Online]. Available: https://en.bitcoin.it/wiki/Mining. [Accessed: 04-Aug-2017].

[9] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008.

[10] L. Eschenauer, V. D. Gligor, and J. Baras, "On Trust Establishment in Mobile As-Hoc Networks," *GerhardGoosJurisHartmanisJanLeeuwenLecture Notes Comput. Sci.*, vol. 2845, no. Chapter 6, pp. 47–66, 2004.

[11] M. Blaze, J. Feigenbaum, J. Loannidis, and A. D. Keromytis, "The Role of Trust Management in Distributed Systems Security."

[12] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "REFEREE : Trust Management for Web Applications,"

Comput. Networks ISDN Syst., vol. 29, no. 8–13, pp. 953–964, 1997.

[13] J. Kindervag, "No More Chewy Centers : Introducing The Zero Trust Model Of Information Security," pp. 1–15, 2010.

[14] K. Townsend, "Don't implement zero-trust security in a virtualized network without reading this overview - TechRepublic," 2015. [Online]. Available: https://www.techrepublic.com/article/dont-implement-zero-trust-security-in-a-virtualized-network-without-reading-this-overview/. [Accessed: 31-Jan-2018].

[15] "Cisco Application Policy Infrastructure Controller Data Center Policy Model - Cisco," 2014. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731310.html. [Accessed: 31-Jan-2018].

[16] "Cisco ACI Security: A New Approach to Secure the Next-Generation Data Center - Cisco," 2014. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-732354.html. [Accessed: 31-Jan-2018].

[17] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, 2014.

[18] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of the Internet of Things," *Adv. Comput. Theory Eng. (ICACTE), 2010 3rd Int. Conf. on.*, vol. 5, pp. 484–487, 2010.

[19] I. Pattison, "4 characteristics that set blockchain apart - Cloud computing news," 2017. [Online]. Available: https://www.ibm.com/blogs/cloud-computing/2017/04/characteristics-blockchain/. [Accessed: 01-Feb-2018].

[20] V. Pureswaran, S. Panikkar, S. Nair, and P. Brody, "Empowering the Edge: Practical Insights on a Decentralized Internet of Things," 2015. [Online]. Available: https://www-935.ibm.com/services/multimedia/GBE03662USEN.pdf. [Accessed: 22-Nov-2016].

[21] "Ethereum Project." [Online]. Available: https://www.ethereum.org/. [Accessed: 18-Mar-2017].

[22] M. Samaniego and R. Deters, "Hosting Virtual IoT Resources on Edge-Hosts with Blockchain," pp. 116–119, 2016.

[23] M. Samaniego and R. Deters, "Virtual Resources & Blockchain for Configuration Management in IoT," *J. Ubiquitous Syst. Pervasive Networks*, vol. 9, no. 2, pp. 1–13, 2017.

[24] M. Samaniego and R. Deters, "Supporting IoT Multi-Tenancy on Edge Devices," *Proc. - 2016 IEEE Int. Conf. Internet Things; IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*, vol. 7, pp. 66–73, 2017.

[25] M. Samaniego and R. Deters, "Using Blockchain to push Software-Defined IoT Components onto Edge Hosts," in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, 2016, p. 58.

[26] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," *Proc. Symp. Oper. Syst. Des. Implement.*, no. February, pp. 1–14, 1999.

[27] W. Assumptions, "Scheduling: Introduction." [Online]. Available: http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched.pdf. [Accessed: 08-Oct-2016].

[28] A. R. Biswas and R. Giaffreda, "IoT and Cloud Convergence: Opportunities and Challenges," *2014 IEEE World Forum Internet Things*, pp. 375–376, 2014.

[29] "Introduction — hyperledger-fabricdocs master documentation." [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/latest/channels.html. [Accessed: 03-Aug-2017].

[30] "RFC 7641 - Observing Resources in the Constrained Application Protocol • CoAP, ," 2015.

[31] Golang.org, "The Go Programming Language," 2016. [Online]. Available: https://golang.org/pkg/crypto/rsa/#example_EncryptOAEP. [Accessed: 13-Aug-2017].

[32] INTEL, "The Intel® Edison Module | IoT | Intel® Software," 2017. [Online]. Available: https://software.intel.com/en-us/iot/hardware/edison. [Accessed: 02-Sep-2017].

[33] "No Title." [Online]. Available: https://www.arduino.cc/en/ArduinoCertified/IntelEdison. [Accessed: 02-Sep-2017].

[34] "Gobot - Golang framework for robotics, drones, and the Internet of Things (IoT)." [Online]. Available: https://gobot.io/. [Accessed: 01-May-2017].

[35] S. Ingenuity, "SparkFun Electronics," 2009. [Online]. Available: https://www.sparkfun.com/. [Accessed: 02-Sep-2017].

[36] "AES encryption." [Online]. Available: http://aesencryption.net/. [Accessed: 22-Nov-2016].

[37] M. Samaniego and R. Deters, "Blockchain as a Service for IoT," in *2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*, 2017, pp. 433–436.