Secure, Mass Web of Things Actuation Using Smart Contracts-Based Digital Twins

Iakovos Pittaras, Nikos Fotiou, Christos Karapapas, Vasilios A. Siris, George C. Polyzos
Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{pittaras,fotiou,karapapas,vsiris,polyzos}@aueb.gr

***Abstract*—The proliferation of Internet of Things (IoT) devices and applications that need to cooperate unattended highlights the need for seamless interoperability and intrinsic security. We argue that Distributed Ledger Technologies (DLTs), due to their decentralized nature, transparent operations, immutability, and availability, can enhance the security, reliability, and interoperability of such IoT systems. In this paper, we advance the integration of W3C's Web of Things (WoT) standards with DLTs and smart contracts, introducing smart contracts as "Digital Twins" of (physical) devices, or whole Cyber-Physical subsystems. Namely, we introduce a DLT-based architecture for controlling devices across federated IoT systems, securely, reliably, and with full auditability. The proposed architecture provides mass actuation and service composition with notable security properties, such as full auditability, transparency, and high availability. Specifically, a single request, with multiple action parameters and conditions, can trigger the reliable and secure actuation of a large number of possibly physically dispersed actuators.**

***Index Terms*—Digital Twins, DLTs, Blockchain, IoT, WoT, Ethereum, Interoperability, Auditability, Availability**

I. Introduction

The Internet of Things (IoT) is envisioned to be an ecosystem of interconnected devices that can collect, share, and act on data. The continuously increasing number of IoT devices creates opportunities for new applications, which merge the cyber with the physical world. However, developing such applications is not a trivial task, mainly due to different (and often competing) protocols and standards. This diversity, combined with business logic, and various policy requirements and regulations, results in IoT systems being mostly isolated and fragmented. In order to deal with these challenges, the Web of Things (WoT) W3C working group [1] leverages Web technologies to implement an interoperable IoT architecture. The WoT architecture builds on widely used Web standards and technologies, and enables IoT devices discovery and access using REST APIs, over popular application layer protocols (such as HTTP(s)).

The environment in which IoT devices need to operate is often unpredictable and continuously changing, and since the real world can be directly impacted, security and safety are a serious concern. Often, auditability is desired and can even be the last resort for recovering from a security incident. One approach for coping with the insecurity of IoT devices is the use of digital twins [2], i.e., a virtual representation of the IoT device stored in a more powerful and secure network location (e.g., a Web server); users instead of interacting with the actual device are interacting with its digital twin. All (valid) state modifications of the virtual twin are securely transmitted to the actual device.

Many companies, such as Amazon[1] and Microsoft,[2] are already providing such Cloud-based services. Nevertheless, these solutions are vendor specific and result in "vendor lock-in" situations. Additionally, these services lack transparency: given the pervasiveness of IoT systems and their access to sensitive information, security and privacy concerns arise. Furthermore, these systems may occasionally suffer from outages making IoT devices inaccessible [3], [4]. To address these issues, we explore how Distributed Ledger Technologies (DLTs) can be used for building secure and reliable digital twins that offer transparency, availability, and auditability.

Our solution uses the Ethereum blockchain [5] and leverages smart contracts to provide an event-based system for coordinated IoT operations, a medium for exchanging "tokens" that grant access to IoT devices, as well as a logging service of user actions. With the proposed system, we are making the following contributions:

- We enable smart contract-based digital twins of WoT "virtual entities", i.e., entities that integrate one or more IoT devices, and we allow "consumers" to execute "actions" on these devices.
- We facilitate IoT device management, mass actuation, and service orchestration by making consumers oblivious to the actual IoT devices.
- We allow device "owners" to define a cost for each provided action. This cost is measured in owner-specific Ethereum tokens, which can be sold (including offline using fiat currency), enabling novel business models.
- We design, develop, and evaluate our proposed system in a proof of concept implementation.

Compared to existing approaches, our solution improves interoperability by adopting the WoT architecture, enhances security by removing the need for a trusted entity that "hosts" the digital twin, and increases system availability by implementing its core functionality in a smart contract. In addition, by leveraging the event-driven communication provided by

[1]https://aws.amazon.com/iot-core/?c=i&sec=srv
[2]https://azure.microsoft.com/en-us/services/digital-twins

Ethereum, our solution enables mass actuation, in the sense that a user can interact with multiple IoT devices by sending just a single transaction, and it facilitates service composition. Namely, one action can trigger the secure actuation of multiple smaller and diverse actions. Furthermore, users are IoT device vendor-agnostic, since they communicate with the digital twin and not the actual devices. Finally, by using custom Ethereum tokens as a means for paying for the provided services, not only we enable novel business models, but we also allow using our system in a private Ethereum deployment.[3]

A. Usage Scenarios

As the IoT becomes more and more popular, we observe a new usage pattern, in which IoT devices and systems are deployed in multi-tenant environments, e.g., a business environment, where many different and untrusted parties interact with them. Our proposed system is focused on securing distributed IoT devices and services that are accessed by many untrusted parties. For illustration, we will use a smart building as a usage scenario.

Consider a smart building, owned and managed by a company, referred to here as the building manager. The building is composed of multiple floors, with portions of each floor leased out to tenants by the building manager. The building manager is the owner of the resources (IoT devices) deployed on the shared areas of the smart building. The building manager could allow tenants to interact with the IoT devices of the building, under a defined cost. The building manager could also grant ephemeral permissions to guests. This could depict a real-life scenario occurring in many cases, such as: (a) in smart homes, where the homeowner installs IoT devices and he/she should allow to the rest of the family or to guests to interact with them, or (b) in big business buildings, where employees or users from different organizations need to interact with IoT devices that are common for all the tenants of the building.

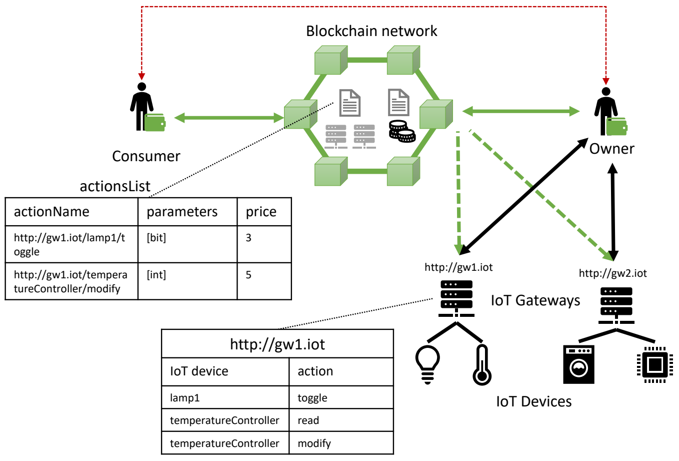
II. Related Work

Blockchain technology has been included in or considered for many IoT-based systems, e.g., for the energy sector [6], supply chains [7], and gaming [8], among others. Earlier work by two of the authors [9] discusses how blockchains and DLTs can provide novel security mechanisms for the IoT. They also claim that the use of smart contracts can automate the interaction with IoT devices. All these efforts highlight the advantages of using blockchains in IoT systems, in particular in "verticals" and they use a "clean-slate" approach. In our work, we integrate blockchain technology with a standardized technology, W3C's WoT, targeting interoperability and "horizontal" applications.

Digital twins are getting more and more attention. There are several research efforts that try to integrate digital twins in IoT applications and use cases. Chevallier et al. [10] present a reference architecture for creating and managing digital twins for smart buildings. The proposed digital twin and its data, stored in a database, are used for managing and monitoring the building. Liu et al. [11] propose a framework for an indoor safety management system that is based on digital twins. Authors, using the Building Information Modelling (BIM) and data collected by IoT sensors, develop a digital twin, which is used by a Support Vector Machine (SVM) to automatically obtain the types and levels of danger. Furthermore, Mohamadi et al. [12] propose a smart city digital twin paradigm. The digital twin is progressively updated with real time data by the smart city's systems in order to become "smarter" and able to predict the city's performance and growth. Simillarly, White et al. [13] design a digital twin for smart cities. The proposed digital twin is composed of 6 layers. The final layer, which is the digital twin, is built on the data that is produced from all the other layers. Then, authors use the digital twin to conduct various simulations (flooding simulation, crowd simulation, etc.) in order to extract information about the city. These efforts design and implement digital twins in order to use them as monitoring tools and for performing various simulations, which cannot be performed in the actual cyber-physical systems. However, in our work, we are using the digital twin as an indirection of the actual IoT devices. We are using them to perform real actions on the IoT devices, instead of extracting information about them.

Other recent research efforts try to combine digital twins with DLT–some such efforts are briefly discussed next. Yaqoob et al. [14] present some possible use cases and applications, architectures, and tool sets that can enhance digital twins to be more effective in real-life industrial problems; to this end, they propose the integration of the blockchain technology into digital twins by suggesting that it can be used as a storage place for the digital twin's data. Huang et al. [15] propose a system based on a blockchain to address the data management problems of digital twins used in product lifecycle management. The authors exploit the peer-to-peer network of the blockchain to connect all the interested parties. The network is also used to share the data securely among the participants. Furthermore, the authors use the blockchain to store all the data along with a timestamp, e.g., all the actions of the digital twins, in blocks. Lee et al. [16] propose a framework that exploits digital twins and blockchains for traceable data communication in construction projects. The framework selectively stores, in the blockchain, and share important project-related information traceably. Shen et al. [17] propose a blockchain-based framework for secure sharing of big digital twin data. The authors use Cloud storage to store the encrypted big digital twin data, while the hash of the big data is stored on the blockchain. Nielsen et al. [18] developed a prototype digital twin, which is connected with the Ethereum blockchain. In particular, they have implemented Non-Fungible Tokens (NFTs) in Ethereum, in order to represent physical devices in the blockchain. Then, each token is configured with data from the actual device by the corresponding digital twin. All these efforts use blockchain technology for providing auxiliary functionality, such as data storage, or as a means for data sharing, whereas our proposed architecture realizes the actual

[3]Other DLTs supporting smart contracts or equivalent constructs could obviously be used instead.



actionName	parameters	price
http://gw1.iot/lamp1/toggle	[bit]	3
http://gw1.iot/temperatureController/modify	[int]	5

http://gw1.iot	
IoT device	action
lamp1	toggle
temperatureController	read
temperatureController	modify

Fig. 1. System overview.

digital twin on the blockchain as a smart contract to achieve its goals, really integrating DLT and digital twin functionality. While the goal of previous proposals in the abovementioned works is to secure the data generated by the devices, our goal is to secure the actual IoT devices, ensuring their correct functioning and availability.

III. System overview

In this section, we present an overview of our solution. Our proposed system is composed of the following entities (see also Figure 1):

- *IoT gateways* and *IoT devices*
- The smart contract-based digital twin of a "virtual entity"
- A smart contract that manages the owner's specific tokens
- An *owner* (*building manager*) that administrates the gateways and the IoT devices
- *Consumers* (*tenants*) that interact with the IoT devices

All gateways together compose a WoT "virtual entity". A virtual entity is the composition of one or more IoT devices, e.g., a room consisting of several IoT devices. The "virtual entity" provides a single WoT Thing Description (TD) [19] that contains the capabilities of all IoT devices. As IoT devices, we consider actuators, although sensors can easily be handled by our system as well.

Consumers do not access IoT gateways/devices directly, instead they interact with the digital twin of the virtual entity implemented as a smart contract and deployed on the Ethereum blockchain. Thus, consumers should own a public/private key pair used for signing transactions sent to the blockchain. On the contrary, the owner can access and configure her IoT devices and the corresponding gateways directly.

From a high level perspective, the entities in our system interact with each other as follows. Initially, the owner physically deploys the IoT devices and pairs them with a gateway. Then, she creates the digital twin of the virtual entity, composed of all IoT devices, and she configures the smart contract accordingly. The virtual entity includes *actions* that can be implemented by a single IoT device or by orchestrating multiple IoT devices. Similarly, an action may correspond to multiple interactions with an IoT device. For example, an action "turn on all the lights of the floor" of a virtual entity, may result in instructing multiple light bulbs to be switched on. A gateway is responsible for mapping an action of the virtual entity to the corresponding action(s) of the real IoT device(s).

A consumer can gain access to a virtual entity, hence to (specific) IoT devices, by obtaining some owner-specific Ethereum *tokens*. When a consumer wants to perform an actuation operation, he deposits some tokens to the smart contract, which are held as escrow, and he sends a transaction that includes the action, and the appropriate parameters. The consumer can read the blockchain, with zero cost, to learn the available actions and the corresponding parameters. The smart contract verifies the transaction; if it is valid and if the amount of tokens held in escrow is sufficient, it generates an event. The IoT gateway is configured to "watch" the blockchain for events emitted by the smart contract, thus, the event that was triggered by the transaction results in gateway executing the corresponding actuation process. The communication between the gateway and the IoT devices is specified by the device vendor (out of the scope of this paper).

At the end of this process, an amount of tokens is transferred from the escrow to the owner. The amount of tokens required per operation is written in the blockchain. Thus, a consumer knows beforehand the amount of tokens required to execute an operation. The cost of an operation, measured in tokens, is the same for all consumers. However, the owner can sell tokens, off chain, to consumers at different prices. This allows to keep consumers-owners business relationships private.

IV. System design

The lifecycle of our system includes the following phases.

1) Setup: Initially, the owner deploys the IoT devices and two gateways (WoT servients, a software stack that implements the WoT-specific functionality of an IoT device). The first WoT servient includes the TD of the virtual entity, and it exposes all the operations of the virtual entity. The second servient acts as a client. From now on, we will refer to the first servient as *server servient*, and to the second as *client servient*. IoT devices are identified by URIs that can be used for performing actuations. In our system, the URIs are composed of the URL of the *server servient* plus the name of the IoT device (see Figure 1). Moreover, the owner creates a smart contract that represents the digital twin of the virtual entity and she deploys it on the blockchain. The smart contract's address on the blockchain is considered well-known. Then, she configures the *client servient* to "watch" the blockchain for the events.

Each *action* of the virtual entity is stored in a data structure in the smart contract, which is referred to as the *actionsList* (see Figure 1). The actions are represented in the smart contract as structs (solidity's way to allow users to create their own data types), which contain (a) an *action name*, (b) the input parameters (the type and the number of parameters), and (c) the defined price expressed in ERC-20 tokens. The parameters field is essentially the dataschema that describes the data format of the actions, properties, and events of a TD[4]. Furthermore, the owner should also create and deploy a smart contract, which creates and manages the owner-specific tokens, implementing the ERC-20 token standard [20].

A consumer contacts the owner, in order to come to an agreement on the price of a token. Then, he has to pay the agreed amount of money to the owner, in fiat currency. When the payment is completed, the owner transfers the agreed amount of the blockchain-based tokens to the consumer's wallet (Ethereum address) and, if necessary, configures the gateway to include a mapping between the consumer's blockchain address to some consumer-specific information. This process takes place off-line and off-chain, in order to keep consumers-owner business relationship private.

Finally, in order for the consumers to learn all the available actions, the appropriate parameters, and the cost for each action, they can call a view function of the smart contract, called *getActions*, if they want to learn about all the available actions, or *getAction* to learn about a specific action.

2) IoT device access: From this point on, a consumer can perform an IoT device actuation request. To do so, he sends a transaction to the smart contract that includes the desired action and the appropriate action parameters. The smart contract verifies that the transaction is correct, in the sense that the action exists in the *actionsList* and the parameters are correct. In addition, it verifies that the address of the consumer has the required number of tokens. In order to perform the latter verifications, the smart contract interacts with the smart contract that handles the tokens, and calls its *balanceOf* function. Then, these tokens are deposited to the smart contract's address. If all the requirements are met, an event, named *PerformActuation* is triggered. The event includes the action name, the parameters, and the address of the consumer. The address of the consumer is needed, in order to send the tokens back to the consumer, if the actuation will not be completed successfully.

The event is eventually "caught" by the *client servient*. This servient finds out which device is available at that moment to best serve the request. Subsequently, the gateway forwards the request to the appropriate devices, which perform the actuation. For example, if the consumer wants to interact with the smart elevator, the servient finds out which of the elevators is available at that moment and forwards the request to the appropriate one. When the action is completed, the *client servient*, configured with owner's blockchain wallet, sends a transaction to the blockchain. In particular, it calls the *endOfActuation* function to transfer the tokens from the smart contract's address to the owner's address, if the action was completed successfully, otherwise it transfers the tokens back to the consumer's wallet.

In the case that a consumer has not spent all the tokens and he does not want to use the provided services anymore, he has two options. The first option is to give back the remaining tokens to the owner and receive back fiat currency. Alternatively, he can sell his tokens to other consumers. The latter allows the creation of a meta-market and enables new business structures. For example, consumers can form 'aliases' and buy a big amount of tokens, potentially achieving a discount.

3) IoT device management: The smart contract-based digital twin includes the *actionsList*, which contains all the properties and actions of the virtual entity. This actions list can only be modified by the owner. In particular, the owner can add a new entry, as well as modify, or delete an existing one. This operation does not require any communication with the consumer. Additionally, an owner can replace a physical IoT device; since the consumer interacts with the virtual entity, which is device independent, this replacement affects only the configuration of the corresponding gateway.

4) Ephemeral interactions and revocation: There might are cases, where the owner should grant ephemeral permissions to guests. In such cases, the guest should come to an agreement with the owner for the price of tokens and the period that they need them. Then, the owner sends to him the agreed amount of tokens and adds the guest to a list, called *guestsList*, along with the corresponding period. This list exists in the *server servient*. From now on, the guest can interact with the IoT devices, as any other consumer. The difference is that when the agreed period is up, the owner calls the *withdraw* function from the smart contract that handles the tokens, to take back the tokens from the guest. This function can only be invoked by the owner, and in essence, it resets the token balance of a user, returning all the tokens back to the owner. Thus, this function can be used also in cases of security breaches. So, the *withdraw* function acts as a revocation mechanism too. The consumers, whose tokens are revoked, can no longer access the IoT devices and perform any actuation, thus the revocation is instantaneous.

V. Evaluation

A. Cost evaluation

To evaluate our solution, we developed a proof of concept implementation of the presented system. Our two servients are based on Eclipse's Thingweb[5], which is a Node.js implementation of the WoT model that is under standardization. For our evaluation, we emulated two IoT devices, one smart lamp and one smart coffee machine. The smart contracts of the system[6] are implemented using Solidity, and the piece of code that interacts with the blockchain in order to "catch" events generated by the smart contracts is implemented using the web3.js library.

All actions performed in our system involve the invocation of a function implemented in a smart contract, which incurs a transaction cost that in Ethereum is expressed as the cost of gas for executing transactions on the EVM. In order to measure the

[4]https://www.w3.org/TR/wot-thing-description/#dataschema

[5]http://www.thingweb.io/

[6]https://github.com/mmlab-aueb/DLT-DigitalTwins

cost required by our smart contracts, we deployed the smart contracts on the Rinkeby Ethereum test network. The cost, measured in gas units, for each one of the functions is shown in Table I.

Smart contract	Operation	Cost measured in gas
Digital twin	contract deployment	2341723
	performActuation	52329
	endOfActuation	43628
	addAction	119934
	modifyAction	41064
	removeAction	46632
	getActions	-
	getAction	-
ERC-20	contract deployment	805618
	balanceOf	-
	transfer	33664
	mintTokens	34786
	withdraw	29450

TABLE I
EVM EXECUTION COST OF OUR CONSTRUCTION BUILDING BLOCKS.

The functions that only read the blockchain and they do not write anything to it, they introduce zero cost. As we observe from the above table, the cost introduced by all functions is not completely negligible. The most expensive actions are those required for deploying the smart contracts on the blockchain ($\approx$ \$126.78 and \$43.62 respectively)[7], which however takes place only once. The most expensive function is the function responsible for adding a new action in the *actionsList*. The `addAction` function costs \$6.49, while the `performActuation` function, which is the function that clients will invoke, costs around \$2.83.

The price of the actions expressed in fiat currency is not standard, since it depends heavily on the price of Ethereum, which fluctuates highly. So, right now, given the price of Ethereum, the cost may be prohibitive for some use cases, like the case of a smart home. In use cases like that, it would be better to deploy our proposed system in a private instance of Ethereum, where no cost is introduced. On the other hand, on use cases, such as a university campus or a big business building, the cost may be acceptable.

In addition to gas, the use of the public Ethereum blockchain incurs a transaction delay, which depends on the block mining time. The average time required by an operation to be executed on Ethereum is ≈ 15 seconds. On the other hand, read requests and actions (view functions) on the blockchain do not need to send a transaction to the network. So, performing read operations incurs no transaction delay at all.

B. Qualitative and security evaluation

Our system has several advantages compared to legacy IoT systems and architectures. First, the use of the DLT as an event-based communication channel, enables mass actuation, namely actuation of multiple IoT devices simultaneously. This can be achieved, because the *client servient* in our system is "watching" the blockchain for events, thus with a single transaction, an action can be performed by many IoT devices. Along the same lines, our system facilitates service composition. Specifically, with just a single transaction, including multiple and appropriate parameters, a consumer can perform multiple different actions. Furthermore, new IoT devices can be seamlessly added in the system, since the owner has only to update the *servients* and the *actionsList*, without the need for redeploying the smart contract or the consumer's application. Similarly, the owner can add easily new servients that expose new TDs, without the need of changing anything to the underlay system. Moreover, with our solution, consumers are IoT gateway/device vendor-agnostic, since they communicate with the IoT devices and the corresponding gateway(s) through the blockchain infrastructure. They just have to send a transaction to the digital twin smart contract, for any device of any vendor.

In the Ethereum network, all blocks are broadcasted to all nodes, and since events are part of the blocks, they are received by all nodes as well. This way, the digital twin residing within the DLT acts as an indirection mechanism between the consumer and the IoT gateway, hence the location of the gateway does not have to be known to the consumers: the gateway can even be unreachable through the Internet, and disconnected from the outside world. In addition, DLTs offer reliability, robustness, and increased availability by design, since transactions are replicated in all miners participating in the network, therefore, there is no single point of failure, and the data is always available to the consumers. So, by implementing the digital twin of the virtual entity in a smart contract, we guarantee that the digital twin will always be live and it will not depend on trusted third parties. However, even if the gateways are secured, network outages may happen, leading to servients being unavailable and miss some events. In such cases, servients can be configured to read the blockchain, once a day or once a week depending on the use case, to see if they have missed an event and act accordingly (perform the actuation delayed or write to the blockchain that the corresponding actuation was not completed successfully).

Regarding the auditability feature, there are two different types of functions a smart contract can provide, *view* only functions, which read the state of the smart contract, and *write* functions, which modify it. The latter are invoked using Ethereum transactions, consequently every function invocation is recorded in a block appended immutably to the ledger. The aforementioned property enhances our system with high auditability in the sense that every interaction with an IoT device is recorded and thus it can be revisited at any time, by anyone. Hence, in case of security incidents or in case of disputes, blockchains can provide undeniable auditing information. Furthermore, in our design, if the actuation was not completed successfully, this is recorded on the blockchain acting as a proof of completed action, provided full auditability.

In addition, the design of our proposed system enables a form of access control. Only the users that have acquired the owner-specific tokens can invoke the available actions. Thus, these tokens except for enabling new business models act as Access Control Tokens (ACTs) too. This property is

[7]As measured on 13 March 2021.

very interesting, since blockchain-based tokens offer numerous advantages [21]. Furthermore, with our design, our system offers effective revocation. We guarantee, through the smart contract that only the owner can revoke the tokens, and once she revokes them from a consumer, the revocation will have immediate effect. However, the consumer once acquired the tokens, he can transfer them to anyone he wants. If we want to avoid situations like that, depending on the use case, we can configure the `transfer` function of the smart contract that manages the ERC-20 tokens to allow the consumers to transfer their tokens only to the owner's address.

Finally, since Ethereum is a public blockchain, anybody can inspect it and read its state and its past blocks. This property constitutes a privacy risk, since a third party can read the blockchain and deduce information about the users, such as who perform which actions, when, etc. Thus, a malicious user can link an Ethereum address with specific actions and deduce some patterns about that address. In our use case, this is not crucial, since our proposed system is responsible for the IoT devices deployed on the shared rooms and floors of the building. However, if we do not want anyone to learn anything, then this can trivially be addressed by using a private instance of Ethereum instead of the public Ethereum.

VI. Conclusions and future work

In this paper, we presented an architecture for implementing digital twins for IoT devices, using the WoT and DLTs. In particular, we leveraged the Ethereum blockchain and its support for smart contracts in order to build the digital twin of a WoT "virtual entity." Our proposed architecture enables consumers to perform actuation operations that eventually end up in one or more IoT devices. Our solution takes advantage of the event broadcast capabilities of Ethereum, the immutability of the transactions, and the transparency of the smart contracts, to provide secure and reliable services. In particular, our solution enables mass actuation, service composition, protection of the real gateways and devices, easier management of the underlay IoT network, improved auditability, increased availability and reliability, and novel business models.

In our work, we used the Ethereum blockchain, which however introduces monetary costs and non negligible delays. For this reason, we are currently experimenting with the use of Hyperldeger Fabric, i.e., a permissioned, private blockchain technology, which supports distributed applications. Fabric smart contracts, as opposed to Ethereum's smart contracts, can interact with the external world, e.g., they can read a value out of an IoT device using HTTP(s). Therefore, using Fabric our solution not only will become faster and cheaper, but it will also allow consumers to perform other operations in addition to actuation. Furthermore, using Fabric, it will be possible to use the actual TD in the smart contract, as opposed to its stripped down version, i.e., the *actionsList*. Moreover, an interesting extension to our system involves multiple DLT-based digital twins interacting with each other, allowing this way service composition at the blockchain level. Similarly, these twins could be implemented using different blockchain technologies and their interaction could take place using *interledger* technologies [22].

Acknowledgment

This work has originated with H2020 project SOFIE (Secure Open Federation for Internet Everywhere, Grant # 779984) and has since been supported in part by the Research Center of the Athens University of Economics and Business.

References

[1] W3C. (2017) Web of Things. https://www.w3.org/WoT/.

[2] M. Grieves, and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Cham: Springer International Publishing, 2017.

[3] BBC. (2020) AWS: Amazon web outage breaks vacuums and doorbells. [Online]. Available: https://www.bbc.com/news/technology-55087054

[4] The Verge. (2021) An Amazon server outage caused problems for Alexa, Ring, Disney Plus, and deliveries. [Online]. Available: https://www.theverge.com/2021/12/7/22822332/amazon-server-aws-down-disney-plus-ring-outage

[5] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[6] M. Andoni and V. Robu and D. Flynn and S. Abram and D. Geach and D. Jenkins and P. McCallum and A. Peacock, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 100, 2019.

[7] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[8] I. Pittaras, N. Fotiou, V. A. Siris, and G. C. Polyzos, "Beacons and blockchains in the mobile gaming ecosystem: A feasibility analysis," *Sensors*, vol. 21, 2021.

[9] N. Fotiou and G. C. Polyzos, "Smart contracts for the internet of things: Opportunities and challenges," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018.

[10] Z. Chevallier, B. Finance, and B. C. Boulakia, "A reference architecture for smart building digital twin," in *2020 International Workshop on Semantic Digital Twins, SeDiT 2020*, vol. 2615, France, 2020.

[11] Z. Liu, A. Zhang, and W. Wang, "A framework for an indoor safety management system based on digital twin," *Sensors*, vol. 20, 2020.

[12] N. Mohammadi and J. E. Taylor, "Smart city digital twins," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[13] G. White, A. Zink, L. Codecá, and S. Clarke, "A digital twin smart city for citizen feedback," *Cities*, vol. 110, 2021.

[14] I. Yaqoob, K. Salah, M. Uddin, R. Jayaraman, M. Omar, and M. Imran, "Blockchain for digital twins: Recent advances and future research challenges," *IEEE Network*, vol. 34, no. 5, 2020.

[15] S. Huang, G. Wang, Y. Yan, and X. Fang, "Blockchain-based data management for digital twin of product," *Journal of Manufacturing Systems*, vol. 54, 2020.

[16] D. Lee, S. H. Lee, N. Masoud, M. S. Krishnan, and V. C. Li, "Integrated digital twin and blockchain framework to support accountable information sharing in construction projects," *Automation in Construction*, vol. 127, 2021.

[17] W. Shen, T. Hu, C. Zhang, and S. Ma, "Secure sharing of big digital twin data for smart manufacturing based on blockchain," *Journal of Manufacturing Systems*, vol. 61, 2021.

[18] C. P. Nielsen, E. R. da Silva, and F. Yu, "Digital twins and blockchain – proof of concept," *Procedia CIRP*, vol. 93, 2020.

[19] S. Kaebish and T. kamiya and M. McCool and V. Charpenay and M. Kovatsch. (2020) Web of Things Thing Description. [Online]. Available: https://www.w3.org/TR/wot-thing-description/

[20] F. Vaogelsteller and V. Buterin. (2015) EIP-20: Token Standard. [Online]. Available: https://eips.ethereum.org/EIPS/eip-20

[21] N. Fotiou, I. Pittaras, V. A. Siris, S. Voulgaris and G. C. Polyzos, "Oauth 2.0 authorization using blockchain-based tokens," in *Proceedings of the NDSS 2020 Workshop on Decentralized IoT Systems and Security (DISS)*, 2020.

[22] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger approaches," *IEEE Access*, vol. 7, 2019.