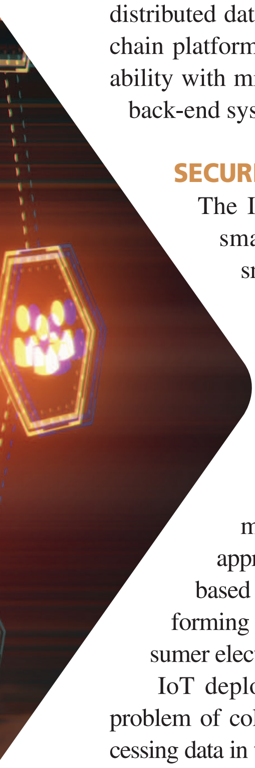
Designing a Blockchain-Based IoT With Ethereum, Swarm, and LoRa

Digital Object Identifier 10.1109/MCE.2018.2880806
Date of publication: 6 February 2019

The software solution to create high availability with minimal security risks.

By Kazım Rıfat Özyılmaz and Arda Yurdakul



TODAY, THE NUMBER OF INTERNET OF THINGS (IoT) devices in all aspects of life is increasing exponentially. Our cities are getting smarter and informing us about our surroundings in a contextual manner. However, we face significant challenges in deploying, managing, and collecting data from these devices. In addition, we must address the problem of storing and mining that data for higher-quality IoT services. Blockchain technology, even in today's nascent form, has the potential to be the foundation for a common, distributed, trustless, and autonomous infrastructure system. This article describes a standardized IoT infrastructure where data are stored on a distributed storage service that is fault-tolerant and resistant to distributed denial of service (DDOS) attacks and data access is managed by a decentralized, trustless blockchain. The illustrated system used LoRa as the emerging network technology, Swarm as the distributed data storage platform, and Ethereum as the blockchain platform. Such a data back end will ensure high availability with minimal security risks while replacing traditional back-end systems with a single "smart contract."

SECURITY, TRUST, AND IDENTITY

The IoT will be the backbone for smart buildings, smart energy systems, smart transportation, and smart health care, which will be the vital components of smart cities [1]. To ensure safe and rapid adoption of IoT solutions, three essential concerns should be recognized: security, trust, and the identity of things [2]. Blockchain technology not only addresses these three issues but also shows a clear path for integrating all kinds of IoT devices with a common blockchain-based infrastructure [3], [4]. This approach defines a different role for every IoT device based on its capabilities and power requirements, conforming with the mobile-edge computing vision for consumer electronic (CE) devices [5].

IoT deployments suffer from the problem of collecting, storing, and processing data in the cloud. An IoT platform should support multiple devices and services from different stakeholders, scale in a reliable and decentralized manner, and offer tools and support for the rapid creation of applications and their execution [6]. Selecting a unified method that enables data transmission from all kinds of IoT devices is another problem. To propose a solution, it is imperative to analyze what the future of the IoT landscape will look like. Ericsson predicts that low-power wide-area (LPWA) technologies, such as LoRa and Sigfox, which operate in an unlicensed band, and cellular-based NarrowBand IoT (NB-IoT), will be the great enablers for mass deployment of low-power end devices [7]. The current paradigm of short-range (near field communication, Bluetooth, Zigbee), mesh-topology (wireless sensor networks) communication, which limits the coverage area of IoT devices, is being challenged by the low-rate, long-range communication paradigm with star topology [8]. This shift in wireless communication technology may enable the massive deployment of low-power, low-cost devices with extended coverage. This gateway-centric approach inherently brings the possibility of implementing software solutions on IoT gateways.

In 2008, Satoshi Nakamoto proposed a novel digital currency, Bitcoin, based on a decentralized, trustless infrastructure [9]. All transactions would be stored on a distributed database called *blockchain* and continuously verified using public-key cryptography by all peers in the system, thus eliminating the need for a central authority. Modifying contents of the chain without being caught is only possible by owning at least 33% of the total computational power [10].

Based on Nakamoto's ideas about blockchains, in this article, we describe a blockchain-based IoT infrastructure for the emerging, gateway-centric communication technologies that most CE devices will use (Figure 1). We also propose different methods of integration for various types of end devices. Our software solution aims to 1) standardize the way IoT devices discover, communicate, and send data to their data repositories; 2) create a peer-to-peer, fault-tolerant, and DDOS-resistant infrastructure for IoT deployment; and 3) facilitate a standard way to query and acquire IoT device data for the creation of next-generation products and services.

To achieve these goals, we investigated how a peer-to-peer network may be used to store data and code fragments,

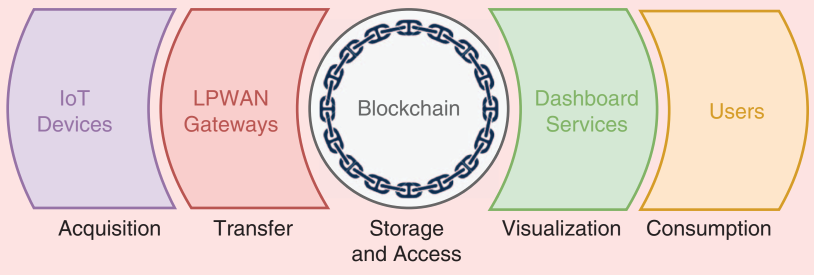


FIGURE 1. IoT system overview.

which, in turn, enables IoT gateways to push data and interact with other peers by means of a unified interface. As a proof of concept, a blockchain client is integrated to a LoRa gateway. To demonstrate this idea, we set up a private (although not mandatory) peer-to-peer network that makes use of these new blockchain-enabled LoRa gateways. The peers in this network send data through the IoT gateway, store it in a torrent-like distributed file system, save handles of data chunks to blockchain, interact with events, and access uploaded data using a blockchain infrastructure.

LPWA NETWORKS

LPWA networks (LPWANs), a new wireless connectivity technology, make use of star networks, as opposed to traditional wireless sensor networks, such as short-range, mesh networks. Because of protocol and transceiver architecture efficiencies, most LPWAN technologies enable low-power, low-cost end devices to communicate over distances of at least several kilometers. It is possible to build inexpensive sensor nodes without Subscriber Identity Module cards and still have robust gateways to connect and transfer data to Internet Protocol-based networks. LPWANs are grouped into two main categories: those for licensed-band operation and those for unlicensed-band operation. Three technologies operating in the licensed band are enhanced machine-type communication (LTE, category M), extended-coverage GSM, and NB-IoT [11]. Licensed technologies operate well in dense urban areas with good quality of service. Meanwhile, unlicensed technologies, such as LoRa and Sigfox, provide generally better coverage while requiring less power and costing less. Their downsides include a lower quality of service and no guaranteed latency [12]. Table 1 shows the operating frequencies, bandwidths, and data rates of these LPWAN technologies [13].

BLOCKCHAIN STATE OF AFFAIRS

BLOCKCHAIN

Blockchain is a distributed database deployed in a peer-to-peer network. Nodes in the system create and broadcast transactions continuously. Predictably, a blockchain consists of blocks, which are cryptographically linked and time-stamped collections of transactions. Nodes constantly verify blocks in the system to stand against malicious attackers trying to alter or forge transactions. All transactions in the system are signed using public-key cryptography and their authenticity is verifiable [9].

An in-depth look at the block structure of blockchain reveals that every block contains a block header and a varying number of transactions stored in a tree structure. In addition, every block header contains a timestamp and two hash values, one for a previous block's header and another for all transactions carried within that block. Because of this, it is possible to verify the integrity of the whole block, including all of the transactions via block header.

In traditional blockchain-based systems, special nodes called *miners* try to find the next block by calculating a solution for a hard-to-compute, but easy-to-verify, mathematical problem, where the difficulty of the problem is set as a constraint that is continuously changing. When a new block satisfying the current difficulty constraint is found, it's propagated to the network as the next valid block and its miner is rewarded for its efforts. A total block creation and propagation mechanism keeps all peers synchronized, i.e., in consensus. The difficulty of this mechanism is changed periodically to keep the block finding time in a predefined interval [9].

ETHEREUM AND SWARM

Ethereum is a blockchain-based infrastructure where stakeholders compile code fragments (smart contracts) that may interact with each other or change the state of accounts on blockchain [14]. Regular Bitcoin transactions contain sender and receiver addresses, value, and a custom scripting system for verification. Ethereum extends the scripting capabilities of Bitcoin to a fully fledged, Turing-complete programming language that aims to create a programming environment [14]. As a result, Ethereum turns out to be a distributed application platform using blockchain technology where users may pick arbitrary formats for transaction or ownership. Ethereum smart contracts are compiled virtual machine opcodes executed by the Ethereum Virtual Machine. The smart contract's functions and events can only be accessed using the mined

Table 1. LPWA IoT connectivity overview.

	LoRa	Sigfox	NB-IoT (Release 13)	Enhanced Machine-Type Communication (Release 13)	Extended-Coverage GSM (Release 13)
Range (km)	<11	<13	<15	<11	<15
Maximum coupling loss (dB)	157	160	164	156	164
Spectrum	Unlicensed <1 GHz	Unlicensed 900 MHz	Licensed long-term evolution	Licensed LTE	Licensed GSM
Bandwidth	<500 kHz	100 Hz	180 kHz	1.08 MHz	200 kHz
Data rate	<50 kilobits/s	<100 bits/s	<250 kilobits/s	<1 megabits/s	<140 kilobits/s

address of the contract and its application binary interface (ABI) (Figure 2).

Swarm is a peer-to-peer storage service integrated with Ethereum. It promises zero downtime and is DDOS resistant, fault tolerant, and censorship resistant [15]. It is a torrent-like service with built-in incentives to guarantee uploaded data persistence due to high coupling with the Ethereum network layer. Hence, it is a strong candidate for a storage service targeting IoT.

BLOCKCHAIN NODE TYPES

The various types of blockchain node types include the following.

- Miner: Miners are special nodes that pack transactions into blocks and run the consensus algorithms that satisfy system requirements to attain a financial benefit. In proof-of-work consensus, miners in the network possess the highest computational power.
- Full node: Full nodes download the whole blockchain and verify the integrity of all transactions continuously, making the infrastructure trustless and decentralized. Sufficient storage and computing power are required to run a full node.
- Thin client: Thin clients only download the block headers that contain the hashes of the transactions within the block. Therefore, it is possible to interact with the blockchain with minimal storage and computing requirements. This approach is called *simplified payment verification* in Bitcoin and *light client* in Ethereum [9], [14].
- Server-trusting client: Bitcoin client application programming interface (BCCAPI) is proposed to make secure, lightweight clients for resource-constrained systems. With BCCAPI, it is possible for a client to connect a server containing the blockchain and run queries against it. Here, the server has only public keys of clients and is unable to create a transaction without a client's approval.

IOT–BLOCKCHAIN INTEGRATION METHODS

Integrating IoT end devices and gateways to a blockchain infrastructure can be accomplished in many different ways, depending on the capabilities and power requirements of end devices and gateway hardware. Assuming that end devices are either battery powered or always on and they are communicating with an always-on gateway connected to the Internet (like a typical LPWAN case), one of the following integration strategies can be used for IoT gateways.

- Gateway as a full blockchain node: The IoT gateway operates as a full node, routing data to the network and verifying integrity at the same time. Integration is relatively easy because no changes are required in the way that end devices communicate. However, gateways should be powerful enough to operate as a full blockchain node. With the gateway's total computing power for defending the integrity of the system, it is possible to achieve a trustless IoT infrastructure.
- Gateway as a thin client: The IoT gateway operates as a thin client by routing data to the network and storing only relevant data fragments. Integration is relatively easy. However, the weakness here is that there should be other full nodes to defend the integrity of the system. A trustless infrastructure can still be achieved, but only with full nodes operating at the cloud side.
- End devices as regular sensors: Battery-powered end devices may be so weak that no additional client logic may be tolerated. In this case, no blockchain client is integrated. Transmitted data are received by an IoT gateway and are pushed to a blockchain infrastructure. This is suitable for extremely low-power sensors that do nothing more than broadcast their data.
- End devices as server-trusting client: A blockchain client using a BCCAPI-like interface may be integrated to battery-powered end devices. This way, the end device will interact with a blockchain node without any storage or computational requirements.
- End devices as thin client: If end devices are not battery powered and always on, they can operate as a thin client. Here, gateways can either be a full node or a transparent switch to relay transactions. If both gateways and end devices use blockchain clients, standardization in terms of data collection can be achieved.

Table 2 shows the differences between traditional and blockchain-based IoT integration where every component

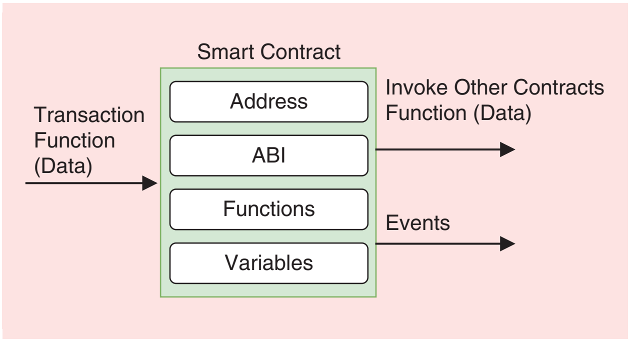

Smart Contract
Transaction Function (Data)
Address
ABI
Functions
Variables
Invoke Other Contracts Function (Data)
Events

FIGURE 2. Smart contract structure.

Table 2. Roles and communication methods in LPWAN IoT infrastructure.

	Battery-Powered End Device	Always-On End Device	IoT Gateway	Cloud Back End
Traditional IoT	Sensor with custom protocol	Sensor with custom protocol	Transparent proxy	Centralized core services
Blockchain IoT	Server-trusting client or sensor with custom protocol	Thin client or server-trusting client	Full node or thin client	Miners and full nodes

acts as a part of a trustless peer-to-peer network and contributes to this network as much as its capabilities allow. This way, data collection and storage may be standardized by using blockchain client protocols.

PROOF OF CONCEPT

For the proof-of-concept implementation, LoRaWAN is selected because it is an unlicensed-band LPWAN technology with an affordable concentrator and end-device hardware. In a previous implementation, a battery-powered LoRa end device's position data were sent to a LoRa gateway, which then routed this data stream through the official Go-lang-based Ethereum client Geth to a private Ethereum network using a smart contract [16]. We extended this preliminary work by 1) storing IoT data not in blockchain but in the Swarm storage service, therefore eliminating the need for a private Ethereum network, and 2) by defining a clear way to access and retrieve data using Swarm and Ethereum smart contracts for additional applications, such as user interface services (e.g., data dashboard) or machine-learning systems [17].

A prototype system uses a LoRa end device, built with a Raspberry Pi 2 computer connected to a Dragino LoRa/GPS Hat and a LoRa gateway, built with a Raspberry Pi 3 computer connected to a LoRa concentrator board named *iC880A* from the Institute of Mobile and Communication Satellite Technology. The IoT gateway runs LoRa protocol software to communicate with low-power end devices. LoRa protocol software consists of a concentrator card driver and a network daemon to forward data packets into a local proxy server. This local server, called *smart proxy*, receives data from the packet forwarder and acts as a mediator to push data into the blockchain-based infrastructure. Finally, Swarm and Ethereum clients complete the data flow.

LoRa end devices wait for their turn and send their data without the need of establishing a connection to any specific party. The always-listening network daemon on the LoRa gateway picks up the transmission and forwards it to the smart proxy. After data are received, they are pushed into the Swarm file storage network using HTTP. A file hash is received and that same file hash will be used to access those particular data in the future (Figure 3).

A smart proxy may communicate with the Ethereum client by means of its JavaScript Object Notation remote procedure call interface. However, to enable a real interaction with the Ethereum network, a smart contract should be deployed first. After being compiled into bytecode, smart contracts are sent just like any other transaction, to be mined by miners. When a smart contract is mined, its address and application binary interface are used to interact with it.

Our smart contract code contains one event, which is *log_action*, and six functions named *is_device_present()*, *get_device_count()*, *get_device_at_index()*, *get_device_timestamps()*, *get_device_data()*, and *set_device_data()*. All functions except *set_device_data()* are constant functions that do not change the contract state. Only *set_device_data()* adds data to blockchain, thus a transaction should be set up. List 1 shows the actual code fragment declaring how device identifications, timestamp values, and Swarm file hashes are connected. IoT gateways and their stored file hashes in the blockchain can be easily enumerated and accessed by using *get_device_timestamp()* and *get_device_data()* functions. *set_device_data()* is the actual smart contract function (List 2) that creates and maps a Swarm file hash to the current block timestamp. As soon as new data are added, *log_action* event is fired and all peers listening to that event get a callback. Ethereum smart contract and LoRa proxy code used for this implementation can be found on the "Bether" project page [17].

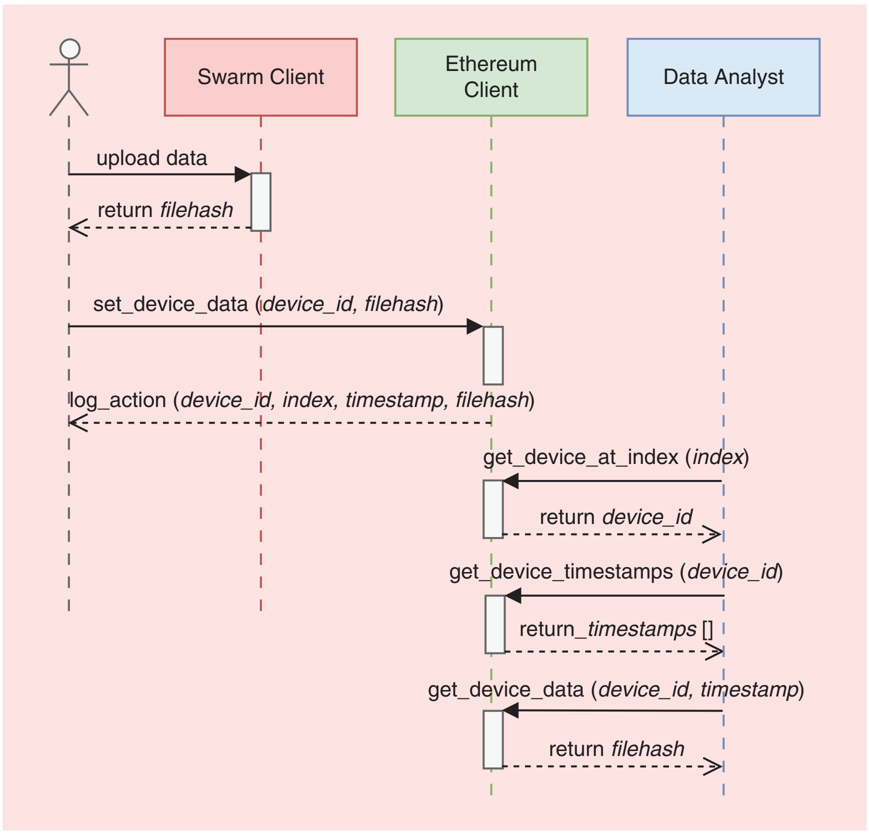

Swarm Client
Ethereum Client
Data Analyst
upload data
return *filehash*
set_device_data (*device_id, filehash*)
log_action (*device_id, index, timestamp, filehash*)
get_device_at_index (*index*)
return *device_id*
get_device_timestamps (*device_id*)
return_*timestamps* []
get_device_data (*device_id, timestamp*)
return *filehash*

FIGURE 3. The IoT data storage and access scenario.

EVALUATION

RESOURCE CONSUMPTION

Our setup has at least three different types of Ethereum node configurations to accommodate underlying hardware resources. The usage statistics below are given 1) for a one-month-old, private

Ethereum installation (around 200,000 blocks) and 2) for the public Ethereum blockchain. Peak memory consumption may vary from setup to setup due to synchronization speed.

- Mining full node: With an active Swarm client, these nodes use between 1.2 and 1.5 gigabytes of memory in a private Ethereum network. In public networks, they need at least 4 gigabytes of memory; it is expected that this requirement will go up in time. These nodes are powerful servers deployed in the cloud.
- Nonmining full node: With an active Swarm client, these nodes use between 300 and 400 megabytes of memory in private Ethereum networks. In public networks they need at least 2 gigabytes of memory to properly synchronize with blockchain. These nodes may be IoT vendor servers, network provider servers, or powerful IoT gateways.
- Nonmining light node: With an active Swarm client, these nodes use around 300 megabytes of memory, whereas Ethereum client only uses around 50 megabytes (200 megabytes in public Ethereum). Because Swarm has no light client mode to limit bandwidth or memory usage at the moment, the memory benefit is minimal. These nodes may be regular IoT gateways and end devices.

List 1. Smart contract data structure.

```
//IoT device data
struct devdata {
    //link for detecting devices
    uint index;
    //blockchain timestamps
    uint[] timestamps;
    //map timestamp values to swarm hashes
    mapping(uint => string) filehashes;
}
//device id array of all received id's
address[] private devindex;
//map device id's to their data
mapping(address => devdata) private devlogs;
//event to log action
event log_action (address indexed devid,
                  uint index,
                  uint timestamp,
                  string filehash);
```

List 2. Smart contract: Store Swarm file hash.

```
//set Swarm data handle for source device
function set_devdata (address devid,
                      string hash)
public returns (uint index,
                uint timestamp) {
//get current block timestamp
ts = now;
//store data receive time (block timestamp)
devlogs[devid].timestamps.push(ts);
//store swarm handle
devlogs[devid].filehashes[ts] = hash;
//store device link
devlogs[devid].index = devindex.push(devid)-1;
//trigger event, signaling received data
log_action(devid, devindex.length-1, ts, hash);
//return device index and timestamp
return(devindex.length-1, ts);
}
```

DATA THROUGHPUT

Data throughput in blockchain systems depends on various metrics and varies in different implementations. Bitcoin imposes throughput limits with its 10-min average block time and fixed 1-megabyte block size. In Ethereum's case, there is no fixed block size but a gas limit per block (i.e., amount of resource to be used by transactions). Similar to resource statistics, throughput statistics are given 1) for a one-month-old, private Ethereum installation and 2) for the public Ethereum blockchain.

Our private Ethereum network has a gas limit of 4,712,388 gas/block and the average gas price is 21,000 gas. Therefore, a block may only contain 224 transactions. Considering that the average block time is 14 s for the private system, the throughput will be 16 transactions/s (or around 1,000 transactions/min). The public Ethereum is in the middle of a difficulty increase as of September 2017. At the time of writing, the public Ethereum system has a gas limit of 6,718,904 gas/block, an average gas price of 21,000 gas, and an average block time of 30 s. Data throughput will in turn be 320 transactions per block, which is 10.6 transactions/s (or 640 transactions/min).

Though transaction throughput seems low to support a full-scale deployment today, note that transactions are created only by IoT gateways and every gateway may serve hundreds of thousands of end devices. The proposed infrastructure can support tens of thousands IoT gateways (and millions of end devices) pushing data every 15 min.

Using blockchain's decentralized, trustless nature in combination with DDOS-resistant, fault-tolerant data storage, a new type of IoT back end may be created.

DISCUSSION

Blockchain systems may be improved for better IoT integration.

- Inefficiency: Bitcoin and Ethereum use proof-of-work algorithms that guarantee every mined block is backed by a certain amount of computational work. This approach is inherently inefficient because every miner in the system is doing hard calculations individually. In the IoT blockchain, a proof-of-stake-based consensus may be much more suitable as discussed in the section "Blockchain State of Affairs." Proof-of-stake algorithms may create monopolies due to stake concentration, but in the case of the IoT, this "bug" may be used as a "feature." Empowering certain trusted parties like system integrators or regulators may indeed be beneficial.
- Encryption and access control: Blockchain-based systems store clear data although transactions are signed with public-key cryptography. When an IoT system is dealing with sensitive data, either payload must be encrypted before it's pushed into blockchain. Otherwise, sophisticated mechanisms to conceal critical data should be used. For example, shielded Zcash transactions are published on a public blockchain, but the sender, recipient, and the amount remain private by using zero-knowledge proofs [18].
- Bandwidth: Gateways in LPWANs are the point of transmission to the cloud for connected end devices. All LPWANs assume gateways are connected to a fast communication link, either wired or wireless. If gateways operate as a full node, bandwidth requirements will increase considerably because of the messaging and synchronization traffic of blockchain.
- Real-time systems: Due to their trustless nature, blockchain-based systems may be able to store data only after a certain period of time, which is determined by the block creation interval. To support real-time applications, data propagation delay should be minimized by proposing new types of consensus functions fine-tuned for IoT scenarios.

CONCLUSION

The IoT is the key to smarter cities, transportation systems, energy systems, and health care. To deal with the increasing number of IoT devices, it is necessary to standardize the method of communication for IoT gateways and create a common IoT back end. Using blockchain's decentralized, trustless nature in combination with DDOS-resistant, fault-tolerant data storage, a new type of IoT back end may be created. In this way, all kinds of IoT end devices may be integrated with this infrastructure based on their computing and storage capabilities. Such an achievement will lead to data-centric business models where application development and data processing can be massively conducted by using smart contracts as demonstrated with our proof-of-concept Bether [17].

ABOUT THE AUTHORS

Kazım Rıfat Özyılmaz (kazim@monolytic.com) is a Ph.D. student in computer engineering at Bogazici University, Istanbul, Turkey.

Arda Yurdakul (yurdakul@boun.edu.tr) is a professor in computer engineering, Bogazici University, Istanbul, Turkey.

REFERENCES

[1] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything you wanted to know about smart cities: The Internet of Things is the backbone," *IEEE Consum. Electron. Mag.*, vol. 5, no. 3, pp. 60–70, 2016.
[2] B. Montgomery, "Future shock: IoT benefits beyond traffic and lighting energy optimization," *IEEE Consum. Electron. Mag.*, vol. 4, no. 4, pp. 98–100, 2015.
[3] J. Lee, "BIDaaS: Blockchain based ID as a Service," *IEEE Access*, vol. 6, pp. 2274–2278, 2017.
[4] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and C. Yang, "The blockchain as a decentralized security framework," *IEEE Consum. Electron. Mag.*, vol. 7, no. 2, pp. 18–21, 2018.
[5] P. Corcoran and S. K. Datta, "Mobile-edge computing and the Internet of Things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 73–74, 2016.
[6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[7] Ericsson. "Cellular networks for massive IoT—enabling low power wide area applications," 2016. [Online]. Available: https://www.ericsson.com/en/white-papers/cellular-networks-for-massive-iot--enabling-low-power-wide-area-applications
[8] L. Vangelista, A. Zanella, and M. Zorzi, "Long-range IoT technologies: The dawn of LoRa," in *Future Access Enablers of Ubiquitous and Intelligent Infrastructures*. New York: Springer Verlag, 2015, pp. 51–58.
[9] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system." 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[10] I. Eyal and E. G. Sirer, Majority is not enough: Bitcoin mining is vulnerable. 2013. [Online]. Available: arXiv.org/abs/1311.0243
[11] 3GPP. "Progress on 3GPP IoT," 2016. [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1766-iot_ progress
[12] J. Bardyn, T. Melly, O. Seller, and N. Sornin, "IoT: The era of LPWAN is starting now," in *Proc. IEEE European Solid-State Circuits Conf.*, 2016, pp. 25–30.
[13] Nokia. "LTE evolution for IoT connectivity," 2015. [Online]. Available: http://resources.alcatel-lucent.com/asset/200178
[14] V. Buterin. "A next-generation smart contract and decentralized application platform," GitHub, 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper
[15] J. H. Hartman, I. Murdock, and T. Spalink, "The Swarm scalable storage system," in *Proc. 19th IEEE Int. Conf. Distributed Computing Systems,* 1999, pp. 74–81.
[16] K. R. Özyılmaz and A. Yurdakul, "Integrating low-power IoT devices to a blockchain-based infrastructure: Work-in-progress," in *Proc. 13th ACM Int. Conf. Embedded Software*, 2017, pp. 13:1–13:2.
[17] K. R. Özyılmaz, "Bether: IoT backend with Swarm and Ethereum smart contracts," GitHub, 2017. [Online]. Available: gttps://github.com/kozyilmaz/bether
[18] E. B. Sasson et al., "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc IEEE Symp. Security and Privacy*, 2014, pp. 459–474.