Distributed Ledger Technology and the Internet of Things: A Feasibility Study

Atis Elsts
University of Bristol
Bristol, UK
atis.elsts@bristol.ac.uk

Efstathios Mitskas
University of Bristol
Bristol, UK
stathis.mitskas.2014@bristol.ac.uk

George Oikonomou
University of Bristol
Bristol, UK
g.oikonomou@bristol.ac.uk

ABSTRACT

Distributed Ledger Technologies have promising applications in the Internet of Things. However, scalability and support for micropayments remain problems for blockchain-based applications, therefore other alternatives such as IOTA should be considered. This work reports on an experimental evaluation of IOTA on several different IoT platforms. We show that even though the communication overhead to join the IOTA network can be significantly reduced by adapting a proxy-based architecture, the computational overhead remains high. We conclude that IOTA is not currently suitable for battery-powered IoT devices.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**;

KEYWORDS

Distributed Ledger Technologies, Internet of Things

ACM Reference Format:
Atis Elsts, Efstathios Mitskas, and George Oikonomou. 2018. Distributed Ledger Technology and the Internet of Things:, A Feasibility Study. In *Proceedings of ACM SenSys (BlockSys 2018).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Distributed Ledger Technologies (DLT) provide a completely novel way for transfer value between parties that do not trust each another. Many Internet of Things (IoT) systems would immensely benefit from having such a way of transferring value in a peer-to-peer, opportunistic fashion. As a result, these is a real need to integrate DLT in the IoT. IOTA [11] is one of cryptocurrencies that aims to be suitable for micropayments and machine-to-machine (M2M) economy in particular.

One of the core claims of IOTA is that the due to their sheer number, a network of IoT devices can "outcompute" a network of traditional high-performance machines, even though the latter individually are much more powerful. Since there already are billions of IoT devices today, and their numbers are expected to increase sharply in the coming years, this is a strikingly interesting claim. However, there is a number of complications. First, the term "IoT device" refers to a vast variety of different devices that are spanning a wide range of computational capabilities [2]. The less powerful IoT devices may not have a sufficient amount of RAM and program memory to run the Proof-of-Work (PoW) and other algorithms required to participate in the IOTA network. Second, the majority of IoT devices are also characterized by the need to save energy. Due to this reason, their available computational power may be much smaller than their nominal power. More analysis is required to understand the current scope of applicability of IOTA to IoT.

To this end, this work makes the following contributions:

- It outlines and compares a number of different network architectures for bridging IoT devices with the IOTA network.
- It quantifies the time and energy consumption for communication with the IOTA network depending on the network architecture.
- It quantifies the time and energy consumption for IOTA operations on a number of hardware platforms.

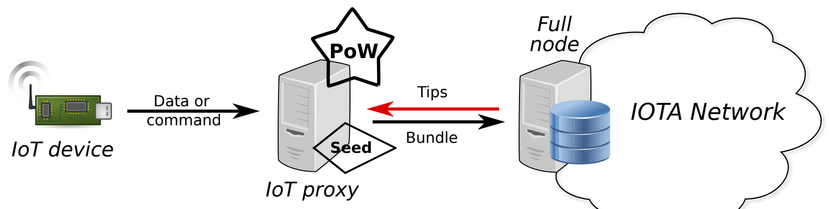
2 BACKGROUND

In many IoT applications, there is a need to transfer value between the IoT devices and other parties. The value can be either in form of valuable, authenticated sensor data, services, or micropayments. At the moment, such transfers are implemented through billing arrangements. However, setting up and maintaining such arrangements is cumbersome and not always feasible. For a few examples, consider cars and road-side units interacting in an intelligent transportation system. Two cars meeting on the road may want to buy data from each another: for instance, data about the road conditions ahead. A car may similarly want to buy data from a road-side unit that is equipped with sensors and has communication links with other nearby units. Finally, an overtaking maneuver between autonomous cars may be facilitated through a small payment with the aim to incentivise moving aside and letting pass by.

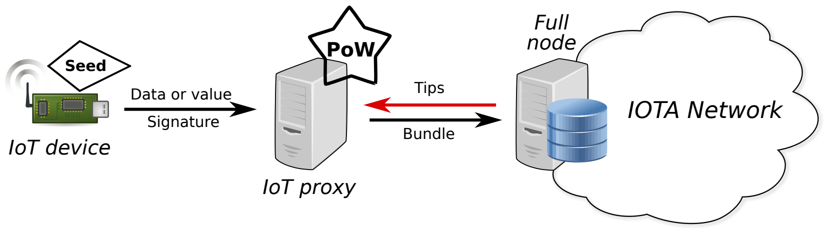
In general, blockchains can be used for these payments. However, permission-less, Proof-of-Work based blockchains [10] require transaction fees, therefore are not directly suitable for micropayments. Private, permission-based blockchains are much more lightweight [3]; however, they necessarily remove other important properties for Proof-of-Work based blockchains, such as censorship resistance. Unlike blockchains, other DLT technologies such as IOTA aims to make the IoT devices first-class actors in the network. IOTA is a cryptocurrency based on the Tangle [11], a novel distributed consensus mechanism that is used DAG. It is developed by the IOTA Foundation, a non-profit organization registered in Germany.

The IOTA network and the Tangle are maintained by IOTA *full nodes.* These full nodes are required to have high computational

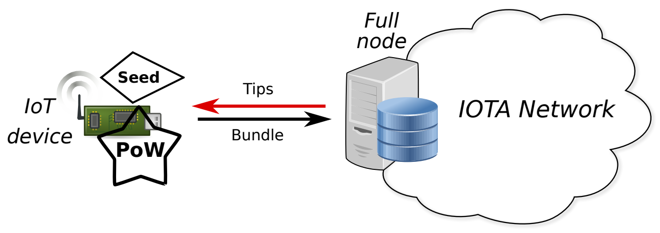
BlockSys 2018, Nov. 2018, Shenzhen, China
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Baseline: using IoT proxy for all operations



(b) Using IoT proxy for the Proof-of-Work (PoW)



(c) Direct communication without a proxy

Figure 1: Different network architectures for bridging IoT devices with the IOTA network.

capacity and high-speed internet access. A much larger number of *light nodes* also participate in the network. Unlike the full nodes, they do not keep a copy of the Tangle in their memory. Nevertheless, these light nodes have a very important role to play. According to the IOTA design vision, that the light nodes are going to perform the bulk of the Proof-of-Work computations required to secure the IOTA network. In this way, the more nodes participate in the network, the more secure it becomes. In theory, many IoT devices are computationally capable of fulfilling the role of the light node. The IOTA security model relies on the assumption that due to their sheer number, the computationally less powerful IoT devices are capable of out-computing dedicated high-power machines that any attackers might have. At the core, both PoW and transaction signing in IOTA consists of computing the Keccak hash function [1] many times over. This function is currently not widely hardware-accelerated, however, it may become such in the future, since Keccak has been adapted as the SHA-3 standard [4].

The Tangle [11] is a directed acyclic graph; the nodes in this graph are IOTA transactions [7]. Each transaction must include the PoW validating two other transactions; this validation corresponds to edges in the Tangle. An IOTA *bundle* is simply an atomic collection of transactions. For example, a value transfer from one IOTA wallet to another typically is implemented as a bundle with at least three transactions: one transaction to empty the source address, one to put the value in the destination address, and one to put the remainder in a new local address.

IOTA for IoT has been analyzed in previous research [6, 12], however, these papers do not consider highly-constrained devices.

Table 1: Estimated payload sizes for each of the architectures.

Payload type	*Architecture #1*	*Architecture #2*	*Architecture #3*
Data only	2186 trytes	–	–
$n_u = 1$	–	135 trytes	2673 trytes
$n_s = 1$	–	2322 trytes	2673 trytes
$n_u = 1, n_s = 1$	–	2457 trytes	5346 trytes
$n_u = 2, n_s = 2$	–	4914 trytes	10692 trytes

3 NETWORK ARCHITECTURES

We consider IoT devices that either want to store their data in the Tangle, or want to perform micropayments. The network infrastructure has to support these operations. One way how to do that is to use a dedicated *IoT proxy* node: a computationally more powerful and energy less restricted node that does some of the operations on behalf of the IoT devices, and consequently relieves their computational load. However, with the proxy approach, the IoT devices continue to rely on the classical client-server model; consequently,

Table 2: Estimated over-the-air transmission energy requirements for each of the architectures.

Payload type	*Architecture #1*	*Architecture #2*	*Architecture #3*	Savings due to *A#2*
Data only	4.5 mJ	–	–	–
$n_u = 1$	–	0.28 mJ	5.5 mJ	95 %
$n_s = 1$	–	4.8 mJ	5.5 mJ	13 %
$n_u = 1, n_s = 1$	–	5.1 mJ	11.0 mJ	54 %
$n_u = 2, n_s = 2$	–	10.1 mJ	22.0 mJ	54 %

they do not fully benefit from the key properties of decentralized cryptocurrencies. For example, some level of trust in the proxy is required. Therefore this approach may not be feasible in case of mobile IoT nodes that want to interact with the IOTA network outside of their “home environments”.

We consider several different architectures:

(1) ***Architecture #1*: Using a proxy for all IOTA operations**. This (Fig. 1a) is the baseline architecture; here, the IoT devices do not perform any IOTA operations. Instead, they fully rely on the IoT proxy.
   - **Benefits.** The least computational load; no additional software required on the IoT device if data-only transfer is sufficient; no need to store potentially valuable IOTA seeds on the IoT devices.
   - **Drawbacks.** An available IoT proxy device is required to conduct all IOTA operations. High level of the trust in the IoT proxy needed, since the proxy is storing IOTA seeds on behalf of the devices.
   - **Payload transmitted by IoT.** Sensor data: same as without IOTA. If value transfer is desired, a custom command protocol may be used to instruct the proxy to transmit this on behalf of the IoT devices.
   - **Payload received by IoT.** None.

(2) ***Architecture #2*: Using a proxy for the Proof-of-Work.** Here (Fig. 1b), the IoT proxy is still present, and the IoT devices rely on it for the PoW. However, the seed is stored on the IoT device, and the proxy does not need to be as highly trusted as in *Architecture #1*.
   - **Benefits.** The communication load can be made relatively lightweight: instead of transmitting the full IOTA bundle, the IoT devices may transmit just the critical parts, for example, the amount, the source and target addresses, and the signature(s). PoW computation is not required.
   - **Drawbacks.** Need to deploy valuable IOTA seeds and update software on the IoT devices. Signing transactions is sill a relatively computationally expensive.
   - **Payload transmitted by IoT.** Signed sensor data and signed value transactions (or parts of them, if optimizations are used).
   - **Payload received by IoT.** None.

(3) ***Architecture #3*: Direct communication without a proxy.** Here (Fig. 1c), the IoT devices directly communicate with a full node.
   - **Benefits.** The simplest architecture: no additional elements required. Fully realizes the trustlessness and censorship-resistance of decentralized cryptocurrencies.
   - **Drawbacks.** Unless the full node offers to do PoW on behalf of others, the PoW has to be done on the IoT device, which may be too computationally expensive for the majority of them.
   - **Payload transmitted by IoT.** The full IOTA bundle.
   - **Payload received by IoT.** IOTA tips: *i.e.*, hashes of two other transactions, by the IOTA protocol required to be validated to submit a new transaction.

For convenience, let us assume that the IoT device produces data in batches of 2187 trytes. This is the maximal number of bytes that can fit in the IOTA transaction’s field `signatureMessageFragment`, which is meant to be reused for data storage if the signature is not present.

With *Architecture #1*, the IoT device needs to transit only the data (2187 trytes). With *Architecture #3*, the IoT device needs to transmit the whole IOTA bundle, which is 2673 trytes times the number of transactions in the bundle, as well as receives 81×2 trytes: the *tips* from the Tangle.

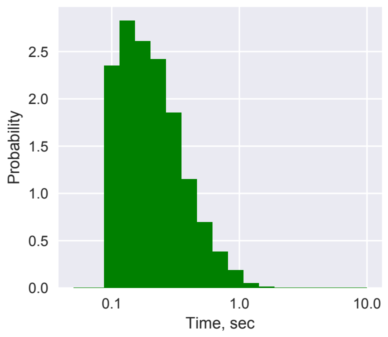
With *Architecture #2* the IoT device wants to transmit authenticated data or a value. In the first case, the data can be put in the `signatureMessageFragment` of the output transaction. Since the output transaction does not have the space for the signature because of the data storage, a new empty output transaction can be created. Depending on the IOTA security level, more than one input transaction is needed to store the signature (the recommended value of this parameter 2, which leads to two input transactions. If just a small amount of data needs to be transmitted (up to 27 trytes), that can be put on the `tag` field of the output transaction; or, since the output transaction does not transfer a value, the `address` field of the recipient can be reused to store the data.

The hash of an IOTA bundle is calculated based only on the following fields: transaction addresses, tags, indexes, and a timestamp. This makes it possible to sign partially constructed bundles. However, the bundle hash and therefore the signature does not directly cover the `signatureMessageFragment` of other transactions in the bundle. If the IoT device does not trust the IoT proxy to fill the output transaction with data, it can circumvent this limitation by putting the hash of the data in the `tag` or `address` field of the transaction, so that the hash of the bundle also covers the data, albeit indirectly.

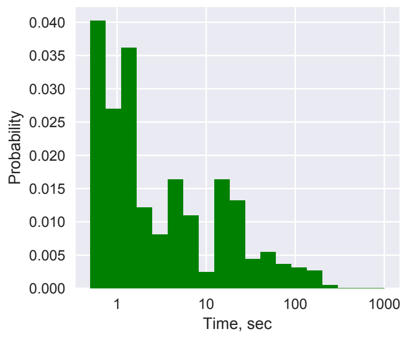
In case the IoT node wants to transfer a value, the structure of the bundle is similar. IOTA does not allow the source address to be reused; therefore, more than one output transaction may be required in order to store the remainder of the value, as the value transferred may be less than the total value stored in the source address.

There is no need to transmit the whole IOTA bundle between the IoT device and the IoT proxy. We propose that the IoT device reduces its communication load following this algorithm:

(1) The IoT device partially constructs the bundle, filling only the address, transaction index, tag and timestamp values.
(2) The IoT device calculates the bundle hash.
(3) The IoT device signs each transaction needing a signature.

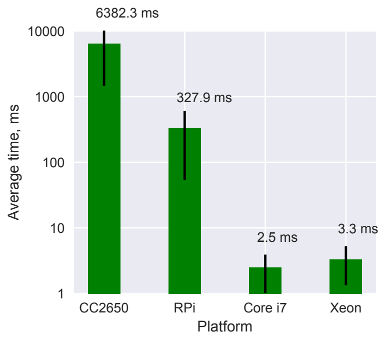


(a) Transaction signing

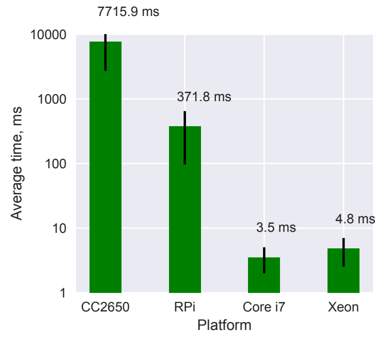


(b) Proof-of-Work computation

Figure 2: Distribution of the time (a) to generate a signature for a transaction ($S = 2$) and (b) to compute the Proof-or-Work ($MWM = 14$). Experimental results on Raspberry Pi 3.



(a) With security level $S = 1$



(b) Time to sign a transaction

Figure 3: Time to sign a transaction. Experimental results on the different hardware platforms.

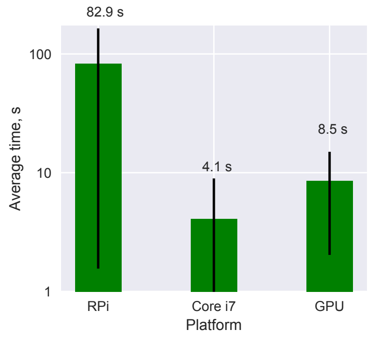


Figure 4: Time to compute the Proof-of-Work for a single a transaction. Experimental results on the different hardware platforms, $MWM = 14$.

(4) The IoT device transmits the data, the destination addresses & IOTA amount for each of them, and the signatures to the IoT proxy.
(5) IoT proxy gets the tips to verify, fully constructs the bundle, performs the PoW for each of the transactions in the bundle, and attaches the bundle to the Tangle.

The size of IOTA address is 81 trytes, tag: 27 trytes, the amount field: 27 trytes. Overall, the amount of trytes T that need that need to be transferred in case of a bundle with n_u unsigned transactions and n_s transactions where the `signatureMessageFragment` is filled, is:

$$T = n_u \times (81 + 27 + 27) + n_s \times (2187 + 81 + 27 + 27) = 135n_u + 2322n_s \quad (1)$$

Some examples of payload sizes are given in Table 1. Let us now assume that the IoT device uses the IEEE 802.15.4 protocol to transmit the data over the air. Results from research literature report that this protocol allows to transmit up to 300 bytes per single millijoule of energy [14], namely, use 3.3 μJ per byte or 2.06 μJ per tryte. Using this estimate, we can calculate how much data needs to be transmitted with each method, then convert this value to trytes, and then estimate how much energy that would require (Table 2).

4 EXPERIMENTAL EVALUATION

4.1 Methodology

4.1.1 Hardware Platforms. For the comparison, we select two IoT platforms and two desktop server platforms. One of them is a Class-1 constrained IoT device [2]: the Texas Instruments (TI) Launchpad with a CC2650 System-on-Chip. It is a development board for rapid prototyping of energy-efficient applications, intended to be used

Table 3: Comparison of the hardware platforms.

Platform name	CPU	CPU core	CPU frequency	RAM size	Power consumption	Number of cores
TI Launchpad	CC2650	Cortex-M3	48 MHz	20 kB	9.6 mW [9]	1 core
Raspberry Pi 3B [13]	BCM2837	Cortex-A53	1200 MHz	1 GB	221.0 mW per core	4 cores
Dell Optiplex	Intel Core i7-6700	–	3400 MHz	16 GB	65 000 mW TDP	8 cores
Dell Precision 7910	Intel Xeon E5-2623	–	3000 MHz	32 GB	105 000 mW TDP	16 cores
Dell Precision 7910	Nvidia Quadro K620	–	1370 MHz	2 GB	41 000 mW max	384 cores

Table 4: Average time measured and energy consumption estimated for the IOTA operations.

Platform name	Time to sign ($S = 2$)	CPU energy to sign	Time for PoW	CPU energy for PoW
TI Launchpad	7716 ms	74 mJ	–	–
Raspberry Pi 3B	372 ms	82 mJ	82.9 sec	54.9 J
Intel Core i7-6700	3.5 ms	28 mJ	4.1 sec	233.2 J
Intel Xeon E5-2623	4.8 ms	31 mJ	–	–
Nvidia Quadro K620	–	–	8.5 sec	93.5 J

with low-power network stacks such Bluetooth Low Energy (BLE) and IEEE 802.15.4. Its System-on-Chip is build on the ARM Cortex-M3 core. The second is Raspberry Pi Model 3B: a very well known and more powerful IoT device. For networking, the Pi has a built in WiFi and BLE module. Its CPU is built on another ARM core: the Cortex-A53 "application processor" core. The other two desktop servers are manufactured by Dell and have Intel processors. One of them is also equipped with a Nvidia Quadro GPU. The summary of the platforms is given in Table 3 (including a separate entry about the GPU).

4.1.2 Software. In order to evaluate IOTA operations, we use software written in C and compile it for the different target platforms. All of the target platforms are running the Linux operating system, with the exception of CC2650 Launchpad, which is running Contiki-NG[1]: a light-weight operating system for the Internet of Things.

Proof of Work. To test the performance of PoW, we use CCurl[2], the C port of the Curl library. This port is developed and maintained by the IOTA Foundation. CCurl makes use of multiple core, by default launching $N - 1$ threads on a N core machine. It also incorporates OpenCL support, which allows it to use the GPU instead of the CPU for computation, if the system has a GPU.

In our experiments, the CCurl code is compiled with the GCC C compiler, with optimization flag `-O3` enabled. We also find that in order to successfully run this code on ARM architectures, `-fsigned-char` compiler option must be enabled, since the `char` type is unsigned by default on the ARM version of GCC, in contrast to signed defaults on `x86` GCC.

The CCurl is not evaluated on the CC2650 Launchpad due to its memory limitations (the CC2650 only has 20 kB RAM). As it stands, the CCurl implementation requires a platform with at least 64 kB or RAM: for example, the `find_nonce()` function alone uses more than 32 kB of stack memory.

[1]https://github.com/contiki-ng
[2]https://github.com/iotaledger/ccurl

We evaluate the software with the Minimum Weight Magnitude (MWM) parameter set to 14, which is the PoW complexity currently used in the IOTA mainnnet.

Transaction preparation and signing. We use the code of IOTA wallet application for Ledger Blue and Nano S hardware wallets[3]. We port this code to the Contiki-NG operating system, making some minor changes in the process: for example, all calls to file system functions and dynamic memory allocation are removed.

We compile the code with the `-O2` optimization level. On CC2650 Launchpad, the Contiki-NG takes full control of the hardware initialization and control. On the server platforms and Raspberry Pi, the "native" architecture of the Contiki-NG OS is used. When compiled for this architecture, the Contiki-NG is run as a single user-space process on top the host operating system (Linux).

We evaluate the software for security levels 1 and 2 ($S = 1, S = 2$). The software utilizes only a single thread and does not support computation on the GPU.

4.2 Results

Figure 2 reports the distribution of the time required to sign a transaction (evaluated 1000 times for each security level) and to compute the PoW (evaluated 100 times). Both the PoW and the computation of the signature are highly nondeterministic: surprisingly, the time to sign can vary up to 10-fold (Fig. 2a) depending on the input.

Figure 3 compares the average time to sign a transaction on the different hardware platforms (note: the bars here and further show the standard deviation). It takes multiple seconds to perform this operation on the CC2650 Launchpad, and there is also a two-orders of magnitude difference between the Pi and the server platforms.

The time to compute the Proof-of-Work for a single transaction is given in Fig. 4. Surprisingly, the Pi is relatively more efficient for this operation, while the GPU significantly underperforms its expectations: doing the PoW on a multi-core server is faster than doing it on the GPU in our experiment.

[3]https://github.com/IOTA-Ledger/blue-app-iota

Table 4 shows the time and energy consumption comparison on the different platforms. For the server platforms and the signing operation, we report the total energy divided by N, where N is the total number of cores. Since the signing operation only utilizes a single core, the maximum system load would be achieved when N signing operations would be run in parallel. This correction is applied because for these platforms, Table 3 lists the total power of the CPU, rather than the consumption per core as for the Raspberry Pi CPU. In contrast, the PoW algorithm already makes use multiple cores ($N - 1$ in a system with N cores), so this correction is not needed. Finally, for the Nvidia GPU, we report the actual energy consumption shown by the `nvidia-smi` command during the test.

4.3 Discussion

4.3.1 Energy usage. [4]

For the PoW, the energy consumption is between 55 and 233 J on the average (Table 4), but can frequently take 10 or more times of that due to the nondeterministic nature of the PoW operation. To put these numbers in perspective, the total capacity of typical IoT batteries ranges from 100 mAh @3.3 V (≈1200 J) [8] to 2700 mAh @3.6 V (≈35000 J) [5]. Clearly, this operation is not suitable for any battery-powered IoT devices. With a nontrivial probability, the 1200 J battery will run out of power before the device completes the first PoW operation. The situation could change in the future if hardware implementations of SHA-3 are widely deployed on IoT devices and the rest of the factors (such as the MWM parameter) remain the same.

Even the signing operation is surprisingly complex. The conventional wisdom is that in embedded systems, communication requires more energy than computation. However, in this case the balance tips towards computation (compare Table 2 with Table 4): a device requires on the order of 10 times more energy to sign a transaction than to transmit the transaction. As a result, the number of transactions to sign is the limiting factor for its battery lifetime. Assuming that the device creates a two-transaction bundle every minute, it consumes $2 \times 74 = 148$ mJ for signing per minute (213 J per day). With this workload, the 1200 J battery is going to last less than 6 days even discounting the energy needed for transmissions and for other operations.

Given the energy usage results, it is clear that on battery-powered devices, both PoW and transaction signing are not practical without hardware-accelerated cryptography. More powerful devices such as Raspberry Pi are capable of doing both operations, but instantaneous transactions are beyond their resources: for instance, a single Raspberry Pi Model 3B can only do the PoW for up to approximately 1000 transactions per day (Table 4).

4.3.2 Safety in numbers? Surprisingly, the Raspberry Pi computes the PoW in just 20 times more time than the Intel Core-i7 server. This means that a network of 20 Pis can successfully compete with a single server (if we restrict the discussion exclusively to *majority-of-PoW* type of attacks). Given the proliferation of IoT devices, it is not unthinkable that the total available computational power of Pi-class devices could indeed already be greater than the available power of server-side devices, or become greater soon.

[4]The original submitted version had an error in this subsection; it has been fixed and the text updated.

Clearly, the IOTA operations can be greatly sped up by a hardware-accelerated implementation. However, this argument equally applies both the server-side and IoT-side platforms, therefore is not clear that adding hardware acceleration would tip the balance in the computational power away from IoT. If IOTA-like cryptocurrencies become more widely used in the future, the manufacturers of IoT devices will have a strong incentive to put hardware accelerators on their devices, which in turn would help to secure it against attackers using dedicated, centralized hardware.

5 CONCLUSION

This paper presents a feasibility evaluation of IOTA on IoT devices. The results show that even though communication overhead of bridging the IOTA network with IoT can be significantly reduced by using a gateway architecture with an IoT proxy server, the computational requirements remain high. In particular, the current battery-powered IoT devices are not suitable for IOTA operations. In contrast, the more powerful Raspberry-Pi class IoT devices are capable of joining the IOTA network as light nodes, and, in theory, it not infeasible that a network of such devices could one day provide sufficient Proof-of-Work capacity to secure the IOTA network, assuming all else remains equal.

ACKNOWLEDGMENTS

This work was performed under the SPHERE IRC, funded by the UK EPSRC Grant EP/K031910/1.

REFERENCES

[1] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 2009. Keccak sponge function family main document. *Submission to NIST (Round 2)* 3, 30 (2009).

[2] C. Bormann, M. Ersue, and A. Keranen. 2014. *Terminology for Constrained-Node Networks.* RFC 7228. IETF.

[3] Ali Dorri, Salil S Kanhere, and Raja Jurdak. 2016. Blockchain in internet of things: challenges and solutions. *arXiv preprint arXiv:1608.05187* (2016).

[4] Morris J Dworkin. 2015. *SHA-3 standard: Permutation-based hash and extendable-output functions.* Technical Report.

[5] EVE (Energy Very Endure). 2010. Technical Specification: ER14505 Lithium-thionyl Chloride (Li-SOCl2) Battery. http://www.evebattery.ru/datas/menu/eve/er14505_eve.pdf.

[6] Bogdan Cristian Florea. 2018. Blockchain and Internet of Things data provider for smart applications. In *IEEE MECO.* 1–4.

[7] IOTA Foundation. 2018. IOTA Developers Documentation. https://docs.iota.org/.

[8] Huizhou Markyn New Energy. 2014. Product Specification for Li-ion Polymer Battery: Model 351536. https://www.powerstream.com/lip/GMB261534-100mAh(with%20PCM)A5.pdf.

[9] Texas Instruments. [n. d.]. CC2650 SimpleLink Multistandard Wireless MCU. http://www.ti.com/lit/ds/symlink/cc2650.pdf.

[10] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[11] Serguei Popov. 2016. The Tangle. (2016), 131.

[12] Rahul Radhakrishnan and Bhaskar Krishnamachari. [n. d.]. Streaming Data Payment Protocol (SDPP) for the Internet of Things. ([n. d.]).

[13] Raspberry Pi (Trading) Ltd. 2016. Compute Module Datasheet.

[14] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. 2012. How low energy is Bluetooth Low Energy? Comparative measurements with ZigBee/802.15.4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE.* IEEE, 232–237.