# Hardware-Accelerated Blockchain-Based Authentication for the Internet of Things

Joanne Marie V. Santos, Jeanne Eunice V. Pascua, and Nestor Michael C. Tiglao(✉)

Ubiquitous Computing Laboratory Electrical and Electronics Engineering Institute, University of the Philippines, Velasquez Street, Diliman, 1101 Quezon City, Philippines
jvsantos@up.edu.ph, {jeanne.eunice.pascua,nestor.tiglao}@eee.upd.edu.ph

**Abstract.** Internet of Things (IoT) is steadily evolving which allows a new paradigm of smart sensors and lightweight devices interacting with one another without human intervention, also known as machine-to-machine (M2M) communication. This allows solution for various fields such as Smart Home Automation. In this scenario, the satisfaction of security and data privacy requirements play a fundamental role. Blockchain technology with the help of cryptography offers a solution by facilitating transactions and the coordination of devices without the need of a central authority. This study aims to improve the current smart home by developing and implementing a blockchain-based authentication system with the use of the Blockchain data structure, protocols and cryptographic algorithms such as Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA-256), and Keyed-Hashing for Message Authentication (HMAC) on a microcontroller board equipped with hardware acceleration. Our performance analysis showed that hardware acceleration provided significant improvement in processing time with a speedup of 5.53 and 7.94 times for AES-128 and SHA-256, respectively, compared to a software implementation counterpart.

**Keywords:** IoT · Blockchain · Lightweight authentication · Hardware acceleration

## 1 Introduction

Internet of Things (IoT) has been gaining popularity over the past decade due to its contributions to automated services and the collection and processing of data [2]. Its capability for machine-to-machine and human-to-machine communication allows the emergence of smart homes, smart grids, and smart cities. These systems provide solutions to various global challenges such as the growing demand for energy services. A viable implementation is a power outlet monitoring system and automation, a *Smart Plug*, which raises the awareness of homeowners of their energy consumption. However, the integration of IoT in the


H. Jin et al. (Eds.): CROWNCOM 2021/WiCON 2021, LNICST 427, pp. 283–295, 2022.
https://doi.org/10.1007/978-3-030-98002-3_21

home presents challenges such as devices' security and data privacy. An overview of the current system framework of the smart plug is presented in Fig. 1. It is composed of four parts: the user interface, the main controller, the smart plug, and the server.

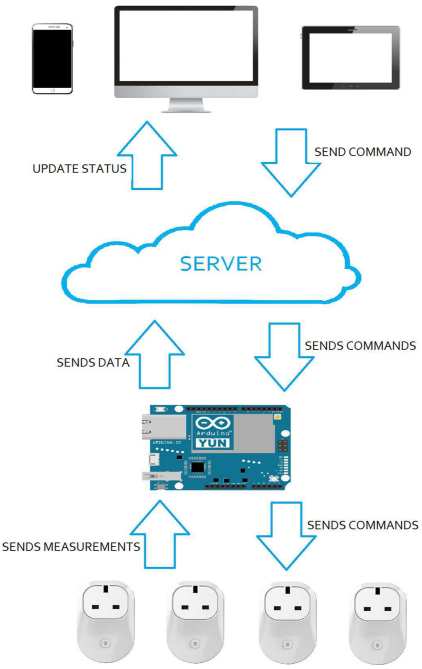



**Fig. 1.** Current system overview

*Blockchain* can be described as a distributed database with a data structure that consists of blocks which are linked back to a previously created block - its parent block. It creates a chain going back to the first block ever created, also called as the *genesis block*. This cascade effect of creating generations of blocks is the key feature of blockchain security because changing the whole blockchain means a recalculation of the subsequent blocks.

The security system, particularly the cryptographic algorithms can be implemented through the use of microcontrollers. Embedded systems, such as microcontrollers are preferred than general purpose CPU's due to space, power and cost saving reasons. Thus, a lightweight authentication system for the Smart Plug can be realized through microcontroller tools for developers. Additionally, various microcontrollers already support these cryptographic algorithms used by blockchain-based systems, as well as, non-blockchain-based implementations.

The implementation of such architecture is suitable for *machine-to-machine (M2M)* communication. M2M refers to communications between computers,

embedded processors, and smart sensors with little or no human interaction [5]. As an example, wireless M2M communication lets machines communicate directly with one another (e.g. data transfer).

Commonly, an additional system with significant cost, area, and computation time is used to mitigate these issues. Thus, in this study, our goal is to design, develop, and implement an authentication system.

The specific objectives of this study are as follows:

- To implement cryptographic algorithms which provide data confidentiality from the smart plug's controller and server gateway
- To develop a hardware implementation of a blockchain-based authentication scheme for device integrity
- To compare various techniques and standards for lightweight devices in machine-to-machine (M2M) communications

Paper Contributions: The major contributions of this paper are as follows: (1) design and implementation of hardware-based blockchain authentication scheme; (2) comparison of various techniques for lightweight authentication algorithms for machine-to-machine communications; and (3) performance evaluation of the implemented schemes.

The rest of the paper is organized as follows. Section 2 provides the related work. Section 3 describes methodology of this study. Section 4 discusses the results and analysis. Section 5 provides our conclusion and recommendations for future work.

## 2 Related Work

In this section, we discuss lightweight cryptography and blockchain technologies. We highlight the need for efficient hardware implementation.

### 2.1 Lightweight Cryptography

A cryptographic system may aim to provide confidentiality, authentication, integrity, non-repufdiation, access control, etc. Basically, there are two classifications of cryptography - symmetric and asymmetric - based on how messages are encrypted and decrypted. Symmetric encryption uses a private key to encrypt and decrypt an encrypted message. On the other hand, asymmetric encryption uses the public key of the recipient to encrypt the message. To decrypt the message, the recipient will have to use his/her private key to decrypt.

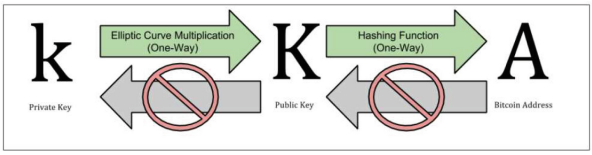
Lightweight cryptography is studied for constrained devices for the IoT. Its properties have been discussed in ISO/IEC 29192. The properties are described based on their target platforms such that in hardware implementations, smaller chip size and lower energy consumption are desirable while in software implementations, smaller code and RAM size are preferable. Lightweight cryptography should also deliver adequate security. In symmetric cryptography, aside from

AES, block ciphers CLEFIA, PRESENT, and the hash function SHA-3 are considered. Meanwhile, there are no asymmetric cryptography primitives that meet enough lightweight properties or can execute at a reasonable time. However, Elliptic Curve Cryptography (ECC) is said to have a relatively small footprint than other asymmetric primitives [10].

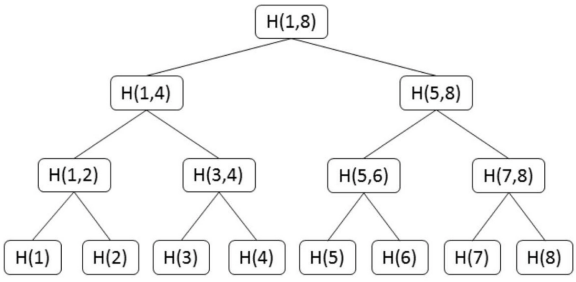
### 2.2 Blockchain

A blockchain can be implemented in two ways: a (1) *public blockchain* and a (2) *private blockchain*. A public blockchain allows anyone to participate as long as the transactions are valid. On the other hand, in a private blockchain, reading data and sending transactions for validation may only be done by a predefined list of entities [8].

The original implementation of the blockchain comes from Bitcoin [16], which uses public key cryptography. Keys and bitcoin addresses and their transformation can be seen in Fig. 2a. Bitcoin uses a Merkle Tree or a Binary Hash Tree, which is a data structure used for summarizing and verifying the integrity of data [14]. This is based on the Merkle's signature scheme as seen in Fig. 2b.




(a) Private key, Public key and Bitcoin address (adopted from [3])




(b) A merkle hash tree from Niaz et al.'s work [17]

**Fig. 2.** Blockchain implementation of Bitcoin

Another work combines blockchain and cryptography which focuses on ensuring IoT device authentication to the gateway [7]. It uses hybrid cryptography - symmetric cryptography for the transmission of transaction and asymmetric cryptography for the transmission of the symmetric key. The IoT device generates a session key using Password-Based Key Derivation Function 2 (PBKDF2).

On the other hand, the encryption of the session key uses RSA. Meanwhile, blockchain transaction structure, as shown in Table 1, consists of the transaction ID, data length, data, signature of the sender and public key of the receiver [7].

**Table 1.** Transaction structure of blockchain for IoT from Gaurav et al. [7]

| Field | Purpose |
|---|---|
| Transaction ID | Unique number of transaction |
| Data length | Number of bytes of data |
| Data | Actual message |
| Signature of sender | Identity of sender |
| Public key of sender | Used for encryption, decryption of data and data length |

Due to the increasing number of M2M devices, hierarchical network architectures are proposed. Devices, or nodes, are usually embedded in smart devices which reply to requests or sends data packets to the gateway in a single hop or multi-hop patterns. The gateway, on the other hand, acts as an entrance to another network. Meanwhile, the software agents are the connections between the gateway and applications that report data to the user.

We see the emergence of more related on hardware acceleration using FPGA platform for cryptographic hash computations [4] and bitcoin mining for different cryptocurrencies [1,21]. Furthermore, more application-specific blockchain such as in P2P energy trading or transactive energy are being explored [13,20]. To address the scalability, latency, and energy requirement challenges of blockchain processing, work such as in [15] are focusing on integrating a more efficient hardware-based concensus algorithm.

## 3 Methodology

Advanced Encryption Standard (AES) is included in the ISO/IEC 18033-3 standard for encryption algorithms in the 128-bit block ciphers category. It is a symmetric block cipher that cuts a message into blocks, with a length of 128 bits, and encodes them individually using a pre-shared key which will be stored in the server and in the devices before system deployment [9]. The cryptographic algorithms such as AES-128 and the blockchain hashes will be implemented using the peripheral device which is the NXP Freedom Development Board (FRDM-K82F). This effectively decreases the amount of work to be done by the smart plug controller - Arduino 101. The communication between the smart plug and the peripheral device is done using Inter-Integrated Circuit (I2C).

The device to gateway communication will consist of nRF24L01+ wireless transceivers which is suitable for ultra low power wireless applications and operates in 2.4 GHz ISM band [18]. This uses SPI communication between the Arduino 101.

The *MultiCeiver* feature of the nRF24L01+ can also be utilized for a system with many devices. It can contain a set of six (6) parallel data pipes of star network topology. This provides unique addressability in the physical and link layer as seen in Fig. 3.

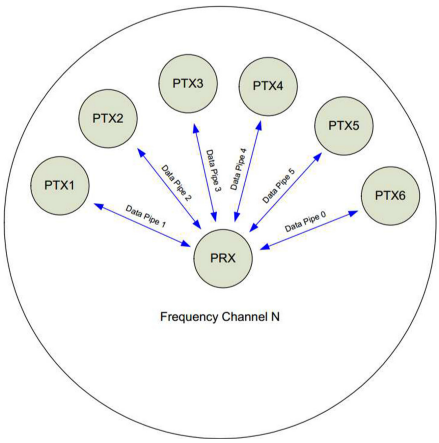



**Fig. 3.** Unique addressability in nRF24L01+ from [18]

Handling multiple devices can cause concern on packet collisions. Thus, the system utilizes Time Division Multiple Access (TDMA) where each device transmits data at a specific time intervals. The system architecture with the necessary communication between the devices, and the devices to gateway is shown in Fig. 4.

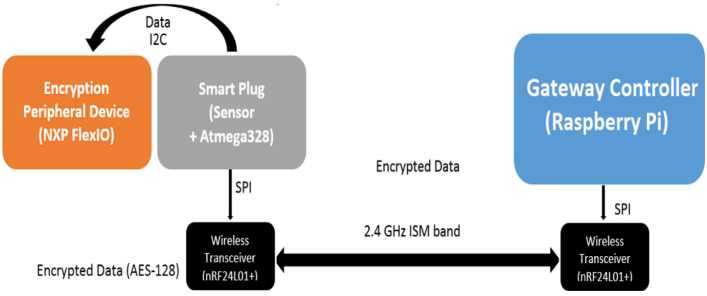



**Fig. 4.** Device to gateway diagram with wireless transceivers

### 3.1 Keyed-Hashing for Message Authentication (HMAC)

Message authentication codes (MACs) that are based on cryptographic hash functions are called Hash-based message authentication codes (HMACs).

HMACs combines public keys, private keys, and a hash for message authentication. An HMAC is computed by the following formula [12]:

$$HMAC(K, text) = H(K \oplus opad \,||\, H((K \oplus ipad)||(text)))$$

where,

$K$ is the shared secret key
$text$ is the message input
$opad$ is 0x5C repeated $B$ times ($B = 64 \equiv$ byte-length)
$ipad$ is 0x36 repeated $B$ times
$\oplus$ is the bitwise exclusive-or operation
$||$ is concatenation.

There is a well-known practice with MACs that truncates the output. It has an advantage of less information available to the attacker and a disadvantage of less bits to predict for the attacker. It is recommended, however, to limit the length of the truncated output to be not less than half the length of the hash output. For instance, HMAC-SHA-1-80 denotes HMAC computed using SHA-1 function with output truncated to 80 bits [12]. Additionally, *Authenticated Encryption (AE)* is necessary. For the study, *encrypt-then-authenticate* shown in Fig. 5 must be used since it is proven to be generically secure for implementing secure channels [11].

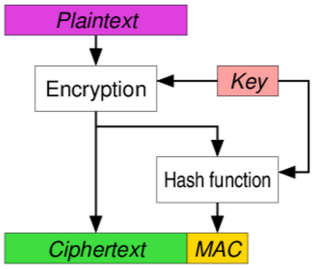



**Fig. 5.** Encrypt-then-authenticate mechanism [6]

### 3.2 Authentication of Genesis Block

To simultaneously provide authentication and encryption for new data to be linked in the blockchain, the authentication scheme is based on the (1) general protocol of symmetric message authentication using HMAC, (2) the blockchain and (3) AE's method of encrypt-then-authenticate. The algorithm to be used for the genesis block, the first block in the blockchain, which contains the data from the device to be verified by the server is shown in Algorithm 1.

The HMAC is truncated for the reason that the maximum payload length that the transceiver can operate only takes up to 32 bytes as shown in Fig. 6.

**Algorithm 1.** Genesis block symmetric message authentication and encryption

1: Devices $A$, $B$, $C$ and $D$ compute $E_A$, $E_B$, $E_C$ and $E_D$.
2: Devices $A$, $B$, $C$ and $D$ compute their $M_n = HMAC\,SHA256_{128}(K, E_n)$.
3: Devices $A$, $B$, $C$ and $D$ send their $M_n$ and $E_n$ to $S$.
4: Server $S$ checks whether $M_n \stackrel{?}{=} HMAC\,SHA256_{128}(K, E_n)$.


**Fig. 6.** Payload from devices to server $S$ containing encrypted data and authentication code

### 3.3 Authentication of Subsequent Blocks

For a node to create a block in the blockchain, the incoming block header should contain the previous block hash. Thus, the algorithm to be used for the authentication of the subsequent blocks is shown in Algorithm 2.

**Algorithm 2.** Authentication and encryption of subsequent blocks

1: Devices $A$, $B$, $C$ and $D$ compute $E_A$, $E_B$, $E_C$ and $E_D$.
2: Devices $A$, $B$, $C$ and $D$ compute their $M_n = HMAC\,SHA256_{128}(K, E_n)$.
3: Devices $A$, $B$, $C$ and $D$ send $H_{previous}$.
4: Server $S$ checks $H_{previous}$.
5: Devices $A$, $B$, $C$ and $D$ send their $M_n$ and $E_n$ to $S$.
6: Server $S$ checks whether $M_n \stackrel{?}{=} HMAC\,SHA256_{128}(K, E_n)$.

The server has to verify first that the incoming block's previous block hash field - $H_{previous}$ is a hash known to the server. This is the last block on the chain. The protocol can further be illustrated in Fig. 7.

### 3.4 Server

TCP/IP yields a better throughput and less packet loss compared to UDP when using the IEEE 802.11 standard. The current Application Program Interface (API) uses HTTP, which uses TCP/IP to communicate from the gateway to the web server which is built using Node.js. The HTTP Client will run in the gateway and send requests to the server in order to communicate with it and update the database for the data.

The *Requests* library for python was used in order to simplify the HTTP communication between the gateway and the remote server. It utilizes the urllib3 python library for automatic keep-alive and HTTP connection pooling [19].

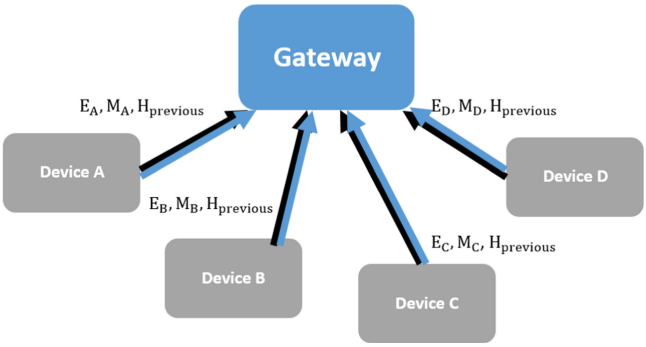



**Fig. 7.** Blockchain-based authentication protocol for devices' new data

### 3.5 Database

The web server is built using Node.js and MongoDB for the management of the database. We rearrange the database structure used in the current home automation system to fit the MongoDB data-as-documents architecture and add a new category to include the header of the previous block of transactions in order to create the blockchain. The new database structure can be seen in Fig. 8.

- ***yun_devices*** contains the list of devices connected to the main controller. Each device will be identified using a *node_ID*. It will also contain a table called the *block_headers*.
- ***device_sched*** contains the schedule of devices for actuation.
- ***device_power*** contains power consumption of each device connected to the smart plug. It will be sorted into *power_hour*, *power_day*, *power_week*, *power_month* and *power_year*.
- ***device_status*** contains the status of each smart plug (e.g. connected, identified, modified)
- ***block_headers*** contains the metadata of each blocks. This is separated from the node data for faster retrieval of the blocks for authentication.

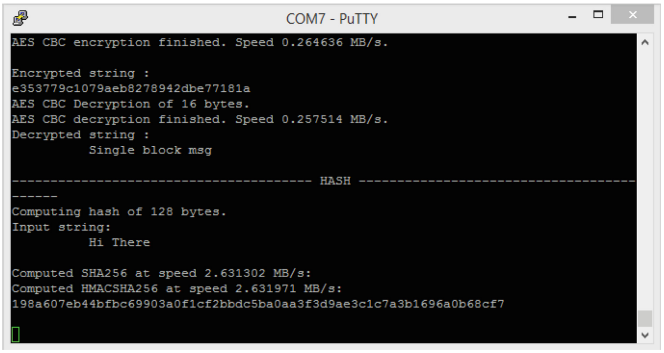
## 4 Results and Analysis

**Functionality Testing**
To check the operation of AES-CBC-128 in Kinetis FRDM-K82F and Arduino 101, test vectors from NIST, specifically GFSbox, KeySbox, VarKey and VarTxt Known Answer Test Values were used.

SHA-256 is the hash function used in HMAC. To check its operation, the Secure Hash Algorithm Validation System (SHAVS) short messages tests for byte-oriented implementations were used. Aside from the NIST test vectors, RFC test vector was also used for the HMAC. Sample outputs of both the AES and HMAC which are displayed via PuTTY can be found in Fig. 9.


**Fig. 8.** Database structure


(a) Sample output for AES-CBC-128




(b) Sample output for HMAC-SHA-256-128

**Fig. 9.** Sample outputs for test vectors on devices

Additionally, the encryption and decryption on the server was tested using the test vectors as seen in Fig. 10. Meanwhile, the test vectors for the HMAC validation was taken from RFC 4231. From the tests conducted, the integrity of implemented encryption, decryption and HMAC algorithms was proven.

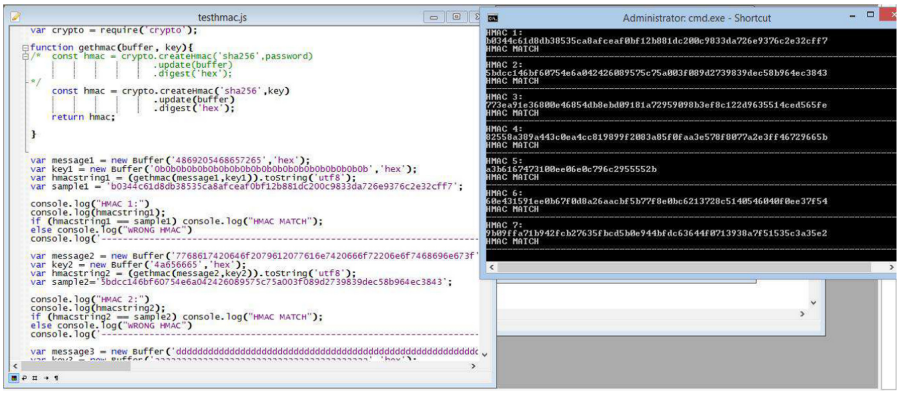



**Fig. 10.** Sample outputs for test vectors on server

**Non-functionality Testing**

The performances of the FRDM-K82F and Arduino 101 are compared by measuring the processing time of the algorithms. FRDM-K82F has ARM Cortex M4 with 150 MHz clock speed. It also has a Memory-Mapped Cryptographic Acceleration Unit (mmCAU). Meanwhile, the Arduino 101 has Intel Curie microprocessor with 32 MHz clock speed.

The algorithms for encryption (AES-CBC-128) and HMAC (SHA-256) are tested using 10 vectors from the functionality testing. The AES-CBC-128 ran at an average of 61 µs in Kinetis FRDM-K82F and 335 µs in Arduino 101. Meanwhile, SHA-256 ran at an average of 28 µs in Kinetis FRDM-K82F and 220 µs at Arduino 101.

**Table 2.** Average processing times (in $\mu$s) and speedup

| Encryption algorithm | Kinetis FRDM-K82F | Arduino 101 | Speedup factor |
|---|---|---|---|
| AES-128 | 60.665264 | 335.2 | 5.53 |
| SHA-256 | 27.6773738 | 219.7 | 7.94 |

## 5 Conclusions and Future Work

The past implementation of the smart home automation system only had a log-in authentication which is a user authentication. An authentication system has

been implemented for the machine-to-machine communication using blockchain and cryptographic algorithms such as AES-CBC-128 and HMAC-SHA-256-128 which made the system more secure.

The algorithms were implemented using familiar IoT devices e.g. Arduino, Raspberry Pi and Kinetis MCU. The proposed protocol for authentication and encryption can be used by developers to secure their IoT devices and to ensure the security and privacy of their users.

Several testing procedures were performed to test its functionality and non-functionality attributes. Functionality tests include the implementation of test vectors from NIST to check the output's accuracy for both the Kinetis FRDM-K82F and Arduino 101. There were 10 test vectors for AES-CBC and 10 test vectors for SHA-256. Meanwhile, non-functionality tests include the measurement of processing n time in the IoT devices.

It was found that using a hardware accelerator coprocessor which can run independently of the CPU, particularly the Memory-Mapped Cryptographic Accelerator Unit (mmCAU) for Kinetis MCU, can significantly increase the processing time of cryptographic algorithms. The processing time of AES-CBC-128 improved by a factor of 5.5. Meanwhile, the processing time of SHA-256 improved by a factor of 7.9 as compared with the Arduino alone. However, there are limitations in the communication between the Arduino 101 and the peripheral device (Kinetis FRDM-K82F) such as its accuracy and transmission duration.

It is also worth noting that there are differences in the implementation of the blockchain for this study as opposed to the original implementation of blockchain which is based on Bitcoin. The researchers have determined that implementing a private blockchain with symmetric cryptography is more suitable for the particular application of IoT security on account of the limitation of resources in IoT devices and their traditional threat model.

## References

1. Abdulmonem, M.H., EssamEddeen, J., Zakhari, M.H., Hanafi, S., Mostafa, H.: Hardware acceleration of dash mining using dynamic partial reconfiguration on the ZYNQ board. In: 2020 32nd International Conference on Microelectronics (ICM), pp. 1–4 (2020). https://doi.org/10.1109/ICM50269.2020.9331815
2. Aggarwal, V.K., Sharma, N., Kaushik, I., Bhushan, B.H.: Integration of blockchain and IoT (B-IoT): architecture, solutions, and future research direction. IOP Conf. Ser. Mater. Sci. Eng. **1022**(1), 012103 (2021). https://doi.org/10.1088/1757-899X/1022/1/012103
3. Antonopoulos, A.M.: Mastering Bitcoin: Unlocking Digital Crypto-Currencies, 1st edn. O'Reilly Media Inc., Newton (2014)
4. Atiwa, S., Dawji, Y., Refaey, A., Magierowski, S.: Accelerated hardware implementation of blake2 cryptographic hash for blockchain. In: 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1–6 (2020). https://doi.org/10.1109/CCECE47787.2020.9255709
5. Chen, M., Wan, J., Li, F.: Machine-to-machine communications: architectures, standards and applications. KSII Trans. Internet Inf. Syst. **6**(2), 480–497 (2012). https://doi.org/10.3837/tiis.2012.02.002

6. Commons, W.: Authenticated encryption scheme (encrypt-then-mac), April 2015
7. Gaurav, K., Goyal, P., Agrawal, V., Rao, S.L.: IoT transaction security. In: 2015 5th International Conference on the Internet of Things. VMware Software India Pvt. Ltd. (2015)
8. Group, B.: Digital assets on public blockchains, March 2016
9. Kak, A.: Lecture notes on computer and network security (the advanced encryption standard). Lecture notes, April 2016. https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf
10. Katagi, M., Moriai, S.: Lightweight cryptography for the Internet of Things (2012)
11. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_19
12. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-hashing for message authentication. RFC Informational 2104, February 1997. https://tools.ietf.org/html/rfc2104
13. Kwak, S., Lee, J.: Implementation of blockchain based P2P energy trading platform. In: 2021 International Conference on Information Networking (ICOIN), pp. 5–7 (2021). https://doi.org/10.1109/ICOIN50884.2021.9333876
14. Mahony, A.O., Popovici, E.: A systematic review of blockchain hardware acceleration architectures. In: 2019 30th Irish Signals and Systems Conference (ISSC), pp. 1–6 (2019). https://doi.org/10.1109/ISSC.2019.8904936
15. Mohanty, S.P., Yanambaka, V.P., Kougianos, E., Puthal, D.: PUFchain: a hardware-assisted blockchain for sustainable simultaneous device and data security in the internet of everything (IoE). IEEE Consum. Electron. Mag. **9**(2), 8–16 (2020). https://doi.org/10.1109/MCE.2019.2953758
16. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009). http://www.bitcoin.org/bitcoin.pdf
17. Niaz, M.S., Saake, G.: Merkle hash tree based techniques for data integrity of outsourced data. In: GvD, pp. 66–71 (2015)
18. Nordic Semiconductor ASA: nRF24L01+ Single Chip 2.4 GHz Transceiver (2008), preliminary Product Specification v1.0
19. Reitz, K.: Requests: HTTP for Humans (2016). http://docs.python-requests.org/en/master/
20. Saha, S.S., Gorog, C., Moser, A., Scaglione, A., Johnson, N.G.: Integrating hardware security into a blockchain-based transactive energy platform (2020)
21. Zhong, G., Javaid, H., Saadat, H., Xu, L., Hu, C., Brebner, G.: FastProxy: hardware and software acceleration of stratum mining proxy. In: 2019 Crypto Valley Conference on Blockchain Technology (CVCBT), pp. 73–76 (2019). https://doi.org/10.1109/CVCBT.2019.00013