IoT-based Access Management Supported by AI and Blockchains

Eryk Schiller, Elfat Esati, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH—8050 Zürich, Switzerland
Emails: [schiller|stiller]@ifi.uzh.ch, elfat.esati@uzh.ch

***Abstract*—This work specifies, implements, and evaluates access management based on face recognition. The system developed uses Internet-of-Things (IoT) for video surveillance, Artificial Intelligence (AI) for face recognition, and Blockchains (BC) for immutable permanent storage and provides excellent properties in terms of image quality, end-to-end delay, and energy efficiency.**

***Index Terms*—Management of Access Control, Video Surveillance, Blockchains, Internet-of-Things, Artificial Intelligence**

I. Introduction

Blockchains (BC) [3] offer immutable storage of data records in a distributed Peer-to-Peer (P2P) network. BCs can help IoT infrastructures deal with centralization by removing a single point of failure [6], [16]–[18]. BCs bring significant advantages in terms of information provenance, non-repudiation, authenticity, and immutability, thus, increasing the overall information security [5].

Internet-of-Things (IoT) is transforming the world of things, which impacts many economic sectors, such as manufacturing, transportation, automotive, consumer goods, and healthcare [4]. IoT devices of new generation can run complex tasks in a distributed fashion but IoT comes with challenges such as platform centralization, security and privacy issues in communication protocols as well as vulnerability to device attacks, *e.g.,* Mirai, related to poor maintenance of IoT infrastructures [6], [12].

Artificial Intelligence (AI) may provide accurate data analysis in real-time. However, the design and development of an efficient AI-based data analysis tool comes with challenges too, such as centralization and transparency [21]. Therefore, integrating BCs with AI can produce a robust approach to resolve those issues. Transparency can be achieved by gathering AI decisions in a BC to provide a precise, immutable track ordered in time.

Therefore, the simultaneous application of BC, IoT, and AI, shows a successful synergy transforming data acquisition, analysis, and storage [5], [14], [20]. One may expect many proposals and architectures combining BC, IoT, and AI, however, there exist a few use-cases, wherein those three technologies complement each other. Therefore, the main goal of this paper is the design and development of a use-case, *i.e.,* an access management approach, using a BC-enabled IoT Architecture coupled with AI.

The remainder of this paper is organized as follows. Section II introduces the related work. While Section III provides new use-cases and specifies the architecture of the system, Section IV details its implementation and evaluates the performance of the system developed. Finally, Section V summarizes this work.

II. Related Work

This overview addresses most recent projects and research on the integration of AI, BC, and IoT.

A. Internet-of-Things

IoT transforms the interaction with everyday things [4]. Smart objects (*i.e.,* devices equipped with micro-controllers, sensors, and actuators) are often connected to the Internet allowing for information harvesting, sophisticated information processing at the fog, edge, or cloud level, and actuation in a given environment. Most micro-controllers [1], [9] provide minimal processing power and, in return, require low energy consumption, thus, allowing for a longer lifetime of an object on the single battery charge.

B. Internet-of-Things and Artificial Intelligence

Due to low processing power of IoT devices, there is a need for lightweight approaches in AI. This gives rise to new concepts, such as Tiny Machine Learning [10] or Tiny Deep Learning [13], which are able to operate on constrained devices. As an example, MCUNet [2] brings deep learning to low capacity devices for image, audio, or video recognition. Another study [23] worked on image processing and cloud offloading. Two approaches with deep learning were tested, *i.e., (i)* cloud offloading and *(ii)* deep learning on IoT devices, from the real-time and energy efficiency point of view. Executing machine learning on an IoT device consumed more energy compared to cloud offloading. However, it comes with drawbacks such as latency starting with 2 seconds and going up to 5 seconds, which is higher compared to AI on IoT devices. Furthermore, it infers variability in response times, making it unreliable in real-time applications.

Esp Eye [9], equipped with Tensilica LX6 dual-core processor, is to our knowledge the first micro-controller that performs real-time face recognition. Esp Eye supports Esp Who [22], which performs both: face detection and face recognition. Esp Who implements MTMN for face detection, which refers

to both MT-CNN (Multi-Task Cascaded Convolutional Networks) [24] and MobileNets (MN) [11] as well as Face Recognition model based on Convolutional Neural Network (FRMN) [19].

C. Internet-of-Things and Blockchains

Several research attempts close the gaps of IoT systems by removing *(i)* the centralized control as well as tackle *(ii)* the problem of provenance, non-repudiation, authenticity, and immutability in IoT data streams with the help of BC [6], [16]–[18].

D. Artificial Intelligence and Blockchains

There is a high research interest focused on BC and AI analyzed in various domains and applications. Like IoT, the AI domain also suffers from various problems such as security, privacy, transparency, explainability, and trustworthiness. A BC complementing an AI system promises to close those shortcomings [7]. However, most of research items in this domain do not develop use-cases or provide actual implementations.

E. Artificial Intelligence, Blockchains, and Internet-of-Things

Several attempts shed the light on the benefits of the IoT, BC, and AI convergence through reviews or explorations but lack a concrete implementation in use-cases [5], [14]. However, BlockIoTIntelligence [21] proposes an architecture that utilizes BCs and AI in IoT. BlockIoTIntelligence aims to achieve decentralized big data analysis considering security and centralization issues of IoT applications in various domains such as smart city, healthcare, and intelligent transportation. BlockIoTIntelligence claims high accuracy, reasonable latency, and improved security.

F. Approaches Similar to This Work

[15] designs and implements a camera-based sensor for room capacity monitoring. That work aims to count the number of people in a room with the help of a Raspberry PI (RPI) [1] equipped with a camera and Machine Learning (ML).

G. The Newly Proposed Approach

This approach differs from related work implementations by providing a solid use-case in access management. Furthermore, this approach combines all three techniques at the same time, *i.e.,* AI for image processing, BC for immutable tamper-resistant storage (*e.g.,* for auditing reasons), and IoT for data harvesting (*i.e.,* providing the video stream). Furthermore, this approach follows the novel TinyML paradigm, in which face detection and recognition run directly on an IoT device.

III. Use-Case and Architecture

Driven by the specific use-case the description of the architecture follows as a generalized approach.

A. Use-case

The system employs real-time face detection and recognition of authorized individuals to grant access to an institution. The access is granted or denied by the system automatically. When access is granted, the door to an institution may open automatically without any intervention. However, when access is denied, the door will remain closed, preventing the user from accessing a given resource. When an individual needs access, their picture is taken, processed, and stored in the immutable BC, preventing future tampering with data and enabling immutable storage that provides a solid foundation for auditing purposes.

B. Architecture

Based on hardware components available, the software architecture was designed and reasons for these design decisions taken to materialize the idea of IoT-based AI surveillance with BC are provided. Furthermore, different approaches, technologies, and communication protocols, considered throughout the design decisions, are described.

1) Hardware Components: The system is composed of an IoT device, an IoT Gateway (GW), and infrastructure supporting a BC. The image capturing and face recognition are handled by Esp Eye [9] IoT device, which is based on a double-core architecture supporting the 240 MHz CPU frequency, equipped with a 2-Megapixels OV2640 camera and an IEEE 802.11 network adapter. The IoT GW, equipped with an IEEE 802.11 network adapter as well, serves as the middle man, which waits for data (*i.e.,* images and meta-data) coming from Esp Eye devices to be inserted into the BC. The communication between Esp Eye devices and the IoT GW is achieved through the IEEE 802.11 network. To provide integrated experimental facilities, a BC runs locally on the IoT GW. However, the BC can be spanned among multiple machines organized as a BC network on the Internet. Since HyperLedger Fabric (HLF) [3] was selected as the BC platform, and its official build is provided for Intel-based CPUs, the IoT GW is an Intel-based machine. It is, however, expected that HLF might run on low-capacity devices, such as ARM-based RPI devices. To this end, the HLF developer (*i.e.,* IBM) shall provide an appropriate compilation environment to support ARM-based devices as well.

2) Software Architecture: Fig. 1 depicts a high-level overview of the system architecture. It is essential to mention that the software design shall be compatible with many underlying hardware architectures. However, the Esp Eye is required for the success of this project.

Esp Eye uses the OV2640 camera to capture images. Images are sent to the face detection and face recognition modules. Face detection (*i.e.,* MTMN) and face recognition (*i.e.,* FRMN) run directly on the Esp Eye device. The outcome, *i.e.,* an images accompanied with meta-data, is provided toward the video streaming service, which connects with the Esp Server running on the IoT GW. The Esp Server provides the image toward the ESP Plugin (*cf.* the IoT GW), which submits the Transaction (TX) to the immutable ledger with the help

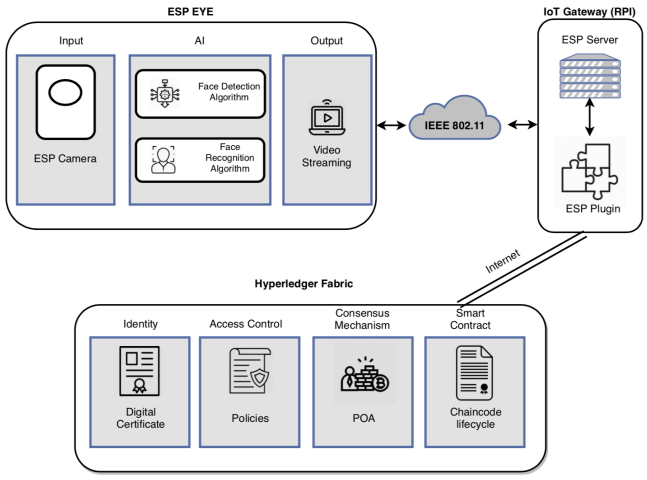

Fig. 1: System Architecture

of the HLF Software Development Kit (SDK). HLF stores images coming from Esp Eye devices in immutable storage. HLF runs typical services required to run a BC, ieidentity management, access control, and a consensus mechanism. Finally, HLF maintains a smart contract, which is responsible for handling data received from its clients (*i.e.,* Esp Plugin residing on the IoT GW). HLF matches very well the use-case (*cf.* Section III-A), because a private permissioned BC is better suited for video surveillance due to privacy and BC block size reasons.

3) Communication Protocols: Hypertext Transfer Protocol (HTTP) is employed between Esp Eye and the IoT GW to communicate in the client-server architecture. The Representational State Transfer (REST) architectural style is used for inter-machine communication because REST is considered a lightweight communication paradigm. The JavaScript Object Notation (JSON) data interchange format is employed to carry the actual information in the system.

C. Implementation

The implementation determines the data flow between Esp Eye, IoT GW, and HLF as shown in Fig. 2, where data circulates from the left (*i.e.,* the Esp Eye device) to the right (*i.e.,* HLF) [8]. Several Application Programming Interfaces (API) and data structures are used to materialize the system.

1) Esp Eye Transmission Overview: Esp Eye analyses each frame with the MTMN face detection and if a face is detected, the image is provided towards the FRMN face recognition algorithm. Hence, if a face is detected, but not necessarily recognized, the HTTP client is activated, while Esp Eye devices act as clients communicating with the remote HTTP server on the IoT GW.

The video service takes the image equipped with meta-data, *i.e.,* device Identifier (ID), detected face ID, timestamp, and forwards it to the IoT GW. This is achieved with the help of a REST API call using the HTTP POST request. The node.js-based server located in the IoT GW receives the request (*e.g.,* an image with the details of a person detected provided as meta-data). Furthermore, the role of the node.js server is to properly acknowledge the successful reception of the transmission coming from Esp Eye devices.

There are four significant parameters to be stored in the BC reflected in the JSON document provided by Esp Eye devices. First, as multiple Esp Eye devices may be employed in access management, the device ID is essential, since the framework needs to distinguish particular devices from which the information is coming. Second, a face ID is needed, allowing for personal identification without processing the captured frame again. The successfully identified person on the sensor implies that access was granted to given resources protected by this access management system. Additionally, the timestamp identifies the time moment when the person is detected. Finally, the image frame is provided. All four parameters are sent as a JSON document. After receiving the document, the node.js server located on the IoT GW responds with a status code.

Esp Eye is programmed in C++ with the help of the Arduino Integrated Development Environment (IDE), which has to be equipped with the ESP32 board support. The device issues HTTP requests using the POST method towards the HTTP API exposed by the the IoT GW using the *application/json* method. JSON is handful for sending plain text or any other data types. Since HLF also uses a JSON format to store assets, this work converts the image into a data type supported by JSON as well. Hence, the best option is to store the image as a string. To this end, the image is converted to BASE64.

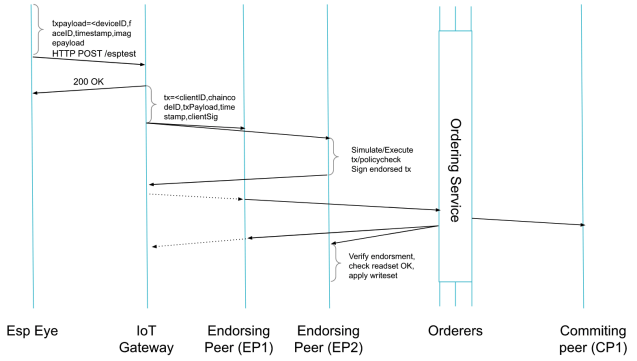

Fig. 2: Sequence Diagram Showing TX Execution

2) HLF Transaction Submission: The node.js HTTP server is responsible for receiving the data and forwarding it to the HLF. In order to interact with HLF, the node.js-based server uses the HLF SDK providing an API to submit TX to the ledger. The process of submission takes place right after the image has arrived from an Esp Eye device. The HLF TX data, also referred to as an asset, is a collection of JSON key-value pairs. Since the JSON format for the image transfer is used from the start, node.js may forward a similar JSON file to the BC.

This work implements a chaincode (*i.e.,* smart-contract) running on HLF. To run a smart-smart contract, *(i)* HLF has to be configured and running, *(ii)* the HLF channel is established, *(iii)* the chaincode implemented in this work and its endorsement policy is deployed within the channel. Each Esp Eye is assigned with a public/private key pair, which is used to sign the HLF TX. Furthermore, every Esp Eye possesses a valid X.509 certificate, *i.e.,* public key of the node signed with the HLF Certificate Authority (CA), appropriately recognized by the HLF Membership Service Providers (MSP). For more information, please consult the HLF documentation [3].

The IoT GW acts as an HLF client. Therefore, the IoT GW initiates the HLF TX with a JSON document received from Esp Eye (*cf.* Sect. III-C1), which becomes the TX payload. In this work, however, the IoT GW retrieves the key pair of a given Esp Eye device and signs the TX on behalf of the device using the HLF SDK. Although TX signing is possible locally on the Esp Eye device, this was not implemented here, since the HLF client SDK is not ported for Esp Eye devices yet.

This work configures two HLF endorsers (*i.e.,* EP1 and EP2, *cf.* Fig. 2). They receive the TX proposal, which includes a client ID, the chaincode ID, the TX payload, a timestamp, and the client signature. The endorsement policy in this work requires both endorsement peers to endorse the received HLF TX before the TX might be submitted toward the ordering service. First, EP1 and EP2 check the format of the TX. Second, every TX has to posses a valid signature of a client appropriately registered within the MSP. Third, the client has an authorized member of the HLF channel. When all conditions have been verified, the endorsers invoke the chaincode using the JSON document received. Eventually, the TX is executed, however, it does not yet update the ledger. Now, the endorsers sign the proposed TX and send it back to the HLF client on the IoT GW. The intent of the HLF client is to submit the TX to the ordering service and update the ledger. Before the HLF submits the final version of the TX, HLF clients send the TX endorsed toward the ordering service. Now, the TX is equipped with signatures of endorsing peers. While the ordering service may receive TXs from other clients or ESP devices, the ordering service orders TXs according to a sequence number and packages them into blocks. When the maximum number of TXs allowed in a block is reached or the maximum block-time has passed blocks are sent to committing peers to be included in the ledger for an immutable storage. Upon receiving a broadcast message with the created block from the orderers, committing peers verify the signatures of ordering nodes within a given block. If the committing peers fail to verify the signature of ordering peers, the ledger will not include and rejects the newly created block.

IV. Evaluation

The evaluation integrates two Esp Eye sensors (dual-core Tensilica LX6 processor with a maximum frequency of 240 MHz, 8 MB PSRAM, and 4 MB flash) and regular macOS based computer having 2-core Intel Core i5 running at 2.7 GHz, 512 GB SDD disk, and 8 GB RAM. To begin testing, several steps are recommended. *(i)* The Esp Eye sensors are programmed and powered up using an external charger power bank; 10 face profiles are uploaded on the device. *(ii)* HLF is started with the help of docker containers. Currently, one committing peer and two endorsing peers are configured. The ordering service is set to *solo*, *i.e.,* one node ordering HLF TXs. The HLF channel is configured and JS-based chaincode implemented is deployed. *(iii)* The IoT GW starts the node.js-based HTTP server listening on port 8585. For the implementation details of all components, please consult [8].

A. Image Quality

The OV2640 camera embedded in Esp Eye is also supported by the Esp Who platform. The camera can be configured in terms of frame size and pixel format. The FRAME-SIZE_QVGA (320 px$\times$240 px) is selected. The sensor can deliver 5.2 fps (*i.e.,* 190 ms to deliver an image) of this image quality.

B. Processing Delay of Face Detection and Recognition

The idea of performing face detection and recognition on Esp Eye is a novel approach because typically face detection and recognition run either on the cloud or on a local computer.

Face detection and recognition takes around 1 s on Esp Eye. Around 120-125 ms is needed to receive the image from the sensor. Furthermore, Esp-Eye required around 50-55 ms to perform the face detection with MTMN, when no face is provided in the image. If MTMN does not detect faces, FRMN is not activated, which results in the 0 ms FRMN completion time. Typically, when the face is detected, MTMN requires around 150-170 ms to detect a face and FRMN requires around 650-700 ms to complete. Therefore, the total time including face detection (*i.e.,* MTMN) and recognition (*i.e.,* FRMN) results in 920-1000 ms, which includes the image acquisition, MTMN, and FRMN. The FRMN algorithm displays around 99% percent accuracy, however, more studies are needed to evaluate its performance on face detection in a real-system.

C. End-to-End Processing Delay

This work measures the end-to-end delay by printing the timestamp at different individual processing steps. *(i)* The image is captured, but no face detection/recognition has been performed yet. *(ii)* A face has been detected/recognized and the image is sent to IoT GW. *(iii)* The image has reached the IoT GW. *(iv)* The image has been submitted to HLF. *(v)* The image is inserted in the ledger and the ordering service is finished.

Table I shows all processing stages starting with the Esp Eye image capturing until the image reaches the ledger. Sending the image from Esp Eye to the IoT GW takes almost 2 s due to low throughput of the TCP communication on Esp Eye. HLF consumes little more than 2 s. This is influenced by two configurable parameters, *i.e.,* BatchTimeout and BatchSize. This work configures BatchTimout at 2 s. The end-to-end delay experienced in the system is 5.3 s from the moment

image is taken until the inclusion of the BC TX in the BC. Therefore, almost real-time use-cases can be supported using HLF as a communication backend.

TABLE I: End-to-End Delay Measurements

Action Point	Timestamp
Image is captured by Esp Eye	**2021-01-22T14:39:26Z**
Image is being sent to IoT Gateway	**2021-01-22T14:39:27Z**
Image is received by IoT Gateway	**2021-01-22T14:39:28.947Z**
Image is submitted to Fabric	**2021-01-22T14:39:29.111Z**
Image has reached all the peers	**2021-01-22T14:39:31.313Z**
Total Time	**5.313 s**

D. Energy Efficiency of Esp Eye

The experimentation setup was used to measure the energy consumption of Esp Eye against image quality, face detection, and face recognition. After several tests with different image qualities and parameters, there was no difference experienced in Esp Eye energy consumption. Throughout the experiment, the energy consumption remained at the constant level of 600 mW (in total, the device consumed 600 mWh within an hour of operation), which allows for a 7-hour operation on an alkaline battery of 4,200 mWh capacity. Furthermore, there is no difference in energy consumption when a face is recognized or no face is detected.

V. Summary, Discussion, and Future Work

This paper provides the first access management system, which utilizes Artificial Intelligence (AI), Blockchains (BC), and Internet-of-Things (IoT) in an integrated use-case. Thus, the user needs to present his/her face in front of a camera to access a resource. The system takes the image of that person and checks, whether this given user has the right to access a given resource. Face detection and recognition are performed directly on the IoT device. To detect faces, a MT-CNN (Multi-Task Cascaded Convolutional Network) with MobileNets (MN) was deployed. Furthermore, the Face Recognition model is based upon a Convolutional Neural Network (FRMN). To establish a good level of transparency, the AI decisions on access rights as well as images taken by the sensor are stored in the immutable, tamper-resistant storage implemented with the help of the HyperLedger Fabric (HLF). The performance of the system was evaluated at an excellent level, where a 5.3 s end-to-end delay is reached.

Acknowledgements

This paper was supported partially by *(a)* the University of Zürich UZH, Switzerland, and *(b)* the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No. 830927, the CONCORDIA project.

References

[1] "Raspberry Pi 3 Model B." https://www.raspberrypi.org/products/raspberry-pi-3-model-b, last visit: October 2, 2019.

[2] D. Ackerman, "System Brings Deep Learning to "Internet of Things" Devices," https://news.mit.edu/2020/iot-deep-learning-1113, [Online; accessed 10-December-2020].

[3] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018.

[4] H. F. Atlam, R. J. Walters, and G. B. Wills, "Intelligence of Things: Opportunities Challenges," in *2018 3rd Cloudification of the Internet of Things (CIoT)*, 2018, pp. 1–6.

[5] H. F. Atlam, M. A. Azad, A. G. Alzahrani, and G. Wills, "A Review of Blockchain in Internet of Things and AI," *Big Data and Cognitive Computing*, vol. 4, no. 4, 2020. [Online]. Available: https://www.mdpi.com/2504-2289/4/4/28

[6] A. Banafa, "IoT and Blockchain Convergence: Benefits and Challenges," *IEEE Internet of Things*, Jan. 2017.

[7] T. N. Dinh and M. T. Thai, "AI and Blockchain: A Disruptive Integration," *Computer*, vol. 51, no. 9, pp. 48–53, 2018.

[8] E. Esati, "IoT-based Access Management, GitHub Repository," https://github.com/eesati/Master-Thesis, Jan. 2021.

[9] Espressif, "ESP-EYE Development Board," https://www.espressif.com/en/products/devkits/esp-eye/overview, [Online; accessed 10-December-2020].

[10] S. Gopinath, N. Ghanathe, V. Seshadri, and R. Sharma, "Compiling KB-Sized Machine Learning Models to Tiny IoT Devices," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2019. New York, NY, USA: ACM, 2019, pp. 79–95. [Online]. Available: https://doi.org/10.1145/3314221.3314597

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[12] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *IEEE Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[13] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "Mcunet: Tiny deep learning on iot devices," *arXiv preprint arXiv:2007.10319*, 2020.

[14] B. Parker and C. Bach, "The Synthesis of Blockchain, Artificial Intelligence and Internet of Things," *European Journal of Engineering and Technology Research*, vol. 5, no. 5, pp. 588–593, May 2020. [Online]. Available: https://www.ejers.org/index.php/ejers/article/view/1912

[15] S. A. I. Quadri and P. Sathish, "IoT based Home Automation and Surveillance System," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2017, pp. 861–866.

[16] S. Rafati-Niya, E. Schiller, I. Cepilov, and B. Stiller, "BIIT: Standardization of Blockchain-based I2oT Systems in the I4 Era," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–9.

[17] E. Schiller, E. Esati, S. Rafati-Niya, and B. Stiller, "Blockchain on MSP430 with IEEE 802.15.4," in *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 345–348.

[18] E. Schiller, S. Rafati-Niya, T. Surbeck, and B. Stiller, "Scalable Transport Mechanisms for Blockchain IoT Applications," in *2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium)*, 2019, pp. 34–41.

[19] S. Shang, H. Liu, Q. Qu, G. Li, and J. Cao, "FRMN-A Face Recognition Model Based on Convolutional Neural Network," in *IOP Conference Series: Materials Science and Engineering*, vol. 585, no. 1. IOP Publishing, 2019, p. 012101.

[20] S. Singh, P. K. Sharma, B. Yoon, M. Shojafar, G. H. Cho, and I.-H. Ra, "Convergence of Blockchain and Artificial Intelligence in IoT Network for the Sustainable Smart City," *Sustainable Cities and Society*, vol. 63, p. 102364, 2020.

[21] S. K. Singh, S. Rathore, and J. Park, "BlockIoTIntelligence: A Blockchain-enabled Intelligent IoT Architecture with Artificial Intelligence," *Future Generation Computer Systems*, vol. 110, 09 2019.

[22] O. Source, "ESP-WHO Overview," https://github.com/espressif/esp-who, [Online; accessed 23-September-2020].

[23] J. Tang, D. Sun, S. Liu, and J. Gaudiot, "Enabling Deep Learning on IoT Devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[24] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.