Year: 2020

Blockchain on MSP430 with IEEE 802.15.4

Schiller, Eryk ; Esati, Elfat ; Stiller, Burkhard

Abstract: This work develops an integration of Blockchains (BC) with the Internet-of-Things (IoT) using a highly constrained TelosB IoT platform based on the MSP430 processor family and CC2420 IEEE 802.15.4-compliant radio interfaces. The system is evaluated in an indoor office environment focusing on overhead and energy efficiency of BC transaction (TX) transmissions.

Blockchain on MSP430 with IEEE 802.15.4

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

***Abstract*—This work develops an integration of Blockchains (BC) with the Internet-of-Things (IoT) using a highly constrained TelosB IoT platform based on the MSP430 processor family and CC2420 IEEE 802.15.4-compliant radio interfaces. The system is evaluated in an indoor office environment focusing on overhead and energy efficiency of BC transaction (TX) transmissions.**

I. Introduction

Blockchains (BC) as a storage of data in Internet-of-Things (IoT) use-cases (*e.g.,* supply chain monitoring, smart industries, or industrial pollution monitoring) provide security in terms of data authenticity, non-repudiation, and immutability. Heavily constrained IoT devices may send data to the BC for tamper-resistant storage. Previous studies delivered promising results on the integration of IoT devices as wallets in BC ecosystems using LoRa, NBIoT, or LTE Cat. M network technologies [8]. This study here demonstrates the integration of the BAZO BC [9] with the TelosB platform based on the MSP430 processor family and the IEEE 802.15 communication standard [7]. TelosB devices are programmed to work and send transactions (TX) to the BC over IEEE 802.15.4 radio interfaces, where each TX is signed by the Elliptic Curve Digital Signature Algorithm (ECDSA). Due to the fact that the cryptographic primitives are costly, BC TXs are kept open for several data chunks reported by the sensor. The arriving signature closes an open BC TX, consisting of outstanding data chunks, which results in a trade-off between real-time delivery of BC TXs and the protocol overhead as well as energy efficiency of the mechanism developed. Therefore, the solution is currently not meant for highly real-time applications.

This work is organized in Sect. II with a compact overview of related work. While Sect. III describes the architecture, Sect. IV describes the experimental setup and evaluates the performance. Finally, Sect. V concludes this work.

II. Brief Related Work on IoT and Blockchains

A TelosB mote [7] is a wireless sensor node of an open source platform designed for IoT experimentation in research. It is a low powered device based on the MSP430 microcontroller family equipped with 10 kByte of Random Access Memory (RAM) and 48 kByte of flash. Therefore, TelosB may be categorized as a Class 1 IoT device [2]. Furthermore, TelosB supports IEEE 802.15.4 compliant interfaces with the Maximum Transmission Unit (MTU) of 127 Byte.

A Raspberry Pi 3B (RPI) [6] is a small computer equipped with computing essentials, *e.g.*, the ARM Cortex CPU, RAM but not necessarily of the latest generation, having little energy requirements. As an example, instead of a hard drive, a RPI is equipped with a flash memory card, on which an Operating System (OS) may be installed. It also offers USB connectors, a video output, and a WiFi adapter.

A. Operating Systems for IoT

Contiki [3] requires 10 kByte of RAM and 30 kByte of Read Only Memory (ROM) and runs an event driven kernel facilitating programming through the protothread library.

B. Blockchains for IoT

BIIT [8] is a BC architecture for IoT applications describing the role of software components. It also provides guidelines on the organization of the protocol transporting BC TXs using different network technologies, however, BIIT does not consider the IEEE 802.15.4 physical layer yet. Please consult BIIT for more details on its design as well as the state of the art in BC-IoT integrated systems.

By default, BIIT uses the BAZO Proof-of-Stake BC [9], which was chosen as a default BC platform, because unlike Proof-of-Work (PoW) BCs, it is highly energy efficient. BAZO, therefore, perfectly matches the requirements of IoT deployments, however, may result in a lower security in comparison to PoW BCs. Furthermore, a PoS BC might significantly reduce block time to further reduce the TX delay.

III. Architecture

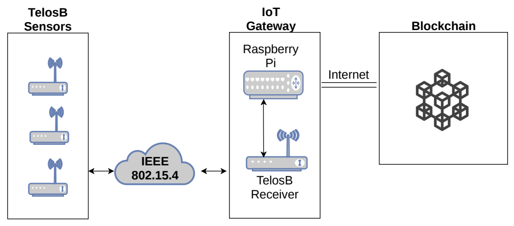


Figure 1. Hardware Architecture

A global picture of the hardware architecture, including devices and their interactions, (*cf.* Fig. 1) shows TelosB [7] and Raspberry Pi 3B [6] devices. They setup the BC, in which BC TXs originating at TelosB-based IoT devices are stored immutably in the Bazo BC [9]. RPI is a very lightweight platform, serves as a Bazo BC hub, and can either run a Bazo BC instance locally or provide a pass-through toward BC miners in the Internet. However, by default RPI is not IEEE 802.15.4 compliant. Thus, one TelosB dongle is directly attached to the RPI through the USB port to provide the RPI device with the IEEE 802.15.4 interface. Note that an RPI is a small computer of the ARM architecture providing a 1.2 GHz ARM quad-core CPU and 1 GB RAM running a regular general purpose OS.

A. Operating System

The OS for IoT devices, such as for the TelosB device, should have a small memory footprint, because TelosB only offers 10 kByte of RAM and 48 kByte of flash memory. With such constraints the TelosB platform does not support general purpose OSes. Here, Contiki 2.7 [3] is used as the IoT software platform to control TelosB sensors and the TelosB receiver attached to the RPI device.

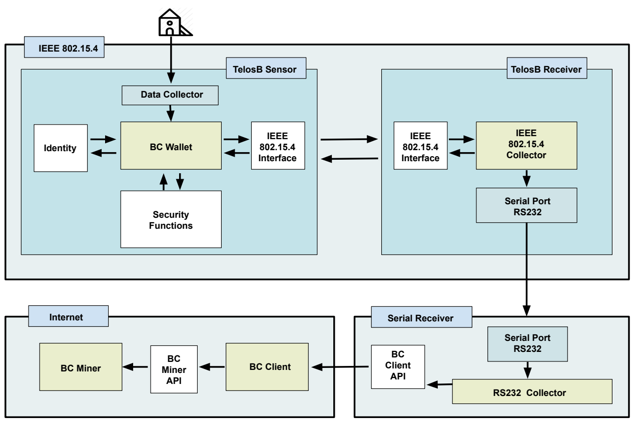


Figure 2. Software Architecture

B. Software Architecture

The software composition loosely follows the BIIT architecture [8] (*cf.* Fig. 2). Software components enable the information propagation via two main communication protocols. The IEEE 802.15.4 protocol is used for the communication between TelosB devices submitting BC TXs to the RPI, while the Internet Protocol (IP) is employed to forward TXs between the RPI and the network of BC miners residing in the Internet. Currently, only one BC miner is run locally on the RPI.

The information chain begins with raw data collected from the environment through a TelosB Sensor device using the Data Collector. The BAZO BC Wallet, implemented as a Contiki protothread, obtains data from from the Data Collector and assembles the BAZO BC TX. To assemble the BAZO BC TX, the BC Wallet uses the identity stored on the device, *i.e.,* the 64 byte-long Ed25519 key pair [1] and the set of security functions. SHA3-256 [5] provides cryptographic digests for several fields in the 77 Byte TX header, *e.g.,* the digest of the source/destination public keys, while SHA3-256 and Ed25519 together deliver the 64 byte-long transaction signature (witness data). The use of SHA3-256 is justified, because it is a fast and energy-efficient cryptographic primitive. Furthermore, the Ed25519 ECDSA is much faster and requires shorter keys than regular Rivest-Shamir-Adleman (RSA) cryptosystems, while offering a similar level of security. These properties make SHA3-256 and Ed25519 excellent candidates for the set of security functions supported by an IoT system.

To send a BC TX, the BC Wallet uses the IEEE 802.15.4 interface. The Medium Access Control (MAC) destination address of the TelosB receiver and the BC TX destination addresses are pre-configured on the TelosB sensor. The TX is propagated through the air in the form of IEEE 802.15.4 data chunks. The BAZO BC TX header, however, exceeds the length of the IEEE 802.15.4 MTU. Thus, a single BC TX has to be fragmented into several data chunks, which are received by the TelosB receiver installed on the RPI. The TelosB receiver implemented collects IEEE 802.15.4 packets originating on the TelosB sensor using IEEE 802.15.4 interfaces. All packets are forwarded to the Contiki protothread-based IEEE 802.15.4 Collector, which sends, in turn, the payload received to the RPI device through the serial port.

The Serial Receiver, a regular Node.js[1] application running on the RPI, constantly monitors the serial port. It gradually receives data chunks sent by TelosB sensors one by one using its RS232 Collector. When several chunks are received and the BAZO BC TX sent is reassembled into one piece, the Serial Receiver provides the complete TX to the BAZO BC Client using the BC Client Application Programming Interface (API). The BC Client forwards the TX to the BAZO BC miner for permanent storage using the appropriate API. The communication between the Serial Receiver, BC Client, and BC Miner is IP-based. Here, for a simplified setup, Serial Receiver, BC Client, and BC Miner run on the same RPI. However, BC Client and BC Miner may run on different hosts, *e.g.,* a BC Client in the edge, while the BC Miners may reside within the cloud.

C. IEEE 802.15.4 Network Functions

The Rime stack [4] of Contiki is an alternative network stack used when the overhead of IP becomes too heavy. Since this work operates TelosB devices, the Rime unicast transmission model matches well the constrained nature of the MSP430 environment and is used for TelosB sensors and receivers attached to the RPI. The senders hard-encode the Rime address of the destination and directly send data chunks to the TelosB receiver. The job of the transmitter is to retransmit a given chunk of data until the corresponding acknowledgment (ACK) indicating a successful transmission arrives from the TelosB receiver. Every chunk is equipped with a cyclic sequence number, which in combination with the ACK mechanism and retransmissions provide a reliable delivery.

D. BAZO Blockchain Transaction Transmission Scheme

Fig. 3 outlines the communication protocol between TelosB sensors and the Serial Receiver. During the initialization phase the Ed25519 public key of the TelosB sensor is sent to the Serial Receiver. This allows the Serial Receiver to register a wallet in the BC in case the TelosB sensor is connected with the infrastructure for the first time.

[1]https://nodejs.org/

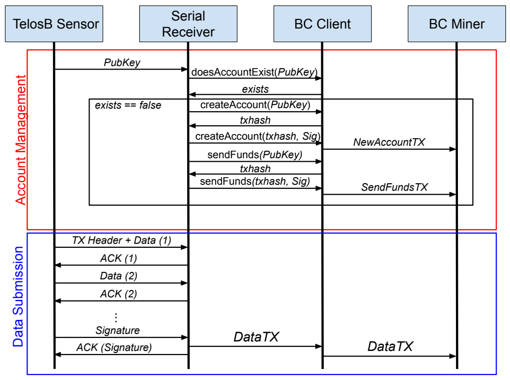


Figure 3. Software-based Communications Architecture

Every distinct BAZO BC TX consists of several chunks of data exchanged between a TelosB sensor and the Serial Receiver on the RPI. Initially, the TelosB sensor opens a BAZO BC TX by issuing a header and the first data fragment. When new data is available on the TelosB sensor, it submits a new chunk of data in a given open TX. In the last step, the TelosB sensor closes the open BAZO BC TX by sending an Ed25519 signature, which is used to verify the integrity and authenticity of the entire TX.

Due to the fact that the SHA3-256-based TX Identifier (TXID) is derived over the entire TX structure, all submitted chunks within an open TX have to be cached on both connection ends, *i.e.,* the TelosB sensor and Serial Receiver (*cf.* Fig. 4). When all chunks of data in a given transaction are shared between the sender and receiver, the resulting TXID will be equal on both connection ends (refer to the reliable data delivery as of Sect. III-C). Finally, the Ed25519 ECDSA signature computed over the TXID using the Ed25519 private key stored on TelosB sensors certifies the TX.

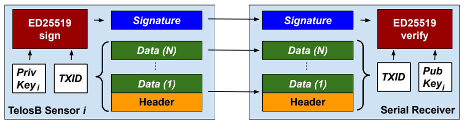


Figure 4. Overall Architecture

The Serial Receiver may confirm the TX data integrity. To this end, the Serial Receiver first derives the TXID, *i.e.,* the SHA3-256 digest of the TX. Second, it verifies the signature of the TXID provided by the TelosB sensor. This employs the ED25519 ECDSA *verify procedure* fed with the public key previously registered with the BAZO BC by the sensor in the initialization phase, the TXID, and the corresponding signature provided by the TelosB sensor. When TXs are successfully verified by the Serial Receiver, they are passed to the BC Client using the BC Client API, which in turns forwards them to BC Miners for permanent immutable storage.

IV. Evaluation

The testbed of this work contains TelosB Sensor nodes powered with AA batteries, however, only one node at a time is used to send BAZO BC TXs. Furthermore, one grid-powered RPI device is attached to the TelosB Receiver. The RPI runs the Raspberry Pi OS equipped with the Node.js framework.

SHA3-256[2] and TweetNaCl Ed25519 [1] were provided as security functions on the TelosB sensor. The corresponding compilation options for Ed25519 are set according to [8] resulting in 45,928 bytes programmed on the flash memory of the sensor. The execution time of a single Ed25519 sign function is costly and takes around 280 s on a TelosB device. There are, however, more efficient implementations of Ed25519 ECDSA such as Arduino Cryptographic Library (ACL)[3]. While TweetNaCl displays poor performance, the code is highly portable and works on different architectures with no changes. The low performance of Ed25519 functions (requiring several seconds to complete occupying the CPU at 100%) is the reason why the TelosB device keeps the BAZO BC TX open until several data chunks (*e.g.,* measurements) are accumulated in a single TX.

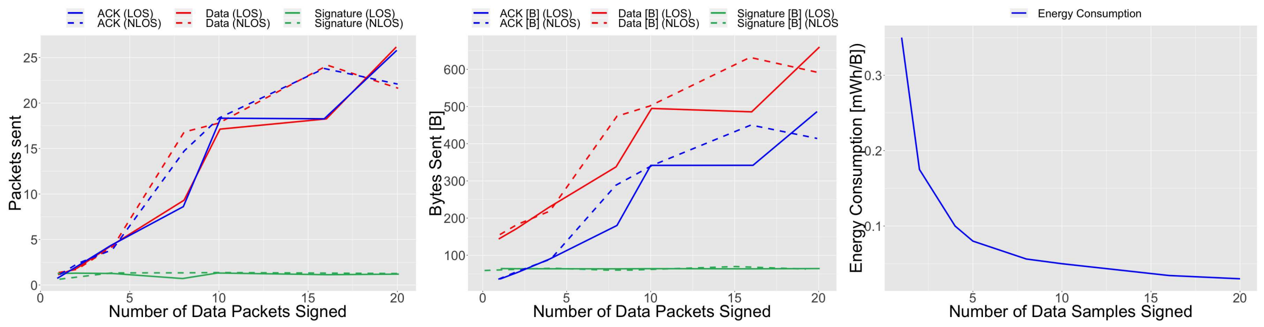
The BAZO BC Wallet uses the SHA3-256 and Ed25519 functions to calculate the TXID and Ed25519 signatures of a given TX. The TelosB Receiver on the RPI, however, does not need any BC-related security features, since it only passively forwards data chunks received on the IEEE 802.15.4 interface to the serial port. To increase data privacy in the IEEE 802.15.4 network, the symmetric Advanced Encryption Standard (AES) might be additionally provided on the TelosB Sensor and Receiver. The Serial Receiver does, in turn, need SHA3-256 and ED25519 cryptographic primitives to verify a BAZO BC TX delivered by TelosB sensor nodes. As the Serial Receiver is a Node.js application, the regular JS-SHA3 and Ed25519 packages were installed on the RPI using the Node.js Package Manager (NPM). The BC Client and BC Miner are directly ported from [8], as they did not require any implementation changes.

The TelosB device sends traffic with a bandwidth of 250 kbit/s using channel 26 (2.48 GHz) and the Contikimac channel access method [3]. TelosB employs the 802.15.4 compliant CC2420 Transceiver chip [7]. The default transmission power of CC2420 equals 0 dBm is also configured here.

Two experimental setups abbreviated as Line Of Sight (LOS) and Non Line Of Sight (NLOS) use a single office or two neighboring offices, respectively. Every BC TX is composed of a configurable number of data packets being signed, while every data chunks weights 20 Byte. A packet sent is retransmitted up to 10 times if the corresponding ACK does not arrive.

The evaluation of the BAZO BC TX scheme (*cf.* Fig. 5) compares the transmission performance based on the number of packets sent including retransmissions and the traffic volume measured in bytes. The packet overhead, *i.e.,* the

[2]https://github.com/brainhub/SHA3IUF
[3]https://rweather.github.io/arduinolibs/crypto.html



(a) Packets Sent in the IEEE 802.15.4 Network (b) Bytes Sent in the 802.15.4 Network (c) Energy Consumption per Data Byte Exchanged in the 802.15.4 Network

Figure 5. Performance of the BAZO BC Implementation on the TelosB Sensor Family

fraction of additional packets sent on top of data packets, *cf.* the x-axis, in the LOS scenario is constantly at around 100% (*e.g.,* 106% for a BC TX consisting of 16 data chunks), while higher packet values are experienced in NLOS scenario due to retransmissions. This substantial packet overhead is the consequence of ACK messages confirming every packet sent. The payload overhead, *i.e.,* additional traffic volume sent on top of data packets signed and sent, *cf.* x-axis, comprises mostly of the heavy weighted BAZO BC header, the Ed25519 signature, and the ACK overhead. The traffic volume of around 14 times the size of the data chunk is registered, when a TX comprises of one chunk. It sharply decreases with the number of data samples signed in one TX reaching only 2.8 times the size of 16 data chunks, *i.e.,* when 16 chunks are gathered in one TX.

The measurements of energy consumption (*cf.* Fig. 5) indicate that energy efficiency improves with an increasing number of data chunks enclosed in a single TX. TinyNaCl is computationally expensive and, therefore, introduces this overhead. In conclusion, sending multiple data chunks signed with one Ed25519 signature is preferable. The energy efficiency of this platform is slightly lower than those findings as reported in [8], since BIIT uses a more efficient Ed25519 implementation. Overall, the battery powered TelosB device with two AA batteries of 1,200 mWh each, waking up once daily to report measurements and including 20 data chunks per BC TX can run for approximately 10 years assuming nearly 0 A deep sleep current.

V. Findings and Conclusions

According to the authors' best knowledge, this work presents the first prototypical implementation of a BC on the constrained MSP430 platform operating in an indoor environment with IEEE 802.15.4 CC2420 interfaces. It is operational in an IoT setting and integrates basic IoT communication capabilities with BCs. The packet and byte overhead of the TX delivery was established at an acceptable level. In comparison to the previous work [8], the IEEE 802.15.4 MTU allows for a less aggressive TX fragmentation than BIIT in LoRa networks, while there is no need to fragment the signature in the IEEE 802.15.4 network.

It is worth noting that the energy consumption upon TX submission remains at a high level due to the currently only available Ed25519 implementation on MSP430, but the deep sleep energy consumption of TelosB is much lower than the energy consumption of the AVR device family experienced in previous work allowing for long lasting deployments. The system is still open for improvements as soon as more performant software libraries become available for the MSP430 platform, since the software architectures chosen here are flexible and modular.

VI. Acknowledgements

This paper was partially supported by (a) the University of Zürich UZH, Switzerland and (b) the European Union Horizon 2020 Research and Innovation Program under grant agreement No. 830927, namely the H2020 Concordia Project.

References

[1] D. J. Bernstein, B. Van Gastel, W. Janssen, T. Lange, P. Schwabe, and S. Smetsers, "TweetNaCl: A Crypto Library in 100 Tweets," in *International Conference on Cryptology and Information Security in Latin America*, Florianpólis, Brazil, Springer, 2014, pp. 64–83.

[2] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," in *Internet Research Task Force, RFC*, vol. 7228, May 2014.

[3] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, Tampa, FL, US, 2004, pp. 455–462.

[4] A. Dunkels, F. Österlind, and Z. He, "An Adaptive Communication Architecture for Wireless Sensor Networks," in *Proceedings of the 5th ACM International Conference on Embedded Networked Sensor Systems*, Sydney, Australia, 2007, pp. 335–349.

[5] M. J. Dworkin, "SHA-3 Standard: Permutation-based Hash and Extendable-Output Functions," NIST, Gaithersburg, MD, US, Tech. Rep., 2015.

[6] S. Ferdoush and X. Li, "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications," in *The 9th International Conference on Future Networks and Communications (FNC'14)*, vol. 34, Niagara Falls, ON, CA, Elsevier, 2014, pp. 103 – 110.

[7] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Fourth IEEE International Symposium on Information Processing in Sensor Networks (IPSN) 2005*, Boise, ID, US, 2005, pp. 364–369.

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

[9] S. Rafati-Niya and B. Stiller, "BAZO: A Proof-of-Stake (PoS) based Blockchain," IFI-TecReport No. 2019.03, UZH, Zürich, Switzerland, May 2019.