BCTrust: A decentralized authentication blockchain-based mechanism

Mohamed Tahar HAMMI, Patrick BELLOT, Ahmed SERHROUCHNI
LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France
{hammi,bellot,serhrouchni}@telecom-paristech.fr

***Abstract*—*Internet of Things* becomes a major part of our lives, billions of autonomous devices are connected and communicate with each other. This revolutionary paradigm creates a new dimension that removes the boundaries between the real and the virtual worlds. The *Wireless Sensor Networks* are a masterpiece of the success of this technology, using limited capacity sensors and actuators, industrial, medical, agricultural and many other environments can be covered and managed automatically. This autonomous interacting things should authenticate each other, and communicate securely. Otherwise malicious users can cause serious damages on such systems. In this paper we propose a robust, transparent, flexible and energy efficient *blockchain*-based authentication mechanism called BCTrust, which is designed especially for devices with computational, storage and energy consumption constraints. In order to evaluate our approach, we realized a real implementation with C programming language, and *Ethereum Blockchain*.**

***Index Terms*—Blockchain, Authentication, Security, IoT, Migration, Ethereum, WSN, Industrial Environment.**

I. Introduction

Internet of Things (IoT) is an emerging technology where devices are smart (have computing and memorization capacity), autonomous, and exchange information between each other without human involvement. *Wireless Sensor Network (WSN)* is the part of the IoT concerning limited capacity devices (nodes), used to gather data and manage various environments. Usually, a WSN is composed of groups of nodes interacting with each other. Each group is managed by a Personal Area Network Coordinator (CPAN) which is an unlimited capacity node. If no security is required, mobile nodes can migrate from one cluster to another transparently, and the association of new nodes can be easily realized. However, in a secure mode, the authentication of new nodes who do not belong to the same cluster is a complicated task, because the authentication of such nodes require a decentralized authentication authority. In other words, if we represent the mobile node by a human person, the CPAN by a government, and the nodes cluster by a country, it is almost impossible to authenticate a person in a foreign country, if no worldwide authentication system is deployed. Therefore, we need to create a decentralized and flexible mechanism allowing secure and transparent migration of nodes.

In 2008, a person or a group of persons under the name of *Satoshi Nakamoto* proposed a peer to peer electronic cash system called *Bitcoin*[1]. This system allows a decentralized electronic payment, that provides direct and secure transactions between two communicating entities without relying on trust, but rather on electronic proof. The operation of *Bitcoin* is based on a consensus protocol used in order to maintain a distributed ledger known as the *blockchain* [2]. The data stored in these ledger pages are protected from deletion, tampering, and revision [3]. Professors from Harvard define the *blockchain* as "*an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way. The ledger itself can also be programmed to trigger transactions automatically*"[3].

Although the *blockchain* technology was created for *Bitcoin*, it can be used regardless of the *Bitcoin* ecosystem. We can define the *blockchain* as a transactional database duplicated and distributed over several nodes (computers, data-centers, etc) assembled in a peer-to-peer network. The database represents a ledger of information, called transactions or facts, which can be anything such as monetary transactions, health data, power plants data, etc. The *blockchain* is a chain of blocks, each one is made of a body and a header. The body is a set of facts, which can be stored in plaintext or ciphertext. Whereas the header contains information about the block, such as the *merkle root* [4], which is a hash of the block transactions, the timestamp, the block number, and the hash of the previous block. As a result, the system creates a chain of linked and ordered blocks, and the longer the chain, the harder is to falsify it. Depending on the type and the nature of the *blockchain*, a participating network node can read facts, or read and write facts. In order to add a new fact to the *blockchain*, a network consensus procedure should be established for determining where this fact should appear in the ledger. *Blockchains* can be permissionned (private) such as *Hyperledger Fabric* [5], or permissionless (public) like *Bitcoin* and *Ethereum* [6]. The first category makes restrictions for the consensus contributors (only the chosen trustful actors have the rights to validate transactions). It does not require lot of computation to reach a consensus, thus it is not time neither energy consuming. The second one uses an unlimited number of anonymous nodes, that communicate securely using pair of private/public keys. A part of these nodes is on competition to validate and add new blocks to the *blockchain* in order to get a reward [1] [6]. This operation is called *mining*, and these nodes are called *miners*. The *blockchain* is safe and the network consensus is reached, while 51% of the nodes are honest.

In addition to managing the electronic cash system, *Ethereum* represents a platform for decentralized applications, and process programs called *smart contracts*. *Smart contracts*

978-1-5386-1734-2/18/$31.00 ©2018 IEEE

are executed by participating nodes using an operating system known as *Ethereum Virtual Machine* [6]. *Ethereum* block size is shorter than in *Bitcoin*, and the validation time takes only 14 seconds compared to 10 minutes in *Bitcoin*.

In order to strengthen the *Bitcoin* and *Ethereum* systems, and make them more resistant to modification attacks, *miners* have to calculate the called *proof of work* (PoW) [4] [7] and add it to the block during its validation. The *proof of work* is a data processing challenge, that is difficult (costly and time-consuming) to produce but easy for others to verify. Thus, it complicates the task for attackers wanting to falsify previous blocks, against honest nodes who only try to add new onces.

Usually permissionless *blockchain* are energy and time consuming, because it includes a computation amount to strengthen security of the system. Note that *Ethereum* can also be used as a private *blockchain*, thus the participating nodes are chosen and the PoW mechanism is no longer required.

Benefiting from the power of *blockchains*, and in order to create a secure, decentralized, and transparent nodes migration system, we propose in this paper a new *blockchain*-based authentication mechanism called *BCTrust*, based on the principle of "*The friend of my friend is my friend*". Following, in Section II we present some related works realized and proposed to secure the IoT. In Section III we explain in details the operation mode of our mechanism and show its advantages and utility. The real implementation and the evaluation results of our solution are drawn in Section IV. Finally, in Section V, we conclude and we discuss our future works.

II. Related work

For devices authentication and data integrity in the IoT, *Datagram Transport Layer Security* (DTLS) based approaches were proposed in [8] and [9]. They use digital certificates provided and managed by a public key infrastructure. These solutions ensure a strong authentication and secure data exchange. However, asymmetric algorithms such as *Rivest Shamir Adleman* (RSA) [10] take an important execution time, and consumes lot of energy. This problem of energy consumption was treated in [11] by combining DTLS with *Constrained Application Protocol* (CoAP) [12] and the *Elliptic Curves Cryptography* (ECC) [13]. However the need of the root certificate authority reduces the flexibility of the system.

In previous work [14], we proposed a *Pre-Shared Key* (PSK) based approach, that uses a mechanism build on the *One Time Password* (OTP) algorithm. In order to ensure a mutual authentication between nodes, the protocol requires the exchange of four messages (*association request*, *authentication request*, *authentication response*, and the *association response*). Furthermore we used the *Advanced Encryption Standard* (AES) with *Galois/Counter Mode* (GCM) [15] and/or with *Counter with CBC-MAC* (CCM) modes of operation to ensure both of the integrity and the confidentiality of data.

A similar work was realized in [16], which uses the *Message Integrity Code* (MIC) authentication mechanism instead of OTP. It requires also the exchange of four messages and uses the AES-CCM for the protection of the exchanged data.

These approaches have proved their robustness, lightness, and energy consumption efficiency. However the assignment of pre-shared keys limits the flexibility of the nodes mobility, which want to move from a cluster of nodes to another.

Most of the classical authentication and security mechanisms are centralized and sometimes require a third trustful party. By giving a total decentralization, a high security of data, and a global view of a system, the *Blockchain* is the golden solution [17][18]. The first approach that comes to mind, is to register on the *blockchain* the identifier and the appropriate keypair of each device, and then, the device is controlled by the asymmetric cryptography. This solution is robust and safe, however, the asymmetric cryptography is not desired for the limited capacity sensors.

Researchers in [19] propose an interesting protocol called *Certcoin*, which is a decentralized public key infrastructure (PKI), based on *Bitcoin*. This decentralized PKI creates, publishes, manages, updates, stores, and revokes keys corresponding to given identities. Based on the strength provided by the mining mechanism of *Bitcoin*, it ensures a good way to authenticate devices. This protocol solves the problem of centralization of PKIs, and spare systems from the use of trusted authorities, which sometimes can be very expensive. However, authenticating constrained devices using the *Merkle Hash Tree accumulator* [19] is very costly.

Following, we will discuss our proposed approach that aims to secure the constrained devices in IoT, and resolves the issues mentioned above.

III. Proposed approach

In a previous work [14], we proposed a secure communication protocol for the IoT, more precisely for securing the Wireless Sensor Networks (WSNs). As shown in Figure 1,

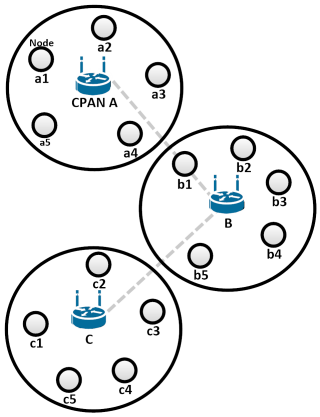

Figure 1: A general WSN topology

WSN are generally represented by a set of clusters of limited capacity nodes, each cluster is managed by an unlimited one, which is the CPAN. Our previous work allows a mutual authentication between a device and a CPAN belonging to the same cluster, using personalized pre-shared keys. It provides also a strong and fast confidentiality and integrity services for the packets exchange. The weakness of this solution resides

in its flexibility. That is to say, a node a_x belonging to a cluster A can not be associated to another cluster B without involving an administrator for installing another personalized key corresponding to the cluster B. Thus, the automatic and transparent migration of a device from a cluster to another is not allowed.

In this paper we propose a new security *blockchain*-based mechanism called *BCTrust*. This mechanism is totally different from the other approaches seen in Section II. *BCTrust* is based on the principal of "*The friend of my friend is my friend*". Which means that, if a device is authenticated in one cluster, it becomes trustful and accepted by all the other clusters. The *blockchain* ensures that the stored information is available for all the participating nodes, and protected from modifications.

Our solution is deployed on *Ethereum blockchain*, which provides one of the greatest ledgers in the world, has a big community that follows it, and allows to develop any type of application on it. It ensures secure transactions, and such as *Bitcoin*, *Ethereum* is tested in a large scale. Lastly, we can take a copy of its current ledger and use *Ethereum* freely in local. We added a layer on *Ethereum*, that transform it from a permissionless blockchain to permessionned one (using *modifiers* [20]), while keeping the power and the security of the permessionless blockchain. In the *smart contract*, only a set of trustful nodes have the writing rights on the *blockchain*. These privileged devices are the CPANs of the network. Each CPAN has a pair of private/public keys, allowing it to securely make transactions with the *blockchain*.

Figure 2 explains the operating principle of *BCTrust*. We

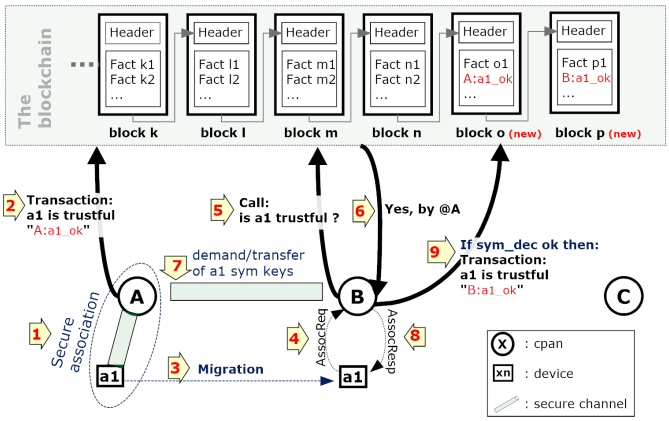

Figure 2: BCTrust security mechanism

have a device $a1$ belonging to the cluster managed by the CPAN A. (1) The first secure association of $a1$ with A is established in a secure procedure (detailed in [14]). (2) Once the device is authenticated, A sends a *transaction* to the *blockchain* "A:a1_ok", which means that A is the guarantor that $a1$ is trustful, and A has the shared symmetric keys for the secure data exchange with $a1$. This *transaction* is stored into a new block validated by the participating CPANs. (3) When $a1$ migrates to the cluster that is managed by B, (4) it sends an association request to B. (5) B sends a *call* to the *blockchain* in order to check if $a1$ was authenticated before or not. Unlike the *transaction* operation, the *call* does not modify the state of the *blockchain*, it is used only for reading stored data. (6) When B is informed by the *blockchain* that $a1$ is trustful and was authenticated by A, (7) it requests A for $a1$'s symmetric key, using an asymmetric secure channel. (8) B associates the device, then (9) sends the *transaction* "B:a1_ok" after a successful encryption/decryption operations with $a1$. If $a1$ wants again to migrate to the cluster of C, the latter ask (*call*) the *blockchain*, and the result will be ["A:a1_ok", "B:a1_ok"], in this case C choses the closest neighbor for getting $a1$'s symmetric keys.

As mentioned in [14], the symmetric secure communication between a device and a CPAN uses, in addition to the key, an initialization vector (IV). The IV changes after each communication session using a counter value, in order to avoid cryptanalysis and replay attacks.

The migration mechanism is summarized in Algorithm 1, Algorithm 2, and Algorithm 3.

Algorithm 1: Parameters and functions definition

parameter:
- X: Object // X is the CPAN
- d: Object // d is the device
- sk: Key // the symmetric key
- msg: Message
- $authList$: AddressList
- $authenticator$: Address
- $channel$: Channel
- BC: Blockchain
- const ok: State
- const $auth\_req$ String

Function : ClosestNeighbor(AddressList $list$)
// returns the closest neighbor
Function : FirstAssociation(Address $deviceAddr$)
// otp based authentication
// creation of sk
Function : SecChannel(Object $obj1$, Object $obj2$)
// secure channel establishment between $obj1$ and $obj2$
Function : SymDec(Message msg, Key k)
// symmetric decryption of msg using k

IV. Evaluation and Numerical Results

A. Experimental framework

Our approach is *blockchain*-based, therefore it is reinforced by all the advantages that the *blockchain* provides:

1) Denial of service and distributed denial of service (dos/ddos) represent one of the most deployed and dangerous network attacks. These attacks aims to stop the availability of a service. Due to the decentralized and duplicated nature of the *blockchain*, it is very hard, if not impossible, to target this type of architecture with a dos/ddos attack.

Algorithm 2: Migration mechanism CPAN side

```
begin
    X ← this
    msg ← receive () // assoc req
    authList = BC.SendCall (msg.address)
    // returns the list of the
       device's authenticators
    authenticator = ClosestNeighbor (authList)
    if (authenticator = NULL) then
        FirstAssociation (msg.address)
    else if (authenticator = X.address) then
        // successful association
    else
        channel = SecChannel (X, authenticator)
        sk = channel.getKey (msg.address)
        // successful association
    wait() // wait for receiving messages
    msg ← receive () // encrypted message
    if (SymDec (msg, sk) = ok) then
        BC.SendTransaction (msg.address, ok)
```

Algorithm 3: Migration mechanism device side

```
begin
    d ← this
    send (X.address, msg)
    msg ← receive ()
    if (msg.text = auth_req) then
        FirstAssociation (d)
    else if (SymDec (msg, sk) = ok) then
        // successful association
    else
        // failed association
```

2) Using the *Elliptic Curve Digital Signature Algorithm* (ECDSA) ensures fast and robust transactions [21].
3) Thanks to the consensus mechanism, stored data can not be falsified and an established task can not be repudiated.
4) Our approach provides a global view of all the communicating things, therefore the migration operation is realized safely, transparently, with hight flexibility, and without involving human administrators.
5) Adding a counter to the IV value (see [14]), protects the secret symmetric key and the exchanged messages (between a device and a CPAN) from the cryptanalysis and the replay attacks.

In order to evaluate our approach, we used a real implementation on a network composed of two CPANs A and B and one device that has the following characteristics: Dresden Elektronik deRFsam3 23M10-R3, have 48 kB of RAM, 256 kB of ROM and a Cortex-M3 Processor. We are aware that a real IoT network contains a bigger number of nodes. However, the goal of our study is to give a proof-of-concept of our approach. More precisely, we aim to prove (1) the robustness of our ecosystem and its flexibility regarding nodes migration and (2) the evaluation of such an approach on constrained devices. Thus, the usage of such a minimal architecture is more than sufficient.

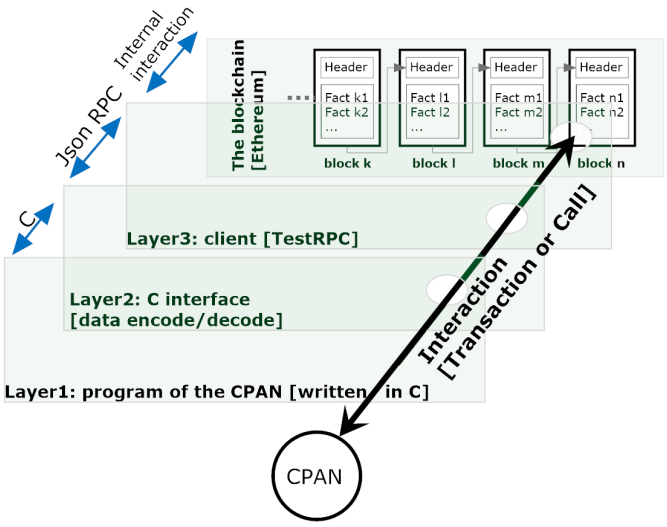

Figure 3: BCTrust layers

We deployed an *Ethereum blockchain* copy, and *TestRPC* [22] as client for interacting with the *blockchain*. Our code is implemented with the C language. For interacting with the *blockchain* through *TestRPC*, we developed an interface that encode/decode data in order to be interpreted per *Ethereum*. It uses *JSON RPC* for the communication, where *JSON* is a standard textual data format, and *RPC* a remote procedure call system. This interface allows programs written in C to communicate with *TestRPC* in remote procedure call mode, thus interact with the *blockchain*. After preparing the necessary tools and programs, we created our *smart contract* in *Solidity* language [20]. The Figure 3 shows the different layers of the architecture. When we start the system, the contract is compiled and stored in the *blockchain*, providing a special address called the *contract address*. The latter is used to locate the *smart contract* in the *blockchain*, in order to interact with it. *TestRPC* provides ten user accounts, therefore ten key pairs (private and public keys). Our *blockchain* starts with ten blocks, each one is numbered and timestamped. To evaluate the performances of the migration mechanism, we measured the execution time and calculated the power consumption of: (1) our classical approach [14] (1^{st} association) explained briefly in Section II, let's call it *classical association*, and (2) the following associations during the node migration, let's call them *BCTrust associations*.

B. Numerical results

We have already compared our *classical association* authentication mechanism with other methods in [14], and we have proven that our approach realizes good performances.

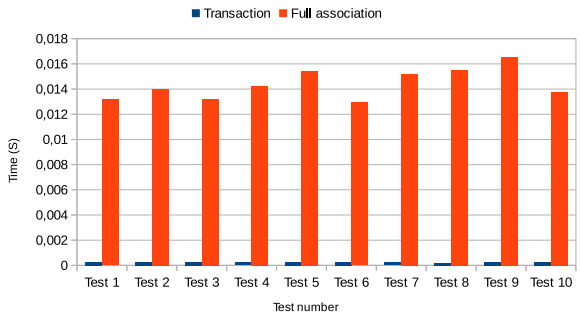

Figure 4: BCTrust assoc times

For *BCTrust*, first, we realized ten migration tests of the device $a1$ from the cluster of CPAN A to the cluster of B. Figure 4 shows the results of the full associations including transactions times. One can note that the BCTrust full association operation is very fast, especially because transaction validation time using a local copy of *Ethereum blockchain* takes on average 0,000250, compared to the real *Ethereum blockchain* which takes 14 seconds to validate a transaction. After that, for proving that our system is energy-efficient, we compute the power consumed by the device $a1$ during its migration to the cluster of B by calculating the power consumed by the communication P_C, and the power consumed by the processing P_P. These results allow us to have an idea about the amount of power required, and how much we economize energy by using BCTrust association compared to the classical association.

The power consumption for the communication can be calculated using the equation below (Equation 1):

$$\boxed{\begin{aligned} P_C = N_T[P_{te}(T_{on} + T_{wu}) + P_o(T_{on})] + \\ N_R[P_{re}(R_{on} + R_{wu})] + P_{idle} \times T_{idle} \end{aligned}} \quad (1)$$

Where:

- P_{te} is power consumed by transmitter;
- P_{re} is power consumed by receiver;
- P_o is output power of transmitter;
- T_{on} is transmitter "on" time;
- R_{on} is receiver "on" time;
- T_{wu} is start-up time for transmitter;
- R_{wu} is start-up time for receiver;
- N_T is the number of times transmitter is switched "on";
- N_R is the number of times receiver is switched "on";
- P_{idle} is the is power consumed during the idle mode;
- T_{idle} is the duration of the idle mode.

Based on the micro-controller data-sheet, the intensities I of P_{te} and P_{re} are respectively equal to 14 mA and 12 mA. The used vlotage U is 3 V. P_o is 1 mW. P_{idle} is 0,0012 mW. Both of T_{wu} and R_{wu} are equal to 0,032 mS. Knowing that :

$$\boxed{P = UI} \quad (2)$$

Then $P_{te} = 42\ mW$; and $P_{re} = 36\ mW$.

In order to compute T_{on} and R_{on}, we took the maximum size of packet per transmission/reception, that is equal to 136 $Bytes$. Using the Equation 3 and knowing that the channel bandwidth is equal to 250 $kBit/S$, we have:

$$\boxed{T_{on} = R_{on} = \frac{Exchanged\ bytes}{Channel\ band\ width}} \quad (3)$$

$T_{on} = R_{on} = \frac{136\ Bytes}{250\ kBits/S} = \frac{1088\ Bits}{250\ Bits/mS} = 4,352\ mS.$

In our demonstration, the communication between $a1$ and A is based on the standard *IEEE 802.15.4*, and between A and B (between CPANs) is based on the standards *TCP/IP*. The T_{idle} is calculated using the Equation 4:

$$\boxed{\begin{aligned} T_{idle} = Full\ association\ time - [N_T \\ (T_{wu} + T_{on}) + N_R(R_{wu} + R_{on})] \end{aligned}} \quad (4)$$

P_C in classical association:

The average time of the full association is equal to 34,450 mS. We exchange 4 messages (N_T = 2, and N_R = 2). Using Equation 4:
T_{idle} = 16,914 mS.
Therefore, using Equation 1:
p_C = 692,628 mW.

P_C in BCTrust association:

With the same parameters, we compute the power consumption for the communication in *BCTrust* association between $a1$ and $CPAN_B$. The average of the full association time is 14,406 mS. $N_T = 1$ and $N_R = 1$.
$T_{idle}\prime = 5,638\ mS$.
$P_C\prime$ = 346,311 mW.

One can note that the power consumed by the device in *BCTrust* association represents 49.9% of what is consumed during the classical association, thus we economize almost the half of the devices association communication power.

In order to evaluate the total consumed power during the secure association, we need also to calculate the process power consumption (P_P). We used the power consumption model presented in [23]. P_P is processed by the Equation 5.

$$\boxed{P_P \approx N \times C_{load} \times U^2 \times f} \quad (5)$$

Where N is the number of clock cycles per task, C_{load} is the load capacitance, U is the supply voltage, and finally, f is the clock frequency.

In this estimation we will take into account only the consumption of the used algorithms, (1) HMAC [24] for the classical association, and (2) AES-CCM for the BCTrust association. Based on studies realized on similar hardware, described in [25]:

$$\begin{cases} perf(HMAC) \approx 340000, \\ \quad for\ 512\ Bits\ input\ blocks. \\ perf(HMAC) \approx 275000, \\ \quad for < 512\ Bits\ input\ blocks. \\ perf(AES-CCM) \approx 9677, \\ \quad for\ 128\ Bits\ input\ blocks. \end{cases}$$

Where $perf$ is the software performance in $clock\ cycles$.

P_P in classical association:

For the 1^{st} association, the device $a1$ uses HMAC for processing an input data of 1344-Bits, thus it consumes $\approx$ 955000 $clock\ cycles$ (N is equal to $0,955 \times 10^6$). Knowing that C_{load} = 7.5 pF (7.5 $\times$ 10^{-12} F), U = 3 V, and f = 32 kHz (32000 Hz). And using Equation 5, the process power consumption of the 1^{st} association is equal to: $P_P \approx 2,0628W = 2062,8\ mW$

P_P in BCTrust association:

For the migration association, the device exchanges 2 messages, and process only the body part (123 $Bytes$), thus each message is processed by 8 iterations. The total $clock\ cycles$ number of the migration association is equal to 2 $\times$ 9677 $\times$ 8, so N = 0,154832 $\times 10^6$.
Using the same parameters, we get:
$P_{P\prime} \approx 334,437\ mW$
One can note that the power consumed by the device in the migration association is very small compared to what is consumed during the 1^{st} association, thus we economize $\approx$ 83,79% of the devices association process power.

The estimated full power consumption of the classical association is equal to 2755,428 (mW), whereas the *BCTrust* association 680,748 (mW). The energy consumed during the full association time is calculated using Equation 6:

$$\boxed{P = \frac{E}{t},\ thus\ E = P \times t} \tag{6}$$

Where P is the power (in mW), E is the energy (in mJ), and t is the time (in S).
$E_{classical} \approx$ 2755,428 $\times$ 0,034 $\approx$ 93,684 mJ
$E_{BCTrust} \approx$ 680,748 $\times$ 0,014 $\approx$ 9,530 mJ

According to these results, using *BCTrust* association make us save more than 89,82 % of energy. The table I summaries and compares the results obtained above.

Association	**Classical**	**BCTrust**
Execution time	0,034 (S)	0,014 (S)
P_C	692,628 (mW)	346,311 (mW)
P_P	2062,8 (mW)	334,437 (mW)
Total power consumption	2755,428 (mW)	680,748 (mW)
Total energy consumption	93,684 (mJ)	9,530 (mJ)

Table I: Evaluation results

V. Conclusion and future works

In this paper we proposed a flexible and energy efficient security mechanism for the Internet of Things, which holds its power from the *blockchain* system. Apart from the 1^{st} association of a device, it reduces its secure association from four exchanged messages (in previous work [14]) to two messages, and does not require key derivation processing. The major processing operations and messages exchange are realized between a CPAN and the *blockchain*, or between a CPAN and another one, which are unlimited capacity devices, that does not pose any problem of energy consuming, nor storage neither processing capacity. *BCTrust* provides a global view of the network, and a decentralized authentication system.

In our futur work, we will study the scalability of our approach and the impact of network size on its performances.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
[2] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. pages 45–59. 13th USENIX Symposium on Networked Systems Design and Implementation, 2016.
[3] Marco Lakhani Karim R Iansiti. The Truth About Blockchain. Harvard University, Harvard Business Review, January 2017.
[4] Bitcoin.org is community. Bitcoin Developper Guide. *Online. https://bitcoin.org/en/developer-guide#block-chain-overview*, 2016.
[5] Marko Vukolić. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 3–7. ACM, 2017.
[6] Ethereum community. Ethereum Homestead Documentation. *Online. http://www.ethdocs.org/en/latest/index.html*, 2016.
[7] Ethereum community. Ethash. *Etherium, wiki. https://github.com/ethereum/wiki/wiki/Ethash*, April 2017.
[8] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. {DTLS} based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8):2710–2723, 2013.
[9] M. Panwar and A. Kumar. Security for IoT: An effective DTLS with public certificates. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 163–166, March 2015.
[10] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.
[11] A. Capossele, V. Cervo, G. De Cicco, and C. Petrioli. Security as a CoAP resource: An optimized DTLS implementation for the IoT. In *2015 IEEE International Conference on Communications (ICC)*, pages 549–554, June 2015.
[12] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (CoAP). Internet Engineering Task Force (IETF), Request for Comments: 7252, 2014.
[13] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.
[14] Mohamed Tahar HAMMI, Erwan LIVOLANT, Patrick BELLOT, Ahmed SERHROUCHNI, Pascale MINET. A lightweight IoT security protocol. page 7. Cyber Security in Networking Conference (CSNet2017), 2017.
[15] Morris J. Dworkin. SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States, 2007.
[16] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *ZigBee Specification*. ZigBee Alliance, September 2012.
[17] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4, 2016.
[18] Hitesh Malviya. How blockchain will defend iot. 2016.
[19] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. CertCoin: A NameCoin Based Decentralized Authentication System. 2014.
[20] Ethereum foundation. Solidity documentation. *Online manual. https://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html*, 2017.
[21] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63, 2001.
[22] Fast Ethereum RPC client for testing and development . *Online. Test RPC. https://github.com/ethereumjs/testrpc* , 2015.
[23] D. Lozneanu, A. Costache, and M. Romanca. Applications of energy model in WSN nodes. In *2014 International Conference on Optimization of Electrical and Electronic Equipment*, pages 852–857, May 2014.
[24] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. RFC 2104: HMAC: Keyed-hashing for message authentication. 1997.
[25] Omar Alfandi, Arne Bochem, Ansgar Kellner, Christian Göge, and Dieter Hogrefe. Secure and authenticated data communication in wireless sensor networks. *Sensors*, 2015.