A Blockchain-based Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things

Mohamed Ali Kandi, Djamel Eddine Kouicem, Hicham Lakhlef, Abdelmadjid Bouabdallah, Yacine Challal

A Blockchain-based Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things

Mohamed Ali Kandi[1], Djamel Eddine Kouicem[1], Hicham Lakhlef[1], Abdelmadjid Bouabdallah[1] and Yacine Challal[2]
[1]Sorbonne Université, Université de Technologie de Compiègne, CNRS, UMR7253 Heudiasyc-CS 60319-60203 Compiègne Cedex, France
[2]Laboratoire de Méthodes de Conception de Systèmes, École nationale Supérieure d'Informatique, Algiers, Algeria
Email: $\{mohamed-ali.kandi, djamel-eddine.kouicem, hicham.lakhlef, madjid.bouabdallah\}$ @$hds.utc.fr, y\_challal$@$esi.dz$

***Abstract*—The Internet of Things (IoT) is an emerging technology that aims to extend connectivity to all everyday devices. One of the main challenges that are slowing down its development is how to secure the Device-to-Device communication. Among all the security issues, the Key Management (KM) is one of the most challenging. The difficulty lies in the fact that most of the IoT devices suffer from a lack of resources. Although different protocols were proposed, most of them do not consider the dynamic nature of the IoT. Other solutions rely on a centralized entity to distribute the new keys upon a change in the network. However, this entity becomes a single point of failure and the main target of attacks. We propose a novel blockchain-based decentralized KM protocol. In addition to being resilient, scalable and dynamic, our solution uses the blockchain technology to securely distribute the KM on several entities.**

***Index Terms*—Internet of Things, Device-to-Device communication, Security, Key Management, Blockchain.**

I. Introduction

The Internet of Things (IoT) consists of extending connectivity beyond standard devices (such as computers, tablets and smartphones) to all everyday objects. These objects can then automatically communicate in a peer-to-peer manner. This increases their functionalities and allows them to offer new services for the benefit society, which until then were not able to provide. The IoT is an emerging technology that has the potential to improve our daily lives in a number of ways. Smart homes, for example, involve using smart devices to ensure comfort, convenience and energy efficiency to the homeowners. Autonomous vehicles are able to automatically exchange data to maintain traffic flow, avoid crashes and improve the environment.

Although some of the IoT applications are currently available, many challenges are slowing down their development. Securing Device-to-Device communication is one of the main problems facing the IoT [7]. This is because most of its devices suffer from a lack of resources in terms of storage, computation, communication and energy. Among all the security issues, the Key Management (KM) is one of the most challenging. The KM is the core of secure communication. Its main role is to provide the network members with secret cryptographic keys that are used to encrypt and decrypt the exchanged data. Although different KM protocols were proposed to secure Device-to-Device communication, each of them presents its own limitations.

The existing solutions rarely consider the dynamic nature of the IoT. Based on key pre-distribution, they store the keys in the nodes' memory before their deployment. It is therefore difficult to add new nodes to the network afterwards. To be able to update the keys upon a change in the network, a centralized entity is required. However, it becomes a single point of failure and the main target of attacks. If the central entity fails, the entire system will stop operating and if it is attacked, the whole network will be compromised. To address these issues, we propose a novel KM protocol. In addition to being resilient, scalable and flexible, our solution is decentralized using the blockchain technology and smart contracts. A blockchain is a decentralized and secure storage technology. It first appeared in Nakamoto's Bitcoin paper describing a new decentralized cryptocurrency [14]. This technology is used today in various applications, including the KM.

Thanks to the blockchain features (decentralization, immutability and traceability [13]), we show that our solution allows to spread the cryptographic material across several entities. The aim is to avoid a single point of failure and to make it more difficult to access or modify this secret material. Moreover, a captured entity cannot modify the cryptographic keys without the consent of others. The blockchain also makes it possible to trace the actions of a compromised entity. Finally, being implemented on the IoT gateways, the blockchain management does not involve any additional cost on nodes, except those imposed by the KM.

The remainder of this paper is organized as follows: in Section II, we discuss the related works. In Section III, we detail our solution. In Section IV, we present the security analysis. In Section V, we evaluate the performance of our solution. In Section VI, we conclude our work.

II. Related Works

Although different KM protocols were proposed to secure Device-to-Device communication, each of them presents its own weaknesses. According to the encryption technique used, the KM approaches can be classified into two categories: symmetric and asymmetric. Symmetric schemes involve the use of the same key for encryption and decryption, while asymmetric approaches use two different keys. Asymmetric protocols usually imply intensive computing, which makes them impractical on most of the IoT constrained devices [20]. For this reason, we focus in this work on symmetric schemes.

A. Key Management schemes

Most of the symmetric KM systems proposed to secure Device-to-Device communication are based on pre-distribution. The keys are stored in the nodes' memory before their deployment. These protocols can in turn be classified into two categories. Deterministic schemes [1, 6, 17] establish a direct secure link between each pair of nodes. These approaches guarantee a total connectivity coverage at the expense of storage. They are therefore not scalable. Probabilistic schemes [2, 4, 15, 21, 22] store fewer keys on nodes, but do not guarantee a secure connectivity between each two neighboring communicators. Intermediate nodes may be necessary to establish secure links between them. This lack of connectivity involves additional calculation and communication and thereby more energy consumption [21]. Pre-distributed schemes are motivated by the fact that they do not require a third party to assign keys to nodes. However, it is difficult to add new nodes to the network afterwards. These protocols are more suitable for static networks, whose members do not change frequently.

In previous works [9, 10], we proposed a KM solution for dynamic network such as the IoT. It allows nodes to securely join and leave the network at any time. Keys are then automatically distributed on the other members. Also, unlike the above-mentioned protocols, our previous solution provides a good compromise between scalability and connectivity. To achieve this balance, the network members are distributed into logical sets. A device shares then a distinct pairwise key with each member of its set and a unique pairwise set key with the members of each of the other sets. The drawback of this solution is that it relies on a centralized entity and is subject to the problem of single point of failure. We then propose a novel decentralized KM protocol based on the blockchain.

B. Blockchain solutions

A blockchain is a decentralized and secure storage technology. Its name derives from the fact that it is composed of blocks of transactions, each storing a cryptographic hash of the previous one. A blockchain relies on cryptography, smart contracts and consensus algorithms to securely replicate an application on several entities. Consensus algorithms (such as Proof of Work -PoW- [14], Proof of Stake -PoS- [18] and Practical Byzantine Fault Tolerance Algorithm -PBFT- [3]) guarantee that each entity records the same transactions in the same order. Smart contracts are functions that are defined beforehand and stored in the blockchain. They are automatically run by its participants when they receive transactions.

The blockchain first appeared in Nakamoto's Bitcoin paper describing a new decentralized cryptocurrency [14]. Recently, researchers began to use the blockchain to decentralize the KM. The authors of [11, 12] proposed a blockchain-based KM to secure group communication in intelligent transportation systems. In [13], a blockchain was used to distribute the KM for Hierarchical Access Control in the IoT. These works do not consider Device-to-Device communication and use the PoW consensus algorithm. Our solution is based on PBFT, which is more efficient and less energy intensive [19].

III. Our solution

Our solution is organized into two layers (Figure 1). The node layer uniformly distributes the network members into logical sets and provides them with secret cryptographic keys. A node shares a distinct pairwise key with each member of its set and a pairwise set key with the members of each of the other sets. The blockchain layer manages the blockchain and its participants. The aim is to securely decentralize the KM. It guarantees that the system continues to operate even if some of its participants fail or are the target of malicious attacks. It also ensures that the compromise of a participant does not jeopardize the security of the entire network. The blockchain is implemented on powerful IoT gateways and is separated from the constrained devices. The aim is to not involve any additional cost on them, except those imposed by layer 1. Due to space constraints, we briefly present layer 1. Fore more details, please refer to our previous work [9].

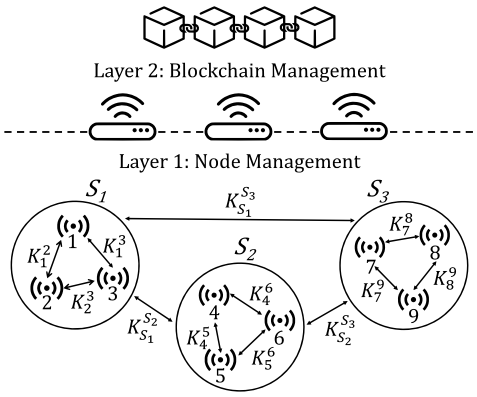


Fig. 1: Architecture of our solution.

A. Layer 1: Node Management

Layer 1 manages two types of keys: Data Encryption Keys ($DEKs$) and Key Encryption Keys ($KEKs$). The $DEKs$ are symmetric pairwise keys that are used by nodes to encrypt the data exchanged between them. A node holds two types of $DEKs$: a pairwise node key (for each member of its set) and a pairwise set Key (for each set of the network). The $KEKs$ are used to secure the communications between the KM and the nodes to protect the $DEKs$. A node stores two $KEKs$: a node and a set keys. The storage cost on a node is proportional to the sum of its set's size and the number of sets in the network.

1) Set Management: the set management consists of distributing nodes on sets while minimizing the number of keys they store. To have the same number of keys stored on each member, we opted for a uniform distribution (i.e. the n nodes of the network are distributed into $\sqrt{n}$ sets of $\sqrt{n}$ members each). For this purpose, two algorithms are used: Assignment and Reorder. The Assignment Algorithm is run when nodes join the network and assigns them to the right sets. This algorithm takes as input the size of the network and outputs the ID of the selected set. The Reorder Algorithm is run, after a node leaving, to keep the distribution of nodes uniform. The algorithm takes as input the size of the network and tries to merge or remove sets when it is possible.

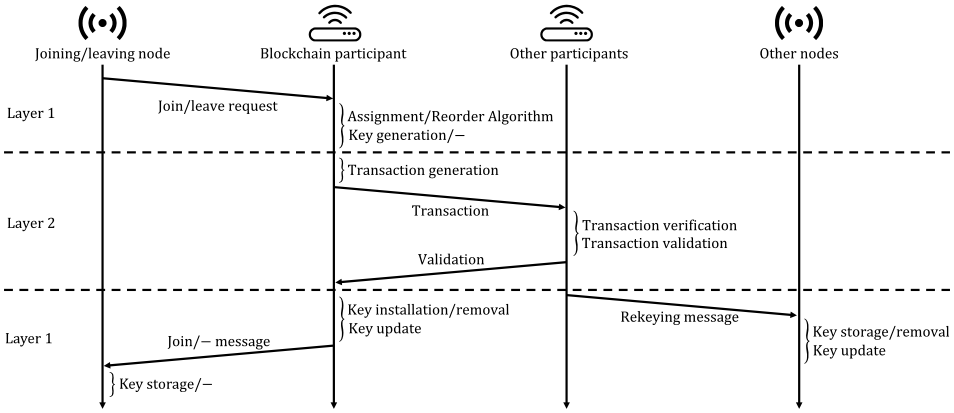


Fig. 2: Decentralized rekeying upon a network change using a blockchain.

2) Rekeying upon a network change: When a node joins the network, the KM runs the Assignment Algorithm to select a set and return its ID. The KM generates then a node ID, a secret code, a node key and a pairwise node key for each member of its set. Next, the KM randomly generates a refresh key, which is used with a key derivation function to update the selected set key and the pairwise set keys known by its members. Finally, the KM distributes these new keys on the appropriate nodes after ciphering them using the $KEKs$. When a node receives a rekeying message, it decrypts it using its node or set key. It then stores the new keys and updates some of those it knows. Similarly, if a node leaves the network or is evicted because it is compromised, the KM runs the Reorder Algorithm if a set removal or merging is possible. Next, the KM removes the node's key and all the pairwise keys associated to it. The same steps as for node joining are followed to update the keys known by the leaving node.

B. Layer 2: Blockchain Management

The aim of this layer is to decentralize the KM using a private blockchain. Unlike public blockchain, such as Bitcoin, only authorized participants are allowed to access or modify the content of a private one. This limited number of participants usually makes the blockchain management more efficient. The network therefore contains selected IoT gateways (BPs for Blockchain Participants) that generate, validate and store transactions upon a network change. A blockchain transaction is the storage unit that corresponds to a specific event, which is a rekeying operation in our case. As shown in Figure 3, a transaction is composed of: the rekeying operation (join, leave or evict), the node ID, the ID of the set of the node, the cryptographic hash of the node's secret code and the refresh key used to update the keys.

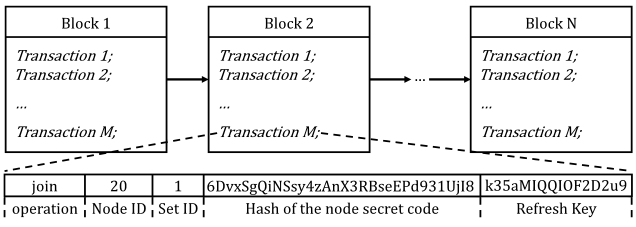


Fig. 3: Example of a blockchain transaction.

1) Transaction management upon a network change: When a BP receives a join or leave request, it first uses the layer 1 for the set and node management. Before distributing the keys, the layer 1 calls the layer 2 to generate, validate and store a transaction in the blockchain. The goal is that all the BPs are aware and agree to perform the rekeying operation. To achieve this, smart contracts are used. They are previously defined functions (e.g. implementation of the Assignment Algorithm) that are stored in the blockchain. They can be automatically run by all BPs upon the reception of a transaction. If the transaction corresponding to the current rekeying operation is correctly stored in the blockchain, the layer 2 informs the layer 1. The latter distributes then the keys on the appropriate nodes after ciphering them using the $KEKs$ (Figure 2).

Transaction generation: When the layer 2 receives the information from layer 1 about a rekeying operation, it starts by generating the corresponding transaction. The layer 2 then stores the transaction in a temporary memory (while waiting for it to be validated) and broadcasts it to all other PBs.

Transaction verification: When a BP receives a transaction, it becomes a validator. It runs smart contracts to verify the correctness of the received transaction. In the case of a node joining, the validator reruns the Assignment Algorithm to confirm that the node was assigned to the right set. It also checks if the node ID and the hash of the secret code have not already been used for another node. If a node leaves the network or is evicted, the validator checks if it is actually a member. It also verifies if there is a match between the node ID, the set ID and the cryptographic hash of the node's secret code. If the validator judges that the transaction is correct, it adds the transaction to its temporary memory.

Transaction validation: Periodically or when the temporary memory is complete, the BPs run a consensus algorithm (e.g. PoW, PoS or in our case PBFT). The aim is to achieve a consensus between them on whether the block of transactions, contained in their temporary memories, can be included to the blockchain or not. Once the block containing a given transaction is correctly added to the blockchain, this transaction is considered as valid and the layer 1 is informed. The latter can then distribute the generated keys on the appropriate nodes.

2) Blockchain participants management: The BPs act as gateways between the nodes and the blockchain. The aim is to not involve any additional cost on nodes, except those imposed by the layer 1. When a node wishes to join the network, it sends a request to a BP. If the transaction corresponding to this request is validated by the other BPs and is correctly added to the blockchain, the node is attached to the gateway that initiates the joining process. It will remain attached to it until the node moves, leaves the network or when the BP fails or gets compromised. Meanwhile, the BP manages (generates, stores and updates) the keys associated to the node and provides it with the elements allowing it to update its keys.

3) System availability: When a BP fails or when it is a target of malicious attacks, the nodes attached to it become orphans. Each of them sends then a rejoin request to an other BP. When a BP receives a rejoin request, it agrees with the sender on new $KEKs$ so they can securely communicate. Next, the node sends the hash of its secret code to be able to get authenticated. The BP consults the blockchain and checks if the hash received corresponds to that of the node. As the node in question is the only one able to generate the hash of its secret code, the BP concludes that it is really a network member. If it is the case, the node is then attached to this gateway without having to add new transactions to the blockchain. Avoiding the blockchain management makes the rejoin operation much more efficient. More importantly, the failure of a BP does not prevent the system from working.

4) Node mobility and sleeping: As when a BP fails, a node can use its secret code to get authenticated with another BP if its actual BP is no longer in range. Furthermore, a node can sleep if it does not have a work in progress to save energy. During sleeping, the node turns off its radio and will not receive the rekeying messages. Note that these messages contain the refresh keys that allow the network members to update their keys. Thus, the sleeping node will not have the opportunity to update its keys. However, when it wakes up, it will need the new keys to be able to securely communicate with the other network members. It will then send to its BP a rekey request containing the last refresh key it received. Since all the refresh keys are stored in the blockchain, the BP can retrieve and send to the node the refresh keys it missed. It will then be able to update its keys without having to add new transactions to the blockchain.

IV. Security analysis

Resilience is the measure of the impact of the capture of a member (node or BP) on the network.

A. Resilience against node capture

We start by evaluating resilience against node capture.

Lemma 1: A node can decrypt a number of links equal to:

$$D = n - 1 + (\sqrt{n} - 1)(n - \sqrt{n}) = (\sqrt{n} - 1)(n + 1) \quad (1)$$

Proof: A node can decrypt the communications linking it to the $n - 1$ other network members as well as the links between the $\sqrt{n} - 1$ members of its set and the $n - \sqrt{n}$ other nodes.

Proposition 1: The percentage of links that a compromised node can decipher is equal to:

$$P = \frac{D}{T} = \frac{2(n+1)}{(\sqrt{n}+1)n} \to 0, \text{ as } n \to \infty \quad (2)$$

Proof: From lemma 1 and the fact the the number of links in a network of n nodes is equal to $T = C_n^2 = \frac{n(n-1)}{2}$, we obtain this percentage.

Proposition 2: The capture of the whole network requires the compromise of all the network members.

Proof: Deciphering all the intra-set communications requires the knowledge of all the pairwise node keys associated to it. This is only possible if all the set members are captured. Also, deciphering all the inter-set communications requires the knowledge of all the pairwise set keys. This is only possible if at least a member of each set is compromised.

B. Resilience against BP capture

We assume that the blockchain is tamper proof, but the BPs do not trust each other as they can be compromised. The number of BPs is p and $\frac{n}{p}$ nodes are attached to each of them.

Proposition 3: The percentage of links that a compromised BP can decipher is equal to:

$$P = \frac{D}{T} = \frac{2np - n - p}{(n-1)p^2} \to \frac{2p-1}{p^2}, \text{ as } n \to \infty \quad (3)$$

Proof: A BP is responsible for the management of the keys associated to the nodes attached to it. Therefore, if it gets compromised, it will be able to decipher the $\frac{n}{2p}(\frac{n}{p} - 1)$ links between them. It will also be able to decipher the communications between its $\frac{n}{p}$ nodes and the $n - \frac{n}{p}$ other members of the network. It can then decrypt a total number of links equal to $D = \frac{n}{2p}(\frac{n}{p} - 1) + \frac{n}{p}(n - \frac{n}{p})$.

Proposition 4: The capture of the whole network requires the compromise of all the BPs.

Proof: As shown in the proof of proposition 2, deciphering all the communications requires the knowledge of all the pairwise keys. This is possible only if all the BPs are captured.

C. Comparison and discussion

In a previous work [9], we proved that our solution provides a level of resilience, for large networks, comparable to the perfect resilience offered by deterministic protocols such as [16]. However, we assumed that the central entity is secure and that only the nodes can be compromised. In the current work, we propose a decentralization based on blockchain as in practice the central entity can be captured. Thanks to the blockchain features, the KM is securely decentralized so that the compromise of a BP has no effect on the others. Thus, compared to our previous solution based on a centralized entity [9] (Number of BPs is equal to one), which once captured the whole network is compromised, only a part is compromised when PBs are captured (Figure 4). We showed that the rate of compromised links is inversely proportional to the number of BPs. Thus, the more we increase the number of BPs, the more resilient is our solution. In the following, we analyse the effect of this parameter on performance to help the reader to choose the best compromise between resilience and performance.

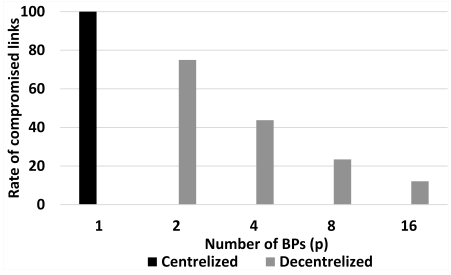


Fig. 4: Variation of the rate of captured links according to p.

V. Performance evaluation

The performance evaluation of our solution consists of three parts. First, We propose a theoretical analysis on the node side. Since the performance on the BPs depends on the blockchain used, we performed simulations to evaluate it. Finally, we compare the results obtained to some of the existing solutions.

A. Theoretical analysis

We start by analyzing the protocol overheads on nodes.

Proposition 5: Storage and calculation costs on nodes are of the order of $O(\sqrt{n})$, while the communication is $O(1)$.

Proof: Using our solution, a node knows a secret key, $\sqrt{n}-1$ pairwise node keys, a set key and $\sqrt{n}-1$ pairwise set keys. It then stores in total $2.\sqrt{n}$ keys. Moreover, regardless of the rekeying operation performed (e.g. node joining or node leaving), a node receives a constant number of messages and updates the $2.\sqrt{n}$ keys it knows.

B. Simulations

Due to space constraints, we assume that (unlike nodes) the BPs have enough storage and focus on their response time. It is the time separating the reception of a join or leave request from the sending of a response to the node. This time is equal to the sum of the execution time of layer 1 (T_k) and that of layer 2. The layer 2 consists of two operations: transaction generation and verification (T_e); transaction validation (T_a). The total response time (T_r) is therefore equal to:

$$T_r = T_k + T_e + T_a \tag{4}$$

1) Simulation assumptions: To evaluate this response time, we conducted simulations on a laptop with an Intel Core i7 CPU and 4GB RAM. Each BP is run on a different docker container. The length of a transaction is 57 Bytes (Operation -1B-, Node ID -4B-, Set ID -4B-, Hash of the node secret code -32B-, Refresh key -16B-). For greater accuracy, each result mentioned in the following is the average of dozens of tests. Our results are obtained using Tendermint [8], which is a blockchain application platform. We mainly choose Tendermint for two reasons. First, the application layer of Tendermint can be written in any programming language. Second, Tendermint is a very powerful blockchain engine based on the PBFT consensus algorithm. Using Tendermint, hashing energy is not required to validate the next block. Therefore, compared to some of the most used consensus algorithm, PBFT reduce calculations and thereby energy consumption [19].

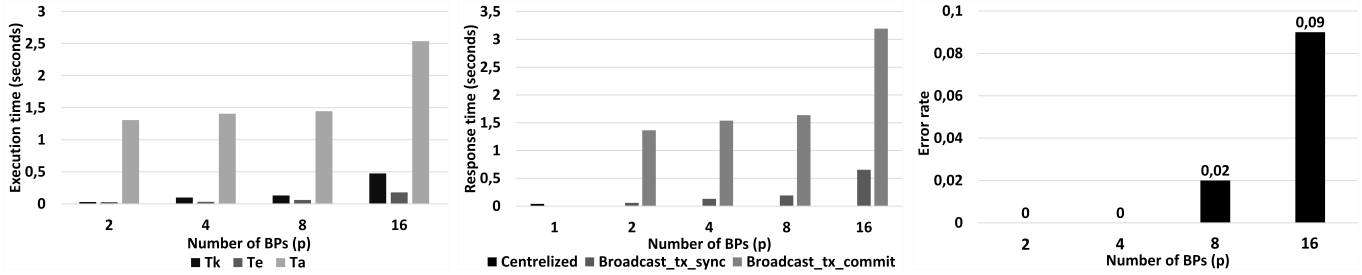
2) Results: The results of the simulations are plotted in Figure 5a. They show the variation of the execution times of the different steps mentioned above. Two points come out of this. First, the more there are BPs, the more the execution time of the different operations increases. This is because there is more exchange between the different BPs. The results also show that the most time-consuming operation is the validation of the transaction because of the consensus algorithm.

C. Comparison and discussion

In our previous work [9], we showed that our solution provides the best compromise between the IoT requirements (connectivity, efficiency, scalability and flexibility) on the node side (considering only Layer 1). Before dealing with layer 2, we briefly recall this comparison. In addition to being resilient, our solution requires less storage on the nodes ($O(\sqrt{n})$) than the deterministic scheme presented in [6] ($O(n)$). It also provides a better connectivity compared to the probabilistic schemes presented in [2, 16]. Using our solution, the probability that two neighboring nodes share a common key is always equal to 1, while it is approximately lower bounded by 0.632 in [6] and does not exceed 0.25 in [16]. Thus, our solution does not require additional calculation and communication to establish secure links. Finally, although deployment knowledge schemes [5] provide good connectivity, they are based on nodes' location. Our solution operates well regardless of the position of nodes and supports their dynamic deployment as they can join and leave the network at any time. It is then more flexible and suitable for dynamic networks.

Regarding the BPs' response time, the simulations showed that the validation of the transaction is the most expensive. To reduce the total response time, we took advantage of the fact that Tendermint offers multiple endpoints to respond to a request. We used the $broadcast\_tx\_commit$ endpoint to get the results of Figure 5a. In this mode, the request returns after the transaction is committed (i.e. included in the blockchain). This approach is reliable, but can take on the order of a second. Tendermint provides an another mode, $broadcast\_tx\_sync$, in which the request returns as soon as the transaction is verified and does not wait for a block to be stored. The validation time (T_a) is therefore not included in the calculation of the total response time. In the absence of similar works, we only compare the response time using the centralized version [9] (Number of BPs is equal to one) and the two decentralized modes, according to the number of BPs (Figure 5b).

The results confirmed that the $Broadcast\_tx\_sync$ significantly reduces the response time and brings it closer to that of the centralized version. However, it does not guarantee that the transaction will be stored in the blockchain. Thus, a new attempt to add the transaction is necessary. To help the reader to choose the mode that is most suitable for a given application, we studied the error rate of the $Broadxcat\_tx\_sync$. We then calculated the percentage of transactions which are not successfully stored in the blockchain, from the first time, according to the number of BPs (Figure 5c). The results show that the more there are BPs, the more the error rate increases.



(a) Variation of the different execution times. (b) Variation of the total response time. (c) Variation of the error rate.

Fig. 5: Simulation results.

To sum up, the more we increase the number of BPs, the less nodes are attached to each BP. This improve the resilience of our solution since the capture of a BP does not jeopardize a large part of the network. Also, the more there are blockchain participants, the more it becomes impossible for a malicious BP to compromised the blockchain. On the other hand, the more we decrease the number of BPs, the more we improve the performance of our solution. According to the needs of the application, we must choose the number of BPs that provides the best comprise between resilience and performance.

VI. Conclusion

In this paper, we proposed a novel blockchain-based decentralized Key Management protocol for secure Device-to-Device communication. Unlike most of the existing solutions based on pre-distribution, our protocol supports the dynamic deployment of nodes. When a node joins or leaves the network, the KM updates the keys and distributes them to the remaining network members. To avoid the single point of failure problem, our solution uses the blockchain technology to securely decentralize the KM on several participants. We then showed that the system will still be able to operate even if an entity fails. We also proved that the compromise of an entity will not jeopardize the security of the whole network.

In future works, we intend to propose a novel consensus algorithm, which will be more suitable for heterogeneous networks such as the IoT. We also plan to design a new key agreement method for the key exchange (between BPS and nodes) to replace asymmetric encryption that is inappropriate for the IoT. We finally intend to implement our solution on real IoT platforms and to consider more parameters for testing.

Acknowledgments

This work was carried out and funded by Heudiasyc UMR CNRS 7253 and the Labex MS2T.

References

[1] E. Baburaj et al. “Polynomial and multivariate mapping-based triple-key approach for secure key distribution in wireless sensor networks”. In: *Computers & Electrical Engineering* 59 (2017), pp. 274–290.

[2] W. Bechkit, Y. Challal, A. Bouabdallah and V. Tarokh. “A highly scalable key pre-distribution scheme for wireless sensor networks”. In: *IEEE Trans on Wireless Communications* 12.2 (2013), pp. 948–959.

[3] M. Castro, B. Liskov et al. “Practical Byzantine fault tolerance”. In: *OSDI*. Vol. 99. 1999. 1999, pp. 173–186.

[4] J. Choi, J. Bang, L. Kim, M. Ahn and T. Kwon. “Location-based key management strong against insider threats in wireless sensor networks”. In: *IEEE Systems Journal* 11.2 (2015), pp. 494–502.

[5] J. Choi, J. Bang, L. Kim, M. Ahn and T. Kwon. “Location-based key management strong against insider threats in wireless sensor networks”. In: *IEEE Systems Journal* 11.2 (2017), pp. 494–502.

[6] T. Choi, H. B. Acharya and M. G. Gouda. “The best keying protocol for sensor networks”. In: *Pervasive and Mobile Computing* 9.4 (2013).

[7] E. I. W. Group and I. IoT. *IoT Developer Survey Results*. 2017. URL: https://fr.slideshare.net/IanSkerrett/iot-developer-survey-2017.

[8] T. Inc. *Tendermint*. 2020. URL: https://docs.tendermint.com/master/.

[9] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. “A Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things”. In: *2019 IEEE Global Communications Conference (Globecom2019)*. Waikoloa, USA, Dec. 2019.

[10] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. “A versatile Key Management protocol for secure Group and Device-to-Device Communication in the Internet of Things”. In: *Journal of Network and Computer Applications* 150 (2020), p. 102480.

[11] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah and Z. Sun. “Blockchain-based dynamic key management for heterogeneous intelligent transportation systems”. In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1832–1843.

[12] A. Lei, C. Ogah, P. Asuquo, H. Cruickshank and Z. Sun. “A secure key management scheme for heterogeneous secure vehicular communication systems”. In: *ZTE Communications* 21 (2016), p. 1.

[13] M. Ma, G. Shi and F. Li. “Privacy-Oriented Blockchain-based Distributed Key Management Architecture for Hierarchical Access Control in the IoT Scenario”. In: *IEEE Access* 7 (2019), pp. 34045–34059.

[14] S. Nakamoto et al. “Bitcoin: A peer-to-peer electronic cash system”. In: (2008).

[15] S. Ruj, A. Nayak and I. Stojmenovic. “Pairwise and triple key distribution in wireless sensor networks with applications”. In: *IEEE Transactions on Computers* 62.11 (2012), pp. 2224–2237.

[16] S. Ruj, A. Nayak and I. Stojmenovic. “Pairwise and triple key distribution in wireless sensor networks with applications”. In: *IEEE Transactions on Computers* 62.11 (2013), pp. 2224–2237.

[17] I.-C. Tsai, C.-M. Yu, H. Yokota and S.-Y. Kuo. “Key management in Internet of Things via Kronecker product”. In: *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE. 2017, pp. 118–124.

[18] P. Vasin. “Blackcoin’s proof-of-stake protocol v2”. In: *URL:https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf* 71 (2014).

[19] M. Vukolić. “The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication”. In: *International workshop on open problems in network security*. Springer. 2015, pp. 112–125.

[20] M. S. Yousefpoor and H. Barati. “Dynamic key management algorithms in wireless sensor networks: A survey”. In: *Computer Communications* (2018).

[21] F. Zhan, N. Yao, Z. Gao and G. Tan. “A novel key generation method for wireless sensor networks based on system of equations”. In: *Journal of Network and Computer Applications* 82 (2017), pp. 114–127.

[22] J. Zhang, H. Li and J. Li. “Key establishment scheme for wireless sensor networks based on polynomial and random key predistribution scheme”. In: *Ad Hoc Networks* 71 (2018), pp. 68–77.