Enhancing the security of the IoT LoraWAN architecture

Sarra Naoui, Mohamed Elhoucine Elhdhili, Leila Azouz Saidane
CRISTAL Laboratory
ENSI, University of Manouba,
Tunisia
Emails: naoui.sarra1@gmail.com, med_elhdhili@yahoo.es, leila.saidane@ensi.rnu.tn

Abstract—**The Internet of things (IoT) is invading our lives by connecting surrounding human things to the Internet. This allows a smarter and comfortable human living space. However, it exposes private and sensitive data as well as human appliances to attackers. Security solutions based on cryptography might solve these problems. However, these solutions need the use of encryption keys that must be managed securely and properly. We investigate, in this paper, existing key management protocols proposed for the Internet of things. Then, we propose a solution to enhance the security of the IoT loraWAN architecture. Our solution is inspired from an existing key management solution that uses proxy nodes to alleviate the computation in the constrained node side. We propose the use of a reputation system for proxy nodes selection which permits a reliable and attacks free security solution.**

1. Introduction

In the past, people communicate either with each others or with machines. But, with the appearance of the IoT, physical objects have the ability to communicate with each others without human intervention. Thus, the IoT can be defined as the interconnection of uniquely identifiable objects within the existing Internet infrastructure where objects collect data from the world around us and share it across the Internet to be utilized for various interesting applications as smart grids, smart cities, smart homes and e-health [12].

The aim of the IoT is to make our everyday lives better and easier and to enhance the productivity and efficiency of enterprises and employees. However, it will have an impact on our private lives if the collected data is compromised. Thus, to achieve a complete vision of the IoT, security solutions must be used to protect sensitive and private data from attacks and malicious use. This usually needs the use of cryptographic keys that must be properly managed according to the IoT characteristics, especially, the heterogeneity of nodes making constrained ones unable to execute heavy cryptographic or key management functions. As a consequence, key management protocols and security architectures solutions adapted for the IoT contexts have been proposed in the literature. The proposed key management protocols are classified into two main categories as specified in [1]: two party communication key management protocols and group communication ones where a common key is created between two (respectively three or more) entities to secure their communications. For the security architecture, many projects were launched to design a common secure architecture based on the analysis of the researchers and industry needs where the LoraWan architecture is among the most recent projects. In this paper we focus on the LoraWan architecture that we will enhance. This architecture has been proposed to provide interoperability and secure exchange among smart things. This has been solved by several layers of encryption [17] ; a network layer that is responsible for the end node data authentication. This is verified via an AES128 secret key shared between the end device and the network server. Then, an application layer that is responsible for ensuring the device data privacy by using an AES128 secret key shared between the end node and the user application. In this paper, we propose a solution to enhance the security of the LorAWan architecture that is based on an existing distributed two party key management protocol and a reputation system.

The rest of this paper is organized as follows. Section 2 presents two party existing IoT key management protocols described in the literature. In section 3, we give an overview of the IoT LoraWan architecture. In section 4 and 5 we detail the proposed solution then we evaluate it. In section 6 we conclude the paper and outline our future works.

2. Related work

Several protocols have been proposed for key management in the IoT. The majority of these protocols is deeply related to IoT applications contexts and relies on the well-known suite of cryptographic algorithms: the Diffie Hellman (DH) asymmetric key agreement algorithm, Elliptic Curve Cryptography (ECC) asymmetric algorithm and secure hash algorithms such as SHA and hash chains. In this paper, we focused on existing two party IoT key management protocols presented in the literature. We propose an improvement of one of these protocols to be used to enhance the IoT LoraWAN security.

Several two party IoT key management protocols have been proposed in the literature. Authors of [13] proposed a key management protocol for the industrial and mobile systems called KMIC. Its purpose is to create a common session key between two different nodes using the ECQV (Elliptic Curve Qu-Vanstone) for the generation of implicit certificates, the ECDH (Elliptic Curve Diffie Hellman) exchange to create a common session key and nonces to avoid replay attacks and to guarantee data authentication. To construct a session key, the protocol includes an exchange of four messages between two nodes A and B (shown below) and the usage of implicit certificates. Noting that the implicit certificates are based on ECQV algorithms. The most important with ECQV is that a public key Pb_A of node A with identity I_A can be derived by another node B using A's implicit certificate P_A and the certification authority's public key C as follows: $Pb_A = C + P_A.H(P_A, I_A)$ Where H is an arbitrary hash function.

1) First message: node A sends its certificate received from the certification authority and its nonce generated randomly to node B. B computes A's public key using the received certificate. Then B computes the common master key using its private key and A's public key. Afterwards, B uses the master key to derive a secret key.
2) Second message: node B sends its certificate received from the certification authority and its nonce to node A. A computes B's public key using the received certificate. Then A calculates the common master key using its private key and B's public key. Afterwards, A uses the master key to derive a secret key.
3) Third message: node A sends the Message Authentication Code (MAC) of the secret key to B to check that they agreed on the same key.
4) Fourth message: similarly node B sends the MAC of the secret key to A.

Once the key verification is successful, the key will be ready for use.
This protocol has many advantages. First, it uses implicit certificates that get the same benefits of ordinary certificates but with lower processing and smaller storage capacity. Each node in the network has a key pair (private and public) calculated from an exchange with the certification authority, and is able to calculate the public key of another node in the network after a simple exchange of their certificates. Second, the use of implicit certificates attaches a public key to its owner which makes the protocol robust against Man-in-the-middle (MITM) attacks. Third, the reduction of exchanged messages (just four messages) to create a session key, which reduces energy consumption. Another advantage is the use of nonces to protect the communication from replay attacks. However, KMIC protocol might not be scalable as the number of stored keys will increase with the increase of the number of nodes to communicate with because each node must save the keys used to communicate with all other nodes in the network.

In [14], a proxy based end to end key establishment protocol is described. This protocol allows two IoT objects in a local network with energy and computing constraints to create a common session key with the help of a subset of k neighbours called proxies as described in figure 1. These proxies work together to help in the heavy cryptographic operations. In order to derive the session key, the protocol uses the Diffie Hellman (DH) exchange and threshold cryptography based on the polynomial function f that verifies : $f(x) = q_0 + q_1 x + ... + q_{k-1} x^{k-1}$; where $q_1, q_2, ..., q_{k-1}$ are uniform, random and independent and p is a random high prime number. The function f can be derived according to Lagrange formula given by equation 1.

$$f(x) = \sum_{i=1}^{k} f(i). \prod_{j=1,\ j\neq i}^{k} \left(\frac{x-j}{i-j}\right) mod\ p \qquad (1)$$

The DH private key a of the transmitter node is equal to q_0 and can be expressed, using equation 1.

$$a = f(0) = \sum_{i=1}^{k} f(i). \prod_{j=1,\ j\neq i}^{k} \left(\frac{-j}{i-j}\right) mod\ p \qquad (2)$$

and the shares of a are $a_i = f(i)$.
So, for a communication between two nodes A and B, the creation of a session key involves the following steps (see figure 1):

1) Node A calculates n values $f(1), f(2), ..., f(n)$ of the polynomial function f and $f(0)$ with $n > k$. Then, it sends for each proxy i the value $f(i)$ encrypted using the pre-shared key k_{ai}
2) Each A's Proxy neighbour decrypts the message. Then, it calculates its part of A's DH public key $g^{f(i)} mod\ p$ and sends it to B's proxy neighbours encrypted using the shared secret key k_i.
3) Each B's Proxy neighbour decrypts the received $g^{f(i)} mod\ p$ then encrypts it using the pre-sahred key k_{bi} and sends it to node B.
4) Node B, Upon receiving the messages, does the following :
    a) Calculates C_i coefficients as follows: $C_i = \prod_{i=j,\ i\neq j}^{k} \left(\frac{-j}{i-j}\right)$
    b) Calculates the DH public key of node A using Langrange formula and C_i values as follows: $\prod_i g^{f(i)^{c(i)}} modp = g^{\sum f(i).c(i)} modp = g^a modp$.
    c) Calculates its DH private key b given by: $b = f'(0) = \sum_{i=1}^{k} f'(i). \prod_{j=1,\ i\neq j}^{k} \left(\frac{-j}{i-j}\right)$
    d) Calculates the shared secret key $K_{AB} = g^{ab} modp$.
    e) Encrypts $f'(i)$ and C_i values, then sends them to its proxy neighbours.
5) Each B's Proxy neighbour, upon receiving the messages, calculates: $g^{f'(i)} modp$, $g^b modp$ and $g^{bC_i} modp$ which is the DH public key of node B. Then sends the encrypted $g^{bC_i} modp$ to A's proxy neighbours.
6) Each A's Proxy neighbour uses it's value $f(i)$ to calculate $K_i = (g^{bC_i} modp)^{f(i)}$,then encrypts and delivers it to node A.

7) Node A uses equation 3 to compute the shared secret key K_{AB}.

$$K_{AB} = \prod_i K_i = \prod_i (g^{bC_i} mod\ p)^{f(i)} = g^{ab} mod\ p \tag{3}$$

This protocol exploits nodes heterogeneity where the heavy cryptographic operations are ensured by powerful neighbour nodes. Each proxy calculates a key portion but cannot derive the final key that is only calculated at the transmitter and receiver nodes which make it robust against Man-in-the-middle attacks. But the robustness of a cryptographic key depends on the number of proxy neighbours that will participate in the key generation process. In fact, the loss of a message or a fake one can influence the computation of the final key because the key is constructed from the messages received by all neighbours. In addition, the protocol supposed that the group of proxy nodes is stored into the transmitter and receiver nodes, but when the neighbouring nodes are mobile (e.g. Smartphone), the list of proxy nodes must change, so we need to discover all nodes present in the transmitter and receiver neighbourhood before sending data.

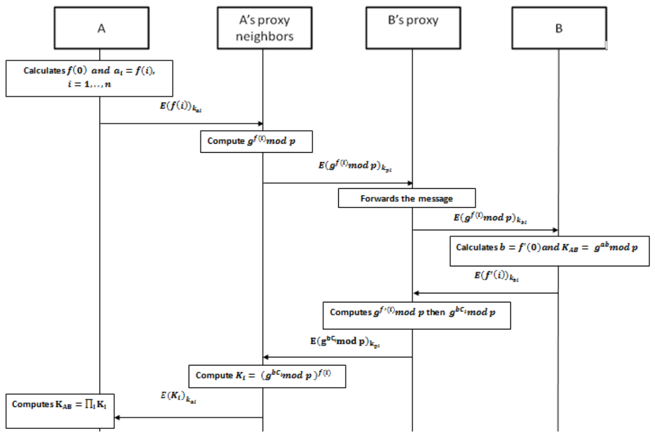
A
A's proxy neighbors
B's proxy
B
Calculates $f(0)$ and $a_i = f(i)$, $i = 1,\dots,n$
$E(f(i))_{k_{ai}}$
Compute $g^{f(i)} mod\ p$
$E(g^{f(i)} mod\ p)_{k_{pi}}$
Forwards the message
$E(g^{f(i)} mod\ p)_{k_{bi}}$
Calculates $b = f'(0)$ and $K_{AB} = g^{ab} mod\ p$
$E(f'(i))_{k_{bi}}$
Computes $g^{f'(i)} mod\ p$ then $g^{bC_i} mod\ p$
$E(g^{bC_i} mod\ p)_{k_{pi}}$
Compute $K_i = (g^{bC_i} mod\ p)^{f(i)}$
$E(K_i)_{k_{ai}}$
Computes $K_{AB} = \prod_i K_i$

Figure 1: DH exchange

In the same way, authors of [11] proposed a cooperative end to end Key management protocol for E-Health applications in the context of IoT. This protocol offloads heavy cryptographic functions on third parties and creates a session key between a highly constrained node (sensor) and a server according to the following steps:

1) A sensor node generates a secret S that is split and sent to the third parties. Pre-shared keys are used to secure each part of the secret S and message authentication codes (MAC) are used for authentication.
2) Upon receiving the secret parts, each third party uses asymmetric algorithms to send the received secret securely to the server. Digital signatures are used to ensure authentication. Noting that each third party gives its certificate including its public key to the server and proves that it is authorized by the sensor node to act on its behalf.
3) After a successful authentication and decryption of the received parts, the server reassembles the secret S which is used to derive a master key. The derivation is based on hash function agreed upon during an initialization phase.

After this collaborative exchange, the sensor nodes and the server are able to derive a session key to encrypt and authenticate the exchanged data.

Another solution for DH key establishment between resources constrained nodes and unconstrained ones was proposed in [4]. This solution, called D-HIP (distributed Key Exchange scheme for HIP-based IoT), proposes a distributed and lightweight key exchange protocol to reduce the computationally expensive cryptographic operations of HIP (Host Identity Protocol) described in [2]. Heavy cryptographic operations are delegated to less constrained nodes called proxies in the neighborhood. Thus, each proxy participates in the computation and delivery of a part of initiators public DH key and also takes a part in the derivation of the common DH key before its building at the initiator.

3. LoraWan

LoraWan technology has been proposed to optimize the LowPower Wide-Area Networks (LPWAN) for battery lifetime, cost, capacity and range. This section presents the LoraWan architecture and the techniques it uses for communication.

3.1. LoraWan achitecture

The LoraWan architecture has been proposed in the context of IoT to provide interoperability among smart things without complex local installations. The LoraWan architecture is composed of end nodes, gateways, a network server and an application server as presented in figure 2. First, the end node sends the gathered data to one or multiple gateways using the Lora physical layer. Then, each gateway will send the received data from end nodes to the network server using some backhaul (WI-Fi, cellular, Ethernet or satellite). The network server is the intelligent entity that will manage the network, perform the security checks, perform adaptive data rates, filter the redundant received packets...etc. Table 1 presents the advantages of this architecture compared to other proposed ones for IoT.

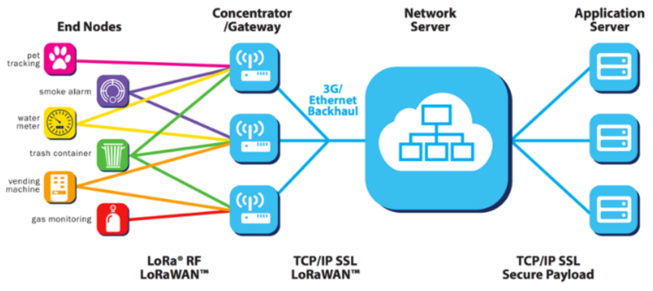

Figure 2: LoraWan architecture [17]

TABLE 1: LoraWan advantages

Advantage	Description
battery lifetime reduction	Nodes are asynchronous and just communicate when they have to send data.
High network capacity	It is achieved by using adaptive data rate and utilizing a multichannel multi-modem transceiver in the gateway where simultaneous messages on multiple channels can be received.
Different device classes	Three classes are defined : A, B and C. The classes differs on the downlink communication latency that is chosen depending on the device lifetime.
Security	Two layers of security are defined; one for the network and one for the application using AES encryption.

3.2. LoraWan security

For data security, LoraWan uses two security layers as presented in figure 3 ; the network and the application layers. The network layer is responsible of the end node data authentication. This is verified via an AES128 secret key shared between the end device and the network server. As for the application layer, it is responsible for ensuring the device data privacy by using an AES128 secret key shared between the end node and the user application.

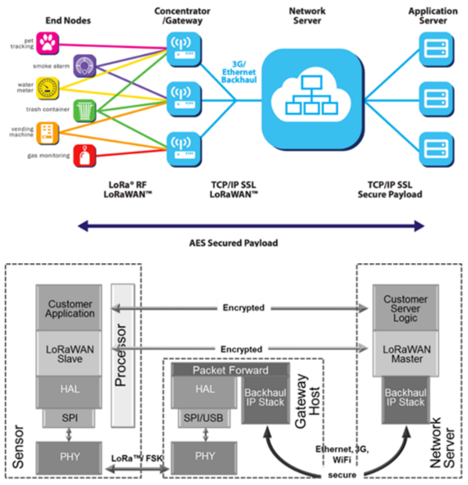

Figure 3: LoraWan Security architecture [17]

LoraWan uses an AES128 secret key to secure the exchange between the end node and the network server. If an attacker gets that secret key, he could read and modify all messages exchanged between the end node and the network server. Thus, we propose the use of a session key between the end node and the network server. This key will be changed in each session which permits a more secure communication.

4. Proposed solution

We propose a solution to enhance the LoraWAN security protocols. Our solution uses a reputation system and is inspired from the key management solution described in [14] that uses proxy nodes to alleviate the computation in the constrained node side which will improve the LoraWan architecture security as described in figure 4. The shared AES128 key will be replaced by a session key. This key will be established between the end node and the network server according to the solution described in [14]. However, we introduce the use of a reputation system in order to select trustworthy nodes as proxies. This permits not only to improve the key robustness since only confident nodes are involved in the key derivation phase, but also to avoid attacks that can be launched by malicious nodes.

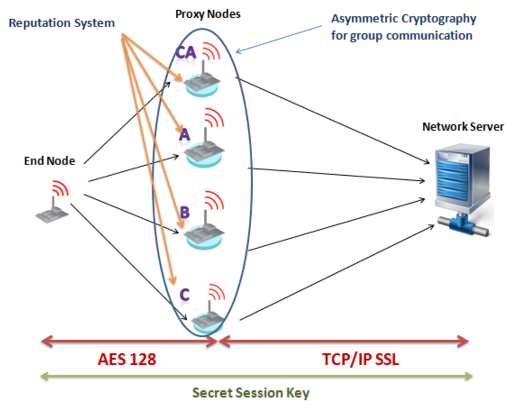

Figure 4: New Security architecture

4.1. Network architecture

In the proposed solution, we based our work on the LoraWan architecture that interconnects heterogeneous nodes with various capabilities in terms of energy resources and computing power. So, we consider three different nodes:

1) End node: it is a highly resource-constrained node that does not support heavy cryptographic operations.
2) Gateways: less constrained nodes present at the end nodes neighborhood and able to perform heavy cryptographic operations.
3) Network server: with high storage capabilities, computing power and high energy.

4.2. Assumptions

We assume that:

1) the communication between the end node and the proxy nodes is secured using AES128 secret key.
2) The AES128 secret key is exchanged securely between the end node and each gateway.
3) The first Certification Authority and its corresponding AES128 secret key are defined in advance into the end node dynamic list before deployment.

4.3. Overview of the protocol

The protocol described in [14] exploits nodes heterogeneity where the constrained nodes execute cryptographic functions implicitly and powerful neighbour nodes execute the heavy cryptographic operations. The advantage of this protocol is that each proxy calculates a portion of the session key but cannot derive the final key that is only calculated at the transmitter and receiver nodes. But the problem is that each node saves the list of proxy nodes and their corresponding pre-shared keys which requires a large storage capacity when the number of neighbours increases. Also the robustness of a cryptographic key depends on the number of proxy neighbours that will participate in the key generation process and the loss of a message or a fake one can influence the computation of the final key. So, as an improvement for this solution, we propose that the transmitter node discovers the proxy nodes and choose the best ones to participate in the key derivation. Then, it stores them in a dynamic list which is updated at each key computation.

4.4. Protocol details

In our solution, we assume that each proxy node will collect informations about other nodes to derive the reputation of each proxy. The collected informations are either direct informations that node has collected by itself or recommendations received from other nodes. Then, the node gathers data in a trusted table that contains all the neighbours and their trust value as shown in figure 5. The trust value is calculated by evaluating the nodes' behavior including relaying data packets, sending correct partial keys, normal leaving, normal joining, available power, and available bandwidth. A well behaving node will see an increase of its trust values maintained as seen by other nodes whereas a bad behaving node will see the contrary.

The proxy node with the highest trust value will be selected as a Certification Authority (CA) that will handle the authentication, authorization and key management for the proxy nodes. The CA has three tables; a certification table used to manage the proxy nodes' certificates, a trust table that contains the trust value of each proxy node and a black list that contains all the nodes with zero trust values . At bootstrap, we suppose that the first CA is preselected. Then, it will be updated according to the dynamic nodes trust values.

Each proxy node will send its trust table to the CA secured with its public key. The CA will reassemble the tables received from all proxy nodes and build a group trusted table that will be sent to the end node secured with the AES128 secret key. The end node will be based on the group trusted table to choose the best neighbours that will participate in the key derivation as in the example presented in figure 5 and to inform all proxy nodes about the new CA .

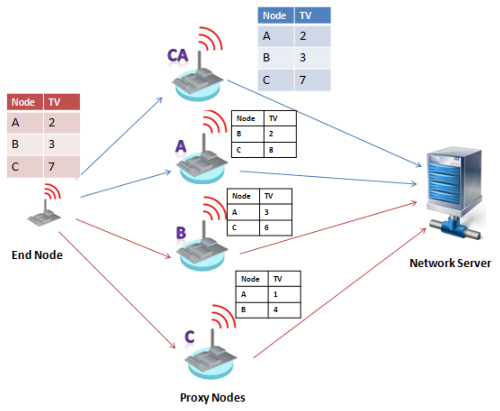

Figure 5: Choosing the best neighbours

4.4.1. Protocol initial phase. Initially, the CA forms a group of gateways (proxy nodes) and creates two tables which are the certification table and the trust table (which is initially empty). Then, the CA multicasts the public key of each gateway to all other group members. So, every proxy node has all the public keys of other group members. The public keys will be used to secure the informations exchanged between the group members on whom the proxy nodes will be based to create their trust table. Then, each member sends its trust table secured with its public key to the CA . Upon receiving all trust tables, the CA makes the average of the received trust values and creates the final trust table that will be sent to the end node. So, when the end node wants to share data with a group of gateways, it uses the received table to choose the best proxies with the highest trust values.

4.4.2. Joining node phase. When a new gateway wants to join the group of proxy nodes as in figure 6, it broadcasts a message to all group members to discover the group CA. The first proxy node that will receive the message responds with a message containing the group identifier and the certification authority identifier. Then, the node sends a joining request to the CA containing its identifier and its public key. The CA checks whether the new node already has a trust value in the trust table or if it is placed in the black list, if not, it will save it in the trust table with trust value equal to 1. Then, the CA responds with a message containing the group identifier and the node trust value, this message will be encrypted with the CA private key and with the node public key. Once the node decrypts the message, it becomes active.

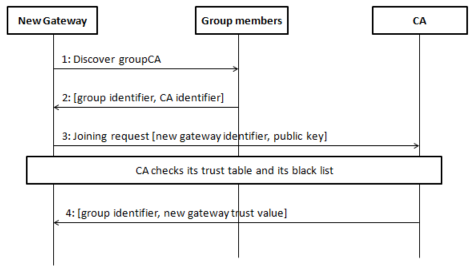
New Gateway
Group members
CA
1: Discover groupCA
2: [group identifier, CA identifier]
3: Joining request [new gateway identifier, public key]
CA checks its trust table and its black list
4: [group identifier, new gateway trust value]

Figure 6: Joining phase

4.4.3. Leaving node phase. As described in figure 7 The node sends a leaving message to the CA containing its identifier. Upon receiving the message, the CA deletes the node information from the certification table and updates its trust value (that will be decremented by 1). Then, the CA responds with a message to inform the node that its certificate has been deleted and can quit successfully.

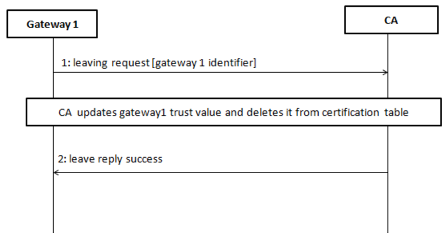
Gateway 1
CA
1: leaving request [gateway 1 identifier]
CA updates gateway1 trust value and deletes it from certification table
2: leave reply success

Figure 7: Leaving phase

4.4.4. Choosing new certification authority phase. When the CA updates its trust table and finds that an existing gateway has a better trust value, it notifies it to be the new CA. The new CA must update the group members certifications for security reasons. So, it sends a message to each group member for a request to issue a new certificate. Each proxy node sends its identifier and its public key to the new CA which will send the new individual certificate to each gateway. Noting that the old CA will behave as a proxy node.

5. Proposed solution security evaluation

The proposed solution verifies all the security objectives, especially authentication, confidentiality and integrity. In fact, the sensitive data is encrypted with the corresponding keys which guarantees the message confidentiality. Also, the protocol uses AES128 secret key to secure communication between the end node and the getaways which is relatively fast and does not require heavy computation for constrained nodes. It uses SSL for securing communication between the gateways and the network server which provides security and authenticity. The messages exchanged between the group of proxy nodes are secured using node's public keys. This ensures authenticity and protect the communication from man-in-the-middle attacks because the message can be decrypted only by the CA which possesses the proxy node private key. In addition, the protocol is robust against man-in-the-middle attack because the message sent to a new joining member is encrypted using the new node public key and the CA private key and none of the proxy that receives one share of the secret can derive the final session key.

6. Conclusion

In this paper, we proposed a solution that enhances the security of the IoT LoraWAn architecture. Our solution is inspired from an existing two party key management solution and uses a reputation system to select trustworthy neighbour nodes as helpers for some heavy cryptographic functions. We proposed the use of a session key between the end node and the network server to avoid that a same AES secret key is used for all exchanges where getting the key by an attacker will allow him to decrypt all exchanges. For the future work, we plan to add a markov chain model in the proposed solution to evaluate the trust value of each proxy node based on its behaviour history. We also propose to model the change of trust state of each proxy node with a markov chain model where several events (relaying packets, normal joining, normal leaving, ...) are used to change from one trust state to another.

References

[1] Y. Challal and H. Seba, *Group Key Management Protocols: A novel Taxonomy*, International Journal of information technology, 2005, pp. 105-118.

[2] R. Moskowitz ,P .Nikander, P .Jokela and T. Henderson,*Host Identity Protocol*,IETF RFC 5201, April 2008.

[3] J. Shen, M. Sangman and I. Chung, *A Novel Key Management Protocol in Body Area Networks* , ICNS 2011 : The Seventh International Conference on Networking and Services, pp. 246-251.

[4] Y. Ben Saied and A. Olivereau, *D-HIP: A distributed key exchange scheme for HIP-based Internet of Things*, World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a, June 2012, pp. 1-7.

[5] J. Liu, Y. Xiao and C.L. Philip Chen, *Authentication and Access Control in the Internet of Things*, ICDCSW, 2012, 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops 2012, pp. 588-592.

[6] Yue Li, *Design of a Key Establishment Protocol for Smart Home Energy Management System*, 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), June 2013, pp. 88-93.

[7] L. Veltri, S. Cirani, S. Busanelli and G. Ferrari, *A novel batch-based group key management protocol applied to the Internet of Things*, Ad Hoc Networks,November 2013, vol. 11, pp. 27242737.

[8] R. Chandramouli, M. Iorga and S. Chokhani,*Cryptographic Key Management Issues and Challenges in Cloud Services*, Secure Cloud Computing, December 2013, pp. 1-30.

[9] K. Zhao and G. Lina, *A survey on the Internet of Things Security*, 9th International Conference on Computational Intelligence and Security (CIS), 2013, pp. 663-667.

[10] N. Renugadevi, G. Swaminathan and A.S Kumar,*Key Management Schemes for Secure Group Communication in Wireless Networks-Asurvey*, International Conference on Contemporary Computing and Informatics,November 2014, pp. 446-450.

[11] M. Riyadh AbdmeziemAffiliated, T. Djamel , *A Cooperative End to End Key Management Scheme for E-health Applications in the Context of Internet of Things*, Ad-hoc Networks and Wireless, February 2015, vol. 8629, pp. 35-46.

[12] A. Whitmore, A. Agarwal and Li Da Xu, *The Internet of Things-Asurvey of topics and trends*, Information Systems Frontiers, April 2015, vol. 17, pp. 261-274.

[13] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia and G. Bianchi, *Key Management Protocol with Implicit Certificates for IoT systems*, IoT-Sys '15 Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems,2015, pp. 37-42.

[14] An. Braeken, P. Kumar, A. Gurtov, M. Ylianttila, *Proxy-based end-to-end key establishment protocol for the Internet of Things*, 2015 IEEE International Conference on Communication Workshop (ICCW), pp. 2677-2682.

[15] M. Riyadh Abdmeziem, T. Djamel and I. Romdhani, *A Decentralized Batch-Based Group Key Management Protocol for Mobile Internet of Things (DBGK)*, 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015, pp. 1109-1117.

[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, *Internet of Things: A survey on Enabling Technologies, Protocols and Apllications*, IEEE Communications Survey and Tutorials, 2015, vol. 17, pp. 2347-2376.

[17] LoRa Alliance Technical Marketing Workgroup, *technical overview of LoRa and LoRaWAN*, 2400 Camino Ramon, San Ramon, CA 94583, help@lora-alliance.org