Wireless Personal Communications (2021) 120:113–137
https://doi.org/10.1007/s11277-021-08437-9

Reliable and Adaptive Distributed Public-Key Management Infrastructure for the Internet of Things

Samia Belattaf[1] · **Mohamed Mohammedi**[2] · **Mawloud Omar**[3] · **Rachida Aoudjit**[1]

Accepted: 22 March 2021 / Published online: 3 April 2021

Abstract

The Internet of Things (IoT) is an emerging paradigm in the world of computer networks, where physical objects are connected over the Internet. Given the huge number of connected devices that are potentially vulnerable, highly significant risks emerge around the issues of communication security. Unfortunately, the conventional security methods are hard to adapt due to the heterogeneity and resource-constrained objects. In this paper, we propose a reliable and adaptive distributed public-key management infrastructure for the IoT. The main purpose of our proposal is to mitigate the resource issue for public-key certificate management to design an effective security mechanism, which offers reliable end-to-end security services. In order to evaluate the performances of our proposal, we have conducted intensive simulations with comparison to the literature, in which we have obtained promising results in terms of storage, communication, response time and resilience against malicious nodes.

Keywords Key management · Security · Certification · IoT

1 Introduction

The Internet of Things (IoT) is taking over an important position in our daily life and playing a crucial role in economic development. It gets to be a critical innovation to create an intelligent environment. IoT is a large scale network interconnecting heterogeneous devices designed from different technologies. Though the IoT growth makes devices smarter, their usage might be compromised by intruders, which increases dramatically the attack area on the network. The impact of attacks gotten to be more damaging, where several enterprises had been subject of intrusions. Since IoT is composed of resource-limited devices, this makes them highly vulnerable, and require specific security solutions. In this paper, we give a focus

✉ Mawloud Omar
mawloud.omar@univ-eiffel.fr

[1] Laboratoire de Recherche en Informatique (LARI), Faculté de Génie Electrique et Informatique, Université Mouloud Mammeri, Tizi-Ouzou, Algeria

[2] Laboratoire d'Informatique Médicale, Faculté des Sciences Exactes, Université de Bejaia, 06000 Bejaia, Algeria

[3] LIGM, ESIEE Paris, Université Gustave-Eiffel, Noisy-le-Grand, France

on key management, which is one of the most important services to deploy in such environment in order to secure the exchanged data between the network devices.

Security solutions based on symmetric keys are considered as practicable to offer end-to-end security services, such as confidentiality, integrity and authenticity. However, the key management with symmetric key-based solutions is costly and not scalable, especially when we consider inter-domain communication in the IoT. The reason for this is the requirement of one key per peer and the pre-deployment of these keys. Public-key cryptography is an alternative way of providing end-to-end security and it is the dominating security solution on the Internet. Public-key cryptography solves the key management issues and allows peer authentication without a pre-shared secret on both sides. Furthermore, public-key cryptography, specifically the certificate-based approach, offers additional services that enhance the vision of the IoT. Symmetric key-based solutions fall short, such as service access based on the group membership or service type. However, some works (such as [1, 2]) consider that public-key cryptography with certificates requires more resources than symmetric key-based solutions. In this context focuses the framework of this work by addressing the resource issue in public-key management in the IoT context.

In this paper, we propose a reliable and adaptive distributed public-key management infrastructure for the IoT. The IoT network is organized into various disjoint communities. Each Community operates under a specific public-key certificate management policy, such as hierarchical certification, Web-of-trust based certification, and threshold certification. In our framework, the key management is performed through the trust propagation among communities in the global graph of trust. A community contains a set of objects and a delegate object. The community members send the authentication request to the delegate object, which further sends it to the external communities. The delegate object is selected among those having a high degree of resources and trustworthy in both inside and outside the delegated community. Regarding the application requirements, the public-key certificate recovery process can be implemented in two different ways, namely: proactive or reactive.

We summarize the main contributions of this paper as follows:

1. The proposition of a public-key management infrastructure, which makes several autonomous communities coexist;
2. The proposition of a distributed algorithm for the determination of strongly connected trust components connecting the IoT communities;
3. The proposition of an objective function selecting the most reliable delegate objects;
4. The implementation of the public-key certificate recovery process in two different ways: proactive and reactive.

The remaining of this paper is structured as follows. In Sect. 2, we review the literature. In Sect. 3, we present the detailed description of the proposed framework. In Sect. 4, we analyze its performances with comparison to the literature by assessing the storage, communication, response time, and the resilience against malicious nodes. Finally, we conclude this paper in Sect. 5.

2 Related Work

Security in the IoT has become an essential necessity to better prevent malicious elements from sneaking into the network to spy on, inject, or reuse data. Great efforts have been made these last years to develop security mechanisms to enhance communications security in the IoT [3, 4]. With the increasing attention given to key establishment solutions using low-power security primitives, research on the development of efficient key management infrastructures has proliferated. In order to ensure the robustness of key management infrastructures, most of the solutions are based on cryptographic primitives, key graphs, Kronecker product, hardware security primitives, etc.

In [5], Sciancalepore et al. proposed a key management protocol using implicit certificates to provide the security services in the IoT. This protocol is based on two main algorithms: ECQV (Elliptic Curve Qu-Vanstone) and ECDH (Elliptic Curve Diffie Hellman). The former is used for the generation of implicit certificates for the authentication process, and the latter for the key agreement between the communication entities. The implicit certificates ECQV allow to provide the authentication service between the IoT entities. The ECDH algorithm allows each entity of the IoT, which has authenticated itself with ECDV, to compute the same secret key from an authenticated public-key.

In [6], Roman et al. analyzed the solutions for the session key establishment issue in a client-server architecture for the IoT. They studied the suitability of two security mechanisms such as public-key cryptography and pre-shared keys. To generate the same session key for the same objects, the authors analyzed the applicability of link-layer oriented key management system.

In [7], Dahshan proposed a distributed key management protocol based upon elliptic curve cryptography (ECC) among the IoT entities. During the initialization phase, the certification authority generates a key pair (public and private key), which will be provided to all the IoT entities. This key pair is the private key of each entity, which will be used only when generating a threshold signature. After the network deployment, each entity executes the key generation protocol to deliver its session key pair.

In [8], Tsai et al. proposed a new key management protocol by using Kronecker Product for the IoT. The key establishment process is based on the construction of the symmetric matrix to generate keys between the different node pairs. Thus, each node pair will have the same cryptographic key to secure its future communication. The computation of this symmetric shared key between the node pair has been done separately between the two nodes since the matrix used in the computation is symmetrical. Therefore, the nodes must compute the corresponding indexes of the column vectors of the used matrix. Both nodes should obtain an identical cryptographic key for this computation.

In [9], Ben Saied et al. proposed a collaborative key management protocol for the IoT. The main purpose of this protocol is to address the security issues and requirements that key management protocols of the literature suffer from. The principle is to delegate the cryptographic load of nodes constrained in resources to other assistance nodes less resource-constrained, which are in close proximity. This by exploiting the spatial heterogeneity of the IoT. In order to ensure collaborative behavior in Ben Saied et al.'s protocol, two techniques were designed for the mode of transport and key agreement.

In [10], Veltri et al. proposed a centralized distribution and key management protocol for the IoT, which aims to minimize the system computation load and network traffic. The group key distribution process is provided through a central entity called KDC (Key Distribution Center) to deal with the issue of the frequent joins and leaves of users in the IoT.

To solve the scalability issue of group/multicast key management, Wong et al. [11] proposed a protocol of the logical key hierarchy. This protocol is based on key graphs with a hierarchy structure, and the key set collection of all the system users is managed and maintained by the key server. A user *i* wishing to join a communication group begins by sending a request to the key server. By using a specific authentication protocol the two communications parties mutually authenticate each other. If the authentication process is accomplished successfully and the user *i* is accepted to be a member in the group, then the user *i* shares a symmetric individual key with the key server.

In [12], Gu and Potkonjak proposed a group key management protocol for energy constrained networks, which aims to secure group communications in the IoT. The protocol is based on a low energy PUF (physically unclonable function) structure named multistage interconnected PUF. The PUF hardware security primitives used in this protocol added an extra security layer at the protocol stack, which provides an important role in the persistent storage and group key management mechanism. This is ensured with each reconfiguration of the MIPUF of all the entities. Consequently, the main concern of this protocol is to ensure key management tasks based on MIPUF, including key distribution, key storage, and rekeying.

In [13], Hoglund et al. proposed an automated certificate registration protocol named PKI4IoT (Towards Public Key Infrastructure for the Internet of Things) for heavily constrained devices. The goal is to ensure the end-to-end security between both certification authorities (CAs) and recipient IoT devices. In order to issue traditional X.509 digital certificate to IoT devices by existing CAs, the authors designed a lightweight profile for X.509 certificates with CBOR encoding, called XIOT. In the designed profile, edge devices convert X.509 digital certificates to and from the XIOT format over constrained networks.

In [14], Braeken et al. proposed an anonymous lightweight proxy based key agreement protocols, called ALPKA. This protocol allows to establish a common shared key between two highly resource-constrained IoT devices without prior trust relation through a proxy. The use of symmetric key cryptography makes this protocol lightweight. The authors added the feature of anonymity to the system. Hence, no information on the identity of the sender and the receiver can be derived by an outsider from the messages exchanged.

In [15], Tabassum et al. proposed a Smart Object (SO)-based key management technique for the IoT that uses a combination of symmetric and asymmetric cryptosystems. This smart objects are capable for registering, generating, and distributing keys. In their proposal, the authors used the open-source Message Queuing Telemetry Transport (MQTT) protocol to facilitate communications between the source and the destination nodes. To demonstrate their system, they developed a hybrid approach that can simulate/emulate the proposed key management technique by integrating physical smart object with simulation data.

Up until now much important work has been carried out on key management for the IoT, but there are still many gaps to be filled. In other words, there is again a long road ahead to get rid of security issues. In order to ensure an optimum security level within such a system, the entities must contribute to security mechanisms (public-key cryptography and pre-shared keys) to agree on certain security parameters such as secret cryptographic keys. These security mechanisms, which rely on cryptographic operations more or less complex causing a computation and storage loads, as well as a high communication overhead. However, the IoT is heavily constrained in hardware resources: they have some limitations in terms of bandwidth, memory space, battery capacity, and computing power. Despite these restrictions, the entities require a lightweight key management system. The aim of this work is to mitigate the resource issue in public-key management in the IoT to design an effective security mechanism, which can offer end-to-end security services in such a system.

3 The Proposed Framework

In this section, we discuss first the certificate format in the IoT context, and then our proposal main operations: key management infrastructure, key authentication and delegate object selection.

3.1 Certificate Format

Most of the existing approaches ensures the object authentication through the well known certificate standard, such as X.509, which is expensive in terms of storage. Therefore, the transmission of these certificates requires significant bandwidth, thus it is not well suitable for the IoT. In the following, we describe the structure of certificate used in this article.

- Certificate type: this field indicates if the certificate is whether ordinary or partial.
- Encryption tools: this field identifies the algorithm with which the public-key is used (e.g., RSA, DSA, and ECC). It also contains the hash function and the identifier for the algorithm used by the signer (or to generate the signature).
- Signer identifier: this field indicates who signs the certificate. It could be a certification authority, a server among a distributed certification authority, or a simple object, following the community certification policy.
- Validity period: this field is represented by two dates, namely, the start and end dates of the certified public-key.
- Object identifier: this field indicates the holder object identity.
- Public-key: this field contains the public-key of the object.
- Signature: this field contains a digital signature, which could be either complete or partial following the community certification policy.
- Delegated authority: this field is optional, which indicates whether the holder object is in addition a certification authority or not.
- Certificate identifier: an abstract attribute composed by the signer identifier, the object identifier and the certificate period of validity.

3.2 Key Management Infrastructure

The proposed infrastructure makes several autonomous communities coexist. Each one of them operates under a specific public-key certificate management policy. The alliance of these policies is performed through a global infrastructure, which is in charge to ensure the passage from one community to another. In the framework of this work, we focus on the most popular certification models, namely hierarchical certification, Web-of-trust based certification, and threshold certification. It should be noted that our proposal is not restricted to these certification models. Other models can be integrated to the global trust graph.

The hierarchical certification model is adapted to infrastructure-based sub-networks, where a certification authority could be centralized. This authority delivers public-key certificates for the supervised community objects as well as for other sub-servers, which exercises the same authority over the objects belonging to the hierarchy lower levels. The access point toward the global infrastructure is realized through the community root certification authority. In the Web-of-trust based certification model, there are no certification authorities. Each community object certifies public-keys to another object based on their local trust knowledge. All the issued certificates in the system form a certificate graph. The access point toward the global infrastructure is realized through one object of the community. With the threshold certification models, a distributed authority ensures the key management service. To cope with the unprompted nature of certain types of sub-networks, the service is distributed among a set of servers, which jointly deliver a public-key certificate for an object. The authority private-key is split into several shares and each server holds one of them. Each server generates a partial view of the object certificate, and later, the holder combines them to get the complete certificate. The access point toward the global infrastructure is realized through the community distributed certification authority. Figure 1 illustrates an example of key-management architecture. The global trust graph connects five communities. For instance, the community C_1 operates under a hierarchical certification model, the community C_2 operates under a threshold certification model, and the community C_3 operates under a Web-of-trust based certification model. The certification authority is distributed hierarchically by delegation in community C_1 starting from a central server. However, it is distributed partially among certain servers in case of community C_2, and completely distributed among all the community C_3 nodes.

The key management in each community is performed following its own policy. However, in large scale, the key management is performed through the trust propagation among

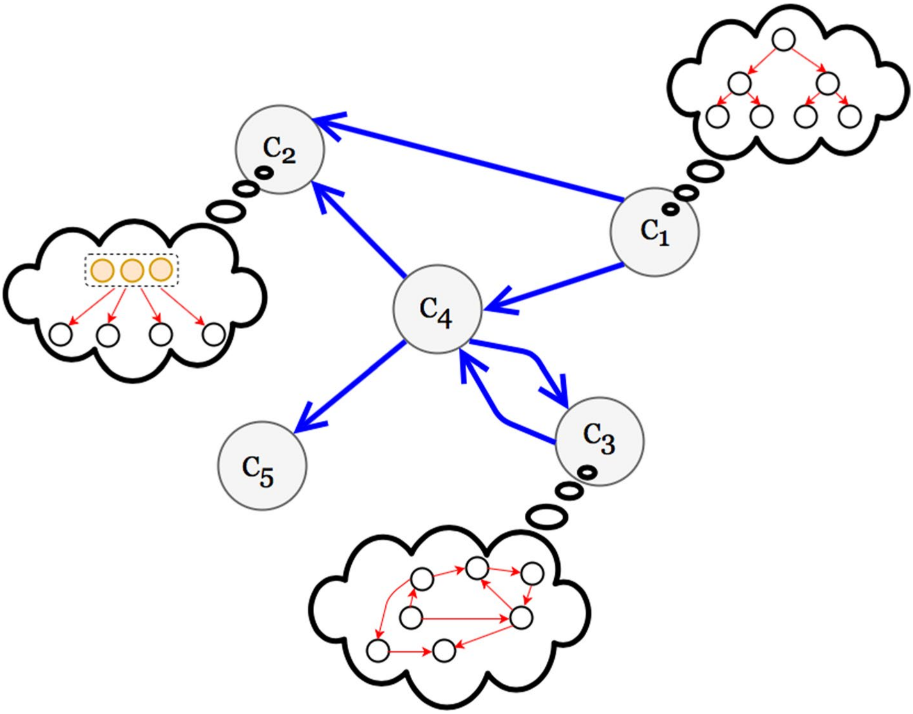


Fig. 1 Communities cohabitation

communities in the global graph of trust. Each community is delegated by an object in order to pass in outside the authentication requests. The delegate objects are selected among those having a high degree of resources and trust connectivity at both inside and outside the delegated community. The delegate object could be a certification authority, a server from a distributed certification authority, or an object among those in the Web-of-trust model based sub-networks. In the latter case, the delegate object could be the one having the highest number of certificates issued for or by it. In the global infrastructure, the delegate objects are in charge to store and distribute their public-key certificates without any centralized server. On the outside of the communities, all the delegate objects have a similar function and no specific operation is assigned to specific delegate objects.

In the context of IoT, several security services are imposed. In the robustness point of view, services such as key independence, forward and backward secrecy have to be ensured. The key management system has to be capable to prevent any device that has left a communication session being able to access to the future exchanged data. In other hand, it has to be capable to prevent a device that joins a new communication session being able to access to the former data. The generated keys must be independent and the disclosure of any key should not affect the other ones. In quality-of-service point of view, the key management system should not introduce extra overhead on the devices in terms of storage, computation and communication. The framework that we propose address all these services, which are designed to be implemented in the application layer. The keys generation and certification are in charge of trustworthy and high-performance devices, elected through a fain-grained process of selection.

3.3 Key Authentication

In the case of hierarchical certification model based sub-networks, the public-key authentication is done by verifying the validity of the certificate holding that key. If the certificate signature is valid, the requester object can trust the corresponding public-key. The requester object can go up towards higher certification authorities in order to verify the complete certificate chain. In the case of Web-of-trust certification model based sub-networks, the public-key authentication done by verifying the validity of the certificate chain starting from the requester object to the requested object. In the case of threshold certification model based sub-networks, the community is supervised by a coalition of servers. The requester object combines the set of requested object partial certificates.

The requester object is not a specific function in the network. Each object O_i is a requester object at each moment it requires to authenticate another object O_j. Therefore, the requester object O_i is the entity in charge to perform the key authentication for that communication session. It carries out this process by collecting and checking the certificate chain of the object O_j. The process of certificate gathering is handled by the delegate community objects, which are in charge to store their community certificates.

The issued certificates are exchanged periodically among the delegate objects. This allows each one holding the global view of the trust graph. Checking the certificate chain that connects both the delegate community objects performs the key authentication. The inter-community authentication is performed between two delegate objects O_i and O_j belonging under two different communities C_i and C_j, respectively. Upon receiving the authentication request, O_j responds by sending its public-key certificate. Then, the delegate object O_i looks up for a trust chain. In the case of absence of a trust relationship, the delegate object O_i ignores the request. Otherwise, O_i authenticates O_j by the signature verification of the certificate chain.

Regarding the application requirements, the public-key certificate recovery process can be implemented in two different ways: proactive or reactive. With the proactive approach, the delegate objects exchange periodically the issued inter-community certificates. Hence, when a delegate object needs to explore the certificate chains leading to another object, the process will be done in its locally. On the other hand, with the reactive approach, the certificate recovery is performed on-demand. When a delegate object needs to authenticate the destination public-key certificate, it collects the appropriate certificate chain through a distributed process. The delegate object requester sends a request to the delegate objects that directly trusts them. Each intermediate delegate object includes its own public-key certificate in the request and perform the same process. Finally, the destination delegate object adds its own public-key certificate and sends back the chain to the delegate object requester.

3.4 Delegate Object Selection

The delegate object selection is an important step to achieve efficiency in terms of trust graph connectivity. It consists to select the optimal component of objects connecting -in terms of trust- all the network communities. Thereby, the main purpose is to extract from the trust graph a strongly connected component that relies all the communities of the network. In addition, a selected object should be representative in regard of its community, trustworthy, and effective in terms of resources. In this context, a trustworthy object is selected among those having a maximum number of certificates issued for them (in-trust). An resource-effective object is selected among those having a maximum number of certificates issued by them (out-trust).

Algorithm 1 Strongly connected components determination process executed by an object $\mathcal{O}_i$.

```
repeat
  $IN_{\mathcal{O}_i} \leftarrow \mathcal{O}_i.id$ ;
  $OUT_{\mathcal{O}_i} \leftarrow \mathcal{O}_i.id$ ;
  %% $IN_{\mathcal{O}_i}$ represents the lowest identity of objects reaching $\mathcal{O}_i$ and $OUT_{\mathcal{O}_i}$ the lowest identity of reachable objects by $\mathcal{O}_i$. %%
  Send $\mathcal{O}_i.id$ to the $\mathcal{O}_i$'s in-trust and out-trust objects ;
  repeat
    for all received message $\mathcal{X}$ from $\mathcal{O}_j$ do
      if $\mathcal{O}_j$ is an in-trust object and $\mathcal{X} < IN_{\mathcal{O}_i}$ then
        $IN_{\mathcal{O}_i} \leftarrow \mathcal{X}$ ;
        Send $\mathcal{X}$ to the $\mathcal{O}_i$'s out-trust objects ;
      end if
      if $\mathcal{O}_j$ is an out-trust object and $\mathcal{X} < OUT_{\mathcal{O}_i}$ then
        $OUT_{\mathcal{O}_i} \leftarrow \mathcal{X}$ ;
        Send $\mathcal{X}$ to the $\mathcal{O}_i$'s in-trust objects ;
      end if
    end for
  until all the messages reception
until $IN_{\mathcal{O}_i} = OUT_{\mathcal{O}_i}$
%% $IN_{\mathcal{O}_i} = OUT_{\mathcal{O}_i}$ means that $\mathcal{O}_i$ and $IN_{\mathcal{O}_i}$ are in the same strongly connected component.%%
```

We model the trust graph by an oriented graph $G(O, R)$, where O represents the set of objects and R the set of trust relationship between the objects. With the aim to select the delegate objects, the following steps are executed. The first step consists of the determination of strongly connected components in the trust graph. Two objects O_i and O_j of the trust graph would be in the same strongly connected component if there is at least one path connecting O_i to O_j and inversely. In order to achieve this operation, we propose a distributed version of the approach developed in [16]. The distributed approach is presented by Algorithm 1. The second step consists of the selection of the strongly connected component allowing each community being represented by at least an object. These objects will be the candidates in the delegate object selection. In case of several candidate objects for a given community, the delegate will be the one object O_i that maximizes F_i, such as

$$F_i = \begin{cases} d_i + \frac{1}{|d_i^+ - d_i^-|}, & d_i^+ \neq d_i^- \\ d_i, & \text{otherwise,} \end{cases} \tag{1}$$

where d_i, d_i^+ and d_i^- represent the object O_i's degree, out-degree and in-degree, respectively. Maximizing F_i consists of maximizing $d_i = d_i^+ + d_i^-$. The out-degree d_i^+ reflects the resource capability (computation and storage) and the in-degree d_i^- reflects the degree of trustworthiness.

Let's take the illustrative example depicted by Fig. 2. The trust graph connects three communities, namely $C_1 = \{1, 2, 3, 4\}$, $C_2 = \{5, 6, 7\}$ and $C_3 = \{8, 9, 10, 11\}$. By executing the first step of strongly connected components determination, we lead as illustrated in Fig. 3 to three components: $SCC_1 = \{1, 2, 5, 6, 7, 8, 9, 10, 11\}$, $SCC_2 = \{3\}$, $SCC_3 = \{4\}$. The strongly connected component SCC_1 is the one that represents all the

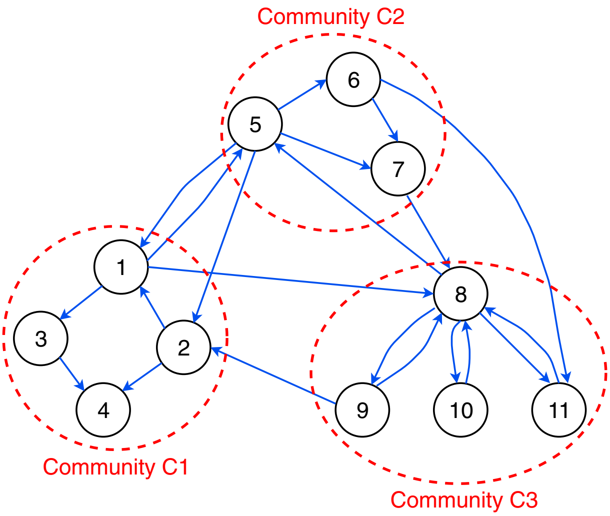


Fig. 2 Illustrative example

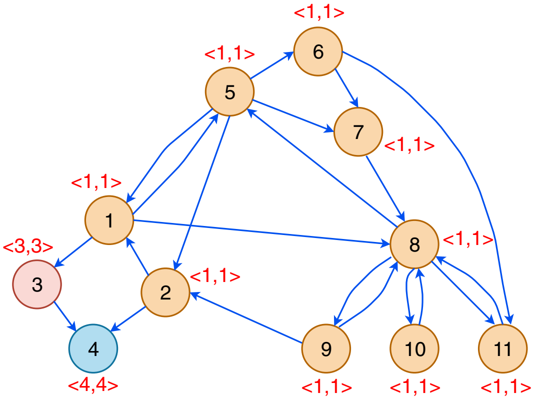


Fig. 3 Strongly connected components determination ($< IN_{\mathcal{O}_i}, OUT_{\mathcal{O}_i} >$)

communities. The objects 1 and 2 represent the community C_1, the objects 5, 6 and 7 represent the community C_2, and the objects 8, 9, 10, and 11 represent the community C_3. On the other hand, the other strongly connected components, SCC_2 and SCC_3 allow just the representation of community C_1. In Table 1, we present the objective function evaluation regarding each community. The object 1 is selected to act as delegate for the community C_1, the object 5 is selected to act as delegate for the community C_2, and the object 8 is selected to act as delegate for the community C_3. The final structure is illustrated in Fig. 4.

Our work does not focus on energy in the delegate object selection. We believe this is not a strong issue in the context of IoT. The proposed framework is not designed to operate with hostile, isolated and/or completely autonomous equipment as considered in Wireless Sensor Networks. The IoT's devices (such as smartphones, connected watches, etc.) are mostly equipped with rechargeable batteries, which let them permanently operational. That's why we are focusing mainly on resources (computation and storage) and trustworthiness in the delegate object selection.

Fig. 4 Delegate object selection

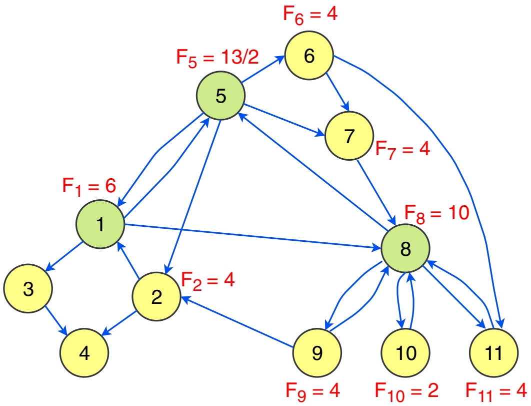


Table 1 Delegate objects determination

Community	i	d_i	d_i^+	d_i^-	F_i	Selected object
C_1	1	5	2	3	6	1
	2	4	2	2	4	
C_2	5	6	2	4	$\frac{13}{2}$	5
	6	3	1	2	4	
	7	3	2	1	4	
C_3	8	9	5	4	10	8
	9	3	1	2	4	
	10	2	1	1	2	
	11	3	2	1	4	

4 Performance Evaluation

In this section, we present the simulation environment, evaluate the performances of our proposal under several metrics, and finally compare it to the literature.

4.1 Simulation Environment and Methodology

The numerical simulations, which validate the proposed framework, have been developed under a Java virtual machine (JVM), JRE 9.0.4, developed on a machine characterized by 2.30 GHz Intel® Core™ i3-2350M Processor, 4 GB RAM, 512 GB SSD, and 64-bit Windows operating system. We have simulated the proposed public-key management infrastructure via 750 objects managed by three autonomous communities. Each of them operates under a specific public-key certificate management policy: (1) hierarchical certification, (2) Web-of-trust based certification, and (3) threshold certification. The alliance of these policies is achieved through a global infrastructure. Thus, the latter consists of a set of immobile objects, randomly scattered in the deployment area of 1000 m^2. These objects present different features in memory and processing power, we have considered for this purpose two types of objects, namely, ordinary and delegate objects, according to the size of their memory and the rate of their processor. The overall simulation parameters are summarized in Table 2.

The performance parameters retained during the experiment of the proposed framework are as follows: (1) Storage cost: it is defined as the memory space exploited by an object during the different phases of the proposed framework for storing both keys and cryptographic parameters; (2) Communication cost: it is the measure of the communication load between objects based on the packet size exchanged between them; (3) Response time: is the time elapsing between the end of the request addressed to the proposed key management infrastructure and the beginning of the response. This is of paramount importance since it measures the reactivity of the proposed framework.

4.2 Results

In this subsection, we present and discuss the obtained simulation results from the aforementioned performance metrics. The results presented hereafter are obtained by varying the network size from 75 to 750 objects.

4.2.1 Storage Cost

Figure 5 shows the average storage cost variation using RSA cryptosystem for proactive protocol corresponding to the network size. From the plot, we note that the storage cost increases with the network density, which is quite logical. This is due to the fact that the objects of each community, which operates under a specific public-key certificate management policy, stores a significant number of both keys and cryptographic parameters. Consequently, each time we increase the key size of the RSA cryptosystem, this results in increased storage cost. In the ongoing, we present the storage cost obtained by the proposed framework with a very lightweight cryptosystem to see the gains of changing the cryptosystem, which appears.

Figure 6 shows the average storage cost variation using ECC cryptosystem for proactive protocol corresponding to the network size. From this plot, we can also see that the storage cost increases with the increase in the network size. Indeed, the storage cost generated by the ECC based version is lower than that obtained with RSA based version. Because ECC

Table 2 Simulation parameters

Parameter	Value
Deployment area	1000 m^2
Deployment type	Random
Number of communities connected by the trust graph	3
Number of delegate objects	3
Network size	75–750 objects
Number of malicious objects	5–20 objects
Public-key RSA certificate size	103–268 Bytes
Public-key ECC certificate size	89–147 Bytes
Hash function used	SHA-1
Digital signature algorithm used	ECDSA or RSA
Number of executions	25 iterations

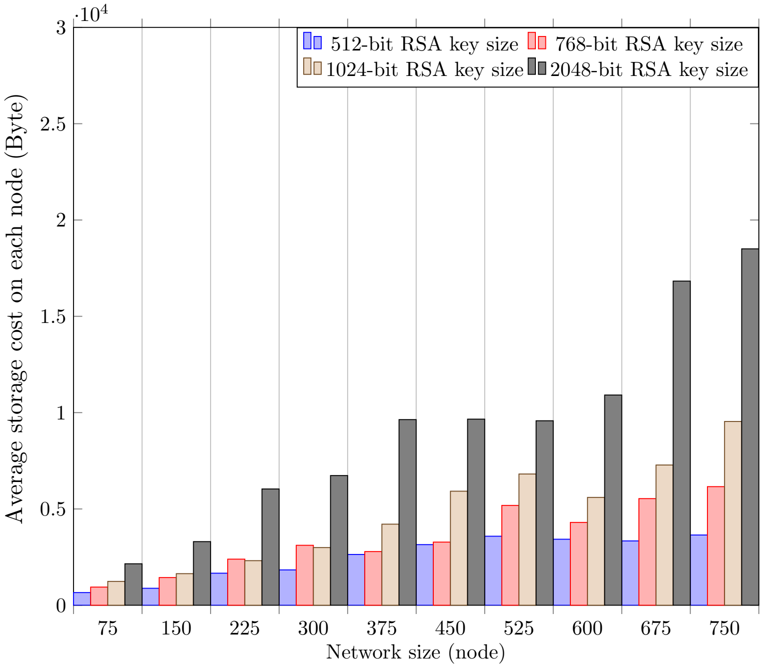


Fig. 5 Average storage cost using RSA cryptosystem versus increasing network size

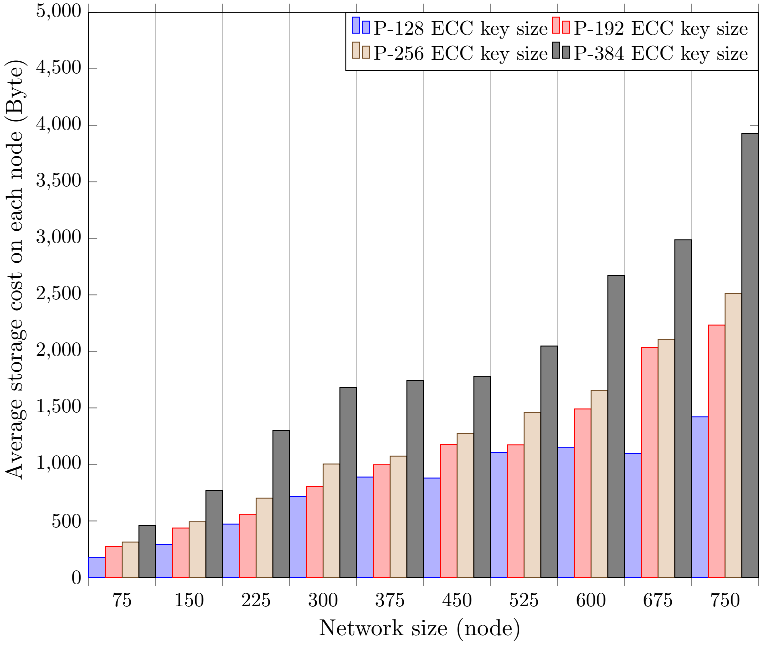


Fig. 6 Average storage cost using ECC cryptosystem versus increasing network size

requires smaller keys to provide equivalent security offered by RSA cryptosystem, thereby reducing storage requirements. In the continued, we observe the impact of the reactive and proactive protocol on the storage cost using the ECC cryptosystem with a P-256 bit key size.

Figure 7 shows the average storage cost variation using ECC cryptosystem for both proactive and reactive protocols corresponding to the network size. From the plot, we can see that the more the network size increases, the more the storage load produced by the two protocols increases. Indeed, the reactive protocol considerably reduces the storage load compared to the proactive one. This is expected because of the periodic exchange of intercommunity certificates issued between delegate objects, which obliges each delegate object to periodically store and update certificate data so that the delegate objects can explore certificate chains leading to another network object. Therefore, if a delegate object needs to explore one certificate chain leading to another, it looks at the certificate table stored in its locality.

Faulty or unreliable objects can be introduced into the network from intruders to spy, inject or reuse data to cause any system malfunction. These anomalies can be produced by any type of malicious cryptographic attacks, which can negatively affect sensitive system information, and data integrity. Likewise, these malicious objects can make each network community incoherent in their decisions, and they can cause significant damage to the system if they take control of it. During further, we evaluate the average size of non-stored keys as reported in Fig. 8 by varying the malicious object number from 5 to 20 on the whole network using a topology consisting of 750 objects. During data flow, the malicious objects are deployed in the area of interest randomly, where the probability of affecting is less than 1.

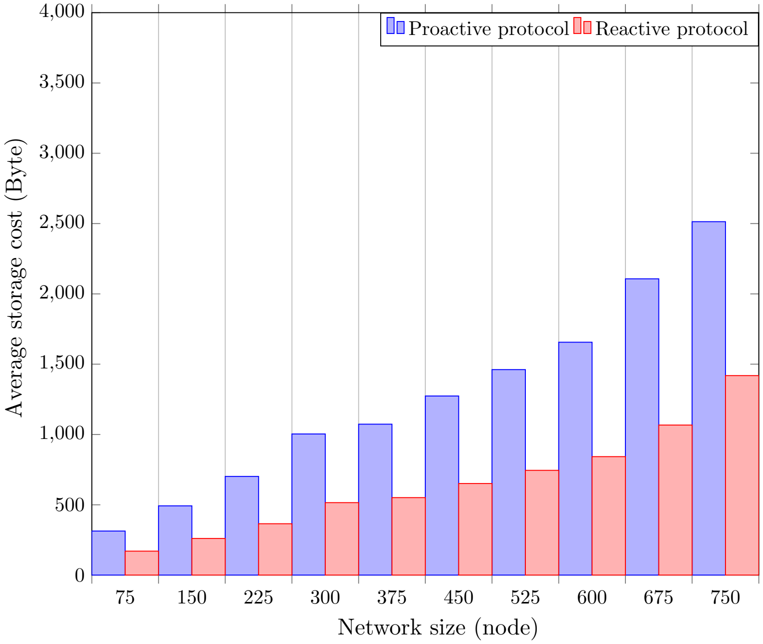


Fig. 7 Average storage cost using ECC cryptosystem for proactive and reactive protocols versus increasing network size

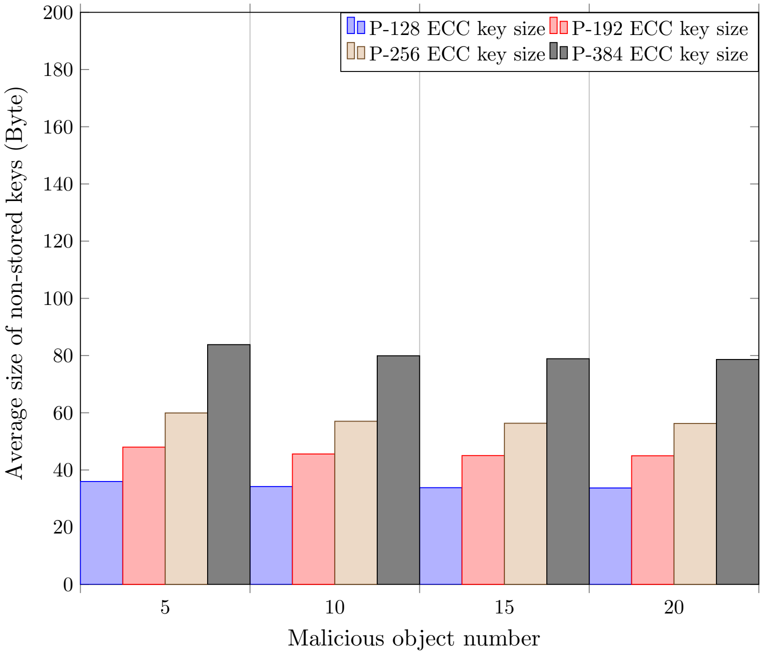


Fig. 8 Average size of non-stored keys using ECC cryptosystem versus increasing malicious objects

Figure 8 shows the variation of the average size of non-stored keys using ECC cryptosystem for proactive protocol in function of the malicious object number. Indeed, the results show that the average size of non-stored keys decreases slightly by increasing the malicious object number throughout the whole network. Nevertheless, the proposed framework distinguishes between benign (trusted) and malicious objects within the entire network. This is because isolating the suspected objects prevent them from participating in all activities performed throughout the network.

4.2.2 Communication Cost

Figure 9 show the communication cost variation using RSA cryptosystem for proactive protocol corresponding to the network size. From the results, we note that the more the network size increases, the more the communication load of the objects increases. This is due to the network density, which makes the object number of each community, becomes more and more significant and therefore the transmission load becomes important too. As well, with 2048-bit RSA key size, the communication cost is quite significant compared to that obtained with other RSA key sizes. 2048-bit RSA key size allows to assure better security than other RSA key sizes for as long as possible, and therefore the communication load is enormous. In what follows, we will also present the communication cost obtained with the version based on the ECC cryptosystem.

Figure 10 shows the communication cost variation using ECC cryptosystem for proactive protocol corresponding to the network size. Regarding the obtained results, the version of ECC cryptosystem offers better performances, while generating a few

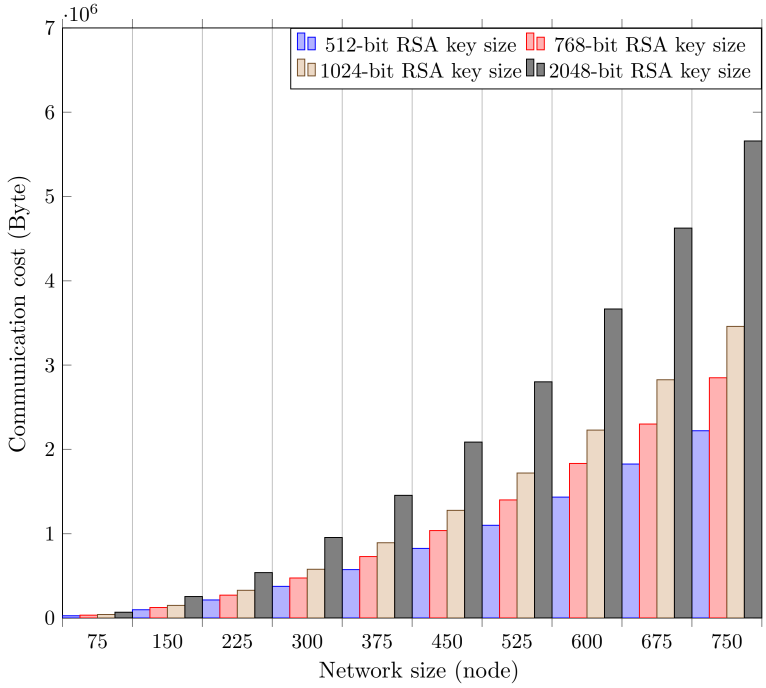


Fig. 9 Communication cost using RSA cryptosystem versus increasing network size

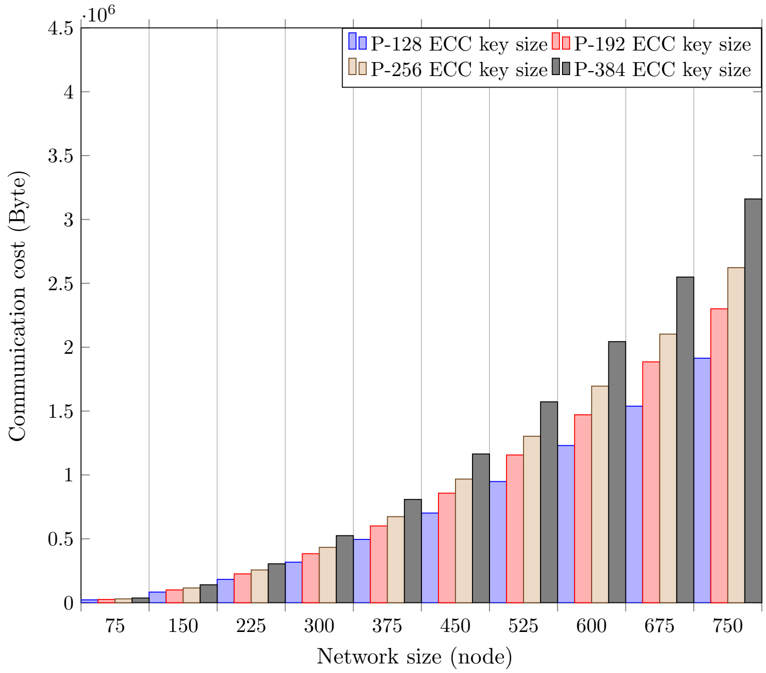


Fig. 10 Communication cost using ECC cryptosystem versus increasing network size

communication loads with much shorter keys compared to the version based on the RSA cryptosystem. This comes down to the smallest keys needed with the ECC cryptosystem-based version, which reduces transmission requirements. Thereafter, we discuss and examine the impact of reactive and proactive protocol on the communication cost using the ECC cryptosystem with a P-256 bit key size.

Figure 11 shows the communication cost variation for both proactive and reactive protocols corresponding to the network size. As shown, the communication cost in both protocols increases with increasing network size. However, the communication load caused by network overload is too high in the proactive protocol compared to the reactive one. This is due not only to periodically exchanging inter-community certificates delivered between delegate objects, but also to regularly updating certificate tables leading to any objects in the network. To see the influence of malicious objects on the communication cost, we computed this quantity in the ongoing, according to the proactive principle, while using the ECC cryptosystem.

Figure 12 shows the communication cost variation using ECC cryptosystem for proactive protocol in function of the malicious object number in the whole network. From the plot, we can see a minor decrease in the transmission load due to the increase in the malicious object number on the network. Although the malicious object number has no impact on the communication cost, the obtained results using ECC with P-384 bit key size are a little bit high compared to others as ECC cryptosystem with P-384 bit key provides a higher security level.

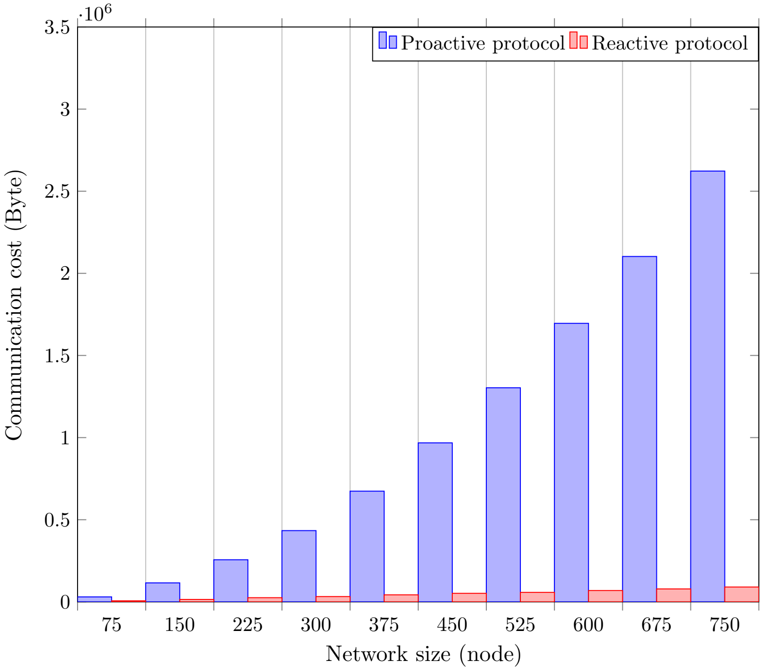


Fig. 11 Communication cost using ECC cryptosystem for proactive and reactive protocols versus increasing network size

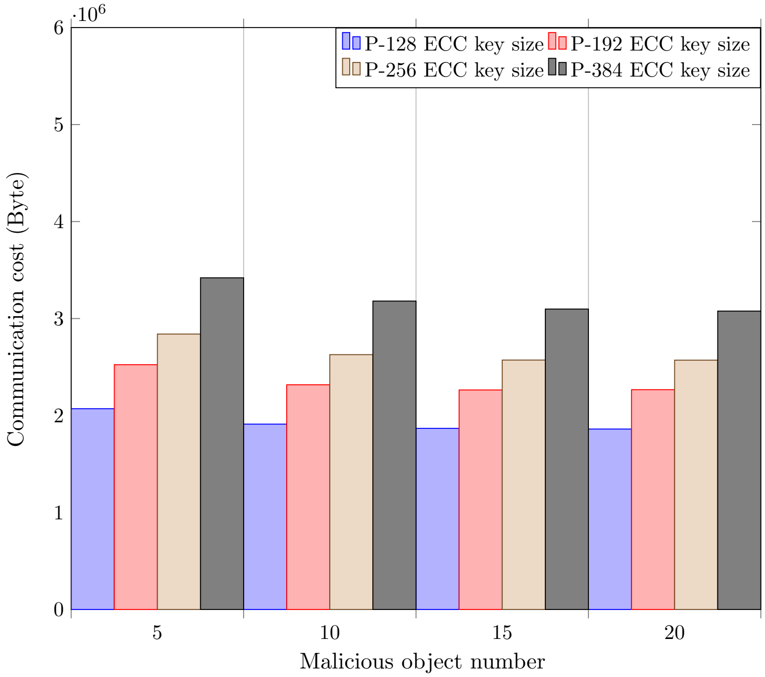


Fig. 12 Communication cost using ECC cryptosystem versus increasing malicious objects

4.2.3 Response Time

Figure 13 shows the response time variation using RSA cryptosystem for proactive protocol corresponding to the network size. In fact, we observe that the response time increases with the network density, which is quite logical as the number of messages exchanged between the objects rises due to inter and intra-community communications. The obtained results indicate that whatever the RSA key size used, the response time increases. Indeed, even if the response time provided by using 2048-bit RSA key is greater than that elapsed by using the other RSA keys, but this ensures sufficient security for a long-lasting control message. To see the advantage of using the ECC on the interactive applications, during subsequent, we evaluate the response time of the proposed framework with the ECC cryptosystem.

Figure 14 shows the response time variation using ECC cryptosystem for proactive protocol corresponding to the network size. Regarding the obtained results, the version based on ECC cryptosystem demonstrates better results compared to RSA. It generates smaller keys and takes less time in cryptographic operations. Consequently, the shorter keys make the ECC-based version a very attractive option for the objects with limited storage capacity and processing power. In what follows, we present the response time, which is evaluated for both proactive and reactive protocols by using the ECC cryptosystem with a P-256 bit key size.

Figure 15 shows the response time variation using ECC cryptosystem for both proactive and reactive protocols corresponding to the network size. We note that the total response time required by the proactive protocol is less than the reactive one. In the

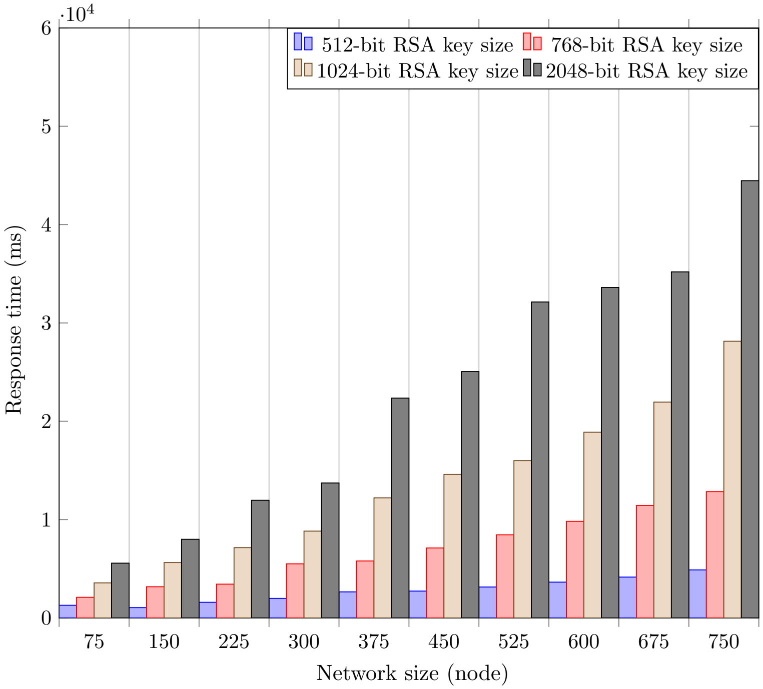


Fig. 13 Response time using RSA cryptosystem versus increasing network size

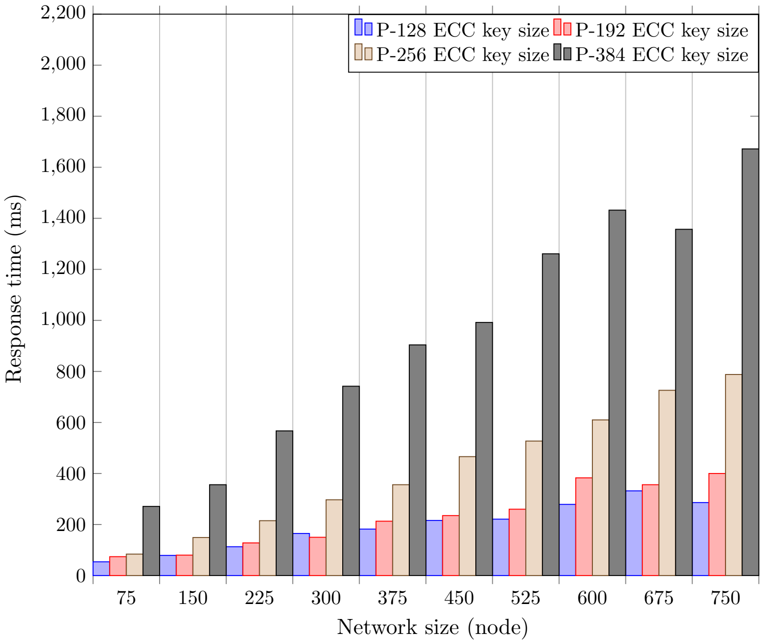


Fig. 14 Response time using ECC cryptosystem versus increasing network size

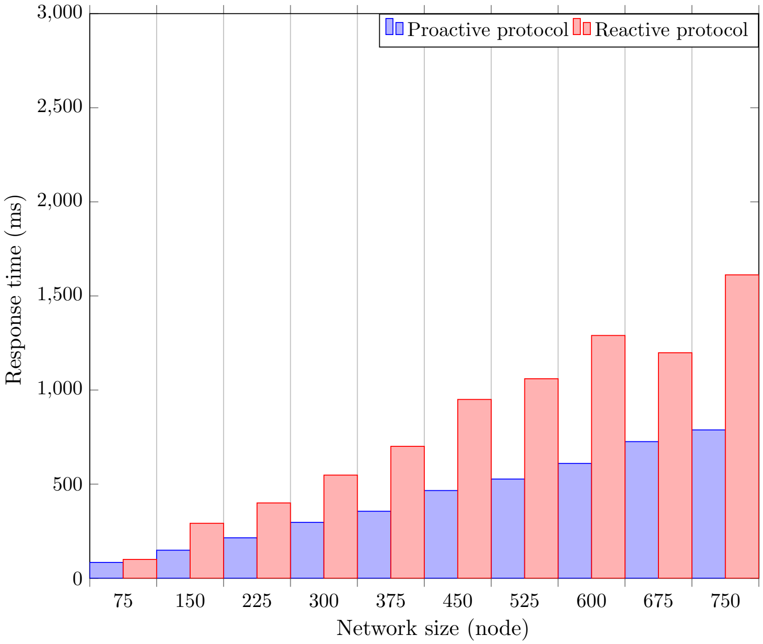


Fig. 15 Response time using ECC cryptosystem for proactive and reactive protocols versus increasing network size

proactive protocol, when a delegate object explores a certificate chain leading to another object, the process is done locally. However, in the reactive protocol, when a delegate object needs to authenticate a destination public-key certificate; it collects the appropriate certificate chain through a distributed process. Consequently, the process of exploring certificate chains diminishes the interactive application performance. In the continued, we evaluate the response time with malicious objects presence to observe the influence of these objects on the whole network consisting of 750 objects. The proactive principle is used under the ECC cryptosystem with a P-256 bit of key size.

Figure 16 shows the variation of the malicious object detection time necessary to detect them across the network using ECC cryptosystem in function of the malicious objects. From the graph, it should be noted that the detection time increases with the increase in the malicious object number in the three communities of the network. We also observe that the detection time using an ECC P-384 key size is greater, regardless of the size of three other ECC used keys.

4.3 Comparison

In this subsection, we present the obtained results of comparison of our proposal with PKI-4IoT protocol [13]. We observe the average size of non-stored keys, communication cost, and malicious object detection time variations of both protocols depending on the increase of malicious objects. The proactive principle is used under the ECC cryptosystem with a P-256 bit of key size.

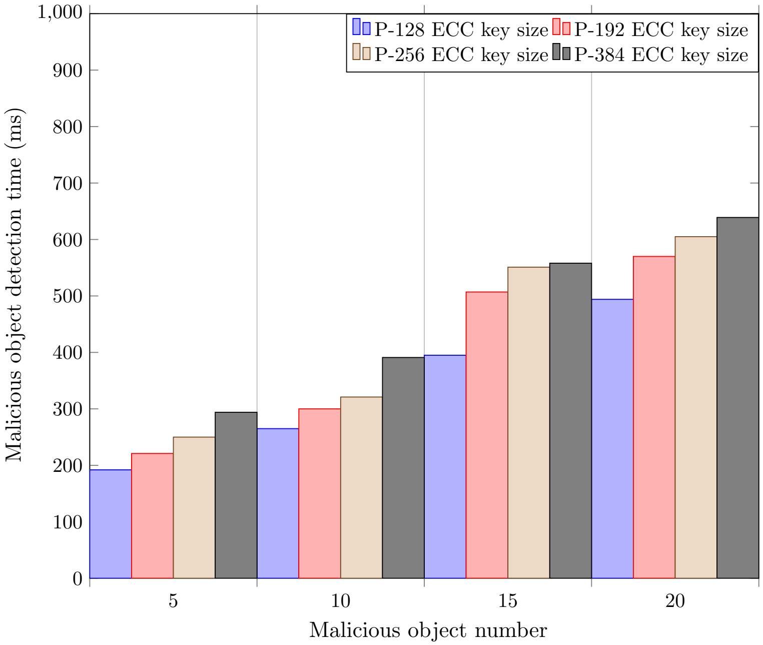


Fig. 16 Malicious object detection time using ECC cryptosystem versus increasing malicious objects

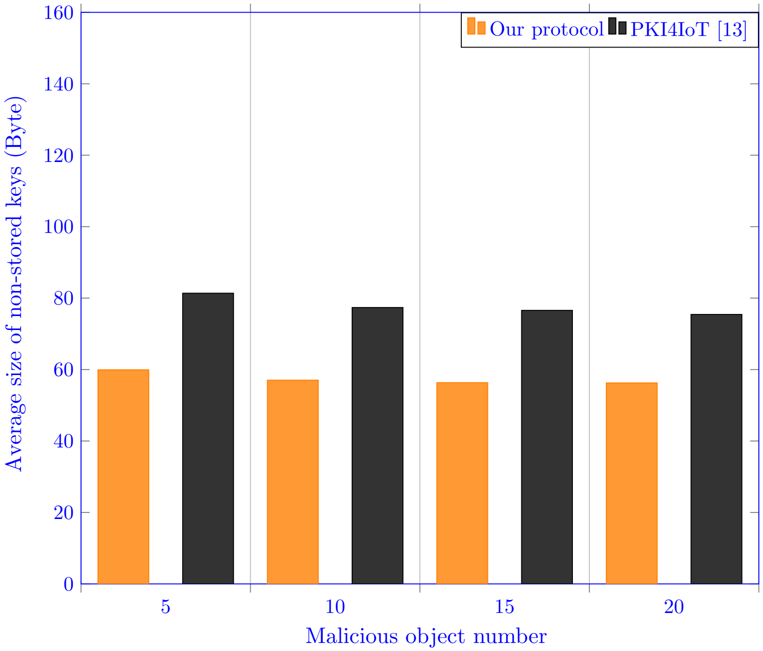


Fig. 17 Average size of non-stored keys using ECC cryptosystem versus increasing malicious objects

Figure 17 shows the average size of non-stored keys in function of malicious object number. From this plot, we note that the average size of non-stored keys decreases slightly by increasing the malicious object number in the whole network for both protocols. In fact, the average size of non-stored keys in the proposed framework, is lower than that of PKI4IoT. Besides, the proposed framework distinguishes more effectively between trusted objects and malicious objects across the network compared to PKI4IoT.

Figure 18 shows the communication cost variation in function of malicious object number. From the results, we observe the communication cost is a little high in PKI4IoT compared to the proposed framework. Furthermore, there is a slight decrease in the transmission load due to the increased number of malicious objects on the network for both protocols. However, the increase of malicious object number influences PKI4IoT much more than the proposed framework. This is due to the effectiveness of the security strategy adopted in the proposed protocol, which detects and eliminates the malicious objects.

Figure 19 shows the malicious object detection time in function of malicious object number. As shown, the malicious object detection time in both protocols increases with increasing the malicious object number. Regarding the obtained results, the proposed framework can achieve a high identification success time against a large number of malicious objects across the network. This is explained by the limited communication load between the objects by limiting the number of requests exchanged between them and the absence of expensive arithmetic operations. Unlike PKI4IoT, which relies on many operations to provide the end-to-end security between recipient IoT devices and certificate authorities.

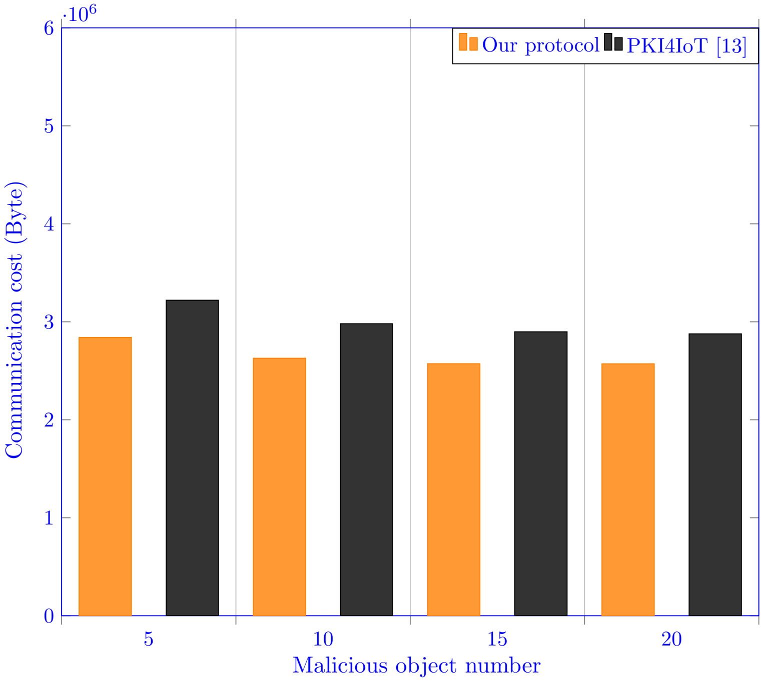


Fig. 18 Communication cost using ECC cryptosystem versus increasing malicious objects

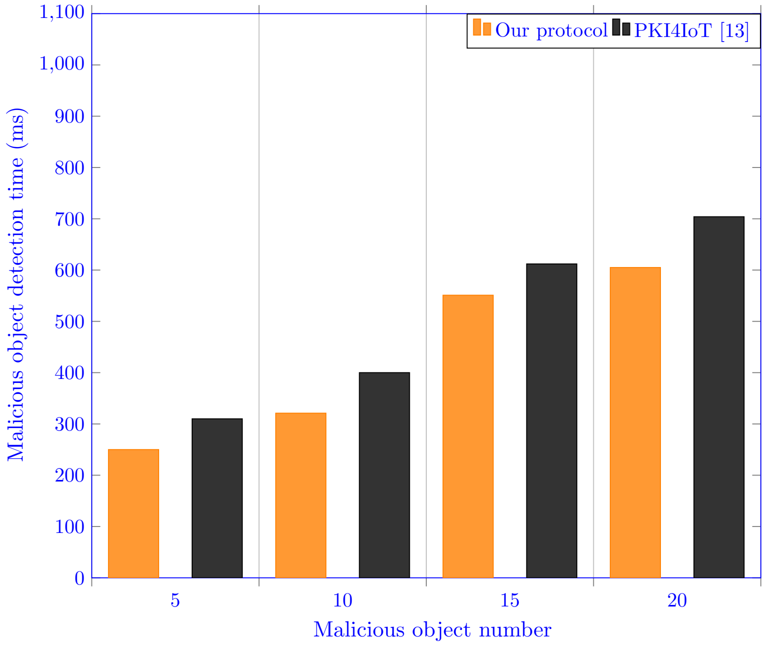


Fig. 19 Malicious object detection time using ECC cryptosystem versus increasing malicious objects

5 Conclusion

In this paper, we have proposed a reliable and adaptive distributed public-key management infrastructure for the IoT. This framework optimizes the resource in managing public-key certificates for homogeneous and heterogeneous environments. The main purpose behind this study was to design an effective security mechanism in the IoT context and therefore offer reliable end-to-end network security services. The proposed framework allows several communities of objects to communicate. The key management in each community is performed following its own policy, such as hierarchical certification, Web-of-trust based certification, and threshold certification. Each community is delegated by an object in order to pass outside authentication requests. The delegate object is selected among those having a high degree of resources and trust connectivity at both inside and outside the delegated community. In this regard, simulations have been carried out with comparison to the literature so as to assess the performances of our proposal. The obtained results are very encouraging by reducing response time, transmission and storage loads.

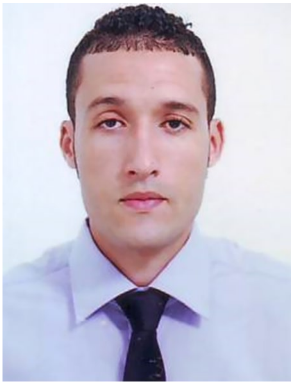
Acknowledgements This work was carried out in the framework of the research activities of the LARI laboratory (Laboratory of Research in Informatics), which is affiliated to the Faculty of Electrical Engineering and Computer Science of the University of Mouloud Mammeri, Tizi-Ouzou, Algeria. It was the result of collaboration with the LIMED (laboratory of Medical Computing) laboratory, which is affiliated to the faculty of exact sciences of the university of Bejaia, Algeria and the LIGM laboratory of the University of Gustave Eiffel, France. This work has been sponsored by the General Directorate for Scientific Research and Technological Development, Ministry of Higher Education and Scientific Research (DGRSDT), Algeria.

References

1. Shafagh, H. (2013). *Leveraging public-key-based authentication for the Internet of Things*. Thesis, RWTH Aachen University, Germany.
2. Nafi, M., Bouzefrane, S., & Omar, M. (2020). Matrix-based key management scheme for IoT network. *Ad Hoc Networks, 97,* 102003.
3. Rafaeli, S., & Hutchison, D. (2003). A survey of key management for secure group communication. *ACM Computing Surveys (CSUR), 35*(3), 309–329.
4. Lavanya, S., & Usha, M. (2017). A survey on key management in Internet of Things. In *The proceedings of the international conference on intelligent computing systems* (pp. 572–580).
5. Sciancalepore, S., Capossele, A., Piro, G., Boggia, G., & Bianchi, G. (2015). Key management protocol with implicit certificates for IoT systems. In *The proceedings of the 2015 workshop on IoT challenges in mobile and industrial systems, ACM IoT-Sys'15* (pp. 37–42).
6. Roman, R., Alcaraz, C., Lopez, J., & Sklavos, N. (2011). Key management systems for sensor networks in the context of the Internet of Things. *Computers and Electrical Engineering, 37*(2), 147–159.
7. Dahshan, H. (2016). An elliptic curve key management scheme for Internet of Things. *International Journal of Applied Engineering Research, 11*(20), 10241–10246.
8. Tsai, I. C., Yu, C. M., Yokota, H., & Kuo, S. Y. (2017). Key management in Internet of Things via kronecker product. In *The proceedings of the IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)* (pp. 118–124).
9. Ben Saied, Y., Olivereau, A., Zeghlache, D., & Laurent, M. (2014). Lightweight collaborative key establishment scheme for the Internet of Things. *Computer Networks, 64,* 273–295.
10. Veltri, L., Cirani, S., Busanelli, S., & Ferrari, G. (2013). A novel batch-based group key management protocol applied to the internet of things. *Ad Hoc Networks, 11*(8), 2724–2737.
11. Wong, C. K., Gouda, M., & Lam, S. (2000). Secure group communications using key graphs. *IEEE/ACM Transactions on Networking, 8*(1), 16–30.
12. Gu, H., & Potkonjak, M. (2018). Efficient and secure group key management in IoT using multistage interconnected PUF. In *The proceedings of the international symposium on low power electronics and design* (pp. 1–6).
13. Hoglund, J., Lindemer, S., Furuhed, M., & Raza, S. (2020). PKI4IoT: Towards public key infrastructure for the Internet of Things. *Computers and Security, 89,* 1–16.
14. Braeken, A., Liyanage, M., & Jurcut, A. D. (2019). Anonymous lightweight proxy based key agreement for IoT (ALPKA). *Wireless Personal Communications, 106*(2), 345–364.
15. Tabassum, T., Hossain, S. A., Rahman, M. A., Alhamid, M. F., & Hossain, M. A. (2020). An efficient key management technique for the Internet of Things. *Sensors, 20*(7), 20–49.
16. Guerrieri, A. (2016). *Distributed graph algorithms. Presentation slides*. Trento: University of Trento.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Samia Belattaf is assistant professor and research member of LARI Laboratory at the University of Mouloud Mammeri of Tizi Ouzou (UMMTO), Algeria. She was deputy head of computer science department in charge of post-graduate studies at the same university from November 2017 to November 2020. She received her Magister degree in Computer Science from the University of Bejaia (Algeria) in 2005. She got her engineer diploma in computer science from the University of UMMTO in 2003. Her research interests include security in Internet of Things, wireless sensor networks and ad-hoc networks.



Mohamed Mohammedi received his Ph.D, Master, and license degrees in computer science from the University of Bejaia, in 2018, 2013, and 2011 respectively. He is a member of the laboratory of Medical Computing (LIMED). His research interests include routing and security in the Internet of things, wireless body area networks, vehicular ad hoc networks, wire-less sensor networks, and smart cities. He is also looking to the challenging issues associated to biomedical signal processing, and driver's drowsiness detection technologies.

Mawloud Omar is member of LIGM laboratory and associate professor at ESIEE Paris of the university of Gustave Eiffel (France). He got his PhD and Magister degrees in computer science from the university of Bejaia (Alge-ria) with collaboration with the university of technology of Compiegne (France). He was Senior Researcher at IRT SystemX (France). Before, he worked for several years as lecturer and researcher at the university of technology of Compiegne and the university of Bejaia. He published sever-al works in journals and conferences and participates actively in national and European projects. His research activities revolve around networking and cybersecurity. He is looking to the challenging issues related in particular to the Internet of things, 5G/6G networks, connected vehicles and industrial environments.

Rachida Aoudjit received her Ph.D. in Computer Science from Mouloud Mammeri University of Tizi Ouzou (UMMTO), Algeria. She is currently a re-search member at LARI laboratory and professor at UMMTO. Her current research areas cover Mobile and Wireless networks, Vehicular networks, Security, IoT and Smart Agriculture.