R. Rios, D. Nuñez, and J. Lopez, "Query Privacy in Sensing-as-a-Service Platforms", *32nd International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2017)*, IFIP Advances in Information and Communication Technology (AICT) vol. 502, pp. 141154, 2017.
http://doi.org/10.1007/978-3-319-58469-0_10
NICS Lab. Publications: https://www.nics.uma.es/publications

Query Privacy in Sensing-as-a-Service Platforms

Ruben Rios, David Nuñez, and Javier Lopez

Network, Information and Computer Security (NICS) Lab
Computer Science Department
University of Málaga, Spain
{ruben,dnunez,jlm}@lcc.uma.es

Abstract. The Internet of Things (IoT) promises to revolutionize the way we interact with the physical world. Even though this paradigm is still far from being completely realized, there already exist Sensing-as-a-Service (S^2aaS) platforms that allow users to query for IoT data. While this model offers tremendous benefits, it also entails increasingly challenging privacy issues. In this paper, we concentrate on the protection of user privacy when querying sensing devices through a semi-trusted S^2aaS platform. In particular, we build on techniques inspired by proxy re-encryption and k-anonymity to tackle two intertwined problems, namely query privacy and query confidentiality. The feasibility of our solution is validated both analytically and empirically.

1 Introduction

The interconnection of computational and sensing devices to the Internet is expected to transform every single aspect of our lives. This novel paradigm, already known as the Internet of Things (IoT) [16], brings about a whole set of innovative services and Sensing-as-a-Service (S^2aaS) [22,23] platforms play a fundamental role as they allow querying IoT devices. In this model, the sensing devices deployed by companies, administrations or citizens can be queried through a sensing server, which acts as gateway, as shown in Fig. 1. This model is already a reality and there are some companies, like Amazon (cf., AWS IoT platform [1]), which are offering the infrastructure necessary for delivering these sort of services.

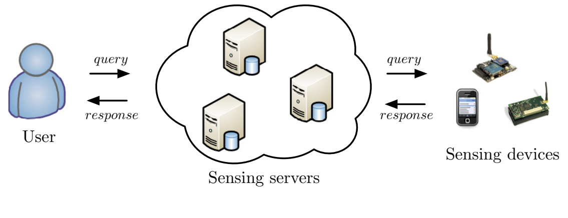


Fig. 1. Sensing-as-a-Service Platform

While this model offers great opportunities to both industry and citizens, it also poses serious privacy risks. In particular, there is the possibility of exposing user interests to honest-but-curious sensing servers since they act as intermediaries for the sensing devices. Therefore, it is paramount to provide the users of these platforms with mechanisms that allow them to remain unlinkable from the sensing devices they are interested in querying. This is precisely the main objective of this paper, to provide a solution to query privacy in Sensing-as-a-Service scenarios where the access to the readings of sensing devices is managed by a semi-honest sensing server, which may be interested in profiling the users of the platform.

One may think that such a solution can be achieved with traditional public key cryptography but there are some notable limitations to this approach. First, the user needs to be aware of the public key of every single sensing device, which raises evident usability and scalability issues. Moreover, it is necessary for the user to check the status of the public keys, such as whether or not they have been revoked. Moreover, if a query is intended for multiple (or all) sensing nodes, the user has to query the sensors individually. This not only implies more energy and bandwidth waste but it is also highly advisable to hide these issues from the user, so as to facilitate the development and adoption of S^{2}aaS platforms.

The main contribution of this paper is the QPSP (Query Privacy for Sensing Platforms) protocol. The proposed protocol is based on proxy re-encryption and k-anonymity techniques to provide both query confidentiality (i.e., hiding the query itself and the sensed data) and query privacy (i.e., hiding the nodes replying to the queries) in semi-trusted S^{2}aaS platforms. The proposed scheme is, to the best of our knowledge, the first solution to exploit these notions to protect query privacy issues in sensing scenarios.

This rest of this paper is organized as follows. Section 2 analyzes previous papers describing query privacy solutions in related domains. Next, in Section 3 we provide a detailed description of the problem addressed by the QPSP protocol and identify some general assumptions that are applicable to the rest of the paper. The various phases of the QPSP protocol are described in Section 4 and its privacy guarantees are analyzed in Section 5. In addition, we experimentally evaluate the feasibility of our solution with current sensing devices in Section 6. Finally, Section 7 presents the conclusions of the paper and outlines some potential lines of future research.

2 Related Work

Most of the research in query privacy has been done in the area of Wireless Sensor Networks. Although this problem can be trivially solved by making all sensor nodes reply to every query, it also imposes severe energy requirements on the sensor nodes. Consequently, some authors have striven to find the right balance between privacy protection and energy consumption. The authors in [13] propose reducing the amount of traffic generated by using data-aggregation. This solution is only suitable for a particular type of query. A more general approach is presented in [7], where the authors propose transmitting bogus queries to the network to hide the destination of real queries. Instead of sending bogus queries, the authors in [12] propose hiding the recipient of the queries by sending them on a particular path of nodes that contains the actual destination. Unfortunately, the user needs to define the path, which is impractical for large-scale sensor networks.

A completely different approach is to unlink the original data source from the current location of the data, which is mostly achieved by having two types of nodes: sensing and storage nodes. The authors in [14] propose having several data replicas so that user queries are forwarded to a number of random points with the hope that the query arrives at some of them. PriSecTopK [19] concentrates on enabling top-k querying with the help of order-preserving encryption. A major limitation of this scheme is the need for shared secrets between the user and each sensing node. In addition, some papers [8,24] have considered the problem of privacy-preserving range queries. Basically, the idea behind these schemes is to transform data and queries into special codes that can be processed by storage nodes without leaking information in the case they are compromised. All these solutions restrict the user to a particular type of query.

Finally, some effort has been made to protect query privacy in urban sensing scenarios. The approach followed in [11] is again based on data replication and storage devices. Their scheme is complemented with bogus replies to hide the data sources. A noteworthy difference with respect to our work is the adversarial model, which is an external attacker located at the edge of the network.

3 Problem Definition

This section deals with the definition of the problem. First, we present a general description of the system and then we illustrate the capabilities of the adversarial model. This section introduces the main assumptions applicable to the rest of the paper.

3.1 System Model

The system we are aiming for is composed of a substantial number of sensing devices which can be queried through a sensing server. Without loss of generality, we assume that the sensing devices are organized into clusters, where one node acts as the head or leader of each cluster. However, it is also reasonable to assume a more general model where the sensing server provides access to several sensing networks, like the one depicted in Fig. 2, which is a typical configuration in fully-fledged IoT scenarios. Note that the secure selection of cluster heads is beyond the scope of this paper and the interested reader is referred to [21] for a survey.

Moreover, there are at least $n > 1$ cluster heads in the sensing network. The cluster heads are considered to be able to communicate with one another and also with the sensing server. In the more general model, the cluster heads of each of the sensing networks are also necessarily interconnected. In either case,

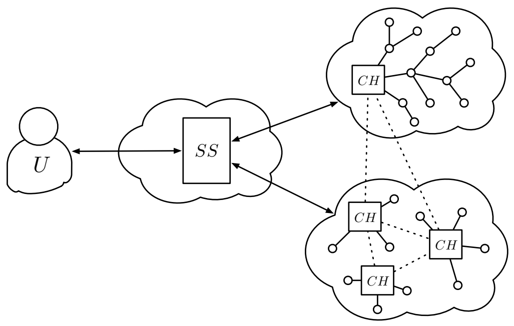


Fig. 2. General System Model

the communication with other cluster heads can be done directly through the Internet or by using the routing information available to them after the execution of a secure clustering protocol. The routing information also allows the cluster heads to determine how to reach any sensing device in the network.

Finally, we focus on a scenario where the readings of the sensing devices are publicly available to anyone willing to access them. This is, for example, the case of a Smart City [9]. Another important assumption is that the sensing devices, including the cluster heads, are owned and managed by an entity (e.g., the city council) other than the one that governs the access to the readings of the devices, namely the sensing server (e.g., Amazon). Moreover, the two entities are considered not to collude against the users.

3.2 Adversarial Model

The adversarial model considered in this paper is *semi-honest* (or honest-but-curious), which means that the adversary is assumed to follow the protocol but may try to benefit from a privileged role in the system to obtain information beyond what is permitted. More precisely, the adversary is interested in learning information about the interests of a particular user based on the queries he/she issues and the nodes responding to them.

We assume that the sensing server is a semi-honest adversary which has the following capabilities:

– **Content analysis**: inspects any packet it receives in order to retrieve sensitive information. The analysis is not limited to the payload of the packets but may also include the packet headers. Thus, the adversary may learn the query contents, the sensed data and the identities of the parties involved in the communication.
– **Statistical analysis**: analyzes the features of the communication flow including the distribution of messages, the time at which messages are delivered or received, the transmission rate, and so on. The goal of this type of attack is to discover patterns in the transmissions in order to infer sensitive information.

The hearing range of the adversary is also an important aspect to consider when dealing with traffic analysis attacks. Typically, a semi-honest adversary is internal and limits its actions to the traffic addressed to it or traversing it. Nonetheless, in this paper we assume a more powerful adversary, which is allowed to extend its hearing range to the sensing network. The attacker is allowed to collude with external entities located in the vicinity of the sensing devices.

Moreover, we consider that a semi-honest sensing server can try to cheat by slightly modifying its behavior as long as it does not deviate from the protocol specification. For example, the adversary can craft random numbers at will instead of using a pseudo-random number generator for that purpose. The adversary can benefit from vague protocol specifications or randomly defined operations.

4 Query Privacy for Sensing Platforms Protocol

This section provides a detailed description of the QPSP protocol. First, we present a brief overview of the protocol and then continue with the explanation of each of the phases involved in it . Prior to that, we introduce some cryptographic background.

4.1 Overview

The QPSP protocol consists of three phases: initialization, query, and response. During the initialization a *global public key*, denoted by pk_P, is generated by the cluster heads in a distributed way. This global public key corresponds to the sensing network as a whole and no single entity controls the corresponding decryption key in order to reduce the possibility of key compromise. This phase also deals with the generation of the corresponding re-encryption keys.

The global public key pk_P is used to encrypt the queries sent to the sensing server, which transforms them using techniques from proxy re-encryption into new, encrypted queries that can be decrypted by the cluster heads only. This is done using special keys called *re-encryption keys*. During this process, the content of the query remains unaltered and cannot be obtained by the sensing server. Once the query has been decrypted by a cluster head, it is forwarded to the appropriate sensing device without disclosing its identity to the gateway.

The response phase is simpler. The confidentiality of the response is secured from the user end by incorporating a fresh key into the query to be used to encrypt the content of the response. From an abstract point of view, the communications are basically a two-message exchange between a user and a sensing device but some traffic obfuscation mechanisms are introduced to prevent leaking information.

4.2 Preliminaries

This section introduces some cryptographic notions that will be used during the definition of the QPSP protocol. Due to space limitations we do not go into details.

Bilinear pairings Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order q. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ satisfying the properties of bilinearity, non-degeneracy, and computability (see [17] for more details). Depending on the characteristics of the groups involved, there are essentially three types of pairings, namely Type-1, Type-2 and Type-3. In this paper we use Type-3 pairings as they achieve the best trade-off between security and efficiency [15].

Proxy Re-Encryption From a high-level viewpoint, proxy re-encryption is a type of public-key encryption that enables a proxy to transform ciphertexts under Alice's public key into ciphertexts decryptable by Bob's secret key. In order to do this, the proxy is given a re-encryption key, generated by Alice, which makes this process possible. There are multiple proxy re-encryption proposals in the literature, the most prominent are those of Blaze et al. [6] and Ateniese et al. [4].

4.3 Initialization phase

In this phase, the sensing platform sets up the necessary public parameters and cryptographic keys. As mentioned, we describe a distributed key generation procedure, principally performed by the cluster heads. Finally, we also present a key validation procedure in order to guarantee the correctness and validity of the key generation process.

Setup Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a Type-3 pairing, and g and h generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let Z be the result of computing $e(g, h)$. The public parameters of the system are the elements of the tuple (e, g, h, Z).

Key Generation The main goal of this procedure is to create the global public key for the sensor domain, denoted by pk_P, with no associated private key. In parallel, it is also necessary to create re-encryption keys that enable the transformation of ciphertexts between the global public key and the cluster heads' public key.

First, each cluster head CH_i generates a key pair $(pk_i, sk_i) = (h^{x_i}, x_i)$, where x_i is sampled uniformly at random from $\mathbb{Z}_q$. The cluster heads distribute their public keys among the rest, so we can assume that after this step, the cluster heads knows each others' public key. Next, each cluster head independently generates a temporal secret value p_i sampled uniformly at random from $\mathbb{Z}_q$, and computes the values $u_i = Z^{p_i}$ and $v_{ij} = (pk_j)^{p_i} = h^{p_i x_j}$, for all $j \in \{1, ..., N\}$. Finally, it sends $(u_i, \{v_{ij}\})$ to the sensing server for aggregation.

Once the sensing server has received the inputs from all the cluster heads, it computes the global public key and corresponding re-encryption keys as follows:

$$pk_P = \prod_{i=1}^{N} u_i = \prod_{i=1}^{N} Z^{p_i} = Z^{p_1 + ... + p_N} = Z^p \tag{1}$$

$$rk_{P \to i} = \prod_{j=1}^{N} v_{ji} = \prod_{j=1}^{N} h^{x_i p_j} = h^{x_i (p_1 + ... + p_N)} = h^{x_i p} \tag{2}$$

Note that this procedure guarantees that the secret $p = p_1 + ... + p_N$ (which is the private key associated with the global public key) is never computed explicitly and that it cannot be recovered efficiently, by the Discrete Logarithm hardness assumption.

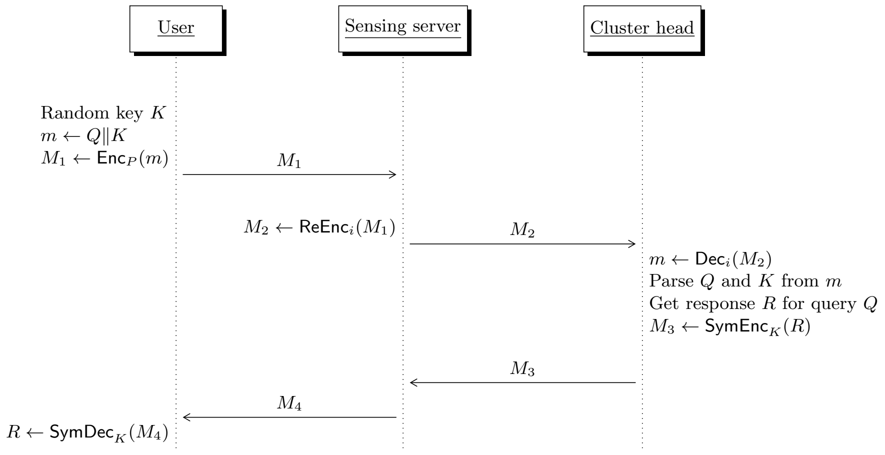
Key Validation Given that the sensing server aggregates the inputs from all the cluster heads in order to create the global public key and the associated re-encryption keys, it is possible (although not sensible) that it misbehaves during the aggregation process, for example, by discarding the input and publishing an alternative global public key for which it controls the corresponding decryption key. Recall that the goal of the distributed key generation process is to create a global public key with no associated private key.

In Appendix A.1 we describe a procedure for key validation, in which the cluster heads interact with each other and the sensing server. The existence of such a procedure acts as a deterrent to possible misbehavior from the sensing server, since it represents an efficient mechanism to detect any deviation from the agreed key generation process. Therefore, we can assume that the sensing server does not misbehave during key generation.

4.4 Query phase

This phase comprises the first direction of the communication, which is from user to cluster head, via the sensing server (as shown in Fig. 3). It comprises two messages: the first between user and sensing server, and the second between sensing server and the cluster head.

Message 1 (Encryption) The first message of the protocol is constructed on the user side, and is delivered to the sensing server. The idea is that the user encrypts the query with the global public key pk_P, and the encrypted query is later re-encrypted to a cluster head of the sensing platform.



User
Sensing server
Cluster head
Random key K
$m \leftarrow Q \| K$
$M_1 \leftarrow \mathsf{Enc}_P(m)$
M_1
$M_2 \leftarrow \mathsf{ReEnc}_i(M_1)$
M_2
$m \leftarrow \mathsf{Dec}_i(M_2)$
Parse Q and K from m
Get response R for query Q
$M_3 \leftarrow \mathsf{SymEnc}_K(R)$
M_3
M_4
$R \leftarrow \mathsf{SymDec}_K(M_4)$

Fig. 3. Main part of the QPSP protocol

The encryption scheme[1] we propose is based on Ateniese et al.'s proxy re-encryption scheme [4], which is proven IND-CPA secure. Let us suppose that the input to be encrypted is represented by an element $m \in \mathbb{G}_T$. The user samples random $r \in \mathbb{Z}_q$ and produces the ciphertext $CT = (CT_1, CT_2)$ as follows:

$$\mathsf{Enc}_P(m) = (g^r, m \cdot (pk_P)^r) = (g^r, m \cdot Z^{pr})$$

An important point when it comes to defining the actual protocol messages is that the initiator (i.e., the user) does not need to own any kind of key (either symmetric or asymmetric) to query the responder (i.e., a sensing device); that is, the initiator is unauthenticated. To the contrary, the responder has a public key, in this case, a global public key for the sensing platform. Therefore, the initiator can use this public key as a means to set up a secure channel for the response, by encrypting a fresh random key K with the public key of the sensing platform (as with the query Q). This is reminiscent of the one-pass key transport technique described in [20, Section 12.5.1]. Therefore, the first message of the protocol, which is named M_1 and generated by the user, is basically the encryption of the query Q and a fresh random key K to be used for securing the response, as shown in Fig. 3.

Message 2 (Re-encryption) Message M_1 is received by the sensing server, which transforms it into a ciphertext, decryptable by one of the cluster heads. Since, in principle, the encrypted query does not convey any metadata, the sensing server simply chooses some arbitrary cluster head (either at random or

[1] In practice, the proposed scheme would be used to encrypt a fresh random key, which in turn will be used to encrypt the actual message with a symmetric encryption algorithm, following a hybrid encryption approach. For simplicity in the description, we will obviate this.

following some network delivery criteria). Therefore, the server is basically a blind gateway between the users and the sensing network.

Let us suppose that the sensing server chooses cluster head CH_i then, the corresponding re-encryption key is $rk_{P\to i} = h^{x_i p}$. The sensing server transforms the original ciphertext CT into a new ciphertext CT' intended for CH_i as follows:

$$\mathsf{ReEnc}_i(CT) = (e(CT_1, rk_{P\to i}), CT_2) = (Z^{prx_i}, m \cdot Z^{pr})$$

The final message M_2 is simply the re-encrypted ciphertext.

4.5 Response phase

This phase includes the decryption and delivery of the query to the actual recipient as well as the response to the user.

Message 3 (Decryption) The cluster head CH_i receives message M_2, containing ciphertext $CT' = (CT'_1, CT'_2)$ and decrypts it using its own secret key as follows:

$$\mathsf{Dec}_i(CT') = CT'_2 \cdot (CT'_1)^{-1/sk_i} = m \cdot Z^{pr} \cdot (Z^{prx_i})^{-1/x_i} = m$$

The resulting output m is parsed as the original query Q and a response key K. Next, the cluster must deliver m to the actual destination but as this process is subject to traffic analysis we propose a transmission mechanism inspired by the notion of k-anonymity.

All cluster heads use a *deterministic mapping function* to choose k destinations. This function is such that it receives one identifier as input and always returns the set of k identifiers, which include the original one. Therefore, CH_i can either forward the actual query (encrypted under a shared key) to the k sensing devices output by the mapping function or simply send bogus queries to the $k-1$ cover destinations. In either case, the cluster heads must share secrets to encrypt the message and thus prevent content analysis attacks.

After reaching the k sensing devices, all of them must behave in the same way. As a result, devices receiving a (cover) query must reply to it, possibly with bogus or synthetic data. Finally, the results collected by the corresponding cluster heads are relayed to the cluster head who originally received the query, which filters out cover messages and selects the true response R. Finally, the cluster head encrypts it using the key K to produce message M_3; alternatively, the sensing device itself can encrypt the response, provided the cluster head sends the key K together with the query. Finally, message M_3 is delivered to the sensing server, which simply forwards it to the user.

Message 4 (Delivery of response) The user receives message M_4 from the sensing server, and decrypts it using the key K in order to retrieve the response to his original query.

5 Security Analysis

The analysis presented in this section concentrates on the two security properties the QPSP protocol aims to protect, namely query confidentiality and query privacy.

5.1 Query Confidentiality

The encryption scheme we use for protecting queries from the user to the cluster heads is essentially a restricted version of Ateniese et al.'s proxy re-encryption scheme [4], adapted to a Type-3 pairing setting. This encryption scheme satisfies the security notion of indistinguishability against chosen-plaintext attacks (IND-CPA), under the External Diffie-Hellman assumption (XDH) [5]. For reasons of space, we omit the full security proof, but the rough idea is as follows.

Assuming that there is an adversary $\mathcal{B}$ that wins the IND-CPA security game, it is possible to construct an algorithm $\mathcal{A}$ that uses $\mathcal{B}$ to solve the XDH problem (which is essentially the DDH problem in $\mathbb{G}_1$ but in a pairing setting). Adversary $\mathcal{A}$ receives a DDH tuple $(g, g^a, g^b, g^c) \in \mathbb{G}_1^4$, and is asked to decide whether $c = a \cdot b$. In order to decide, it simulates the environment for the adversary $\mathcal{B}$ by taking element g^a for generating the global public key $pk_P^* = e(g^a, h) = Z^a$, and elements g^b and g^c for generating the challenge ciphertext $CT^* = (g^b, m_\delta \cdot e(g^c, h)) = (g^b, m_\delta \cdot Z^c)$. It can be seen that when $c = a \cdot b$, the challenge ciphertext is a valid encryption of m_δ under pk_P^*, and $\mathcal{B}$ guesses δ correctly with non-negligible advantage. When the guess is correct, adversary $\mathcal{A}$ outputs $c = a \cdot b$, which solves the DDH problem in $\mathbb{G}_1$ with the non-negligible advantage. To the contrary, when c is random, $\mathcal{B}$ does not have any advantage in guessing δ, and hence, neither does $\mathcal{A}$. Overall, it can be seen that $\mathcal{A}$ still solves the XDH problem with non-negligible advantage.

5.2 Query Privacy

We have just shown that the sensing server cannot obtain information from content analysis but may still perform statistical analysis. Next we show that even when the sensing server does not strictly follow the protocol specification, it learns nothing.

First, let us assume a single protocol run. After a single query, the sensing server may learn, with the help of external colluders, that the user has queried one of the k nodes responding to the query. Note that if k is sufficiently large, query privacy is ensured as long as the mapping function has been designed taking into account the properties of l-diversity and t-closeness [18]. In the case that the mapping does not respect these notions, the attacker may not know which particular device replied to the query but still learn that the user is interested in, for instance, radiation levels. We assume that this is dependent on the scenario and thus beyond the scope of this paper.

Let us now assume that the user is repeatedly issuing queries. If the queries are addressed to different sensing devices, the analysis is similar to the single execution case. However, if the user is regularly querying a particular device and the sensing server is aware of this, the adversary is incapable of determining which of the elements in the anonymity set is the actual recipient. Even if the sensing server tries to cheat by choosing the cluster heads at will, it is still unable to reduce the size of the anonymity set because all cluster heads use the same mapping function.

The sensing server's last resort is to craft its own queries and submit them to the sensing network. This is possible since we are considering a public sensing network. However, there is no incentive for it to do so since the only thing the attacker will learn is the mapping function for a particular node, which is not secret, and the data sensed by the node. These data are not sensitive since they are obtained from a query issued by the sensing server and not by a particular user. Thus, the interests of the user are not revealed.

Finally, recall that the sensing server and the sensing devices are supposed not to collaborate. This is because the former knows the identity of the user and the latter is aware of the nodes of interest to the user. To reduce the risk of privacy exposure, the identity of the user can be further protected with the help of an anonymity network, such as Crowds or Tor.

6 Experimental Evaluation

In order to experimentally evaluate the overhead of the cryptographic operations in our proposal, we implemented a proof of concept in C using the Apache Milagro Crypto Library [2]. Since we defined the cryptographic scheme over a Type-3 pairing setting, it was necessary to use an elliptic curve that supports this. We selected a 256-bit Barreto-Naehrig (BN) curve, which is suitable for said pairings and offers a good trade-off with respect to security and performance [3].

We used three different execution platforms, in order to simulate the characteristics of the entities involved in the potential use cases. For the user and sensing server, we used a laptop with an Intel Core 2 Duo processor @ 2.66 GHz and 8 GB of RAM. For the cluster heads, we performed our tests in two different platforms: a Raspberry Pi Gen 1 Model B (SoC Broadcom BCM2835, 32-bit, single core, 700Mhz, 512 MB), and an Intel Galileo Gen 1 (SoC Intel Quark X1000 32-bit, single core, single-thread, P54C/i586 400Mhz, 256 MB).

Table 1. Performance of the cryptographic operations for different execution platforms

Entity	Platform	Operation	Cost (ms)
User	Laptop	Encryption	7.58
Sensing server	Laptop	Re-Encryption	11.55
Cluster head	Raspberry Pi	Decryption	46.20
Cluster head	Intel Galileo	Decryption	122.20

Table 1 shows the results of our experiments, categorized by the type of operation, entity and execution platform. Both encryption and re-encryption of queries are tested on a PC-like platform, while decryption is done in a sensor-like device. Experiments were executed 100 times and the average value was taken. It can be seen that the results are of the order of 10 ms in the user- and server-side, while they range between 46 and 122 ms on the side of the cluster head, depending on the execution platform. Further optimizations include the pre-computation of pairings for re-encryption, given that re-encryption keys are fixed arguments to pairings. In this regard there are several techniques, such as [10] which reports a speed-up around 30%. It is also possible to study different curves in order to find the more efficient ones on the side of the cluster head, that is, curves that minimize the cost associated with exponentiations on $\mathbb{G}_T$.

7 Conclusions

This paper has presented the QPSP protocol. The proposed solution prevents users from being profiled by semi-trusted Sensing-as-a-Service platforms. This is achieved with the help of proxy re-encryption primitives and traffic obfuscation at the sensing network. More precisely, the user sends queries to the platform encrypted with a global public key generated by a set of cluster heads, the query is then re-encrypted by the sensing server and forwarded to one of the cluster heads, which is responsible for transmitting the query to the final destination using a privacy-preserving routing protocol.

The proposed solution is intended for Sensing-as-a-Service platforms where the data collected by the sensing devices are public and therefore anyone can query the network. We are planning to extend our solution to a scenario where the access needs to be both authenticated and respectful of privacy. Moreover, we are exploring network management issues such as the revocation of nodes and how to deal with the addition of new cluster heads once the network has been deployed.

Acknowledgements

This paper has been partially supported by the Junta de Andalucía through the project FISICCO (P11-TIC-07223) and by the Spanish Ministry of Economy and Competitiveness through the PERSIST project (TIN2013-41739-R).

References

1. Amazon Web Services. How the AWS IoT Platform Works. https://aws.amazon.com/iot/how-it-works/. [Last Access: 11/2016].
2. Apache Milagro. Apache Milagro Crypto Library. https://milagro.apache.org/.
3. Diego F Aranha, Paulo SLM Barreto, Patrick Longa, and Jefferson E Ricardini. The realm of the pairings. In *Selected Areas in Cryptography–SAC 2013*, pages 3–25. Springer, 2013.

4. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, 2006.
5. Giuseppe Ateniese, Jan Camenisch, and Breno De Medeiros. Untraceable rfid tags via insubvertible encryption. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 92–101. ACM, 2005.
6. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. *Advances in Cryptology—EUROCRYPT'98*, pages 127–144, 1998.
7. Bogdan Carbunar, Yang Yu, Weidong Shi, Michael Pearce, and Venu Vasudevan. Query privacy in wireless sensor networks. *ACM Trans. Sen. Netw.*, 6(2):14:1–14:34, March 2010.
8. F. Chen and A. X. Liu. Privacy- and integrity-preserving range queries in sensor networks. *IEEE/ACM Transactions on Networking*, 20(6):1774–1787, Dec 2012.
9. Annalisa Cocchia. *Smart and Digital City: A Systematic Literature Review*, pages 13–43. Springer International Publishing, Cham, 2014.
10. Craig Costello and Douglas Stebila. Fixed argument pairings. In *International Conference on Cryptology and Information Security in Latin America*, pages 92–108. Springer, 2010.
11. E. De Cristofaro and R. Di Pietro. Adversaries and countermeasures in privacy-enhanced urban sensing systems. *IEEE Systems Journal*, 7(2):311–322, June 2013.
12. Emiliano De Cristofaro, Xuhua Ding, and Gene Tsudik. Privacy-Preserving Querying in Sensor Networks. In *18th International Conference on Computer Communications and Networks*, ICCCN '09, pages 1–6, San Francisco, CA, 3-6 Aug. 2009. IEEE Computer Society, Washington, DC, USA.
13. Roberto Di Pietro and Alexandre Viejo. Location privacy and resilience in wireless sensor networks querying. *Comput. Commun.*, 34(3):515–523, March 2011.
14. T. Dimitriou and A. Sabouri. Privacy preservation schemes for querying wireless sensor networks. In *IEEE International Conference on Pervasive Computing and Communications Workshops,*, pages 178–183, 2011.
15. Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
16. Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.
17. Mehmet Sabır Kiraz and Osmanbey Uzunkol. Still wrong use of pairings in cryptography. Cryptology ePrint Archive, Report 2016/223, 2016. http://eprint.iacr.org/.
18. Ninghui Li, Tiancheng Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE 2007. IEEE 23rd International Conference on Data Engineering*, pages 106–115, April 2007.
19. Xiaojing Liao and Jianzhong Li. Privacy-preserving and secure top-k query in two-tier wireless sensor network. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 335–341, Dec 2012.
20. Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography.* CRC press, 1996.
21. Péter Schaffer, Károly Farkas, Ádám Horváth, Tamás Holczer, and Levente Buttyán. Secure and reliable clustering in wireless sensor networks: A critical survey. *Computer Networks*, 56(11):2726 – 2741, 2012.
22. X. Sheng, J. Tang, X. Xiao, and G. Xue. Sensing as a service: Challenges, solutions and future directions. *IEEE Sensors Journal*, 13(10):3733–3741, Oct 2013.
23. Willard Tu. Sensors as a Service on the Internet of Things. White paper, ARM Ltd., January 2014.

24. Xiaoying Zhang, Lei Dong, Hui Peng, Hong Chen, Deying Li, and Cuiping Li. Achieving efficient and secure range query in two-tiered wireless sensor networks. In *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*, pages 380–388, May 2014.

Appendix A

A.1 Key Validation Procedure

The validation process has to check three requirements: (i) that the global public key is the result of aggregating the inputs from all the cluster heads, (ii) the re-encryption keys are also generated by aggregation, and (iii) the re-encryption keys are correctly associated with the global public key (i.e., they allow the transformation of ciphertexts from the global public key to the cluster heads' keys).

Validation requirement (i) can be checked simply by engaging all the cluster heads in an incremental ring-style protocol, where each of them receives an intermediate value and multiplies it by its input u_i for the public key; hence, at the end of the protocol, the global public key (which we assume is published by the system and known by all the cluster heads) should be obtained. As long as no cluster head is corrupted, this requirement can be correctly checked.

Validation requirements (ii) and (iii) involve re-encryption keys. A complication here is that, although the global public key is, indeed, public, re-encryption keys may not be (and possible should not be) public[2]. However, it is possible for each cluster head to challenge the service provider and to test if the response corresponds to a valid re-encryption key as follows. Each cluster head CH_i computes a random challenge value g^k, and sends it to the sensing server, which responds with the value $e(g^k, rk_{P\to i})$. Now the cluster head checks whether the following equation holds:

$$e(g^k, rk_{P\to i}) = (pk_P)^{x_i k}$$

It can be seen that, given $rk_{P\to i} = h^{x_i p}$ and $pk_P = Z^p$, the correct response from the service provider will be $Z^{x_i k p}$, which is indeed equal to $(pk_P)^{x_i k}$. Therefore, each cluster head can independently check requirements (ii) and (iii) and report to the others in the case there is a problem. Since, the correctness of the global public key has been checked before, and this public key is used during the validation of the re-encryption keys, it can be safely accepted that all the keys are correct.

[2] In the proxy re-encryption literature, knowledge of a re-encryption key and the corresponding recipient's private key can be used in some cases to derive the original private key. This is known as a "collusion attack"[4].