Scientific Research Publishing

Journal of Information Security, 2018, 9, 299-314
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

SecSPS: A Secure and Privacy-Preserving Framework for Smart Parking Systems

Ali Alqazzaz[1], Ibrahim Alrashdi[1], Esam Aloufi[1], Mohamed Zohdy[2], Hua Ming[1]

[1]Computer Science and Engineering Department, Oakland University, Rochester, MI, USA
[2]Electrical and Computer Engineering Department, Oakland University, Rochester, MI, USA
Email: aalqazzaz@oakland.edu, iralrashdi@oakland.edu, aloufi@oakland.edu, zohdyma@oakland.edu, ming@oakland.edu

How to cite this paper: Alqazzaz, A., Alrashdi, I., Aloufi, E., Zohdy, M. and Ming, H. (2018) SecSPS: A Secure and Privacy-Preserving Framework for Smart Parking Systems. *Journal of Information Security*, **9**, 299-314.
https://doi.org/10.4236/jis.2018.94020

Received: August 3, 2018
Accepted: October 12, 2018
Published: October 15, 2018

Abstract

Smart parking systems are a crucial component of the "smart city" concept, especially in the age of the Internet of Things (IoT). They aim to take the stress out of finding a vacant parking spot in city centers, due to the increasing number of cars, especially during peak hours. To realize the concept of smart parking, IoT-enabling technologies must be utilized, as the traditional way of developing smart parking solutions entails a lack of scalability, compatibility with IoT-constrained devices, security, and privacy awareness. In this paper, we propose a secure and privacy-preserving framework for smart parking systems. The framework relies on the publish/subscribe communication model for exchanging a huge volume of data with a large number of clients. On one hand, it provides functional services, including parking vacancy detection, real-time information for drivers about parking availability, driver guidance, and parking reservation. On the other hand, it provides security approaches on both the network and application layers. In addition, it supports mutual authentication mechanisms between entities to ensure device/data authenticity, and provide security protection for users. That makes our proposed framework resilient to various types of security attacks, such as replay, phishing, and man-in-the-middle attacks. Finally, we analyze the performance of our framework, which is suitable for IoT devices, in terms of computation and network overhead.

Keywords

IoT, Publish/Subscribe, Messaging Protocol, Security, Parking System

1. Introduction

Due to the rapid increase in automobile numbers, finding an available parking space in city centers during peak hours has become a serious problem for drivers. It is estimated that 30% of daily traffic jams in crowded areas is caused by car-owners looking for vacant parking spaces, and that a driver spends, on average, 7.8 minutes trying to find an available spot [1] [2]. This problem not only consumes time and fuel, but increases air pollution and driver frustration. As the situation becomes worse, so the demand for smart parking systems and services is rapidly growing. The Internet of Things (IoT)-enabling technologies have great potential for providing an ideal solution—a smart parking system to significantly reduce traffic congestion and improve the quality of life of citizens.

To minimize hassle and inconvenience for drivers, several solutions have been proposed in recent years. Most of them entail building parking guidance information (PGI) systems for better parking management [3]-[10]. PGI systems are able to provide drivers with dynamic information on the location of vacant parking spaces in car lots within controlled environments, and direct them to available parking spots. The accurate operation of PGI systems is based on the use of sensors that are able to detect the presence of vehicles, and thus can monitor parking spots. Such systems cannot guarantee the availability of such a parking spot when the driver actually arrives at the parking facility, however. Other researchers have used various technologies to ensure smoothness of traffic in and around parking spots, including the Global Positioning System for parking spot detection, based on self-localization, video cameras for collecting and collating information on vehicle parking spaces, radio frequency identification (RFID) technologies for entering and exiting parking spots, cloud-based, and text-messaging-based parking reservation services [11]-[21]. To their credit, all of these proposed solutions have introduced sensible improvements in the field of parking management, but they still suffer from a lack of suitability and adaptability to IoT requirements to ensure their openness, reliability, and networking accessibility. First, they utilize the traditional request/response communication model, which is not suitable for building large-scale IoT solutions, and handling massive volumes of data. Moreover, they rely on HTTP as the messaging protocol, which is not the ideal choice for IoT devices. In addition, all of them are characterized by several functional requirements, but they do not pay enough attention to the non-functional requirements, among which security and privacy play important roles, due to the existence of diverse cyber attacks targeting most cyber-physical systems.

To solve the aforementioned parking problems, and to fully realize the concept of a smart parking system, IoT-enabling technologies must be taken into account. This paper proposes a secure and privacy-preserving framework for smart parking systems called SecSPS, which consists of three main components: a sensor network that is responsible for monitoring vehicles going in and out of the parking facility; one or more smart gateways, based on the size of the car lot; and a broker, who takes responsibility for information dissemination in real time.

Firstly, the SecSPS framework can provide a real-time parking information and navigation service to users in search of parking spots, so that drivers can easily and quickly find vacant parking spaces. As a result, the time and gasoline consumed in search of free parking spaces can be reduced. It also helps in reducing carbon monoxide emissions, among other pollutants, as well as reducing traffic congestion in city centers.

Secondly, parking reservation is a key feature of any smart parking system. Therefore, our framework enables drivers to reserve a parking spot in advance, in order to ensure the availability of the vacant parking space by the time they arrive. On one hand, less time spent parking leads to less stress and happier customers, *i.e.*, improves overall customer experience. On the other hand, this creates car lots that are easy to manage, while maximizing revenues and efficiency.

Thirdly, to the best of these authors' knowledge, this is the first framework based on publish/subscribe (pub/sub) architecture, which is a very powerful way for IoT devices to interact. This architecture offers distributed, asynchronous, loosely-coupled many-to-many communication between message producers (publishers) and message consumers (subscribers). In our case, each parking facility (publisher) publishes its own available parking information under a topic, and drivers (subscribers) who are interested in that information find about it more or less instantly by simply subscribing to the same topic.

Finally, our framework provides a secure communication channel among end-points when sending data over the Internet by using a transport layer security (TLS) protocol. Unlike with HTTP, a pub/sub client need only establish a connection once per session, rather than re-establishing a connection with every request, which makes the TLS less costly in terms of CPU and bandwidth. Moreover, the framework guarantees confidentiality, integrity, and notification authenticity by encrypting the packet's payload. In other words, it allows end-to-end encryption for the application data, even for untrusted environments. This approach does not require any additional custom mechanism on the broker side for decrypting the data in order to route the message to subscribers.

The reminder of this paper is organized as follows: Section 2 fills in the required background; Section 3 describes the system model, threat model, and design goals; Section 4 provides a detailed description of the proposed framework; Section 5 analyzes the security of our framework; and Section 6 covers the performance evaluation. Finally, we conclude our work in Section 7.

2. Background

2.1. Publish/Subscribe (Pub/Sub) Model

The pub/sub model is an alternative to the traditional client/server model, whereby a client communicates directly with an endpoint. It is a data-centric architecture, whereby messages are delivered to interested destinations without knowing the IP addresses of these destinations. In other words, it decouples the sender of a specific message (publisher) from another client, who is getting the message (subscriber), and allows communication via a third component (the broker). It also provides a greater scalability than the traditional approach because operations on the broker side can be parallelized and processed in an event-driven manner [22]. The aforementioned three main entities in such a system—publisher, subscriber, and broker—are shown in **Figure 1**. Publishers are the generators and owners of the content, *i.e.*, the service providers. Subscribers are any members who are interested in receiving the particular content and subscribe to it. The subscribers receive the desired content through an information delivery system—the broker.

The broker can filter messages in various ways, so that each subscriber gets only the messages they are interested in [23]. The first such way is a topic-based filtering, where messages are filtered by topic or subject. The receiving client subscribes only to the topic(s) they are interested in, and then gets notified based on that/those topics. Topic names are generally represented by a URL-like notation with a hierarchical structure. The second approach is a content-based filtering, where the broker filters the message, based on the actual content of the considered event(s). A big drawback of this type is that the content of the message must be known in advance and cannot be encrypted. The third option is a type-based filtering, which filters according to the type (class) of the message (event). In this scenario, subscribers can listen to all messages, which are of type *X*, or any subtype thereof. As a downside to this option, subscribers need to know in advance the structure of the published data.

2.2. Transport Layer Security (TLS) Protocol

In 1999, the Internet Engineering Task Force standardized a new cryptographic protocol called a TLS protocol. It primarily aims to achieve three main goals—authentication, confidentiality, and data integrity—which are critically important to securely communicate over the Internet. Confidentiality can be achieved using symmetric encryption with a strong block cipher, such as the advanced encryption standard (AES). On the other hand, authentication is accomplished with public-key cryptography. Finally, data integrity can be checked using message authentication code (MAC). In TLS, confidentiality and authentication are achieved through a series of messages called a "handshake".

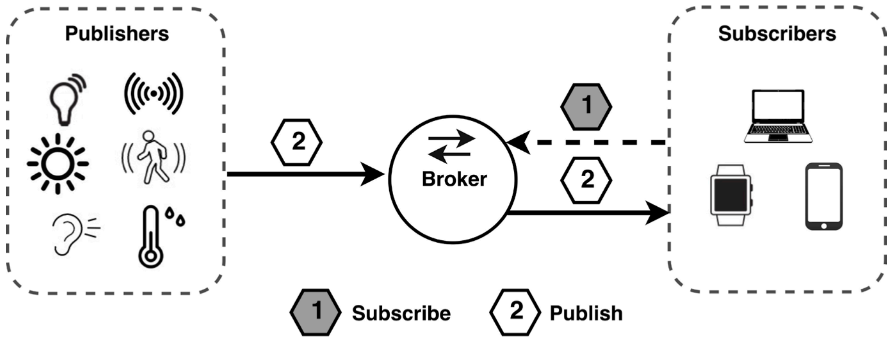

Figure 1. The publish/subscribe architecture.

As shown in Figure 2, the handshake process starts with the exchange of *ClientHello* and *ServerHello* messages, for purposes of exchanging certificates and negotiating a cipher suite that will be used during the session. Cipher suites are a combination of security algorithms that usually include a key exchange algorithm, an encryption algorithm, and a MAC algorithm. An example of a cipher suite name is TLS_RSA_WITH_AES_128_CBC_SHA, which defines a session that uses:

- RSA for key establishment and authentication;
- a 128-bit AES in Cipher Block Chaining (CBC) mode for confidentiality; and
- a secure Hashing Algorithm (SHA) for integrity.

After validation and negotiation of certificates are completed, both client and server exchange a secret key (session key). Finally, they exchange *Finished* messages to tell each other that, from now on, everything will be authenticated and encrypted. This handshake is officially complete when the client and server exchange the *Finished* messages [24] [25].

3. Models and Goals

In this section, we cover the main components of our framework, threat model, and design goals.

3.1. Proposed Framework

As shown in Figure 3, our proposed framework consists of several components, involving a large number of parking spaces equipped with sensor nodes, a smart gateway, a broker, and clients. In the following subsections, we briefly describe the main functions of each component. Note that our framework is applicable to a wide range of parking areas, including malls, airports, universities, city centers, and hospitals. Also, it is suitable for both indoor and outdoor deployments.

3.1.1. Sensor Nodes

The main objective of this component is to monitor each car lot and detect the presence of vehicles, in order to calculate the total number of free parking spaces

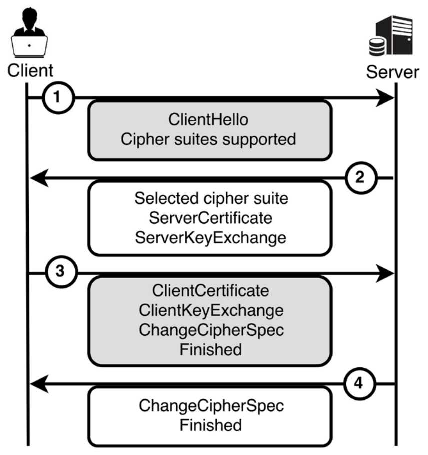

Figure 2. Illustration of TLS handshake process.

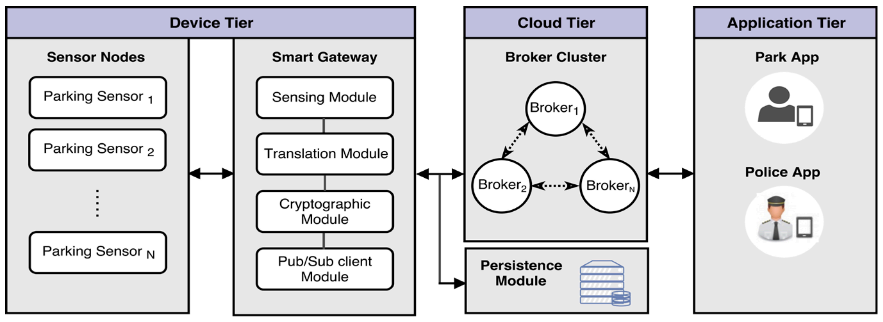

Figure 3. The architecture of our proposed framework (SecSPS).

in each car lot at any time. To this end, each parking spot is equipped with a sensor to identify its status; however, there are various sensing technologies that can be reliably used to detect vehicles, technologies involving ultrasonic, magnetic, radar, and optical (infrared) sensors. In identifying the right technology, different factors must be taken into account, including size of target, sensing range, sensor mounting, accuracy, whether the sensor is indoor or outdoor, cost, and environmental conditions. That being said, it may be beneficial to combine two sensing techniques to achieve an advanced level of accuracy.

3.1.2. Smart Gateway

In general, IoT gateways perform various critical functions, such as device connectivity, protocol translation, data filtering and processing, security, updating, management, and more. Here, the smart gateway is responsible for receiving the states of parking spots from the sensor nodes, analyzing and encrypting this data, and sending them to the specific broker.

3.1.3. Broker

The broker is the heart of any pub/sub system. It is primarily responsible for receiving all encrypted messages from the smart gateway, filtering them, deciding who is interested in them, and then sending the messages to the subscribed clients. Since the application data itself stays encrypted all the time, and the broker has no way of looking into the encrypted data, it uses the unencrypted messages metadata (*i.e.*, topic name) for routing. Note, the broker can be either public or private; for example, it can be owned by the Department of Transportation.

3.1.4. Client

A client is any electronic device, from a micro-controller up to a fully-fledged server that is equipped with custom software that enables connecting to a broker over the Internet, such as a laptop, tablet, smartphone, or desktop. It can connect to the broker, subscribe to one or more topic(s), and be notified whenever there are new messages.

3.2. Threat Model

In [26], authors show that the semi-honest model tends to be widely accepted by the scientific research community, as it offers a sufficient level of security within reasonable computational and communication costs. Thus, we assume the semi-honest model in this paper, meaning that all parties correctly follow the rules of the protocol. But they also attempt to obtain as much information as possible. This model does not take into account powerful attackers who have physical access to the devices and control them, however. In addition, we assume that the certificate authority (CA), which issues certificates, is secure and trusted.

3.3. Design Goals

This section is dedicated to identifying the goals that our framework should satisfy.

- **Correctness**: If both the broker and the client follow the rules honestly, the client can get correct real-time parking availability information. Moreover, the client with a parking reservation is guaranteed to get a parking spot by the time he/she arrives. In other words, the framework must provide all its services in a correct manner when the rules are honestly followed.
- **Security**: The framework must protect and guarantee the confidentiality and integrity of transmitted data, and keep it secure over the network. On one hand, the attacker cannot get the original data when given the encrypted messages. On the other hand, the framework can secure the communication channels between clients and brokers. Moreover, it should provide mutual authentication between the pub/sub clients and the broker.

4. Framework Description Details

In this section, we describe in detail the proposed framework, which can use any pub/sub messaging protocol, such as Message Queuing Telemetry Transport (MQTT). In general, we will use the term "users" when referring to drivers or vehicles. Initially, parking facility owners are required to register their identities to a trusted authority (e.g., governmental transportation authorities) before participating and launching secure services, in order to guarantee authentication and enable secure communications. The trusted authority makes sure attackers do not gain control of the network and protects sensitive data. Therefore, the parking facility owners need to send registration and connection requests to the broker's security manager seeking the permission for providing service(s) (e.g., creating topics).

4.1. Parking Space Monitoring and Detection

As mentioned earlier, the parking spot status information at each car lot is important, so that the broker can get and manage information in real time. Therefore, each parking space is equipped with a sensor (e.g., ultrasonic sensors), which is capable of sensing and detecting free spaces. When a vehicle is detected, a message is transmitted to the smart gateway to be interpreted and processed. This phase of the framework is known as the monitoring module.

4.2. Data Aggregation

The smart gateway aggregates all the readings from the sensors, processes these data, and calculates two things: the number of free parking spots for each floor, and the total number of free parking spots at the car lot with the corresponding vacant percentages. Next, it encrypts all this information with the secret key of the target topic, as will be described below. Finally, it sends the encrypted message to the broker for distribution and delivery, over TLS. Here, TLS is used to create a secure communication channel between the smart gateway and the broker, using the handshake mechanism. Using a secure channel makes it more difficult for third parties to intercept, or eavesdrop on, messages in transit. The smart gateway uses the broker's certificate, which is issued by a trusted and secured authority, to verify its identity before sending a bit.

As mentioned earlier, our framework provides an additional security layer by supporting the exchange of encrypted messages (known as payload encryption). This approach ensures end-to-end encryption, preventing eavesdropping along the way, and spoofing of valid application data. In this approach, only the payload of the message is encrypted (PUBLISH packet payload) to ensure that there is no additional mechanism needed on the broker side for decrypting the data, in order to deliver the message to the subscriber. The broker uses the unencrypted packet metadata (e.g., topic name) for routing, the packet data itself stays encrypted, and the broker has no way of looking into the encrypted data, as shown in Figure 4.

4.3. Information Dissemination

The broker is responsible for ensuring that messages are delivered to the correct subscribers. In general, the broker performs several operations, including connect, disconnect, publish, subscribe, and unsubscribe. These operations are available for both users and parking owners who are authorized by the security agent of the broker. The security agent is an entity that is responsible for the security evaluation of each request sent to the broker (e.g., checking the resource access permissions).

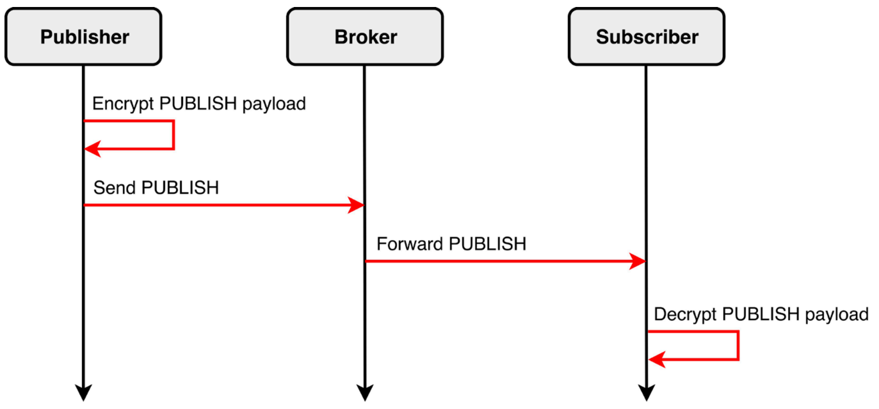

Figure 4. End-to-End packet payload encryption.

When a parking facility is granted permission to create a topic, it only needs to send a "publish" request that includes the topic name and the message in encrypted format. The parking owner can define a set of access-control policies associated with their topics to restrict topic access, based on user attributes. These policies can be formally defined as a conjunction of attribute conditions $\{cond_1 \wedge \dots \wedge cond_n\}$. Each attribute condition is in the form of $<name_x, op, l>$, where $name_x$ is the name of the attribute, op is a comparison operator, such as =, ≠, <, ≤, ≥, >, and l is the value of attribute x.

4.4. Finding a Vacant Parking Space

End-users are provided with a custom mobile application (Parking Application). This application enables them to find the available parking spaces near them, or near their final destination, get the right directions to the target parking spot, make a reservation, check the remaining parking time, and get notification when the parking time has expired.

First of all, the user is required to connect to the broker through the mobile app, which uses TLS. When the TLS handshake takes place, the client needs to validate the identity of the broker using its X.509 certificate. After the handshake process is completed, an encrypted communication between client and broker is established, and no attacker can understand the content of the communication. Next, the available car lots are displayed to the user, based on the user's current location or final destination. Each car park is considered as an individual topic. The users can subscribe to one or more topics, based on their needs. After subscribing to a certain topic, they would start receiving messages from the corresponding car lot. Each topic provides real-time parking information, including parking rate, number of available regular parking spots, number of vacant accessible parking spots, and total number of parking spaces. The end-user will be notified whenever there is an update on the parking information. Note that the received messages will be encrypted, and the user needs to decrypt them using the topic password.

4.5. Parking Reservation

For smart parking systems, the parking reservation service is a key feature. Our framework allows a user to reserve a parking spot in advance to guarantee a free spot by the time the user arrives. After determining the target car lot, the driver can send a message to the broker under a subtopic named *reservation*. For example, let us assume that the topic corresponding to the chosen car lot is called t_1, and the driver wants to reserve a parking space at it. Then he will need to publish a message under a certain topic, namely, t_1/*reservation*. The published message contains different information, such as phone number, license plate number, type of parking spot, and parking time. By default, the car lot is the only subscriber to this topic. Therefore, the car lot will be immediately notified whenever there is a reservation request, and will decrease its total number of free parking spaces accordingly. Note that each car park maintains its own database, which stores all reservation requests. When the driver arrives at the entrance gate, they must be checked to see whether or not they have a reservation. If they have one, the gate opens and they can park in any free space. Otherwise, they will be rejected. Thus, the entrance gate unit must query the reservation database to accomplish this task. This verification process may be done using the user's mobile parking pass.

5. Security Analysis

In this section, we analyze the security of the proposed framework against different cyberattacks, and how it can counter these attacks. We assume that the end-user's mobile device has a secure environment in which to perform cryptographic operations.

5.1. Phishing Attacks

The intension of this type of attack is to steal sensitive information, such as username, password, and credit card numbers, by masquerading as a trusted entity. The proposed framework is an anti-phishing mechanism because there is no exist for static username and password during the authentication phase. In addition, there is no existing for username and password update. When the TLS handshake takes place, both client and broker authenticate each other using their X.509 certificates. As a result, the broker is able to verify the identity of the client, and vice versa, with no credentials required.

5.2. Man-in-the-Middle (MITM) Attacks

A man-in-the-middle (MITM) attack is a common type of cybersecurity attack that allows a malicious element to insert itself into a conversation between two parties, impersonate both parties, and gain access to information that the two parties are trying to send to each other. In general, a successful MITM execution has two distinct steps—interception and decryption. The first phase intercepts the network traffic before it reaches its destination. The second phase decrypts any two-way SSL traffic without alerting the user or application. Therefore, we need to provide some method of authentication for messages in order to be secured against MITM attacks. To block and prevent the risk of MITM, we rely on TLS to exchange messages over a secure channel. In such a structure, a client and broker exchange certificates, which are issued and verified by a trusted CA. Since we are assuming that this CA is trusted and secure, then the certificates, issued by the CA, can be used to authenticate the messages sent by the owner of such a certificate. Thus, our framework relies on a mutual authentication mechanism to thwart both ends of the MITM attack.

5.3. Replay Attacks

In this type of network attack, the intruder captures valid network traffic and then sends the same data transmission to its original destination, posing as the original sender. It is obvious that this kind of attack requires the ability to intercept the network traffic, as well as the ability to perform a masquerade attack. Moreover, the intruder will be able to perform the attack, even if the packet payload is encrypted. Therefore, our proposed framework uses the TLS to create a secure communication channel between two parties. Consequently, no attacker can eavesdrop on any part of the communication. This enables our framework to resist replay attacks.

5.4. Broker Hijack

For whatever reasons, some people and small businesses may prefer to use a free public broker; however, using a free public broker comes with a price. It increases the risk of being compromised by malicious hackers. If any broker is compromised, it literally opens a gateway for the attacker to gain access to sensitive information, and we can easily lose the data confidentiality and privacy of the user. Our framework takes that into account, and provides end-to-end encryption of the application data. In this scenario, if attackers get control over the broker, they still cannot look into the data itself because the data is encrypted. Thus, user privacy and confidentiality of encrypted messages can be guaranteed.

5.5. Shoulder-Surfing Attacks

This type of technique (looking over the victim's shoulder) is commonly used to obtain confidential information, such as username and password. It can also be performed remotely, using hardware assistance such as binoculars. This makes the static username and password not good enough for authentication. To prevent a shoulder-surfing attack, we use X.509 client certificates, which allow the client to be authenticated before establishing a secure connection.

6. Performance Analysis

As discussed earlier, our framework relies on TLS to secure communications over the Internet; however, using TLS comes with a price, as with any security measure. The main cost with TLS is that of resource consumption (e.g., CPU and network bandwidth), which is significantly higher compared to plain TCP. There are two main sources of such overhead. The first source is the TLS handshake process, *i.e.*, the number of handshakes and the size of the messages transmitted in each handshake. The second source is related to the involved cryptographic operations while sending each message, *i.e.*, the cipher suite employed. Our design is based on the key observation that the pub/sub client only needs to establish a connection once per session, however, unlike with other protocols, such as HTTP, which require a connection to be re-established upon every request. In other words, the TLS handshake takes place only once in the lifetime of the client.

To evaluate the impact of using TLS on the CPU utilization of a pub/sub broker, we conducted an experiment on the Eclipse Mosquitto broker using 10,000 real clients. For this experiment, we compare the CPU utilization for scenarios with and without TLS. We install the Mosquitto broker on a 2.5 GHz-processor computing machine using 16 GB of RAM. Table 1 presents our experiment parameters. We connect the clients to the broker in batches of 100 clients per second, subscribe all clients to a unique topic per client, and finally each client publishes one message in 10 seconds. We conducted two types of experiment: with and without TLS. Each experiment was repeated five times and the average results were reported.

Figure 5 presents the result of our experiment. It is noticeable that once the TLS handshake phase is finished, the CPU consumption is very small, and not worth considering compared to the scenario without TLS. However, the worst-case scenario may happen when the broker faces frequent client reconnects, so that the TLS handshake consumes too much CPU. In such scenarios, the following alternative techniques could be used to minimize the consumption of resources.

6.1. Using Elliptic Curve Cryptography (ECC) Certificates

Using elliptic curve cryptography (ECC) certificates instead of RSA certificates could significantly reduce the computation overhead. ECC creates stronger security keys with shorter key lengths than RSA does, which makes it faster and more efficient to implement. Table 2 shows a comparison of key sizes between ECC and RSA in the provision of a certain equivalent security level. Because of the smaller key size with an ECC certificate, less data is transmitted during the TLS handshake. Therefore, ECC certificates require less CPU and memory, and increase network performance accordingly. Due to this, the ECC approach is more suitable for IoT devices, as it reduces computational time, as well as data transmitted and stored.

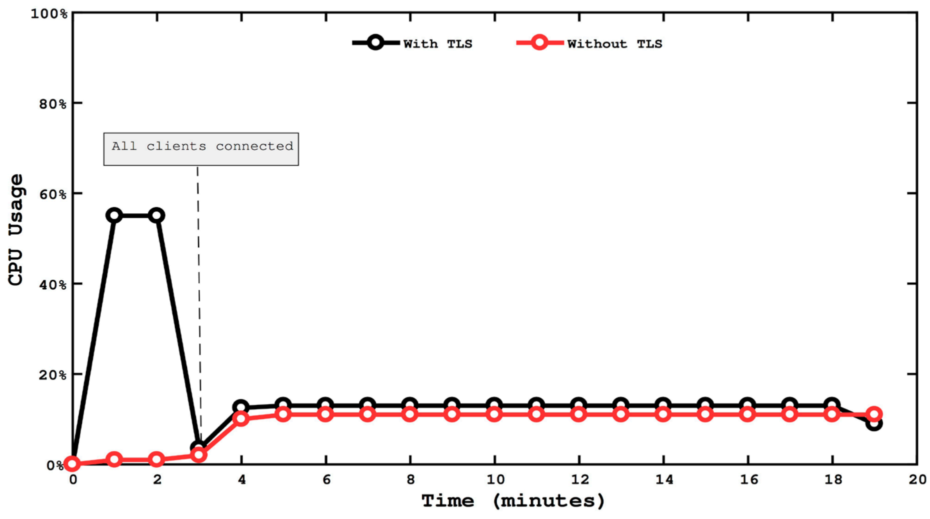

Figure 5. The CPU utilization for scenarios with and without TLS.

Table 1. Experiment parameters.

Parameter	Value
Pub/Sub Clients	10,000
Messages per second	1000
Connections per second	100
Quality of Service Level	1
Cipher suite	TLS_RSA_WITH_AES_128_CBC_SHA
Certificate key size	2048

Table 2. Comparison of RSA and ECC keys length.

Security Level	RSA Key Length	ECC Key Length	Ratio
80	1024	160	6:1
112	2048	224	9:1
128	3072	256	12:1
192	7860	384	20:1
256	15,360	512	30:1

6.2. Using Session Resumption

TLS session resumption is a technique that allows to reuse of an already negotiated TLS session after reconnecting to the broker, so that the client and broker do not need to do the full TLS handshake again. In short, this technique can be used to avoid a complete TLS handshake whenever a client reconnects, in order to reduce the overhead. Figure 6 shows what the TLS handshake looks like when using session resumption with session ID. Note that, the *ClientHello* message will contain extra bytes for the session ID that it wants to resume. Using this approach can reduce the overhead of establishing a new TLS connection from 1789 bytes to 332 bytes, *i.e.*, an 81.4% reduction in the size of the messages transmitted in each handshake [27].

6.3. Using Load Balancers

A load balancer plays an important role in traffic routing and traffic shaping for IoT solutions. One advantage of using it is the TLS offloading. In such a mechanism, expensive cryptographic operations take place on the load balancer, instead of the broker. This can increase the broker's performance remarkably. Figure 7 shows the architecture of using load balancers with brokers.

6.4. Using Broker Clusters

The pub/sub architecture depends on a broker as the central distributor of messages. To avoid the single-point-of-failure potential in such messaging systems, broker clusters are required. A broker cluster is a distributed system that acts as one logical broker. It contains multiple broker nodes that are typically installed

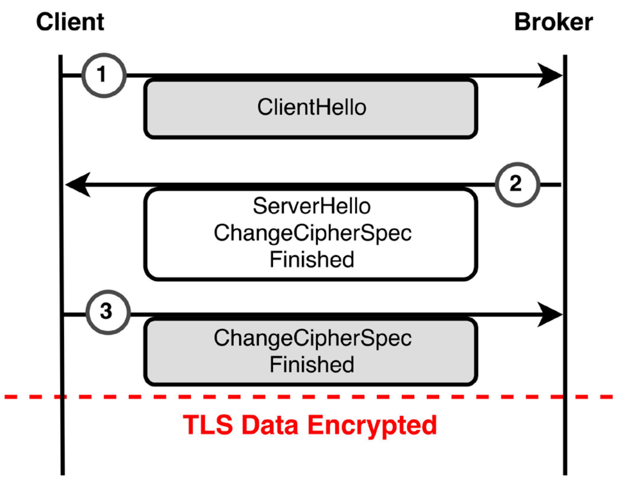

Figure 6. The TLS handshake using session resumption with session ID.

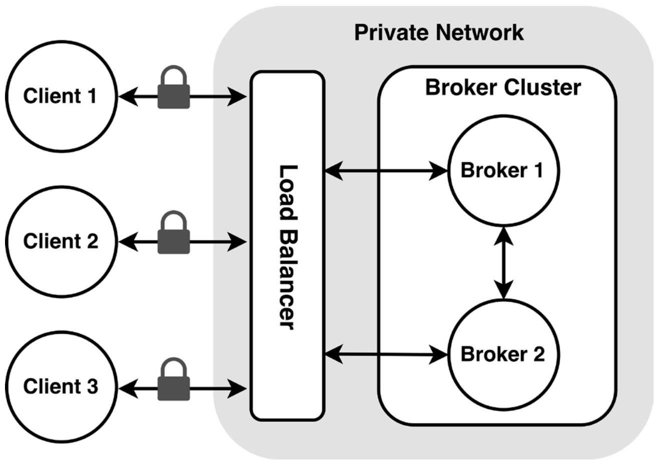

Figure 7. The relationship between load balancer and broker cluster.

on different physical machines, and connected over a network. Thus, clients can connect to any broker node to resume sessions, and this increases the availability of the provided services. Also, it can easily scale from a few broker nodes to thousands. Another advantage of using a broker cluster is that it is fully resilient and fault-tolerant in case of infrastructure problems.

7. Conclusion and Future Work

Cybersecurity is currently a growing issue in the IoT, which has tremendous benefits in smart city applications, such as smart parking systems. In this paper, we have proposed a secure and privacy-preserving framework for smart parking systems, utilizing the pub/sub messaging model. It ensures the confidentiality, integrity, and availability of real-time information by relying on two main security mechanisms—a secured communication channel via TLS, and end-to-end encryption for application data. It provides various services to the end-user, including real-time parking information dissemination, car park navigation, and parking reservation. Our framework is resilient to various security attacks, such as replay, man-in-the-middle, and chosen-plaintext attacks. It has a low overhead, due to the ECC-based certificates, which makes it ideal for securing IoT-constrained devices. In the near future, we will implement a prototype smart parking system, based on the proposed framework, on a large scale, in the real world, to evaluate its performance metrics more precisely.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Inci, E. (2015) A Review of the Economics of Parking. *Economics of Transportation*, **4**, 50-63. https://doi.org/10.1016/j.ecotra.2014.11.001

[2] Shoup, D.C. (2006) Cruising for Parking. *Transport Policy*, **13**, 479-486. https://doi.org/10.1016/j.tranpol.2006.05.005

[3] Thompson, R.G. and Bonsall, P. (1997) Drivers Response to Parking Guidance and Information Systems. *Transport Reviews*, **17**, 89-104. https://doi.org/10.1080/01441649708716974

[4] Rajabioun, T. and Ioannou, P.A. (2015) On-Street and Off-Street Parking Availability Prediction Using Multivariate Spatiotemporal Models. *IEEE Transactions on Intelligent Transportation Systems*, **16**, 2913-2924. https://doi.org/10.1109/TITS.2015.2428705

[5] Sakai, A., Mizuno, K., Sugimoto, T. and Okuda, T. (1995) Parking Guidance and information Systems. *Proceedings of the 6th Vehicle Navigation and Information Systems Conference*, Seattle, 30 July-2 August 1995, 478-485.

[6] Liu, Q., Lu, H., Zou, B. and Li, Q. (2006) Design and Development of Parking Guidance Information System Based on Web and GIS Technology. *Proceedings of the 6th International Conference on ITS Telecommunications*, Chengdu, 21-23 June 2006, 1263-1266.

[7] Polak, J.W. (1990) Parking Guidance and Information Systems: Performance and Capability. *Traffic Engineering and Control*, **31**, 519-524.

[8] Kotb, A.O., Shen, Y.C. and Huang, Y. (2017) Smart Parking Guidance, Monitoring and Reservations: A Review. *IEEE Intelligent Transportation Systems Magazine*, **9**, 6-16. https://doi.org/10.1109/MITS.2017.2666586

[9] Yoo, S.E., Chong, P.K., Kim, T., Kang, J., Kim, D., Shin, C., Sung, K. and Jang, B. (2008) PGS: Parking Guidance System Based on Wireless Sensor Network. *Proceedings of the 3rd International Symposium on Wireless Pervasive Computing*, Santorini, 7-9 May 2008, 218-222.

[10] Caicedo, F. (2010) Real-Time Parking Information Management to Reduce Search Time, Vehicle Displacement and Emissions. *Transportation Research Part D: Transport and Environment*, **15**, 228-234. https://doi.org/10.1016/j.trd.2010.02.008

[11] Mainetti, L., Palano, L., Patrono, L., Stefanizzi, M.L. and Vergallo, R. (2014) Integration of RFID and WSN Technologies in a Smart Parking System. *Proceedings of the 22nd International Conference on Telecommunications and Computer Networks*, Split, 17-19 September 2014, 104-110.

[12] Wang, M., Dong, H., Li, X., Song, L. and Pang, D. (2017) A Novel Parking System Designed for Smart Cities. *Proceedings of the Chinese Automation Congress*, Jinan,

20-22 October 2017, 3429-3434.

[13] Hanif, N.H.H.M., Badiozaman, M.H. and Daud, H. (2010) Smart Parking Reservation System Using Short Message Services (SMS). *Proceedings of the International Conference on Intelligent and Advanced Systems*, Manila, 15-17 June 2010, 1-5.

[14] Fraifer, M. and Fernstrm, M. (2016) Smart Car Parking System Prototype Utilizing CCTV Nodes: A Proof of Concept Prototype of a Novel Approach towards IoT-Concept Based Smart Parking. *Proceedings of the 3rd IEEE World Forum on Internet of Things*, Reston, 12-14 December 2016, 649-654.

[15] Funck, S., Mohler, N. and Oertel, W. (2004) Determining Car-Park Occupancy from Single Images. *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, 14-17 June 2004, 325-328.

[16] Kianpisheh, A., Mustaffa, N., Limtrairut, P. and Keikhosrokiani, P. (2012) Smart Parking System (SPS) Architecture Using Ultrasonic Detector. *International Journal of Software Engineering and Its Applications*, **6**, 51-58.

[17] Mathur, S., Kaul, S., Gruteser, M. and Trappe, W. (2009) ParkNet: A Mobile Sensor Network for Harvesting Real Time Vehicular Parking Information. *Proceedings of the MobiHoc S3 Workshop on MobiHoc S3*, New York, 18 May 2009, 25-28. https://doi.org/10.1145/1540358.1540367

[18] Masmoudi, I., Elleuch, W., Wali, A. and Alimi, A.M. (2017) Smart Drivers' Guidance System Based on IoT Technologies for Smart Cities Application. Springer International Publishing, Cham.

[19] Khanna, A. and Anand, R. (2016) IoT Based Smart Parking System. *Proceedings of the International Conference on Internet of Things and Applications*, Pune, 22-24 January 2016, 266-270. https://doi.org/10.1109/IOTA.2016.7562735

[20] Pham, T.N., Tsai, M.F., Nguyen, D.B., Dow, C.R. and Deng, D.J. (2015) A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies. *IEEE Access*, **3**, 1581-1591. https://doi.org/10.1109/ACCESS.2015.2477299

[21] Zhang, Z., Li, X., Yuan, H. and Yu, F. (2013) A Street Parking System Using Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*, **9**, Article ID: 107975. https://doi.org/10.1155/2013/107975

[22] Eugster, P.T., Felber, P.A., Guerraoui, R. and Kermarrec, A.M. (2003) The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, **35**, 114-131.

[23] Esposito, C. and Ciampi, M. (2015) On Security in Publish/Subscribe Services: A Survey. *IEEE Communications Surveys Tutorials*, **17**, 966-997. https://doi.org/10.1109/COMST.2014.2364616

[24] Eastlake, D. 3rd (2011) Transport Layer Security (TLS) Extensions: Extension Definitions.

[25] Turner, S. (2014) Transport Layer Security. *IEEE Internet Computing*, **18**, 60-63. https://doi.org/10.1109/MIC.2014.126

[26] Lindell, Y. and Pinkas, B. (2008) Secure Multiparty Computation for Privacy-Preserving Data Mining. Encyclopedia of Data Warehousing and Mining. https://eprint.iacr.org/2008/197

[27] Diro, A.A., Chilamkurti, N. and Kumar, N. (2017) Lightweight Cybersecurity Schemes Using Elliptic Curve Cryptography in Publish-Subscribe Fog Computing. *Mobile Networks and Applications*, **22**, 848-858. https://doi.org/10.1007/s11036-017-0851-8