A Low-cost Vehicle Tracking Platform using Secure SMS

Rune Hylsberg Jacobsen[1], Drini Aliu[1] and Emad Ebeid[2]
[1]*Department of Engineering, Aarhus University, Finlandsgade 22, Aarhus, Denmark*
[2]*The Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Campusvej 55, Odense, Denmark*

Keywords: Location-based Services, Low-cost, Vehicle Tracking, Short Message Service, SMS, Security, Internet of Things.

Abstract: This paper investigates the possibility of elevating SMS (Short Message Service) to support a cloud-based vehicle-tracking platform. A secure application protocol over SMS is introduced to circumvent the security issues that may succumb the SMS. We propose a cost-effective Internet of Things (IoT) solution to countries dominated with GSM mobile infrastructure taking into consideration technology usage and the mobile network infrastructure available. A commercial off-the-shelf (COTS) IoT device such as the Raspberry Pi single board computer is depicted relying on GPS-GSM technologies envisioned as the in-vehicle device. Furthermore, a cloud web-platform is built involving the de-facto modern web-applications standards and carefully tailored concerning the security aspects. The cost-effectiveness of our solution results from the use of COTS components, open source software, and cheap SMS subscription packages.

1 INTRODUCTION

Outdoor positioning and object tracking are well-established disciplines in science and engineering. In this regards, the Global Positioning System (GPS) has been a great success story (Bajaj et al., 2002) and a large number of tracking systems has been developed based on GPS technology. However, the widespread use of GPS-based object tracking is hindered by high system cost. This has led to the study of low-cost systems (Hasan et al., 2009). Even more problematic is people's concern with location-tracking services (Barkhuus and Dey, 2003). A malicious user can get unauthorized access to location updates thus disclosing the position of an object or a person.

The work presented here is inspired by an urgent need for a potential low-cost IoT infrastructure in countries dominated by 2G mobile infrastructure and where opportunities of using 3G communication and beyond have not yet been fully exploited (Nielsen and Jacobsen, 2005). In Macedonia, only a few mobile network operators have upgraded to support 3G technology with a wide coverage area. However, almost the entire young generation is owning a smartphone, which has profoundly changed how they socially inter-communicate. Consequently, the habit of Short Message Service (SMS) has been replaced with messaging apps like WhatsApp, Viber, and Line, which require Internet connectivity. Because of this, mobile network operators have modified prices of the different subscription packages offered. This has resulted in an extremely low price for SMS communication and costly packages for 3G data usage. Essentially, these SMS package deals are promised with an unlimited number of SMS as long as they are within the mobile network of the Macedonian mobile network operator. As a result, the price cut has offered new opportunities to provide cost-effective IoT solutions that revolve around the use of mobile communication technologies with high coverage such as SMS. Unfortunately, Global System for Mobile communications (GSM) has a number of security flaws (Toorani and Beheshti, 2008a) and it can be concluded that the system is not living up to current security standards. To address the security and privacy concerns of end-users, SMS needs to be elevated with added security measures to circumvent the inherent shortcomings of SMS in today's IoT solutions.

In this paper, we propose a low-cost, secure application protocol over SMS that connects with a cloud-based vehicle tracking system. The system is based on an end-to-end secure application protocol that uses a two-way communication between Mobile Stations (MSs) and a Data Concentrator (DC) entity connected to the cloud. The system offers an Application Programming Interface (API) that enables MSs to push

Jacobsen, R., Aliu, D. and Ebeid, E.
A Low-cost Vehicle Tracking Platform using Secure SMS.
DOI: 10.5220/0006366201570166
In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 157-166
ISBN: 978-989-758-245-5

location information to the cloud. MSs are authorized based on a *whitelist* concept. A DC is authorized by the cloud service by using a light-weight implementation of the OAuth 2.0 protocol. The accompanying web platform will be an IoT cloud service where van companies may sign up, authenticate themselves and manage their vehicle data.

2 RELATED WORK

Real-time tracking systems have been a field of interest for many researchers during the past decades. Tracking systems have been introduced for various purposes like vehicle position tracking systems, anti-theft systems, fleet management systems and intelligent transportation systems. Some of the proposed systems involve the use of SMS communication hence providing cost-effectiveness due to the low rate for SMS and the restricted availability of later generation of mobile network infrastructures (i.e., 3G and 4G in developing countries).

Verma and Bhatia presented a tracking system using GPS and GSM technologies (Verma and Bhatia, 2013). However, the system neglected to cover security aspects of using SMS service to send location data which may be considered as compromising privacy. In another study, Lee et al. depict a hybrid vehicle tracking system (GPRS alternative) able to send data over GSM (Lee et al., 2014). Although the study includes many technical details about how data is stored and later visually presented to the users, the authors seem not to have addressed security and privacy concerns imposed when sending private information over SMS. A recent study reports on a vehicle tracking and monitoring system built on top of the GPS-GSM technology emphasizing the provided security against possible vehicle thefts but failing to address the security perspective of the GSM networks and SMS itself (Kumar et al., 2016). An extensive vehicle tracking and monitoring system using a single-board computer (Raspberry Pi) with an SIM900A module have been proposed and designed (Shinde et al., 2015). Besides the exciting features, they introduce such as interfacing a file system of a single board computer from a smartphone to select and load a pre-fixed path to ensure secure traveling, SMS security aspects, and proposed rectification mechanism are not depicted. In common, the above-referenced studies all rely on the security provided in SMS, which is known to be insufficient (Toorani and Beheshti, 2008a).

Some studies report on solutions that aim to provide authentication, integrity, confidentiality, and non-repudiation for SMS. SMSSec is a protocol founded on a combination of asymmetric-key and symmetric-key cryptography that relies on a two-factor authentication strategy for their initial handshaking phase (Lo et al., 2008). A major drawback of the protocol is the implications produced by their inclusion of Rivest, Shamir, & Adleman (RSA) encryption scheme with 2048-bit keys resulting in ciphertext block that exceeds 140 bytes. Thus, two SMS messages are needed for a single protocol message in the initialization of the handshaking phase. For their communication back from the server, a symmetric technique is used with smaller key-size to ensure that the message sent back fits within a single SMS message.

Another proposed solution called SMSS uses a certificate-based protocol that suggests following typical rules of Public Key Infrastructures (PKIs) by issuing of a unique public-private key pair for each user (subscriber) and the involvement of a Certificate Authority (CA) (Toorani and Beheshti, 2008b). It applies an encryption scheme based on Elliptic Curve Cryptography (ECC) that uses public keys for the establishment of a shared secret key. Moreover, each time an entity tries to establish communication, it must preconditionally query the OCSP server completely delegating validations to a trusted server. However, this appears to be done in an insecure way using plain text via SMS. The system has a number of vulnerabilities. The OCSP server's original address can be spoofed. The SMS response, that the source entity expects to receive after trying to establish and communicate with the destination entity, can be forged. An attacker can derive the signature and can replace the public key of the destination entity to be used by the originating identity for encrypting its messages. This induces problems in decrypting of the incoming messages since they were not rightfully encrypted. Furthermore, the proposed protocol would require that network operators modify their existing infrastructures.

Other protocols are based on the identity-based cryptosystems, where users need to communicate with a trusted agent providing a Private Key Generator (PKG) whose master key must be kept secret. The requirement to have an authentic CAs public key for verifying certificates in a solution as SMSS is replaced by the requirement to have authentic PKG's system to bootstrap the ID-based cryptosystems (Jacobsen et al., 2015). The proposed ID-based encryption scheme will require network operators to adapt their infrastructure as well as require them to preload and configure secret information for their corresponding private key within the Subscriber Identity Module (SIM) cards before distributing them to the

subscribers. The proposed ID-based scheme surely provides a simplification of key distribution since all public keys can be derived from the identities of the users.

The SecureSMS protocol proposed by Saxena and Chaudhari makes use of two trusted entities within the workflow: the Authentication Server (AS) and CA (Saxena and Chaudhari, 2014). The protocol uses several messages (5-6 messages) to establish a secure connection for specified period of time. They also state that the generation of the delegation key, that the AS computes for a particular timestamp, uses a shared secret key between the MSs and the AS, which has to be embedded onto the SIM at manufacturing time.

While some tracking systems based on the SMS protocol have been proposed they rely on the security inherited from GSM to protect location data. In contrast, our system advances security beyond the security provided by SMS and apply this to a cloud-based vehicle tracking platform.

3 SYSTEM DESIGN

Figure 1 shows a conceptual overview of the proposed vehicle tracking system. The system consists of Mobile Stations (MSs), a Data Concentrator (DC), and an IoT cloud service employing a RESTful web platform where van companies manage their data, and public users can access, search, and filter these data with the consent of van companies operating a fleet of vehicles. The MS is an IoT device which reports GPS measurements to its assigned DC. The DC is part of an IoT cloud service platform used by van companies to access the data reported from their devices, monitor their MSs in real-time, manage their van schedules and manage their data availability to the public endpoints in different formats.

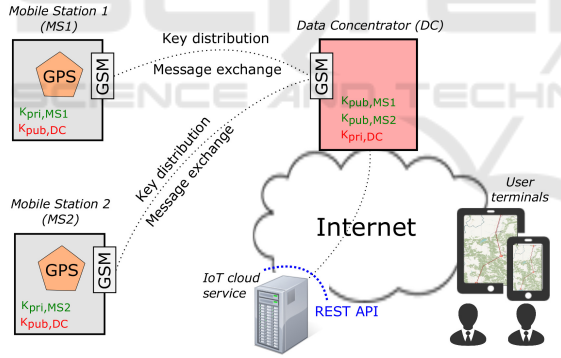

Figure 1: Conceptual system overview.

3.1 Mobile Station (MS)

The MS of the system is the entity envisioned to be a consumer product that would be available for purchase and supposed to work with ordinary SIM cards. Each MS will be bound to a DC device. The MS will communicate with DC by applying a secure SMS protocol.

3.2 Data Concentrator (DC)

A DC is a physical IoT device equipped with a GSM module. It is able to receive the GPS measurements via SMS from the particular MS that are assigned to it. The DC resides on the edge of the cloud. A Representational State Transfer (REST) API offers access to an application that will be deployed to the cloud, which will serve as the core endpoint where the DC will be pushing the received data to over HyperText Transfer Protocol (HTTP) over SSL/TLS i.e., HTTPS. The system can be scaled by adding more DCs (horizontal) and the IoT cloud service can be scaled with respect to CPUs and memory (vertical) as the rate of the incoming requests increases. In order to make our system flexible enough, so that it can support load balancing after the first establishment, we introduce a special type of message that will serve to regulate this.

3.3 Communications

The SMS interface depicted in Figure 1 shows a two-way communication, where data flows between the DC to the MS will be less frequent as designed in the protocol. The DC acts as a proxy gateway towards the Internet whose duty is to communicate the location data to a pre-defined web endpoint, each time they would receive an SMS from a particular device. By pushing and storing this location data from a particular MS device to the cloud service, we make it possible for users to access this information in real-time. The pushing of location data received by SMS to the cloud will be performed via HTTP endpoints, a well-tailored web-application following the REST architectural style (Fielding and Taylor, 2000).

The DC will be pre-processing the received SMS messages and is envisioned to hold the logic of distinguishing whether the inbound SMS originates from an authentic source before location data is pushed to the cloud. The REST API will be extended with a token-based authentication layer by implementing a light-weight security protocol reusing design practices from available authentication and authorization frameworks to secure the DC to cloud communication. It is a “must” to hinder any possibilities of other

Internet-enabled appliances to push the location data and ensure that the entire data pushing process will be done in a secure and reliable way.

4 PROTOCOL DESIGN

In order to provide a sufficient level of security, another layer of security must be provided on top of SMS. We have designed our protocol to meet the limitations of the constrained payload size of a single SMS message (limited by the constraints of the signaling protocol to 140 octets). In order to keep our messages within the design goal of using as few messages as possible, we have to carefully choose between alternative encryption schemes and key sizes. Consequently, trade-offs have been made with respect to the efficiency, usability, and security we are targeting to achieve for our platform. Below we present the essential system requirements followed by a description of the protocol design.

4.1 Requirement Analysis

The secure SMS protocol used in our vehicle tracking platform must support confidentiality as well as message authentication and integrity. For the message formats, the message content of an SMS protocol messages is divided into multiple fields to accommodate the various security checks required. One byte of the message denotes the message type. The message type is an indicator of how to process the preceding information. Each MS must be assigned a unique device identity (ID), represented by 16 bits, and configured with a binding to a DC. The device ID value will be carried in all messages sent from the MS. The communication between an MS and their assigned DC will be encrypted. After each encryption of the plaintext of the constructed messages, the ciphertext will be encoded. The encoding will ensure that no bit sequence of the generated ciphertext will occasionally be interpreted as an ending symbol in the GSM communication. Furthermore, a message must contain a timestamp introduced in Unix epoch format. Finally, the protocol should support protection against replay attack as well as the ordering of messages by using the timestamp.

4.2 Protocol Messages

Two sets of messages define the communication between the MS and the DC. In the following, the message types of our protocol are introduced.

4.2.1 Mobile Station to Data Concentrator

The MS supports three types of messages in its communication with the DC: `HELLO`, `REPORTING` and `ALIVE` messages. `HELLO` messages are sent upon powering on the MS. It is a special message type notifying the system that the MS has been powered on and is ready to begin tracking. As a unique message type, a significant role of the `HELLO` message is that it requests a `SESSION_TOKEN` message from the system, which carries a *token* to be used in the communication with the IoT cloud service. `REPORTING` messages, which are sent right after receiving a `SESSION_TOKEN` message from an authentic DC carries the location data. These `REPORTING` messages are sent periodically every 5 minutes (default configuration) unless the MS has paused its movement. Finally, `ALIVE` messages are sent when a delay of sending the `REPORTING` messages occurs because the MS has paused its movement. A tolerance of 5-10 meters is used.

4.2.2 Data Concentrator to Mobile Station

The DC supports three message types in its communication with the MSs: `SESSION_TOKEN`, `CHANGE_DC`, and `CHANGE_PUB_KEY`. The `SESSION_TOKEN` messages are mainly responses to `HELLO` messages from the MSs. A `SESSION_TOKEN` message will be returned to the sender, after having received a valid `HELLO` message checked against a *whitelist* of the registered phone number. Finally, `CHANGE_DC` message is a type of a `SESSION_TOKEN`, which alongside distributing the token to be used for the reporting session, also orders the receiving MS to change its communication to a newly assigned DC. The `CHANGE_PUB_KEY` messages are designed to be sent from the DC to disseminate the global public key in use for the MSs.

4.3 Message Exchange

ECC is established as our preferred encryption scheme since it can support the same level of security as RSA but with much shorter key lengths. There will be a public key ($K_{pub,DC}$) of the DC distributed to all MSs and the associated private key ($K_{pri,DC}$) resides on the DC. For communication from the DC to MS, each of the MS will have a unique secret key pre-installed at provision time. This means that the DCs or the cloud infrastructure must possess the corresponding key-value pairs for each of the MS in the field.

Figure 2 illustrates the scenario where a van driver powers on an MS, sends a `HELLO` message, receives a token from the DC and starts reporting location data periodically. In Figure 3, we show the interaction

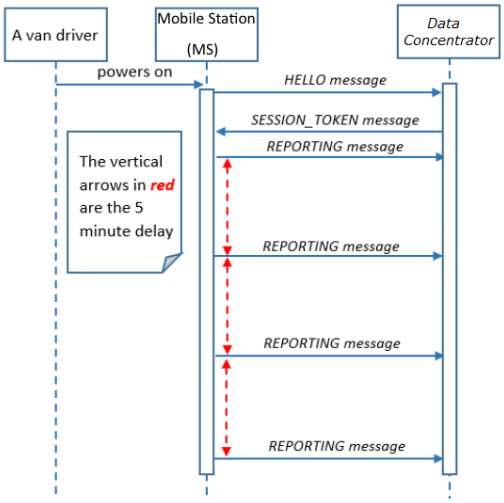

Figure 2: A normal activity for MS movement. After the initialization phase, the MS starts periodic reporting.

between an MS and a DC for a moving vehicle that stops for 6-7 minutes when trying to wait for more passengers at the stop. As one may see, every mes-

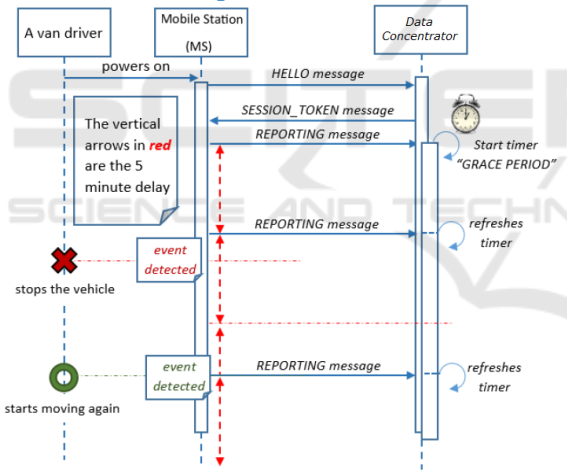

Figure 3: MS is powered on. The vehicle starts moving but then stops for 6-7 minutes.

sage sent out from an MS is done in a fire-and-forget manner, without any acknowledgments from the DC. This is also the case with `ALIVE` messages. The scenario shown in Figure 3 will require an MS to detect these events accordingly and take decisions on itself. An internal algorithm is able to calculate this by using the GPS data and the system internal clock.

5 IMPLEMENTATION

In the following, we will introduce the key implementation aspects of the MS, DC, and the IoT cloud service.

5.1 Mobile Station (MS)

For the MS hardware, a Raspberry Pi 1(B+) single board computer platform is chosen because of its low cost and high degree of flexibility. The hardware is extended with a red-green-blue (RGB) LED, which will elevate the ease of use. The visual light signals provide a simple user interface of the MS to signal status of the device (booting, connected/not connected etc.). The MS hardware is equipped with a GPS module to allow the hardware to determine position based on GPS satellites. Additional geo-data can be used to refine the accuracy of the estimation algorithm such as the current driving speed to be sent alongside the vehicle position. The Raspberry Pi device keeps track of time so that the timestamps can be correct when pushing data to the IoT cloud service. The MS system time will be kept up to date and synchronized periodically.

As for the decision of this module, we went with a reasonably priced and yet a small GPS component like the UBLOX-NEO-6M GPS module. It is a complete GPS smart antenna that includes an embedded antenna (1575R-A E). This module comes with a fast time-to-first-fix, one-second navigation update, and low-power consumption. It matches with the capability of the sensitivity required for vehicle navigation, as well as other location-based applications (ublox, 2011). For the General Packet Radio Service (GPRS) module, we chose to integrate the SIM800L hardware building block (SimCom, 2013) in our system design because of its ease of use and low cost.

Figure 4 pictures the hardware platform for the MS. The device runs the Linux OS Raspbian with

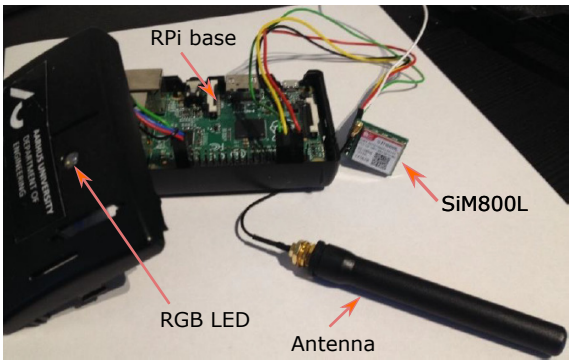

Figure 4: Raspberry Pi (RPi) with an RGB LED light and SIM800L attached.

Python and the programming of the platform can be done right away after installing the necessary software libraries. In order to be able to facilitate communication, it is required to create a thread that will run in the background and fetch geo-data provided by the *gpsd* service that has been initialized. The variables to be extracted and then used for the MS are the latitude, longitude, current speed and the global time. Figure 5 shows a typical console output resulting from the GPS output.

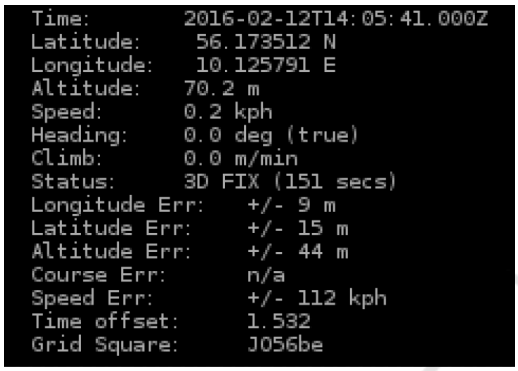

Figure 5: Example of a console output results from the *gpsd* service.

5.2 Data Concentrator (DC)

The DC is based on the same hardware platform as the MS. The main loop of the application will be reading SMS messages without using the silent mode read because a single phone number (SIM card), that will be attached to a GSM module, will only serve the DC functionality. The GSM modules are constrained devices with limited processing capabilities and can only be accessed synchronously through the interface. They support up to 115,200 bits per seconds but also recalling that the operational commands issued take also time to be performed. A starting point could be to poll the GSM module to read arrived messages, each 5 seconds and read its result. Then right after reading, to delete the read messages avoiding a message queue to build up. Another important component of the main application will be the software component that will be responsible for pushing the received content through the specified endpoints. This will be done through asynchronous requests from a library named *requests*. The responses will be expected via a callback. If a particular request fails to connect with the servers, the request will be queued to be retried later.

There are different ways a DC can send `SESSION_TOKEN/CHANGE_DC` messages. Due to simplicity, we have chosen a solution where the DC listen to the request callbacks whenever a `HELLO` message gets a reply and interpret the particular HTTP response. We must be careful building our application using a non-blocking way of pushing the location. Sequentially executing a request and blocking until the server response has arrived is not a very flexible, nor a scalable solution. To maximize the throughput and utilize the DC's resources efficiently, the application will have to asynchronously execute requests and handle them via a callback function. For this, we use a software library like *grequests*. This library offers asynchronous capabilities of making HTTP requests and specifying their connection timeout variables, retry number similar to using the built-in library *request*. For all the requests towards the cloud, their connection timeout would be set at 2 seconds. A request that fails to get a response will be re-tried twice, also considering a maximum pool of concurrent requests would be set accordingly.

The DC maintains a *whitelist* of authorized phone numbers to check if messages received comes from an authorized MS before the corresponding data is pushed to the IoT cloud service. This list will be retrieved from a dynamic endpoint: `/tel/<country_abbreviation>/whitelist` where the country abbreviation variable (e.g. DK, MK) can be any country. The process of retrieving the whitelist will be among the first instructions of the DC when it is booted. This will be done by interacting with the cloud from the specified variables denoting from which server port the REST API will be available. When the whitelist is retrieved, it has to be kept up to date since additional MSs can be continuously registered/de-registered through the web platform. A publish-subscribe paradigm would fit best for this purpose, utilizing a messaging service technology to notify DCs for every change that happens after their first fetch. However, this latter part was not implemented in our prototype.

5.3 IoT Cloud Service

The IoT cloud service offers a REST API for handling incoming messages originating from the MSs. These messages are routed through and authenticated by the DC. The message exchange uses HTTPS. The SSL/TLS protocol ensures mitigation of replay attacks and provides confidentially to content that is pushed through the specific resource endpoints.

The IoT cloud service is developed by using the Flask micro web framework (Grinberg, 2014). A key challenge is to limit access of which of the online enabled devices will have the privilege of using the IoT cloud service to push the geo-data. DCs are the only

devices that are authorized to communicate with the REST endpoint of the cloud service.

To support authorization of DCs, we have implemented a simplified version of the OAuth 2.0 protocol (Hardt, 2012). Starting from the roles of OAuth 2.0, we do not need a *client* because the application making requests on the protected resources will be a single entity i.e., the program installed on each of the DCs. The *resource server* and *authorization server* can be brought up as a single entity in our case. Thus, from the protocol flow of OAuth 2.0 (Hardt, 2012) the authorization request (step A) and the authorization grant (step B) between the *client* and the *resource owner* are skipped. For the authorization grant type, we are choosing the resource owner password credentials anticipating a high degree of trust between the resource owner and the client, which are fully controlled in our system. We can choose to rely on this scheme so future partners can scale up the IoT cloud-based backbone of our system just by logging in, to find the particular valid token to set their DC up and running.

To enhance security beyond the level provided by HTTPS, we decided to create our own token-based authentication as an extra layer to enhance our protection. The token issued by our IoT cloud platform, will be generated by a cryptographically secure random generator and will have a length of 95 bytes. With a Base64 representation, this translates to 128 ASCII characters. Each of the generated tokens will be stored in a database with a 24 hour expiration time. These tokens will be used by the DC as means of identifying itself with the cloud service and gain the authorization rights to push geo-data to it. The token's value is intended to be installed and deployed to the DC. The system will automatically refresh expired tokens by creating new ones each time they get expired. This is accomplished by a daemon mechanism that checks token validity against their creation timestamp. A DC that tries to push geo-data via the POST method to an endpoint using a just expired token (the previously valid token) would get a reply the new valid token and retry the same exact request with the obtained token.

6 EVALUATION

This section analyzes various aspects of the proposed protocol including security such as authentication, attack resistance, and performance.

6.1 Authentication Between MS and DC

In our proposed SMS protocol, both the MS and the DC can authenticate each other for any of the message types designated to them. The authenticity of messages arriving from a DC is achieved through the secrecy of the encryption keys. We can rely on authentication by encryption method, since we can ensure that only our IoT cloud service possesses the corresponding public keys corresponding to the secret keys deployed in each of the MS and the fact that they are completely unique from each other. It is impossible to forge authentic messages without knowing: 1) the phone number being used for the DC, 2) the MS's phone number the attacker is targeting and whether they are associated together, and 3) the particular public key used to encrypt the messages flowing from the DC to the designated phone number being used with the MS. Any attempt to forge an authentic message will fail due to the cryptographic integrity. We are also protected by the prior integrity check offered from the encoding technique and the posterior timestamp integrity.

To ensure authentication for messages that arrive in the DC, our protocol relies on two important techniques to tackle message authenticity issue for different message types. The first one, the inclusion of long-term random 10 digits number that is shared between the user (van company) through the web platform as soon as a phone number is successfully verified via the two-factor authentication verification process. This is the main technique that basically limits the attacks to send out `SESSION_TOKEN` message containing the valid token. The second setting, mainly used to ensure authenticity for the `REPORTING` and `ALIVE` messages that are sent from an MS, is concerned with the 50 byte token that the MS is envisioned to be obtained from the `SESSION_TOKEN` messages. Because we are sure that only the phone number that actually sent the valid `HELLO` message will receive the SMS message, respectively the particular `SESSION_TOKEN` message. Any attempts to snoop and derive the token from the `SESSION_TOKEN` content, from intermediary transmission layers in the GSM infrastructure will end in a failure, since the whole message content is strongly encrypted with ECC cryptography, providing full confidentiality.

6.2 Resistance to Attacks

SMS messages are easily intercepted during its transmission which thwarts the privacy of location data that flows out of the MS. Our platform ensures full confidentiality and no SMS message content is dis-

closed because the whole two-way communication is encrypted. That being said, the communication from an MS to the DC is encrypted with a public key, which is the same key deployed to all of the MSs. The only party that possesses the secret pair of this global public key deployed, is the IoT cloud service. The communication from the DC towards the MS (phone number being used) is encrypted with a different public key that is unique for each MS.

Replay attacks are another security challenge to be addressed. The timestamp field introduced in the structure of our proposed protocol will serve as a mechanism to prevent replay attacks in both ends of the communication. MS will only accept and execute valid message from the DC whose timestamp is bigger than the timestamp created upon sending its `HELLO` message on power-up and neglect any messages that arrive earlier than this `HELLO` message. After having received any valid message from an authentic DC, the timestamp variable is overwritten with the particular timestamp when the message was issued making replay attacks of previously issued commands impossible.

Along the same lines, the DC will have to store the last-received `HELLO` message timestamp and a mechanism is employed to check the replay of previous `HELLO` messages that were used on earlier trips. In this way, we will prevent previous `HELLO` messages to generate another valid token and issue a `SESSION_TOKEN` reply, making the `HELLO` message of a one-time use type.

6.3 Performance Evaluation

We have examined the computational overhead required for the encryption and encoding techniques performed on a low-cost and resource-constrained device as the Raspberry Pi single board computer. The tests were done using a Python built-in module called *timeit*, which avoids common traps for measuring execution times providing more accurate results than other techniques of time measurement. The encryption tests are done with a 160 bit key size over the p160 elliptic curve provided by the *seccure* crypto-library. Table 1 lists the results from tests using p160 elliptic curve and the Base92 encoding/decoding technique with the MS hardware platform.

We also tested the time required to read the contact phone number via the SIM800L module, where the user is supposed to save the 10-digits secret number, retrieved from the web platform. That is done by the issuing `AT+CPBF=<contact_to_find>` that initiates a find operation on the specified storage of the phone book entries. It takes 2.1 seconds in average and up to 3 seconds to read the phone number of a contact.

Table 1: Records of time used to encrypt/decrypt and encode messages.

Tasks	Average time
Encrypt + encode `HELLO`	113 ms
Encrypt + encode `REPORTING/ALIVE`	124 ms
Decode + Decrypt `HELLO`	64 ms
Decode + Decrypt `REPORTING/ALIVE`	70 ms

6.4 Web Service Testing

The experimental setup to test the IoT cloud service are performed using mock-up data to emulate MS movement and to see that the endpoints are correctly receiving the information from the DCs. The tools used for testing includes Postman, Twilio, and Nexmo. Postman is a browser plugin to test the REST API endpoints by allowing the construction of requests quickly, save them in collections and analyze the responses. Twilio offers a cloud-based messaging platform that enables the use of virtual phone numbers to potentially substitute the physical DCs. Nexmo is another cloud-based messaging platform we used to send mock-up SMS messages from an MS with a spoofed originator's address and check how they are received by the Twilio platform. By using the mentioned protocol messages formats, the authentication mechanisms, location update mechanism, have been successfully verified. However, a thorough documentation of the testing of the IoT cloud service is beyond the scope of this paper.

7 DISCUSSION

The low-cost motivation of the project was realized by choosing cheap Commercial off-the-shelf (COTS) products and open source software, assembling them into a prototype for an MS. The cloud infrastructure was built based on a software development process where the requirements elicitation was done through observations of the current transport industry environment in Macedonia.

A main goal of the work was to design a secure and reliable application-level protocol that would elevate mature technology like GSM into a value added service for supporting vehicle-tracking systems. We have successfully identified and analyzed the weaknesses that this service inherits and circumvented its drawbacks by enhancing the security of the de-

sign. The choice of ECC over RSA allows our protocol to be easily adapted and applied even to constrained devices with limited computing capabilities. The Base92 encoding technique proved to be more efficient in the production of shorter encoded ciphertext than traditional alternatives. The message protocol fields of the shared 10-digit number and the reporting session token can be of variable length, thus carefully tailored according to the type of the application and the MS capabilities. The token length of 50 bytes was set as a default for our platform and assessed to provide a sufficiently good level of security.

A REST API for location data to be pushed to the cloud infrastructure. The DC can, through the proposed API, maintain their current valid token configuration via a fully automated mechanism. Our system can be applied in different countries without the necessity to physically own DC devices equipped with local phone numbers. However, the possibility of replacing a DC with a cloud service like e.g., a rented Twilio phone number, imposes modifications to our cloud infrastructure including the REST API. Not only implying that the logic of decoding and decryption must be done on cloud's end, but also introduce a completely new Universal Resource Identifier (URI) approach how the messages are pushed through the endpoints. The latter problem arises because Twilios platform only allows us to set a single dedicated URI where they will retrieve the SMS-related data either by POST or GET method. Therefore, to support that we must create a unique universal endpoint especially dedicated from this alternative, where Twilio will be transmitting information upon SMS reception on their side, directly to our servers. In this case, the phone number in our application will not be distinguished from the URI segment, but rather from their passed parameter 'From' alongside other important parameters of the SMS message that is pushed from Twilio's servers.

As part of the project, an online survey was conducted to a group of people from different cities around Macedonia that are traveling via inter-city lines that van (bus) companies offer. The total number of the survey responses collected was 53, from where 12 people within the 16-20 age group, 39 belonging to the age group of 21-25 and 2 people within the 26-30 age range. All of the interviewees stated that they own a smartphone. The majority of them (77.4%) answered that they often use the navigational apps on the smartphone to find their way around cities and 71.1% of interviewees would be willing to share their location with these navigational apps. A considerable amount of interviewees (37.5%), that belong to the latter group, have privacy concerns and with respect to sharing their location information with apps or websites that have nothing to do with navigational services.

When asked how do they approach finding information about their current available trips, the majority of the interviewees (60.4%) claim that they personally call the van companies to ask about their offered lines, schedules, and location information. However, the majority of the interviewees (65.6%) of the group that claimed to call van companies by phone, complain about their office telephone lines being busy and 25% of them think that the location information the van companies give over the phone can be misleading. All of the interviewees have been asked about how they deal with issues when the vehicle of the particular trip is late according to the schedule. In this case, 47.2% of them would immediately call their office phone number for explanation and 30.2% would furthermore try and contact the vehicle driver of the trip asking for status and plans.

As the last question, they were asked how they would like to be informed about the arrival times of the van (buses) if a web platform existed where they could subscribe to receive notifications about expected arrival. The answers indicate a lead of the SMS communication by 67.9%, followed by email (22.6%) and lastly voice-calls (9.4%).

8 CONCLUSIONS

In this paper, we have presented a low-cost and secure vehicle tracking platform to be used in countries dominated with GSM mobile infrastructure. The system is based on a security-enhanced messaging protocol delivered over SMS. The vehicle tracking platform employs a Mobile Station (MS) based on a single-board computer that accesses an IoT cloud service through a Data Concentrator (DC) residing on the edge of the cloud.

The proposed platform solves the problems of finding the reliable, up-to-date schedules and unambiguous locations of where the vehicle picks up the passengers. The system will offer a search capability presenting passengers with choices over current alternatives and our platform will be aware of all the available trips that intercept that route (city). Hereby, the platform would significantly reduce the number of phone calls a van company receives as well as their employees (drivers) phones.

The steps of achieving the low-cost motivation of the project were realized by choosing cheap COTS products. Future work will include an investigation of the Raspberry Pi Zero hardware as a potential plat-

form to drive down the cost of the MS further. We will investigate further the potential seamless integration with cloud-based platforms such as Twilio and Nexmo providing virtual phone numbers to possible fully substitute the function of the DC. This could potentially also improve the scalability of the platform as more virtual phone numbers can be activated as the number of MSs increases.

REFERENCES

Bajaj, R., Ranaweera, S. L., and Agrawal, D. P. (2002). GPS: location-tracking technology. *Computer*, 35(4):92–94.

Barkhuus, L. and Dey, A. K. (2003). Location-based services for mobile telephony: a study of users' privacy concerns. In *Interact*, volume 3, pages 702–712. Citeseer.

Fielding, R. T. and Taylor, R. N. (2000). Principled design of the modern web architecture. In *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pages 407–416, New York, NY, USA. ACM.

Grinberg, M. (2014). *Flask web development: developing web applications with python*. O'Reilly Media, Inc.

Hardt, D. (2012). The oauth 2.0 authorization framework.

Hasan, K. S., Rahman, M., Haque, A. L., Rahman, M. A., Rahman, T., and Rasheed, M. M. (2009). Cost effective gps-gprs based object tracking system. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 18–20.

Jacobsen, R. H., Mikkelsen, S. A., and Rasmussen, N. H. (2015). Towards the Use of Pairing-Based Cryptography for Resource-Constrained Home Area Networks. In *2015 Euromicro Conference on Digital System Design*, pages 233–240.

Kumar, B. H., Tehseen, S. F., Thanveer, S., Krishna, G. V., and Akram, S. M. (2016). Vehicle monitoring and tracking system using gps and gsm technologies.

Lee, S., Tewolde, G., and Kwon, J. (2014). Design and implementation of vehicle tracking system using gps/gsm/gprs technology and smartphone application. In *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pages 353–358. IEEE.

Lo, J. L.-C., Bishop, J., and Eloff, J. H. P. (2008). Smssec: An end-to-end protocol for secure sms. *Computers & Security*, 27(56):154.

Nielsen, T. T. and Jacobsen, R. H. (2005). Opportunities for ip in communications beyond 3g. *Wireless Personal Communications*, 33(3):243–259.

Saxena, N. and Chaudhari, N. S. (2014). Securesms: A secure sms protocol for vas and other applications. *Journal of Systems and Software*, 90:138–150.

Shinde, P. A., Mane, Y. B., and Tarange, P. H. (2015). Real time vehicle monitoring and tracking system based on embedded Linux board and android application. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pages 1–7.

SimCom (2013). SIM800L Hardware Design V1.00. Technical report, SIMCom.

Toorani, M. and Beheshti, A. (2008a). Solutions to the GSM Security Weaknesses. In *2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 576–581.

Toorani, M. and Beheshti, A. (2008b). SSMS - A secure SMS messaging protocol for the m-payment systems. In *2008 IEEE Symposium on Computers and Communications*, pages 700–705.

ublox (2011). NEO-6 u-blox 6 GPS Modules Data Sheet. Technical Report GPS.G6-HW-09005-E.

Verma, P. and Bhatia, J. (2013). Design and development of gps-gsm based tracking system with google map based monitoring. *International Journal of Computer Science, Engineering and Applications*, 3(3):33.