Security analysis of mobile crowd sensing applications

Nsikak P. Owoh and M. Mahinderjit Singh
Universiti Sains Malaysia, Penang, Malaysia

Received 23 July 2018
Revised 30 August 2018
Accepted 24 October 2018

Abstract

The proliferation of mobile phones with integrated sensors makes large scale sensing possible at low cost. During mobile sensing, data mostly contain sensitive information of users such as their real-time location. When such information are not effectively secured, users' privacy can be violated due to eavesdropping and information disclosure. In this paper, we demonstrated the possibility of unauthorized access to location information of a user during sensing due to the ineffective security mechanisms in most sensing applications. We analyzed 40 apps downloaded from Google Play Store and results showed a 100% success rate in traffic interception and disclosure of sensitive information of users. As a countermeasure, a security scheme which ensures encryption and authentication of sensed data using Advanced Encryption Standard 256-Galois Counter Mode was proposed. End-to-end security of location and motion data from smartphone sensors are ensured using the proposed security scheme. Security analysis of the proposed scheme showed it to be effective in protecting Android based sensor data against eavesdropping, information disclosure and data modification.

Keywords Dynamic analysis, Advanced encryption standard, Authentication, Encryption, Galois counter mode, Secure socket layer

Paper type Original Article

1. Introduction

The power of mobile devices is utilized in the new sensing paradigm called Mobile Crowd Sensing (MCS) [1]. This new and ever-growing trend exploits sensing and mobility features of mobile phones, and wearable devices to obtain knowledge such as personal and surrounding context, location, traffic conditions, noise levels, etc. It is estimated that by 2018, there will be about 3.3 billion connected mobile devices [2], and new mobile applications such as smart city [3], medical cyber physical systems [4] and real-time mobile cloud applications are expected to attain their full potentials [5].

Declarations of interest: None.

Author contributions: Nsikak Owoh and *Manmeet Mahinderjit Singh* performed the experiments; *Nsikak Owoh* wrote the paper; *Manmeet Mahinderjit Singh* reviewed the paper and made necessary corrections.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Publishers note: The publisher wishes to inform readers that the article "Security analysis of mobile crowd sensing applications" was originally published by the previous publisher of *Applied Computing and Informatics* and the pagination of this article has been subsequently changed. There has been no change to the content of the article. This change was necessary for the journal to transition from the previous publisher to the new one. The publisher sincerely apologises for any inconvenience caused. To access and cite this article, please use Owoh, N. P., Mahinderjit Singh, M. (2022), "Security analysis of mobile crowd sensing applications", *Applied Computing and Informatics*. Vol. 18 No. 1/2, pp. 2-21. The original publication date for this paper was 25/10/2018.

Applied Computing and Informatics
Vol. 18 No. 1/2, 2022
pp. 2-21
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2210-8327
DOI 10.1016/j.aci.2018.10.002

In smartphones for instance, inherent sensors such as gyroscope, accelerometer, GPS, magnetometer are used for acquisition of both personal and environmental data. They also have high computation and communication capabilities which enable processing and transmission of sensed data [6]. These features make mobile sensing devices different from the traditional IoT objects (e.g., mote-class sensors).

MCS can be formally defined as a platform that allows citizens with sensing devices (smartphones, tablets, wearable devices) collect and contribute sensed data which are later aggregated and fused in the cloud to extract information useful for people-centric delivery [1]. Environment, traffic, social behaviour and healthcare monitoring are possible by fusing and analyzing multi-dimensional information from mobile sensing devices. In MCS applications, sensor data can be collected with active user involvement (as in participatory sensing) [7] or automatically with minimal user involvement (opportunistic sensing) [8]. Also, these applications can be grouped into two categories, *personal* and *community* sensing based on the type of event being observed at any given time. In personal sensing applications, an individual is the focus of the sensing event; examples are human activity recognition (e.g., walking, jogging, running) and transport mode prediction [9]. Meanwhile, community sensing focuses on large-scale events that cannot be captured easily by a single individual. Examples are air pollution and traffic congestion monitoring. In this category of sensing, events are accurately measured when sensed data are gathered from several individuals (participants). However, data collection is the main purpose for the development of either personal or community sensing applications.

Despite its benefits, MCS applications still face challenges such as quality and reliability of sensed data (data and user trustworthiness) [10], incentivizing participants [11,12], energy consumption of mobile sensing devices [9,13], sensor data annotation [14], security and privacy [15,16]. The quality and reliability of sensed data is a lingering issue in MCS applications, as participants could deliberately report low-quality or fake data. Furthermore, the quality of sensed data can be reduced when data from faulty sensors are collected and recorded during sensing activities. To improve data quality in MCS, data selection, quality estimation and fault filtering techniques are necessary. However, user's participation determines the quality of collected data, which makes incentivizing of users important in achieving a successful MCS system [17].

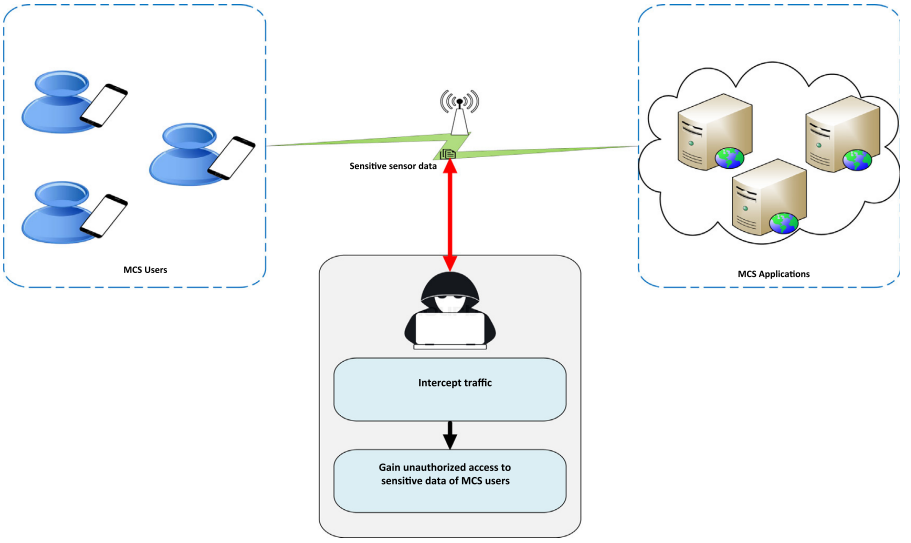

Figure 1.
Overview of attack on sensitive sensor data in MCS.

Security and privacy is another pressing issue and this raises concerns with personal data shared in MCS applications as sensitive information such as location of users are vulnerable to privacy attacks [1]. An adversary can intercept MCS traffic in order to capture sensitive information of users contained in sensor data as shown in Figure 1. For example, GPS sensor readings can be used by an adversary to infer personal information of individuals pertaining to their daily routes to work and their home locations [9].

Efforts have been made by researchers in detecting loopholes in both sensing applications and transmission protocols responsible for the vulnerability of sensitive information of users. However, an in-depth vulnerability analysis of raw data from smartphone based location and motion sensors is lacking. To this effect, we show how dynamic analysis can be used to test the security of raw sensor data in Android-based sensing applications. We aim to analyze the possibility of sensor data interception and location information disclosure of users during mobile sensing. To achieve this, Burp suite (a penetration testing tool) is used for the vulnerability assessment of 40 Android-based sensing applications. Thereafter, a security scheme that offers end-to-end security to sensitive data during mobile sensing is proposed. The contributions of this paper are as follows:

1. Investigate sensing applications for possible interception and disclosure of GPS data streams.
2. Analyze captured traffic between mobile sensing applications and their respective web servers
3. Propose an enhanced encryption and authentication scheme based on AES-256/GCM for securing location and motion data in MCS.

The outline of this paper is as follows. Section 2 presents related works while materials and methods are elaborated in Section 3. In Section 4, results and discussion from the dynamic analysis of 40 Android-based (Smart City, Health and Fitness) applications are presented. The proposed encryption and authentication scheme is presented in Section 5. Section 6 concludes the paper.

2. Literature review

This section presents few works proposed in tackling some of the identified issues in MCS. With emphasis on smart city, we discuss efforts made by researchers in improving user incentive mechanisms, enhancing data trustworthiness and user reputation. With respect to security and privacy in MCS, this section also reviews previous works that have been done in analyzing Android and iOS applications using either static, dynamic or hybridized (static and dynamic) vulnerability analysis techniques.

2.1 Data trustworthiness and user reputation

Data trustworthiness is a major concern in mobile crowd sensing, since acquired data is mostly used for decision making that affects the quality of life of citizens [18]. Also, user trustworthiness refers to the average reputation of a user over a certain period of time [19].

Data trustworthiness in user incentivization was studied in Kantarci et al. [20]. The authors used both statistical and recommendation-based user reputation to ensure data trustworthiness. Also, Kantarci et al. [21] proposed Social Network-Assisted Trustworthiness Assurance (SONATA), a recommendation-based method that identifies malicious users that spread false information in MCS systems through manipulation of sensor readings. With this approach, the probability of manipulating data in MCS is reduced using the vote-based trustworthiness analysis and Sybil detection techniques.

Pouryazdan et al. [5] classified data trustworthiness based on soft and hard reputation of MCS participants. Hard reputation in this case refers to accuracy from mobile sensing devices when used by participants in sensing activities. Soft reputation on the other hand refers to the malicious behaviour of the participants. They evaluate the performance of anchor-assisted, vote-based and collaborative reputation in mobile crowd sensing and conclude that hybridization of these approaches improve reputation scores of users in MCS.

2.2 User incentive mechanisms

Reliability of users in MCS can be sustained through incentive mechanisms as users tend to submit reliable data when good incentives are offered. Incentive mechanism is designed to inspire active human participation in MCS [19].

In an effort to increase user participation in MCS, several incentive mechanisms have been proposed; game theoretic methods [22], auction-based approaches [23], monetary [24] and non-monetary methods [25].

Yan et al. [26] propose a cloud-assisted architecture for MCS based urban transportation systems. In order to gain more participants, a component of the system called Mechanism of more Contributions and more Feedback Services (MCFS) is used as an incentive mechanism to collect more sensing data from drivers.

On the other hand, Obinikpo et al. [27] proposed queue theory to model target coverage in MCS. The model is based on birth-and-death mechanism which represents the arrival and exit of sensors in an MCS environment. This solves problems relating to network coverage, target clashes faced by sensors and as well as ensures efficient power usage during sensing.

2.3 Security analysis of MCS applications

From the user's perspective, security and privacy is a major concern in MCS, especially as MCS applications mostly gather sensitive sensor data of users which can be used to infer behavioural patterns of participants [19]. This makes it necessary to analyze potential disclosure of such sensitive information of users by sensing applications [28].

Fahl et al. [29] introduced *MalloDroid* (an extension of *Androguard*), a static code analyzer which they used to test 13,500 popular and free Android apps for security vulnerabilities against Man-in-the-Middle attacks. They focused their analysis on the communication protocols used by apps (i.e. HTTP or HTPS) and this was done by extracting URLs. Results from their analysis showed that 1074 out of the entire apps tested had vulnerable SSL/TLS codes which made apps susceptible to MITM attacks. Furthermore, SSL/TLS misuses were identified by the authors while carrying out manual analysis of 100 selected apps including 41 apps that transmitted sensitive information of users. They were able to capture login credentials of most apps and could also inject and execute code in an app that was developed using a vulnerable app-development framework.

Sounthiraraj et al. [30] presented SMV-HUNTER, a system that automatically analyzes large-scale Android apps for SMV. The system is composed of a static analysis component that recognizes possible vulnerable apps and also a dynamic analysis component that ratifies the vulnerable status of apps. The modular and non-specific system could be employed for other vulnerability analysis. The efficiency of the developed system was tested with 23,418 apps downloaded from Google Play Store where the static analysis spotted 1453 possible vulnerable apps and when dynamic analysis was performed on them, it was proven that 726 were actually vulnerable.

He et al. [31] investigated security and privacy risks in Android-based *mHealth* apps. The investigation was based on the following: (1) potential attack surfaces, (2) threat escalation, and (3) threat severity. In the first stage, 160 apps were downloaded from Google Play Store which included 80 free apps in Health & Fitness and 80 free Medical apps. From the attack

surfaces identified, authors of this work presented areas that required security, which are: Third Party Services, Internet, Logging, Bluetooth, SD Card Storage, Exported Components, and Side Channels. In the second stage, they selected 27 apps for analysis and presented three attack surfaces that needed security, such as: Internet, Third Party Services, and Logging. In the third and final stage, 120 apps that transmit sensitive information were selected and analyzed to determine to severity of Internet communications of sensitive information transmission. Results showed that majority of the apps tested transmit unencrypted data over the Internet and also use third party storage and hosting services.

He et al. [32] selected 20 free and paid *mHeatlh* apps from Android and iOS Market Store for security and privacy analysis. They used most downloaded and high rated apps as selection criteria. Their aim was to identify apps that require user registration (name, address and email); apps that allow users to update their personal profile; apps that require user authentication (username and password) and the different data storage locations (device storage or cloud storage) used by apps. They focused on authentication related features in the downloaded apps with respect to user's privacy. Results presented by the authors proved that a large number of apps tested lack provision for user data control where users can delete their personal information. They also showed that most of the apps share users' information with third party. Sadly, authors of this work affirmed to an earlier research done in McCarthy [33] which presented the fact that most free apps do not implement any form of security mechanisms (such as SSL) during transfer of user information from mobile apps to their respective websites. Meanwhile, Knorr and Aspinall [34] presented a threat analysis method for Android-based *mHealth* apps that monitor hypertension and diabetes. Authors of this work tested apps under this category as well as their associated web servers in order to evaluate their privacy policies. From their analysis, the following conclusions were made: (1) Sensitive data transmitted by *mHealth* apps lack adequate security such as encryption which is due to the fact that app developers do not prioritize security during app development. (2) Security mechanisms are non-trivial (3) the ever changing functionalities in new apps which require regular security testing.

Previous works such as Fahl et al. [29] and Sounthiraraj et al. [30] analyzed a large number of Android apps using most downloaded and highly rated apps as selection criteria. They also developed tools that employ both static and dynamic analysis techniques in detecting vulnerable apps. On the other hand, He et al. [31] and Knorr and Aspinall [34] focused on analyzing apps that acquire health related data of users (mHealth apps). However, the security vulnerabilities of raw sensor data such as GPS data during mobile sensing have not been fully explored. More so, no effective security solution or countermeasure was proposed in these works. Hence the need for an effective and efficient security scheme that will protect sensitive sensor data during mobile crowd sensing.

3. Methodology

This section presents the applications selected for analysis together with the dynamic analysis tool used. The Android-based sensing applications are categorized into three distinct groups and the sensors employed for sensing are also highlighted.

3.1 Sensing applications

Sensing applications that will be tested are Android-based and they employ location (GPS) and motion (accelerometer and gyroscope) sensors for data acquisition. For the analysis, 40 apps are downloaded from Google Play Store and are grouped into the following categories: smart city, smart health and fitness apps. Free apps with high ratings are used as selection criteria. Table 1 presents the apps to be tested including sensors and communication medium used by each app.

Sensing Applications	Description	Sensors used	Communication Medium
Smart City Apps			
RTA smart Parking	Used to search and pay for parking spots	GPS/Accelerometer	Wireless/Cellular Network
Free Parking	Used to search for the free parking spot	GPS/Accelerometer	Wireless/Cellular Network
Sound Meter	Measures environmental noise	GPS/Speaker	Wireless/Cellular Network
Clean Air Make More	Measures environmental noise	GPS/Speaker	Wireless/Cellular Network
Air Quality Real-Time AQI	Provides real-time data on air quality	GPS/Accelerometer	Wireless/Cellular Network
Traffic Authority	Traffic and navigation application	GPS/Camera	Wireless/Cellular Network
Road BUMP	Offers information on traffic conditions	GPS	Wireless/Cellular Network
Pothole Finder	Identifies location of potholes on streets	GPS/Camera	Wireless/Cellular Network
Here WeGo- City Navigation	Detects and reports pothole to users	GPS/Accelerometer	Wireless/Cellular Network
GPS Map: Navigation and Maps	Employed for navigation	GPS/Accelerometer	Wireless/Cellular Network
Traffic Lanes 2	Detects routes for better navigation	GPS/Camera	Wireless/Cellular Network
Clean City Networks	Used to controls traffic lights	GPS/Accelerometer	Wireless/Cellular Network
Urban Services Management	Monitors & manage waste bins in a city	GPS/Camera	Wireless/Cellular Network
Smart city Budapest	Monitors waste bins in a city	GPS/Camera	Wireless/Cellular Network
SmartMaps: GPS Navigation & Maps	Provides cab services to users	GPS/Camera	Wireless/Cellular Network
Waze-GPS, Maps, Traffic Alerts & Sat Nav	Informs users of road conditions	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Paker, Find available parking	Finds available parking spots	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Echelon	Offers smart city lighting	GPS/Accelerometer	Wireless/Cellular Network
Health Apps			
MyFitnessPal	Records weight and calories of users	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Jawbone Up	Offers health related tips	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Babylon: online doctor & symptom checker	Offers access to reliable medical advice	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Health Assistant	Monitors a range of health parameters	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
iTriage	Healthcare decision support application	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Health Infinity	Tracks weight, BMI calories & heart rate	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Argus	Tracks workout, weight loss & heart rate	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Lose it!	Weight loss tracker	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Noom Coach	Monitors calories & weight of users	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Lifesum	Tracks and import fitness data	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth

(*continued*)

Table 1. Sensing apps for analysis.

Table 1.

Sensing Applications	Description	Sensors used	Communication Medium
Fitness Apps			
Strava	Tracks runs and rides	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Moves	Activity tracking application	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Runkeeper	Tracks & records walks and runs of users	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Activity Tracker	Recognizes activities (running, etc.)	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
JEFIT	Tracks and logs workouts	GPS/Accelerometer	Wireless/Cellular Network
Runtastic	Running, fitness and exercise tracker	GPS/Accelerometer/Gyroscope	Bluetooth/Wireless/Cellular Network
MapMyFitness	Monitors workouts and exercises of users	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
SportsTracker	Tracks daily workouts of the user	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
FitStar	Pervasive fitness trainer application	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
StrongLifts	Monitors workout sessions of the user	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network
Fitbit	Tracks fitness activities of the user	GPS/Accelerometer/Gyroscope	Wireless/Cellular Network/Bluetooth
Pacer	Tracks movement & exercise of the user	GPS/Accelerometer/Pedometer	Wireless/Cellular Network

3.2 Dynamic analysis tool

Dynamic analysis enables testing of running apps irrespective of the programming language used for the development of such apps. With this kind of testing, false positives are minimal due to the involvement of human expert in the analysis process [35]. There are several dynamic analysis tools that can be used for effective vulnerability assessment of apps. In this work, we used *Burp Suite* for black box testing since studies such as [36,37] have shown that it is one of the most effective and efficient tools for vulnerability assessment of web and mobile applications. *Burp Suite* has many testing features that provide effective vulnerability analysis. The Intruder tab is used for the automation of customized attacks against web applications in order to identify and exploit all known vulnerabilities. The Spider tab, offers crawling functions during penetration testing. The Repeater tool is employed to modify HTTP requests and to analyze their responses. The tool works as an intercepting proxy and can be configured to intercept, log, display and modify HTTP traffic. The main function of Burp suite is that it offers an overview of transmitted messages and parameters which allows the penetration tester (security researcher) to have full control of messages in order to simulate different attack scenarios.

3.3 Testing method

For the app testing, we used a Samsung Galaxy S4 smartphone running Android 5.0.1 Lollipop. We used a laptop running Kali Linux 4.13.10 to connect to the smartphone and to run *Burp Suite*. To effectively intercept SSL traffic between the sensing apps and their respective servers, a root certificate was installed on the smartphone. After setting up the test environment, we launched each app and created dummy accounts (username and passwords) where necessary. Thereafter, we used these apps like a regular user and at the same time tried to intercept traffic between the mobile sensing device (smartphone) and their respective web servers. Consequently, we sniffed sensitive information pertaining to raw GPS data each time location information were received. From the traffic captured, we recorded apps that use either HTTP or HTTPS connections. The dynamic analysis method used during the experiment allowed us to analyze and record SSL details of all running applications. We tried to implement a passive man-in-the-middle attack on apps that only employ HTTP and an active man-in-the-middle attack on apps that implemented SSL (HTTPS) incorrectly. The result of the experiment is presented in the next section.

4. Results and discussion

The results obtained from the vulnerabilities analysis with respect to the three different categories of apps tested are presented in this section. Results from all 18 smart city apps tested showed that interception of traffic between the mobile sensing device (smartphone) and web servers of respective sensing applications were possible during dynamic analysis. Furthermore, we observed that it was possible to obtain GPS data pertaining to location information of the user when the SSL connection was circumvented. Similarly, all 10 apps tested in the healthcare category were also vulnerable to traffic interception. Sensitive location information of the user were disclosed from the GPS data obtained during mobile sensing in all apps in this category. Lastly, all 12 apps tested in the fitness category were also vulnerable to traffic interception as it was possible to sniff plaintext GPS data gathered from the user. Table 2 summarizes the obtained results from the three categories of apps tested.

During the analysis, it was observed that most apps do not employ secure communication channel for the transfer of sensor data from the mobile sensing device (client-side) to the server, which makes it easy for an adversary to capture sensitive location information of users. Furthermore, when performing the analysis, it was possible to capture sensitive data

Sensing Applications	Interception of Traffic	Disclosure of Sensitive Location Information
Smart City Apps		
RTA smart Parking	✓	✓
Free Parking	✓	✓
Sound Meter	✓	✓
Clean Air Make More	✓	✓
Air Quality Real-Time AQI	✓	✓
Traffic Authority	✓	✓
Road BUMP	✓	✓
Pothole Finder	✓	✓
Here WeGo- City Navigation	✓	✓
GPS Map: Navigation and Maps	✓	✓
Traffic Lanes 2	✓	✓
Clean City Networks	✓	✓
Urban Services Management	✓	✓
Smart city Budapest	✓	✓
SmartMaps: GPS Navigation & Maps	✓	✓
Waze-GPS, Maps, Traffic Alerts & Sat Nav	✓	✓
Paker, Find available parking	✓	✓
Echelon	✓	✓
Health Apps		
MyFitnessPal	✓	✓
Jawbone Up	✓	✓
Babylon: online doctor & symptom checker	✓	✓
Health Assistant	✓	✓
iTriage	✓	✓
Health Infinity	✓	✓
Argus	✓	✓
Lose it!	✓	✓
Noom Coach	✓	✓
Lifesum	✓	✓
Fitness Apps		
Strava	✓	✓
Moves	✓	✓
Runkeeper	✓	✓
Activity Tracker	✓	✓
JEFIT	✓	✓
Runtastic	✓	✓
MapMyFitness	✓	✓
SportsTracker	✓	✓
FitStar	✓	✓
StrongLifts	✓	✓
Fitbit	✓	✓
Pacer	✓	✓

Table 2.
Results from analysis of sensing apps.

(location information) of the user even when SSL was used due to its poor implementation in most apps. All apps tested were vulnerable to traffic interception and location information disclosure with a 100% rate as depicted in Figure 2. This proves that the privacy of MCS users is not guaranteed when using sensing apps that lack in-depth security on sensed data. To achieve maximum security, sensed data must be effectively and efficiently encrypted and authenticated during sensing irrespective of the communication channel used.

MCS may be used to offer real-time information to users on road and traffic conditions. However, maintaining user's privacy through effective security of their location

information remains an unsolved problem. Results from the vulnerability analysis of sensed data in MCS applications presented in this paper show that raw sensor readings can easily be intercepted and sensitive information disclosed to an adversary. This is as a result of lack of or ineffective security mechanisms of sensor data from Android-based sensing applications.

To ensure effective security during mobile sensing, a scheme that offers in-depth encryption and authentication of sensor data is required. Our proposed scheme which provides such services is discussed in some detail in the next section.

5. Proposed security scheme

Analysis presented in Section 3 is in line with earlier research work presented in Fahl, Harbach [29] which affirms that improper usage of SSL (such as trusting all certificates, allowing all hostnames and mixed-mode/No SSL) allows an active MITM attacker to have unauthorized access to plaintext information transmitted via a compromised encrypted channel. This entails that sensitive information of users transmitted in plaintext using an SSL connection that is incorrectly implemented or forced open by an attacker can compromise the confidentiality of sensed data. Considering the possibility of app developers to wrongly implement SSL in application codes which can be exploited by an attacker, we propose the encryption and authentication of sensed data (location and motion data) during mobile sensing before transmission to their respective servers. Table 3 presents notations and symbols used in the formulation of algorithms for the proposed security scheme. They are used both in the encrypt-then-authenticate and authenticate-only algorithms.

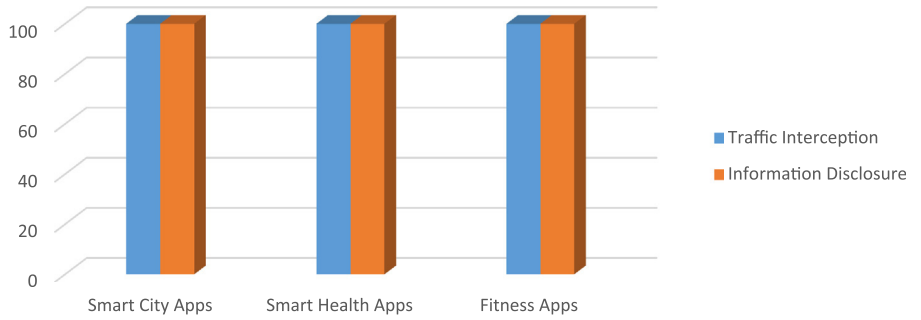

Figure 2.
Results from analysis of Android-based sensing apps.

Symbol	Definition
bc	Block counter
$+$	Concatenation
IV	Initialization Vector
n	Number of blocks
PT	Plaintext
E_k	AES encryption
$\oplus$	XOR
CT	Ciphertext
h	Hash function (GHASH)
len	Length
$\rightarrow$	Output

Table 3.
Notations and symbols.

5.1 Encryption and authentication of location-based sensor data

Authenticated Encryption with Associated Data (AEAD) schemes offer authentication, integrity and confidentiality services seamlessly by integrating the operations of a cipher and of a message authentication algorithm using a single key [38]. Commonly used AEAD schemes are, Galois Counter Mode (GCM), Encrypt-then-Authenticate-then Translate (EAX), and Cipher Counter Mode (CCM). On one hand, our proposed scheme employs GCM mode of operation with an underlying AES-256 block cipher to implement encrypt-then-authenticate mechanism on location data from GPS sensor. On the other hand, the authenticate-only mechanism is implemented on motion data from sensors such as accelerometer, gyroscope, etc. The algorithm is shown in Table 4 while the process diagram is illustrated in Figure 3.

Location data from GPS sensor which contain sensitive information of MCS users is first encrypted then authenticated to ensure confidentiality and integrity. As shown in Figure 3, using GCM algorithm, the plaintext data denoted as PT is first divided into blocks (Counters of 1, . . ., n), and then XORed. As shown in Figure 3, the first block value is 1 which is encrypted using AES 256 with key K (E_K). The output of the encrypted counter is XORed

Algorithm 1

Input: Location data from Smartphone-based GPS sensor

1. Initialize $bc + IV$
2. Encrypt $bc + IV$ using E_k
3. $\boldsymbol{PT} \oplus E_k \rightarrow \boldsymbol{CT}$
4. Increment bc; Incrementing block counter with new Initialization Vector
5. Repeat 2–3 for all n blocks
6. Initialize h; Initial state of the hash function
7. Encrypt h using E_k; by forwarding 128 bits of zeros to AES
8. $\boldsymbol{CT} \oplus h \rightarrow \boldsymbol{CT_h}$
9. Repeat for alln_{+1}
10. $len(PT) + h$
11. Return $\boldsymbol{CT_h}$

Output: Encrypted and authenticated GPS data

Table 4.
Algorithm to encrypt and authenticate location data.

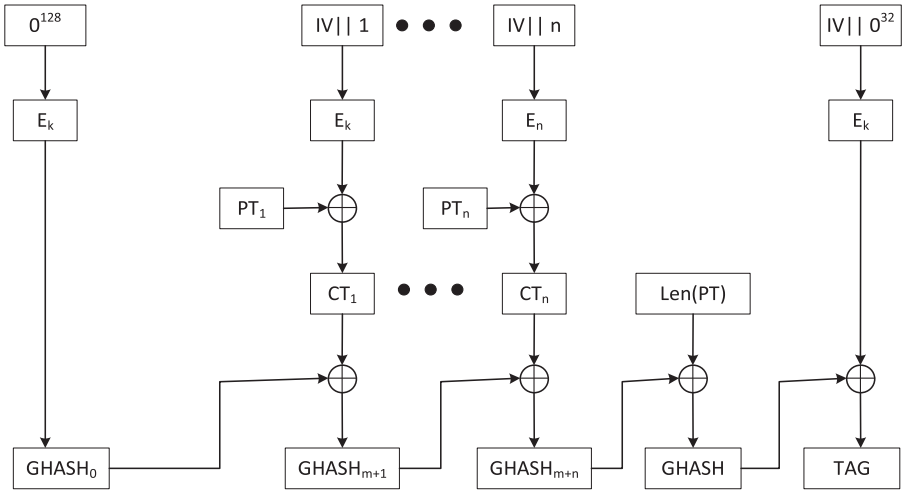

Figure 3.
Process flow of encrypt-then authenticate algorithm using Galois Counter Mode (GCM).

with the plaintext (location data, *PT*) which generates a ciphertext *CT*. This process is continued for all *n* blocks of plaintext (all GPS data). To ensure randomness, the Counter is concatenated with an *Initialization Vector* (*IV*) which serves as the nonce (which can only be used once). The *IV* is 96 bits and the counter uses 32 bits which sums up to the 128 bits block for the AES encryption enabling up to 2^{32} before the counter moves over to the next block. To achieve authenticity and integrity, the ciphertext (*CT*) is XORed and the output is hashed using GHASH (a hashing function in GCM) and the output is passed from one stage unto the next stage of the algorithm until the n_{th} block. The hash function is initialized by sending 128 bits of zero (0^{128}) and encrypted through the AES 256 algorithm then through the hashing function. Additionally, the length of the block (*len (PT),* i.e. *length of the location-based data)* is added to the hash together with the initialization vector (*IV*). The output of this process is an encrypted and authenticated GPS data.

Remark. Without loss of generality, our assumption is that there is a Key Distribution Centre (KDC) in the form of a key server which handles key establishment, so that both the client/server (mobile app and web server) can share an encryption key E_k.

5.2 Authentication of motion-based sensor data

Galois Message Authentication Code (GMAC) is a component of GCM (Galois Counter Mode), used for authentication of messages without encryption. GMAC is an incremental algorithm which after computing the MAC of a message M, the cost of computation of the message $M^{'}$ is proportional to the hamming weight between those messages [39]. Algorithm for the authenticate-only of motion data is presented in Table 5 with its process flow shown in Figure 4.

The hash function GHASH is defined by $\text{GHASH}(H, A, C) = X_{m+n+1}$ where the inputs of A and C are the incremental authentication function represented formally as $incr(F||I)$ obtained from $F||(I + 1 \bmod 2^{32})$. GMAC supports incremental tag generation for different messages, and modifications within a fixed-length message by attaching data to a message and data truncation from the beginning to the end of the message. In the proposed scheme, GMAC is employed for authentication of additional motion data. With this mechanism is place, data integrity is assured thereby identifying any form of data fabrication or modification from an adversary [40]. 128 bits which is the largest tag size for GMAC with AES block cipher is employed in the proposed scheme. GMAC offers an efficient method to authenticate large datasets, implementing the computation of new authentication tags after a slight modification is made.

5.3 Conceptual framework of the proposed scheme

The proposed scheme will be implemented as a generic Android application which will encrypt and authenticate data from location (GPS) and motion (accelerometer and gyroscope)

Algorithm 2

Input: Additional data from Smartphone-based motion sensors
1. Initialize h; Initial state of the hash function
2. Encrypt h using E_k; by forwarding 128 bits of zeros to AES
3. $\boldsymbol{PT} \oplus h \rightarrow \boldsymbol{PT_h}$
4. Repeat for all n_{+1}
5. $len(PT) + h$
6. Return $\boldsymbol{PT_h}$

Output: Motion data authenticated

Table 5. Algorithm to authenticate motion data.

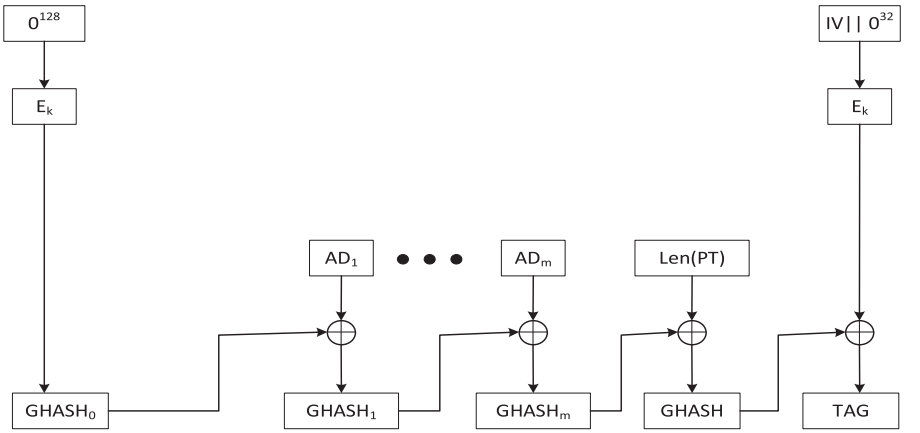

Figure 4.
Process flow of the authenticate-only algorithm using Galois Message Authentication Code (GMAC).

sensors. These data can be used by mobile sensing applications that require secure location and motion data from Android based smartphone. Components and their respective interactions in the proposed scheme are shown in Figure 5.

Android is an open source software stack developed for smartphones and tablets. It consists of a Linux kernel, an Android middleware and the application layer. The Linux kernel offers basic functions such as memory, process scheduling, device drivers and the file system. Above the Linux kernel is the middleware layer, which has native libraries, the Android runtime environment and the application framework. The native libraries offer vital functionalities such as graphic processing. The Android runtime environment consists of core Java libraries and the Dalvik Virtual Machine, which is meant for certain requirements of resource constrained mobile devices [41]. The main security in Android are application sandboxing and a permission framework [41]. Vulnerabilities in Android's security architecture renders sensing applications more susceptible to security attacks. To this end, the proposed security scheme serves as a middleware for encrypting and authenticating sensed data in Android-based mobile devices.

5.4 Implementation of proposed scheme

Java's cryptography package (java.crypto) was used from the SUN JCE standard library for the key generation. JCE offers the needed cryptographic primitives for management of security in Android based applications. The basic API packages included in JCE are the standard java.crypto, java.security and java.math and these packages call the arithmetic primitives present in the OpenSSL native library, which include multiplication and modular squaring. Also, a lightweight version of Bouncy Castle library that offers high level execution functions is included in these APIs. The version of AES with 256-bit key was implemented, including the methods Ecrypt( ), Decrypt( ), Auth( ) and KeySchedule( ) using the GCM mode of operation. Each cipher's key generation algorithm is called once during the start process and the set of round keys are stored and used for every recursive call to the encryption/decryption and the authenticate methods. Android Studio was employed for the development and deployment of the cryptographic algorithm as well as for the collection of data for the evaluation of the proposed security scheme. In an effort to optimize AES during implementation, we removed the use of local buffers to maintain the state while using the global variable to store the key schedule. Furthermore, key for each round was generated during the encryption process rather that precomputing and storing them in the RAM.

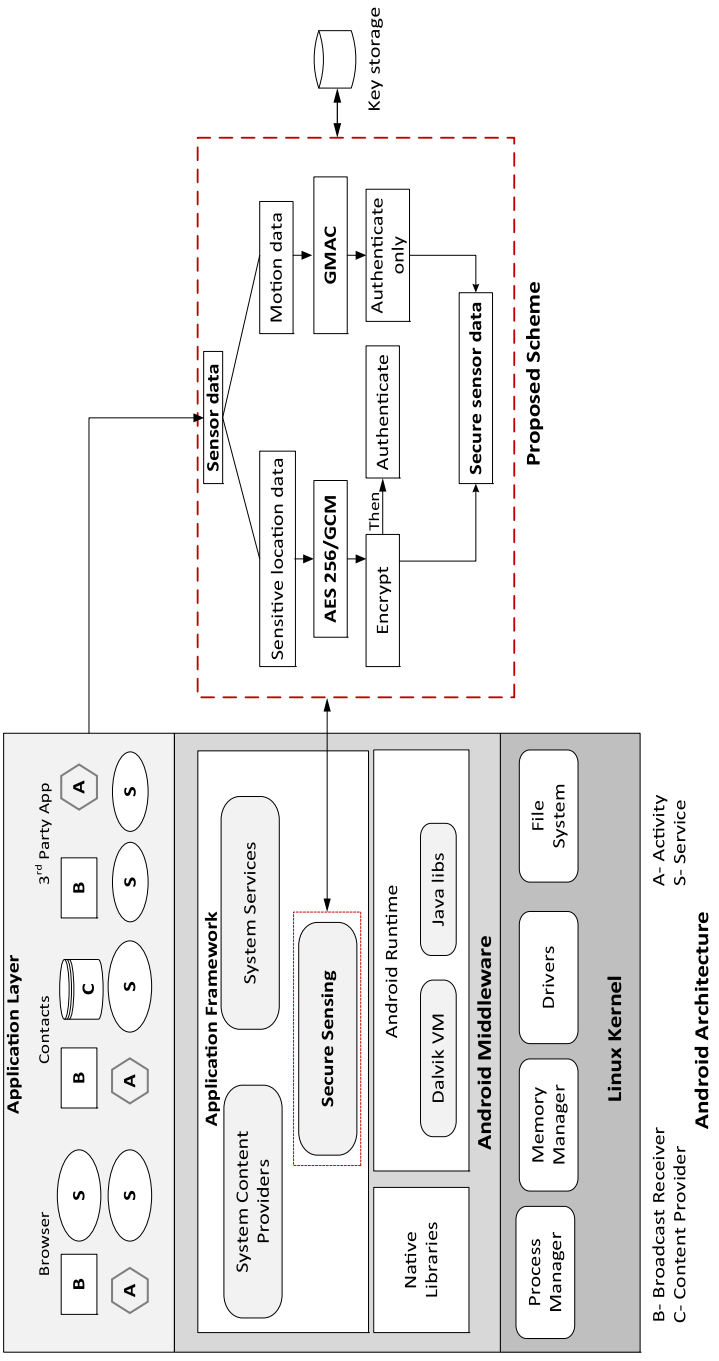

Figure 5. Android implementation of the proposed security scheme.

Transferring data in memory was minimized and the MixColumns transformation was written using the 16-bits memory. The experiment was implemented on a Samsung smartphone and its features are shown in Table 6.

5.5 Performance evaluation

To evaluate the performance of the implemented scheme, metrics such as speed (execution time) and memory were used. The phone was fully charged to 100% and other applications were also installed and used on the smartphones. This allows us to test the scheme in real world conditions.

5.5.1 Execution time. This refers to the number of plaintext data that can be encrypted/decrypted as well as authenticated in a second. A segment of our code implementation outputs the time it takes to encrypt and decrypt a single block of data. This is used to calculate the throughput of the implemented scheme, which is obtained by dividing the total plaintext encrypted (in bytes) by the encryption time. Figure 6 shows the time it takes (in milliseconds) to encrypt a single block of location data from GPS sensor. On the other hand, Figure 7 depicts the time it takes (also in milliseconds) to *authenticate-only* a single block of motion data from accelerometer and gyroscope sensors using the GMAC property of GCM. Less time is used to authenticate a single block of motion based data compared with the time used to encrypt the same size of data from location sensor (GPS).

Table 6.
Features of smartphone used for the implementation of the AES-GCM scheme.

Property	Specification
Type	Samsung Galaxy S4
Operating system	Android OS, v4.2.2 (Jelly Bean)
Memory	2 GB RAM, 16/32 GB
CPU	Quad-core 2.3 GHz Krait 400
GPU	Adreno 330
Chipset	Qualcomm MSM8974 Snapdragon 800
Battery	Li-Ion 2600 mAh battery

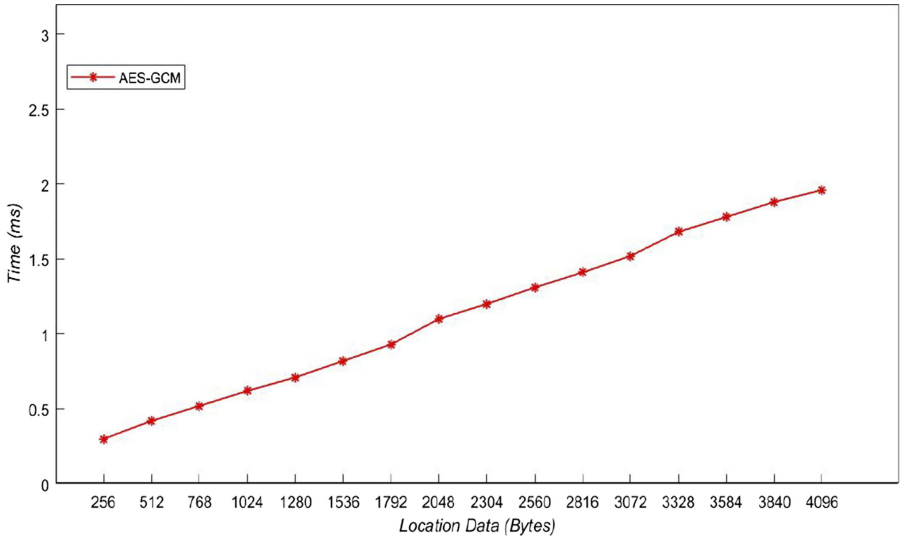

Figure 6.
Execution time of AES-GCM.

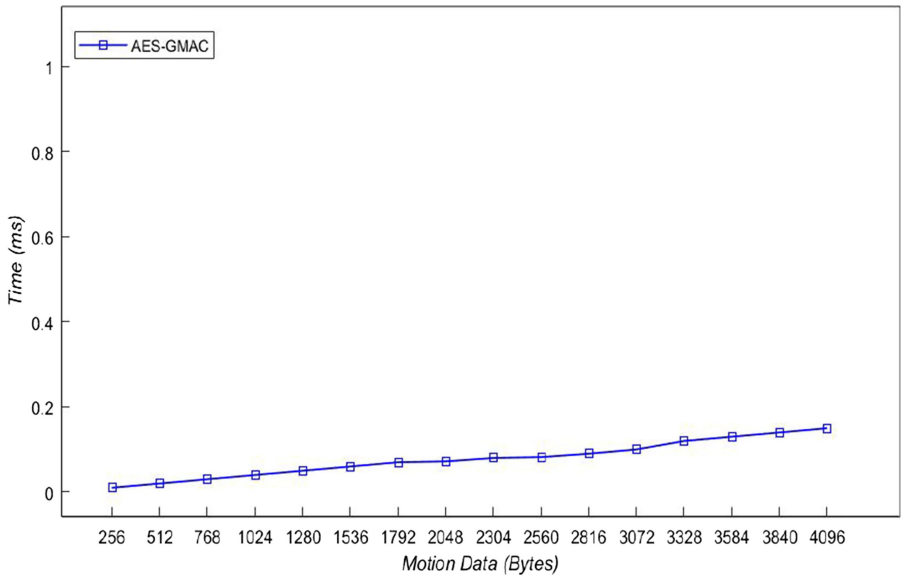

Figure 7.
Execution time of AES-GMAC.

5.5.2 Memory usage. The RAM or data memory is the high-speed, volatile onboard memory in smartphones. Presently, most smartphones come with memory of 2 GB to 3 GB. To get the RAM memory footprint, we used the ActivityManager.getMemoryInfo( ) method in Android and generated results were stored in the internal storage of the smartphone. As shown in Figure 8, encrypting data uses up more RAM memory of the smartphone followed by the decrypting of data. Authenticating data from motion sensors uses the least memory.

We compare the speed (execution time) and memory usage of the our scheme with AES-CBC based scheme as implemented in Li et al. [42]. Figure 9 shows our scheme performs fairly well in terms of speed (encryption time) when compared to Li, Yan [42]. AES-GCM mode of operation operates slightly slower than AES-CBC. However, improved execution time and memory usage was observed due to the optimization performed during implementation.

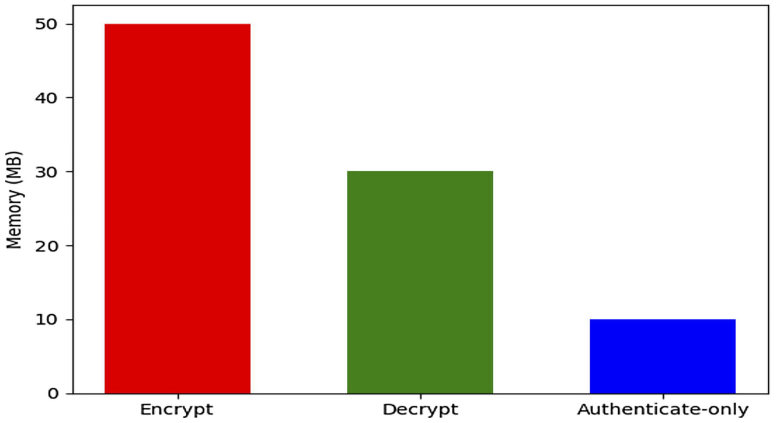

Figure 8.
Memory usage of implemented AES-GCM scheme.

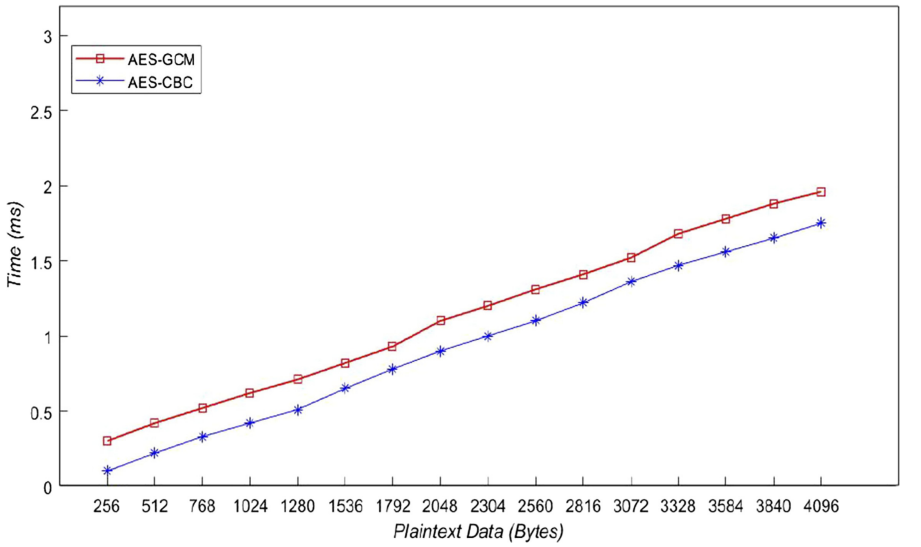

Figure 9.
Benchmark of implemented AES-GCM with AES-CBC.

5.6 Security analysis

Our proposed security scheme aims to mitigate eavesdropping, information disclosure and modification of sensitive location and motion based sensor data in MCS. As justified by [43], AES-GCM is efficient and effective in ensuring data security when correctly implemented.

5.6.1 Data security accomplishment. Based on the nature of sensor network and wireless communication, sensor data could easily be intercepted and modified as proven in previous sections of this paper; this poses great danger in life-critical cases. The encryption and authentication scheme implemented for Android based smartphone ensures data confidentiality, integrity and authentication. *Burp suite* was again employed to analyze traffic containing sensor data encrypted and authenticated using our implemented security scheme. We observed the following from the security testing performed:

5.6.2 Eavesdropping/information disclosure. The scheme implemented as an Android app encrypts location data from GPS as well as authenticates motion data from accelerometer and gyroscope sensors. Results presented in previous sections show that improper/no implementation of SSL in sensing applications could lead to successful interception and leakage of sensitive information of MCS users. The implemented scheme (security app), ensures that all location and motion data from smartphone sensors are effectively encrypted and authenticated. This thwarts the efforts of eavesdroppers in gaining access to sensitive information of MCS users.

5.6.3 Data modification. Any attempt by an adversary to modify sensor data can be detected using the unforgeable tag generated by the GMAC algorithm. This guarantees integrity and authenticity of sensor data from MCS applications.

6. Conclusion

In this paper, we presented an analysis of 40 Android-based sensing applications. The applications were categorized into three distinct groups namely, smart city, smart health and fitness apps. We used Burp Suite, a tool that employs dynamic analysis to identify apps that are vulnerable to SSL exploitation (such as MITM attack), eavesdropping and sensitive information disclosure. The analysis revealed possibility of traffic interception between client-side (smartphone) and the server-

side (web server) in all apps tested. In this paper, we also showed that sensitive GPS data pertaining to real-time location of the user were disclosed in all apps tested.

Results from the analysis show that an adversary with the right tools and technical skills can exploit an SSL connection especially when it is wrongly implemented. When this happens, sensitive data (such as geolocation coordinates and login credentials) transmitted via the encrypted channel (SSL) are revealed in plaintext to the attacker, which compromises confidentiality and threatens user's privacy. To effectively protect sensed data, we proposed and implemented a security scheme that offers in-depth security through the encryption and authentication of data from location and motion sensors. The proposed scheme employs AES 256-GCM algorithm to ensure confidentiality, integrity and authenticity of sensor data in MCS. Results from the performance analysis of the proposed scheme show high execution time (encryption/decryption time) while the memory usage is considerable low.

References

[1] B. Guo, Z. Yu, X. Zhou, D. Zhang, From participatory sensing to mobile crowd sensing, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on. 2014, IEEE, pp. 593–598.

[2] Gartner, Connected Things in Smart Cities. [Online]. Available: http://www.gartner.com/newsroom/id/3175418, 2016.

[3] M. Pouryazdan, B. Kantarci, T. Soyata, H. Song, Anchor-assisted and vote-based trustworthiness assurance in smart city crowdsensing, , IEEE Access 4 (2016) 529–541.

[4] A. Page, T. Soyata, J.-P. Couderc, M. Aktas, B. Kantarci, S. Andreescu, Visualization of health monitoring data acquired from distributed sensors for multiple patients, in: Global Communications Conference (GLOBECOM), 2015, IEEE, pp. 1–7.

[5] M. Pouryazdan, B. Kantarci, T. Soyata, L. Foschini, H. Song, Quantifying user reputation scores, data trustworthiness, and user incentives in mobile crowd-sensing, IEEE Access 5 (2017) 1382–1397.

[6] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, X. Mao, Incentives for mobile crowd sensing: a survey, IEEE Commun. Surv. Tutorials 18 (1) (2016) 54–67.

[7] J.A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M.B. Srivastava, Participatory sensing, 2006, pp. 1–6.

[8] N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A.T. Campbell, A survey of mobile phone sensing, IEEE Commun. Mag. 48 (9) (2010) 140–150.

[9] R.K. Ganti, F. Ye, H. Lei, Mobile crowdsensing: current state and future challenges, 49(11), IEEE Commun. Mag. (2011) 32–38.

[10] M. Talasila, R. Curtmola, C. Borcea, Mobile crowd sensing, Google Scholar (2015) 1–10.

[11] Y. Wen, J. Shi, Q. Zhang, X. Tian, Z. Huang, H. Yu, Y. Cheng, X. Shen, Quality-driven auction-based incentive mechanism for mobile crowd sensing, IEEE Trans. Veh. Technol. 64 (9) (2015) 4203–4214.

[12] H. Jin, L. Su, H. Xiao, K. Nahrstedt, Inception: incentivizing privacy-preserving data aggregation for mobile crowd sensing systems, in: Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2016, pp. 341–350.

[13] H. Ma, D. Zhao, P. Yuan, Opportunities in mobile crowd sensing, IEEE Commun. Mag. 52 (8) (2014) 29–35.

[14] N. Pius Owoh, M. Mahinderjit Singh, Z.F. Zaaba, Automatic annotation of unlabeled data from smartphone-based motion and location sensors, Sensors 18 (7) (2018) 2134.

[15] D. He, S. Chan, M. Guizani, User privacy and data trustworthiness in mobile crowd sensing, IEEE Wirel. Commun. 22 (1) (2015) 28–34.

[16] D. Zhang, L. Wang, H. Xiong, B. Guo, 4W1H in mobile crowd sensing, , IEEE Commun. Mag. 52 (8) (2014) 42–48.

[17] L. Liu, W. Wei, D. Zhao, H. Ma, Urban resolution: new metric for measuring the quality of urban sensing, , IEEE Trans. Mob. Comput. 14 (12) (2015) 2560–2575.

[18] P. Bellavista, A. Corradi, L. Foschini, R. Ianniello, Scalable and cost-effective assignment of mobile crowdsensing tasks based on profiling trends and prediction: the participant living lab experience, Sensors 15 (8) (2015) 18613–18640.

[19] C. Chowdhury, S. Roy, Mobile crowd-sensing for smart cities, Smart Cities (2017) 125–154, http://dx.doi.org/10.1002/9781119226444.ch5.

[20] B. Kantarci, P.M. Glasser, L. Foschini, Crowdsensing with social network-aided collaborative trust scores, , in: Global Communications Conference (GLOBECOM), IEEE, 2015, pp. 1–6.

[21] B. Kantarci, K.G. Carr, C.D. Pearsall, SONATA: social network assisted trustworthiness assurance in smart city crowdsensing, Int. J. Distrib. Syst. Technol. (IJDST) 7 (7) (2016) 59–78.

[22] D. Yang, G. Xue, X. Fang, J. Tang, Incentive mechanisms for crowdsensing: crowdsourcing with smartphones, , IEEE/ACM Trans, Networking (TON) 24 (3) (2016) 1732–1744.

[23] H. Jin, L. Su, D. Chen, K. Nahrstedt, J. Xu, Quality of information aware incentive mechanisms for mobile crowd sensing systems, in: Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2015, pp. 167–176.

[24] J.-S. Lee, B. Hoh, Dynamic pricing incentive for participatory sensing, Pervasive Mob. Comput. 6 (6) (2010) 693–708.

[25] K. Han, E.A. Graham, D. Vassallo, D. Estrin, Enhancing motivation in a mobile participatory sensing project through gaming, in: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on. 2011, IEEE, pp. 1443–1448.

[26] H. Yan, Q. Hua, D. Zhang, J. Wan, S. Rho, H. Song, Cloud-assisted mobile crowd sensing for traffic congestion control, Mobile Networks Appl. 22 (6) (2017) 1212–1218.

[27] A.A. Obinikpo, Y. Zhang, H. Song, T.H. Luan, B. Kantarcih, Queuing algorithm for effective target coverage in mobile crowd sensing, IEEE Internet Things J. 4 (4) (2017) 1046–1055.

[28] G.A. Fink, T.W. Edgar, T.R. Rice, D.G. MacDonald, C.E. Crawford, Security and Privacy in Cyber-Physical Systems, in Cyber-Physical Systems, Elsevier, 2017, pp. 129–141.

[29] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, M. Smith, Why Eve and Mallory love Android: an analysis of Android SSL (in) security, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ACM, 2012, pp. 50–61.

[30] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, L. Khan, SMV-hunter: Large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps, in: Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14. 2014, Citeseer, pp. 1–12.

[31] D. He, M. Naveed, C.A. Gunter, K. Nahrstedt, Security concerns in Android mHealth apps, , in: AMIA Annual Symposium Proceedings, American Medical Informatics Association, 2014, pp. 645–655.

[32] S. He, D.-H. Shin, J. Zhang, J. Chen, Toward optimal allocation of location dependent tasks in crowdsensing, in: INFOCOM, 2014 Proceedings IEEE, 2014, IEEE, pp. 745–753.

[33] M. McCarthy, Experts warn on data security in health and fitness apps, BMJ: Br. Med. J. 347 (2013) 1–9.

[34] K. Knorr, D. Aspinall, Security testing for Android mHealth apps, in: Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on, 2015, IEEE, pp. 1–8.

[35] C. Joshi, U.K. Singh, Performance evaluation of web application security scanners for more effective defense, Int. J. Sci. Res. Publ. (IJSRP) 6 (6) (2016) 660–667.

[36] A. Doupé, M. Cova, G. Vigna, Why Johnny can't pentest: an analysis of black-box web vulnerability scanners, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2010, pp. 111–131.

[37] Gartner, Research metodologies: Magic quadrant. [Online]. Available: http://www.gartner.com/technology/research/methodologies/researchmq.jsp, 2013.

[38] M.A. Simplicio, B.T. de-Oliveira, C.B. Margi, P.S.L.M. Barreto, Carvalho, M. Näslund, Survey and comparison of message authentication solutions on wireless sensor networks, Ad Hoc Netw. (11, 2013.) http://dx.doi.org/10.1016/j.adhoc.2012.08.011.

[39] M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography: the case of hashing and signing, in: Annual International Cryptology Conference, Springer, 1994, pp. 216–233.

[40] N. Ferguson, Authentication weaknesses in GCM, Comments submitted to NIST Modes of Operation Process, 2005.

[41] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, B. Shastry, Practical and lightweight domain isolation on android, , in: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, ACM, 2011, pp. 51–62.

[42] J. Li, H. Yan, Z. Liu, X. Chen, X. Huang, D.S. Wong, Location-sharing systems with enhanced privacy in mobile online social networks, IEEE Syst. J. (2015) 1–10.

[43] P. Szalachowski, B. Ksiezopolski, Z. Kotulski, CMAC, CCM and GCM/GMAC: advanced modes of operation of symmetric block ciphers in wireless sensor networks, Inf. Process. Lett. (2010) 247–251.

Corresponding author
M. Mahinderjit Singh can be contacted at: manmeet@usm.my