Log your car: Reliable maintenance services record

Hafizah Mansor[1], Konstantinos Markantonakis[2], Raja Naeem Akram[3], Keith Mayes[4], and Iakovos Gurulian[5]

Information Security Group, Smart Card Centre,
Royal Holloway, University of London,
United Kingdom
{Hafizah.Mansor.2011[1], RajaNaeem.Akram.2008[3], Iakovos.Gurulian.2014[5]}@live.rhul.ac.uk,
{K.Markantonakis[2], Keith.Mayes[4]}@rhul.ac.uk

Abstract. A maintenance services logging system is a useful tool for car owners to keep track of the car's condition and also can increase the market value of the car. Logging systems range from manual, paper-based, to automated, cloud-based systems. The automated process provides ease of use and availability of the records. A secure protocol is required to ensure that the workshop and service record are authentic, and hence the records are reliable. In this paper, we propose a secure protocol for automated maintenance services logging systems, through the use of a mobile application called AutoLOG. The multiple electronic control units (ECUs) used to support the connected and intelligent vehicle's technology are used to support the digital automated logging system. The car is the trusted entity that generates the log. The records are stored in an authorised mobile device and uploaded onto a cloud server to ensure availability. The proposed protocol is implemented to measure the performance and is formally analysed using Scyther and CasperFDR, with no known attack found.

1 Introduction

Maintenance services conducted are manually recorded by workshops. The services are either not recorded at all, or recorded in a logbook kept by the car owners. For reminder purposes, to notify the car owner of the next service date, a workshop may attach a sticker on the windshield. Recent use of mobile applications that have been introduced for car maintenance include a notification reminder for the next service date. However, other than the date of the next service, the type of maintenance repair, replacement or any other services might also be useful for the car owner to keep track of. The process of recording is manual, whereby the user has to manually enter the information using the mobile application. The logbook not only will keep the owner up-to-date on the health status of the car and avoid higher maintenance cost due to breakdowns, but it will also add value to the car, especially when the car is resold. The recent implementation which notifies the date of next service is available through the car dealer's cloud server [6]. When a car is being serviced by a car dealer, the details are uploaded onto the car dealer's server. When the next service date is nearing, the car owner will be contacted by the car dealer as a reminder.

An Electronic Control Unit (ECU) is a microcontroller that controls the operations of a car. In modern cars, there can be around 70 ECUs that control the overall operations of the vehicle [20]. Each ECU is responsible for different operations, such as body control, engine control and telematics. The different ECUs are connected within a car through networks such as Local Interconnect Network (LIN) [40], Controller Area Network (CAN) bus [22], FlexRay [25] and Media Oriented Systems Transport (MOST) [27]. The OBD-II (On-Board Diagnostic) port is a port that interfaces the outside world to the in-vehicle networks [41]. The port can be interfaced with a Wi-Fi, Bluetooth or serial connection using the ELM327 interface [17].

1.1 Problem statement

The challenges in a maintenance services record system are to provide integrity, authenticity and reliability of the data. The process of recording the maintenance log is manual, and the car owner normally does not have access to the data, unless it is manually recorded in a logbook that he/she keeps. The car owner cannot validate the services being conducted but must trust the information provided by the workshop through the receipts or documents provided. Furthermore, it is inconvenient to keep these receipts and/or documents for all the records of maintenance services. Equally, a potential buyer does not have an assurance that the records in the maintenance log and the workshops who had performed the services are authentic.

1.2 Contribution

In this paper we propose a protocol for a secure automated process of recording the maintenance services for car maintenance. It ensures integrity of the data stored as well as the authenticity of the data and party conducting the service. The automated process ensures the maintenance data is available all the time to the car owner. The list of services/repairs conducted by the workshop is also validated. The protocol is not only useful to the car owners, but also benefits the garages, car sales organisations and vehicle manufacturers.

2 Maintenance services logging system

A maintenance logging system allows the car owner to keep the car's maintenance services record updated and hence can reduce the cost of maintenance by avoiding major car breakdowns. Other than that, it can also add value to the car when it is resold [10]. The potential buyer is assured that the car is in good condition as it is well maintained. The logging of the maintenance services also shows the party who conducted the services: for example, if it is conducted by a reliable and trusted workshop or car dealer.

2.1 Manual maintenance services logging system

A manual maintenance services logging system is where the process of uploading and storing the records of the services is performed manually. The workshop will issue receipts to show the list of services performed, or enter this information on the car's physical logbook.

In a manual maintenance logging system, a malicious entity can:

(i) Fake a signature to show that the service is conducted by a recognised dealer or workshop. If the process is manual and using a printed document, the document is stamped as a proof of signature. This stamp can be forged.
(ii) Fake a record to show that a service is conducted when it is not. Dates can easily be changed or faked.
(iii) Change the list of maintenance services conducted.

For a manual logging system, the manipulation could be conducted by the car dealer or the car owner. The purpose is to increase a car's value when reselling it [5,7]. The owner might also collude with a workshop to falsify the records. The car dealer might falsify the records themselves, or an untrustworthy workshop might falsify the list of conducted services, repairs or parts replaced to obtain a higher profit.

2.2 Automated services logging system

In this system, the process of recording and storing the log is automated, mainly operated by the workshop or car dealer. An automated process can ensure availability of data and ease of use. In the automated logging system, the potential storage locations are the electronic data logger, mobile devices or cloud server. The electronic data logger resides in the car and is connected to the CAN bus as one of the nodes. The mobile device is an external device (to the car), and it requires a connection to communicate with the in-vehicle network. The log could also be stored on a cloud server.

3 Proposed solution

The framework is shown in Fig. 1. The mobile device gives both the graphical user interface (GUI) and the connectivity. The mobile application supporting our proposed protocol is called AutoLOG. The process starts with the workshops updating in the car log of the list of services conducted, the date the service was conducted and the next of date of service. In order to communicate with the car, the workshop uses a diagnostic tool (DT). The diagnostic tool will communicate with the car through its Central Communication Unit (CCU). The CCU is a type of Electronic Control Unit (ECU). It is the first node any external device

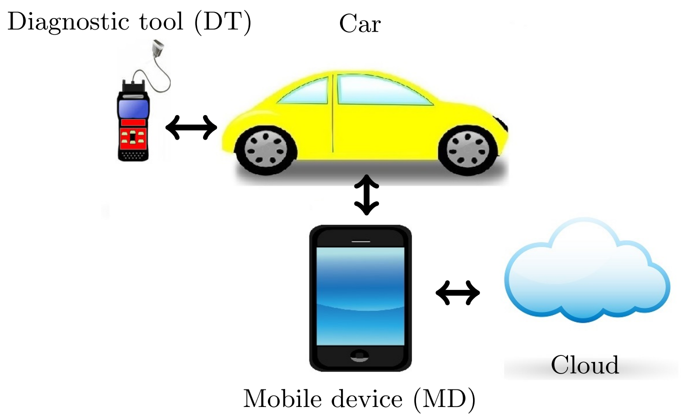


Fig. 1. Framework for the automated maintenance logging system

will have to go through in order to communicate with other ECUs. The related sensors and/or ECU(s) for the service will validate the information given to the CCU by the DT. After the validation, the CCU will store the latest record of maintenance services. The mobile device will then retrieve this data from the CCU and upload the data to the cloud. This way, the records are always available on both the mobile device and the cloud. In case the mobile device is lost, the data is always available on the cloud. In this proposal, the trust foundation is moved from the workshop to the car's sensors and ECU nodes. The cloud server could be owned by the user, trusted third party, community or government body. The cloud ownership is out of the scope of this paper.

Our proposal is based on the EVITA project [19], which proposed an embedded Hardware Security Module (HSM) in the ECU to ensure secure communications for on-board system. As proposed in the EVITA project, each ECU has its own HSM. This suggests that any node communicating through the CAN bus is required to have access authorisation in order to send or receive messages. In our proposal, the mobile device and diagnostic tool act as a communicating node through the CAN bus, and so requires access authorisation. This avoids the issue of unauthorised access to the in-vehicle networks, especially the CAN bus. The assumption on our proposal is based on the capability of the sensors in the car to validate the services being performed. While this may not be available now in current implementations, future firmware updates may introduce a version of this work.

3.1 Related work

There are many mobile applications available in the market that provide maintenance service logging systems [1–4, 9]. However, these applications require manual information to be entered by the car owner. After a car is serviced or repaired, the information can either be keyed in, or a photo of the document is captured to be stored. The data can later be stored in the cloud, depending on the application feature. Some applications require a manual upload to the cloud, while other applications will store the data automatically to the chosen cloud server.

Another type of maintenance service logging system is the one provided by car dealers [6, 11]. When a car is being serviced by a car dealer, details are uploaded onto the car dealer's server. When the next service date is nearing, the car owner will be contacted by the car dealer as a reminder.

A most recent development for car maintenance logging systems was introduced in "AUTObiography" by Motoriety [8]. This service logs the maintenance services record onto the cloud. Trusted workshops registered with Motoriety can use the service and will digitally sign the services conducted to be stored on the cloud. The data, or the "biography" of the car will then be available on the cloud, and can be passed from one owner to another. All the records are being managed by the service, can be retrieved by the car owners, and owners will get reminders for the next service date.

In the above-mentioned works, the trust is completely in the hands of the workshops. If an untrustworthy workshop fakes an item in the list of services conducted, nobody could prove this. On the other hand, a trustworthy workshop might mistakenly insert an item as a result of human error, since the process of recording and keying in the data is manual.

There are proposals for reminder notifications of the next service date [12,21]. There is also a system proposed using a passive radio frequency identification (RFID) device to detect the repairs/services being conducted [13].

Table 1 shows the added features of AutoLOG compared to other related works, which include manual and automated systems. AutoLOG, AUTObiography and a car dealer's cloud server provide automation, but the mobile applications do not [1–4, 9]. Data ownership of the records belong to the car owner in AutoLOG, AUTObiography and the discussed mobile applications. However, the ownership of the records in a car dealer's cloud server belongs to the car dealer. Data availability is supported by all the works discussed including AutoLOG. However, since the uploading process is manual for these mobile applications, data availability depends on this manual process. Unlike other related works, in our proposal, we consider the capability of the ECUs to validate the services. Security is a feature provided by all three automated systems. The car owners have the flexibility to choose from a range of different workshops for AutoLOG, AUTObiography and the discussed mobile applications. However, in the car dealer's cloud server system, the options of workshops are limited to the ones appointed by the car manufacturers.

The reason for not choosing TLS protocol for this application is because it is too much for CCU/ECU devices to cope with. The TLS protocol is bulky and has many implementation options. This will lead to more vulnerabilities. Our proposed protocol is very specific for this application, eliminating additional vulnerabilities. The TLS protocol is also slower in performance [38].

Table 1. Features of AutoLOG compared to other related works

Features	AutoLOG	AUTObiography	Mobile applications	Car dealer's cloud server
Automation	✔	✔	✘	✔
Data ownership	✔	✔	✔	✘
Data availability	✔	✔	✔	✔
Validation of services	✔	✘	✘	✘
Security	✔	✔	✘	✔
Options of workshops	✔	✔	✔	✘

3.2 Threat model

In the maintenance services logging system, assets to be protected are the read and write access authorisation and the authentication and integrity of the data. Potential attackers are untrustworthy workshops, owners and hackers with financial motivation, and potential buyers attempting to reduce the selling price. The two most likely threats are:

i) Dishonest mechanic charges owner for a full service, but may have done little/nothing.
ii) Owner changes service log to make the car more attractive to a buyer.

There are a number of possible attacks that could be performed in a digital maintenance logging system as follows:

(i) Denial of service (DoS) attack: to cause an availability issue, where data stored is not able to be retrieved, or data cannot be stored. Denying access of data to an authorised party is also a method of DoS.
(ii) Impersonation attack: to impersonate an authorised party to conduct further attacks, for example, an attacker impersonating an authorised workshop to log a record showing that the service is conducted by a certain trusted workshop.
(iii) Data manipulation attack: to change the list of the services, either by changing the data before or after the storage.
(iv) Replay attack: by replaying the same record of service to be stored on a different date to fake a record.

An additional assumption of the threat model in the digital automated system is the attacker cannot break well-established cryptographic algorithms.

3.3 Security requirements

From the architecture, the security requirements can be elicited [26]. In general, a maintenance services logging system should satisfy the following security requirements:

– Integrity: The data stored should not be changed, modified or added to, to ensure that the record of the maintenance services integrity is protected.

- Authentication: Data authentication and data origin authentication should be in place. This is to ensure that the data comes from an authorised party (workshop and car) and the data itself is authentic.
- Non-repudiation: To ensure that the data stored by the workshop can be verified, i.e., the workshop cannot deny that the data stored originated from its diagnostic tool and services were conducted by the workshop or dealer.
- Freshness: To ensure no replay attack is possible, hence a record of services cannot be logged/replayed if it is not actually performed.

The records should not be linked to a car owner's personally identifiable information (PII). Hence, privacy is not a concern in the maintenance services record, unlike various other applications involving cyber-physical systems [39].

3.4 Protocol goals

This section discusses the requirements of each party involved in the automated maintenance service log update.

(i) Car: The car requires authentication of the diagnostic tool, authentication of mobile device and data integrity of the information transferred from the diagnostic tool.
(ii) Mobile device (MD): The mobile device requires authentication of the car (CCU) and data integrity of the information transferred from the CCU.
(iii) Diagnostic tool (DT): The diagnostic tool requires authentication of the car (CCU).

3.5 Protocol assumptions and preconditions

Assumptions and preconditions on the successful use of AutoLOG are as follows.

1. The mobile application is installed on a mobile device and the cloud server is properly set up for the data to be stored.
2. The nonces generated (by DT, CCU and MD) should be random and not predictable.
3. The ECUs and sensors are equipped with the capability to validate the services being conducted on the car. For example, the sensor can validate the parameter given by the CCU, such as the serial ID of a new component. The proposal [13] could be used for this purpose. For a start, the firmware update status could be logged. Cars are now full of electronic modules that may require firmware updates. As part of the normal service, logging the status of all this firmware (which may then trigger updates) could be useful. So, when buying a second-hand car, not only does the potential buyer know it had a normal service on a particular date but also whether its IT/electronic systems have been "serviced" (kept up-to-date).
4. The cloud is securely managed. A user is authenticated to access the cloud server, and only authorised users have access to the data. However, even if an attacker is able to get access to the data in the cloud, the main concern is to protect the integrity of the data, which is provided by our protocol.

5. The data is always automatically transmitted to the phone and later to the cloud. If data is not updated after a certain time, the owner will be notified.

3.6 Protocol key distribution

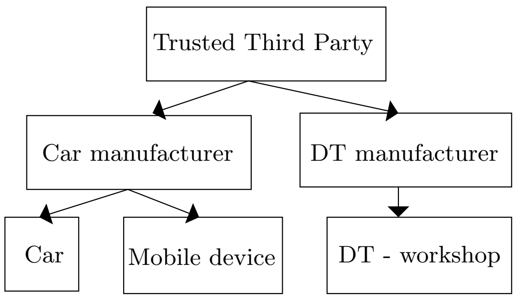


Fig. 2. Hierarchy for the key distribution

Figure 2 shows the hierarchy for the key distribution. The hierarchy may be implemented for a specific car manufacturer. A car manufacturer may have a list of trusted workshops and diagnostic tool manufacturers. Each diagnostic tool of a workshop has its own set of public and private keys, one set for signature and one set for encryption. The digital certificates contain the public keys of the diagnostic tools, which tie the diagnostic tool to a workshop, and are available at a diagnostic tool manufacturer server, which is under a trusted third-party server. The cars and mobile devices are registered under the car manufacturer, which is also under the same trusted third-party server of the diagnostic tool manufacturer. Each CCU has its own set of public and private keys, which are pre-installed during manufacturing. These keys are updated by the car manufacturer. The mobile device needs to be registered to the car manufacturer in order to communicate with the car. After the AutoLOG application is installed on the mobile device, the registration of the mobile device via the AutoLOG application will enable the mobile device to communicate with the car. Based on the input parameters during registration, which include the Vehicle Identification Number (VIN), the car manufacturer will share a symmetric key, k_{ccu-md} of the CCU with the intended mobile device. Similarly, the diagnostic tool will acquire the public key of the CCU from the trusted third party. The CCU, which is the master ECU in the car, has the records of all ECUs. The records of ECUs include their IDs, the hash content of the firmware and their symmetric keys to communicate with the CCU, $k_{ccu-ecu}$. The keys are stored in the HSM for the car (CCU and ECUs) and the diagnostic tool. For the mobile device, the keys are stored in a secure memory for example on a secure element.

Table 2. Notations for the protocol

DT	Diagnostic tool
CCU	Central Communication Unit
MD	Mobile device
dt	ID of DT
ccu	ID of CCU
md	ID of MD
pk_x	Public key of x, x= DT or CCU
sk_x	Private key of x, x= DT or CCU
na, nb, nc, nd	Message Authentication Code (MAC) keys
ne, nf	AES keys
$k_{ccu-ecu}$	Symmetric key shared between CCU and ECU
k_{ccu-md}	Symmetric key shared between CCU and MD
ENC	Encryption using RSA
enc	Encryption using AES128
$sign$	Sign using RSA
MAC	HMAC using SHA256
$a\|\|b$	a concatenate with b
$a \oplus b$	a XOR b
$mile$	Mileage
$servicetype$	Type of service (basic, full or major)
$servicedate$	Date of service
$nextdate$	Next date of service
$serviceupdate$	Command to conduct the log update
$serviceupdatereq$	Command to obtain the log update
$validateservice$	Command to validate service from CCU to ECU
$serviceupdateready$	Response from CCU to acknowledge it is ready with updated data
$ackready$	Response from ECU to acknowledge it is ready for validation
ack	Acknowledgement
$s1, s2, s3$	List of services, repairs and/or updates conducted

3.6.1 Protocol description To communicate with the car, the mobile device is connected to the OBD-II port via Wi-Fi or Bluetooth. Once connected, the mobile device will be authenticated, to determine whether it is authorised to retrieve the requested data. Once authenticated, the mobile device is connected to the CAN bus, and able to access the required data. The protocol notations are as shown in Table 2. The protocol, which is divided into three phases, is shown in Tables 3, 4 and 5. The first part (Table 3) shows the communication between the DT and CCU.

1. In the first message, the DT will send its ID, concatenated with update notification, *serviceupdate*, and a nonce, *na*. These parameters are signed with the DT's private key and encrypted with the CCU's public key. The digital signature is using signature with message recovery. The signature is then encrypted with the CCU's public key. The objective is to protect the secret nonce *na*, so that only the authorised CCU is able to obtain the value of *na*. The CCU will decrypt the message and verify the signature of the DT. From that, it will obtain the *na*, to be sent in the second message to the DT.
2. In the second message, the CCU will send its ID concatenated with the acknowledgment receipt of service update command and nonce *na*. It will be concatenated with its own generated nonce, *nb*. This message will be signed by its private key and encrypted with DT's public key. This signature is also using signature with message recovery and then encrypted with DT's public key in order to protect *nb*.

3. The DT will then decrypt the message to get the nonce nb. The nonces na and nb are used for MAC computation for the following messages between DT and CCU. The DT will then reply with all the required information, i.e. the maintenance type of service conducted (either basic, full or major), the service date, the next date of service to be conducted and the mileage reading, and concatenated with the MAC of all the parameters. The MAC is to ensure that the integrity of the data can be verified by the CCU.
4. The CCU will acknowledge the receipt of these parameters and concatenate it with the MAC.
5. Upon receiving the acknowledgment, the DT will send the list of services, repairs or updates conducted; in this example, they are $s1$, $s2$ and $s3$.

Table 3. DT-CCU update of services protocol

1. DT	: M1= $ccu\|serviceupdate\|na$
DT $\rightarrow$ CCU	: $dt\|ENC_{pk_{ccu}}\{sign_{sk_{dt}}\{M1\}\}$
2. CCU	: M2=$dt\|ack\|na\|nb$
CCU $\rightarrow$ DT	: $ccu\|ENC_{pk_{dt}}\{sign_{sk_{ccu}}\{M2\}\}$
3. DT	: M3= $ccu\|servicetype\|servicedate\|nextdate\|mile$
DT$\rightarrow$ CCU	: $dt\|M3\|MAC_{na\|nb}\{M3\}$
4. CCU	: M4=$dt\|ack$
CCU $\rightarrow$ DT	: $ccu\|M4\|MAC_{na\|nb}\{M4\}$
5. DT	: M5=$s1\|s2\|s3$
DT$\rightarrow$ CCU	: $dt\|M5\|MAC_{na\|nb}\{M5\}$

6. The next part is the communication between the CCU and the related ECU(s), as shown in Table 4. The CCU will validate the list of services, repairs and/or updates claimed by the DT. The related ECUs, equipped with sensors to verify the services/repairs/updates conducted, will respond accordingly. The CCU will send a command *validateservice* and a nonce nc, which is encrypted with $k_{ccu-ecu}$ to ensure only authorised ECU can read the nonce.
7. The ECU will decrypt the message to obtain the nonce nc. It will then send a message to acknowledge the receipt of nonce nc, and that it is prepared for the validation process, and will send its own generated nonce nd. This message is encrypted with the same $k_{ccu-ecu}$. The CCU will then decrypt the message in order to obtain the nonce nd. These nonces nc and nd are used for MAC computation for the following messages between CCU and corresponding ECU.
8. The CCU will send the list of services/repairs/updates conducted, concatenated with a MAC.
9. The ECU, after verifying the MAC received from the CCU, will validate each service/repair through its related sensors. After validating the list, it will send an acknowledgment of whether or not the validation is successful, concatenated with a MAC. If all items in the list are true, only the acknowl-

edgment is sent with a MAC. Otherwise, the failed item is included in the message.

Table 4. CCU-ECU validation of services protocol

6. CCU	:	M6=$ecu\|\|validateservice\|\|nc$
CCU → ECU	:	$ccu\|\|enc_{k_{ccu-ecu}}\{M6\}$
7. ECU	:	M7=$ccu\|\|ackready\|\|nc\|\|nd$
ECU → CCU	:	$ecu\|\|enc_{k_{ccu-ecu}}\{M7\}$
8. CCU	:	M8=$ecu\|\|s1\|\|s2\|\|s3$
CCU→ ECU	:	$ccu\|\|M8\|\|MAC_{nc\|\|nd}\{M8\}$
9. ECU	:	M9=$ccu\|\|ack$
ECU → CCU	:	$ecu\|\|M9\|\|MAC_{nc\|\|nd}\{M9\}$

10. The last part of the protocol is where the mobile device retrieves the list of services/repairs/updates from the CCU. The mobile device will send a message containing its ID concatenated with a command of *serviceupdatereq* and a nonce ne, which is encrypted with a pre-shared symmetric key between the mobile device and CCU, k_{ccu-md}. The encryption is to ensure the confidentiality of the nonce ne. Only the authorised CCU will be able to decrypt the message and obtain ne.
11. The CCU will decrypt the message to get the nonce ne and then will reply with a message telling that a new service is available. If the service has already been retrieved before, it will send a different message to inform the MD. The message contains the ID of the mobile device, *serviceupdate* reply, concatenated with nonce ne and its own generated nonce nf. They are encrypted with the pre-shared symmetric key between the mobile device and CCU, k_{ccu-md}. The nonces ne and nf are used for AES computation for the proceeding messages between the CCU and MD.
12. The MD will then decrypt the message to obtain the nonce nf and will then send an acknowledgment message to the CCU. This message is encrypted using the nonces as the key.
13. The CCU will then start sending the required service information to the MD, i.e., the maintenance type of service conducted (either basic, full or major), the service date and the next date of service to be conducted, the current mileage and the signature of this message. The signature is using signature with appendix. The signature is used to verify that the message originates from the CCU. The record transferred to the mobile device will not be able to be changed, because only the CCU has the private key to sign the message.
14. The MD, upon receiving these data, will be able to verify the origin of the message (i.e., CCU) by verifying the signature. It will then acknowledge the receipt of this message, in an encrypted message using AES128.
15. The CCU will next send the list of services/repairs/updates conducted. They are also appended with a signature for the same reason as in step 13, i.e. origin authentication and integrity protection.

16. Finally, the MD, upon receiving and storing these data, will send an acknowledgment encrypted using AES128 to the CCU. This will notify the CCU that the latest maintenance services record has been retrieved.

Table 5. MD-CCU request for services update protocol

10.	MD	: M10=$serviceupdatereq \| \| ne$
	MD→ CCU	: $md \| \| enc_{k_{ccu-md}}\{M10\}$
11.	CCU	: M11=$md \| \| serviceupdate \| \| ne \| \| nf$
	CCU → MD	: $ccu \| \| enc_{k_{ccu-md}}\{M11\}$
12.	MD	: M12=ack
	MD → CCU	: $md \| \| ccu \| \| enc_{(ne\oplus nf)}\{M12\}$
13.	CCU	: M13=$servicetype \| \| servicedate \| \| nextdate \| \| mile$
	CCU → MD	: $ccu \| \| enc_{(ne\oplus nf)}\{M13\} \| \| sign_{sk_{ccu}}\{enc_{(ne\oplus nf)}\{M13\}\}$
14.	MD	: M14=$ccu \| \| ack$
	MD→ CCU	: $enc_{(ne\oplus nf)}\{M14\}$
15.	CCU	: M15=$s1 \| \| s2 \| \| s3$
	CCU → MD	: $ccu \| \| enc_{(ne\oplus nf)}\{M15\} \| \| sign_{sk_{ccu}}\{enc_{(ne\oplus nf)}\{M15\}\}$
16.	MD	: M16=$md \| \| ack$
	MD → CCU	: $enc_{(ne\oplus nf)}\{M16\}$

3.7 Security analysis

The protocol is first analysed using informal analysis. Then, formal analysis is conducted using CasperFDR [24] and Scyther [16] tools to verify the protocol and provide indicative results.

3.7.1 Informal analysis of the protocol Based on the threat model discussed in the previous section, the protocol addresses them accordingly.

Denial of service attack (DoS) could be conducted:

(i) by stealing the mobile device. If the mobile device is stolen, all the records are still available on the server. A stolen mobile device would not be able to tamper with the available stored data, because the data is signed by the car's CCU.

(ii) by disabling connectivity between mobile device and CCU to disable the update. Since the logging process is automated, once a mobile device is authenticated to the CCU, it will ask for an update every time they are connected. If the update is not conducted, the owner will be notified.

(iii) by introducing manual errors. However, the process may repeat and retry the update. The diagnostic tool will likely abort after a few attempts. A notification message will be prompted after a certain retry limit. An error could occur in normal use; however it could also be evidence of an attack. The data will always be consistent, as the mobile device will verify with the

CCU whether the last data has been retrieved. If not, the CCU will retain the last record.

(iv) by causing the related ECUs/sensors to malfunction. During the second phase, i.e., the validation of the services, the ECU will acknowledge that the services are correctly being performed as given by the DT to the CCU in the previous phase. In this phase, all the related sensors will verify the correctness of the provided data. If any of the sensors fail, this will be displayed on the diagnostic transmission code (DTC, which is the error code) before the services is being performed. The faulty sensor should be fixed prior to updating the maintenance services logging system.

Impersonation of recognised workshop or dealer is prohibited with the use of digital signature to ensure only authorised DT can conduct the storing of information to CCU.

Data manipulation attack (change, deletion or insertion) could be conducted at three different stages:

(i) From the DT side: Digital signature is used to ensure that only authorised DT can sign the message required. Therefore, the message is authentic and comes from an authorised party, unless the private key is compromised.
(ii) After storing the information to the CCU, and during retrieval of data from CCU to the MD: The CCU only stores the last record of maintenance service conducted. This information is important to the car owner, in order to know the last service record. If the adversary wanted to modify or manipulate this one record, he/she needs to have the access to the CCU information, i.e., key to read and/or write to the specific memory address.
(iii) After storing the information in the mobile application or server: Fake records could be inserted to increase the car resale value. With this protocol, this is impossible because the record is protected by the CCU's signature to ensure its integrity is protected. The mileage can also prove the age of the car when the service is conducted.

Replay attack is not possible through the use of random nonces for each transaction.

The proposal also addresses all the security requirements discussed in Section 3.3, as follows:

– Integrity: The data stored could not be changed, modified or added. To ensure that the record of maintenance service is integrity protected, MAC and digital signatures are used.
– Authentication: Data authentication and data origin authentication should be in place. MAC is used to verify the data origin authentication.
– Non-repudiation: Digital signatures are used to ensure that the workshop and the car cannot deny their own data.
– Freshness: Freshness is verified by using nonces and the mileage reading.

3.7.2 Formal analysis of the protocol using CasperFDR and Scyther tools

The security requirements to be verified include confidentiality and authentication properties. Aliveness, agreement and synchronisation are part of the authentication property. Scyther is an automated tool for the verification of security protocols [16]. CasperFDR tool uses Communication Sequential Process (CSP) files to be analysed using Failure Divergence Refinement (FDR) [24]. CasperFDR and Scyther input scripts are as in link: CasperFDR and Scyther input scripts. The protocol is modelled as follows. The DT knows the CCU, but does not know the MD. MD only communicates with the CCU and not with the DT.

The protocol security objectives are key confidentiality and internal (CCU-ECU) and external (DT-CCU and MD-CCU) authentication. From our Scyther and CasperFDR input scripts, the following security claims are made and verified.

(i) Confidentiality: To verify the confidentiality of the cryptographic keys. The key confidentiality includes confidentiality of secret nonce (na, nb, nc and nd: used as the MAC keys, and ne and nf: used as the AES keys), and all secret keys (sk_{dt}, sk_{ccu} and sk_{md}).
(ii) Authenticity: To verify the authenticity of all entities involved in the process (DT, CCU and MD). This includes agreement and aliveness tests as defined in [15, 23]. In Scyther, additional authentication property, i.e., synchronisation is also verified. Synchronisation considers the content and ordering of the messages [15].

Analysis using CasperFDR The security properties verified are secrecy, aliveness and agreement. The confidentiality property is to verify the secrecy of the nonces (na, nb, nc and nd) that are used as keys for MAC computations, and (ne and nf) that are used as keys for AES computations. The aliveness property is to verify the aliveness between DT and CCU, between CCU and ECU and between MD and CCU. The agreement property is to ensure the agreement of variables shared between DT and CCU (na and nb), between CCU and ECU (nc and nd) and between MD and CCU (ne and nf). The threat model is that the attacker knows all the entities involved, i.e., the DT, CCU, ECU and MD, and their corresponding public keys. No known attack was found in the protocol.

The scripts are divided into three parts for the three different parts of the protocol. The full script for the first part (DT-CCU) can be found in link: CasperFDR input script. The script starts with #Free variables declaration, which declares all the variables used in the protocol. It is followed with the #Protocol description. This describes the messages being transmitted (in sequence) during the information passing from DT to CCU, which starts from service update notification (i.e., *1.a ->c:a,{{c,serviceupdate,na}{SK(a)}}{PK(c)}*). In *3. a ->c:a,c,service,mile,h(a,c,service,mile,na,nb)*, the list of services is passed from DT to CCU in clear text but appended with MAC of the message. It is the same in *5. a - >c:a,s1,s2,s3,h(s1,s2,s3,na,nb)*. Only DT and CCU can compute the MAC and verify them based on the shared keys in the previous message.

In the #Processes, all the involved entities in the protocol and their knowledge are declared. For example, *INITIATOR(a,c,serviceupdate,na,service,mile, s1,s2,s3) knows PK,SK(a)* , where a is the DT and c is the CCU.

The #Specification declares all the assertions made to verify the security properties. The confidentiality of na and nb are declared as Secret(a,na,[c]) and Secret(c,nb,[c]). As an authentication verification, the aliveness property between DT-CCU and the Agreement property between DT-CCU are verified.

The #Actual variables section describes the names of the actual agents, servers and the actual variables such as agent a is DT and agent c is CCU. In the #Functions section the public and secret keys are declared (symbolic PK,SK). The #System section again declares all the involved entities in the protocol and their knowledge, but with their actual names. For example, *INITIATOR(DT,CCU,Serviceupdate,Na,Service,Mile,S1,S2,S3).*
The #Intruder Information declares the intruder X who has the knowledge of all the entities involved and their public keys, and its own public and secret keys, i.e., IntruderKnowledge=DT,CCU,X,PK.

All the specifications made are verified and no attack is found for all the assertions.

Analysis using Scyther The security properties verified are secrecy, non-injective synchronisation, non-injective agreement and aliveness. The secrecy property is to verify the confidentiality of the nonces that are used as keys for MAC computations. The non-injective synchronisation property is to verify that parties know who they are communicating with, agree on the content of the messages and the order of the messages. The non-injective agreement is to verify that parties agreed on the content of the variables. The aliveness property is to verify that the intended communication partner (DT-CCU, CCU-ECU and MD-CCU) has executed some events. In Scyther, all the security properties are modelled in role-base. The properties are viewed from the local view of each role.

The full script for MD-CCU communication can be found in link: Scyther input scripts. In this section, the discussion is about the third part of the protocol, i.e., between MD-CCU. The script starts with functions declarations (line 1-4). Then, we have macros of messages to make the script neat and easy to follow (line 8-14). Next, the events and claims are made for each role (MD: line 16-42 and CCU: line 44-69).

For example, for MD role, the examples of events are *send_10(md,ccu,m10)* and *recv_11(ccu,md,m11)*, which means the MD sends the macro $m10$ to the CCU and later receives macro $m11$ from the CCU. Claims are the security properties to be verified. For example, for the MD role, *claim_I3(md,SKR, ne)* is for confidentiality. Authentication properties are verified through Agreement (*claim_I6 (md,
Weakagree)*, *claim_I2(md,Niagree)*), Synchronisation (*claim_I1(md,Nisynch)*), and Aliveness (*claim_I4(md,Alive)*).

The default verification setup was used (i.e., five maximum number of runs, type-matching and to find best attack with ten maximum patterns per claim). The results for all the claims made are verified as “Ok” in the “Status” with

"Verified" and "No attacks" in the "Comments". This means that no attack was found within the bounded or unbounded statespace; the security property has been successfully verified [14].

3.8 Protocol implementation

The protocol was then implemented on a PIC32MZ Microchip microcontroller and an Android device to obtain indicative performance results.

3.8.1 Implementation platform Our approach of implementation is to observe the computation time on the DT, CCU, ECU and the mobile device separately. The mobile device communicates via Wi-Fi, while the DT, CCU and ECU via CAN bus. There is a Wi-Fi module connected to the CCU to receive the Wi-Fi messages from the mobile device and convert these messages into UART messages. There is another interface module between the Wi-Fi module and the CCU to translate UART messages into CAN messages and vice versa. The DT, CCU and ECU are simulated using a microcontroller with all the functions required to be an actual ECU with cryptographic engines. PIC32MZ2048ECM144 [37] is chosen as the implementation platform for all three components (DT, CCU and ECU). It is a 32 bit microcontroller with 2048 KB of flash and 512 KB of SRAM, and operates at 200 MHz clock. It supports CAN bus communication, as required in an ECU. The hardware cryptographic engines support the computation of cryptographic algorithms to produce faster performance. For the mobile device, the application protocol is loaded into a LG Nexus 5 with a Quad-core 2.3 GHz Krait 400 CPU running on Android 5.1. PIC18F4580 is used as the interface module to translate UART-CAN messages. PIC18F4580 [29] is an 8 bit microcontroller with 32 KB of flash and 256 bytes of RAM. It operates with a 16 MHz clock and supports CAN bus and UART communication. For the Wi-Fi module, the Wi-Fi G demo board [36] is used.

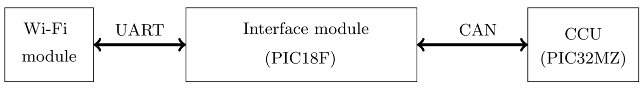


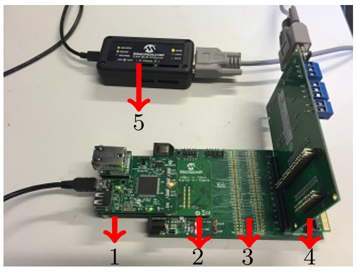
Fig. 3. CCU's setup for communication with MD

3.8.2 Experiment setup For the DT, CCU and ECU setup, the simulation of the messages from and to each component uses the Microchip CAN bus analyser tool [31]. The tool can be used to observe the messages sent from the PIC32MZ microcontroller and also to send messages to it. On the PIC32MZ part, the PIC32MZ2048ECM144 starter kit [35] is connected to a CAN PICtail daughter board [32] through a starter kit adapter [34] and an I/O expansion board [30]. The CAN PICtail daughter board is then connected to the CAN bus analyser.

The setup is shown in Fig. 4. The computation performance is measured based on cycle count given by MPLABX debugger.

For the interface module (using PIC18F4580), an additional CAN transceiver, MCP2551 [28], is connected to the PIC18. The interface module is then connected to MCP2200 breakout module [33] to observe the UART messages. The performance of communication is measured using an oscilloscope. The performance of Wi-Fi communication is measured using the “Inspector” feature from the internet browser.

All the protocol messages are in the data byte of the CAN message. The header of the CAN message is used in the same manner as in current implementation where it indicates what operation is to be handled. Based on the proposed protocol, the length of a message is more than eight bytes, hence, all the messages will need to be divided into more than one CAN message due to the limited number of bytes (8 bytes) of data per CAN message transmission. The messages are divided into three to eighteen messages to be transmitted via CAN.



1 PIC32MZ starter kit
2 Starter kit adapter board
3 I/O expansion board
4 CAN PICtail daughter board
5 CAN bus analyser

Fig. 4. Lab setup for DT, CCU and ECU through CAN bus communication

3.8.3 Performance results The computation and communication performance is as shown in Table 6. The communication includes the transfer of data from the Wi-Fi module to the middle interface module (via UART) and from the middle interface module to the CCU (via CAN). To the authors' knowledge, there is no related work that proposes an automated maintenance services logging system that we can compare the performance with. However, the total time for the protocol to complete is only about 883 ms. This shows that the protocol is efficient and practical for implementation. Although the computation time for AES and HMAC is faster using PIC32MZ as compared to the Android phone, the RSA computation is longer for PIC32MZ. This is because PIC32MZ has cryptographic engines for AES and HMAC which compute the algorithms at hardware level. Hence, this results in a faster computation time. The communication time is longer for the third part of the protocol, because the messages

Table 6. Protocol performance on LG Nexus 5 and PIC32MZ; for Protocol part I: A=DT, B=CCU, for Protocol part II: A=CCU, B=ECU, for Protocol part III: A=MD, B=CCU

Protocol part	**Message**	**Time(ms)**			
		Computation		Communication	Total time (ms)
		A	B		
I	1	53.041	52.691	1.825	107.557
	2	52.680	53.012	1.825	107.517
	3	0.102	0.086	0.859	1.046
	4	0.084	0.079	0.752	0.915
	5	0.077	0.086	0.752	0.914
II	6	0.099	0.050	0.537	0.686
	7	0.039	0.083	0.537	0.659
	8	0.103	0.083	0.859	1.045
	9	0.083	0.078	0.537	0.697
III	10	0.605	0.049	57.627	58.280
	11	0.805	0.083	72.818	73.705
	12	0.382	0.036	50.031	50.450
	13	1.216	39.459	163.962	204.636
	14	0.231	0.031	34.841	35.103
	15	1.082	39.609	163.962	204.652
	16	0.199	0.030	34.841	35.069
Total					882.933

from the mobile device need to go through Wi-Fi, be converted to UART messages, then to CAN messages. It is the same for the communication from the CCU to mobile device, where the messages from the CCU are in CAN, then converted to UART, and later to Wi-Fi. The baud rates of communication are at 9600 bps for UART and at 1 Mbps for CAN. The communication time can be further improved if CAN FD [18] is used, where one message can contain up to 64 bytes of data, instead of just 8 bytes.

4 Conclusion

The automated logging of car maintenance services helps car owners to keep track of the car maintenance record and avoid major breakdowns that can contribute to large costs. Having a secure protocol to conduct the automated logging can ensure that no records can be faked or modified. This will not only help the owner during the ownership of the car but also during car reselling, by increasing value of the car's price through showing that the car has been well maintained. The use of a mobile device gives a user interface as well as connectivity for the car, and thus helps the widespread use of this application since not all cars have connectivity and/or user interface. The proposed protocol provides integrity, authenticity and reliability of the data. It is also efficient and practically implementable.

References

1. aCar - Car Management, Mileage. `https://play.google.com/store/apps/details?id=com.zonewalker.acar.pro`. Last visited on 22/10/2016.

2. Auto Care. `https://itunes.apple.com/gb/app/auto-care-free-car-maintenance/id576958809?mt=8`. Last visited on 22/10/2016.
3. AUTOsist. `https://itunes.apple.com/gb/app/autosist-car-motorcycle-vehicle/id897916520?mt=8`. Last visited on 22/10/2016.
4. Car Minder. `https://itunes.apple.com/us/app/car-minder-plus-car-maintenance/id310809791?mt=8`. Last visited on 22/10/2016.
5. Fake Car Service Histories Spot One In 8 Easy Steps. `https://www.onlinespyshop.co.uk/blog/tips-guides/fake-car-service-histories-spot-one-in-8-easy-steps/`. Last visited on 22/10/2016.
6. Ford's Vehicle Repair and Service. `http://www.ford.co.uk/OwnerServices/VehicleServiceandRepair/ServicingyourFord/VehicleServicing`. Last visited on 22/10/2016.
7. How to Spot a Fake Service History. `http://www.autoexpress.co.uk/car-news/58853/how-spot-a-fake-service-history`. Last visited on 22/10/2016.
8. Motoriety. `http://motoriety.co.uk/`. Last visited on 22/10/2016.
9. My Cars. `https://play.google.com/store/apps/details?id=com.aguirre.android.mycar.activity`. Last visited on 22/10/2016.
10. The Value of a Full Service History. `http://www.telegraph.co.uk/cars/advice/the-value-of-a-full-service-history/`. Last visited on 22/10/2016.
11. Toyota Service History. `http://www.toyota.com/owners/parts-service/history`. Last visited on 22/10/2016.
12. F. Amouzegar and A. Patel. Vehicle Maintenance Notification System Using RFID Technology. *International Journal of Computer Theory and Engineering*, 5(2):312, 2013.
13. G. J. Boss, P. G. Finn, A. H. II Rick, B. M. O'Connell, J. W. Seaman, and K. R. Walker. Tracking Vehicle Maintenance using Sensor Detection, Nov 2012. US Patent 8,311,698.
14. C. Cremers. *Scyther User Manual*, draft edition, February 2014.
15. C. Cremers and S. Mauw. *Operational Semantics and Verification of Security Protocols*. Springer, 2012.
16. C. J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification*, pages 414–418. Springer, 2008.
17. ELM Electronics. ELM327L. `https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327L_Data_Sheet.pdf`. Last visited on 22/10/2016.
18. F. Hartwich. CAN with Flexible Data Rate, 2012.
19. O. Henniger. EVITA:E-Safety Vehicle Intrusion Protected Applications. Technical report, EVITA, 2011.
20. O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl. Security requirements for automotive on-board networks. In *Intelligent Transport Systems Telecommunications,(ITST), 2009 9th International Conference on*, pages 641–646. IEEE, 2009.
21. H. Hiraoka, N. Iwanami, Y. Fujii, T. Seya, and H. Ishizuka. Network Agents for Life Cycle Support of Mechanical Parts. In *Environmentally Conscious Design and Inverse Manufacturing, 2003. EcoDesign'03. 2003 3rd International Symposium on*, pages 61–64. IEEE, 2003.
22. Road vehicles – Controller Area Network (CAN) – part 1: Data link layer and physical signalling. Standard, International Organization for Standardization, February 2013.
23. G. Lowe. A Hierarchy of Authentication Specifications. In *Computer Security Foundations Workshop, 1997. Proceedings., 10th*, pages 31–43. IEEE, 1997.

24. G. Lowe. Casper: A Compiler for the Analysis of Security Protocols. *Journal of Computer Security*, 6(1):53–84, 1998.
25. R. Makowitz and C. Temple. FlexRay- A Communication Network for Automotive Control Systems. In *2006 IEEE International Workshop on Factory Communication Systems*, pages 207–212, 2006.
26. O. Mavropoulos, H. Mouratidis, A. Fish, E. Panaousis, and C. Kalloniatis. Apparatus: Reasoning About Security Requirements in the Internet of Things. In *International Conference on Advanced Information Systems Engineering*, pages 219–230. Springer, 2016.
27. Media Oriented Systems Transport Specifications, 2006.
28. Microchip. High-Speed CAN Transceiver. `http://ww1.microchip.com/downloads/en/devicedoc/21667d.pdf`, 2003. Last visited on 22/10/2016.
29. Microchip. PIC18F2480/2580/4480/4580 Data Sheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/39637c.pdf`, 2007. Last visited on 22/10/2016.
30. Microchip. Starter Kit I/O Expansion Board Information Sheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/51950A.pdf`, 2010. Last visited on 22/10/2016.
31. Microchip. CAN BUS Analyzer Users Guide. `http://ww1.microchip.com/downloads/en/DeviceDoc/51848B.pdf`, 2011. Last visited on 22/10/2016.
32. Microchip. CAN/LIN/J2602 PICtail (Plus) Daughter Board Users Guide. `http://ww1.microchip.com/downloads/en/DeviceDoc/70319B.pdf`, 2011. Last visited on 22/10/2016.
33. Microchip. MCP2200 Breakout Module User's Guide. `http://ww1.microchip.com/downloads/en/DeviceDoc/52064A.pdf`, 2012. Last visited on 22/10/2016.
34. Microchip. PIC32MZ Embedded Connectivity (EC) Adapter Board Information Sheet. `http://ww1.microchip.com/downloads/en/DeviceDoc/50002199A.pdf`, 2013. Last visited on 22/10/2016.
35. Microchip. PIC32MZ Embedded Connectivity (EC) Starter Kit Users Guide. `http://ww1.microchip.com/downloads/en/DeviceDoc/70005147A.pdf`, 2013. Last visited on 22/10/2016.
36. Microchip. Wi-Fi G Demo Board Users Guide. `http://ww1.microchip.com/downloads/en/DeviceDoc/50002147A.pdf`, 2013. Last visited on 22/10/2016.
37. Microchip. PIC32MZ Embedded Connectivity (EC) Family. `http://ww1.microchip.com/downloads/en/DeviceDoc/60001191F.pdf`, 2015. Last visited on 22/10/2016.
38. Pedro Miranda, Matti Siekkinen, and Heikki Waris. TLS and Energy Consumption on a Mobile Device: A Measurement Study. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 983–989. IEEE, 2011.
39. N. E. Petroulakis, I. G. Askoxylakis, A. Traganitis, and G. Spanoudakis. A Privacy-level Model of User-Centric Cyber-Physical Systems. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 338–347. Springer, 2013.
40. M. Ruff. Evolution of Local Interconnect Network (LIN) Solutions. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 5, pages 3382–3389. IEEE, 2003.
41. SAE J1962 Revised APR2002. Standard, SAE Vehicle Electrical and Electronics Diagnostics Systems Standards Committee, April 2002.