Secure Data Communication in Autonomous V2X Systems

Denis Ulybyshev, Aala Oqab Alsalem, Bharat Bhargava, Savvas Savvides, Ganapathy Mani
Computer Science Department, CERIAS
Purdue University
West Lafayette, USA
{dulybysh, alsalema, bbshail, ssavvide, manig}@purdue.edu

Lotfi Ben Othmane
Electrical and Computer Engineering Department
Iowa State University
Ames, USA
othmanel@iastate.edu

***Abstract*—In Vehicle-to-Everything (V2X) communication systems, vehicles as well as infrastructure devices can interact and exchange data with each other. This capability is used to implement intelligent transportation systems applications. Data confidentiality and integrity need to be preserved in unverified and untrusted environments. In this paper, we propose a solution that provides (a) role-based and attribute-based access control to encrypted data and (b) encrypted search over encrypted data. Vehicle Records contain sensitive information about the owners and vehicles in encrypted form with attached access control policies and policy enforcement engine. Our solution supports decentralized and distributed data exchange, which is essential in V2X systems, where a Central Authority is not required to enforce access control policies. Furthermore, we facilitate querying encrypted Vehicle Records through Structured Query Language (SQL) queries. Vehicle Records are stored in a database in untrusted V2X cloud environment that is prone to provide the attackers with a large attack surface. Big datasets, stored in cloud, can be used for data analysis, such as traffic pattern analysis. Our solution protects sensitive vehicle and owner information from curious or malicious information cloud administrators. Support of indexing improves performance of queries that are forwarded to relevant encrypted Vehicle Records, which are stored in the cloud. We measure the performance overhead of our security solution based on self-protecting Vehicle Records with encrypted search capabilities in V2X communication systems and analyze the effect of security over safety.**

Keywords-privacy; access control; database privacy; cloud data management; cloud security; encrypted search

I. INTRODUCTION

In V2X communication systems, vehicles and roadside objects can communicate and share data with each other. It is needed to ensure for the data owner that each V2X communication network node can access only those data items the node is authorized for. Role-based and attribute-based access control mechanism, which guarantees that unauthorized data accesses are denied, is required. In our approach, we use an Active Bundle [1] [2] [4] [6] to store sensitive data in encrypted form and to transfer sensitive data between V2X network nodes. Active Bundle (AB) is a self-protecting structure that consists of key-value pairs in encrypted form, access control policies and policy enforcement engine (Virtual Machine) [1]. It supports role-based and attribute-based access control in centralized and decentralized peer-to-peer networks. Our framework supports centralized data exchange, employing cloud for storing vehicle and owner data. This can be useful if it is necessary to query archived data and apply data analysis to big V2X datasets. Cloud platforms are vulnerable to large attack surface that could violate the privacy of data stored in cloud or shared with web services. It is necessary to protect sensitive vehicle and owner data from malicious or curious cloud administrators if data is stored in untrusted cloud. In addition to access control policies, used in role-based access control, our model for data exchange between V2X nodes considers client attributes [5], including the following:

1) Level of cryptographic capabilities of the browser, which sends data requests by means of http(s) messages.
2) Authentication method (password-based vs. hardware-based vs. fingerprint). Password-based authentication method is considered to be least secure.
3) Type of the device (Mobile vs. Desktop).
4) Trust level, which is constantly recomputed, based on the following metrics: CPU/Memory usage, number of sent/received data requests, number of denied data requests, number of communication failures.
5) Context (e.g., normal vs. emergency).

Our approach does not rely on Trusted Third Party (TTP) to issue keys for the recipient services, since keys for data request are generated on-the-fly by the Active Bundle kernel during data decryption phase, based on the execution flow. Our model supports complex access control policies that can be written in Java language [1], having an opportunity to express a variety of access control policies.

Untrusted cloud provider may host a database of Vehicle Records (VRs), where each VR is represented by an Active Bundle [1] [4], which contains vehicle and owner data in encrypted form. The database contains extra-attributes used for indexing, which are also stored in encrypted form. The database engine needs to be able to execute SQL queries, operating on encrypted data, in order to retrieve relevant VRs for further data processing. This is achieved through the use of Homomorphic Encryption (HE), which allows operations over encrypted data. Fully Homomorphic Encryption (FHE) allows arbitrary operations over encrypted data, but despite advances [7] still exhibits prohibitive overhead. Instead, we focus on using Partially Homomorphic Encryption (PHE), following the approach of CryptDB [3], which allows computation over encrypted data with respect to specific operations but exhibits practical overhead. When it comes to safety features of the V2X systems, heavy security features may come at a cost due to communication, computation, and storage overhead of the features. Expensive security solutions may hinder the safety of the system by prioritizing security features over safety.

This paper has two main contributions:

1) Privacy-preserving data exchange method for V2X communication systems, which provides data confidentiality and integrity for *data transfers in decentralized networks*. Our method supports both role-based and attribute-based access control. It also allows detecting and preventing multiple scenarios of data leakages made by insiders.
2) Capability of encrypted search over encrypted Vehicle Records, which are stored in untrusted cloud. Our method provides confidentiality and integrity of data stored in untrusted clouds. Subset of SQL queries over encrypted data is supported.

The rest of the paper is organized as follows: section II presents related work. Section III presents the core design of our system. Section IV evaluates performance of the system. Section V concludes the paper.

II. Related Work

In Intelligent Transport Systems (ITS) it is critical to ensure the privacy of the vehicles by not identifying their real identities. Both European standards (ETSI) and US standards (WAVE) have specific requirements to establish and maintain vehicles privacy. Based on the ETSI standards, anonymity, pseudonymity, unlinkability and unobservability are all required to establish privacy [8].

Ranchal et al [2] proposed a Framework for Enforcing Security Policies in Composite Web Services (EPICS), which protects data privacy throughout the service interaction lifecycle. The framework instantiates and extends the Active Bundle concept [4] for the Service Oriented Architecture (SOA) style. The solution ensures that the data are distributed along with the client policies, that dictate data access, and with an execution monitor that controls data disclosure. The framework empowers data owners with control of data disclosure decisions outside their trust domains and reduces the risk of unauthorized access. In this paper, we extended this approach with the following:

(a) Support of encrypted search over encrypted data records.

(b) Detection and prevention of multiple scenarios of data leakages that can be made by authorized insiders to unauthorized parties.

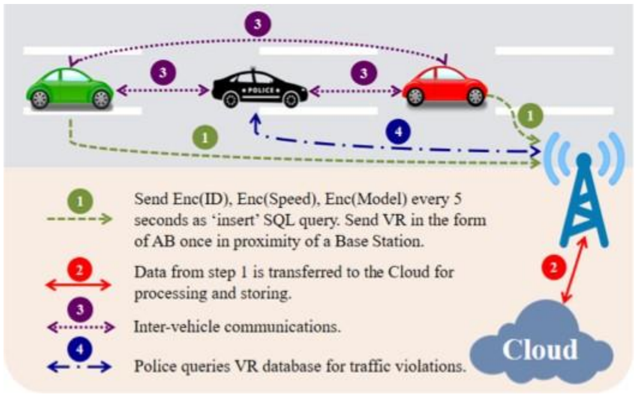
(c) Set of attributes used in our data dissemination model includes level of client browser's cryptographic capabilities, authentication method and type of the device / network.

Zhang et al [24] proposed an approach to detect malicious users and cloud providers, based on their trust levels. Network nodes are partitioned into domains. In our approach, client's trust level is used by an Active Bundle kernel to approve or reject data request issued by the client. Service is classified as a malicious one if it is detected that service is involved in leaking data to unauthorized entities.

In 2013 Microsoft came up with a CipherBase [9] product, which is a SQL database system for storing and processing strongly encrypted (i.e., using FHE) data. Cipherbase is based on a combination of customized trusted hardware and Microsoft SQL Server. It simulates FHE on top of non-homomorphic encryption schemes (e.g., AES in CBC-mode) by integrating trusted hardware. Application logic is decoupled from encryption. Our approach is to use PHE to achieve more practical encrypted search, and abandon FHE which can lead to slowdowns in the order of 10^9 times [10]. CryptDB [3] is a seminal work that demonstrates how PHE can be used to enable the execution of secure SQL queries over encrypted data with practical performance. Crypsis [11] and Cuttlefish [12] show that PHE can be used in cloud environments utilizing multiple computing nodes to perform data analytics of large datasets in a distributed fashion. Our approach uses PHE to provide encrypted search over encrypted set of extra-attributes used for indexing the database of VRs, that are also stored in encrypted form and provide role-based and attribute-based access control. Our framework provides confidentiality and integrity of vehicles and owners data, stored in VRs.

III. Core Design

A. System Architecture


3
3
3
1
1
4
1
Send Enc(ID), Enc(Speed), Enc(Model) every 5 seconds as 'insert' SQL query. Send VR in the form of AB once in proximity of a Base Station.
2
Data from step 1 is transferred to the Cloud for processing and storing.
2
3
Inter-vehicle communications.
4
Police queries VR database for traffic violations.
Cloud

* Figure 1. V2X Communication Network

In our architecture, vehicles and road infrastructure objects can exchange data between each other by means of VRs. VR contains owner and vehicle data in encrypted form and is implemented as an Active Bundle [1] [4], which is discussed in Sections I and III B. An example of a VR is given in Table 2. Data exchange between vehicles is shown as type 3 in Fig. 1. In addition, our solution supports SQL queries over encrypted database of extra-attributes used for indexing Vehicle Records database, stored in cloud. The idea is to improve performance of data requests to VRs by filtering relevant VRs in the first phase using an auxiliary

*Clipart taken from https://i.stack.imgur.com/EM34o.png; http://www.clker.com/cliparts/a/0/d/e/11971194531912449946Chrisdesign_Beetle_(car).svg.hi.png; http://www.clker.com/cliparts/R/7/e/b/5/X/red-car-hi.png; https://cdn1.iconfinder.com/data/icons/vehicles-roads-road-symbols/32/street-medium-512.png; http://www.clker.com/cliparts/s/E/z/g/z/V/cloud-hi.png; http://carpng.com/wp-content/uploads/thumb/police-car-icon-5642-0.png indexing database, which contains extra-attributes, and then querying only relevant VRs. As extra-attributes, we use

TABLE 1. ENCRYPTED INDEXING DATABASE INDEXDB

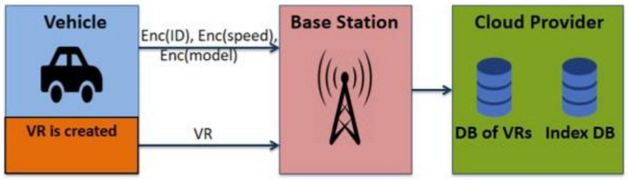
ID	Speed	Model	Timestamp
Enc(001)	Enc(65)	Enc(Toyota)	02/18/2018 15:28
Enc(002)	Enc(66)	Enc(Ford)	02/18/2018 15:29
Enc(003)	Enc(67)	Enc(Mercedes)	02/18/2018 15:31
Enc(004)	Enc(68)	Enc(Mitsubishi)	02/18/2018 15:44
⋮	⋮	⋮	⋮
Enc(1000)	Enc(84)	Enc(Chevrolet)	02/18/2018 23:59

vehicle ID, speed and model that are sent in encrypted form as insert SQL query to encrypted database, stored on a cloud provider, every 5 seconds. RSA encryption scheme is used for that. Vehicle has keys needed to encrypt ID, speed and model.

On a cloud instance, each received record is timestamped. Fragment of indexing database, *IndexDB* table, is given in table 1. ID is a primary key and it maps every record to VR, that is also stored on the same cloud. VR is created once when vehicle enters the proximity area of a base station. Base station forwards data from vehicles to the cloud provider and, thus, it is not expected from it to have any computational power or storage capacity. As a use case scenario, law enforcement vehicle might need to get a personal data of drivers who exceeded speed limits on the highway. Then, in the first phase, SQL query is sent to encrypted indexing database:

```
select ID from IndexDB
where speed > speed_limit;
```

In the second phase, http GET request for license plate number and drivers home address will be sent to relevant VRs, i.e. to VR with IDs returned by SQL query in the first phase.



** Figure 2. System Architecture

As a second use case scenario, the Intelligent Transport System might need to figure out the traffic pattern on a highway in a particular time interval, e.g. during the rush hour. In the first phase, SQL query is sent to encrypted indexing database, IndexDB, to get IDs of all the vehicles which are driving with the speed between 55 and 65 mph. It indicates no traffic jam and normal traffic conditions:

```
select ID from IndexDB
where speed between 55 and 65;
```

In the second phase, http GET request for license plate number and home address will be sent to relevant VRs, i.e. to VRs with IDs returned by SQL query in the first phase. It can be done in order to figure out vehicles from what state or region are the majority on the highway during the rush hour.

**Clipart taken from http://www.iconsplace.com/download/black-car-256.png; https://upload.wikimedia.org/wikipedia/commons/thumb/4/49/Applications-database.svg/2000px-Applications-database.svg.png; https://userscontent2.emaze.com/images/61d62748-ced8-4896-95ef-4c6742928511/68498c6a03c04904903928f0d81aa783.png

B. *Vehicle Record*

Our solution relies on Active Bundle (AB) for secure data exchange between V2X nodes. Each VR is represented as an extended AB, extended with data leakage detection capabilities and an extended attribute-based access control.

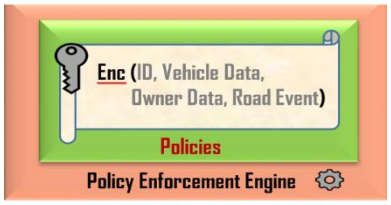

Enc (ID, Vehicle Data, Owner Data, Road Event)
Policies
Policy Enforcement Engine

Figure 3. Vehicle Record

AB is a self-protecting structure that incorporates data in encrypted form, access control policies and policy enforcement engine (Virtual Machine). Data is stored in AB as a non-relational database in the form of key-value pairs with encrypted values. Here is the example of key-value pair: `{ab.vehicleLicensePlate : Enc(123ABC) }`

License Plate value is 123ABC and it is stored in encrypted form. Each data item is encrypted with a separate AES symmetric key, which is generated on-the-fly based on the execution flow. VR data items are shown in table 2.

TABLE 2. VEHICLE RECORD DATA ITEMS

VR			
ID	**Owner's Info**	**Vehicle's Info**	**Road Events**
	• Name • Address • Phone • Driver's license number	• VIN • License plate • Health Check ➢ Engine temperature ➢ Fluids Level ➢ Tires pressure	• Traffic jam • Accident • Road work • Obstacle

Firstly, when a service, representing a vehicle, requests data from VR, hosted by another vehicle (see type 3 communication between vehicles on Fig. 1), the identity of the requesting service is verified. In the authentication phase, data requesting service presents its X.509 certificate signed by a trusted Certificate Authority (CA) to the VR [1]. If authentication succeeds, service attributes are evaluated and enforced by the policy enforcement engine, which is part of VR. If trust level of service is sufficient then process of evaluating applicable access control policies and context (e.g. traffic accident, emergency vs. normal) starts. It determines what data can be be disclosed to the requesting service. Based on that evaluation, symmetric decryption keys are derived to decrypt accessible data items. Details of communication procedure between web service and AB are covered in [2], [4]. Similar VR communication workflow takes place when data request is sent to the VR hosted by the cloud provider, e.g. in type 4 communication on Fig.1. For instance, type 4 might be used when law enforcement entity queries database of VRs in order to identify vehicles which exceeded the speed limit. After authentication phase, attributes of the client are evaluated. Client attributes include cryptographic capabilities of a browser [13] [14], if data request is sent from the web browser, and authentication method (password-based, fingerprint, hardware-based). This procedure is covered in details in [5]. The rest of the VR communication workflow is the same as described above for communication of type 3. Demo video of the prototype, illustrating this attribute-based access control concept, is available [15].

VR is tamper-resistant and provides integrity of the stored data. It protects data communications between services in V2X communication systems from man-in-the-middle and masquerade attacks, as well as from message tampering and fabrication. VR is written in Java and implemented as a Java Executable Archive (JAR). Access control policies are implemented using JavaScript Object Notation (JSON) [16]. Extensible Access Control Markup Language (XACML) [17] policy language is also supported for specifying access control policies. WSO2 Balana [18] is used for policy evaluation. Table 3 shows the example of access control policies for VR in V2X network.

TABLE 3. VEHICLE RECORD ACCESS CONTROL POLICIES

Allow				
Resource	Driver's License Number	VIN	Owner's Address	Traffic Events
Subject's Role	Law Enforcement	Law Enforcement, Car Repair	Law Enforcement, Car Repair, Insurance	Law Enforcement, Insurance, Other Drivers
Action	Read	Read	Read	Read

V2X network communication objects are represented as web services, e.g. Law Enforcement, Insurance, Car Repair, Driver. Client can be a vehicle or the computer representing Intelligent Transportat System or Law Enforcement. Since Raspberry Pi hardware has credit-card size and moderate power consumption, we assume it can represent vehicle's communication hardware.

Assumptions:

1) Hardware platform that runs VR is trusted. OS kernel is trusted, as well.
2) For communication between all the web services https protocol is used.

C. Encrypted Search

To allow querying vehicle and owner data without leaking sensitive information, we use PHE schemes to encrypt data items. This allows us to execute queries over encrypted data according to the homomorphic operations that these PHE schemes provide. More precisely, if $E(x)$ and $D(x)$ denote the encryption and decryption functions for input data x respectively, an encryption scheme is said to be partially homomorphic with respect to operation $\odot$ iff $\exists \otimes\ s.t. D(E(x1) \otimes E(x2)) = x1 \odot x2$. Table 4 shows the encryption schemes used in our approach along with the supported operations of each scheme and an example query. In Google Cloud we deployed a database of VRs, that store ID, road events, vehicle and owner data in encrypted form. As discussed in section III B, each VR is stored as an extended AB. On the same cloud provider we also deployed indexing

TABLE 4. OPERATIONS SUPPORTED BY DIFFERENT ENCRYPTION SCHEMES

Encryption Scheme	Homomorphic Property	Supported Operations	Example
Paillier	AHE	+, SUM	Count sum of tolls paid by vehicles on a highway
ElGamal	MHE	*	Count covered distance which is multiplication: time * average speed
Boldyreva et al.	OPE	<, >, MIN, MAX	select ID, Speed, Model from IndexDB where Speed between 71 and 80
SWP	SRCH	Tokenized search	select Model from IndexDB where issue LIKE %battery%
AES	DET	Exact search	select ID from IndexDB where Model = 'Ford'

database that contains extra-attributes used for VR indexing. In our case, extra-attributes are car ID, model and speed. Extra-attributes are stored in encrypted form as well, but as a relational database, not as key-value pairs, in contrast with VR. Each tuple in indexing database table is timestamped by cloud provider once data packet with ID, speed and model is received from the vehicle every 5 seconds (see Fig. 1). As illustrated on Fig. 4, ID maps each extra-attribute tuple in indexing database to its corresponding VR, i.e. detailed record of a given vehicle, that contains vehicle owner's personal information, vehicle registration information, VIN Number, results of the latest vehicle's health check, etc - see Table 2.

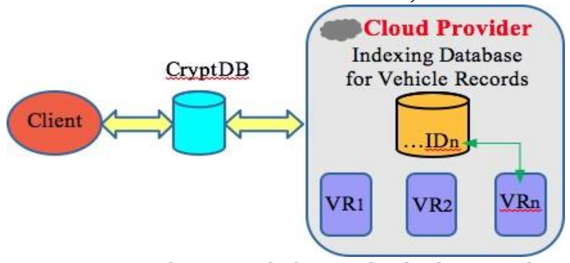


Figure 4. Mapping between Vehicle Record and indexing attribute

As it can be seen from Fig. 5, our encrypted search is implemented by leveraging the design of CryptDB [3] open-source database engine. When users submit SQL queries, a proxy at the client side intercepts the query and transforms it into a semantically equivalent query that operates over encrypted data. The transformed query is then deployed to the encrypted database. The functions shown in Table 4 are implemented as User-defined functions in the SQL server. CryptDB stores encrypted data and it is able to execute set of SQL queries, operating on encrypted data. It never releases decryption key to a database. Thus, when compromised, only ciphertext is revealed and data leakage is limited to data for users who are currently logged in. CryptDB provides confidentiality of Vehicle Records database, hosted by cloud provider. The crucial difference in our design compared to CryptDB is the two-layer approach in retrieving encrypted information. The first layer uses a set of extra-attributes (as described in the example in Section III A) which are appropriately encrypted using PHE to allow filtering and retrieving the relevant VRs. The second layer comes from the data protection mechanism provided by the VRs themselves.

Let us consider the use case when it is required for law enforcement entity to figure out personal data of drivers who exceeded speed limit of 65 mph on a highway and went above 76 mph. Then the required query will look like that:

```
SELECT ID FROM IndexDB WHERE SPEED > 76;
```

Converted query:

```
    SELECT c1 FROM Alias1
    WHERE ESRCH (Enc(Speed), Enc(76));
```

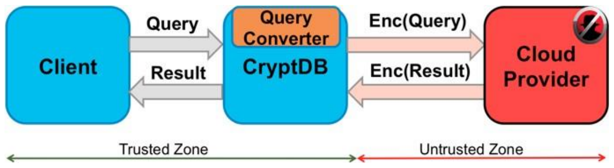


Figure 5. Encrypted Database Architecture

Result is received in the plaintext form at the client's side, but it is still encrypted on Cloud Providers side and, thus, is protected against curious or malicious cloud administrators.

D. *Safety vs. Security*

Safety is one of the vital aspects of V2X systems to maintain integrity and reliability of the system. However, security features add additional computation, communication, as well as storage costs to the system. The wireless communication devices in V2X systems are usually built with non-tamper-resistant hardware with basic configuration and often deployed in unpredictable and harsh environments [19]. Due to cost constraints, V2V systems are equipped with inexpensive processors that can process only limited information [20]. Information arrival rate (packet rate) in many traffic scenarios is very fast and, as a result, the verification of that information is slowed down. Packets that are not verified in a particular time epoch will be dropped from the security queue [21]. Security of the system will be compromised without the verification of packets in time, which causes packet loss. As Fig. 6 shows, V2X systems need to have a balanced approach among safety, security, and Quality of Services (QoS). But packet loss reduces the QoS of V2V significantly.

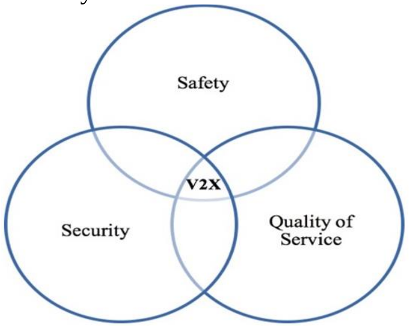


Figure 6. Safety and Quality of Service with Security Overhead

Several approaches have been proposed in literature to deal with this issue. Priority-based schemes are presented in [20]. [22] proposes processing of critical packets first and add others to the verification queue. Our research involves assigning a trust factor to the information provider.

IV. Evaluation

We evaluated performance overhead for different encryption schemes on V2X devices in section A. Section B has performance evaluation for inter-vehicle communication. Round-trip time (RTT) is measured between the moments when vehicle service issues data request to another vehicle and retrieved data are received by the requesting service. It includes authentication, authorization, data leakage check, key derivation, and data disclosure phases. In section C we evaluated performance of queries sent to VRs hosted by Google cloud instance. ApacheBench, ver.2.3, utility is used on client side for RTT measurements in sections B and C. Detailed configuration setup for our framework is available in the tutorial [23]. Section D has the evaluation of execution times for SQL queries, sent to encrypted database.

A. *Encryption Schemes Overhead of V2V Devices*

To keep generated data secure, V2X devices need to encrypt the data before sending them to the network. In this evaluation, we consider the encryption and decryption overhead of various encryption schemes on a device with relatively small computation capabilities, to evaluate the feasibility of encryption from the point of view of V2X devices. We compare the time needed for a) encryption and b) decryption between a server and a V2X device and show the results (average of 100 runs) in Figures 7 and 8. Specifically, for the V2X device we use Raspberry Pi Zero W, which is a widely popular device that has capabilities similar to devices found in vehicle tracking, highway cameras and various other IoT devices. Raspberry Pi Zero W comes with a 1GHz 32-bit single-core CPU and 512MB RAM. For the server we use a MacBook Pro with a 2.2 GHz Intel Core i7 CPU and 16GB RAM, running macOS Sierra 10.12.6. Encryption schemes were implemented in C using the SSL library (OpenSSL version 1.0.1t) and its BIGNUM primitive for big number computations.

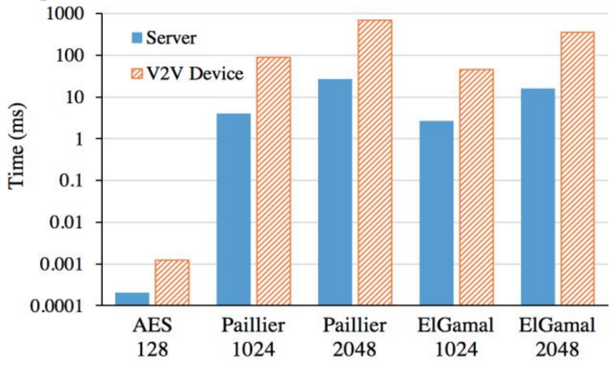


Figure 7. Encryption time comparison between a server and Raspberry Pi with crypto systems implemented using OpenSSL (logarithmic y-scale)

We observe that the average ratio of execution time across all encryption schemes on Raspberry Pi over the execution time on the server for encryption and decryption is 18.95 and 17.12 respectively. This overhead is acceptable, especially when the V2X device is not expected to make too frequent measurements. Furthermore, this overhead can be drastically reduced using pre-computation, as described in [22].

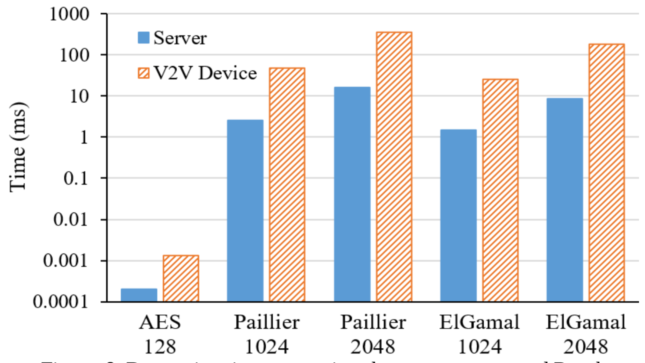


Figure 8. Decryption time comparison between a server and Raspberry Pi with crypto systems implemented using OpenSSL (logarithmic y-scale)

B. *Latency Evaluation for communication between two vehicles*

In this experiment, we simulate inter-vehicle communication by having two Raspberry Pi's communicating with each other. One vehicle (Raspberry Pi) hosts a VR in the form of AB and listens to the opened port 5555. Second vehicle (also represented by Raspberry Pi) sends data request to that VR over TCP/IP network, port 5555. Vehicle ID field is requested. We measure data request latency that also includes network delays. Two vehicles are represented by two Raspberry PIs with separate IP addresses.

Vehicles 1, 2 (Raspberry PI 3 Model B)
Hardware: ARMv7 Processor rev 4 @1.2GHz, RAM 1GB
OS: Raspbian GNU/Linux 9.1 (stretch)

Results on Fig. 9 represent inter-vehicle communication round trip time when first initial data request is sent to another vehicle. We consider it as a special case since the very first request to VR takes significantly longer to be executed due to initial authentication phase and initial evaluation of attributes. Using a VR with embedded attribute-based access control and tamper resistance, adds 6.4% performance overhead.

In the next experiment, we run 50 data requests in a row. As it can be seen from Fig. 10, mean RTT has been decreased 4.64 times for basic AB and 4.37 times for VR.

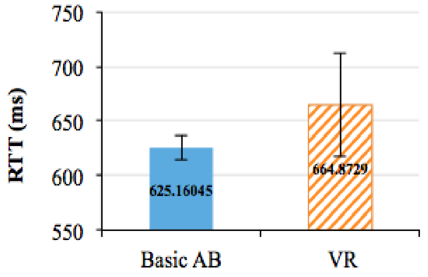


Figure 9. Inter-vehicle Communication Round Trip Time (initial request)

Using a VR with embedded attribute-based access control and tamper resistance, adds 13% performance overhead.

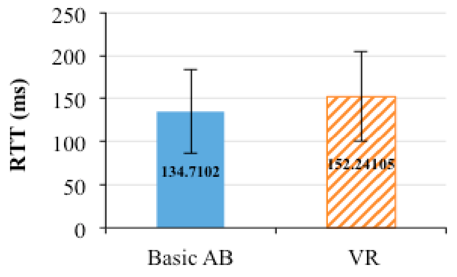


Figure 10. Inter-vehicle Communication Round Trip Time

C. *Latency Evaluation for data requests to Vehicle Records*

In this experiment we aim to measure latency of data request sent to VR, that is hosted by Google Cloud instance with the following characteristics:
Hardware: Intel(R) Xeon(R) CPU 2.30GHz
OS: Linux Debian 4.9.65-3+deb9u2 (2018-01-04) x86_64, kernel 4.9.0-5-amd64, 64 bit
Request for Vehicle ID is sent to VR, represented as an extended AB, and to basic AB, that, in contrast with VR, does not support neither extended attribute-based access control nor tamper-resistance nor data leakage detection. Both VR and basic AB run on a Google cloud instance that hosts a database of VRs and Indexing Database (see Fig. 4). Basic AB, as well as VR, has 4 access control policies. Examples of access control policies are given in table 3. Data request is issued from the service to basic AB and VR, running on a remote cloud service, which has external IP address. Thus, we measure RTT for a cloud data request and consider network delays between vehicle and cloud provider.

Results on Fig. 11 represent latency when first initial data request is sent to cloud. We consider it as a special case since the very first request to AB and to VR takes significantly longer to be executed due to initial authentication phase and initial evaluation of attributes. Using a VR with embedded attribute-based access control and tamper resistance, adds 5.1% performance overhead. In the next experiment, we run 50 data requests in a row. As it can be seen from Fig.12, mean

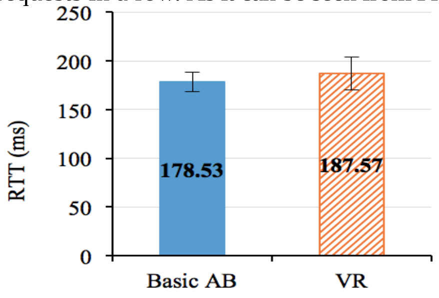


Figure 11. Performance overhead of Vehicle Record, hosted by Google Cloud (initial request)

value of RTT has been decreased 2.57 times for basic AB and 2.62 times for VR. Tamper-resistance support adds overhead since the hash value of an AB and its modules (code, access control policies, certificates) is verified by AB kernel when

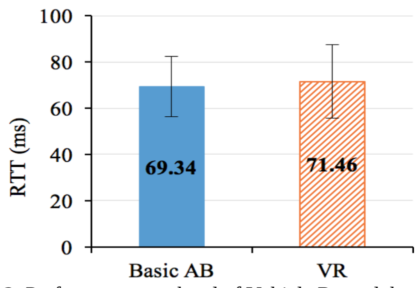


Figure 12. Performance overhead of Vehicle Record, hosted by Google Cloud

data request arrives. Detection of client's browser cryptographic capabilities and authentication method impose extra overhead since before responding to the client's request, AB needs to know the level of the browsers cryptographic capabilities and check whether it is sufficient. Authentication method is also processed by AB kernel. Using a VR with embedded attribute-based access control and tamper resistance, adds 3.1% performance overhead.

D. *Encrypted Search Evaluation*

In this experiment, we deploy two databases, a regular one where no encryption is used and therefore all sensitive information is leaked, and one that employs homomorphic encryption to keep data confidential while allowing search over encrypted data. Data request processing has two phases: (1) filter relevant VRs from the regular/encrypted database and (2) extract accessible requested data items from relevant VRs, stored as ABs.

We have performed the evaluation on a Linux Ubuntu 12.04.5 LTS machine (kernel 3.13.0-32-generic, 64-bit) with a 1.9GHz CPU and 1GB RAM using MySQL as our database engine. The database contains a single table named IndexDB and it was populated with 1000 tuples. Fragment of IndexDB table is given in table 1. In this experiment, the client issues the following five SQL queries:

Equality query (Q1):
```
SELECT ID FROM IndexDB WHERE model = 'Ford'
```

Inequality query (Q2):
```
SELECT ID,speed,model FROM IndexDB WHERE speed > 80
```

Inequality query, shortened (Q3):
```
SELECT ID FROM IndexDB WHERE speed > 80
```

Range query (Q4):
```
SELECT ID, speed, model FROM IndexDB WHERE speed
BETWEEN 71 AND 80
```

Range query, shortened (Q5):
```
SELECT ID FROM IndexDB WHERE speed BETWEEN 71 AND 80
```

Q1 retrieves all records that match an exact value for the model of the vehicle and therefore requires an exact match. Q2 retrieves all records with speed greater than a certain value and therefore it requires order comparisons. Q3 is the same as Q2 except that the query returns only the ID of the records. In the case the query is executed in an encrypted database, this query demonstrates the time spent in decrypting the results, in comparison to Q2. Similarly, Q4 and Q5 filter records whose speeds are within a certain range of values. We measure query execution time for plaintext database and compare it with the query execution time when data and search query are encrypted. We encrypted the model column using a deterministic encryption scheme and the speed column using an order-preserving encryption scheme.

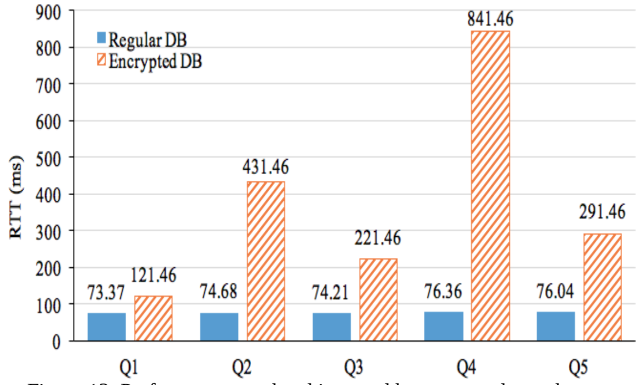


Figure 13. Performance overhead imposed by encrypted search over encrypted data

The results of this experiment are shown in table 5.

TABLE 5. SQL QUERY EXECUTION TIME FOR REGULAR/ENCRYPTED INDEXING DATABASE

	Query Execution Time (ms)	
	Regular Database	Encrypted Database
Q1	1.91	50
Q2	3.22	360
Q3	2.76	150
Q4	4.90	770
Q5	4.58	220

All reported times are the average of 50 executions. Query execution time grows 26 times for Q1 over encrypted database. This time includes the decryption time for the results. For Q2, the overhead is 112 times when all three columns need to be decrypted or 54 times when only the ID needs to be decrypted as in Q3. For Q4, the overhead 157 times and for Q5 the overhead is 48 times. In the next evaluation, we measure overall data request latency, which consists of two parts: (a) Query execution time for SQL query that is sent to indexing database in order to filter relevant VRs (b) Time to process data request by VR kernel. In our experiment there is one relevant VR filtered in phase (a) and then data request for vehicle owners home address is sent to relevant VR in phase (b). Results are shown on Fig. 13.

V. CONCLUSION

We presented a privacy-preserving data exchange model for V2X systems that provides data confidentiality and integrity. It supports both role-based and attribute-based access control, as well as detection of data leakages, made by authorized insiders to unauthorized entities. Evaluated attributes include trust level, level of client browser's cryptographic capabilities, client's authentication method and context. Our approach can be used for secure inter-vehicle data communications in untrusted environments. Operation in decentralized peer-to-peer networks is supported. We also support complex access control policies and on-the-fly data updates that can be made by authorized parties. Measured data transaction latency between two vehicles is 152 msec. Using a VR with embedded extended attribute-based access control and tamper resistance, adds at most 13% performance overhead, compared to the baseline, when tamper-resistance is not provided, and cryptographic capabilities of a client's browser are not evaluated. Our approach shows that security features for inter-vehicle data communication can be implemented at a small cost without overshadowing the safety features.

Our secure data access methodology can be also used to protect data, hosted by untrusted cloud provider, from curious or malicious cloud administrators. Sensitive vehicle data and owner data are stored in cloud in encrypted form and a large subset of SQL queries over encrypted VRs is supported. These SQL queries are used to filter out relevant VRs in the first phase of data request. Extra-attributes such as ID, speed and model are used for VR indexing. In the second phase, the data request is sent to the relevant VRs, stored in encrypted form. Measured execution time for phase 1 SQL equality query to encrypted indexing database of extra-attributes is 50 msec. Query execution time grows 26 times for equality query to encrypted database compared to querying plaintext database. Overall round-trip time for data request, sent to VR, is 121 msec. Extra attributes used for indexing are encrypted using a deterministic encryption scheme and an order-preserving encryption scheme. Our model supports range queries, the overhead for querying encrypted database is 157 times greater compared to querying plaintext database with range queries.

FUTURE WORK

We plan to work on a mechanism to provide inter-vehicle data communication in the presence of network failures. Impact of small "window of opportunity" for data transmission on transmission quality will be studied.

In addition, several experimental environments will be used to evaluate performance overhead in heterogeneous settings, with multiple vehicles and with varying amount of data exchanged between vehicles.

ACKNOWLEDGMENT

This publication was made possible by NPRP grant # [7-1113-1-199] from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors. The authors would like to thank Dr. Leszek Lilien, Miguel Villarreal-Vasquez and Servio Palacios for their help and valuable feedback.

REFERENCES

[1] R. Ranchal, "Cross-domain data dissemination and policy enforcement," PhD thesis, Purdue University, 2015.

[2] R. Ranchal, B. Bhargava, P. Angin, and L. B. Othmane, "Epics: A framework for enforcing security policies in composite web services," *IEEE Transactions on Services Computing*, 2018.

[3] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 85–100.

[4] L. B. Othmane, "Active bundles for protecting confidentiality of sensitive data throughout their lifecycle," PhD thesis, Western Michigan University, 2010.

[5] D. Ulybyshev, B. Bhargava, M. Villarreal-Vasquez, A. O. Alsalem, D. Steiner, L. Li, J. Kobes, H. Halpin, and R. Ranchal, "Privacy-preserving data dissemination in untrusted cloud," in *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*. IEEE, 2017, pp. 770–773.

[6] L. Lilien and B. Bhargava, "A scheme for privacy-preserving data dissemination," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 3, pp. 503–506, 2006.

[7] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 75–92.

[8] "ETSI - TS 102 941 - intelligent transport systems (ITS); security; trust and privacy management engineering 360," 2018. [Online]. Available: http://standards.globalspec.com/ std/1530232/etsi-ts-102-941

[9] A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan, "Orthogonal security with cipherbase." in *CIDR*. Citeseer, 2013.

[10] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in cryptology–crypto 2012*. Springer, 2012, pp. 850–867.

[11] J. J. Stephen, S. Savvides, R. Seidel, and P. Eugster, "Practical confidentiality preserving big data analysis." in *HotCloud*, 2014.

[12] S. Savvides, J. J. Stephen, M. S. Ardekani, V. Sundaram, and P. Eugster, "Secure data types: a simple abstraction for confidentiality-preserving data analytics," in *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 2017, pp. 479–492.

[13] "W3C web cryptography API," 2018. [Online]. Available: https://www.w3.org/TR/WebCryptoAPI/

[14] "Web authentication: an API for accessing scoped credentials," 2018. [Online]. Available: http://www.w3.org/TR/webauthn/

[15] D. Ulybyshev and B. Bhargava, "Secure dissemination of EHR demo video." Available: https://www.dropbox.com/s/4wg3vuv52j4s16v/NGCRC-2017-Bhargava-Demo1.wmv?dl=0

[16] "Lightweight data-interchange format JSON," 2018. [Online]. Available: http://json.org/

[17] "Extensible access control markup language (XACML) version 3.0," 2018. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[18] "WSO2 balana implementation," 2018. [Online]. Available: https://github.com/wso2/balana

[19] G. Mani, "Clone attack detection and data loss prevention in mobile ad hoc networks," *International Journal of Space- Based and Situated Computing*, vol. 5, no. 1, pp. 9–22, 2015.

[20] A. Belle, M. Falcitelli, M. Petracca, and P. Pagano, "Development of ieee802. 15.7 based its services using low cost embedded systems," in *ITS Telecommunications (ITST), 2013 13th International Conference on*. IEEE, 2013, pp. 419–425.

[21] M. A. Javed and E. B. Hamida, "On the interrelation of security, QoS and safety in cooperative ITS," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1943– 1957, 2017.

[22] J. J. Stephen, S. Savvides, V. Sundaram, M. S. Ardekani, and P. Eugster, "Styx: Stream processing with trustworthy cloud-based execution," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 2016, pp. 348–360.

[23] D. Ulybyshev, B. Bhargava, L. Li, J. Kobes, D. Steiner, H. Halpin, B.An, M.Villarreal, R.Ranchal, T.Vincent, "Secure dissemination of EHR in untrusted cloud," Project Tutorial, Purdue University, 2016.

[24] P. Zhang, Y. Kong, and M. C. Zhou, "A Domain Partition-Based Trust Model for Unreliable Clouds," IEEE Trans. Information Forensics Security, to appear in 2018