Towards Privacy-Preserving Platooning Services by means of Homomorphic Encryption

Nicolas Quero
Expleo France
nicolas.quero@expleogroup.com

Aymen Boudguiga
CEA LIST
aymen.boudguiga@cea.fr

Renaud Sirdey
CEA LIST
renaud.sirdey@cea.fr

Nadir Karam
Expleo France
nadir.karam@expleogroup.com

***Abstract*—Platooning is an upcoming technology which aims at improving transportation by allowing a leading human-driven vehicle to automatically guide multiple trucks to their respective destinations, saving driver time, improving road efficiency and reducing gas consumption. However, efficient linkage of trucks to platoons requires the centralization and processing of business-critical data which truck operators are not willing to disclose. In order to address these issues, we investigate how homomorphic encryption can be used at the core of a protocol for privately linking a vehicle to a nearby platoon without disclosing its location and destination. Furthermore, we provide experimental results illustrating that such protocols achieve acceptable performances and latencies at practical platoon database scales (serving around 500 simultaneous clients on a single platooning server processor core with sub second latency over databases of up to ≈60000 platoons scattered among over 250 destinations).**

I. Introduction

During the last decade, transportation systems have been evolving toward autonomous driving. In addition, they will soon benefit from wireless vehicle-to-vehicle and vehicle-to-infrastructure communications within Intelligent Transportation System (ITS) architectures. The combination of wireless connectivity and autonomous driving capabilities is creating new services not only for increasing vehicle safety but also improving traffic management as well as drivers and passengers comfort. However, most of these services collect data about drivers and their vehicles, challenging their privacy. In this work, we discuss privacy and confidentiality issues for the special application of platooning.

Platooning achieves a trade-off between autonomous and regular driving. It allows vehicles to automatically follow a leading vehicle driven (or not) by a human. One single driver can therefore drive a platoon of multiple vehicles towards their respective (and possibly different) locations. Currently, autonomous driving is easier to implement on highways rather than in more complex urban areas, and platooning tests are focusing on trucks more than on small vehicles. Indeed, putting several trucks in a common convoy reduces air drag and fuel consumption as well as improves safety, especially in freight corridors. Then, platooning grants truck drivers the possibility to find a nearby platoon when they need to have a break. Truck drivers can rest while the platoon is advancing towards its destination. Nevertheless, providing such a linkage service for a trucking company (or many trucking companies) requires accessing trucks locations and destinations which are sensitive data possibly leaking information about their business.

In this work, we propose a privacy-preserving protocol for platoons setup using Homomorphic Encryption (HE), one of the mainstream cryptographic techniques for computing over encrypted data thus preserving data confidentiality by design. More specifically, we propose a linkage service that does not need to access in clear form the sensitive data of trucks location or destination in order to provide sets of nearby platoons. In particular, the service relies on a central server referred to as the Platooning Service Provider (PSP) which makes computations over the clients encrypted data and then provides, as an encrypted output, adequate nearby platoons. As such, the PSP has no access to the clients clear data. However, Homomorphic Encryption is known for its intrisincs computational cost, and using it at practically relevant scales and latencies generally needs careful crafting of its integration in terms of using specific optimizations and minimizing as much as possible the footprint and complexity of encrypted domain calculations. With this in mind, we designed a protocol requiring only a few homomorphic operations per query (depending on the flavor) relying heavily on batching, a technique which allows using certain HE scheme in Single Instruction Multiple Data (SIMD) mode [26].

The remainder of this paper is organized as follows. Section II presents the state of the art regarding ITS privacy in general and location privacy in particular. Section III introduces Fully Homomorphic Encryption (FHE) and batching. Section IV describes the platooning use-case. Section V details our proposed protocol for privacy-preserving platoons selection. Section VI presents experimental results. Finally, section VII discusses perspectives and possible improvements for our protocol.

II. State of the art

A. Privacy in the ITS Context

The main privacy concern in the ITS context is remaining anonymous and untraceable at the network level [23]. A usual

Symposium on Vehicles Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23054
www.ndss-symposium.org

solution to this issue is signing vehicle-to-vehicle or vehicle-to-infrastructure communication with pseudonym certificates which are managed by a dedicated Public Key Infrastructure (PKI). Vehicles then periodically change their pseudonym certificates. Other solutions were proposed to ensure location privacy [2], [18], [27], [28]. These approaches consist mainly in using anonymization and obfuscation techniques. As another approach, Ghane et al., [13] relied on a Differentially Private Data Streaming (DPDS) system to ensure vehicles' data privacy when they are shared with untrusted Edge servers. Kargl et al., [17] also used differential privacy techniques to protect Floating Car Data (FCD). Höfer et al., [15] proposed a privacy-preserving charging for electrical vehicles called POPCORN. They relied on group signatures and anonymous credentials.

Rizzo et al., [24] trained a decision tree to classify drivers' behavior (as aggressive or defensive), while preserving the privacy of collected data and the confidentiality of the decision tree computed by the insurance company. They used a secure version of the ID3 algorithm to build the decision tree by using the homomorphic properties of Paillier's cryptosystem [22]. They also relied on Paillier's cryptosystem homomorphic properties during the classification phase. In 2019, El Ormi et al., [21] proposed a privacy-preserving k-means clustering for driving style recognition. They relied on multiparty computation for computing the distances to clusters centroids. Meanwhile, they used Paillier's cryptosystem during the computation of the new centroids. That is, as Rizzo et al., [24] they only needed a homomorphic additive scheme. Another approach using semi-private function evaluation and based on Yao's Garbled circuits was proposed in [14] for the calculation of tariff of car insurance companies while hiding users data as well as the insurance's private data. Boudguiga et al., [3] proposed to use homomorphic encryption for driver classification as concentrated or not with a small neural network of 4 layers.

B. HE Application to Location Privacy

In this paper, we focus on the problem of location privacy. In [1], An et al., proposed a COVID-19 contact tracing algorithm that checks whether a user has been in contact with a patient who had COVID-19 while ensuring that the users location data remains encrypted while the patients history of locations needs to be accessible by the authorities. The results showed that one comparison between a user and up to 8192 data points simultaneously was taking between 3 and 7 seconds on a smartphone, as well as needing 1 MB of data sent through the server (both ways). In [7], Carpio et al., combine additive homomorphic encryption with opportunistic networking to provide a protocol ensuring market equity for car-sharing applications.

III. Preliminaries on Homomorphic Encryption

In 2009, Gentry [11] made a breakthrough in cryptography by specifying the first Fully Homomorphic Encryption (FHE) scheme. He proposed a homomorphic encryption scheme E that computes $E(m_1 + m_2)$ and $E(m_1 \times m_2)$ from encrypted messages $E(m_1)$ and $E(m_2)$ (and keep on doing so indefinitely). Then, many leveled HE and FHE schemes have been proposed in the literature [4]–[6], [8]–[10].

In this work, we use BFV [4], [10] or CKKS [8] encryption schemes from the Microsoft SEAL library. They are two leveled homomorphic encryption schemes that allow to compute many additions and a bounded number of multiplications of ciphertexts. BFV and CKKS rely on the Ring Learning With Errors (RLWE) as the underlying hard problem [20]. BFV encrypts integers with RLWE samples, so it introduces some noise during the encryption. However, it provides a mechanism for noise removal during decryption which allows for recovering the exact result of the applied homomorphic computation.

On the other hand, CKKS considers the encryption noise as a part of the message, like when approximating real numbers using floating-point arithmetics. Note that this encryption noise does not disturb the most significant bits of the plaintext m as long as it remains small enough. That is, CKKS decrypts the encryption of m as an approximated value $m + e$ where e is a small noise. CKKS authors propose to scale plaintexts by a factor δ before encryption to reduce precision loss due to noise increase during computation.

Let us suppose that we will use the following cyclotomic polynomial (X^N+1), where N is a power of two, for defining our plaintexts and ciphertexts rings, for BFV or CKKS respectively. The plaintext space will be the ring $\mathbb{Z}_t[X]/(X^N + 1)$ for BFV, and $\mathbb{Z}[X]/(X^N + 1)$ for CKKS. Meanwhile, the ciphertext space will be $\mathbb{Z}_q[X]/(X^N + 1)$.

BFV can encode a vector of integers as a single plaintext polynomial if t is prime and $t = 1 \mod 2N$. This plaintext polynomial is then encrypted. Meanwhile, CKKS encodes a vector of Gaussian integers as a single polynomial in the ring $\mathbb{Z}/(x^N + 1)$ before encryption. Indeed, BFV and CKKS support *batching*, a Single Instruction Multiple Data (SIMD) encoding.

With batching, each entity E_i can encode a vector of d messages $(m_{i,1}, \ldots, m_{i,d})$ in a single plaintext polynomial p_i. d is equal to N with BFV or $\frac{N}{2}$ with CKKS. Then, p_i is encrypted either with BFV or CKKS to obtain a ciphertext polynomial c_i. Let us consider two batched ciphertexts c_1 and c_2 encrypting $(m_{1,1}, \ldots, m_{1,d})$ and $(m_{2,1}, \ldots, m_{2,d})$, respectively. Adding c_1 to c_2 outputs c_{add}, a batched ciphertext of $(m_{1,1} + m_{2,1}, \ldots, m_{1,d} + m_{2,d})$. Meanwhile, multiplying c_1 by c_2 outputs c_{mul}, a batched ciphertext of $(m_{1,1} \times m_{2,1}, \ldots, m_{1,d} \times m_{2,d})$.

One interesting application of batching is Private Information Retrieval (PIR) [12], a technique which allows to extract a record from a database without revealing which one. A PIR request consists of sending an array of ciphertexts with the same size as that database. This array contains encryptions of 0 in all positions, except for that corresponding to the information that we want to retrieve where it contains encryption of 1. When using RLWE-based schemes, PIR can be implemented more efficiently with batching. For example,

we consider the batched ciphertext c_1 encoding $(m_1, \ldots, m_d)$ (where d is even). We suppose that we want to obtain a batched ciphertext c_3 encoding only message with odd index. In this case, we have just to multiply c_1 with the batched ciphertext c_2 encoding $(1, 0, 1, \ldots, 1, 0)$, to get c_3 encoding $(m_1, 0, m_3, \ldots, m_{d-1}, 0)$.

In the following, we will rely on batching with BFV and PIR when describing our protocol for private platooning. That is, d will be equal to N.

IV. Platooning Linkage Description

Suppose that a truck driver on the road is about to reach a time when (by law) he has to rest. He would normally have to drive until the next highway station where he would rest for prescribed period. Platooning offers this driver the possibility to join and automatically follow a nearby convoy rather than stopping for a while. To find a nearby platoon, a driver installs an application on his truck ADAS to search for a nearby platoon. An example of simple protocol for platoons linkage is as follows:

1) The driver indicates to the platooning application that he/she wants to join a platoon.
2) The application sends the driver location and destination to a remote Platooning Service Provider (PSP).
3) The PSP compares the driver location and destination with a platoons database.
4) The PSP sets a meeting point for the driver and the platoon.
5) The driver joins the platoon at the meeting point.

This protocol can be easily implemented when drivers locations and destinations are provided in clear to the PSP. However, these are sensitive information that a transportation company is not willing to share mainly due to business restrictions. This issue can be overcome by making the PSP compute the platoon choice over encrypted data. Amongst other techniques, this can be achieved by means of Homomorphic Encryption (HE) techniques as we investigate in the sequel.

In the following sections, we refer by client to a truck driver using the platooning application for joining a platoon. This application manages the platooning linking requests and answers for every driver. This local app is also in charge of messages encryption, decryption and key management.

V. Privacy-Preserving Platooning Linkage

In this section, we first describe our considered threat model in section V-A. Then, we detail our proposed protocol for a private selection of platoons in sections V-B and V-C. We discuss key management in section V-D and HE scheme choice in section V-E.

A. Threat Model

In this work, we consider that all clients are honest entities. Meanwhile, we consider that the PSP is honest-but-curious.

In the honest-but-curious model, many entities $(E_1, \ldots, E_n)$, having as secret information $(s_1, \ldots, s_n)$, participate to a protocol P to compute a function $F(s_1, \ldots, s_n)$. Each entity $E_{i,i\in[1,n]}$ is assumed honest and must follow each step of P. However, $E_{i,i\in[1,n]}$ is curious. That is, $E_{i,i\in[1,n]}$ will try to extract information about other entities secrets $s_{j,j\neq i}$. P is secure in the honest-but-curious model if each $E_{i,i\in[1,n]}$ has no other information than $F(s_1, \ldots, s_n)$ at the end of the protocol. In our case, the PSP will try to recover information about the location and destination of the clients when processing their requests.

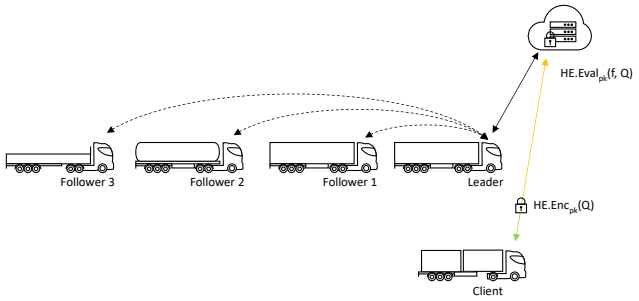

Fig. 1. Platooning linkage example

In addition, we focus here on ensuring the confidentiality of data in use on the server and do not consider other security properties such as entities authentication or message integrity validation. We can extend our protocol by "off-the-shelf" cryptography algorithms to provide such properties[1].

At this stage, it should be emphasized that the honest-but-curious threat model is too weak for real-world deployment scenarios and, as such, that it is not reasonable to rely on homomorphic encryption alone in such a context. In the real-world, homomorphic encryption should therefore be considered only as what it is: a countermeasure to confidentiality threats from the server doing the calculations. Therefore, in real-world deployment scenarios, homomorphic encryption must always be embedded in a higher level protocol resorting to additional off-the-shelf techniques e.g., for strong authentication of all parties or confidentiality and integrity on all exchanges.

B. Protocol Description

Figure 1 presents the privacy concerns addressed by our protocol. First, we assume that the PSP has a clear database for the locations of the platoons leaders. We suppose that platoons locations are public knowledge as opposed to trucks locations. The platoons leaders can be either special vehicles dedicated for platooning (in which case they have no privacy requirements, especially if they obey a fixed schedule), or normal trucks that will get a retribution in exchange of their leadership (and privacy loss). As such, the leaders will update

[1] We refer, for example, to message authentication codes to provide message integrity

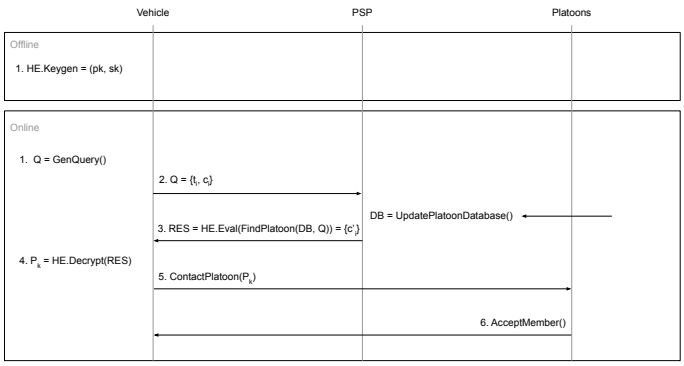

Fig. 2. Privacy-Preserving Protocol for Platoon Linkage

Time of prediction (min)	Road 1 Segment 1	Road 1 Segment 2	…	Road R Segment N
5	P1, P14, P17	P9, P13	…	P47, P4
10	P1	P9, P14, P17	…	P6
15	∅	P1	…	∅
…	…	…	…	P19, P15

Non encrypted database

Time of prediction (min)	Road 1 Segment 1	Road 1 Segment 2	…	Road R Segment N
5	[∅]	[∅]	…	[∅]
10	[∅]	[P9, P14, P17]	…	[∅]
15	[∅]	[∅]	…	[∅]
…	…	…	…	[∅]

Encrypted output

[] : FHE encryption under the key of the client

Fig. 3. Platoons Database Structure

the platooning database, without disclosing information about the members of their respective platoons.

We present an example of a platooning database in Figure 3. This database gathers platoons with respect to their locations using a time-step of 5 minutes. The database is organized in two dimensions with rows and columns indexed by time and location, respectively. Note that a location corresponds to the current road segment of a platoon leader. Every platoon leader is identified by a number P_k.

When a client wants to join a platoon at a certain time-step, it generates a request Q to the PSP. The latter contains the time-step t_i and a PIR request c_i encrypted with the client homomorphic public key. c_i is a batched ciphertext with almost all the slots equal to 0 except for the index that corresponds to the target platoon location. For example, let us assume that a client wants to retrieve the identifiers of the platoons at the road segment indexed by j for $j \in [\![1, N]\!]$.

Original set	$\{3$	11	7	$1\}$
Output binary set	$\{0011$	1011	0111	$0001\}$

Fig. 4. Example encoding of a set of platoons with $E = 4, M = 16$

The client encrypts in c_i a plaintext encoding the following slots: $(0_{i,1}, \ldots, 0_{i,j-1}, 1_{i,j}, 0_{i,j+1}, \ldots, 0_{i,N})$[2]. Then, the client sends $Q = \{t_i, c_i\}$ to the PSP.

The PSP runs FindPlatoon() from step 2 in Figure 2. The PSP gets the database row corresponding to the time-step t_i and converts it to a plaintext p_i encoding $(s_{i,1}, \ldots, s_{i,N})$. Each $s_{i,j}$ is a set of identifiers for platoons that can be joined in a time-step t_i at road segment (or location) j. Then, the PSP multiplies c_i by p_i (as described in section III) to get a ciphertext c'_i associated to the plaintext encoding the following slots: $(0_{i,1}, \ldots, 0_{i,j-1}, s_{i,j}, 0_{i,j+1}, \ldots, 0_{i,N})$. Finally, the PSP sends $RES = \{c'_i\}$ to the client.

The client decrypts c'_i to recover the set of platoons identifiers $s_{i,j}$. He has only to choose one of the platoons in order to join it. The client can contact this platoon directly to check whether he can join it at the given time or not. That is, given a platoon identifier P_k, we assume that the client is able to privately retrieve additional data about the platoon in order to contact it. This step requires an additional privacy-preserving interaction because the client cannot disclose to the server the platoon it wants to join. This interaction can be performed for example with a PIR request, a direct low-range communication with the platoon, or any other adequate solution.

We give in Figure 3 an example of a result obtained after a client asked for the element of index $[i = 2, j = 2]$, where the brackets mean that $s_{i,j}$ are encrypted under the FHE of the client.

C. Platoons Identifiers Encoding in Slots

We describe in this section how we encoded the platoons identifiers associated to one road-segment (i.e., location) into one slot. We remind that when using batching with BFV (as presented in section III), a slot is an element of $\mathbb{Z}_t$ and so has a value between 0 and $t-1$. In practice, t can be quite high, for example with Microsoft SEAL library [25], t can be up to 60-bit long.

We propose to encode at one slot a set of up to E platoons (meaning that a set in one cell of 3 contains a maximum of E platoons identifiers). If for example $t = 2^{20}$ and $E = 4$, we have 20 bits to encode up to 4 platoons identifiers. Therefore, one platoon identifier must not exceed 5 bits so the maximum platoon ID is here $2^5 = 32$. More generally, we can encode M platoon identifiers in a slot of size t, where $M = \lfloor \frac{log_2(t)}{E} \rfloor$.

We give in Figure V-B an example of a set of platoon encoding.

D. Key management

As we will see in Section VI, the protocol easily scales to practically relevant dimensioning and is naturally suited to a

[2] N is the number of slots (described in section III) and the degree of the cyclotomic polynomial as we are using the BFV scheme.

multi-client setting where each client can use its own FHE keys obliviously to the PSP. The reason for that is because the platoons database is not encrypted (or at least, not encrypted under FHE [3]) and the only operation necessary from the PSP are plaintext-ciphertext multiplication. Therefore, we do not require the PSP to use a public key nor an evaluation key for LWE-based homomorphic cryptosystems (this claim would not be valid for other additive-only schemes such as the Paillier cryptosystem). Still, this observation would not be valid if the database were encrypted in FHE, as a client request would have to interact with a database encrypted under the same FHE key. In that latter case, the PSP would have to resort to the more complex machinery of homomorphic key switching or a full-blown multi-key FHE solution. So, being able (from a confidentiality point of view) to let the platoon database in clear form and using xLWE-based HE schemes leads to multi-key support "for free" which is an interesting property. Under that assumption, the PSP can simply receive a ciphertext from any client (encrypted under that client FHE key) and multiply it with its database to produce the result *without having to use any public data or key linked to the client*.

E. Choice of the HE Scheme

As already emphasized, we designed our protocol so as to minimize the footprint of encrypted-domain computations. First, we avoid multiplications between ciphertexts as it is the most costly homomorphic operation. However, we do not use an additive homomorphic encryption scheme like Paillier [22] because the use of the BFV scheme[4] allows large gains in efficiency due to its SIMD-like homomorphic computations abilities [16]. Besides, the choice of RLWE-based schemes allows for straightforward plaintext-ciphertext computation which does not require explicit knowledge of any public or evaluation key from the server. Thus, the PSP does not have to store any client homomorphic public key as would be needed when using the Paillier scheme for example. Another class of homomorphic encryption schemes exists, providing only bitwise operations (XOR/AND). The most famous such scheme is the gate flavor of TFHE [9], which has the benefit of allowing for unbounded sequences of operations thanks to an efficient bootstrapping procedure. However, as TFHE does not support batching, the operations it provides do not allow an efficient implementation of our protocol (contrary to BFV, BGV or CKKS, as already discussed).

VI. Experimental Results

A. Realistic System Dimensioning

In this section we propose a set of parameters along with an implementation of the protocol aiming for a full scale deployment in a moderate size country such as France. In particular, we need a service able to handle a sufficient number of platoons with reasonable latency (say less than 10 s). For example, 100000 trucks is a conservative upper bound on the daily truck traffic on the French highway system and 10% of that number is a conservative upper bound for the number of platoons per day. To instantiate our protocol, we propose a set of parameters which achieve an appropriate trade-off between efficiency and scale. We choose $N = 8192$ so a ciphertext has 8192 slots exploitable with batching. In each slot, we can encode a number up to a plaintext modulus t which we define to be a 60 bits number[5] (which has to be congruent to $1 \mod 2N$). Therefore, we have 60 bits that we can split as explained in Section V-C (actually 1 bit has to be removed because all the 60-bit numbers are not accessible as we perform operations modulo a 60-bit number but we omit this detail as it only requires to add one more bit which has very little impact on performances). We split those 60 bits in 4 15 bits entries, meaning that a set in the database in Figure 3 will be of size 4. Then, we are left with 15 bits for each platoon to represent its ID composed of a destination ID on 8 bits and a platoon number on 7 bits. This means that we can encode up to 255 destinations (0 stands for no platoon), and 128 platoons per cities. With these settings, we can have 128 different platoons for every 256 destination (note that a country such as France counts around a 100 districts, each with 3 to 4 large cities, so the orders of magnitude roughly match). Note that some destinations may expect more platoons than others and the encoding can be adjusted accordingly. For example, halving the encoding space of the least common destinations can free some space that we can use to handle more platoons in the most common destinations.

B. Performance Results

We implemented the most computationally expensive part, the multiplication between the request and the platoons database where the client asks for one time interval. We did our tests on a Intel(R) Core(TM) i5-1145G7 @ 2.60GHz 1.50 GHz (with only one core activated) and 8 GB of RAM. We used the Microsoft SEAL library with the default coefficient modulus ($218 = 43 + 43 + 43 + 44 + 44$ bits) for 128 bits of security. The following table gives the average time per homomorphic operation of 10000 experiments under the parameters set above.

Operation	Time (ms)
Encryption	2.84
Plaintext/Ciphertext Multiplication	1.99
Decryption	1.10

Note that the coefficient modulus can be lowered to improve performances as well as security (see [19]). This is because we perform non costly operations and the larger q is, the more clear equivalent operations we can achieve per homomorphic ones. As such, these unitary performances lead to very practical perspectives: in particular they are consistent with a system able to reply with a sub second latency (still of course depending on the delays of the wireless telecom

[3]Symmetric encryption techniques can be used to encrypt the platoons database but this deserves more investigation. See also discussion in VII-B

[4]As would be the case for other RLWE-based scheme such as, for example, BGV.

[5]Larger modulus can be used, however 60 bits is a common limitation of concrete librairies due to machine word constraints.

infrastructure). From the infrastructure point of view, it is the homomorphic computation on the PSP side which defines the critical path and dimension the computing resources (i.e., the single homomorphic multiplication which runs in about 2 ms). Therefore, a single PSP processor core can serve about 500 *simultaneous* clients in a second. Therefore, we expect that a 10 to 20 core server is enough for handling up to 100000 daily clients. Provided current multi-core server costs, this is clearly reasonable. Having said that, since both the query and response ciphertexts takes up to 446 KB (i.e. less than a 400×400 resolution image), it is likely that the critical latency path is on the wireless telecom infrastructure as wireless network often provides lower bandwidth in the more rural parts of the highway system.

VII. Perspectives and extensions

As already emphasized in Sect. V-A, in real-world deployment scenarios, homomorphic encryption must always be embedded in a higher level protocol resorting to additional techniques for mitigating threats not covered by the baseline honest-but-curious model underlying FHE. In this section, we present some possible extensions.

A. Preventing Database Cloning

In this section, we provide a countermeasure to the threat of a client that tries to clone the platooning database. He sends a PIR request encoding many slots equal to 1. As such, he recovers different platoons' identifiers associated to different road segments.

We remind that for a certain time-step, a legitimate client will put 1 only in one slot of his PIR request. Indeed, a client is not supposed to be at two different road segments at the same time-step, except when multiple paths are possible but we omit this case as a matter of simplicity. As such, for a legitimate client, the sum of all the slots will always be equal to 1 and all but one slot will be encoding a zero.

First, the PSP receives an input ciphertext c_i and runs the PIR request as described in section V-B to get an encrypted answer c_i'. We propose to make the PSP compute an additional ciphertext c_s:

$$c_s = \mathsf{Sum}(\mathsf{MultNext}(\mathsf{Shuffle}(c_i))) \times rand$$

where,

- $\mathsf{Shuffle}()$ randomly shuffles the slots of a ciphertext. In practice, shuffling requires slots rotation which is an operation provided by homomorphic encryption schemes supporting batching.
- $\mathsf{MultNext}()$ multiplies each slot of the entry with the next one.
- $\mathsf{Sum}()$ computes and returns the sum S of all the slots. That is, it returns a ciphertext encoding a plaintext with a value of S in each slot.
- $rand$ is a random batched plaintext i.e., a plaintext encoding random values in each of its slots. Each random value is sampled in $[\![1, t-1]\!]$.

c_s is equal to 0 whenever a client performs a correct query namely c_i encodes 0 in all but one slot. If multiple slots encode non zero numbers, the chances that c_s outputs a random number in every slot increases with the amount of slots encoding a non zero number. Indeed, c_s outputs a random number in every slot if $\mathsf{MultNext}()$ outputs at least one non zero slot. This happens when $\mathsf{Shuffle}()$ outputs at least two consecutive non null slots.

c_s is then added to the response c_i' (i.e., $c_i' = c_i' + c_s$) before returning the result to the client. The client then gets (after removing the FHE layer) an unusable answer i.e., encrypted by a one-time mask it does not know. This technique adds a cost of a ciphertext/ciphertext multiplications which is doable as long as the client shares a public relinearization key to the PSP (although we may consider sending back non relinearized ciphertexts to avoid this, yet with a larger communication overhead). However, the $\mathsf{Shuffle}()$ procedure is more costly to perform when using a batched ciphertext (as it involves slot rotation operators).

B. Handling of query timings

In our protocol, the PSP has access to the estimated future locations of the platoons and the client sends requests to which the PSP answers with a set of (encrypted) potential platoons. The main information our protocol aims at preserving is the location and destination of the clients. With data being encrypted at all times, what the PSP (or an attacker with access to the database) has access to is the platoons database and the time when the client performed the request. The remaining threat is any inference attack associating the non encrypted platoons database with the encrypted query timing which should not reveal to the PSP which platoon has been followed by a given client. For example, if the PSP disposes of a database of a thousand platoons predicted locations and one client performs an encrypted query over this database to find a nearby platoon, if a few moments later the predicted locations from the thousand platoons do not change except for one precise platoon, we can assume that this platoon changed its path so that the client could join it. In this scenario, the PSP (or any attacker) could associate the non encrypted platoons database with the timing of the queries to obtain the platoon which the client has joined. To prevent this kind of inference attack between the platoons database and the timing of the queries, we can delay the updates on the platoons database such that a minimal amount of queries has to be done before that they update their upcoming locations. This amount of queries can be enforced for example by sending the update every 5 minutes (or any other relevant time scale), assuming that an attacker would then only have the information that some amount of clients asked for a platoon, and some amount of platoons changed their path within the 5 minutes. The attack therefore cannot target specific clients when the amount of clients becomes too high. Another countermeasure, of course, would be to encrypt the database homomorphically which prevents anyone from accessing it. However the update of the

database should be handled carefully and the simple multi key support discussed in V-D becomes jeopardized.

VIII. Conclusion

Privacy enhancing tools can unlock new possibilities for the intelligent transportation systems environment. In this paper, we have proposed a first protocol which allows for a client of a platooning solution to access a nearby platoon through a PSP, while ensuring the client location privacy. The client sends an encryption of its predicted future location to the PSP, which delivers, by means of encrypted-domain calculations, in return an encrypted set of adequate platoons. The performances of the protocol scale easily to practical levels: the homomorphic operations run in around 6 ms in total (with 2 ms on server side), and it naturally expands into the multi-client settings without the addition of any specific mechanisms. The protocol can be further improved by encrypting platoons locations, in exchange for the simplicity and the performances of the native multi-user protocol we have proposed.

References

[1] Y. An, S. Lee, S. Jung, H. Park, Y. Song, T. Ko *et al.*, "Privacy-oriented technique for covid-19 contact tracing (protect) using homomorphic encryption: Design and development study," *Journal of medical Internet research*, vol. 23, no. 7, p. e26371, 2021.

[2] P. Asuquo, H. Cruickshank, J. Morley, C. P. A. Ogah, A. Lei, W. Hathal, S. Bao, and Z. Sun, "Security and privacy in location-based services for vehicular and mobile communications: An overview, challenges, and countermeasures," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4778–4802, 2018.

[3] A. Boudguiga, O. Stan, A. Fazzat, H. Labiod, and P.-E. Clet, "Privacy preserving services for intelligent transportation systems with homomorphic encryption," in *International Conference on Information Systems Security and Privacy*, 2021.

[4] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886.

[5] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption Without Bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12, 2012, pp. 309–325.

[6] Z. Brakerski and V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages." in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 6841. Springer, 2011, pp. 505–524.

[7] M. Carpio, S. Robles, A. Sánchez, and C. Borrego, "Market equity for car-sharing applications using homomorphic encryption and opportunistic networking," in *2021 Wireless Days (WD)*. IEEE, 2021, pp. 1–5.

[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," Cryptology ePrint Archive, Report 2016/421, 2016, https://eprint.iacr.org/2016/421.

[9] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds," in *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*. Springer, 2016, pp. 3–33.

[10] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.

[11] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, no. 2009, 2009, pp. 169–178.

[12] C. Gentry and S. Halevi, "Compressible fhe with applications to pir," Cryptology ePrint Archive, Paper 2019/733, 2019, https://eprint.iacr.org/2019/733. [Online]. Available: https://eprint.iacr.org/2019/733

[13] S. Ghane, A. Jolfaei, L. Kulik, K. Ramamohanarao, and D. Puthal, "Preserving privacy in the internet of connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[14] D. Günther, A. Kiss, L. Scheidel, and T. Schneider, "Poster: Framework for semi-private function evaluation with application to secure insurance rate calculation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2541–2543.

[15] C. Höfer, J. Petit, R. Schmidt, and F. Kargl, "Popcorn: Privacy-preserving charging for emobility," in *Proceedings of the 2013 ACM Workshop on Security, Privacy & Dependability for Cyber Vehicles*, ser. CyCAR '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 37–48. [Online]. Available: https://doi.org/10.1145/2517968.2517971

[16] N. Kaaniche., A. Boudguiga., and G. Gonzalez-Granadillo., "Efficient hybrid model for intrusion detection systems," in *Proceedings of the 19th International Conference on Security and Cryptography - SECRYPT,*, INSTICC. SciTePress, 2022, pp. 694–700.

[17] F. Kargl, A. Friedman, and R. Boreli, "Differential privacy in intelligent transportation systems," in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. ACM, 2013, pp. 107–112.

[18] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *ICPS '05. Proceedings. International Conference on Pervasive Services, 2005.*, 2005, pp. 88–97.

[19] K. Laine, "Simple encrypted arithmetic library 2.3.1," *Microsoft Research https://www.microsoft.com/en-us/research/uploads/prod/2017/11/sealmanual-2-3-1.pdf*, 2017.

[20] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings." in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 6110. Springer, 2010, pp. 1–23.

[21] O. E. Omri, A. Boudguiga, M. Izabachene, and W. Klaudel, "Privacy-preserving k-means clustering: an application to driving style recognition," in *Network and System Security*, X. Liu, Joseph K.and Huang, Ed. Cham: Springer International Publishing, 2019, pp. 685–696.

[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[23] S. Petit, Jonathanand Dietzel and F. Kargl, *Privacy of Connected Vehicles*. Cham: Springer International Publishing, 2018, pp. 229–251. [Online]. Available: https://doi.org/10.1007/978-3-319-98161-1-9

[24] N. Rizzo, E. Sprissler, Y. Hong, and S. Goel, "Privacy preserving driving style recognition," in *Connected Vehicles and Expo (ICCVE), 2015 International Conference on*. IEEE, 2015, pp. 232–237.

[25] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, Jan. 2023, microsoft Research, Redmond, WA.

[26] N. Smart and F. Vercauteren, "Fully homomorphic simd operations," Cryptology ePrint Archive, Report 2011/133, 2011, https://eprint.iacr.org/2011/133.

[27] W. Wang, M. Min, L. Xiao, Y. Chen, and H. Dai, "Protecting semantic trajectory privacy for vanet with reinforcement learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–5.

[28] Y. Zhao and I. Wagner, "On the strength of privacy metrics for vehicular communication," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 390–403, 2019.