Privacy-preserving Vehicle Tracking Using Homomorphic Encryption

Hoang Anh Bui
HCLTech
Hanoi, Vietnam
email: anh.bui@hcl.com

Simy Chacko
HCLTech
Chenai, India
email: simy_c@hcl.com

Duc Cuong Nguyen
HCLTech
Hanoi, Vietnam
email: cuong_nguyen@hcl.com

***Abstract*—Transportation systems use location data to track and coordinate vehicle fleets. As locations pertain to sensitive information of users, privacy has become a key concern. Users can be tracked without their consent; location data could be (unintentionally) leaked, leading to costly consequences. Existing work on solving this problem often relies on using trusted servers for computation, or on masking, anonymization, obfuscation techniques on untrusted servers. Hence, either a trusted server is needed, or one must trade data utilization for privacy, unfortunately. Homomorphic encryption could be leveraged on location-based services to eliminate the need of using user's raw location while certain computations on encrypted data are still permitted. However, homomorphic encryption (1) only allows certain arithmetic (i.e., addition, multiplication) operations on encrypted data, and (2) incurs significant performance overhead. This impedes the application of homomorphic encryption in preserving the privacy of user in location-based services. This paper presents a novel approach to track a vehicle using homomorphic encryption. To this end, we propose a communication scheme that allows a vehicle to report its encrypted location to an untrusted cloud service. We design an efficient algorithm on homomorphically encrypted data to determine if vehicles are following predefined trips using advanced transformation techniques. Our evaluation shows that we can indeed flexibly employ homomorphic encryption in privacy-preserving location tracking with acceptable performance overhead. Our work only needs $\approx$ *6* milliseconds to compute results on encrypted location.**

***Index Terms*—Homomorphic encryption; privacy preserving; location tracking; trajectory monitoring**

I. Introduction

Location tracking allows companies to trace and coordinate the movements of their vehicle fleets. Further, location is heavily used for different purposes such as targeted advertising [6], recommendations [16], vehicle's trajectory monitoring [18], and many more [14]. Numerous incidents related to location tracking have been reported [17].

To tackle this issue, two main lines of work have been proposed: (1) using trusted service (with sufficient security and privacy protection) and (2) techniques to mask, obfuscate, or anonymize location data. On one hand, trusted services could become a single point of failure. Once become untrusted (e.g., for financial gains), they could potentially leverage user's locations for un-ethical usages (e.g., selling, leaking). On the other hand, data obfuscation techniques, most of the time, reduce data utilization. To potentially fill this gap, Homomorphic Encryption (HE) can be utilized to provide location-based service on encrypted data. It offers two crucial benefits: data privacy-preserving and data utilization. However, it remains challenging to employ HE in location-based services for several reasons. Homomorphic encryption only allows certain arithmetic operations on encrypted data such as multiplication and addition. This impedes the practicality of HE. Homomorphic encryption further incurs significant performance overhead, which is a constraint for many applications.

In this paper, we propose a privacy-preserving approach to track vehicles using HE. To overcome the constraints of applying HE, we design an effective algorithm to determine if a vehicle is following a pre-defined trajectory with only the allowed (on encrypted data) arithmetical operations. Accompanied with the algorithm is an efficient communication protocol that only requires few rounds of data exchanges. Our evaluation shows that it is possible to bring HE into privacy-preserving location processing with acceptable performance (and communication) overhead. Our proposed algorithm needs on average *6* milliseconds to check if an encrypted location is on a specific road-segment. We further discuss techniques to work with HE and the implication of our work.

Outline: This paper is organized as follows. We give an overview of related work in Section II. We describe our methodology for privacy-preserving vehicle tracking in Section III. Our approach is evaluated in Section IV. Section V discusses our findings. Section VI concludes our paper.

II. Related Work

One of the popular approaches to (partially) preserve location's privacy is reducing location's resolution. Particularly, cloaking techniques can be used to obfuscate user's location in a specific cloaked region [3]. *K*-anonymity was also proposed to obfuscate user locations [15]. The authors combined generalization (using less specific but semantically consistent locations) and suppression (completely removing the locations) techniques to achieve anonymity for user's locations. Dummy (fake) location data can also be provided together with actual (true) locations to confuse attackers [13]. However, all these data techniques degrade the utility of location data. In certain use-cases, the data might no longer be useful. Further, attackers can try to reconstruct the obfuscated data [4]

Multiple cryptography-based approaches were proposed to protect user's positions. Users could be notified if friends were in their vicinity using symmetric key to encrypt their locations [11]. For untrusted cloud services, Maris et al. introduced an architecture to distribute the management of user's location using secret sharing [10]. Recent work of Guldner et al. proposed homomorphic encryption to examine if a location is inside/outside a fencing area [5]. This however

 ISBN: 978-1-68558-092-6

used perpendicular projection of encrypted location to examine its position, which involves approximating functions for non-basic mathematical operations (e.g., *sine*, *cosine*).

To the best of our knowledge, none of these works use homomorphic encryption with advanced transformation techniques (on encrypted data) without linear approximation involved to provide privacy-preserving location tracking.

III. Vehicle Tracking Using Homomorphic Encryption

In this section, we provide details on how we perform vehicle tracking without learning plaintext locations. This includes details of our settings, our novel algorithm, and a few running examples on how our algorithm works.

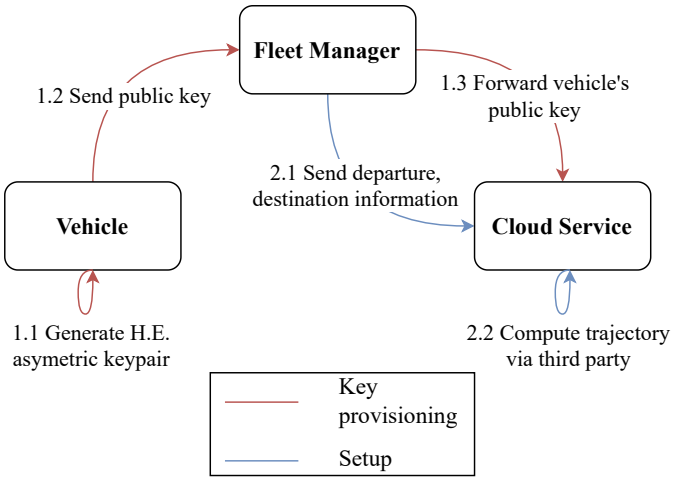

Figure 1. Setup phase.

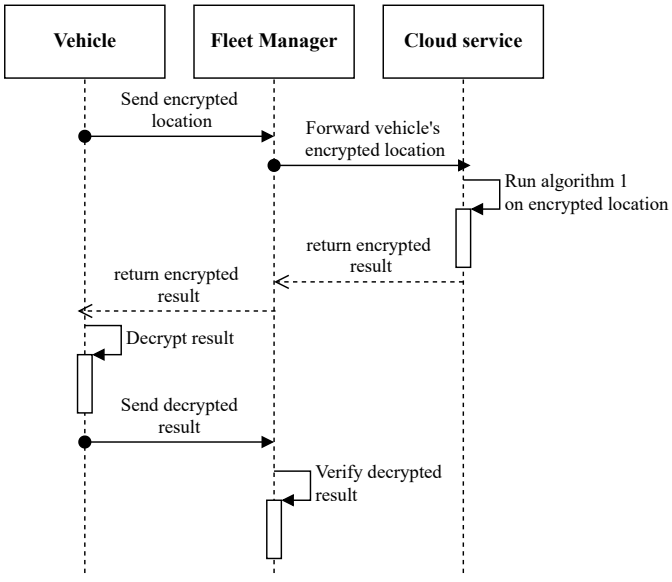

Figure 2. Communication flow.

In our settings, we assume the following parties:

- **Vehicle**: who needs to report its location periodically to a Fleet Manager. A requirement is no plaintext location can leave the vehicle.
- **Fleet Manager**: who needs to check if the aforementioned vehicle following a predefined trip. Fleet manager only needs to know whether Vehicle is following any possible route (of a predefined trip) or not. Fleet Manager is trusted, however it could be compromised.
- **Cloud Service**: who performs all computations on encrypted data to provide the answer of whether Vehicle is following any possible route (from A to B) or not. The cloud service is honest but curious, it may try to extract user's locations. In the worst scenario, cloud service could be compromised.

At the beginning of a trip, Fleet Manager provides the Vehicle a departure (A) and a destination (B) information to start the trip. Vehicle can take any route to get from A to B. Figure 1 describes the setup phase of our work. We propose this setting because it represents typical fleet management scenarios. Specifically, we want to outsource the computation to a cloud service while protecting user's privacy. In case the Cloud Service or the Fleet Manager gets compromised, it is impossible for them to learn vehicle's location as location is encrypted and only information of A and B is available in plaintext.

As we are dealing with homomorphic encryption, we are only allowed to use certain arithmetic operations on encrypted data (e.g., addition, multiplication, subtraction). However, prior algorithms to determine if a location is on a specific segment were all designed for plaintext locations. Besides, working with homomorphically encrypted data will incur significant performance overhead, we aim to tackle this by having minimal communication rounds among different communication parties. Hence, we cannot re-use existing algorithms.

Threat model: The adversary's goal here is to learn vehicle's location. It might try to collect vehicle's current and past location for different (unethical) purposes by attacking the Fleet Manager or the Cloud Service. In our setting, we assume the GPS location is always reported correctly (e.g., by using trusted hardware component in the vehicle [7]). We assume the Cloud Service and Fleet Manager in our setting is semi-honest (e.g., similar to prior works [9][12]). When Fleet Manager or Cloud Service is compromised, they are not able to extract vehicle's location as location is always encrypted and only the Vehicle holds the decryption key.

Homomorphic encryption scheme: As we work with location's coordinates in the form of longitude and latitude (e.g., float numbers), we use the *CKKS* scheme as this scheme can work with float data type.

Key provisioning: Our work uses the public-key scheme with four operations: (1) *KeyGen* produces a public key (*pk*) and a secrete key (*sk*); (2) *Encrypt* uses *pk* to encrypt a plaintext *m* and produces a cipher-text *c*; (3) *Decrypt* uses *sk* to decrypt the above cipher-text *c*; (4) *Evaluate* uses *pk* to perform computation (e.g., addition and multiplication) on a set of cipher-text $C = c_1, c_2, ..., c_n$

Vehicle is responsible for setting up the encryption text with encryption (private) key and computation (public) key. The encryption key never leaves the Vehicle while the computation key is shared with the Cloud Service for running computation on encrypted data.

Algorithm 1: Algorithm to determine if an encrypted location is on a predefined trip

```
1  Function shift_coordinate():
       Input  : coordinate, base
       Output: shifted_coordinate
2      return [coordinate.x - base.x, coordinate.y -
        base.y]
3  Function change_basis():
       Input  : coordinate, vector
       Output: changed_coordinate
       /* Perform coordinates
          transformation (e.g.,
          cooridnates rotation with an
          angle called α formed by x-axis
          and the segment P). See Figure 4
          for examples.                  */
4  Function generate_noise()():
       Input  :
       Output: random_list
       /* Generate a list of random
          length. Each element is a pair
          of 2 random numbers.           */
5  Function main():
       Input  : routes, loc, security_number
       Output: ret
6      ret = []
7      for segment in routes do
8          segment.start = call: shift_coordinate
            (segment.start, segment.start)
9          segment.end = call: shift_coordinate
            (segment.end, segment.start)
10         loc = call: shift_coordinate (loc,
            segment.start)
11         vector = (segment.end.x - segment.start.x,
            segment.end.y - segment.start.y)
12         segment.end = call: change_basis
            (segment.end, vector)
13         loc = call: change_basis (loc, vector)
14         lat_del = (loc.x - segment.start.x)*(loc.x -
            segment.end.x)
15         ret.append([lat_del, loc.y])
16     ret.append(call: generate_noise())
17     ret = call: shuffle(ret)
18     return ret
```

A. Trip Tracking

The workflow of our approach is described in Figure 2. In the setup phase, the Fleet Manager shares a predefined trip — that the vehicle needs to follow — to the Cloud Service for later computation. In the running phase, the Vehicle first encrypts its current location using its encryption key and shares the encrypted location with the Fleet Manager. Upon receiving the encrypted location, Fleet Manager forwards it to the Cloud Service. Cloud Service runs algorithm 1 on the encrypted location to produce encrypted results that indicate whether a given location is on any road segment (of the predefined trip). Cloud Service further adds noise and shuffles the results of algorithm 1 before sending them back to the Fleet Manager. This makes it harder for a compromised Fleet Manager to learn about the current route segment and route that the Vehicle is following. Fleet Manager now receives the encrypted results, it will forward to the Vehicle to ask for decryption. Vehicle, upon receiving the encrypted results, decrypts them and gets the results in plain text. Vehicle then forwards the plaintext results back to the Fleet Manager. Fleet Manager first checks in the results if any number is negative. If there is a negative number, Vehicle is on a segment of a possible route. Subsequently, if a tolerance distance is defined, Fleet Manager then checks the value of the second parameters (associated with the negative number) to see if it is less than the predefined tolerance. If that is the case, the Vehicle is following the predefined trip.

B. Algorithm 1

Pseudo code in Algorithm 1 illustrates how we perform computation on encrypted data to provide results that indicate whether a location is on a possible route of a predefined trip. In Algorithm 1, only multiplication, addition, and subtraction are used on vehicle's location as only these arithmetic operations are allowed on homomorphically encrypted data. We take as inputs (1) a set of possible routes (representing the predefined trip) that the vehicle can follow. This set of routes is represented by an array of route segments.; and (2) vehicle's current location that has been homomorphically encrypted. For each *segment* in all possible routes, Algorithm 1 first performs coordinates shifting (line 8, 9, 10) and basis change (line 12, 13) to avoid complex operations on encrypted location. The returned results (variable *ret*) are an array consisting of encrypted obfuscated distances. Once, decrypted (by the Vehicle) the Fleet Manager checks if (1) the Vehicle is on any segment of any possible route (i.e., if any number in the array is negative); and (2) the corresponding distance of the Vehicle to the segment is within the predefined tolerance.

Running example: Figure 3 shows the original coordinates in gray of the vehicle and the current *segment* in Algorithm 1 that we are examining. If we want to calculate the distance of the vehicle's location to *segment*, we must use some forms of advanced mathematical operations (e.g., *sine*, *cosine*). This is however not possible on homomorphically encrypted data. We therefore need to transform the current coordinates so that we can calculate the distance using only subtraction, multiplication, and addition. We shift all coordinates such that start location of the segment ($segment_{start}$)

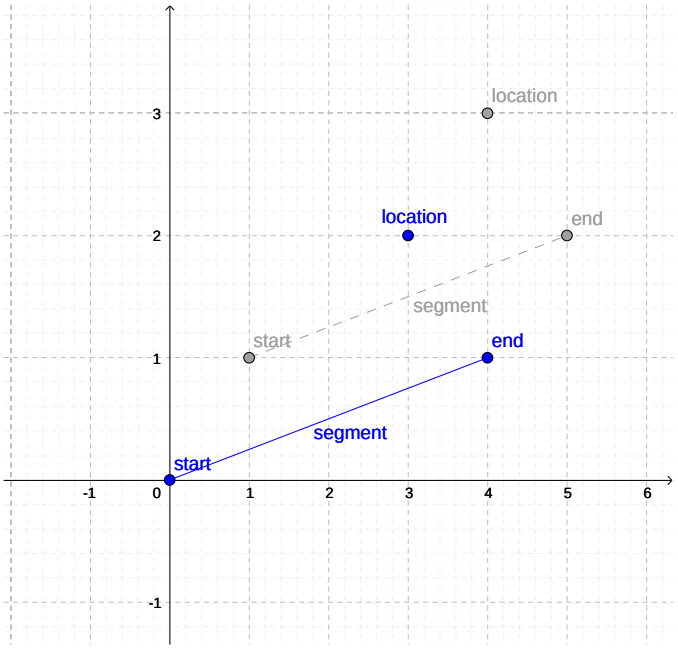

Figure 3. Coordinates shifting. New coordinates are marked blue.

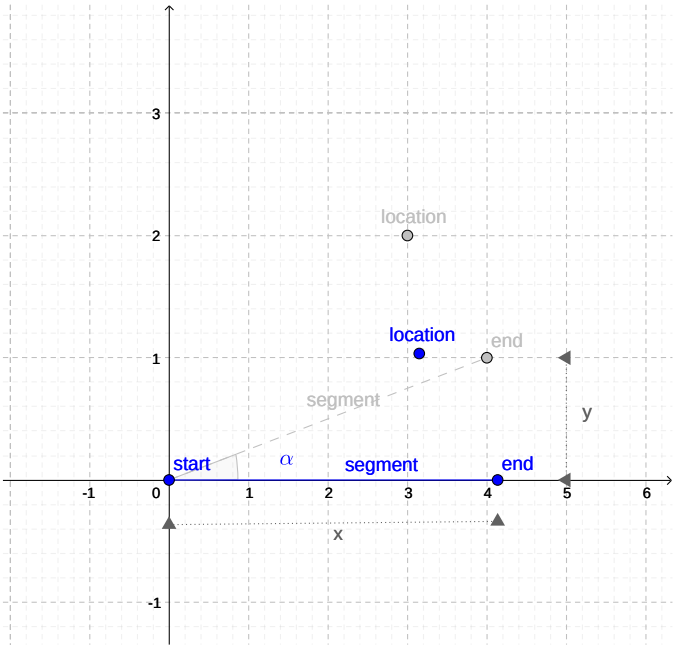

Figure 4. Coordinates with basis changed with α angle . New coordinates are marked blue.

becomes the origin (`0,0`). Shifted coordinates are marked blue. At this point, the distance calculation related to vehicle's location$_{x,y}$ still involves complex mathematical operations (that are not allowed on homomorphically encrypted data). We then aim to rotate all coordinates such that the distance can be calculated only using allowed operations on encrypted locations. We set to rotate all coordinates so that the current segment lies on the x-axis. This gives us the distance of the current location to the segment, which is exactly the value of location$_y$ without further complex distance calculation. To this end, we apply the following equation to transform/rotate the all coordinates (i.e., change of basis [2])

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}^{-1} \times \begin{bmatrix} x_{old} \\ y_{old} \end{bmatrix}$$

The coordinates of vehicle's location$_{x,y}$ and segment *S* after coordinates' basis have been changed is shown in Figure 4. The functions *sin(α)* and *cos(α)* operated on plaintext data only (e.g., `x` and `y` are derived from segment *S* and the x-axis). Finally, we can see that, the closest distance between vehicle's location$_{x,y}$ and *S* is exactly the value of location$_y$.

Line 14 in Algorithm 1 performs calculation to determine if location$_{x,y}$ is with-in the segment *S*. The variable `lat_del` now indicates (after being decrypted) if a location is with-in the segment *S* (i.e., having a negative value) or not (i.e., having a positive value). The value of location$_y$ indicates the distance of the vehicle to the segment *S*. If the distance is less than (or equal to) a predefined tolerance, the Fleet Manager can consider that the Vehicle is still following the predefined trip. Otherwise, the Vehicle is not following. Line 15 adds to the results the distance from location$_{x,y}$ to segment *S*. Having the distance from an (encrypted) location to a *segment* allows us to later check if the distance is smaller than a predefined threshold (e.g., distance tolerance). If it is, the location is considered on the segment. This means, the vehicle is still following a predefined trip.

Finally, line 16 creates a random set of pairs to the returned results. This is to resemble (fake) variables `loc.`y and `lat_del`, which represent *dummy* route segments. Hence, given a fake pair (`loc.`y and `lat_del`), both elements must be greater than 0 (i.e., location is not on the current dummy route segment and its distance is a positive number). Subsequently, line 17 shuffles the results. This makes it harder for Fleet Manager (e.g., in case it is compromised) to associate the returned distance (of a location to a segment) with, which segments. Particularly, Fleet Manager knows the distance of vehicle's location to a segment but not the exact segment. For plaintext values (e.g., *segment.start*, *segment.end*), certain *sine*, *cosine* functions are used (*sine/cosine* of the angle of segment *S* and the x-axis), yet only on the coordinates of *segment.start* and *segment.end*. This is possible as predefined trip (and its possible routes) is not encrypted.

IV. Evaluation

To determine the effectiveness of our solution, we measure its computation and communication overhead.

A. Setup

Our setting environment is based on a Virtual Machine with 1 CPU 2.10 GHz and 4GB RAM, running Ubuntu 18. For our evaluation, `TenSeal` [1] is picked, as it enables quick implementation. We only care about *CKKS* scheme, since our data (location's coordinates) is represented by float numbers. Fleet Manager defines a trip that can be reached via 3 possible

 ISBN: 978-1-68558-092-6

routes. Each possible route contains 10 segments. We choose number 10 to represent a typical (inner-city) route. This means Algorithm 1 must iterate over 30 route segments to track if the Vehicle is following the predefined route.

B. Results

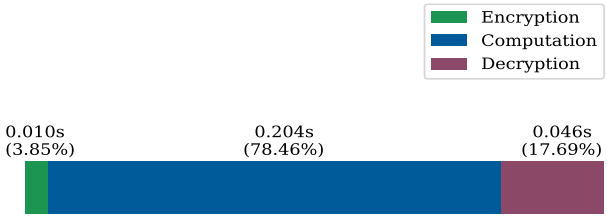

Figure 5. Execution time attribution.

We consider three phases for evaluation: encryption, computation, and decryption. In this phase, we do not consider the time spent on exchanging data among parties. Computation here refers to the execution of Algorithm 1 on the encrypted location leaves the Vehicle. We perform this evaluation 30 times on different locations to measure the execution time.

As illustrated in Figure 5, encrypting vehicle's location takes 3.85% (0.01 seconds) of the execution time. Besides, computation (running Algorithm 1 on encrypted location to return the encrypted result) takes the majority (78.46%) of the computation time with $\approx$ 0.204 seconds. Decryption on the other hand takes 17.69% the execution time with 0.046s.

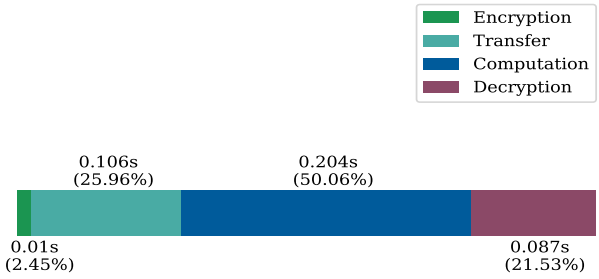

Figure 6. Total execution time attribution.

In the second evaluation, we further consider the communication time among three parties (Fleet Manager, Vehicle, and Cloud Service). As Figure 6 shows, average transfer time (among all parties) — from when encrypted location is shared by Vehicle until the decrypted results arrive at Fleet Manager — is 0.106 seconds. This means around one-forth of the total execution time is spent on exchanging data. setting. We conducted this experiment in a local network. This means the latency is negligible. Overall, the time taken for Fleet Manager to determine if a Vehicle is following a predefined trajectory is 0.407 seconds.

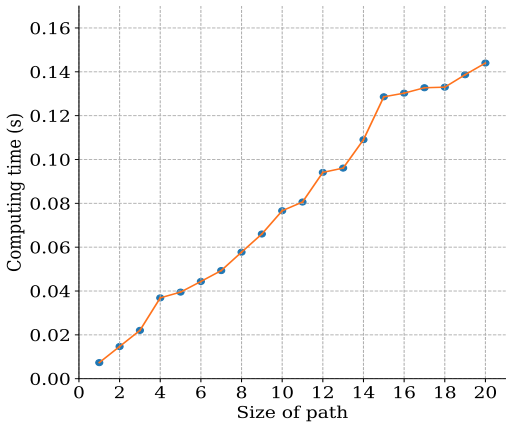

Figure 7. Correlation between execution time and N.O segments in a trajectory.

Figure 7 shows the correlation between the number of segments (segments' size) and the execution time (e.g., running Algorithm 1). We can see that the time needed to run Algorithm 1 increases linearly as the size of segments (of a predefined trajectory) increases. On average, Algorithm 1 takes $\approx$ *6* milliseconds to compute results on a single segment.

Comparison to plaintext execution: To examine the computation overhead of our approach, we compare the execution on encrypted data against plaintext data. We run our comparison 50 times on different locations to examine the difference. On plaintext location, running Algorithm 1 takes 0.79 milliseconds while on homomorphically encrypted location it takes 60.9 milliseconds. This means, computation on encrypted location, as anticipated, takes significantly more time (77x execution time).

Accuracy: The accuracy of our approach depends on two factors: The approximation of the CKKS scheme and the distance tolerance set by Fleet Manager. As we use the *CKKS* implementation of `TenSeal`, we directly benefit from its precision. In our evaluation, the calculated distance has a deviation of around $\approx$ 1e-8 depending on the unit of the provided coordinates (e.g., meters, kilometers).

V. Discussion

In this paper, we propose novel techniques to work with homomorphically encrypted data to provide vehicle monitoring service that preserve location's privacy, especially when the data processing entities (i.e., Fleet Manager or Cloud Service) are compromised. By proposing novel coordinate's transformation techniques, our work only uses arithmetic operations that are allowed on encrypted data — hence, not requiring any approximation functions, or multiple intermediates collaboration steps from the encryption key holders. Our approach prevents a compromised Fleet Manager or an untrusted Cloud Service from learning vehicle's location while still provides accurate and necessary tracking information.

 ISBN: 978-1-68558-092-6

A. Limitation and Future Work

With Algorithm 1, Cloud Service needs to return a list of (encrypted) numbers. This is not optimal as future work could study a more efficient algorithm that allows checking just one number to learn vehicle's compliance.

Performance overhead: In our evaluation, we see a computation overhead of 77x when running Algorithm 1 on encrypted locations vs. on plaintext locations. However, we are aware that for encrypted locations, there exist other overhead such as encryption, decryption, transfer too. This will result in higher overhead overall. Hence, such a comparison only gives a rough overview of the minimal overhead of running Algorithm 1 on encrypted locations. Further, we did not consider the data sizes (to be transferred) in our experiments to judge the size overhead when applying HE. Future work could consider this aspect to provide a more comprehensive look into applying HE in location tracking.

Finally, our work relies on the security of HE schemes in-use. If adversaries can break such schemes, location data could be revealed. Specifically, CKKS scheme was shown to be vulnerable to secret key recovery under the assumption that attackers can (1) access to encryption oracle; (2) choose the function to be evaluated homomorphically; and (3) access to decryption oracle [8]. This attack exploits the linearity of the decryption function in CKKS and the hints that approximated decrypted results provide.

B. The use of nonlinear functions

One potential direction to use nonlinear functions is to identify their corresponding approximated (linear) alternatives so that they could be applied on HE data. There are several disadvantages to this approach. Approximated functions are usually complex and could result in extreme performance overhead. With HE data, the more operations are done on it, the more noise will be accumulated. Accumulated noise could lead to the case that encrypted data cannot be decrypted any more. While bootstrapping techniques generally could remove the noise, yet it again adds more performance overhead.

C. Application Analysis

While this work focuses on vehicle tracking, the applications of HE in location-based service is enormous. We can use HE to monitor if a person is inside a fencing area without having to collect their exact (raw) location. This protects user's privacy even in the event of data leakage (both on purpose and unintentionally). Points of interests can also be recommended to users using HE Specifically, so users can search for nearby coffee shops, restaurants, hospitals, etc. without having to provide raw location. Further, the Cloud Service, could store HE locations for future analysis. Analysis results is only be revealed with the consent of the vehicle (i.e., its owner) as the results can only be decrypted by the Vehicle.

VI. Conclusion

This paper presents a novel privacy-preserving approach to track real-time vehicle's compliance to predefined trips. Our work leverages state of the art homomorphic encryption to protect location's privacy while still allowing computations on encrypted data to determine if a vehicle is following predefined trips. We evaluated the effectiveness of our approach and showed that homomorphic encryption can be efficiently used to protect location privacy.

References

[1] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," unpublished, 2021.

[2] J. G. Broida and S. G. Williamson, *A comprehensive introduction to linear algebra*. Addison-Wesley Boston, MA, USA:, 1989, vol. 4.

[3] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. IEEE, 2005, pp. 620–629.

[4] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 31–42. [Online]. Available: https://doi.org/10.1145/1066116.1189037

[5] M. Guldner, T. Spieldenner, and R. Schubotz, "Nexus: Using geofencing services without revealing your location," in *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 2018, pp. 1–6.

[6] H. Haddadi, P. Hui, T. Henderson, and I. Brown, *Targeted Advertising on the Handset: Privacy and Security Challenges*. London: Springer London, 2011, pp. 119–137. [Online]. Available: https://doi.org/10.1007/978-0-85729-352-7_6

[7] S. Hu, Q. A. Chen, J. Joung, C. Carlak, Y. Feng, Z. M. Mao, and H. X. Liu, "Cvshield: Guarding sensor data in connected vehicle with trusted execution environment," in *Proceedings of the Second ACM Workshop on Automotive and Aerial Vehicle Security*, ser. AutoSec '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–4. [Online]. Available: https://doi.org/10.1145/3375706.3380552

[8] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Advances in Cryptology – EUROCRYPT 2021: 40th Annual Internation al Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 648–677. [Online]. Available: https://doi.org/10.1007/978-3-030-77870-5_23

[9] Y. Li and W. Xu, "Privpy: General and scalable privacy-preserving data mining," ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1299–1307. [Online]. Available: https://doi.org/10.1145/3292500.3330920

[10] G. Marias, C. Delakouridis, L. Kazatzopoulos, and P. Georgiadis, "Location privacy through secret sharing techniques," in *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*. IEEE, 2005, pp. 614–620.

[11] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia, "Privacy in geo-social networks: Proximity notification with untrusted service providers and curious buddies," *The VLDB Journal*, vol. 20, no. 4, p. 541–566, aug 2011. [Online]. Available: https://doi.org/10.1007/s00778-010-0213-7

[12] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[13] T. Peng, Q. Liu, D. Meng, and G. Wang, "Collaborative trajectory privacy preserving scheme in location-based services," *Inf. Sci.*, vol. 387, no. C, p. 165–179, may 2017. [Online]. Available: https://doi.org/10.1016/j.ins.2016.08.010

[14] B. Rao and L. Minakakis, "Evolution of mobile location-based services," *Commun. ACM*, vol. 46, no. 12, p. 61–65, dec 2003. [Online]. Available: https://doi.org/10.1145/953460.953490

[15] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, p. 571–588, oct 2002. [Online]. Available: https://doi.org/10.1142/S021848850200165X

 ISBN: 978-1-68558-092-6

[16] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, and S. Wang, "Learning graph-based poi embedding for location-based recommendation," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 15–24. [Online]. Available: https://doi.org/10.1145/2983323.2983711

[17] H. Yun, D. Han, and C. C. Lee, "Understanding the use of location-based service applications: do privacy concerns matter?" *Journal of Electronic Commerce Research*, vol. 14, no. 3, pp. 220–230, 01 2013.

[18] Z. Zhou, W. Dou, G. Jia, C. Hu, X. Xu, X. Wu, and J. Pan, "A method for real-time trajectory monitoring to improve taxi service using gps big data," *Inf. Manage.*, vol. 53, no. 8, p. 964–977, dec 2016. [Online]. Available: https://doi.org/10.1016/j.im.2016.04.004

 ISBN: 978-1-68558-092-6