Privacy-Preserving k-means Clustering: an Application to Driving Style Recognition

Othmane El Omri[1], Aymen Boudguiga[1,2(✉)], Malika Izabachene[2], and Witold Klaudel[1,3]

[1] IRT SystemX, 91120 Palaiseau, France
{othmane.omri,aymen.boudguiga,witold.klaudel}@irt-systemx.fr
[2] CEA-LIST, 91191 Gif-sur-Yvette, France
{aymen.boudguiga,malika.izabachene}@cea.fr
[3] Renault, 1 Avenue du Golf, 78288 Guyancourt, France
witold.klaudel@renault.com

Abstract. With the advent of connected vehicles, drivers will communicate personal information describing their driving style to their vehicles manufacturers, stakeholders or insurers. These information will serve to evaluate remotely vehicle state via an e-diagnostics service, to provide over-the-air update of vehicles controllers and to offer new third parties services targeting profiled drivers. An inherent problem to all the previous services is privacy. Indeed, the providers of these services will need access to sensitive data in order to propose in return an adequate service.

In this paper, we propose a privacy-preserving k-means clustering for drivers subscribed to the *pay how you drive* service, where vehicles insurance fees are adjusted according to driving behavior. Our proposal relies on secure multi-party computation and additive homomorphic encryption schemes to ensure the confidentiality of drivers data during clustering and classification.

Keywords: Pay how you drive · Privacy · Mutli-party computation

1 Introduction

When drivers of connected vehicles subscribe to a *pay how you drive* service (PHYD), their vehicles communicate information about their position, speed, acceleration and braking frequencies to the insurer. The insurer uses this information to classify drivers and to adjust their insurance fees with respect to their driving behaviors.

The problem with PHYD is that the collected data can be used by a malicious adversary to deduce information about drivers such as their home address or workplace, their travelling habits, their speed infractions, etc. Consequently, it is compulsory to provide a privacy-preserving drivers clustering and classification algorithm.

J. K. Liu and X. Huang (Eds.): NSS 2019, LNCS 11928, pp. 685–696, 2019.
https://doi.org/10.1007/978-3-030-36938-5_43

Contribution–In this work, we propose to rely on an unsupervised machine learning algorithm for drivers' clustering as insurers keep confidential their clustering algorithms due to IP and business reasons. Among the unsupervised learning techniques, we choose, the simple *k-means* algorithm. We consider three types of drivers (i.e. $k = 3$): aggressive, normal, and cautious; as proposed by the driving simulator SCANeR Studio[1].

We use secure multi-party computation techniques and an additive homomorphic encryption algorithm to make k-means ensure the privacy of drivers' data. That is, drivers' features are not exposed neither during k-means model training nor during classification[2]. We make use of Yao garbled circuits protocol [1] for Squared Euclidien Distances computation and Paillier [2] cryptosystem for means computation.

Paper Organization–In Sect. 2, we present the state of the art regarding the private processing of vehicles' data. In Sect. 3, we present the background concepts used in these paper such as the k-means algorithm, the secure multiparty computation and the homomorphic encryption. In Sect. 4, we specify our privacy-preserving extension to k-means when applied to drivers clustering. In Sect. 5, we discuss our proposed protocol security and performance. Finally, Sect. 6 concludes the paper and provides future research directions.

2 State of the Art on Private Processing of Vehicles Data

In 2011, Troncoso et al. [3] proposed to install a secure hardware, i.e. a black box, in vehicles to compute the insurance fees locally. The obtained fees are transmitted later to insurances for billing. As such, vehicles' private data are kept secret from insurances. They also specified an auditing mechanism to check that neither the insurance nor the owner of the vehicle cheated of fees. Indeed, they store the data needed for calculating insurance costs on an auxiliary storage. The data are encrypted using a split key between the vehicle owner and the insurance. In case of a dispute, the vehicle owner and the insurance combine their split key to decrypt the auxiliary storage and check how the insurance fee has been computed.

In 2013, Kargl et al. [4] used differential privacy techniques to protect Floating Car Data (FCD). Differential privacy provides mathematical privacy guarantees. However, it allows only to make a limited number of queries, such as computing the sum, the minimum/maximum and the average. It is not well fitted for private k-means calculus.

In 2015, Rizzo et al. [5] proposed a technique to train 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 using the homomorphic properties of Paillier cryptosystem [2].

[1] http://www.oktal.fr/en/automotive/range-of-simulators/range-of-simulators.

[2] Note that our protocol is not only limited to drivers clustering and can be easily generalized to cover all use-cases using k-means for clustering.

3 Background

In this section, we review the key concepts used in this paper.

3.1 *k*-means Clustering Algorithm

k-means algorithm [6] produces automatically k clusters ($\{c_1, \ldots, c_k\}$) from a collection of data sets ($\{d_1, \ldots, d_n\}$) in a simple way. k-means relies on distance and mean computation for data clustering as presented in Algorithm 1. First, we select k random cluster centers ($\{\mu_1, \mu_2, \ldots, \mu_k\}$). Then, we iterate the algorithm until converging to the best choice of clusters' centers or reaching a preselected number of iterations.

input : n data vectors and the number of clusters k
output: k clusters

1 Select k cluster centers $\{\mu_1, \mu_2, ..., \mu_k\}$;
2 **repeat**
3 Assign each data vector $d_{j,j\in[\![1,n]\!]}$ to the closest cluster $c_{i,i\in[\![1,k]\!]}$ whom center is $\mu_{i,i\in[\![1,k]\!]}$ (i.e., the distance between μ_i and d_j is minimum);
4 Replace each cluster center μ_i by the mean of elements d_j belonging to the cluster c_i;
5 **until** *cluster centers do not vary significantly or the number of iterations is reached*;

Algorithm 1. k-means clustering

3.2 Yao Garbled Circuit

Yao's garbled circuit [1] allows two parties to evaluate a boolean circuit C without revealing their respective inputs. The circuit *generator* creates a garbled circuit GC by obfuscating the inputs to C wires. Indeed, for each wire w_i of the circuit, the *generator* chooses randomly two secret values w_i^0 and w_i^1. w_i^0 and w_i^1 are the garbled values corresponding to 0 and 1, respectively. Finally, she creates a garbled table GT_i for each gate G_i. Each line of GT_i contains two garbled inputs $w_{i,1}^j$ and $w_{i,2}^j$ (where $j \in \{0,1\}$) and their corresponding output $E(H(w_{i,1}^j, w_{i,2}^j), w_{i,o}^j)$ where E is an encryption algorithm and H a key generation function. After decryption, GT_i allows to get the garbled value of the output $w_{i,o}^j$.

The circuit generator transmits GC, i.e. all the garbled tables, to the circuit *evaluator*. In addition, the generator provides the evaluator with her garbled inputs $w_{i,g}^j$ for all input gates $G_i \in GC$. Finally, the evaluator recovers her own garbled inputs $w_{i,e}^j$ to the same input gates $GT_i \in GC$ using oblivious transfers [7,8]. At this point, the evaluator can compute all the garbled gates of GC. She starts with the decryption of the *input* gates with the keys $E(H(w_{i,g}^j, w_{i,e}^j), w_{i,o}^j)$, and continues until reaching the final output of the boolean circuit C.

3.3 Additive Homomorphic Public Key Encryption

A public key encryption (PKE) consists of 3 algorithms: KG which given a security parameter λ generates a public key pk and a secret key sk. One produces an encryption of a message m using pk and a randomness. We denote $E_{pk}(m)$ the obtained ciphertext. To decrypt a ciphertext c, one uses the secret key sk and outputs either a message or $\perp$. We denote $D_{sk}(c)$ this output.

We say that a PKE scheme is correct if for $(pk, sk) \leftarrow KG$ and $c \leftarrow E_pk(m)$, we have that $D_{sk}(c) = m$ holds with high probability. The PKE encryption function is assumed to be additively homomorphic i.e. $E_{pk}(m_1 \oplus m_2) = E_{pk}(m_1) \circledast E_{pk}(m_2)$, where $\oplus$ and $\circledast$ are the group laws over the message space and the ciphertext space, respectively. Example of such cryptosystems is Paillier scheme [2].

4 Privacy-Preserving *k*-means Clustering of Drivers

In this section, we detail our proposed protocol for running k-means while keeping drivers' features private. We denote by N the total number of vehicles (or drivers), V_i is the driver identifier for $i \in [\![1, N]\!]$ and $X_i = (x_{i1}, \ldots, x_{im})$ is the vector of features of V_i. k is a fixed integer and denotes the number of clusters ($k = 3$). S is the model provider. In our application, S refers to the insurer server. $C(V_i)$ is the label of the cluster containing X_i.

4.1 Assumptions

Clustering. During clustering, each driver inputs his driving data to the k-means algorithm. These driving data are private and must not be shared or analyzed in plaintext. The k-means algorithm returns to a driver the *index* of the cluster to which he belongs. Meanwhile, the insurer receives the *centers* of all created clusters. The centers of clusters are insurer's private data, and must not be shared with drivers.

Classification. During classification, we assume that each driver inputs his data to the k-means algorithm. In return, the insurer and the driver only obtain the index of the cluster to which the driver belongs.

Communication Model. We assume no communications between vehicles. We only consider direct communications between a vehicle and the insurer. We do not have real-time constraints as driver insurance fees are paid once per month. So, the upcoming computation can be done in background by an insurance application installed in one of the trusted electronic control units of the vehicle.

input : n data entities and the number of clusters k
output: k clusters

1 S randomly selects k cluster centers $\{\mu_1, \mu_2, ..., \mu_k\}$;
2 **repeat**
3 **for** $i = 1..N$ **do**
4 V_i and S engage in a secure two-party cluster attribution protocol;
5 V_i gets $C(V_i)$, S gets no information;
6 $S, V_1, \ldots, V_n$ engage in a secure multi-party mean computation protocol to update cluster centers;
7 **end**
8 **until** *cluster centers do not vary significantly or the number of iterations is reached*;

Algorithm 2. Privacy-preserving k-means clustering

Attacker Model. In this work, we consider a honest-but-curious model. Each of the N players (e.g. vehicles) possesses private features. The model provider S (e.g. the insurer) has no access to these features. One player V_1 is chosen as the dealer player in the honest but curious model; V_1 is the only one who knows the private key sk_1. Each player sends his encrypted features under public key pk_1 to S. We assume that S and V_1 act as honest players. Also, we assume that no collusion between players is possible.

4.2 Proposed Protocol for Privacy-Preserving *k*-means

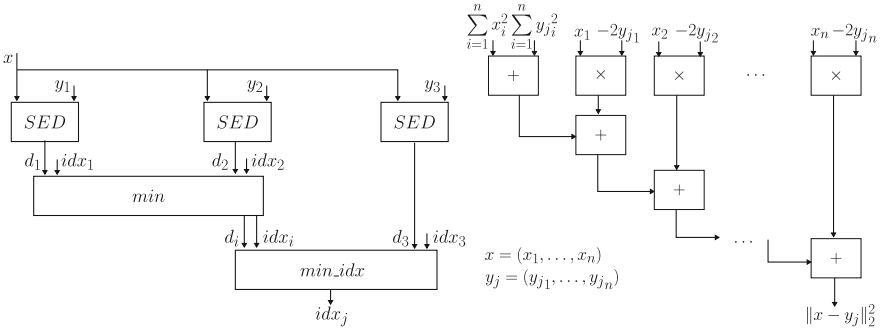
We enhance k-means clustering with secure multi-party computation (as presented in Algorithm 2). First, we propose a *secure two-party protocol for the closest cluster computation* (i.e. for drivers attribution to clusters). Then, we define a protocol for *secure multi-party mean computation*. The computed mean serves to privately update the centers of clusters.

Secure Computation of the Closest Cluster. We use a secure two-party protocol between a vehicle (V_i) and the insurer (S) to compute the closest cluster to V_i. To meet our requirement of keeping the centers of clusters private, we use an unfair version of Yao's protocol, where only the driver (V_i) obtains the result of the computation, the insurer gets no information.

For our protocol, we propose the circuit of Fig. 1a. If the insurer is the *evaluator* of the circuit, she sends the output labels to the vehicle. If the insurer is the *generator* of the circuit, she sends the table mapping each label to its value. In our circuit, we compute the Squared Euclidean Distance (`SED`) and then returns the index of the lowest distance, i.e. the closest cluster. `SED` is computed with respect to the 3 current cluster centers.

First, we use the circuit of Fig. 1b to compute the Squared Euclidean Distance (`SED`) between two n-dimensional vectors $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_m)$. It computes $\|X - Y\|_2^2 = \sum_{i=1}^{m}(x_i - y_i)^2 = \sum_{i=1}^{m} x_i^2 - 2\sum_{i=1}^{m} x_i.y_i + \sum_{i=1}^{m} y_i^2$.

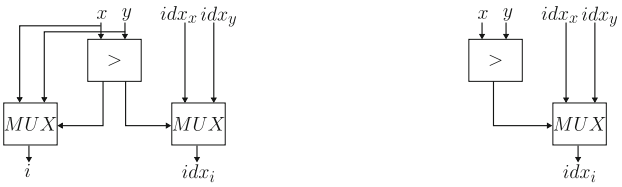
The second circuit computes the minimal distance to a cluster center. We use the `min` circuit described in [9] to compare 2 distances d_1 and d_2. The `min` circuit returns the smallest distance between d_1 and d_2 with its respective index



(a) Global circuit (b) `SED` circuit

Fig. 1. Circuit for closest cluster index calculus

(i.e. 1 or 2). The `min` circuit is composed of a comparison ($>$) gate and two multiplexer gates (`MUX`). The comparison gate takes two inputs x, y and outputs 1 if $x > y$ and 0 otherwise. The `MUX` gate takes 3 inputs x, y and a bit b. If $b = 1$, `MUX` outputs x, otherwise it outputs y. Finally, we remove one `MUX` gate from the `min` circuit to get the `min_idx` circuit (Fig. 2b). The latter outputs the *index* of the minimum distance and *not* the value of the minimum.



(a) `min` (b) `min_idx`

Fig. 2. `min` and `min_idx` circuits

Secure Computation of Clusters' New Centers. The second k-means computation updates the centers of clusters. The new centers are simply the *mean* of driving data that belong to drivers from the same cluster. Our protocol requires a semantic secure and homomorphic additive cryptosystem. Semantic security ensures that encrypting twice the same driving data returns two different ciphertexts. Meanwhile, the homomorphic addition sums encrypted data of drivers belonging to the same cluster. Example of such cryptosystems is Paillier scheme [2].

Our protocol contains 3 phases: an initialization, a secure sum computation and a secure mean computation. First, the insurer S nominates a vehicle as V_1 to initiate the protocol as depicted in Algorithm 3. We assume w.l.o.g that V_1 belongs to the first cluster (i.e. $C(V_1) = 1$). V_1 picks 6 random values

S nominates a vehicle as V_1;
V_1 picks 6 random values $r_{f1}^1, r_{f2}^1, r_{f3}^1, r_{n1}^1, r_{n2}^1, r_{n3}^1$;
V_1 computes:
$c_{f1}^1 = E_{pk_1}(x_1 + r_{f1}^1)$ $\quad$ $c_{n1}^1 = E_{pk_1}(1 + r_{n1}^1)$;
$c_{f2}^1 = E_{pk_1}(r_{f2}^1)$ $\quad$ $c_{n2}^1 = E_{pk_1}(r_{n2}^1)$;
$c_{f3}^1 = E_{pk_1}(r_{f3}^1)$ $\quad$ $c_{n3}^1 = E_{pk_1}(r_{n3}^1)$;
V_1 sends to S: $c_{f1}^1, c_{f2}^1, c_{f3}^1, c_{n1}^1, c_{n2}^1, c_{n3}^1$;
S initializes:
$S_{f1} \leftarrow c_{f1}^1$ $\quad$ $S_{n1} \leftarrow c_{n1}^1$;
$S_{f2} \leftarrow c_{f2}^1$ $\quad$ $S_{n2} \leftarrow c_{n2}^1$;
$S_{f3} \leftarrow c_{f3}^1$ $\quad$ $S_{n3} \leftarrow c_{n3}^1$;

Algorithm 3. Secure multi-party mean computation protocol: initialization phase

for $i = 2..N$ **do**
  Let j be V_i's cluster label, $j = C(V_i)$. V_i computes $c_{fi}^j = E_{pk_1}(x_i)$ and $c_{ni}^j = E_{pk_1}(1)$;
  For $k \in \{1,2,3\} \setminus \{j\}$, V_i computes $c_{fk}^i = E_{pk_1}(0)$ and $c_{nk}^i = E_{pk_1}(0)$;
  V_i sends to S: $c_{f1}^i, c_{f2}^i, c_{f3}^i, c_{n1}^i, c_{n2}^i, c_{n3}^i$;
  S computes:
    $S_{f1} \leftarrow S_{f1} * c_{f1}^i$ $\quad$ $S_{n1} \leftarrow S_{n1} * c_{n1}^i$;
    $S_{f2} \leftarrow S_{f2} * c_{f2}^i$ $\quad$ $S_{n2} \leftarrow S_{n2} * c_{n2}^i$;
    $S_{f3} \leftarrow S_{f3} * c_{f3}^i$ $\quad$ $S_{n3} \leftarrow S_{n3} * c_{n3}^i$;
end

Algorithm 4. Secure multi-party mean computation protocol: sum calculus

$(r_{f1}^1, r_{f2}^1, r_{f3}^1, r_{n1}^1, r_{n2}^1, r_{n3}^1)$ and encrypts them with his own public key pk_1 to obtain 6 ciphertexts $(c_{f1}^1, c_{f2}^1, c_{f3}^1, c_{n1}^1, c_{n2}^1, c_{n3}^1)$. c_f contains encrypted features of vehicles. For example, V_1 belongs to the cluster 1, so V_1 encrypts her features x_1 as $c_{f1}^1 = E_{pk_1}(x_1 + r_{f1}^1)$, while c_{f2} and c_{f3} encrypt r_{f2}^1 and r_{f3}^1, respectively. Meanwhile, c_n indicates whether a vehicle belongs to a cluster or not. For V_1, c_{n1} encrypts $r_{n1}^1 + 1$ while c_{n2} and c_{n3} encrypt r_{n2}^1 and r_{n3}^1, respectively. The encryption results $(c_{f1}^1, c_{f2}^1, c_{f3}^1, c_{n1}^1, c_{n2}^1, c_{n3}^1)$ are transmitted to S which uses them to initialize the sum values $(S_{f1}, S_{f2}, S_{f3}, S_{n1}, S_{n2}, S_{n3})$.

Second, the insurer S requests from each vehicle $V_{i,i\neq 1}$ to provide its encrypted feature. To do so, each vehicle generates 6 ciphertexts $(c_{f1}^i, c_{f2}^i, c_{f3}^i, c_{n1}^i, c_{n2}^i, c_{n3}^i)$ as explained in Algorithm 4. V_i encrypts in c_{fj}^i her features when her cluster label $C(V_i)$ equals j, or 0 otherwise. In the same way, V_i encrypts in c_{nj}^i 1 when her cluster label $C(V_i)$ equals j, or 0 otherwise.

Finally, S picks 6 random values $(r_{f1}^S, r_{f2}^S, r_{f3}^S, r_{n1}^S, r_{n2}^S, r_{n3}^S)$, encrypts them with pk_1 and adds them to $(S_{f1}, S_{f2}, S_{f3}, S_{n1}, S_{n2}, S_{n3})$, respectively. Then, S transmits the obtained results to V_1 as presented in Algorithm 5. V_1 deciphers $(S_{f1}, S_{f2}, S_{f3}, S_{n1}, S_{n2}, S_{n3})$ and substracts $(r_{f1}^S, r_{f2}^S, r_{f3}^S, r_{n1}^S, r_{n2}^S, r_{n3}^S)$ to obtain $(f_1, f_2, f_3, n_1, n_2, n_3)$ which are retransmitted to S. S computes $f_i = f_i - r_{fi}^S, n_i = n_i - r_{ni}^S$ and the new clusters' means as: $\mu_i = f_i/n_i$.

Note that for simplicity concerns, we described the previous algorithm while considering that each vehicle V_i is transmitting one feature x_i. If vehicles are transmitting t features, each $c_{fj,j\in\{1,2,3\}}^i$ will be a vector of encrypted features or 0, namely $c_{fj}^i = (E_{pk_1}(x_1^i), ..., E_{pk_1}(x_t^i))$ or $(E_{pk_1}(0), ..., E_{pk_1}(0))$. In this case,

S picks 6 random values $r_{f_1}^S, r_{f_2}^S, r_{f_3}^S, r_{n_1}^S, r_{n_2}^S, r_{n_3}^S$;
S computes for $i \in \{1,2,3\}$:
$S_{f_i} \leftarrow S_{f_i} * E_{pk_1}(r_{f_i}^S)$ $\quad S_{n_i} \leftarrow S_{n_i} * E_{pk_1}(r_{n_i}^S)$;
S sends to V_1: $S_{f_1}, S_{f_2}, S_{f_3}, S_{n_1}, S_{n_2}, S_{n_3}$;
V_1 computes for $i \in \{1,2,3\}$:
$f_i = D_{sk_1}(S_{f_i}) - r_{f_i}^1$ $\quad n_i = D_{sk_1}(S_{n_i}) - r_{n_i}^1$;
V_1 sends to S: $f_1, f_2, f_3, n_1, n_2, n_3$;
S computes for $i \in \{1,2,3\}$:
$f_i \leftarrow f_i - r_{f_i}^S$ $\quad n_i \leftarrow n_i - r_{f_i}^S$;
S finally computes the new cluster means for $i \in \{1,2,3\}$: $\mu_i = f_i/n_i$;

Algorithm 5. Secure multi-party mean computation protocol: mean calculus

V_1 would compute in the initialization phase the following c_f values: $c_{f_1}^1 = (E_{pk_1}(x_1^1 + r_{f_1}^1), ..., E_{pk_1}(x_t^i + r_{f_1}^1))$, $c_{f_2}^1 = (E_{pk_1}(r_{f_2}^1), ..., E_{pk_1}(r_{f_2}^1))$ and $c_{f_3}^1 = (E_{pk_1}(r_{f_3}^1), ..., E_{pk_1}(r_{f_3}^1))$.

4.3 *k*-means Classification of New Drivers

Once the clusters are properly defined, it becomes easy to privately classify a new driver. The driver and the insurer will engage in the *secure two-party cluster attribution protocol* defined previously. However, the result of this computation is revealed to both parties. The driver cluster will determine the category of the driver and therefore his insurance fee.

5 Protocol Evaluation

In this section, we discuss the correctness and complexity of our proposed scheme.

5.1 Correctness

Correctness of the secure two-party cluster attribution protocol is trivial since the circuits are constructed to compute the correct value. We therefore focus on correctness of the secure multi-party mean computation protocol. When all the vehicles have sent their encrypted features, S computes:

$$S_{f_1} = \prod_{i=1..N} c_{f_1}^i \qquad S_{n_1} = \prod_{i=1..N} c_{n_1}^i$$
$$S_{f_2} = \prod_{i=1..N} c_{f_2}^i \qquad S_{n_2} = \prod_{i=1..N} c_{n_2}^i$$
$$S_{f_3} = \prod_{i=1..N} c_{f_3}^i \qquad S_{n_3} = \prod_{i=1..N} c_{n_3}^i$$

Which is equivalent to[3]:

$$S_{f_1} = c_{f_1}^{\ 1} * \prod_{C(V_i)=1} c_{f_1}^{\ i} * \prod_{C(V_i)\neq 1} c_{f_1}^{\ i}$$
$$S_{n_1} = c_{n_1}^{\ 1} * \prod_{C(V_i)=1} c_{n_1}^{\ i} * \prod_{C(V_i)\neq 1} c_{n_1}^{\ i}$$

We keep considering w.l.o.g that V_1 is in the first cluster. So, $c_{f_1}^{\ 1} = E_{pk_1}(x_1 + r_{f_1}^{\ 1})$ and $c_{n_1}^{\ 1} = E_{pk_1}(1 + r_{n_1}^{\ 1})$. Moreover, for i s.t. $C(V_i) = 1$, we have $c_{f_1}^{\ i} = E_{pk_1}(x_i)$ and $c_{n_1}^{\ i} = E_{pk_1}(1)$, while for i s.t. $C(V_i) \neq 1$ we have $c_{f_1}^{\ i} = E_{pk_1}(0)$ and $c_{n_1}^{\ i} = E_{pk_1}(0)$. Therefore, we obtain:

$$S_{f_1} = E_{pk_1}(x_1 + r_{f_1}^{\ 1}) * \prod_{C(V_i)=1} E_{pk_1}(x_i) * \prod_{C(V_i)\neq 1} E_{pk_1}(0)$$
$$S_{n_1} = E_{pk_1}(1 + r_{n_1}^{\ 1}) * \prod_{C(V_i)=1} E_{pk_1}(1) * \prod_{C(V_i)\neq 1} E_{pk_1}(0)$$

Thanks to the homomorphic property, we rewrite:

$$S_{f_1} = E_{pk_1}(r_{f_1}^{\ 1} + \sum_{C(V_i)=1} x_i)$$
$$S_{n_1} = E_{pk_1}(r_{n_1}^{\ 1} + \sum_{C(V_i)=1} 1) = E_{pk_1}(r_{n_1}^{\ 1} + card\{V_i/C(V_i) = 1\})$$

In the last exchange between V_1 and S, S hides the real values from V_1 by homomorphically adding $r_{f_i}^{S}$ and $r_{n_i}^{S}$, for $i \in \{1, 2, 3\}$. V_1 decrypts S_{f_i} and S_{ni} and removes his random values $r_{f_i}^{\ 1}$ and $r_{n_i}^{\ 1}$, for $i \in \{1, 2, 3\}$. Finally, S obtains the sums $f_j = \sum_{C(V_i)=j} x_i$ and the cardinal of each cluster $n_j = card\{V_i/C(V_i) = 1\}$ for $j \in \{1, 2, 3\}$. S is then able to compute the cluster means $\mu_j = f_j/n_j$.

5.2 Privacy

We check that our protocol fulfills the following requirements in the semi-honest model: (1) the inputs x_i and the closest cluster label $C(V_i)$ should be kept private for each vehicle V_i, (2) the clusters' means μ_i should be kept private for S.

The *secure two-party cluster attribution protocol* is proven in the literature [10] to compute privately the cluster attribution in the presence of semi-honest adversaries.

We therefore focus more on the *secure multi-party mean computation protocol*. In the first part of the protocol, S receives $c_{f_j}^{\ i}, c_{n_j}^{\ i}$ from each V_i for $i \in [\![1, N]\!]$ and $j \in \{1, 2, 3\}$. These values do not reveal information about the vehicles inputs since the Paillier cryptosystem provides indistinguishablity under chosen plaintext (IND-CPA). S cannot distinguish $c_{f_j}^{\ i} = E_{pk_1}(0)$ from $c_{f_j}^{\ i} = E_{pk_1}(x_i)$ and $c_{n_j}^{\ i} = E_{pk_1}(0)$ from $c_{n_j}^{\ i} = E_{pk_1}(1)$. Therefore, S cannot obtain any information

[3] we rewrite only S_{f_1} and S_{n_1} for the sake of clarity.

about a vehicle's features nor its intermediary cluster label. The second part of the protocol is the final exchange between S and V_1. S adds homomorphically to the encrypted sums S_{f_j} and S_{nj} the random values $r_{f_j}^S$ and $r_{n_j}^S$, for $j \in \{1, 2, 3\}$ to hide the sums from V_1.

Upon reception of the (S_{f_j}, S_{nj}), V_1 decrypts them with his private key and substracts his initial random values $r_{f_j}^1$ and $r_{n_j}^1$, for $j \in \{1, 2, 3\}$. The values obtained do not reveal any information since they are still masked by the random values of S. S is finally able to obtain the sums by substracting his random values $r_{f_j}^S$ and $r_{n_j}^S$ from f_j and n_j, for $j \in \{1, 2, 3\}$. In this second part of the protocol, each computed value does not reveal any sensitive information to the concerned party in the semi-honest model. However, when we consider that V_1 and S are malicious, they can collude and recover all the information regarding the other vehicles.

5.3 Complexity

Our protocol relies on Yao garbled circuit for the closest cluster computation. Indeed, once the distances to clusters' centers are computed with `SED` circuit, we compare them and return the index of the closest cluster using the `min` and `min_idx` circuits. In the sequel, we assume that the number of features per vehicle is n and that each feature is l-bit long.

We use the information from Table 1 [11,12] to estimate the number of gates of the closest cluster index calculus circuit (of Fig. 1a). The *size* of a circuit refers to the number of `AND` gates. Meanwhile, the multiplicative *depth* of a circuit refers to the maximum number of `AND` gates on any path of the circuit. Let us denote by $S()$ the function that returns a circuit size and by $D()$ the function that returns a circuit depth. The size and depth of the circuit for the closest cluster computation are $3S(SED) + S(min) + S(min\_idx)$ and $3D(SED) + D(min) + D(min\_idx)$, respectively.

Table 1. Number of gates of elementary circuits

Operand	Method	Depth	Size
$+$ [11]	ripple-carry	$l - 1$	$l - 1$
	Sklansky	$\lceil log(l) \rceil + 1$	$l\lceil log(l) \rceil$
$\times$ [11]	standard	$2l - 1$	$l^2 - l$
	Wallace	$2\lceil log(l) \rceil + 3$	$2l^2 + l\lceil log(l) \rceil$
`MUX` [13]	Kolesnikov and Schneider	1	l
$>$ [12]	Kolesnikov et al. [9]	l	l
	Garay et al. [14]	$\lceil log(l) \rceil + 1$	$3l - \lceil log(l) \rceil - 2$

Using inputs from Table 1, we obtain the results presented in Table 2 when we lower the size of our circuit for closest cluster index calculus to the maximum.

That is, if we consider a garbling method compatible with free-XOR [13], we will have to manage at most 4 ciphertexts per `AND` gate, for a total of $3nl^2 + 5l - 3n$ `AND` gates. In addition, we will engage in $3(n+1)l$ oblivious transfers between the circuit generator and verifier. It is clear that it is important to reduce at maximum the number n of features to improve the circuit computation time and to reduce bandwidth consumption during oblivious transfers.

Table 2. Number of gates of the closest cluster index calculus circuit

Circuit	Size
`SED`	$n(l^2-1)$
`min`	$3l$
`min_idx`	$2l$
Closest cluster index circuit	$3nl^2 + 5l - 3n$

The secure computation of new cluster centers relies on a homomorphic additive cryptosystem (as presented in Algorithms 3, 4 and 5). Each vehicle $V_{i,i\in[\![1,N]\!]}$ encrypts $3n+3$ plaintexts, i.e. a plaintext per feature and per cluster, using the public key of V_1. Then, it transmits them to the insurer S. S sums the received ciphertexts to obtain 3 vectors of encrypted features $S_{f_{i,i\in\{1,2,3\}}}$ and 3 sums of the total number of vehicles in a cluster $S_{ni,i\in\{1,2,3\}}$. Finally, V_1 decrypts these sums for S. That is, V_1 decrypts $3n+3$ ciphertexts. Note that the good choice of the homomorphic additive cryptosystem is of a great importance to reduce the bandwidth consumption during ciphertexts exchange. Indeed, as the number of exchanged ciphertexts $(3n+3)$ depends on the number of vehicles features (n), vehicles may have to transmit large bulks of data due to the size of ciphertext.

Note that for the PHYD use-case, we do not have real-time constraints as insurance fees are paid once a month and drivers may delay their payments by one month. That is, drivers clustering and then classification will be updated monthly. In practice, it is up to the insurer to fix the number of vehicles and to delimit the geographical area used for drivers clustering.

6 Conclusion

We presented in this work a privacy preserving k-means clustering and then classification for driving profiles. The proposed protocol avoids disclosing drivers personal data to semi-honest insurers. It relies on Yao's garbled circuit for the computation of distances to clusters' centers. In addition, it uses a homomorphic additive and semantic secure encryption scheme for the computation of clusters new centers. Our future work will consist in implementing a proof of concept of the proposed solution and providing a complete security proof for the protocol.

References

1. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE (1986)
2. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
3. Troncoso, C., Danezis, G., Kosta, E., Balasch, J., Preneel, B.: PriPAYD: privacy-friendly pay-as-you-drive insurance. IEEE Trans. Dependable Secure Comput. **8**(5), 742–755 (2011)
4. Kargl, F., Friedman, A., Boreli, R.: Differential privacy in intelligent transportation systems. In: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 107–112. ACM (2013)
5. Rizzo, N., Sprissler, E., Hong, Y., Goel, S.: Privacy preserving driving style recognition. In: 2015 International Conference on Connected Vehicles and Expo (ICCVE), pp. 232–237. IEEE (2015)
6. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, vol. 1, pp. 281–297 (1967)
7. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology, pp. 205–210. Springer, Boston, MA (1983). https://doi.org/10.1007/978-1-4757-0602-4_19
8. Naor, M., Pinkas, B., Pinkas, B.: Efficient oblivious transfer protocols. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms SODA 2001, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, pp. 448–457 (2001)
9. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_1
10. Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptology **22**(2), 161–188 (2009)
11. Buescher, N., Holzer, A., Weber, A., Katzenbeisser, S.: Compiling low depth circuits for practical secure computation. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 80–98. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45741-3_5
12. Schneider, T., Zohner, M.: GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 275–292. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_23
13. Kolesnikov, V., Schneider, T.: Improved garbled circuit: free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70583-3_40
14. Garay, J., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_22