PIPC: Privacy- and Integrity-Preserving Clustering Analysis for Load Profiling in Smart Grids

Haomiao Yang, *Member, IEEE*, Shaopeng Liang, Xizhao Luo, *Member, IEEE*, Dianhua Tang, Hongwei Li, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Generally, power utilities can utilize smart-meter data to extract load patterns through load-profiling technologies, such as *K*-means clustering. To improve the efficiency of load profiling, both *K*-means clustering and smart-meter data can be outsourced to powerful clouds. However, clouds are not completely trustworthy: private meter data may be used for commercial interests; *K*-means clustering may also be performed with fewer iterations to save computational costs, which violates the integrity of outsourced clustering. In this article, therefore, a secure *K*-means-clustering scheme is proposed, called privacy-preserving and integrity-preserving clustering (PIPC), which aims to protect the privacy and integrity of load profiling. To this end, two techniques are designed: 1) encrypted distance measurement, in which a public comparison matrix is constructed by securely embedding a secret key matrix and 2) integrity assurance, in which a specific Stackelberg game is designed to create economic incentives. The former, as the core of *K*-means clustering, can protect the privacy of meter data. The latter ensures that clouds can obtain the maximum utility only when clouds execute *K*-means clustering in an honest manner, thereby preserving the integrity of outsourced computing. Experimental results demonstrate that PIPC reaches high clustering accuracy and computational efficiency for load profiling while retaining smart-meter data privacy and outsourced-clustering integrity.**

***Index Terms*—Clustering analysis, integrity, load profiling, privacy, smart grid.**

Manuscript received April 12, 2021; revised August 13, 2021 and September 30, 2021; accepted November 1, 2021. Date of publication November 8, 2021; date of current version June 23, 2022. This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101360001; in part by the National Natural Science Foundation of China under Grant 62072081 and Grant U2033212; in part by the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027; in part by the CCF-Ant Group Research Fund; and in part by the Sichuan Science and Technology Program under Grant 2020JDTD0007. *(Corresponding author: Dianhua Tang.)*

Haomiao Yang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: haomyang@uestc.edu.cn).

Shaopeng Liang and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: polarisliang@qq.com; hongweili@uestc.edu.cn).

Xizhao Luo is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (e-mail: xzluo@suda.edu.cn).

Dianhua Tang is with the Key Laboratory of Science and Technology on Communication Security, The 30th Research Institute of China Electronic Technology Group Corporation, Chengdu 611731, China (e-mail: tangdianhua86@163.com).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/JIOT.2021.3125674

I. Introduction

THE INDUSTRIAL cyber–physical system (ICPS) with communication, computing, and industrial process control is regarded as the core technology of Industry 4.0. ICPS supports extensive applications, such as smart manufacturing, smart transportation, smart cities, etc. In particular, smart grids (SGs) combine electrical network infrastructures with cyber systems, exhibiting typical characteristics of an ICPS. The penetration of smart meters has generated a large amount of data that should be effectively processed to obtain actionable insights for power utilities to take proactive actions after an incident. Power utilities also leverage these data to extract load patterns, which show typical consumer behavior by the load-profiling technology. Specifically, load profiling can improve SG reliability and increase operational efficiency and, thus, there are a variety of SG applications, e.g., demand–response tariffs [1], load forecasting [2], and event detection [3]. In load profiling, *K*-means-clustering analysis is the most commonly used method [4], but smart-meter Big Data also poses a major challenge for efficient *K*-means-clustering tasks, especially if only power utilities on-premises resources can be used.

Numerous enterprises tend to outsource large-scale computing tasks to powerful clouds to save computational costs. Clouds can provide enterprises with advanced Big Data analysis services [5] and, thus, can be leveraged to perform *K*-means clustering on smart-meter Big Data. Nevertheless, these data may contain private information, e.g., enterprise electricity consumption has a high correlation with production activities. Cloud providers may be interested in such sensitive information for commercial interests, which causes significant privacy issues when meter data are uploaded to clouds [6]–[9]. Encryption before outsourcing can protect the privacy of meter data. Specifically, homomorphic encryption (HE) provides clouds with the capability of performing *K*-means clustering on encrypted meter data. Some HE-based *K*-means clustering schemes have been recently proposed [10], [11]; unfortunately, these schemes suffer security or computational efficiency issues. The scheme in [10] is not secure due to the insecurity of the underlying HE [12], and it is the public evaluation key that has been proven to leak private key information [13]. The scheme in [11] has high computational cost due to the usage of fully HE (FHE) [14]. Experiments show that it takes more than 600 h to perform HE-based K-means clustering on 400 data points of two dimensions. Moreover, as the core operation of K-means clustering, the distance comparison is commonly achieved by order-preserving encryption (OPE) [15], [16], whereas the other operations (such as addition and multiplication) are still implemented through HE. This leads to, in outsourced K-means clustering, multiple conversions between HE and OPE (i.e., the client converts HE-based/OPE-based ciphertexts to OPE-based/HE-based ciphertexts and sends converted ciphertexts back to untrusted clouds), which brings large communication overhead. Consequently, a HE-based K-means clustering scheme must be designed in which all K-means-clustering operations can be completed by using HE; meanwhile, this scheme can maintain its security and efficiency at the same time.

On the other hand, clouds do not always return the correct results in outsourced computing. For massive amounts of encrypted data, K-means clustering must run many iterations. Considering computational cost savings, clouds may run only fewer iterations, which will lead to incorrect results being returned. Therefore, cloud-based K-means clustering should preserve the integrity of outsourced computing. The integrity requires clouds to compute and return results in an honest manner, which means clouds should faithfully follow the designated K-means-clustering process and not reduce iterations. Honest computing also ensures the correctness of returned results [17]. Various integrity-assurance schemes for outsourced computing have been proposed [17]–[21]. Unfortunately, most schemes are computationally expensive, e.g., at least two replicas are required in [17], which means that the total cost is at least doubled. To save computational cost, power utilities naturally hope that K-means clustering is outsourced to only one cloud, but such outsourced computing schemes can still meet privacy and integrity requirements at the same time.

In this article, a Privacy-Preserving and Integrity-Preserving K-means Clustering (PIPC) scheme is proposed for load profiling. Through cloud-based outsourced K-means clustering, PIPC greatly improves the efficiency of load profiling on smart-meter Big Data. Furthermore, PIPC protects the privacy of smart-meter data by using vector HE (VHE), which supports efficient encrypted distance measurement [22]. In addition, by designing a specific Stackelberg game, an effective integrity-assurance strategy is developed that can retain the integrity of outsourced K-means clustering by incentivizing clouds to run K-means clustering honestly. Specifically, the contributions of this article are as follows.

1) PIPC is proposed to protect the privacy and integrity of load profiling. In particular, a novel technique for efficient encrypted distance measurement is designed that can preserve the privacy of the critical comparison operation in K-means clustering. The analysis demonstrates that PIPC guarantees the privacy of smart-meter data and the convergence of K-means clustering. Performance evaluation shows that compared with plaintext K-means clustering, PIPC maintains similar clustering accuracy and computational efficiency.
2) An effective integrity-assurance strategy is developed for honestly outsourced K-means clustering, which uses Stackelberg games to limit cloud cheating through economic incentives. Therefore, PIPC avoids heavy cryptographic calculations and cross-checking of multiple replicas. Moreover, PIPC supports K-means clustering on smart-meter data encrypted with different keys, which makes PIPC suitable for practical deployment.

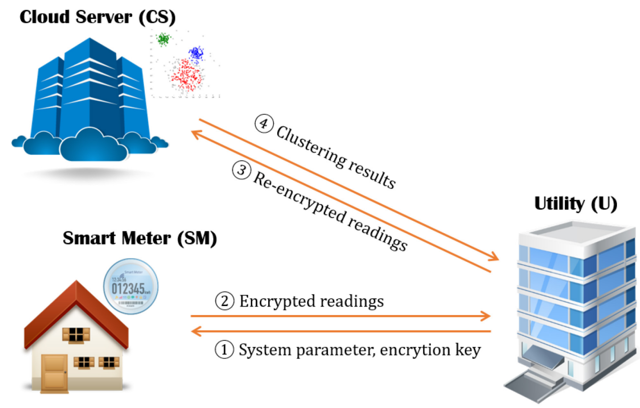


Fig. 1. System model under consideration.

The remainder of this article is organized as follows. In Section II, we present the problem statement. In Section III, we give the preliminaries. We then propose the PIPC scheme in Section IV, followed by privacy and convergence analysis in Section V and the performance evaluation in Section VI, respectively. In Section VII, we review related works. Finally, we conclude this article in Section VIII.

II. Problem Statement

A. System Model

Fig. 1 illustrates our system model involving three participants: 1) Utility (U); 2) Cloud Server (CS); and 3) Smart Meter (SM).

1) *Utility:* U sets up the system, then publicly releasing the system parameter. It also takes responsibility for SM's registration, then privately distributing the secret key to SM through a secure channel. Additionally, U reencrypts meter readings by using the key-switching operation [22], thereby guaranteeing the feasibility of homomorphic computation. It further sends reencrypted readings to CS, then obtaining returned clustering results from CS.
2) *Smart Meter:* SM is installed in the consumer's home collecting power consumption data in real time. It encrypts collected meter readings, then periodically sending U encrypted readings.
3) *Cloud Server:* CS performs K-means clustering on reencrypted readings, then returning results to U.

We have security assumptions as follows. U is trusted since it is actually acted as by the power company. SM is tamper-proof and cannot be physically compromised. CS is not fully trusted: it may pry the privacy of smart meter data; it may also violate the integrity of outsourced computing by running fewer iterations.

Notably, a Stackelberg game [23] between U and CS is created to motivate CS to run outsourced K-means clustering honestly. In our system model, U and CS will take actions in turn according to the opponent's strategy. This process will naturally form a two-party leader and follower game, just like the Stackelberg game. By analyzing how to achieve equilibrium, we can ensure that CS maximizes its utility in the case of no cheating.

The game requires a trusted third-party agent (TTA) to retain funds, such as a deposit of CS. Due to its immutability and traceability, the blockchain has the natural advantage of being a TTA. In this article, we use an off-the-shelf commercial blockchain component to play the role of TTA [24].

B. Design Objectives

PIPC needs to satisfy the following design objectives.

1) *Privacy:* Protect privacy for the entire clustering process.
2) *Integrity:* Guarantee integrity for outsourced clustering.
3) *Convergence and Accuracy:* Perform K-means clustering on outsourced encrypted smart data for load profiling, and maintain convergence and accuracy.
4) *Efficiency:* Manipulate massive encrypted meter data with high efficiency.

III. Preliminaries

A. VHE

The *Distance Comparison* is one of the core operations of K-means clustering; unfortunately, it is a challenging problem to achieve *Distance Comparison* in the ciphertext domain through traditional HE. To solve this challenge, we introduce a new HE method, i.e., VHE, which can support efficient encrypted distance comparison [25]. Although Bogos *et al.* [26] have demonstrated the security issues of the original VHE, an improved VHE ($\mathcal{VHE}$), proposed by Yang *et al.* [22], has fixed such security flaws, which is proved to be semantically secure. The $\mathcal{VHE}$ includes four probabilistic-polynomial-time algorithms as follows.

1) *Setup*(λ):
   a) On the input of security parameter λ, choose randomly two large distinct primes q_1 and q_2 and calculate $q = q_1 \cdot q_2$.
   b) Choose randomly $p, w, n, m \in \mathbb{Z}$ with $w(p-1) < q$, $p \ll q$ and $m < n$.
   c) Choose a discrete normal distribution χ on $\mathbb{Z}$.
   d) Publish publicly VHE parameter as $Param = (q, p, w, n, m, \chi)$.
2) *KG(Param):*
   a) Generate two matrices $\boldsymbol{P}_1, \boldsymbol{P}_2 \in \mathbb{Z}^{n\times n}$ such that $\boldsymbol{P}_1\boldsymbol{P}_2 = \boldsymbol{I}_1$, where $\boldsymbol{I}_1$ is an $n \times n$ identity matrix.
   b) Generate two matrices $\boldsymbol{S}_t = [\boldsymbol{I}_2, \boldsymbol{T}] \in \mathbb{Z}^{m\times n}$ and $\boldsymbol{M}_t = \begin{bmatrix} w\boldsymbol{I}_2 - \boldsymbol{TA} \\ \boldsymbol{A} \end{bmatrix} \in \mathbb{Z}^{n\times m}$, where $\boldsymbol{I}_2$ is an $m \times m$ identity matrix, $\boldsymbol{T} \leftarrow \chi^{m\times(n-m)}$ and $\boldsymbol{A} \leftarrow \chi^{(n-m)\times m}$.
   c) Calculate $\boldsymbol{S} = \boldsymbol{S}_t\boldsymbol{P}_1$ and $\boldsymbol{M} = \boldsymbol{P}_2\boldsymbol{M}_t$.
   d) Keep the secret key $\boldsymbol{S}$ privately and publish the public key $\boldsymbol{M}$.
3) *Enc*($\boldsymbol{x}, \boldsymbol{M}$)*:* Output a ciphertext $\boldsymbol{c} \in \mathbb{Z}_q^n$ by choosing a small noise $\boldsymbol{e} \leftarrow \chi^n$ and calculating $\boldsymbol{c} = \boldsymbol{Mx} + \boldsymbol{e}$, where $\boldsymbol{x} \in \mathbb{Z}_p^m$ is a plaintext and $\boldsymbol{M} \in \mathbb{Z}^{n\times m}$ is the public key.
4) *Dec*($\boldsymbol{c}, \boldsymbol{S}$)*:* Output a plaintext $\boldsymbol{x} \in \mathbb{Z}_p^m$ by calculating $\boldsymbol{x} = \lceil \boldsymbol{Sc}/w \rfloor_q$, in which $\lceil \cdot \rfloor_q$ denotes the nearest integer modulo q, $\boldsymbol{c} \in \mathbb{Z}_q^n$ is a ciphertext, and $\boldsymbol{S} \in \mathbb{Z}^{m\times n}$ is the secret key. Furthermore, we have an invariant as

$$\boldsymbol{Sc} = w\boldsymbol{x} + \boldsymbol{e}. \tag{1}$$

Then, we introduce *key switching*, which is a highly important operation in $\mathcal{VHE}$. The *key switching* converts an old key/ciphertext pair ($\boldsymbol{S}_{\text{old}}, \boldsymbol{c}_{\text{old}}$) to a new pair ($\boldsymbol{S}_{\text{new}}, \boldsymbol{c}_{\text{new}}$) with the same plaintext $\boldsymbol{x}$. Therefore, we have $\boldsymbol{c}_{\text{new}} = \boldsymbol{M}_1\boldsymbol{c}_{\text{old}}$ and $\boldsymbol{S}_{\text{new}}\boldsymbol{c}_{\text{new}} = \boldsymbol{S}_{\text{old}}\boldsymbol{c}_{\text{old}} = w\boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{M}_1$ is a key-switching matrix.

We build $\mathcal{VHE}$ security upon the learning with error (LWE) problem. $\mathcal{VHE}$ achieves semantic security assuming LWE intractability [22].

Algorithm 1 SKM

INPUT: β: Termination condition; K: Cluster number; $\boldsymbol{D} = \{\boldsymbol{x}_i | i = 1, 2, ..., N\}$: Plaintext dataset
OUTPUT: Clustering labels.

```
1:  Set clusters D_j empty and choose initial cluster centroids v_j, j = 1, 2, ..., K
2:  repeat
3:      for i = 1, ..., N do
4:          for j = 1, ..., K do
5:              Calculate d_ij ← ||x_i − v_j||
6:          end for
7:          if d_ij minimum then
8:              Assign x_i to D_j
9:          end if
10:     end for
11:     Recalculate centroids:
            v_j ← (Σ_{x_i∈D_j} x_i) / |D_j|, where j = 1, 2, ..., K
12: until β holds
13: return {(i, j) | x_i ∈ D_j, i = 1, 2, ..., N; j = 1, 2, ..., K}
```

B. K-Means Clustering

Algorithm 1 demonstrates standard K-means clustering (SKM). We divide Algorithm 1 into three stages: 1) *Closest Clustering Calculation* (lines 1–10); 2) *New Centroids Generation* (line 11); and 3) *Iteration Termination* (line 12).

C. Stackelberg Game

The Stackelberg game contains two players: 1) a leader and 2) a follower, who choose strategies sequentially, that is, the follower determines its strategy after observing the leader's strategy. Both are rational aiming to maximize their own utilities [23]. Note that we consider the Stackelberg game with imperfect information that is more realistic than perfect information because it assumes that players may not know all the actions adopted by other players. Thus, players have to make decisions with uncertainty. Concretely, the Stackelberg game is a quintuple of $\mathcal{G} = (\mathcal{P}, \mathcal{A}, p, \mathcal{I}, u)$, where:

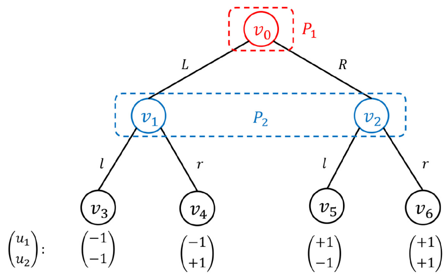


Fig. 2. Example of the Stackelberg game.

1) $\mathcal{P}$ represents the players set;
2) $\mathcal{A}$ represents the actions set;
3) p_i represents the player function of a player $i \in \mathcal{P}$;
4) $\mathcal{I}_i$ represents the information set of a player $i \in \mathcal{P}$;
5) u_i represents the utility of a player $i \in \mathcal{P}$.

$\mathcal{G}$ can be depicted as a game tree. Fig. 2 gives an example of game tree. Each node in the game tree has a label v_i. The player set is $\mathcal{P} = \{P_1, P_2\}$. The action set is $\mathcal{A} = \{L, R, l, r\}$. The function set p assigns actions to nonterminal nodes, where the set $\{L, R\}$ is assigned to v_0, and the set $\{l, r\}$ is assigned to v_1 and v_2. P_1 has $\{v_0\}$, and P_2 has $\{v_1, v_2\}$. The utility u_i is shown at the bottom. The information set is represented as an elongated dotted circle containing certain nodes. P_1 and P_2 have the information sets $\mathcal{I}_1 = \{v_0\}$ and $\mathcal{I}_2 = \{v_1, v_2\}$, respectively.

To analyze our game, we leverage a sequential equilibrium to solve the optimization problem of players' utilities [27]. The sequential equilibrium is composed of a *strategy profile* and a *belief system*. Player i's behavior strategy a_i assigns each information set a probability distribution over the actions. Player i's belief system β_i assigns each information set a probability distribution over the nodes of tree. Specifically, the belief system enables each player to develop the best strategy at each node. The sequential equilibrium is more stringent than the Nash equilibrium. It requires sequential rational strategies to be optimal not only in the whole game but also in each information set. Concretely, we give the definition of the sequential equilibrium as follows.

Definition 1: In a game $\mathcal{G}$, (a_i, β_i) is called a sequential equilibrium if regarding $\forall a_i \neq a_i'$ of Player i, we have

$$u_i(a_i, \mathcal{I}_i, \beta_i) \geq u_i\big(a_i', \mathcal{I}_i, \beta_i\big).$$

We take the game in Fig. 2 as an example. The game has a behavior strategy (a_1, a_2), where $a_1 = ([L(0), R(1)])$ and $a_2 = ([l(0), r(1)])$. Also, it has a belief system (β_1, β_2), where $\beta_1 = ([v_0(1)])$ and $\beta_2 = ([v_1(0), v_2(1)])$. We have $u_1(a_1, \mathcal{I}_1, \beta_1) > u_1(a_1', \mathcal{I}_1, \beta_1)$ and $u_2(a_2, \mathcal{I}_2, \beta_2) > u_2(a_2', \mathcal{I}_2, \beta_2)$. Consequently, there exists a unique sequential equilibrium $((a_1, a_2), (\beta_1, \beta_2))$ in the game.

IV. Proposed Scheme

We first design a distance comparison technique in encrypted domains. We then develop an integrity assurance strategy using the game theory. On the basis, we propose our PIPC scheme for secure load profiling.

A. Encrypted Distance Comparison

First, we design a novel encrypted distance comparison technique, called EDC. We assume that there exist three ciphertext vectors $\boldsymbol{c}_1'$, $\boldsymbol{c}_2'$, and $\boldsymbol{c}_3'$, respectively, corresponding to plaintexts $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ under the same key $\boldsymbol{S}$. We solve this challenge of measuring similarity on such ciphertexts, that is, without decryption, we can learn which vector between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is closer to $\boldsymbol{x}_3$ according to the Euclidean distances. To this end, we set a public comparison matrix as $\boldsymbol{H} \leftarrow \boldsymbol{S}^T\boldsymbol{S}$ and we have Theorem 1 as follows.

Theorem 1: There exist a comparison matrix $\boldsymbol{H}$ and two ciphertexts $\boldsymbol{c}_1'$ and $\boldsymbol{c}_2'$, respectively, corresponding to plaintexts $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Let $\boldsymbol{e}'$ denote a noise vector, and the following equation holds:

$$\big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big)^T \boldsymbol{H}\big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big) = w^2\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 + \boldsymbol{e}'.$$

Proof: First, according to (1), we have

$$\begin{aligned}
&\big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big)^T \boldsymbol{H}\big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big)\\
&\quad= \big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big)^T \boldsymbol{S}^T\boldsymbol{S}\big(\boldsymbol{c}_1' - \boldsymbol{c}_2'\big)\\
&\quad= \big(\boldsymbol{S}\boldsymbol{c}_1' - \boldsymbol{S}\boldsymbol{c}_2'\big)^T \big(\boldsymbol{S}\boldsymbol{c}_1' - \boldsymbol{S}\boldsymbol{c}_2'\big)\\
&\quad= (w\boldsymbol{x}_1 + \boldsymbol{e}_1 - w\boldsymbol{x}_2 - \boldsymbol{e}_2)^T(w\boldsymbol{x}_1 + \boldsymbol{e}_1 - w\boldsymbol{x}_2 - \boldsymbol{e}_2)\\
&\quad= w^2\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 + w(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T(\boldsymbol{e}_1 - \boldsymbol{e}_2)\\
&\quad\quad+ w(\boldsymbol{e}_1 - \boldsymbol{e}_2)^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) + \|\boldsymbol{e}_1 - \boldsymbol{e}_2\|^2.
\end{aligned}$$

Let $\boldsymbol{e}' = w(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T(\boldsymbol{e}_1 - \boldsymbol{e}_2) + w(\boldsymbol{e}_1 - \boldsymbol{e}_2)^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) + \|\boldsymbol{e}_1 - \boldsymbol{e}_2\|^2$. We then prove that $|\boldsymbol{e}'|$ is negligible, where $|\cdot|$ denotes the maximum entry in a vector. Assuming that $\boldsymbol{x} \in \mathbb{Z}^m$ with $|\boldsymbol{x}| = X$ and $\boldsymbol{e} \in \chi^m$ with $|\boldsymbol{e}| = E$, we have

$$\begin{cases}
w^2\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \leq 4w^2mX^2\\
w(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T(\boldsymbol{e}_1 - \boldsymbol{e}_2) \leq 4wmXE\\
w(\boldsymbol{e}_1 - \boldsymbol{e}_2)^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) \leq 4wmXE\\
\|\boldsymbol{e}_1 - \boldsymbol{e}_2\|^2 \leq 4mE^2.
\end{cases}$$

Furthermore, we have

$$|\boldsymbol{e}'| = 8wmXE + 4mE^2.$$

Thus, we have

$$\frac{|\boldsymbol{e}'|}{w^2\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2} = \frac{8wmXE + 4mE^2}{4w^2mX^2} = \frac{2E}{wX} + \frac{E^2}{w^2X^2}.$$

Since $X, E \ll w$, we have

$$\frac{2E}{wX} + \frac{E^2}{w^2X^2} \to 0.$$

Consequently, $|\boldsymbol{e}'|$ is negligible.

Obviously, the following proposition holds, which means even without decryption, we can evaluate the similarity of encrypted vectors. ∎

Proposition 1: Given a comparison matrix $\boldsymbol{H}$ and three ciphertext vectors $\boldsymbol{c}_1'$, $\boldsymbol{c}_2'$, and $\boldsymbol{c}_3'$, respectively, corresponding to plaintexts $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$, the following condition holds: if $(\boldsymbol{c}_1' - \boldsymbol{c}_3')^T\boldsymbol{H}(\boldsymbol{c}_1' - \boldsymbol{c}_3') \leq (\boldsymbol{c}_2' - \boldsymbol{c}_3)^T\boldsymbol{H}(\boldsymbol{c}_2' - \boldsymbol{c}_3')$, then

$$\|\boldsymbol{x}_1 - \boldsymbol{x}_3\| \leq \|\boldsymbol{x}_2 - \boldsymbol{x}_3\|. \tag{2}$$

Then, we simply analyze the intractability of extracting secret matrix $\boldsymbol{S}$ from public comparison matrix $\boldsymbol{H} = \boldsymbol{S}^T\boldsymbol{S}$.

TABLE I
VARIABLES IN OUR GAME

Variable	Meaning
b	Value of outsourced task
c	Computation cost of CS
d	Deposit of CS
v	Verification cost of U
w	Reward of CS
x	Cheating probability of CS
y	Verifying probability of U

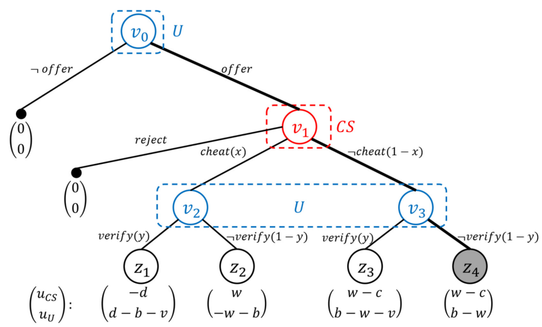


Fig. 3. Our Stackelberg game.

Without loss of generality, we consider a simple case of $\boldsymbol{S}$ containing only an entry, say s. Therefore, solving $s^2 = H \bmod (q_1 \cdot q_2)$ is required, in which q_1 and q_2 are two large primes, and $q_1 \neq q_2$. We can easily reduce the intractability of extracting $\boldsymbol{S}$ to the security of Rabin encryption [28], which has been proven to be secure.

It is worth noting that our EDC technique is not only feasible for K-means but also has the ability to extend some other popular clustering methods, such as DBSCAN [29], density peaks clustering [30], etc.

B. Integrity Assurance

We use the Stackelberg game to preserve the integrity of outsourced K-means clustering, which utilizes economic methods to encourage honest behaviors. Table I lists some variables used in the game. First, we have the following two assumptions.

1) *$0 < c < w$:* For CS, it will not accept the task, if its computation cost c is greater than reward w.
2) *$0 < v < b - w$:* For U, it will not outsource the task, if its verification cost v is greater than the profit, which is the task value b minus the reward w.

Then, we propose our Stackelberg game model, in which U is the leader and CS is the follower. Fig. 3 illustrates the game in the tree structure.

As seen, the player set is $\mathcal{P} = \{CS, U\}$; the action set is $\mathcal{A} = \{\text{offer}, \neg\text{offer}\} \cup \{\text{reject}, \text{cheat}(x), \neg\text{cheat}(1-x)\} \cup \{\text{verify}(y), \neg\text{verify}(1-y)\}$; the information set of CS is $\mathcal{I}_1 = \{v_1\}$; and the information set of U is $\mathcal{I}_2 = \{v_0, v_2, v_3\}$. Note that the possible result set is $\{z_1, z_2, z_3, z_4\}$, which includes the terminal nodes in the tree. The utilities of players are shown below each terminal node. Furthermore, we describe the game process as follows.

TABLE II
PAYOFF MATRIX OF OUR GAME

Case	Player	In	Out	Total
$1: z_1$	CS	0	d	$-d$
	U	d	$b+v$	$d-b-v$
$2: z_2$	CS	w	0	w
	U	0	$w+b$	$-w-b$
$3: z_3$	CS	w	c	$w-c$
	U	b	$w+v$	$b-w-v$
$4: z_4$	CS	w	c	$w-c$
	U	b	w	$b-w$

1) U chooses offer or ¬offer to CS. For offer, U asks the price w for the outsourced K-means clustering task. For ¬offer, the game terminates.
2) CS chooses reject or not. If reject, the game terminates. Otherwise, CS pays a deposit d to get the task.
3) CS performs the task with or without cheating, that is, in the probabilities of cheat(x) and ¬cheat($1-x$), respectively. Then, U decides to verify returned results or not, that is, in the probabilities of verify(y) and ¬verify($1-y$), respectively. As a consequence, there are four cases.

Case 1: CS chooses cheat(x) and U chooses verify(y). In this case, CS loses its deposit d; U obtains d but needs to take its verification cost v. In addition, U loses value b of the task itself.

Case 2: CS chooses cheat(x) and U chooses ¬verify($1-y$). In this case, CS obtains its reward w and withdraws its deposit d. Despite saving the verification cost v, U still loses the task value b and needs to pay reward w to CS.

Case 3: CS chooses ¬cheat($1-x$) and U chooses verify(y). In this case, CS gains the reward w and withdraws its deposit d, but needs to take its computation cost c. U acquires the task value b, but needs to take its verification cost v and pay reward w to CS.

Case 4: CS chooses ¬cheat($1-x$) and chooses ¬verify($1-y$). In this case, CS gains reward w and withdraws its deposit d, but needs to take its computation cost c; U obtains the task value b, but needs to pay reward w.

Accordingly, we get the payoffs (utilities) for the terminal node of each player, as shown in Table II. Furthermore, we define the payoff expectations of E_{CS} and E_U, respectively, as

$$\begin{aligned} E_{CS} &= -dxy + wx(1-y) + (w-c)(1-x)y \\ &\quad + (w-c)(1-x)(1-y) \\ &= (-d-w)xy + cx + w - c \end{aligned} \tag{3}$$

$$\begin{aligned} E_U &= (d-b-v)xy + (-w-b)x(1-y) \\ &\quad + (b-w-v)(1-x)y + (b-w)(1-x)(1-y) \\ &= (d+w)xy - 2bx - vy + b - w. \end{aligned} \tag{4}$$

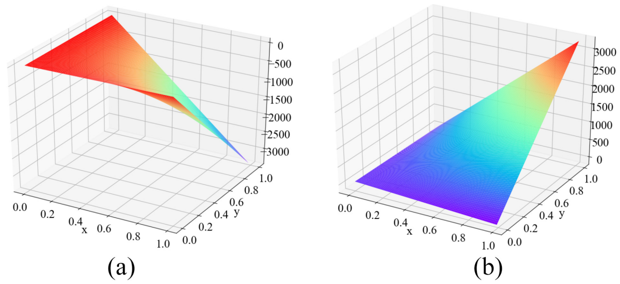


Fig. 4. Payoff expectations. (a) E_CS. (b) E_U.

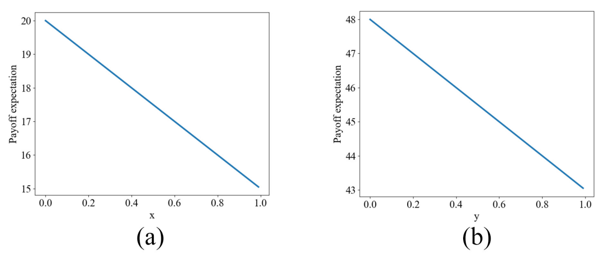


Fig. 5. Payoff expectations of fixing x or y. (a) $E_{CS}(y = 0.01)$. (b) $E_U(x = 0.01)$.

In Fig. 3, bold edges indicate actions taken by players to achieve a unique equilibrium. The gray z_4 is the reachable terminal node in the equilibrium. Therefore, if acquiring the unique equilibrium, the game always ends with z_4. We then prove that by appropriately setting variables, ¬cheat of *CS* and ¬verify of U can be achieved. To this end, ¬cheat of *CS* and ¬verify of U always lead to their highest payoffs. This means E_{CS} drops with the increase of the cheating probability x; E_U drops with the increase of the verifying probability y. Consequently, we have

$$\frac{\partial E_{CS}}{\partial x} = -dy - wy + c < 0$$

and

$$\frac{\partial E_U}{\partial y} = (d + w)x - v < 0.$$

Furthermore, we have

$$y > \frac{c}{d + w} \tag{5}$$

and

$$x < \frac{v}{d + w}. \tag{6}$$

According to (5) and (6), we first set a reasonable deposit d, and then calculate the cheating probability x to ensure that x is also within a reasonable range.

Based on the above considerations, we appropriately set the variables $b = 100$, $c = 30$, $v = 40$, $w = 50$, and $d = 3450$ to find the unique sequential equilibrium in the game. According to (5) and (6), we have the probabilities $x < 0.01$ and $y > 0.01$. Then, the payoff expectations are illustrated in Fig. 4.

To better observe the changes with probabilities, for E_{CS}, we fix $y = 0.01$; for E_U, we fix $x = 0.01$. As shown in Fig. 5, for $\forall 0 \le x' \le 1$, we have $E_{CS(x=0)} \ge E_{CS(x=x')}$, which means cheating always leads to the lower payoff; for $\forall 0 \le y' \le 1$, we have $E_{U(y=0)} \ge E_{U(y=y')}$, which means verifying always leads to the lower payoff. Therefore, *CS* will choose ¬cheat and U will choose ¬verify in order to reach z_4 and get their respective best payoffs. That is, we have the sequential equilibrium $((a_1, a_2), (\beta_1, \beta_2))$ in our game, where

$$\begin{cases} a_1 = ([\text{cheat}(0), \neg\text{cheat}(1)]) \\ a_2 = \big([\text{verify}(0), \neg\text{verify}(1)]\big) \\ \beta_1 = ([v_1(1)]) \\ \beta_2 = ([v_0(1)] \cup [v_2(0), v_3(1)]). \end{cases}$$

It means that the utility u_1 of *CS* satisfies $u_1(a_1, \mathcal{I}_1, \beta_1) > u_1(a_1', \mathcal{I}_1, \beta_1)$ and the utility u_2 of U satisfies $u_2(a_2, \mathcal{I}_2, \beta_2) > u_2(a_2', \mathcal{I}_2, \beta_2)$.

C. PIPC

Based on the above techniques, we propose our PIPC scheme that includes four stages as follows.

1) Initialization: U, in the stage, sets up the system and takes the responsibility of smart-meter registration.

Step 1: By invoking $\mathcal{VHE}.Setup(\lambda)$, U gets $Param = (q, p, w, n, m, \chi)$, the parameter of VHE. U also sets t, the reading interval of *SM*. Finally, U publicly releases $(Param, t)$.

Step 2: Each $SM_i, i = 1, 2, \ldots, N$ sends U its request for registration. U then checks request validity; if valid, U invokes $(\boldsymbol{M}_i, \boldsymbol{S}_i) \leftarrow \mathcal{VHE}.KG(Param)$, storing locally the secret key $\boldsymbol{S}_i$ and transmitting the encryption key $\boldsymbol{M}_i$ to SM_i through a secure channel.

Step 3: U initializes a Stackelberg game between U and *CS*, and publishes monetary variables b, c, v, and w. *CS* then delivers a deposit d to get the clustering task.

2) Preparation: SM_i encrypts its smart-meter reading and then U reencrypts it in the stage.

Step 1: SM_i, every t minutes, extracts its meter reading $\boldsymbol{x}_i$ and invokes $\boldsymbol{c}_i \leftarrow \mathcal{VHE}.Enc(\boldsymbol{x}_i, \boldsymbol{M}_i)$, then sending $\boldsymbol{c}_i$ to U.

Step 2: U performs key-switching operations [22] from $(\boldsymbol{c}_i, \boldsymbol{S}_i)$ to $(\boldsymbol{c}_i', \boldsymbol{S})$ with the same $\boldsymbol{x}_i$. In this case, all $\boldsymbol{c}_i'$ are with the same secret key $\boldsymbol{S}$. This ensures homomorphic computations of K-means clustering.

Step 3: U retrieves $\boldsymbol{x}_i$ by invoking $\mathcal{VHE}.Dec(\boldsymbol{c}_i, \boldsymbol{S}_i)$ for the purpose of real-time electricity surveillance.

Step 4: U calculates $\boldsymbol{H}$, the comparison matrix, as $\boldsymbol{H} \leftarrow \boldsymbol{S}^T\boldsymbol{S}$, finally uploading $\boldsymbol{H}$ and $\boldsymbol{c}_i'$ onto *CS*.

3) Computation: In the stage, *CS* runs privacy-preserving K-means clustering (PPKM).

Step 1: Algorithm 2 demonstrates PPKM. We further divide it into three substages: a) *Closest Clustering Calculation* (lines 1–10); b) *New Centroids Generation* (line 11); and c) *Iteration Termination* (line 12).

Step 2: *CS* finally returns U clustering labels $\{(i, j) | \boldsymbol{c}_i \in \boldsymbol{D}_j', i = 1, 2, \ldots, N; 1, 2, \ldots, K\}$.

4) Settlement: As discussed in Section IV-B, noncheating can be achieved by appropriately setting the payment, penalty, and verify probability. In this case, the rational *CS* will not cheat.

Algorithm 2 PPKM

INPUT: β': Termination condition; K: Cluster number; $\boldsymbol{H}$: Comparison matrix; $\boldsymbol{D}' = \{\boldsymbol{c}_i' | i = 1, 2, ..., N\}$: Ciphertext dataset
OUTPUT: Clustering labels

1: Set clusters $\boldsymbol{D}_j'$ empty and choose initial cluster centroids $\boldsymbol{v}_j', j = 1, 2, ..., K$
2: **repeat**
3: **for** $i = 1, \cdots, N$ **do**
4: **for** $j = 1, \cdots, K$ **do**
5: Calculate the distance $\boldsymbol{d}_{ij}' \leftarrow (\boldsymbol{c}_i' - \boldsymbol{v}_j')^T \boldsymbol{H} (\boldsymbol{c}_i' - \boldsymbol{v}_j')$
6: **end for**
7: **if** $\boldsymbol{d}_{ij}'$ minimum **then**
8: Assign $\boldsymbol{c}_i'$ to $\boldsymbol{D}_j'$
9: **end if**
10: **end for**
11: Recalculate centroids:
$$\boldsymbol{v}_j' \leftarrow \frac{\sum_{\boldsymbol{c}_i' \in \boldsymbol{D}_j'} \boldsymbol{c}_i'}{|\boldsymbol{D}_j'|}, \text{ where } j = 1, 2, ..., K$$
12: **until** β' holds
13: **return** $\{(i, j) | \boldsymbol{c}_i' \in \boldsymbol{D}_j', i = 1, 2, ..., N; j = 1, 2, ..., K\}$

V. Analysis

In this section, we first analyze the privacy of PPKM. Then, we prove that PPKM and SKM have the same convergence region.

A. Privacy

We have analyzed the privacy of VHE and EDC. We have also shown the integrity of outsourced K-means clustering. We will demonstrate the privacy of the entire PPKM process which includes the following three stages.

Privacy for Closest Clustering Calculation: From lines 1 to 10 in Algorithm 2, *CS* calculates and compares encrypted distances between K candidate centroids and ciphertext data set $\{\boldsymbol{c}_i' | i = 1, \ldots, N\}$ using EDC. *CS* further assigns data items to their respective closest clusters. In this way, what *CS* can obtain are only indices of items without leaking any contents of items, thereby protecting the privacy of this substage.

Privacy for New Centroids Generation: According to line 11 in Algorithm 2, *CS* recalculates clustering centroids over encrypted data items. The homomorphism of VHE preserves the privacy of this substage.

Privacy for Iteration Termination: According to line 12 in Algorithm 2, *CS* checks if indices in each cluster change. Consequently, this substage leaks no information on the contents of data items except the status of iteration termination.

B. Convergence

Since PPKM can simulate SKM, then if SKM converges, PPKM will also converge. Concretely, PPKM perfectly simulates SKM in the sense: we first assume that both SKM and PPKM are executed on the same data set and start with the same initial centroids. According to Algorithm 2, in each iteration, if a plaintext $\boldsymbol{x} \in \boldsymbol{D}_j$, then its corresponding ciphertext $\boldsymbol{c} \in \boldsymbol{D}_j'$. Furthermore, the homomorphism of VHE ensures that both SKM and PPKM have the same clustering assignments. Hence, the convergence of SKM guarantees that of PPKM.

TABLE III
Comparison on the BreastCancerWisconsin Data Set

# Items	# Iterations		Accuracy (%)	
	SKM	PIPC	SKM	PIPC
100	4.06	4.11	92.5	92
200	5.21	5.13	93.7	93.1
300	5.01	5.21	94.2	93.6
400	5.13	5.18	94.8	94
500	5.45	5.4	95.3	95.1
600	5.96	5.75	96.4	96

VI. Evaluation

In this section, we conduct extensive experiments to demonstrate the PIPC performance from accuracy, computational time, and communication overhead. To perform simulation, we use a smartphone with a Kirin@1600-MHz ARM processor and 2-GB RAM, running Android 4.2.2, which acts as *SM*. We also use a graphic workstation with NVIDIA V100 GPU and 32-GB RAM, running CUDA 10.1, which plays the roles of *U* and *CS*. Besides, we take practical security parameter $\lambda = 128$. To guarantee HE operation correctness, $\omega = 2^{30}$ is set, which has been verified through experiments. Experimental data come from the UCI repository[1] and our experimental source codes are available.[2]

A. Accuracy

To exhibit PIPC feasibility in load profiling, we use the "BreastCancerWisconsin" [31] data set, which contains 699 instances with ten attributes. We first perform preprocessing by normalizing attributes and then scaling them to integers at intervals of [0, 100]. Table III then compares the accuracy of K-means clustering between SKM and PIPC. PIPC achieves similar accuracy and iterations as SKM while providing privacy protection.

Furthermore, we run PIPC on 11 data sets to demonstrate PIPC wide availability. Table IV illustrates clustering time and accuracy with greatly varying values of N, M, and K, which represent the number of data items, attributes, and clusters, respectively. For all data sets, PIPC only increases time by up to 15%, and sacrifices accuracy of no more than 5%, compared with SKM. Therefore, PIPC achieves better clustering performance while preserving data privacy over multiple data sets.

Particularly, we use the "ElectricityLoadDiagrams" [32] data set including 370 customers with 140 256 attributes. Each attribute represents customer power consumption every 15 min. We further perform dimensionality reduction by summarizing 140 256 attributes into four values. We set $K = 5$ clustering 370 customers into five groups. Table V illustrates PIPC and SKM have the same clustering performance, such as the sum square error (SSE) and the number of iterations (#Iterations) when $K = 5$. We also conduct the experiment to explain why we set $K = 5$ on the ElectricityLoadDiagrams data set. As illustrated in Table V, when $K = 5$, PIPC achieves

[1] https://archive.ics.uci.edu/ml
[2] https://github.com/polaris-liang/PIPC/tree/master

TABLE IV
COMPARISON ON MULTIPLE DATA SETS

				Clustering time(ms)			Clustering accuracy(%)		
Dataset	K	N	M	SKM	PIPC	Added(%)	SKM	PIPC	Decreased(%)
3D spatial network	3	5000	3	37495	40125	7.01	95.6	94.7	0.95%
Breast cancer	2	600	9	2858	3101	8.5	94.1	92.8	1.40%
Tamilnadu electricity	10	1000	3	5455	6045	10.82	95	91.2	4.17%
Frogs MFCCs	20	1500	22	63674	72304	13.55	96.3	94.9	1.48%
Chest-mounted Accelerometer	8	1200	4	5434	5946	9.42	96.7	96	0.73%
Grammatical facial expression	2	1000	300	122770	134568	9.61	92.3	91.7	0.65%
Seeds dataset	16	200	7	774	856	10.59	92.9	92.4	0.54%
Hw dataset	30	1000	6	9688	10289	6.2	95.1	92.2	3.15%
Geographical original of music	40	800	68	122808	140123	14.1	91.6	91.2	0.44%
Household power consumption	50	3000	7	1000807	1150254	14.93	94.8	92.7	2.27%
Winequality-red	5	4000	12	210242	240154	14.23	93.3	92	1.41%

TABLE V
COMPARISON OVER THE ELECTRICITYLOADDIAGRAMS DATA SET

K	Algorithm	#Iterations	SSE	Number of Cluster									
				#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
2	SKM	6	4.80e+19	3	367	–	–	–	–	–	–	–	–
	PIPC	6	4.80e+19	3	367	–	–	–	–	–	–	–	–
3	SKM	9	2.94e+19	3	18	349	–	–	–	–	–	–	–
	PIPC	9	2.94e+19	3	18	349	–	–	–	–	–	–	–
4	SKM	15	1.75e+19	1	3	21	345	–	–	–	–	–	–
	PIPC	15	1.75e+19	1	3	21	345	–	–	–	–	–	–
5	SKM	19	1.13e+18	1	3	17	47	301	–	–	–	–	–
	PIPC	19	1.13e+18	1	3	17	47	301	–	–	–	–	–
6	SKM	29	2.15e+18	1	3	3	16	50	297	–	–	–	–
	PIPC	29	2.15e+18	1	3	3	16	50	297	–	–	–	–
7	SKM	38	3.92e+18	1	3	3	15	24	52	272	–	–	–
	PIPC	38	3.92e+18	1	3	3	15	24	52	272	–	–	–
8	SKM	33	7.30e+18	1	3	3	15	23	39	86	200	–	–
	PIPC	33	7.30e+18	1	3	3	15	23	39	86	200	–	–
9	SKM	54	6.53e+18	1	3	3	14	14	18	34	85	198	–
	PIPC	54	6.53e+18	1	3	3	14	14	18	34	85	198	–
10	SKM	52	3.86e+18	1	1	3	3	14	14	18	34	84	198
	PIPC	52	3.86e+18	1	1	3	3	14	14	18	34	84	198

both a lower SSE and a fewer number of iterations, which indicates that PIPC has a good clustering performance when $K = 5$.

B. Computational and Communication Overheads

We evaluate computational and communication costs of PIPC. For $i = 1, 2, \ldots, N$, computational cost involves the time of encrypting $\boldsymbol{x}_i$ and generating $\boldsymbol{H}$. Communication cost mainly contains the consumed bandwidth of transmitting $\boldsymbol{c}_i$ from *SM* to U, and uploading $\boldsymbol{H}$ and $\boldsymbol{c}'_i$ from U to *CS*. We ignore the small communication cost of sending clustering labels from *CS* to U. Concretely, we analyze these costs as follows.

We first consider computational time. VHE encrypts $\boldsymbol{x}_i$ containing N items with M attributes. Owing to batch encryption, the time complexity is $\mathcal{O}(N)$ not $\mathcal{O}(NM)$. This significantly improves efficiency. On the one hand, U only generates $\boldsymbol{H}$ once and, thus, the generation time is a constant. As illustrated in Fig. 6(a), the time cost increases in linearity with N, and the average time is about 42 ms for each item.

For communication costs, $\boldsymbol{c}_i$ (or $\boldsymbol{c}'_i$) and $\boldsymbol{H}$, respectively, include $N(M+1)$ and $(M+1)^2$ integers; therefore, communication costs in total are $(M+2N+1)(M+1)$. As shown in Fig. 6(b), the communication cost linearly increases with N. Besides, the communication cost has more sensitivity to M than N. This is because it grows linearly with N but quadratically with M. Nonetheless, the overall communication cost is no more than 1.3 MB, even for 700 items with 100 attributes.

We use a smartphones to simulate *SM* and, thus, evaluate *SM* encryption time based on the smartphone with an ARM processor. First, we rebuild VHE encryption program using Java 1.7.0 and NDK R9, and then install it in the smartphone. Furthermore, we change processor frequencies by acquiring root privileges of the android system. Fig. 7 shows *SM* encryption time considering the processor frequency and vector dimension. With frequency drop or dimension increase, the encryption time grows. Nevertheless, it only takes 136.09 ms to encrypt a 10-D vector with the lowest frequency of 208 MHz. Hence, VHE encryption is much efficiently suitable for the resource-constraint *SM*.

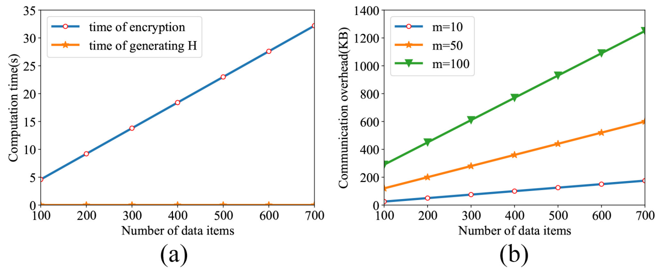


Fig. 6. Costs for privacy preservation. (a) Time cost. (b) Communication cost.

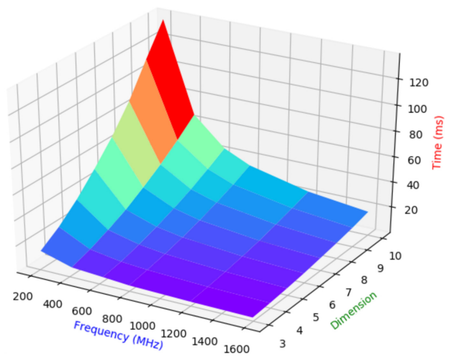


Fig. 7. *SM* encrypting time.

TABLE VI
COMPARISON ON HE-BASED CLUSTERING

Clustering Scheme	HE Primitive	Clustering Time
The Scheme in [11]	TFHE [14]	619 hours
The Scheme in [33]	HEAAN [34]	83 minutes
PIPC	VHE [25]	9 seconds

To further illustrate the efficiency of PIPC, we perform a clustering time comparison using the LSUN data set, as shown in Table VI. In the clustering schemes of Jäschke and Armknecht [11] and Cheon *et al.* [33], and PIPC, which are based on the HE primitives of HEAAN [14], TFHE [34], and VHE [25], respectively, the clustering time is 619 h, 83 min, and 9 s, respectively. As a consequence, PIPC achieves higher efficiency than the other two clustering schemes in terms of clustering time.

VII. Related Works

HE can compute over ciphertexts, which provides a promising approach to protect the privacy of machine learning (ML), which mainly includes classification and clustering. Researchers have proposed various HE-based classification schemes [35]–[38], but to the best of our knowledge only two completely HE-based clustering schemes exist. They involve mean-shift clustering [33] and K-means clustering [11] that are based on HEAAN [34] and TFHE [14], respectively.

On the other hand, there have been numerous studies on the integrity assurance of outsourced computing. They mainly include cryptography-based methods [20], [21], [39]–[41] or replication-based methods [17], [19], [42]. For the former, the client outsources the computationally intensive task to a single cloud server, which then returns cryptographically verifiable results to the client. While cloud computing saves a large amount of computing cost, it is not enough to sustain complex cryptographic computations. This implies that the client has to pay extra expenses incurred by cryptographic algorithms that are 10^3–10^9 times more than that of the computing task itself [40], thereby bringing an extremely great financial cost to the client. For the latter, the client first assigns the same task to multiple clouds, and then they independently calculate the task. Next, the client cross-checks the returned results. Unfortunately, these technologies are still too expensive. In [19], at least three replicas are needed, which implies that the total cost to the client has increased by at least three times. Dong *et al.* [17] utilized the game theory and smart contracts for verification, using only two clouds. Nevertheless, two cloud providers might collude to maximize profits. To resist collusion, Dong *et al.* designed three contracts, i.e., *Prisoner*, *Colluder*, and *Traitor*. This would bring heavy financial pressure to the client because of the cost of renting duplicate clouds and the use cost of multiple contracts.

Apart from that, the existing works can only unilaterally achieve privacy or integrity. Consequently, it is essential to design PIPC schemes for load profiling.

VIII. Conclusion

We have proposed PIPC, a secure and efficient K-means clustering for practical load profiling. In addition, PIPC can support K-means clustering on smart-meter data that are encrypted with different keys, which makes PIPC suitable for multiuser scenarios. In the future work, more machine-learning schemes with privacy and integrity preservation will be investigated, such as HE-based ridge linear regression and deep-learning models.

References

[1] J. Ponoćko and J. V. Milanović, "Forecasting demand flexibility of aggregated residential load using smart meter data," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5446–5455, Sep. 2018.

[2] Z. Liao, H. Pan, X. Fan, Y. Zhang, and L. Kuang, "Multiple wavelet convolutional neural network for short-term load forecasting," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9730–9739, Jun. 2021, doi: 10.1109/JIOT.2020.3026733.

[3] S. S. Negi, N. Kishor, K. Uhlen, and R. Negi, "Event detection and its signal characterization in PMU data stream," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3108–3118, Dec. 2017.

[4] B. Cui, A. K. Srivastava, and P. Banerjee, "Synchrophasor-based condition monitoring of instrument transformers using clustering approach," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2688–2698, May 2020.

[5] Q. Liu, P. Hou, G. Wang, T. Peng, and S. Zhang, "Intelligent route planning on large road networks with efficiency and privacy," *J. Parallel Distrib. Comput.*, vol. 133, pp. 93–106, Nov. 2019.

[6] M. Al-Asli, M. E. S. Elrabaa, and M. Abu-Amara, "FPGA-based symmetric re-encryption scheme to secure data processing for cloud-integrated Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 446–457, Feb. 2019.

[7] J. Ni, K. Zhang, K. Alharbi, X. Lin, N. Zhang, and X. S. Shen, "Differentially private smart metering with fault tolerance and range-based filtering," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2483–2493, Sep. 2017.

[8] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, and X. Shen, "Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox," *IEEE Trans. Cloud Comput.*, early access, Apr. 29, 2020, doi: 10.1109/TCC.2020.2991167.

[9] D. Liu, J. Ni, C. Huang, X. Lin, and X. Shen, "Secure and efficient distributed network provenance for IoT: A blockchain-based approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7564–7574, Aug. 2020.

[10] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user K-means clustering," in *Proc. IEEE CIC*, 2015, pp. 80–89.

[11] A. Jäschke and F. Armknecht, "Unsupervised machine learning on encrypted data," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2018, pp. 453–478.

[12] D. Liu, "Practical fully homomorphic encryption without noise reduction," IACR, Lyon, France, Rep. 468/2015, 2015.

[13] Y. Wang, "Notes on two fully homomorphic encryption schemes without bootstrapping," IACR, Lyon, France, Rep. 519/2015, 2015.

[14] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2017, pp. 377–408.

[15] Y. Peng, H. Li, J. Cui, J. Zhang, J. Ma, and C. Peng, "Hope: Improved order preserving encryption with the power to homomorphic operations of ciphertexts," *Sci. China Inf. Sci.*, vol. 60, no. 6, pp. 62–101, 2017.

[16] N. Almutairi, F. Coenen, and K. Dures, "Secure third party data clustering using data: Multi-user order preserving encryption and super secure chain distance matrices," in *Proc. SGAI*, 2018, pp. 3–17.

[17] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, "Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing," in *Proc. ACM CCS*, 2017, pp. 211–227.

[18] Z. Yan, X. Yu, and W. Ding, "Context-aware verifiable cloud computing," *IEEE Access*, vol. 5, pp. 2211–2227, 2017.

[19] T. Distler, C. Cachin, and R. Kapitza, "Resource-efficient Byzantine fault tolerance," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2807–2819, Sep. 2016.

[20] R. S. Wahby, M. Howald, S. Garg, A. Shelat, and M. Walfish, "Verifiable ASICs," in *Proc. IEEE SP*, 2016, pp. 759–778.

[21] R. Zhu, C. Ding, and Y. Huang, "Efficient publicly verifiable 2PC over a blockchain with applications to financially-secure computations," in *Proc. ACM CCS*, 2019, pp. 633–650.

[22] H. Yang, S. Liang, J. Ni, H. Li, and X. S. Shen, "Secure and efficient k NN classification for Industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10945–10954, Nov. 2020.

[23] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjrungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[24] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Zug, Switzerland, Ethereum, Yellow Paper, 2014.

[25] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *Proc. Inf. Theory Appl. Workshop (ITA)*, 2014, pp. 1–9.

[26] S. Bogos, J. Gaspoz, and S. Vaudenay, "Cryptanalysis of a homomorphic encryption scheme," *Cryptogr. Commun.*, vol. 10, no. 1, pp. 27–39, 2018.

[27] D. M. Kreps and R. Wilson, "Sequential equilibria," *Econometrica J. Econometr. Soc.*, vol. 50, no. 4, pp. 863–894, 1982.

[28] M. Elia, M. Piva, and D. Schipani, "The rabin cryptosystem revisited," *Appl. Algebra Eng. Commun. Comput.*, vol. 26, no. 3, pp. 251–275, 2015.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[30] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[31] W. Wolberg, "Breast cancer Wisconsin (Original)," Data Set, UCI Machine Learning Repository, 1992.

[32] A. Trindade, "ElectricityLoadDiagrams20112014," Data Set, UCI Machine Learning Repository, 2015.

[33] J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2019, pp. 227–249.

[34] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2017, pp. 409–437.

[35] S. Kim, M. Omori, T. Hayashi, T. Omori, L. Wang, and S. Ozawa, "Privacy-preserving naive bayes classification using fully homomorphic encryption," in *Proc. NeurIPS*, 2018, pp. 349–358.

[36] X. Sun, P. Zhang, J. K. Liu, J. Yu, and W. Xie, "Private machine learning classification based on fully homomorphic encryption," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 352–364, Apr.–Jun. 2020.

[37] P. Xie, B. Wu, and G. Sun, "BAYHENN: Combining Bayesian deep learning and homomorphic encryption for secure DNN inference," in *Proc. IJCAI*, 2019, pp. 4831–4837.

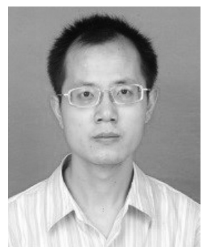
[38] P. Fenner and E. Pyzer-Knapp, "Privacy-preserving gaussian process regression–A modular approach to the application of homomorphic encryption," in *Proc. AAAI*, 2020, pp. 3866–3873.

[39] R. Liu, H. Wang, P. Mordohai, and H. Xiong, "Integrity verification of K-means clustering outsourced to infrastructure as a service (IaaS) providers," in *Proc. SIAM ICDM*, 2013, pp. 632–640.

[40] M. Walfish and A. J. Blumberg, "Verifying computations without reexecuting them," *Commun. ACM*, vol. 58, no. 2, pp. 74–84, 2015.

[41] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang, "Enabling verifiable and dynamic ranked search over outsourced data," *IEEE Trans. Services Comput.*, early access, Jun. 11, 2019, doi: 10.1109/TSC.2019.2922177.

[42] J. van den Hooff, M. F. Kaashoek, and N. Zeldovich, "VerSum: Verifiable computations over large public logs," in *Proc. ACM CCS*, 2014, pp. 1304–1316.



Haomiao Yang (Member, IEEE) received the M.S. and Ph.D. degrees in computer applied technology from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2004 and 2008, respectively.

He has worked as a Postdoctoral Fellow with the Research Center of Information Cross over Security, Kyungil University, Gyeongsan, South Korea, for one year until June 2013. He is currently an Associate Professor with the School of Computer Science and Engineering and the Center for Cyber Security, UESTC. His research interests include cryptography, cloud security, and cybersecurity for aviation communication.

Shaopeng Liang received the B.S. degree in information security from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently pursuing the M.S. degree in cyberspace security.

His current research interests include big data security and artificial intelligence security.

Xizhao Luo (Member, IEEE) received the B.S. and M.S. degrees from Xi'an University of Technology, Xi'an, China, in 2000 and 2003, respectively, and the Ph.D. degree from Soochow University, Suzhou, China, in 2010.

He is currently an Associate Professor with the School of Computer Science and Technology, Soochow University, where he held a postdoctoral position with the Center of Cryptography and Code, School of Mathematical Science. His main fields of interest are cryptography and computational complexity.

Dianhua Tang is currently pursuing the Ph.D. degree with the University of Electronic Science and Technology of China, Chengdu, China.

He is a Senior Engineer with the China Electronic Technology Cyber Security Co., Ltd., Chengdu. His recent research interests include fully homomorphic encryption (FHE) and secure multiparty computation (MPC), and post-quantum cryptography.

Hongwei Li (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in June 2008.

He is currently the Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He worked as a Postdoctoral Fellow with the University of Waterloo, Waterloo, ON, Canada, from October 2011 to October 2012. His research interests include network security and applied cryptography.

Prof. Li is the Distinguished Lecturer of IEEE Vehicular Technology Society.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, social networks, 5G and beyond, and vehicular ad hoc and sensor networks.

Prof. Shen received the R. A. Fessenden Award in 2019 from IEEE, Canada, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015, and the Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, and the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL and the Vice President on Publications of the IEEE Communications Society. He is a registered Professional Engineer of Ontario, Canada, a Fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.