View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Institutional Knowledge at Singapore Management University

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

5-2020

PMKT: Privacy-preserving Multi-party Knowledge Transfer for financial market forecasting

Zhuoran MA
Xidian University

Jianfeng MA
Xidian University

Yinbin MIAO
Xidian University

Kim-Kwang Raymond CHOO
University of Texas at San Antonio

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Finance and Financial Management Commons, and the Information Security Commons

Citation

MA, Zhuoran; MA, Jianfeng; MIAO, Yinbin; CHOO, Kim-Kwang Raymond; LIU, Ximeng; WANG, Xiangyu; and YANG, Tengfei. PMKT: Privacy-preserving Multi-party Knowledge Transfer for financial market forecasting. (2020). *Future Generation Computer Systems*. 106, 545-558. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/5069

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email library@smu.edu.sg.

Author

Zhuoran MA, Jianfeng MA, Yinbin MIAO, Kim-Kwang Raymond CHOO, Ximeng LIU, Xiangyu WANG, and Tengfei YANG

This journal article is available at Institutional Knowledge at Singapore Management University: https://ink.library.smu.edu.sg/sis_research/5069

Published in Future Generation Computer Systems, 2020, 106, 545-558.
https://doi.org/10.1016/j.future.2020.01.007

PMKT: Privacy-preserving Multi-party Knowledge Transfer for financial market forecasting

Zhuoran Ma[a,b,c], Jianfeng Ma[a,c,*], Yinbin Miao[a,b,c], Kim-Kwang Raymond Choo[d], Ximeng Liu[e,f], Xiangyu Wang[a,c], Tengfei Yang[a,c]

[a] *School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[b] *State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*
[c] *Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China*
[d] *School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore*
[e] *Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA*
[f] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China*

A B S T R A C T

While decision-making task is critical in knowledge transfer, particularly from multi-source domains, existing knowledge transfer approaches are not generally designed to be privacy preserving. This has potential legal and financial implications, particularly in sensitive applications such as financial market forecasting. Therefore, in this paper, we propose a Privacy-preserving Multi-party Knowledge Transfer system (PMKT), based on decision trees, for financial market forecasting. Specifically, in PMKT, we leverage a cryptographic-based model sharing technique to securely outsource knowledge reflected in decision trees of multiple parties, and design a secure computation mechanism to facilitate privacy-preserving knowledge transfer. An encrypted user-submitted request from the target domain can also be sent to the cloud server for secure prediction. Also, the use of decision trees allows us to provide interpretability of the predictions. We then demonstrate how PMKT can achieve privacy guarantees, and empirically show that PMKT achieves accurate forecasting without compromising on accuracy.

Keywords:
Knowledge transfer
Privacy-preserving
Decision tree
Secure computation
Multi-parties
Financial market forecasting

1. Introduction

Financial technology (also referred to as FinTech) is crucial in our society, particularly in technologically advanced nations such as Australia, U.S. and Singapore. Examples of FinTech include financial market forecasting services (such as those using machine learning techniques). A study by KPMG,[1] for example, estimated that FinTech services are expected to grow with machine learning in the next few years, potentially providing additional profit-making opportunities for the FinTech stakeholders. In machine learning-based financial market forecasting services, we need sufficient datasets that are relevant and of a certain quality as they play a crucial role in the accuracy of the forecasting results [1–4].

In the real-world, there are only relatively few financial dataset collected in a single financial institution, particularly in smaller and regional financial institutions, which can be used to train the machine learning models. This is partly because of the costs and resources required in the identification, generation and collection of suitably high quality financial data. This complicates the tasks of training an accurate classifier model, and consequently impact on the provision of precise financial market forecasting services. However, if we are able to source relevant financial datasets from different institutions, located in different countries or continents, and combine them, then we will be able to utilize such combined datasets for machine learning model training. In the literature, there are a number of multi-party knowledge transfer techniques proposed, such as the technique presented in [5].

Cloud computing can be leveraged yo mitigate the constraints due to storage and access, due to its underpinning features such as flexible/on-demand access, and reduced computation costs. There are, however, a number of security and privacy considerations in the deployment of systems to facilitate knowledge sharing across multi-parties, particularly those located in countries with different, or conflicting, privacy regulations (e.g., European countries and Asian countries). The latter is particularly problematic for sharing of financial data collected from multiple parties, as such data usually contain sensitive information (e.g., trading volume, and sensitive customer information). Therefore, to achieve

* Corresponding author at: School of Cyber Engineering, Xidian University, Xi'an 710071, China.

E-mail addresses: emmazhr@163.com.cn (Z. Ma), jfma@mail.xidian.edu.cn (J. Ma), ybmiao@xidian.edu.cn (Y. Miao), raymond.choo@fulbrightmail.org (K.-K.R. Choo), snbnix@gmail.com (X. Liu), xywang_xidian@163.com (X. Wang), yangtf@stu.xidian.edu.cn (T. Yang).

1 https://assets.kpmg/content/dam/kpmg/xx/pdf/2018/07/h1-2018-pulse-of-fintech.pdf, last accessed Nov 30, 2019.

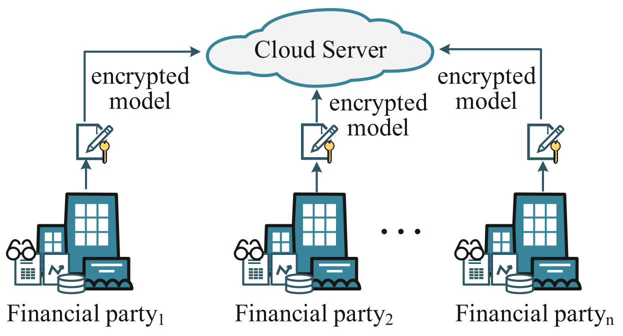


Fig. 1. A system overview of multi-party model sharing.

privacy protection, a number of cryptographic-based outsourced data sharing techniques have been presented in the literature, for example by leveraging homomorphic encryption [6–11]. While using homomorphic encryption allows one to achieve relatively strong privacy guarantees, the efficiency of data computation over homomorphic encryption is low. In addition, it is challenging, if not impractical, to control, trace and maintain data integrity during data sharing processes.

Since parameters of a classifier model also contain invaluable knowledge [12], we posit the potential of using a cryptographic-based outsourced model sharing to achieve multi-party knowledge sharing [13]. Such an approach allows us to achieve strong privacy guarantees, and to avoid information leakage during training, we can pre-train a classifier model prior to outsourcing — see Fig. 1.

In addition to high forecasting accuracy, the interpretability of classifier models is also crucial in the financial industry [14]. Decision Tree (DT) [15], a widely used predictive method, has a number of attractive properties such as intuitive representation, and easy understanding. Therefore, DTs can be trained over multiple related but different datasets to extract relevant knowledge across multi-parties for knowledge transfer [16,17]. Although these DTs have a structural difference, there are similarities in the knowledge described, as demonstrated in the existing research literature [18,19]. However, these existing knowledge transfer on DTs approaches rely on the participation of model parameters, which easily expose sensitive information.

Since there are no existing privacy protection solutions for knowledge transfer on DTs, it is non-trivial to design the privacy-preserving knowledge transfer. Inspired by the above motivations, in this paper, we first propose a **P**rivacy-preserving **M**ulti-parties **K**nowledge **T**ransfer scheme on decision trees for financial market forecasting, hereafter referred to as **PMKT**. PMKT employs the Paillier cryptosystem to design privacy-preserving solutions for each component of knowledge transfer on DTs. The main contributions of this paper are summarized as follows:

- *Secure outsourced model sharing among multi-party*: Since classifier models contain sensitive information, it is necessary to protect the model's privacy. To securely share models across multiple parties, PMKT employs the Paillier cryptosystem to encrypt models prior to sharing them to the cloud server. This allows us to minimize the leakage of the model's privacy.
- *Secure knowledge transfer*: PMKT is designed to provide privacy-preserving knowledge transfer on DTs, which constructs an accurate classifier model for the target domain. Specially, PMKT facilitates secure computation to prevent privacy leakage.
- *Secure classification on-the-fly*: PMKT allows an authorized user in the target domain to upload his/her encrypted real-time requests for secure classification. Upon receiving the requests, the trained transfer model returns relevant encrypted forecasting results without compromising forecasting accuracy.

In the next section, we will briefly review the related literature.

2. Related work

Cryptographic-based Knowledge Sharing. To prevent privacy leakage during knowledge sharing, we can leverage homomorphic encryption, such as partially homomorphic encryption (PHE) and fully homomorphic encryption (FHE). PHE consists of additive homomorphic encryption (e.g., Paillier cryptosystem [20] and Bresson cryptosystem [21]) and multiplicative homomorphic encryption (e.g., ElGamal cryptosystem [22] and unpadded RSA cryptosystem [23]), which only satisfies either additive or multiplicative homomorphic operations. Although some cryptosystems such as the BGN cryptosystem [24] support both additive and multiplicative homomorphic, these schemes only provide one multiplicative homomorphic operation and limited number of additive homomorphic operations. To avoid this limitation, Gentry et al. [25] presented the first FHE scheme, which can provide arbitrary number of additive and multiplicative homomorphic operations. However, the implementation complexity of FHE results in it not been widely employed in real-world applications.

As discussed earlier, there are a number cryptographic-based outsourced data sharing schemes designed for sharing knowledge among multiple parties, such as those presented in [26–28]. While existing data sharing-based methods are promising, since they not only require multiple data owners to contribute individual data in a privacy-preserving manner, (i.e., data are shared in the form of ciphertext form without compromising the privacy of data owners), but they also implement computations over ciphertexts for the classifier training. However, a number of challenges remain. For example, the massive encrypted data that are uploaded to the cloud will require many homomorphic operations over ciphertexts, which is clearly expensive (e.g., due to the time-consuming homomorphic computations during the process of training a classifier). Therefore, instead of uploading the encrypted data to the cloud server, Li et al. [29] proposed a secure classification scheme. The latter allows the classifier owner to share his/her encrypted classifier model with the cloud server, in order to provide the classification service. However, existing outsourced model sharing schemes only support a single-party setting. In [30,31], the privacy-preserving framework is proposed to outsource the classifier models from multiple parties to facilitate global model training. However, they have to guarantee that all classifiers from multiple parties are trained over the single-source domain. The trade-off is a lower accuracy rate.

Knowledge Transfer. Knowledge transfer techniques (also known as transfer learning techniques) [32] allow the distribution of training datasets and test datasets to be different. As parameters shared by the individual models from related source domains contain invaluable knowledge that can be transferred in the target domain, model-based knowledge transfer approaches have been widely used in practice. Benefits of using DT include intuitive representation and easy implementation [33], and not surprisingly there have been attempts to provide knowledge transfer using DTs [34,35]. Such approaches generally rely on the parameters of DTs. A number of DT similarity measurement frameworks have been presented in recent times [36,37], which depend on structural similarity of two DTs from different domains (i.e., source domains and a target domain). In addition, the correctness of knowledge among different domains can be evaluated under the similarity measurement of DT. However, such DT-based knowledge transfer techniques require the participation information of all source domains and the target domain. This is a potential privacy leakage vector.

In this paper, our goal is to utilize the outsourced model sharing from multi-source domains to improve the performance of the target domain. Clearly, privacy preservation guarantee of knowledge transfer across different domains is important in practice, and this necessitates the design of a privacy-preserving knowledge transfer using DTs across multiple different parties scheme.

3. Preliminaries

In this section, we review the definitions of knowledge transfer and similarity measurement on DTs, as well as the Paillier cryptosystem.

3.1. Knowledge transfer

Given a dataset of row vectors $X = \{x_1, \ldots, x_m\}^T$, $Y = \{y_1, y_2, \ldots, y_m\}^T$, where T denotes the transpose operation, X denotes a specific training sample, Y denotes the set of all labels, $x_i \in X$ denotes the ith sample vector, y_i denotes the corresponding label of x_i. Note that the symbol $\mathcal{F}$ represents the space of all sample vectors. Let a dataset be a domain $\mathcal{D}$ consisting of a feature space $\mathcal{F}$ and a marginal probability distribution $P(X)$, which is represented as $\mathcal{D} = \{\mathcal{F}, P(X)\}$. In general, given two different domains $\mathcal{D}_1 = \{\mathcal{F}_1, P_1(X)\}$ and $\mathcal{D}_2 = \{\mathcal{F}_2, P_2(X)\}$, the condition $\mathcal{D}_1 \neq \mathcal{D}_2$ means $\mathcal{F}_1 \neq \mathcal{F}_2$ or $P_1(X) \neq P_2(X)$.

Knowledge Transfer: Given multi-source domains $\mathcal{D}_S = \{\mathcal{D}_{S_1}, \ldots, \mathcal{D}_{S_n}\}$ and a target domain $\mathcal{D}_\mathcal{T}$, the goal of knowledge transfer is to help the target domain to build an accurate classifier by leveraging the knowledge in multi-source domains, where $\mathcal{D}_{S_i} \neq \mathcal{D}_\mathcal{T}$ ($i = 1$ to n).

3.2. Similarity measurement of decision trees

We choose Gini index [38] as the decision criteria (i.e., the best node split is the one with maximum of Gini index), namely CART decision tree, note that we only focus on the binary classification in PMKT. A DT consists of a root node, internal nodes (imply decisions) and leaf nodes (imply classes), and each branch from the root to a leaf node represents a decision sequence of a prediction outcome which may indicate valuable knowledge in the classification [39]. The DT structure is shown in Fig. 2, where $\mathcal{F} = \{f_1, \ldots, f_v\}$ denotes the feature set, $C = \{c_1, c_2\}$ represents the label set. Besides, each leaf node corresponds to a label c_i, dom_{f_k} describes the value global range of f_k, and $ran_{f_k}(c_i)$ denotes the value range of f_k when the label is c_i, where $f_k \in \mathcal{F}$.

In statics, similarity measurement of DT [37] is a well-known approach to compare similarity between two DTs according to the structure component. $V(c_i)$ (see Eq. (1)) denotes the relative importance over feature space on the label c_i, and the relative importance of each feature f_k is defined as $\frac{|ran_{f_k}(c_i)|}{|dom_{f_k}|}$, where $|ran_{f_k}(c_i)| = max_{f_k}(c_i) - min_{f_k}(c_i)$, $|dom_{f_k}| = max_{f_k} - min_{f_k}$. It is worth noticing that $max_{f_k}(c_i)$ and $min_{f_k}(c_i)$ respectively represent the max value and the min value of the feature f_k on the label c_i, and $|dom_{f_k}|$ implies the global value range of each feature, which has no difference over label set.

$$V(c_i) = \prod_{k=1}^{v} \frac{|ran_{f_k}(c_i)|}{|dom_{f_k}|} \quad (f_k \in \mathcal{F}). \tag{1}$$

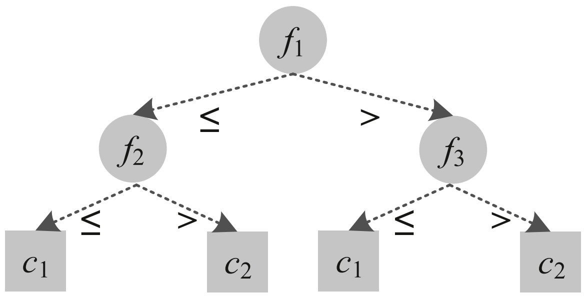


Fig. 2. An example DT structure.

For two related but structurally different DTs, namely dt_1 and dt_2, the similar measurement is represented as Eq. (2), where $SP_{dt}(c_i)$ denotes a structural probability for the label c_i on a DT.

$$\begin{aligned} Sim(dt_1, dt_2) &= Sim(SP_{dt_1}(C), SP_{dt_2}(C)), \\ &= 1 - \prod_{i=1}^{C} |SP_{dt_1}(c_i) - SP_{dt_2}(c_i)|; \\ SP_{dt}(c_i) &= \frac{V(c_i)}{\sum_{j=1}^{C} V(c_j)}. \end{aligned} \tag{2}$$

3.3. Public-key cryptosystem

Here, we introduce the proposed public-key scheme [20,40] based on the Paillier cryptosystem in Fig. 3, which acts as the basis cryptographic primitive in PMKT.

4. Problem formulation

In this section, the system model, threat model and privacy requirements associated with our PMKT are demonstrated, respectively.

4.1. System model

As illustrated in Fig. 4, our system model involves a target domain, n source domains, a Cloud Service Provider (CSP), a Cloud Platform (CP) and a Key Certification Authority (KCA).

- *Key certification authority*: The trusted KCA is responsible for the distribution and management of all keys in the system.
- *Target domain*: A target domain is a single financial institution with comparatively few data. It is hard to provide accurate financial market forecasting service for constantly generated requests from users. Hence, the target domain needs to make use of related knowledge from source domains included in classifiers to construct a more accurate knowledge transfer model.
- *Source domains*: Source domains are similar and related financial institutions/parties to the target domain. Each source domain owns tremendous quantities of related financial data as his/her individual private datasets and is willing to contribute his/her classifier model to help the target domain construct an accurate classifier model. Therefore, source domains should encrypt the classifiers trained over local financial datasets before sharing them with CP.
- *Cloud platform*: CP, which is a semi-honest cloud server with adequate storage space, performs knowledge transfer for the target domain, and provides online financial market forecasting service for authorized users.
- *Cloud service provider*: CSP provides computation services over ciphertexts for knowledge transfer and online financial market forecasting service.

Public-key cryptosystem definition

- **KeyGen**: Given the security parameter ς, distinct odd primes p, q, where $|p| = |q| = \varsigma$. Let $N = pq$ and $\lambda = lcm(p-1, q-1)$, then choose a generator $g \in \mathbb{Z}^*_{N^2}$ of order $(p-1)(q-1)/2$, where $g = (1+N) \bmod N^2$. Finally, the public key is $pk = (N, g)$ and secret key is $sk = \lambda$.
- **KeyS**: The secret key $sk = \lambda$ can be randomly split into two secret shares (i.e., λ_1 and λ_2), which satisfy $\lambda_1 + \lambda_2 \equiv 0 \bmod \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \bmod N^2$ at the same time.
- **Enc**: Given a plaintext $m \in \mathbb{Z}_N$ and a random element $\beta \in \mathbb{Z}^*_{N^2}$, the algorithm outputs the ciphertext as $[\![m]\!] = g^m \cdot \beta^N \bmod N^2$.
- **Dec**: Given a ciphertext $[\![m]\!]$, the secret key $sk = \lambda$ and a function $L(x) = \frac{x-1}{N}$, then the plaintext m is encrypted as $m = L([\![m]\!]^\lambda \bmod N^2)\lambda^{-1} \bmod N$.
- **SDec**: Given a ciphertext $[\![m]\!]$ and a secret share λ_i (i.e., λ_1 or λ_2), compute the decryption share $[\![m]\!]_i = [\![m]\!]^{\lambda_i} \bmod N^2$.
- **WDec**: Given the decryption shares $[\![m]\!]_1$ and $[\![m]\!]_2$, compute the plaintext $m = L([\![m]\!]_1 \cdot [\![m]\!]_2)$.

Fig. 3. Definition of public-key cryptosystem.

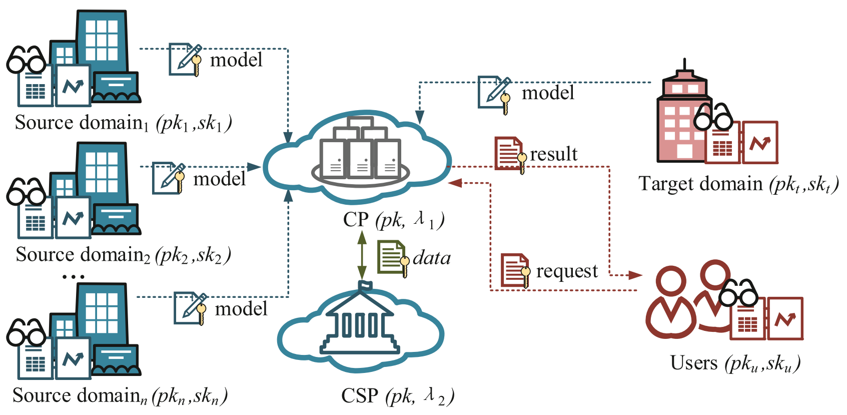

model
Source domain$_1$ (pk_1, sk_1)
model
Source domain$_2$ (pk_2, sk_2)
model
Source domain$_n$ (pk_n, sk_n)
model
result
CP (pk, λ_1)
data
CSP (pk, λ_2)
request
Target domain (pk_t, sk_t)
Users (pk_u, sk_u)

Fig. 4. System model.

- *User*: An authorized user belonging to the target domain can submit his/her encrypted financial requests to CP for accurate financial market forecasting service. When the corresponding encrypted forecasting results are returned by CP, the user can obtain the final forecasting results.

4.2. Threat model

In our threat model, we assume that CP, CSP, source domains, the target domain and users are *honest-but-curious* entities that faithfully performs the pre-defined protocols, but try to learn more private information from other entities. Hence, an active adversary $\mathcal{A}^*$ is introduced with the following abilities in our model:

- $\mathcal{A}^*$ may intercept communications links to acquire the encrypted data.
- $\mathcal{A}^*$ may compromise CP to learn more information about the encrypted models under individual public keys from all source domains and the target domain.
- $\mathcal{A}^*$ may compromise CSP to learn more information about all ciphertexts sent by CP.
- $\mathcal{A}^*$ may compromise a domain (i.e., a source domain or a target domain) to learn more information about other domains.
- $\mathcal{A}^*$ may compromise users in order to acquire information of all encrypted results belonging to the user.

In addition, we assume that CP and CSP are two independent and non-colluding servers. Such assumption has been commonly used in [33,41]. As the most cloud servers are well-established, the collusion will harm their individual interests and credibility. Besides, it is difficult for attackers to compromise both CP and CSP at the same time.

4.3. Privacy requirements

For privacy preservation guarantees, we aim to achieve the following privacy requirements.

- Model Privacy: A privacy-preserving scheme requires the sensitive information of a party cannot be leaked to entities other than its owner. The models containing sensitive parameters should not be disclosed to untrusted parties.
- Privacy-preserving knowledge transfer: To maintain privacy preservation guarantees, the sensitive parameters and intermediate computational results cannot be leaked during the process of knowledge transfer.
- Request Privacy: An authorized user submits a request to CP for accurate financial market forecasting service without leaking the private request to others.
- Result Privacy: CP returns a forecasting result corresponding to the user-submitted financial request, note that the result is only known to the user.

5. Basic components of PMKT

5.1. Key distribution

For the initialization of security primitive in PMKT, the fully-trusted KCA generates keys for all entities in the system model. The detailed process of key distribution is shown as follows:

- The KCA generates key pairs (pk_i, sk_i) for the ith source domain ($i = 1$ to n), (pk_t, sk_t) for a target domain, and (pk_u, sk_u) for an authorized user, respectively.
- The KCA first generates a key pair (pk, sk) for each cloud server (CP or CSP), and splits the secret key sk into λ_1 and λ_2, sk_u into λ_{u_1} and λ_{u_2}, sk_t into λ_{t_1} and λ_{t_2} and sk_i into λ_{i_1} and λ_{i_2}, respectively. Then, λ_1, λ_{u_1}, λ_{t_1}, λ_{i_1} are sent to CP, λ_2, λ_{u_2}, λ_{t_2}, λ_{i_2} are sent to CSP ($i = 1$ to n).

Table 1
Descriptions of secure protocols.

Protocol	Description
Addition	Return the addition result $[\![m_a + m_b]\!]$
Subtraction	Return the subtraction result $[\![m_a - m_b]\!]$
Scaling-down	Return the scale-down result $[\![\lfloor \frac{m}{10^\varepsilon} \rfloor]\!]$
Multiplication	Return the multiplication result $[\![m_a]\!] \times [\![m_b]\!]$
Comparison	Return the comparison result of $[\![m_a]\!]$ and $[\![m_b]\!]$
Division	Return the division result $\frac{[\![m_a]\!]}{[\![m_b]\!]}$
Argmax	Return the index of the largest value of an encrypted set
Transformation	Transform ciphertexts under an authorized key to another authorized key

5.2. Data preprocess

As our cryptosystem only supports integer numbers, the real numbers need to be converted into integers. In our system, we use the approximation and expansion method to solve the limitation. The public scale factor ε is used to convert a real number into an integer. For example, all the data are multiplied by 10^ε, and then approximate these values to the nearest integers for the further encryption, where ε is an integer number. Taking 3.141 as an example, 3.141 can multiply 10^2 ($\varepsilon = 2$), and then the approximation method is used to get 314. In PMKT, without a specific hint, $[\![x]\!]$ implies the encrypted data under the public key pk.

5.3. Secure computation

Since our cryptographical scheme only satisfies additive homomorphic property, we further design secure computation over ciphertexts by revising the original secure computation protocols [42] and using the blinding random number technique in order to avoid the computation limitation, as shown in Table 1. The detailed presentations of secure computation are described in Fig. 5, note that some computations require CP to interact with CSP.

Correctness. The correctness of the above secure computation is verified by the following process:

- The correctness of secure scaling-down:

$$[\![\left\lfloor \frac{M}{10^\varepsilon} \right\rfloor]\!] - [\![r]\!] = [\![\left\lfloor \frac{m + r \cdot 10^\varepsilon}{10^\varepsilon} \right\rfloor]\!] - [\![r]\!]$$
$$= [\![\left\lfloor \frac{m}{10^\varepsilon} \right\rfloor]\!].$$

- The correctness of secure multiplication:

$$[\![res]\!] \cdot S_1 \cdot S_2 \cdot S_3 = [\![M_a \times M_b]\!] \cdot S_1 \cdot S_2 \cdot S_3$$
$$= [\![(m_a + r_1) \times (m_b + r_2)]\!] \cdot [\![r_1 \times r_2]\!]^{N-1} \cdot [\![m_a]\!]^{N-r_2} \cdot [\![m_b]\!]^{N-r_1}$$
$$= [\![(m_a + r_1) \times (m_b + r_2) - r_1 \times r_2 - r_2 \times m_a - r_1 \times m_b]\!]$$
$$= [\![m_a]\!] \times [\![m_b]\!].$$

- The correctness of the secure comparison:

$$[\![res]\!] = ([\![m'_a]\!] \cdot [\![m'_b]\!]^{N-1})^{r_1} \cdot [\![r_2]\!]$$
$$= [\![r_1(2m_a - 2m_b) + r_2]\!]$$
$$\approx [\![r_1(2m_a - 2m_b)]\!] \ (r_2 \ll r_1).$$

  Thus, when $m_a \geqslant m_b$, $|res| \leqslant |N|/2$, when $m_a < m_b$, $|res| > |N|/2$.
- The correctness of the secure division: The Newton–Raphson method is used to approximate the inverse of $[\![m_b]\!]$. Each iterative computation (see Eq. (4)) involves two times secure multiplication and one time secure addition. The correctness of secure division depends on both secure addition and secure multiplication. The corresponding correctness of secure addition and secure multiplication is shown in the above analysis for details.
- The correctness of the secure argmax: Designed on the secure comparison, comparing two encrypted numbers in the set of encrypted data, the final result is computed. The correctness of secure argmax is based on the corresponding correctness of secure comparison.
- The correctness of the secure transformation:

$$[\![m']\!]_{pk_2} - [\![r]\!]_{pk_2} = [\![m + r]\!]_{pk_2} - [\![r]\!]_{pk_2} = [\![m]\!]_{pk_2}.$$

Table 2
Notation descriptions.

Notation	Description
$[\![dom]\!]$	Auxiliary information of a DT
$[\![ran]\!]$	Feature range of a DT
$C = \{c_1, c_2\}$	Label set of a DT
$\mathcal{F} = \{f_1, \ldots, f_v\}$	Feature set of a DT
$paths$	Tree path sets of a DT
$\|ran_{f_k}(c_i)\|$	Feature range $f_k \in \mathcal{F}$ on the label c_i of a DT
$\|dom_{f_k}\|$	Feature range $f_k \in \mathcal{F}$ of a DT
$\frac{\|ran_{f_k}(c_i)\|}{\|dom_{f_k}\|}$	Feature relative importance $f_k \in \mathcal{F}$ with label c_i
$V(c_i)$	Feature relative importance space of a DT
$SP_{dt}(c_i)$	Structural probability of label c_i in dt
$Sim(dt_1, dt_2)$	Similar measurement between two DTs dt_1 and dt_2
w_i	Transfer weight of a DT
$h_{dt}(x)$	Classification result of instance x in dt
$H(x)$	Final classification result of the integrated transfer

6. PMKT framework

In this section, we detailedly describe how to construct the privacy-preserving knowledge transfer and successfully implement secure prediction in the multi-source domains settings. The notation definitions are described in Table 2.

6.1. Overview of PMKT framework

To guarantee both privacy preservation and high accuracy, the involved phases are shown as follows, — see Fig. 6.

- *Locally training decision trees*: Since the knowledge transfer is a method that transfers valuable knowledge contained in the pre-trained models from multi-source domains to the target domain, it is required that DTs should have been pre-trained over individual datasets from multi-source domains.
- *Integrated knowledge transfer*: In PMKT, our ultimate goal is to securely implement the DT-based knowledge transfer. Initially, DT models shared by multi-source domains are measured for a similarity with the DT model shared by the target domain in a privacy-preserving manner. Then, according to the similarity of each DT from source domains, the corresponding transfer weight is assigned to the DT with a privacy-preserving weight distribution approach. Finally, we construct an integrated knowledge transfer model across multiple DTs with privacy guarantees.
- *Secure classification on-the-fly*: We provide secure classification on the integrated knowledge transfer model. On receiving an encrypted user-submitted request, each DT generates individual forecasting results on the encrypted request, which is a part of the integrated knowledge transfer model. Then, combined these results with their individual transfer weights, the final forecasting result of the integrated knowledge transfer will be returned.

Secure computation

Secure Addition/Subtraction: Based on the additive property, it is easy to implement ciphertext addition $[\![m_a + m_b]\!] = [\![m_a]\!] \cdot [\![m_b]\!]$ and the ciphertext subtraction $[\![m_a - m_b]\!] = [\![m_a]\!] \cdot [\![m_b]\!]^{N-1}$. The process of either addition or subtraction can be completed on CP alone without interacting with CSP.

Secure Scaling-down: To main the scale factor of an encrypted data $[\![m]\!]$ without expanding, we design the secure scaling-down to cut the scale factor down by ε, which is shown as follows:

- CP first chooses a random number r from $\mathbb{Z}_N$, computes $[\![r']\!] = [\![r \cdot 10^{\varepsilon}]\!]$, and calculates $[\![M]\!] = [\![m]\!] \cdot [\![r']\!]$. Then, CP obtains the decryption share $[\![M]\!]_1$ with **SDec** algorithm. Finally, CP sends $[\![M]\!]$ and $[\![M]\!]_1$ to CSP for the next calculation.
- On receiving these encrypted data, CSP first calls **SDec** and **WDec** algorithms to obtain M, then approximately cuts down the scale factor by $\left\lfloor \frac{M}{10^{\varepsilon}} \right\rfloor$, finally sends $[\![\left\lfloor \frac{M}{10^{\varepsilon}} \right\rfloor]\!]$ to CP after the encryption.
- CP can obtain the final result with cutting the scale factor down by ε by removing the random number with secure subtraction as $[\![\left\lfloor \frac{M}{10^{\varepsilon}} \right\rfloor]\!] - [\![r]\!]$.

Secure Multiplication: To compute $[\![m_a]\!] \times [\![m_b]\!] = [\![m_a \cdot m_b]\!]$, the specific process is shown as follows:

- CP first chooses two random numbers $r_1, r_2 \in \mathbb{Z}_N$ and uses the blinding technique to generate $[\![M_a]\!] = [\![m_a]\!] \cdot [\![r_1]\!]$ and $[\![M_b]\!] = [\![m_b]\!] \cdot [\![r_2]\!]$. Then, CP uses the **SDec** algorithm with λ_1 to obtain $[\![M_a]\!]_1$ and $[\![M_b]\!]_1$. Finally, these decryption shares, $[\![M_a]\!]$ and $[\![M_b]\!]$ are sent to CSP.
- On receiving these encrypted data, CSP uses **SDec** and **WDec** algorithms with λ_2 to obtain M_a and M_b. Besides, CSP computes $res = M_a \times M_b$, then res is encrypted under the public key pk and sent to CP.
- Once receiving $[\![res]\!]$, CP runs $S_1 = [\![r_1 \times r_2]\!]^{N-1}$, $S_2 = [\![m_a]\!]^{N-r_2}$, $S_3 = [\![m_b]\!]^{N-r_1}$ and $[\![m_a]\!] \times [\![m_b]\!] = [\![res]\!] \cdot S_1 \cdot S_2 \cdot S_3$ to remove random numbers.

In order to keep the scale factor from expanding, after each multiplication operation over ciphertexts, the results need to be scale down with the secure scaling-down as $[\![\left\lfloor \frac{m_a \cdot m_b}{10^{\varepsilon}} \right\rfloor]\!]$.

Secure Comparison: To implement the comparison operation $Comparison([\![m_a]\!], [\![m_b]\!])$ between $[\![m_a]\!]$ and $[\![m_b]\!]$, the specific process is shown as follows:

- CP first calculates $[\![m'_a]\!] = [\![m_a]\!]^2 \cdot [\![1]\!] = [\![2m_a + 1]\!]$, $[\![m'_b]\!] = [\![m_b]\!]^2 \cdot [\![1]\!] = [\![2m_b + 1]\!]$, and selects two random numbers $r_1, r_2 \leftarrow \mathbb{Z}_N$ ($r_2 \ll r_1$), then runs $[\![res]\!] \leftarrow ([\![m'_a]\!] \cdot [\![m'_b]\!]^{N-1})^{r_1} \cdot [\![r_2]\!]$. Finally, CP obtains the $[\![res]\!]_1 \leftarrow$ **SDec**($[\![res]\!]$) with λ_1 before sending $[\![res]\!]$ and its decryption share to CSP.
- Involving the **SDec** and **WDec** algorithms, CSP obtains res. After computing as Eq. 3, CSP sends $[\![res]\!]$ with the encryption.
- CP obtains the final comparison result according to the value of res, if $res = 0$, $m_a \geq m_b$; otherwise, $m_a < m_b$.

$$res = \begin{cases} 1, & |res| > |N|/2 \\ 0, & otherwise \end{cases} \tag{3}$$

Secure Division: Based on the above secure computation, we implement the division operation $\frac{[\![m_a]\!]}{[\![m_b]\!]}$. Due to $[\![\frac{m_a}{m_b}]\!] = [\![m_a]\!] \times [\![m_b^{-1}]\!]$, we use the Newton-Raphson method [43, 44] to approximate the inverse of $[\![m_b]\!]$. Let the function be $f(x) = x^{-1} - m_b$, then iterative computation (see Eq. 4) is used to find the roots of $f(x)$. Assume that x_0 is the approximate root, i.e., $f'(x_0) = 0$ and $f(x_0) + f'(x_0) * (x - x_0) = 0$, where $f'(x)$ is the derivative of $f(x)$. The iteration stops when the difference value is less than the threshold value, i.e., $x_{i+1} - x_i < threshold$. Then, we obtain the x_i as the approximate result of m_b^{-1}.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i * (2 - m_b * x_i). \tag{4}$$

Note that the initial value x_0 required to be a small number, we expand the small number to an integer with the scale factor ε. Moveover, the whole process also involves the secure subtraction, multiplication and comparison over ciphertexts. Once obtaining $[\![m_b^{-1}]\!]$, we compute $[\![m_a]\!] \times [\![m_b^{-1}]\!]$. Owing to the involvement of secure multiplication, the multiplication result has been scaled down. Hence, the final division result does not need to keep the scale factor from expanding.

Secure Argmax: On the basis of secure comparison, given a set of encrypted data, Argmax returns the index of the largest value of the ciphertexts set. For a easy presentation, we denote $index_{max} \leftarrow Argmax(\{[\![m_1]\!], [\![m_2]\!], ...\})$. The specific process is shown in [29].

Secure Transformation: Secure transformation is used to transform the ciphertext under different authorized public keys into a single general authorized public key. Given a ciphertext $[\![m]\!]_{pk_1}$ under the authorized public key pk_1, output $[\![m]\!]_{pk_2}$ under another authorized public key pk_2, i.e., $[\![m]\!]_{pk_2} \leftarrow transfrom([\![m]\!]_{pk_1})$. Note that CP owns a secret share λ_{11} and CSP owns the other secret share λ_{12}, where λ_{11} and λ_{12} are the secret shares of sk_1. The specific process is shown as follows:

- CP encrypts a random number $r \leftarrow \mathbb{Z}_N$ with the authorized public key pk_1, calculates $[\![m']\!]_{pk_1} = [\![m]\!]_{pk_1} \cdot [\![r]\!]_{pk_1}$, and sends $[\![m']\!]_{pk_1}$ and its decryption share computed with λ_{11} by calling **SDec** to CSP.
- Then, CSP obtains m' with λ_{12} by calling **SDec** and **WDec** algorithms, and encrypts m' with the authorized public key pk_2 before sending $[\![m']\!]_{pk_2}$ to CP.
- Finally, CP gets the final result by $[\![m]\!]_{pk_2} = [\![m']\!]_{pk_2} - [\![r]\!]_{pk_2}$.

Fig. 5. The detailed presentations of secure computation.

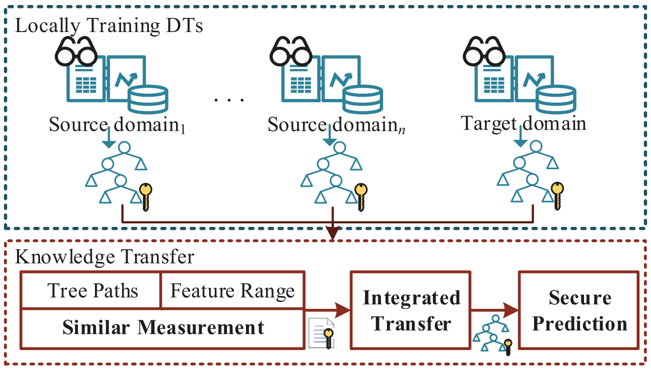


Fig. 6. Overview of PMKT framework.

6.2. Locally training decision trees

With the CART decision tree algorithm, multi-source domains pre-train their individual DTs over local datasets. As the training process runs over plaintexts, it is extremely efficient and real-time during the pre-training process. As shown in Fig. 2, on the pre-trained DT, each tree node contains parameters (e.g., splitting value, splitting feature index, right node and left node). We encrypt a DT by gradually encrypting the tree nodes from top to down. Then, each source domain shares the encrypted DT model with CP.

In addition, for further knowledge transfer, each source domain needs to upload his/her feature range of training dataset as auxiliary information. For privacy preservation, both the auxiliary information encrypted as $[\![dom]\!]_{pk_i}$ and the DT model from the

ith source domain are sent to CP. Therefore, it is required to transform ciphertexts under different public keys pk_i into the same public key pk by using secure transformation.

6.3. Integrated knowledge transfer

In this section, we show how to achieve the integrated knowledge transfer in PMKT. The involved processes are shown in follows.

6.3.1. Privacy-preserving similarity measurement

We perform the privacy-preserving similarity measurement to compute the similarity between two DTs (i.e., one from a source domain and the other from a target domain). To achieve this goal, building tree paths, calculating feature range and label probability on an encrypted DT are the fundamental works for the similarity measurement, which are greatly significant for measurement results. The specific process is shown as follows.

Algorithm 1: TreePaths

```
Input: $node$ as the root node of the encrypted DT model
Output: the set of tree paths $paths = \{nodes_1, nodes_2, ...\}$
if ($node$ equal to null) then
    return $paths$;
leftPaths=$node$+TreePaths($node$.leftChild);
rightPaths=$node$+TreePaths($node$.rightChild);
$paths$.add(leftPaths);
$paths$.add(rightPaths);
if ($paths.size$ equal to 0) then
    $paths$.add($node$);
return $paths$.
```

- *The tree paths in an encrypted DT*: The tree paths on an encrypted DT are converted into the representation of branch sequences, each tree path starts with the root node and ends with a leaf node. Taking Fig. 2 as an example, there is a total of four tree paths: $path_1 = \{f_1, f_2, c_1\}$, $path_2 = \{f_1, f_2, c_2\}$, $path_3 = \{f_1, f_3, c_1\}$, and $path_4 = \{f_1, f_3, c_2\}$. The root node of each tree path is the same, but the leaf node of each path corresponds a different label c_i. Here, we give a brief description about securely obtaining tree paths by traversing an encrypted DT in Algorithm 1. Input the root node of an encrypted DT, which contains the leftChild, rightChild, feature ID (i.e., *fId*), and value. The algorithm recursively executes from root node to the leaf nodes. To represent the tree paths, a set of node arrays (i.e., $paths = \{nodes_1, nodes_2, \ldots\}$) is denoted, where the ith tree path is expressed as a node array $nodes_i$.
- *The feature range on the encrypted DT*: Since each tree path on a DT contains a label that reveals worthwhile information in identifying similarity, the paths are grouped by their corresponding labels. Besides, owing to the auxiliary information $[\![dom]\!]$ contains the value range of each feature, we can obtain the feature range $[\![ran]\!]$ of each feature for different label c_i on an encrypted DT in the form of ciphertext with the aid of $[\![dom]\!]$. The details are shown in Algorithm 2. Through comparing the value of each node on the tree path with the current feature range, the feature range $[\![ran]\!]$ is updated. As all data involved in the process are ciphertexts, secure comparison in Section 5.3 is called.
- *The structural probability on the encrypted DT*: Without accessing to the original dataset, we can get the structural probability with the auxiliary information. Combined Eq. (1), a structural probability $[\![SP_{dt}(c_i)]\!]$ is figured out (– See (2)). As the $|ran_{f_k}(c_i)|$ and $|dom_{f_k}|$ are encrypted data, the secure addition, secure division, secure multiplication in Section 5.3 are involved in the process.

Algorithm 2: FeatureRange

```
Input: The auxiliary information $[\![dom]\!]$ and the tree paths $paths$
       of the encrypted DT model $dt$
Output: Feature range $[\![ran]\!]$
forall the $path_i \in paths$ do
    $c_i = path_i.lable$;
    forall the $node_j \in path_i$ do
        $k = node_j.fId$; //get the feature id
        $[\![max_{f_k}(c_i)]\!] = [\![dom]\!].max_{f_k}$; //max value of $a_{id}$
        $[\![min_{f_k}(c_i)]\!] = [\![dom]\!].min_{f_k}$; //min value of $a_{id}$
        if $node_{j+1}$ equal to $node_j.leftChild$ then
            if $Comparison([\![max_{f_k}(c_i)]\!], [\![node.value]\!]) == 1$ then
                $[\![max_{f_k}(c_i)]\!] = [\![node.value]\!]$;
        else
            if $Comparison([\![min_{f_k}(c_i)]\!], [\![node.value]\!]) == 0$ then
                $[\![min_{f_k}(c_i)]\!] = [\![node.value]\!]$;
        $[\![ran(c_i)]\!].put([\![min_{f_k}(c_i)]\!], [\![max_{f_k}(c_i)]\!])$;
return $[\![ran]\!]$.
```

Similarity measurement between two encrypted DTs is computed as shown in Eq. (5), where $[\![SP_{dt_1}(c_i)]\!]$ and $[\![SP_{dt_2}(c_i)]\!]$ respectively denote the DT structural probability of dt_1 and dt_2 for the label c_i. To provide strong privacy guarantees, Eq. (5) runs over ciphertexts.

$$[\![Sim(dt_1, dt_2)]\!] = 1 - \prod_{i=1}^{2} \left| [\![SP_{dt_1}(c_i)]\!] - [\![SP_{dt_2}(c_i)]\!] \right|. \tag{5}$$

With these above steps, we can estimate the similarity between the encrypted dt_i ($i = 1$ to n) belonging to a source domain and dt_t belonging to the target domain in a privacy-preserving way. Finally, we obtain an encrypted set $\{[\![Sim(dt_1, dt_t)]\!], \ldots, [\![Sim(dt_n, dt_t)]\!]\}$, which represents the similarity between each source domain and the target domain.

6.3.2. Privacy-preserving integrated knowledge transfer

Consecutively, we assign transfer weights to these encrypted DTs belonging to source domains, where each DT belonging to source domains is assigned a transfer weight based on the similarity between dt_i ($i = 1$ to n) from source domains and dt_t from the target domain. In Algorithm 3, the weight normalization is implemented so that a weight w_i belonging to dt_i satisfies $\sum_{i=1}^{n} w_i = 1$, where all weights is in the ciphertext format. Since all weights and similarities are ciphertexts for privacy preservation, secure addition and secure division in Section 5.3 are involved.

Based on the weighted ensemble learning method [43], we aim at implementing the integrated knowledge transfer with these DTs from multi-source domains. The detail process is shown as follows.

- First, we assign transfer weights to each prediction of dt_i ($i = 1$ to n) from source domains, according to the transfer weight of each DT. The forecasting result of a dt_i is represented as $h_{dt_i}(x) = c_i$,
- Then, the weighted forecasting result of the integrated knowledge transfer represented as $H(x)$ is generated, as shown in Eq. (6),

$$H(x) = c_{Argmax \sum_{i=1}^{n} (w_i \cdot h_{dt_i}(x))}, \tag{6}$$

where the secure argmax operation is required.

Algorithm 3: Privacy-preserving Weight Normalization

```
Input: The encrypted similarity set of DT models
       $\{[\![Sim(dt_1, dt_t)]\!], ..., [\![Sim(dt_n, dt_t)]\!]\}$
Output: The encrypted transfer weight array
       $W = \{[\![w_1]\!], ..., [\![w_n]\!]\}$
1 $[\![sum]\!] = \sum_{i=1}^{n} [\![Sim(dt_i, dt_t)]\!]$;
2 for $1 \le i \le n$ do
3     $W[i] = \frac{[\![Sim(dt_i, dt_t)]\!]}{[\![sum]\!]}$ // secure division;
4 return $W$.
```

6.4. Secure classification on-the-fly

- An authorized user submits his/her request $[\![req]\!]_{pk_u}$ after encrypting it with his/her public key pk_u. On receiving an encrypted request $[\![req]\!]_{pk_u}$, each dt_i ($i = 1$ to n), which comprises of the integrated knowledge transfer model, will output the encrypted classification result $h_{dt_i}([\![req]\!])$ for the encrypted request. The specific process of predicting the forecasting result $[\![req]\!]$ of a DT is shown in Algorithm 4. However, as the dt_i runs over the ciphertext domain under the public key pk, $[\![req]\!]_{pk_u}$ needs to be transformed into $[\![req]\!]$ under the public key pk. Hence, the secure transform is involved to implement $[\![req]\!] \leftarrow transform([\![req]\!]_{pk_u})$.
- The final forecasting result of the integrated knowledge transfer is represented as $[\![res]\!]$, and the forecasting result of a dt_i is represented as $h_{dt_i}([\![req]\!])$. Then, $[\![res]\!] = H([\![req]\!]) \leftarrow c_{Argmax \sum_{i=1}^{n}(w_i \cdot h_{dt_i}([\![req]\!]))}$ with secure Argmax.
- Before returning the final forecasting result to the user, the secure transformation is required to be operated $[\![res]\!]_{pk_u} \leftarrow transform([\![res]\!])$. After transforming the ciphertexts under public key pk domain to public key pk_u domain, $[\![res]\!]$ is returned to the user. Then, the user can obtain the final forecasting result by decrypting it with his/her secret key sk_u.

Algorithm 4: Classification

```
Input: The root node of an encrypted DT, the user-submitted
       encrypted request $[\![req]\!]$
Output: The encrypted result $h_{dt_i}([\![req]\!])$
1 if node is a leaf node then
2     return node.label;
3 $f_k = node.fId$;
4 if $Comparison([\![req]\!]_{f_k}.value, [\![node.value]\!]) == 1$ then
5     return classification(node.leftChild, $[\![req]\!]$);
6 else
7     return classification(node.rightChild, $[\![req]\!]$);
8 return $h_{dt_i}([\![req]\!])$.
```

7. Security analysis

In this section, we make security analysis on our PMKT framework to demonstrate that PMKT can achieve the problem formulations in Section 4.3, which is based on semantic security of the proposed public-key cryptosystem evaluated in [20,40] and assumptions that there are non-colluding semi-honest adversaries. Specifically, we define the *real vs. ideal* model to formalize the privacy in PMKT. The model involves challengers (i.e., an entity with a secret key), source domains, target domain and users (a.k.a., $\mathcal{S}_0$), CP (a.k.a., $\mathcal{S}_1$) and CSP (a.k.a., $\mathcal{S}_2$), as well as adversaries $\mathcal{A} = (\mathcal{A}_{S_0}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ and simulators (i.e., $Simulator_{S_0}$, $Simulator_{S_1}$, $Simulator_{S_2}$).

Normally, in the real world, we assume that an adversary $\mathcal{A}$ interacts with a challenger, note that $\mathcal{A}$ also has interactions with a simulator $\mathcal{S}$ in the ideal world. Besides, we consider that PMKT is secure when the view of the adversary in the real world is indistinguishable from its view in the ideal world. And we define T as the required knowledge of $\mathcal{A}$ when the view of the adversary in the real world can distinguish from its view in the ideal world, and define $Adv_{\mathcal{A}}(\cdot)$ to represent whether $\mathcal{A}$ obtains the knowledge.

7.1. Security of computation

Based on the semantic security of the public-key cryptosystem and the assumption that $\mathcal{S}_1$ and $\mathcal{S}_2$ are non-colluding cloud servers, we demonstrate the security of computation over ciphertexts described in Section 5.3.

Theorem 1. *The process of secure scaling-down is secure against semi-honest adversaries with the semantic security of the public-key cryptosystem stated in Section 3.3, and also prevents the adversaries from distinguishing the intermediate computational results with the presence of non-colluding semi-honest adversaries $\mathcal{A} = (\mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. We separately analyze the security of each phase in secure scaling-down with the above *real vs. ideal* model. We build simulators $Simulators = (Simulator_{S_1}, Simulator_{S_2})$ in the ideal world to simulate the view of $\mathcal{A} = (\mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ in the real world.

- During the process of secure scaling-down, two semi-honest cloud serves (i.e., CP and CSP) interactively complete the computation. We provide a proof that the process is secure against both adversaries $\mathcal{A}_{S_1}$ and $\mathcal{A}_{S_2}$ in the real world. Given an encrypted data $[\![m]\!]$, $Simulator_{S_1}$ randomly chooses r and computes $r' = r * 10^{\varepsilon}$, and calculates $[\![r']\!] \leftarrow$ **Enc**(r'), $[\![M]\!] = [\![r']\!] \cdot [\![m]\!]$, then **SDec** algorithm is used to obtain the decryption share $[\![M]\!]_1$. Finally, $Simulator_{S_1}$ returns the encrypted data $[\![m]\!]$ and decryption share $[\![M]\!]_1$ to $\mathcal{A}_{S_1}$. While for $\mathcal{A}_{S_1}$, it contains the ciphertext $[\![m']\!]$ and decryption share $[\![M']\!]_1$. As the public-key scheme is semantic security, it is obvious that if $\mathcal{A}_{S_1}$ can distinguish $[\![m]\!]$ and $[\![m']\!]$ and these decryption shares, then $\mathcal{A}_{S_1}$ needs to obtain secret key sk or the other secret share λ_2. The views of $\mathcal{A}_{S_1}$ cannot distinguish the real word from the ideal world without the corresponding secret key, which is demonstrated as

$$\begin{aligned} &Adv_{\mathcal{A}_{S_1}}(T) = Adv_{\mathcal{A}_{S_1}}([\![m]\!], sk, [\![M]\!]_1, \lambda_2) \\ &\Rightarrow Adv_{\mathcal{A}_{S_1}}([\![m]\!], sk) \vee Adv_{\mathcal{A}_{S_1}}([\![M]\!]_1, \lambda_2) \\ &\Rightarrow (Adv_{\mathcal{A}_{S_1}}([\![m]\!]) \wedge Adv_{\mathcal{A}_{S_1}}(sk)) \vee \\ &\quad (Adv_{\mathcal{A}_{S_1}}([\![M]\!]_1) \wedge Adv_{\mathcal{A}_{S_1}}(\lambda_2)) \\ &\Rightarrow (True \wedge False) \vee (True \wedge False) \Rightarrow False. \end{aligned}$$

- $Simulator_{S_2}$ runs **SDec** and **WDec** algorithms to obtain the plaintext M which is blinded by a random number. It is clear that M is randomly distributed as the random number r is randomly generated. $Simulator_{S_2}$ simulates $\mathcal{A}_{S_2}$ as the above operations. Owing to the blinding technique, if $\mathcal{A}_{S_2}$ can distinguish M, then $\mathcal{A}_{S_2}$ must need to obtain the corresponding random number r, which is demonstrated as

$$\begin{aligned} Adv_{\mathcal{A}_{S_2}}(T) &= Adv_{\mathcal{A}_{S_2}}(M, r) \Rightarrow Adv_{\mathcal{A}_{S_1}}(M) \wedge Adv_{\mathcal{A}_{S_2}}(r) \\ &\Rightarrow True \wedge False \Rightarrow False, \end{aligned}$$

where $\mathcal{A}_{S_2}$ cannot obtain r which is stored in CP. Therefore, the views of $\mathcal{A}_{S_1}$ is unfeasible to distinguish the real world from the ideal world.

Therefore, both CP and CSP learn no sensitive information about the private data and the immediate computational results, the process of secure scaling-down is well-designed to protect privacy. The security proofs of secure multiplication, secure comparison and secure division are similar to that of secure scaling-down under the semi-honest adversaries $\mathcal{A} = (\mathcal{A}_{S_1}, \mathcal{A}_{S_2})$. Next, we illustrate the security of secure transformation. □

Theorem 2. *The process of secure transformation is secure against semi-honest adversaries with the semantic security of the public-key cryptosystem, and also against adversaries to distinguish the intermediate computational results with non-collusion is occurred among semi-honest adversaries* $\mathcal{A} = (\mathcal{A}_{S_0}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. We provide security proof in secure transformation with the above *real vs. ideal* model. We build simulators $Simulators = (Simulator_{S_0}, Simulator_{S_1}, Simulator_{S_2})$ in the ideal world to simulate the view of $\mathcal{A} = (\mathcal{A}_{S_0}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ in the real world, which is similar to the proof of Theorem 1. The only difference is the involvement of $Simulator_{S_0}$ and $\mathcal{A}_{S_0}$.

- $Simulator_{S_0}$ receives the request *req* as the input and simulates $\mathcal{A}_{S_0}$ as follows: $Simulator_{S_0}$ encrypts *req* with the authorized public key pk_1 as $[\![req]\!]_{pk_1} \leftarrow \mathbf{Enc}(req)$, then it returns $[\![req]\!]_{pk_1}$ to $\mathcal{A}_{S_0}$. Due to the semantic security of the public-key cryptosystem, if $\mathcal{A}_{S_0}$ can distinguish the real word from the ideal world, then $\mathcal{A}_{S_0}$ must obtain the corresponding secret key sk_1. $\mathcal{A}_{S_0}$ has no advantage in distinguishing the views from the real world and the ideal world, which is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}_{S_0}}(T) &= Adv_{\mathcal{A}_{S_0}}([\![req]\!]_{pk_1}, sk_1) \\ &\Rightarrow Adv_{\mathcal{A}_{S_1}}([\![req]\!]_{pk_1}) \wedge Adv_{\mathcal{A}_{S_2}}(sk_1) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$
- $Simulator_{S_1}$ simulates $\mathcal{A}_{S_1}$ as follows: $Simulator_{S_1}$ blinds a random number to the encrypted *req*, and calculates decryption share with **SDec** algorithm, then $Simulator_{S_1}$ returns blinded encrypted $[\![req']\!]_{pk_1}$ and its decryption share to $\mathcal{A}_{S_1}$. Since $\mathcal{A}_{S_1}$ has no knowledge of the secret key to obtain the plaintext, with the semantic security game of the public-key cryptosystem, $\mathcal{A}_{S_1}$ cannot obtain the secret key sk_1, which is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}_{S_1}}(T) &= Adv_{\mathcal{A}_{S_1}}([\![req']\!]_{pk_1}, sk_1) \\ &\Rightarrow Adv_{\mathcal{A}_{S_1}}([\![req']\!]_{pk_1}) \wedge Adv_{\mathcal{A}_{S_1}}(sk_1) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$
Therefore, it is hard to distinguish the ciphertexts between the real world and ideal world for $\mathcal{A}_{S_1}$.
- $Simulator_{S_2}$ simulates $\mathcal{A}_{S_2}$ as follows: $Simulator_{S_2}$ obtains the decrypted req' with **SDec** and **WDec** algorithms, and req' is blinded and randomly distributed since the decrypted req' includes the random number which is randomly selected. Owing to the blinding technique, $\mathcal{A}_{S_2}$ cannot obtain the random number r stored in CP, which is blinded to req', the specific process is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}_{S_2}}(T) &= Adv_{\mathcal{A}_{S_2}}(req', r) \\ &\Rightarrow Adv_{\mathcal{A}_{S_2}}(req') \wedge Adv_{\mathcal{A}_{S_2}}(r) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$
Therefore, $\mathcal{A}_{S_2}$ is computationally indistinguishable from the ideal world and the real world.

Obviously, the authorized users, CP and CSP in the process cannot learn any information from the encrypted intermediate results and blinded random numbers. Based on the above secure analysis, we show that no sensitive information about privacy data is disclosed. Hence, we conclude that the secure computation is well-designed to implement calculations over ciphertexts while providing strong privacy preservation. In the following subsection, we demonstrate the security of our PMKT. □

7.2. Security of PMKT framework

In this section, we demonstrate that our PMKT is secure under an active adversary $\mathcal{A}^*$ defined in Section 4.2.

Theorem 3. *PMKT framework can implement privacy-preserving integrated knowledge transfer against the active adversary* $\mathcal{A}^*$.

Proof. We provide a security proof in privacy-preserving integrated knowledge transfer with the adversary $Adv(T)$. □

- If $\mathcal{A}^*$ eavesdrops on the communication links between the source/target domain and the CP, then the original encrypted model and auxiliary data $[\![m]\!]$ will be acquired by $\mathcal{A}^*$. Owing to the semantic security, if $\mathcal{A}^*$ can learn these data, $\mathcal{A}^*$ must hold the corresponding secret key sk to decrypt these ciphertexts. $\mathcal{A}^*$ cannot learn these data without the corresponding secret key sk to decrypt these ciphertexts, which is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}^*}(T) &= Adv_{\mathcal{A}^*}([\![m]\!], sk) \\ &\Rightarrow Adv_{\mathcal{A}^*}([\![m]\!]) \wedge Adv_{\mathcal{A}^*}(sk) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$
- If $\mathcal{A}^*$ eavesdrops CSP, then the original intermediate results M in the plaintext form will be acquired by $\mathcal{A}^*$. Since these intermediate results are randomly distributed as they are blinded random numbers, if $\mathcal{A}^*$ can learn these random numbers r, then $\mathcal{A}^*$ can know the information of plaintexts. However, $\mathcal{A}^*$ cannot know the information of plaintexts as these random numbers r stored in CP, which is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}^*}(T) &= Adv_{\mathcal{A}^*}(M, r) \\ &\Rightarrow Adv_{\mathcal{A}^*}(M) \wedge Adv_{\mathcal{A}^*}(r) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$
- If $\mathcal{A}^*$ compromises the CP or CSP to acquire the secret share (i.e., λ_1 or λ_2), it is impossible to recover the secret key, as the secret key is randomly split by executing **KeyS** algorithm.

Theorem 4. *PMKT framework can securely achieve real-time classification on the integrated knowledge transfer with the presence of the active adversary* $\mathcal{A}^*$.

Proof. The specific process is similar to Theorem 3.

- When $\mathcal{A}^*$ compromises the transmission between the user and the CP, then the user's encrypted requests $[\![req]\!]_{pk_U}$ could be obtained by $\mathcal{A}^*$. Due to the semantic security, if $\mathcal{A}^*$ can learn these encrypted requests $[\![req]\!]_{pk_U}$, then $\mathcal{A}^*$ must obtain the corresponding secret key sk_U. $\mathcal{A}^*$ cannot learn these requests without the corresponding secret key sk_U to decrypt these ciphertexts, which is demonstrated as
$$\begin{aligned} Adv_{\mathcal{A}^*}(T) &= Adv_{\mathcal{A}^*}([\![req]\!]_{pk_U}, sk_U) \\ &\Rightarrow Adv_{\mathcal{A}^*}([\![req]\!]_{pk_U}) \wedge Adv_{\mathcal{A}^*}(sk_U) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$

- If $\mathcal{A}^*$ eavesdrops CSP, all intermediates M_U involved in secure transformation are blinded random numbers. Due to the blinding technique, $\mathcal{A}^*$ can acquire information of plaintexts with these blinding random numbers r_U. $\mathcal{A}^*$ cannot learn the real information of plaintexts as these blinding random numbers r_U are stored in CP, which is demonstrated as

$$\begin{aligned} Adv_{\mathcal{A}^*}(T) &= Adv_{\mathcal{A}^*}(M_U, r_U) \\ &\Rightarrow Adv_{\mathcal{A}^*}(M_U) \wedge Adv_{\mathcal{A}^*}(r_U) \\ &\Rightarrow True \wedge False \Rightarrow False. \end{aligned}$$

- The final encrypted forecasting results on the integrated knowledge transfer are transmitted to the user, which may also be available to $\mathcal{A}^*$. However, even $\mathcal{A}^*$ obtains secret keys from other users, $\mathcal{A}^*$ is still not able to decrypt the encrypted result returned by the CP, as the different user's secret key in our system is unrelated and independently.

Therefore, we conclude that PMKT framework can resist the threats defined in Section 4.2. □

8. Performance analysis

In this section, we first implement the experiment setup in Section 8.1. Then, the correctness and misclassification rate evaluation, computation overhead and communication overhead will be evaluated in Section 8.2. Besides, the real-world financial dataset is used to evaluate the performance of the PMKT.

8.1. Experiment analysis

The PMKT is evaluated by using Java. The experiments are evaluated on our PC testbed (3.30 GHz four-cores processors and 4 GB memory).

8.1.1. Performance of secure computation

To evaluate the performance of secure computation, we execute these ciphertext calculations with the variation of crypto parameter N. As the secure Argmax is designed on secure comparison, the main impact on its performance is the size of a ciphertext set, which decides the times of calling secure comparison. Hence, we do not consider secure Argmax. Here, we evaluate secure multiplication, comparison, division, scaling down, and transformation by executing each algorithm 100 times. When the bit length of N is 1024, it takes 201.03 ms to run secure multiplication, 61.23 ms to run secure comparison, 1139.08 ms to run secure division, 70.63 ms to execute secure scaling-down and 69.67 ms to execute secure transformation. Besides, the communication overhead, the computational cost for CP & CSP, and sum running time are analyzed, — see Fig. 7(a)–(e). We discover that both computational cost and communication overhead of these secure computation increase with the bit length of N. Owing to more encrypted data are needed to be processed with the increase of N, it is required more computation and communication resources.

Since secure multiplication and secure scaling-down in [44] based on Paillier cryptosystem, we focus on the contrast of the performance of secure multiplication, secure scaling-down between our system and [44], — see Fig. 7(f)(g). We observe that the running time of both secure multiplication and secure scaling-down in our PMKT system bring the significant improvement than those of [44].

Table 3
Source domains and target domain.

Domains	$\mathcal{D}_{S1}$	$\mathcal{D}_{S2}$	$\mathcal{D}_{S3}$	$\mathcal{D}_{S4}$	$\mathcal{D}_{S5}$	$\mathcal{D}_{S6}$	$\mathcal{D}_{S7}$	$\mathcal{D}_{S8}$	$\mathcal{D}_{S9}$	$\mathcal{D}_{\mathcal{T}}$
Accuracy	0.88	0.93	0.84	0.85	0.81	0.85	0.81	0.96	0.84	0.62
Direct transfer	0.71	0.78	0.79	0.73	0.71	0.71	0.71	0.72	0.73	0.62

Table 4
Performance metrics comparison.

Method	TPR	TNR	Accuracy
PMKT	94.9%	79.6%	82.3%
Normal	96.6%	80.0%	82.9%

Notes. Normal: the integrated knowledge transfer over plaintexts.

8.1.2. Performance of integrated knowledge transfer and classification

To evaluate the effectiveness of PMKT framework, we assume N to be 1024 bits to realize 80-bit security level.[2] Considering both the accuracy and efficiency, we assume the public factor to be $\varepsilon = 3$.

We perform the experiments on the real financial market dataset called Bank Marketing Dataset[3] from the UCI machine learning repository. The dataset is related with the forecasting goal that if the user will subscribe to a product. Each instance contains 20 features. In the experiment, we divide the dataset into 10 subsets according to the kind of jobs belonging to each instance, namely "$\mathcal{D}_{S1}$", "$\mathcal{D}_{S2}$", ..., "$\mathcal{D}_{S9}$" and "$\mathcal{D}_{\mathcal{T}}$". Note that we choose the "$\mathcal{D}_{\mathcal{T}}$" as the target domain, and other nine subsets "$\mathcal{D}_{Si}$" ($i = 1$ to 9) as source domains.

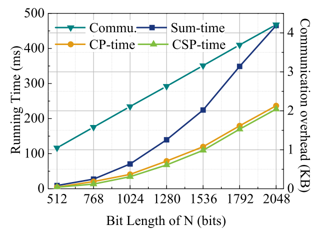
We first pre-train DTs over $\mathcal{D}_{Si}$ ($i = 1$ to 9) and $\mathcal{D}_{\mathcal{T}}$. To evaluate the accuracy of DT models, we use 10-fold cross-validation. The accuracy of these pre-trained DTs is shown in Table 3. See Fig. 7(h), when these pre-trained DTs from $\mathcal{D}_{Si}$ ($i = 1$ to 9) are directly transferred to the target domain $\mathcal{D}_{\mathcal{T}}$ without using PMKT framework, compared the original accuracy of pre-trained DTs with the accuracy of direct transfer, we discover that direct transfer has a significant accuracy loss of each pre-trained DT. In addition, each source domain encrypts his/her pre-trained DT before uploading to the cloud server in PMKT. It averagely costs 922.8 ms to pre-train a DT over each source domain and 32.569 s to encrypt a DT. Besides, it spends 200 bits to transmit an encrypted DT.

For privacy-preserving similarity measurement, it costs 3718s to measure the similarity between each encrypted pre-trained DT from the above nine source domains $\mathcal{D}_{Si}$ ($i = 1$ to 9) and the encrypted pre-trained DT from the target domain $\mathcal{D}_{\mathcal{T}}$. For instance forecasting, it takes 3.57 s for an encrypted request from the target domain. Moreover, as shown in Table 4, we also test the performance of PMKT compared with the performance over plaintexts. We introduce True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN) to analyze the performance of PMKT, where TP indicates the number of correctly classified abnormal instances, FP indicates the number of misclassified normal instances, FN indicates the number of misclassified abnormal instances, TN indicates the number of correctly classifier normal instances. True Positive Rate (TPR), True Negative Rate (TNR) and accuracy are three important performance metrics as computed in

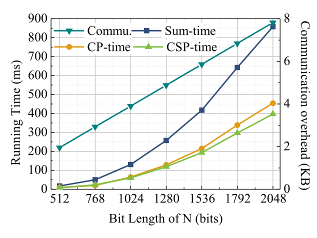
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$
$$TPR = \frac{TP}{TP + FN}, \quad TNR = \frac{FP}{FP + TN}.$$

[2] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

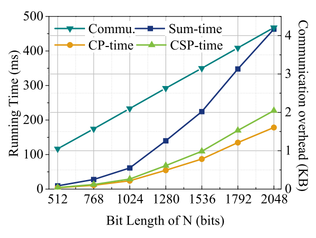
[3] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing.



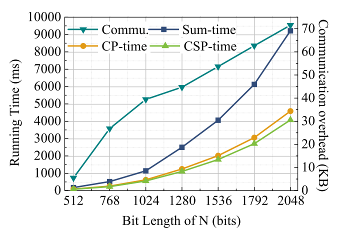
(a) Secure scaling-down.



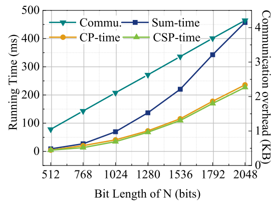
(b) Secure multiplication.



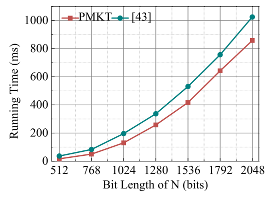
(c) Secure comparison.



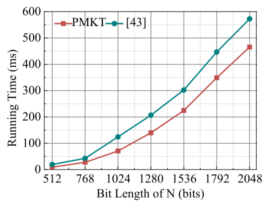
(d) Secure division.



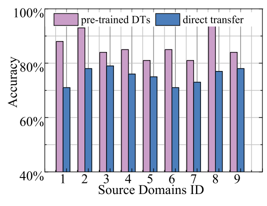
(e) Secure transformation.



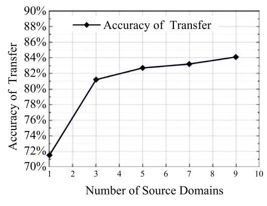
(f) Comparison of secure multiplication.



(g) Comparison of secure scaling-down.



(h) Accuracy comparison.



(i) Accuracy of integrated knowledge transfer.

Fig. 7. Performance of PMKT.

We observe that those performance metrics between PMKT and integrated knowledge transfer over plaintexts are similar. TPR, TNR and accuracy have dropped slightly in PMKT, and the accuracy of PMKT is 82.3% compared with the accuracy over plaintexts (82.9%), the accuracy of our PMKT has a decrease of 0.6%. The reason of decrease is that real-world datasets in decimal value is required to convert into integers before the encryption operation, both the convert operation and encryption operation have a slight impact upon the forecasting accuracy. As demonstrated in Fig. 7(i), we observe that the accuracy of the integrated knowledge transfer impacted by the number of source domains, the variation of forecasting accuracy increases with varying the number of source domains.

8.2. Theoretical analysis

To evaluate the effectiveness of PMKT framework, we analyze the correctness and misclassification rate of our framework. Besides, the computational overhead and communication overhead of our PMKT are shown in following.

8.2.1. Correctness of similarity measurement

The correctness of similarity measurement is verified as follows. Given two related but different DTs (i.e., dt_1 and dt_2), the similarity measurement between these two DTs is represented as $Sim(dt_1, dt_2)$, where $SP_{dt}(c_i)$ reveals the relative importance over feature space on different class labels which has a strong connection with the $dt's$ tree structure and follows the Eq. (7). Since $Sim(dt_1, dt_2)$ relies on distance measurement of the prediction probabilities on each label, combined with Eqs. (2) and (7), more similar the prediction probabilities between two DTs are, the closer $Sim(dt_1, dt_2)$ is to 1. If and only if $SP_{dt_1}(c_i) = SP_{dt_2}(c_i)$ that $Sim(dt_1, dt_2) = 1$, where $c_i \in C$.

$$\sum_{i=1}^{C} SP_{dt}(c_i) = 1. \tag{7}$$

8.2.2. Misclassification analysis of similar measurement

We first make the misclassification analysis of a single DT (i.e., dt_i), which is from a source domain. Then, on this basis, we make misclassification analysis of the integrated knowledge transfer.

Given an instance x, $P(c_k|x)$ which denotes an ideal posteriori probability when x belongs to the label c_k, $e_{c_k}^{dt_i}(x)$ denotes the prediction error on the DT dt_i, the estimated probability is $g_{dt_i}(c_k|x)$ on condition that x belongs to the label c_k, which is computed as Eq. (8).

$$g_{dt_i}(c_k|x) = P(c_k|x) + e_{c_k}^{dt_i}(x). \tag{8}$$

Owing to the existence of prediction error $e_{c_k}^{dt_i}(x)$, the actual classification boundary l deviates from the ideal classification boundary l^*, we denote the bias as $b = l - l^*$. Therefore, there is an instance x belonging to label c_i, while a DT misclassifies x to the label c_j. In this case, the probability of misclassification is represented as Eq. (9), where $c_i, c_j \in C$ and $c_i \neq c_j$.

$$E_{dt_i} = \int_{l^*}^{l} \left| P(c_j|x) - P(c_i|x) \right| p(x) d(x). \tag{9}$$

Owing to the high prediction accuracy of a DT from a source domain, the bias b is small. Hence, $P(c_k|x)$ can obtain a linear approximation as shown in Eq. (10), where $p(x) \approx p(l^*)$ and $x \in [l^*, l]$.

$$P(c_k|x) \approx b \cdot P'(c_k|l) + P(c_k|l^*). \tag{10}$$

The probability of misclassification E_{dt_i} is computed as

$$E_{dt_i} = \frac{p(l^*)}{2} \cdot b^2 \cdot \left(P'(c_i|l) - P'(c_j|l) \right). \tag{11}$$

To implement knowledge transfer from multi-source domains to the target domain, the integrated knowledge transfer is employed in PMKT. On the basis of the above analysis, the misclassification analysis of the integrated knowledge transfer is shown as follows. The estimated probability of the integrated knowledge transfer is shown as

$$g_{dts}(c_k|x) = \sum_{i=1}^{n} w_i \cdot g_{dt_i}(c_k|x). \tag{12}$$

The probability of misclassification is represented as Eq. (13), where $\mu = P'(c_i|l_{dts}) - P'(c_j|l_{dts})$. When $x = l_{dts}$, the prediction probability of the label c_i and c_j is the same, i.e., $g_{dts}(c_i|l_{dts}) = g_{dts}(c_j|l_{dts})$. Combined with the linear approximation, we can obtain $b_{dts} = \mu^{-1}[e_{c_i}^{dts}(l_{dts}) - e_{c_j}^{dts}(l_{dts})]$.

$$E_{dts} = \frac{p(l_{dts}^*)}{2} \cdot b_{dts}^2 \cdot \mu. \tag{13}$$

Since the weight of knowledge transfer belonging to multi-source domains depends on the similarity measurement between dt_i from a source domain and dt_t from the target domain, and follows $\sum_1^n w_i = 1$, then w_i can be represented

$$w_i = \left[\sum_{j=1}^{n} Sim(dt_j, dt_t)\right]^{-1} Sim(dt_i, dt_t). \tag{14}$$

Finally, combined with the above deductions, the probability of misclassification on the integrated knowledge transfer E_{dts} is shown in Eq. (15), where $\sigma_{c_i}^{dt_i}$ and $\sigma_{c_j}^{dt_i}$ respectively represent the variance of $e_{c_i}^{dt_i}$ and $e_{c_j}^{dt_i}$.

$$E_{dts} = \frac{p(l^*_{dts})}{2\mu} \cdot \sum_{i=1}^{n} w_i^2 \left[(\sigma_{c_i}^{dt_i})^2 + (\sigma_{c_j}^{dt_i})^2\right], \tag{15}$$

8.2.3. Computational overhead

In PMKT, we make an assumption that one regular exponentiation calculation with an exponent of length $\|N\|$ requires $1.5\|N\|$ multiplications [40]. Taking **Enc** stated in Section 3.3 as an example, the length of m is $\|N\|$, and g^m is required $1.5\|N\|$ multiplications. Besides, compared with addition/subtraction and multiplication calculations, exponential computing has a greatly higher computational cost. Hence, our analysis ignores the repetitive addition/subtraction and multiplication calculations. For the public-key cryptosystem in Section 3.3, to implement the encryption, **Enc** requires $1.5\|N\|$ multiplications. To implement the decryption, **Dec** requires $1.5\|N\|$ multiplications, and **SDec** and **WDec** requires $1.5\|N\|$ multiplications to process.

For the secure computation, it costs $1.5\|N\|$ multiplications to run secure subtraction for the CP while ignoring the secure addition. To execute secure scaling-down, it costs $4.5\|N\|$ multiplications for the CP, and $4.5\|N\|$ multiplications for the CSP. For secure multiplication over ciphertexts, it costs $10.5\|N\|$ multiplications for the CP, and $7.5\|N\|$ multiplications for the CSP. To run the secure comparison, it is required $3\|N\|$ multiplications for the CP, and $4.5\|N\|$ multiplications. We assume that the iteration time is t to implement the secure division. Hence, it costs $(27t + 10.5)\|N\|$ multiplications for the CP, while $(19.5t + 7.5)\|N\|$ multiplications for the CSP. For secure Argmax, we assume that the size of input set is d, it takes $3d\|N\|$ multiplications for the CP, while $4.5d\|N\|$ multiplications for the CSP. To execute the secure transformation, it is required $4.5\|N\|$ multiplications for the CP, and $4.5\|N\|$ multiplications for the CSP.

For the privacy-preserving knowledge transfer, assume that all DTs in our system have the same height θ of the right subtrees and the left subtrees, where each DT have φ tree paths, and the size of feature space is ν. It costs $\mathcal{O}((\theta\varphi + \nu)\|N\|)$ multiplications for the CP and CSP to implement the privacy-preserving similar measurement between the dt_i from the source domain and dt_t from the target domain. Based on the similarity of n DTs from the source domains, it is required $\mathcal{O}(n(\theta\varphi + \nu)\|N\|)$ multiplications for the CP and CSP to construct the integrated knowledge transfer in a privacy-preserving way. For the real-time prediction, it takes $\mathcal{O}(n\theta\|N\|)$ multiplications for the CP and CSP to implement the high-accuracy prediction.

8.2.4. Communication overhead

In PMKT, the encrypted data and its decryption share require $2\|N\|$ bits for transmission. For the secure computation, it requires $4\|N\|$ bits to be transmitted between the CP and CSP to execute secure scaling-down, secure comparison and secure transformation. To implement secure Argmax, it needs $4d\|N\|$ bits transmitted between the CP and CSP. For the secure multiplication, it requires to transmit $12\|N\|$ bits, while secure division needs $(20t + 8)\|N\|$ bits. During the privacy-preserving knowledge transfer, $\mathcal{O}(n\theta\varphi\|N\|)$ bits are transmitted among the CP and multi-source domains, and $\mathcal{O}(n(\theta\varphi + \nu)\|N\|)$ bits transmitted between the CP and CSP. For the secure classification, it takes $\mathcal{O}(n\theta\|N\|)$ bits between the CP and CSP, and $\mathcal{O}(\|N\|)$ bits between the user and the CP to implement the prediction.

Table 5
Comparative summary.

Method	[16]	[7]	[29]	[45]	PMKT
Knowledge transfer	✓	✗	✗	✗	✓
Privacy preservation	✗	✓	✓	✓	✓
Multi-parties	✓	✗	✗	✓	✓
Secure computation	✗	✓	✓	✗	✓
Computation cost	Θ	$\gg \Theta$	–	$\gg \Theta$	Θ

Notes. Computation cost belonging to the training classifier phase, we define $\Theta = O(m * \nu * h)$, where m is the size of samples, ν is the number of features, h is the depth of a DT. [16] and PMKT run over plaintexts, the computation cost is Θ, while [7] and [45] run over ciphertexts, the computation cost is much bigger than Θ.

8.3. Comparative analysis

In this section, we demonstrate that comparative summary among PMKT and other existing privacy-preserving schemes [7,16,29,45]. In [16], Lee et al. designed a knowledge transfer scheme on DTs, which emphasizes the transfer of valuable knowledge across multiple similar and related source domains. As DTs include sensitive information of source domains which may be disclosed to the untrusted parties. One of the worst drawbacks of [16] is that it results in the issue of privacy disclosure. Considering privacy concerns, Bost et al. [7] proposed a privacy-preserving classification using the Naive Bayes, hyperplane decision and DTs to protect the confidentiality of the data and classifiers. Unfortunately, as classifiers are trained over encrypted data, it costs heavy computational overhead in ciphertext operations. To reduce the training cost, Li et al. [29] proposed a privacy-preserving outsourced model sharing framework to implement secure classification service. However, the framework only supports the single-party setting rather than the multi-party setting. To avoid the limitation, Li et al. [45] presented a scheme based on FHE which provides privacy preservation across multi-parties. Due to the multiplicative homomorphic of FHE, it is not requisite to design secure computation for the ciphertext multiplication. However, owing to the heavy computation workload and implementation complexity, it is different to apply the time-consuming fully homomorphic scheme in real-world. Besides, PMKT removes the above constraints, a comparative summary of these schemes is shown in Table 5.

9. Conclusion

In this paper, we focused on achieving privacy preservation of multi-party knowledge transfer for financial market forecasting. Our proposed PMKT system is design to provide strong privacy guarantees for data, classifier models and user-submitted requests in the multi-party setting. Additionally, theoretical analysis and extensive experimental results on the real-world dataset demonstrated that PMKT is efficient and viable in practice. Future research includes extending PMKT to support multi-class classification and other classifiers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Key Program of NSFC (No. U1405255), the Shaanxi Science & Technology Coordination & Innovation Project, China (No. 2016TZC-G-6-3), the National Natural Science Foundation of China (No. 61702404, No. 61702105), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Fundamental Research Funds for the Central Universities, China (No. JB171504, No. JB191506), the National Natural Science Foundation of Shaanxi Province, China (No. 2019JQ-005), Graduate Innovation Foundation of Xidian University, China (No. 20109194858).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, NIPS (2012) 1106–1114.
[2] Y. Miao, J. Weng, X. Liu, K.-K.R. Choo, Z. Liu, H. Li, Enabling verifiable multiple keywords search over encrypted cloud data, Inform. Sci. 465 (2018) 21–37.
[3] A.N. Jahromi, S. Hashemi, A. Dehghantanha, K.-K.R. Choo, H. Karimipour, D.E. Newton, R.M. Parizi, An improved two-hidden-layer extreme learning machine for malware hunting, Comput. Secur. (2019) 101655.
[4] X. Liu, R. Deng, K.-K.R. Choo, Y. Yang, Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes, IEEE Trans. Emerg. Top. Comput. (2019).
[5] G. Jeong, H.Y. Kim, Improving financial trading decisions using deep Q-learning: Predicting the number of Shares, action Strategies, and transfer learning, Expert Syst. Appl. 117 (2019) 125–138.
[6] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Trans. Serv. Comput. (2017) 1–14.
[7] R. Bost, R.A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data, NDSS (2015) 1–14.
[8] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, Practical attribute-based multi-keyword search scheme in mobile crowdsourcing, IEEE Internet Things J. 5 (4) (2018) 3008–3018.
[9] R. Wang, J. Yan, D. Wu, H. Wang, Q. Yang, Knowledge-centric edge computing based on virtualized d2d communication systems, IEEE Commun. Mag. 56 (5) (2018) 32–38.
[10] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, H. Li, Lightweight fine-grained search over encrypted data in fog computing, IEEE Trans. Serv. Comput. (2018) 1–14.
[11] Y. Miao, X. Liu, R.H. Deng, H. Wu, H. Li, J. Li, D. Wu, Hybrid keyword-field search with efficient key management for industrial internet of things, IEEE Trans. Ind. Inf. (2018) 1–14.
[12] A. Farhadi, I. Endres, D. Hoiem, D. Forsyth, Describing objects by their attributes, in: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009, pp. 1778–1785.
[13] C.H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09), 2009, pp. 951–958.
[14] A. Ghandar, Z. Michalewicz, An experimental study of multi-objective evolutionary algorithms for balancing interpretability and accuracy in fuzzy rulebase classifiers for financial prediction, in: Proc. IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr'11), 2011, pp. 1–6.
[15] C.E. Brodley, P.E. Utgoff, Multivariate decision trees, Mach. Learn. 19 (1) (1995) 45–77.
[16] J. won Lee, C. Giraud-Carrier, Transfer learning in decision trees, in: Proc. International Joint Conference on Neural Networks (IJCNN'07), 2007, pp. 726–731.
[17] N.A. Goussies, S. Ubalde, M. Mejail, Transfer learning decision forests for gesture recognition, J. Mach. Learn. Res. 15 (1) (2014) 3667–3690.
[18] B. Piccart, J. Struyf, H. Blockeel, Empirical asymmetric selective transfer in multi-objective decision trees, in: Proc. International Conference on Discovery Science (DS'08), 2008, pp. 64–75.
[19] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, B. Tseng, Boosted multi-task learning, Mach. Learn. 85 (1–2) (2011) 149–173.
[20] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Proc. International Conference Theory and Applications of Cryptographic Techniques (EUROCRYPT'99), 1999, pp. 223–238.
[21] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in: Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'03), 2003, pp. 37–54.
[22] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inform. Theory 31 (4) (1985) 469–472.
[23] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM 21 (2) (1978) 120–126.
[24] D. Boneh, E.-J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in: Proc. Theory of Cryptography Conference (TCC'09), 2005, pp. 325–341.
[25] C. Gentry, D. Boneh, A fully homomorphic encryption scheme, 20(9) 2009, pp. 1–209.
[26] X. Liu, R. Deng, K.-K.R. Choo, Y. Yang, Privacy-preserving outsourced clinical decision support system in the cloud, IEEE Trans. Serv. Comput. (2017) 1–14.
[27] X. Liu, R. Deng, K.-K.R. Choo, Y. Yang, Privacy-preserving outsourced support vector machine design for secure drug discovery, IEEE Trans. Cloud Comput. (2018) 1–14.
[28] S. Qiu, B. Wang, M. Li, J. Liu, Y. Shi, Toward practical privacy-preserving frequent itemset mining on encrypted cloud data, IEEE Trans. Cloud Comput. (1) (2017) 1.
[29] T. Li, Z. Huang, P. Li, Z. Liu, C. Jia, Outsourced privacy-preserving classification service over encrypted data, Netw. Comput. Appl. 106 (2018) 100–110.
[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. International Conference on Artificial Intelligence and Statistics, (AISTATS'17), Vol. 54, 2017, pp. 1273–1282.
[31] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5) (2018) 1333–1345.
[32] S.J. Pan, Q. Yang, et al., A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[33] Z. Ma, J. Ma, Y. Miao, X. Liu, Privacy-preserving and high-accurate outsourced disease predictor on random forest, Inf. Sci. (2019) http://dx.doi.org/10.1016/j.ins.2019.05.025.
[34] S.M. Jasimuddin, C. Connell, J.H. Klein, A decision tree conceptualization of choice of knowledge transfer mechanism: The views of software development specialists in a multinational company, Knowl. Manag. 18 (1) (2014) 194–215.
[35] N. Segev, M. Harel, S. Mannor, K. Crammer, R. El-Yaniv, Learn on source, refine on target: a model transfer learning framework with random forests, IEEE Trans. Pattern Anal. Mach. Intell. 39 (9) (2017) 1811–1824.
[36] J. O'Neill, P. Buitelaar, Few Shot Transfer Learning BetweenWord Relatedness and Similarity Tasks Using A Gated Recurrent Siamese Network, in: Proc. AAAI Conference on Artificial Intelligence (AAAI'18), 2018.
[37] I. Ntoutsi, A. Kalousis, Y. Theodoridis, A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees, in: Proc. SIAM International Conference on Data Mining (SDM'08), 2008, pp. 810–821.
[38] G. De'ath, K.E. Fabricius, Classification and regression trees: a powerful yet simple technique for ecological data analysis, Ecology 81 (11) (2000) 3178–3192.
[39] A. Bustillo, M. Grzenda, B. Macukow, Interpreting tree-based prediction models and their data in machining processes, Integr. Comput.-Aided Eng. 23 (4) (2016) 349–367.
[40] X. Liu, K.-K.R. Choo, R.H. Deng, R. Lu, J. Weng, Efficient and privacy-preserving outsourced calculation of rational numbers, IEEE Trans. Dependable Sec. Comput. 15 (1) (2018) 27–39.
[41] S. Hu, M. Li, Q. Wang, S.S. Chow, M. Du, Outsourced biometric identification with privacy, IEEE Trans. Inf. Forensics Secur. 13 (10) (2018) 2448–2463.
[42] B.K. Samanthula, Y. Elmehdwi, W. Jiang, K-nearest neighbor classification over semantically secure encrypted relational data, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1261–1273.
[43] J. Gao, W. Fan, J. Jiang, J. Han, Knowledge transfer via multiple model local structure mapping, in: Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08), 2008, pp. 283–291.
[44] Q. Wang, S. Hu, M. Du, J. Wang, K. Ren, Learning privately: Privacy-preserving canonical correlation analysis for cross-media retrieval, in: Proc. IEEE Conference on Computer Communications (INFOCOM'17), 2017, pp. 1–9.
[45] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, K. Chen, Privacy-preserving outsourced classification in cloud computing, Cluster Comput. (2017) 1–10.

Zhuoran Ma received the B.E. degree from the School of Software Engineering, Xidian University, Xi'an, China, in 2017. She is currently a Ph.D candidate with the Department of Cyber Engineering, Xidian University. Her current research interests include data security and secure computation outsourcing.

Jianfeng Ma received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1985, and the M.S. degree and the Ph.D. degree in computer software and telecommunication engineering from Xidian University, Xi'an, China, in 1988 and 1995, respectively. From 1999 to 2001, he was a Research Fellow with Nanyang Technological University of Singapore. He is currently a professor and a Ph.D. Supervisor with the Department of Computer Science and Technology, Xidian University, Xi'an, Chian. He is also the Director of the Shaanxi Key Laboratory of Network and System Security. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.

Yinbin Miao received the B.E. degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and Ph.D. degree with the Department of Telecommunication Engineering from xidian university, Xi'an, China, in 2016. He is currently a Lecturer with the Department of Cyber Engineering in Xidian university, Xi'an, China. His research interests include information security and applied cryptography.

Kim-Kwang Raymond Choo (SM'15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of various awards including the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, ESORICS 2015 Best Paper Award. He is an Australian Computer Society Fellow, and an IEEE Senior Member.

Ximeng Liu received the B.E. degree from the Department of Electronic Engineering, Xidian University, Xi'an, China, in 2010, and the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, in 2015. He is currently a Post-Doctoral Fellow with the Department of Information System, Singapore Management University, Singapore. His current research interests include applied cryptography and big data security.

Xiangyu Wang currently is a Ph.D candidate in Xidian University. He received the B.E. degree with the School of Cyber Engineering from Xidian University, Shannxi, China, in 2017. His research interests include data security and secure computation outsourcing.

Tengfei Yang received the M.S. degree with the School of Physics and Information Technology from Shaanxi Normal University, Xi'an, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering in Xidian University, Xi'an, China. His research interests include multimedia security and applied cryptography.