Towards Efficient and Privacy-Preserving Anomaly Detection of Blockchain-Based Cryptocurrency Transactions

Yuhan Song[1], Yuefei Zhu[1], and Fushan Wei[1,2(✉)]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
weifs831020@163.com

[2] Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China

Abstract. In recent years, a growing number of breaches targeting cryptocurrency exchanges have damaged the credibility of the entire cryptocurrency ecosystem. To prevent further harm, it's crucial to detect the anomalous behaviors hidden within cryptocurrency transactions and offer predictive suggestions. However, details of transaction records must be carefully analyzed for effective detection, and this information could be exploited by adversaries to launch attacks such as de-anonymization and model interference. As a result, it is essential to prioritize privacy preservation when designing an anomaly detection system for cryptocurrency transactions. In this paper, we propose a privacy-preserving anomaly detection (`PPad`) scheme for cryptocurrency transactions based on a decision tree model, which achieves privacy preservation by using additively homomorphic encryption and matrix perturbation techniques. We also design and implement `PPad`'s underlying protocol in a cloud outsourcing environment. The correctness and privacy properties of `PPad` have been proven through detailed analysis. Experimental results show that our scheme can offer privacy assurance with desirable detection effectiveness and efficiency, making it suitable for real-world applications.

Keywords: Anomaly detection · Blockchain · Privacy protection · Homomorphic encryption · Decision tree

1 Introduction

Cryptocurrency is widely recognized as a significant blockchain application, which allows users to securely store monetary assets and make anonymous payments in a decentralized manner. However, the significant economic value of cryptocurrency has made it a prime target for malicious cyber activities. While the security and reliability of cryptocurrency are supported by a stack of cryptographic technologies, potential threats can be introduced by various entities in the cryptocurrency ecosystem, including exchange platforms, wallet providers, and mining pools. In recent years, growing instances of breaches against Bitcoin

D. Wang et al. (Eds.): ICICS 2023, LNCS 14252, pp. 590–607, 2023.
https://doi.org/10.1007/978-981-99-7356-9_35

exchanges have diminished the credibility of Bitcoin ecosystem [13]. In 2014, Mt.Gox, the leading Bitcoin exchange at that time, filed for bankruptcy as nearly 850,000 BTCs worth over $450 million were stolen. In 2016, Bitfinex reported that 119,756 BTC valued at approximately $72 million were stolen, causing the value of BTC to plummet by about 20%. More recently, in January 2022, Crypto.com lost over $30 million in Bitcoin and Ethereum after being hacked by unknown reasons. Additionally, there are many cases in which the amount of tokens stolen is not reported. Therefore, implementing financial regulatory measures on cryptocurrency exchanges, such as transaction auditing and anomaly detection, is essential to prevent further token theft. Anomaly detection in cryptocurrency exchanges primarily concentrates on identifying fraudulent activities within transaction data and offering predictive maintenance suggestions. Recently, various studies have been presented for anomaly detection in different blockchain-based digital currencies [1,2,9]. In these works, the details of transactional data need to be thoroughly analyzed for accurate detection. However, if adversaries misuse this data by connecting it with offline information, the privacy of cryptocurrency users is at high risk of being compromised. In other words, adversaries might perform de-anonymization attacks [8]. Even worse, they could also execute interference [7] and extraction [19] attacks against the detection model. Therefore, it is vital to consider privacy preservation when designing an anomaly detection scheme for cryptocurrency transactions. Unfortunately, this issue has been largely overlooked in existing studies.

In light of this, our research is inspired by the following scenario. Suppose there is a trusted private server that is capable of collecting cryptocurrency transaction records, including anomalous records associated with theft activities. By extracting predefined features that represent the characteristics of anomalous transactions from these records, the private server can create a dataset that comprises transaction features and their classification labels (a normal transaction as "0" and an abnormal one as "1"). After creating the dataset, the private server trains a detection model that is subsequently transmitted to a cloud server that provides anomaly detection services. When a user creates a new transaction on the exchange platform, the private server extracts its feature vector and sends it to the cloud server for evaluation of its potential association with malicious activities. After the cloud server analyzes the transaction, it sends the detection result back to the private server. The private server then takes appropriate action based on the severity of the anomaly. Based on this result, the private server informs the exchange platform whether to proceed with or withdraw the transaction. In this scenario, several privacy concerns arise. First, the cloud server should not be able to access detailed information about transaction data. Second, to prevent interference attacks, the detection model should be kept secret from the cloud server. Third, the detection result should only be known to the private server.

In addition to privacy concerns, the anomaly detection scheme should also achieve a high level of detection accuracy to effectively identify fraudulent activities in cryptocurrency transactions. Furthermore, the scheme should be efficient enough to be used in real-world situations where large volumes of transactions need to be processed in real-time.

Our study introduces a privacy preserving anomaly detection system that achieves both desirable detection effectiveness and efficiency. To sum up, the main contributions of this paper are:

- We propose a general framework for privacy-preserving anomaly detection of cryptocurrency transactions through a secure outsourced computation architecture.
- Based on this framework, we have designed a two-party protocol that employs a decision tree classifier. To ensure privacy preservation, we adopt several techniques, including additively homomorphic encryption and matrix multiplication.
- Through a comprehensive security analysis and computational complexity assessment, we demonstrate that our design can achieve privacy preservation without excessive computational overhead.
- A comprehensive set of experiments was conducted to evaluate the effectiveness and efficiency of the detection system. The results indicate that our system can be deployed in real-time bitcoin-based anomaly detection scenarios with excellent performance.

2 Related Works

Recently, several works on anomaly detection of blockchain transactions have been proposed. Hirshman et al. [6] made the first attempt to figure out atypical transaction patterns in Bitcoin currency. Pham and Lee [15] used three unsupervised learning methods to detect anomalies in the Bitcoin network by analyzing the behaviors of suspicious users. However, this work only identified a few cases of Bitcoin theft. In another work of Pham and Lee [16], they used the laws of power degree & densification and the local outlier factor method (LOF) to analyze two graphs of the Bitcoin network for detecting suspicious users and transactions. Monamo et al. [12] highlighted the advantages of supervised learning models in detection accuracy. Despite the number of studies on anomaly detection of blockchain-based transactions, only a few have considered the issue of privacy protection. In [17], Song et al. introduced a general framework for anomaly detection in blockchain networks and proposed a corresponding protocol, `ADaaS`. However, due to its implementation based on the computationally expensive kNN model, the detection performance and effectiveness of `ADaaS` require further improvement.

In this paper, we adopt privacy-preserving decision tree (PPDT) to construct our anomaly detection protocol. Among existing works of PPDT, methods based on cryptographic technologies are notable for their improved privacy and accuracy guarantees. Lindell and Pinkas [10] were the first to design a PPDT training algorithm by using secure multi-party computation (MPC) and oblivious transfer (OT). For PPDT evaluation, Brickell et al. [4] devised a method by combining Homomorphic encryption (HE) and MPC. Bost et al. [3] used a fully HE-based method and represented the decision tree as a polynomial to enable private evaluation. For better efficiency, Wu et al. [20] introduced additively HE (AHE)

and OT into their scheme. Tai et al. [18] further improved the work in [20]. More recently, Cock et al. [5] adopted secret sharing (SS) to propose a PPDT evaluation method suitable for small trees.

3 Preliminary

This section provides an overview of the essential concepts and techniques that underpin our design. More specifically, we will introduce the *Paillier* cryptosystem, which offers privacy assurances, and the decision tree classifier, which is the underlying model of anomaly detection.

3.1 Paillier Cryptosystem

In this work, we adopt the additively homomorphic encryption scheme *Paillier* [14] for its efficiency and practicability. In its most basic variant, *Paillier* scheme is described as follows:

- **Pai.KeyGeneration** Select two large prime numbers p, q. Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$, where lcm is the least common multiple. Select $g \in \mathbb{Z}^*_{n^2}$ as a random integer while ensuring that n divides the order of g by checking the existence of the following modular multiplicative inverse, $\mu = (L(g^\lambda mod n^2))^{-1} mod n$, where $L(x) = \frac{x-1}{n}$. The public key is $pk = (n, g)$ and the private key $sk = (\lambda, \mu)$.
- **Pai.Encryption** To encrypt a message, we first select a random integer $r \in \mathbb{Z}^*_{n^2}$. Then we get the cipher value by computing $c = g^m \times r^n \bmod n^2$.
- **Pai.Decryption** A message $c \in \mathbb{Z}^*_{n^2}$ is decrypted by computing $m = L(c^\lambda \bmod n^2) \times \mu \bmod n$.

3.2 Decision Tree

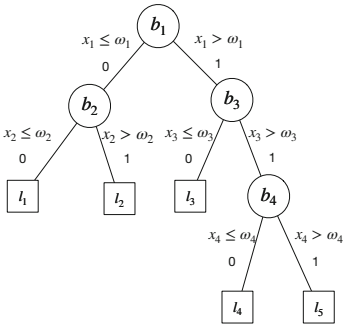


Fig. 1. Decision Tree

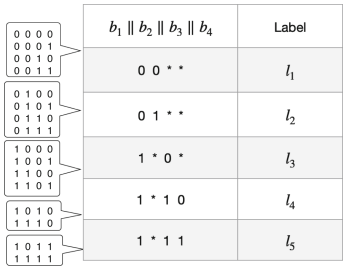


Fig. 2. Decision Table

Decision Tree is a non-parametric supervised learning method used for classification and regression. Due to its interpretability, non-parametric nature, and resilience to outliers, it can model complex, non-linear relationships and automatically select the most informative features, making it beneficial for classification tasks such as anomaly detection. In the hierarchical structure of a decision tree model, a root note, several branches, internal nodes and leaf nodes are included. An internal node corresponds to a partitioning rule (i.e. the threshold of a feature), and a leaf node represents a class label. To classify an instance, the decision tree is traversed from the root node to a leaf node by comparing with the thresholds at each internal node to determine the path to follow. Figure 1 illustrates the decision tree for a feature vector $X = [x_1, x_2, x_3, x_4]$ of a classification query, where the prediction label set is $L = \{l_1, l_2, l_3, l_4, l_5\}$, and the threshold vector is $W = [\omega_1, \omega_2, \omega_3, \omega_4]$.

Here we define a boolean variable b_i as a decision indicator for internal node i. If $x_i \leq \omega_i, b_i = 0$, else $b_i = 1$. As a result, the decision path to each leaf node can be interpreted as a boolean string. For instance, the decision path to the leaf node with prediction label l_1 in Fig. 1 is $b_1 = 0(x_1 \leq \omega_1)$ AND $b_2 = 0(x_2 \leq \omega_2)$, i.e. $b_1||b_2 = 00$. Based on this rule, we place all the decision paths of the tree classifier in a decision table. For the j-th row in the decision table, the first column stores the decision path of the leaf node corresponding to l_j represented as a boolean string, while the second column stores l_j. An internal node not traveled by is represented as dummy node. We use "$*$" to denote its boolean value. Here "$*$" means both 0 and 1. Therefore, each path in the decision table is an isometric boolean string whose length is the number of internal nodes. For example, the boolean string for leaf node with l_1 in Fig. 1 is 00**, which involves 4 rows, 0000, 0001, 0010, and 0011. Hence, as shown in Fig. 2, the decision table for the classifier in Fig. 1 has 16 rows.

4 Problem Formulation

In this work, we propose a system model with two entities that enables cloud-outsourcing anomaly detection while maintaining the privacy of the transaction data, detection model, and detection result.

4.1 System Model

We propose a cloud outsourcing architecture model as depicted in Fig. 3. This architecture includes two entities: the Transaction Committer and the Cloud Server.

Transaction Committer(TC) is a trusted private server acting as an agent of secure data exchange between the ledger and the cloud server. The TC is responsible for receiving large amounts of historical transactions from the blockchain ledger and training the detection model. In addition, TC also collects newly generated transactions from the exchange platform.

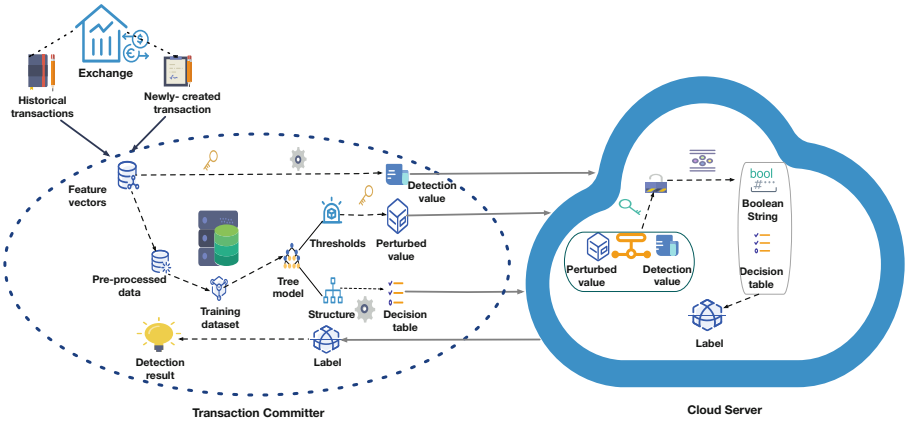


Fig. 3. Architecture Model of the System

Cloud Server(CS) is hosted by a third-party cloud service provider. It provides storage and computational resources for detecting anomalies in newly generated transactions using a pre-trained model in encrypted domain.

Once received the historical transactions from cryptocurrency exchange, TC extracts pre-defined features from each record and creates a feature vector. After pre-processing, TC generates a dataset for training a decision tree model. To facilitate subsequent operations, the decision tree model is processed in two parts: the thresholds of inner nodes and the tree structure. The threshold values are encrypted by TC using the secret key provided by CS. However, since CS holds the key of decryption, TC needs additional perturbation operation to ensure that the returned values are not easily decrypted by CS. As for the tree structure, TC creates a table to store all the decision paths and their corresponding prediction labels. The decision table is then processed by shuffling the paths and encrypting the labels before being sent to CS. Both of the perturbed thresholds and the processed decision table are securely transmitted and stored in the cloud server for later operations.

Once a new transaction is generated in the cryptocurrency exchange, it is sent to TC and transformed into a feature vector there. The feature vector is then encrypted and perturbed before being sent to CS for anomaly detection. CS uses pre-stored perturbed thresholds to calculate a value and extract a boolean string after decryption and comparison operations. The boolean string is then searched in the decision table to find its corresponding label, which is sent to TC for decryption. If the label indicates an anomalous transaction, an alert is sent to the exchange to withdraw the transaction.

4.2 Threat Model

In our model, TC is an honest party while CS is semi-honest. That is to say, it would strictly follow the protocol but may try to record intermediate results

during the execution and learn additional information from them. For instance, *CS* may record the encrypted feature vectors and attempt to recover the raw transaction data by conducting de-anonymization attacks. *CS* may also extract the topology and key parameters of the detection model to conduct interference attacks by sending anomaly detection queries. To mitigate the potential threats, we adopt encryption and perturbation techniques to prevent *CS* from learning sensitive information about the data and model.

4.3 Design Goals

To ensure privacy-preserving and efficient anomaly detection for blockchain-based transactions, the proposed scheme should satisfy the following requirements:

- **Data privacy**: The historical transaction records, newly-created transactions, and detection results are confidential and must not be exposed to *CS* or any other adversaries. Intermediate values during outsourcing and detection processing must also be kept private and not inferred by others.
- **Detection model privacy**: Model parameters such as threshold vector and tree structure, obtained by training plaintext data, must remain confidential from the cloud server and other adversaries.
- **Detection performance**: The anomaly detection system should achieve desirable detection effectiveness while minimizing the additional overhead caused by privacy protection operations.

5 Design and Implementation

In this section, we introduce `PPad` (Privacy Preserving Anomaly Detection), a two-party privacy- preserving anomaly detection scheme based on decision tree classifier.

5.1 Initialization

During this phase, we obtain a decision tree classification model and transform it to a table that stores all the potential decision paths and their respective labels. We also extract the threshold vector used for comparison operations in the following phase.

Upon receiving a set of labeled historical transaction records, *TC* extracts pre-defined features from each record to represent it as a feature vector $\boldsymbol{t}_i = [t_{i1}, t_{i2}, ..., t_{if}]$, where f is the number of features. With these feature vectors and their labels (the label of $\boldsymbol{t}_i$ is $l_i \in 0, 1$, where $l_i = 1$ denotes that $\boldsymbol{t}_i$ is anomalous), a dataset for training is created. Next, *TC* uses **CART** classification algorithm [11] to train a decision tree classifier *clf* with m internal nodes and n leaf nodes. The threshold at each internal node forms a vector $W = [\omega_1, \omega_2, ..., \omega_m]$. According to the comparison result with the each threshold, the decision path is represented as a boolean string $b_1 \parallel b_2 \parallel ... \parallel b_m, b_i$ is either 0 or 1. *TC* can thereby create a decision table, DT, which is composed of 2 columns and 2^m rows to store all the decision paths in *clf* and their respective prediction labels.

5.2 Key Generation

During the this phase, TC randomly selects an invertible matrix $M_1 \in Z_{m\times m}$ and computes its inverse M_1^{-1}, ensuring that $M_1 M_1^{-1} = I$. M_1 and M_1^{-1} are used for perturbation processing in later phase. Besides, TC generates a random permutation π to shuffle boolean variables in DT. TC also generates a private/public key pair for a probabilistic encryption scheme that is secure against chosen plaintext attacks (CPA). For simplicity, we do not provide the specifics of the CPA-secure probabilistic encryption scheme.

Given the security parameter κ, CS uses **Pai.Key Generation** to generate a pair of public and secret keys. After this, CS sends the public key $pk = (N, g)$ to TC, here $N = pq$. Meanwhile, the private key $sk = (\lambda, \mu)$ is kept by CS.

5.3 Model Outsourcing

With the vector of thresholds $W = [\omega_1, \omega_2, ..., \omega_m]$ and decision table DT obtained in **Initialization** phase, TC uses the encryption parameters generated in **Key Generation** phase to process them for meeting the requirements of secure computation in the next phase.

Firstly, TC computes the additive inverse of each threshold ω_i mod $N(i = 1, ..., m)$ which is denoted as $-\omega_i$ and uses the public key received from CS to encrypt it as $c_i =$**Pai.Encryption**$(-\omega_i)$. TC applies random permutation π to shuffle $\boldsymbol{c} = [c_1, c_2, ..., c_m]$, resulting in $\boldsymbol{c}^* = [c_1^*, c_2^*, ..., c_m^*]$. Using $\boldsymbol{c^*}$, TC constructs a diagonal matrix C^* as:

$$C^* = \begin{pmatrix} c_1^* & 0 & \cdots & 0 \\ 0 & c_2^* & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & c_m^* \end{pmatrix}$$

TC randomly chooses a lower triangular matrix $Q \in Z_{m\times m}$, where the elements in the main diagonal are all equal to 1. C^* is then multiplied by Q and the perturbation parameter M_1 to obtain C as $C = QC^*M_1$.

Next, TC applies the random permutation π to each decision path $p_j, (j = 1, 2, ..., 2^m)$ in DT, generating $p'_j = \pi(p_j)$. Meanwhile, the classification label l_j is encrypted as with a CPA-secure probabilistic encryption algorithm as $l'_j = Enc(l_j)$.

After completing the perturbation and shuffling processes, TC sends the resulting perturbed value C and the shuffled decision table DT' to CS. Subsequently, CS stores these values locally.

Algorithm 1 Model Outsourcing

Input: $W = [\omega_1, \omega_2, ..., \omega_m]$, DT, $pk = (N, g)$, M_1.
Output: C, DT'.
Transaction committer:
for $i = 1, 2, ..., m$ **do**
  Compute $-\omega_i$ for $\omega_i \in W$.
  select a random integer $r_i \in \mathbb{Z}_{n^2}^*$.
  Compute $c_i = g^{-\omega_i} \times r_i^N$,
end for
Apply π to shuffle $\boldsymbol{c} = [c_1, c_2, ..., c_m]$ and get $\boldsymbol{c}^* = [c_1^*, c_2^*, ..., c_m^*]$.
Construct $C^*$with $\boldsymbol{c}^*$.
Randomly select $Q \in Z_{m \times m}$.
Compute $C = QC^*M_1$.
for $j = 1, 2, ..., 2^m$ **do**
  Apply π to p_j in DT and get $p'_j = \pi(p_j)$.
  Compute l'_j=$Enc(l_j)$.
end for
Construct DT' with p'_j and l'_j.
Send C and DT' to CS
Cloud server:
Store C and DT' locally.
return C,DT'

5.4 Anomaly Detection

For the newly generated transaction T_r, TC firstly extracts the pre-defined features and created feature vector $\boldsymbol{a}_r = [a_{r1}, a_{r2}, ..., a_{rf}]$. According to the feature chosen at each inner node in clf, $\boldsymbol{a}_r$ is expanded to a m-dimensional vector $\boldsymbol{a}_r^* = [a_{r1}^*, a_{r2}^*, ..., a_{rm}^*]$. Following this, each component in $\boldsymbol{a}_r^*$ is encrypted using *Paillier* algorithm as c_{rj} =**Pai.Encryption**$(a_{rj}^*)(j = 1, 2, ..., m)$.

Secondly, TC applies random permutation π to shuffle $\boldsymbol{c}_r = [c_{r1}, c_{r2}, ..., c_{rm}]$ as $\boldsymbol{c}_r^* = [c_{r1}^*, c_{r2}^*, ..., c_{rm}^*]$. Using $\boldsymbol{c}^*$, TC constructs a diagonal matrix C_r^* in the same form as C^*.

Thirdly, TC uses the perturbation parameter M^{-1} to compute $C_r = M_1^{-1}C_r^*$. C_r is thereby sent to CS for subsequent detection processing.

Upon receiving the perturbed result C_r, CS uses the pre-stored matrix C to compute $D = CC_r$, where the diagonal element $d_i(i = 1, 2, ..., m)$ is decrypted as e_i = **Pai.Decryption**(d_i).

Subsequently, e_i is used to compare with $N/2$. The comparison result is denoted as a boolean variant b_{ri}. If $e_i \leq N/2$, $b_{ri} = 1$, else $b_{ri} = 0$. As a result, the m comparison results $b_{ri}, i = 1, 2, ..., m$ are stored in a boolean sequence $\boldsymbol{b}_r = [b_{r1}, b_{r2}, ..., b_{rm}]$. CS then searches the shuffled decision table DT' to find the item that matches $\boldsymbol{b}_r$ and obtains its corresponding label l_{enc}. After this, the encrypted classification label l_{enc} is sent to TC. TC decrypts l_{enc} to get the final detection result r_d.

Algorithm 2 Anomaly Detection

Input: T_r, DT', $pk = (N, g)$,$sk = (\lambda, \mu)$, M_1^{-1}.
Output: r_d.

Transaction committer:
Construct feature vector $\boldsymbol{a}_r = [a_{r1}, a_{r2}, ..., a_{rf}]$ for T_r.
Expand $\boldsymbol{a}_r$ to $\boldsymbol{a}_r^*$.
for $j = 1, 2, ..., m$ **do**
  select a random integer $r_j^* \in \mathbb{Z}_{n^2}^*$
  Compute $c_{rj} = g^{a_{rj}^*} \times r_j^{*N}$.
end for
Shuffle $\boldsymbol{c}_r$ with π to get $\boldsymbol{c}_r^*$.
Construct C_r^* with $\boldsymbol{c}_r^*$.
Compute $C_r = M_1^{-1} C_r^*$.
Send C_r to *CS*.
Cloud server:
Compute $D = CC_r$ and extract $\boldsymbol{d} = [d_1, d_2, ..., d_m]$ from D.
for $i = 1, 2, ..., m$ **do**
  Compute $e_i = L(d_i^{\lambda} \bmod N^2) \times \mu \bmod N$.

  Compare e_i with $N/2$.
  if $e_i \leq N/2$ **then**
    $b_{ri} = 1$.
  else
    $b_{ri} = 0$.
  end if
end for
Store $\boldsymbol{b}_r = [b_{r1}, b_{r2}, ..., b_{rm}]$.
Search DT' to find l_{enc} that matches $\boldsymbol{b}_r$.
Send l_{enc} to TC.
Transaction committer:
Decrypt l_{enc} as $r_d = Dec(l_{enc})$.
return the detection result r_d.

6 Security Analysis

In this section, we will analyze the security properties of the proposed scheme. Firstly, we will prove the correctness of `PPad` protocol through theoretical analysis. Secondly, we will examine the privacy properties of data processed in the outsourcing and detection phases. Thirdly, we will demonstrate that the detection model is also kept private from the cloud server.

Theorem 1. *(Correctness) If the protocols described in Sect. 5 are honestly followed by TC and CS, TC will obtain the correct detection result eventually.*

Proof. As previously mentioned, for a newly created transaction T_r, its feature vector $\boldsymbol{a}_r$ is expanded to a m-dimensional vector $\boldsymbol{a}_r^*$ and each component is encrypted by *Paillier* algorithm to obtain $\boldsymbol{c}_r$. The shuffled sequence $\boldsymbol{c}_r^* = \pi(\boldsymbol{c}_r)$ is used to construct diagonal matrix C_r^*. After this, C_r^* is perturbed as $C_r =$

$M_1^{-1}C_r^*$, where M_1^{-1} is the inverse of M_1. During the anomaly detection phase, CS uses C_r and the pre-stored C to compute$D = CC_r = (QC^*M_1)(M_1^{-1}C_r^*) = QC^*IC_r^* = QC^*C_r^*$. Since Q is a lower triangular matrix with all elements equal to 1 in the main diagonal, and both C^* and C_r^* are diagonal matrices, the main diagonal elements of D can be computed as $d_k = c_k^* c_{rk}^*$, where $k = 1, 2, ..., m$. For $c_k^* c_{rk}^* = g^{-\omega_k} r_k^N g^{a_{rk}} r'^N_k = g^{a_{rk}-\omega_k}(r_k r'_k)^N$, using the additive homomorphic properties of *Paillier* algorithm, we know that the result of decrypting $c_k^* c_{rk}^*$ is $e_k = a_{rk} - \omega_k \bmod N$, which represents the comparison between the feature value and threshold of the corresponding inner node. Based on the properties of modulo computation, we can infer that if $a_{rk} \geq \omega_k$, the decryption value $e_k \leq N/2$ ($b_r = 1$), else if $a_{rk} < \omega_k$, $e_k > N/2$ ($b_r = 0$). Therefore, the boolean sequence $\boldsymbol{b}_r = [b_{r1}, b_{r2}, ..., b_{rm}]$ denotes the decision path in the tree model for T_r. By searching the decision table DT' with $\boldsymbol{b}_r$, we can retrieve the corresponding encrypted classification label l_{enc}. After decryption by TC, the final detection result is obtained.

Theorem 2. *(Data Privacy) In the execution of our protocol, CS does not have access to any information about the transaction to be detected.*

Proof. During the anomaly detection phase, the m-dimensional feature vector $\boldsymbol{a}_r^*$ of T_r is encrypted by *Paillier* algorithm in a similar manner to the threshold vector during the model outsourcing phase. TC then shuffles $\boldsymbol{a}_r^*$ using π and constructs a diagonal matrix C_r^*. Finally, the perturbation value $C_r = M_1^{-1}C_r^*$ is sent to CS. Since CS knows nothing about M_1^{-1} and its inverse M_1, it cannot obtain the shuffled ciphertext of $\boldsymbol{a}_r$ in the main diagonal of C_r^* from C_r. Therefore, CS cannot decrypt any information about T_r. During the detection processing of T_r, CS only computes the product of C_r and the pre-stored $C = QC^*M_1$. In this step, CS only gets the shuffled product of threshold and T_r's corresponding feature in encrypted version. Therefore, no information about the transaction T_r is disclosed to CS.

Theorem 3. *(Model Privacy) During the execution of our protocol, CS cannot infer any additional information about the decision tree model.*

Proof. TC divides the pre-trained model into two parts, the threshold vector W , and the decision table DT. For each threshold $\omega_i \in W (i = 1, 2, ..., m)$, TC first encrypts it as $c_i = g^{-\omega_i} r_i^N \bmod N^2$. Then, using a random perturbation π, $\boldsymbol{c} = [c_1, c_2, ..., c_m]$ is shuffled to obtain $\boldsymbol{c}^* = [c_1^*, c_2^*, ..., c_m^*]$, which is used to construct the diagonal matrix C^*. Finally, TC computes $C = QC^*M_1$ and sends it to CS. In the previous section, it was explained that the matrix Q is a lower triangular matrix with the main diagonal consisting of m elements equal to 1, and M_1 is an invertible matrix. Even though CS possesses the decryption key, it is still unable to decrypt the value of the thresholds without any knowledge about M_1. While during the phase of anomaly detection, TC computes the perturbation value of T_r's feature vector as $C_r = M_1^{-1}C_r^*$ and sends it to CS. CS can only obtain the product of perturbation values C and C_r. Since Q,

M_1, and M_1^{-1} are randomly selected parameters, CS can not deduce anything about C^* from this product value. Therefore, it is impossible for CS to know the plaintext version of W by decrypting C^*.

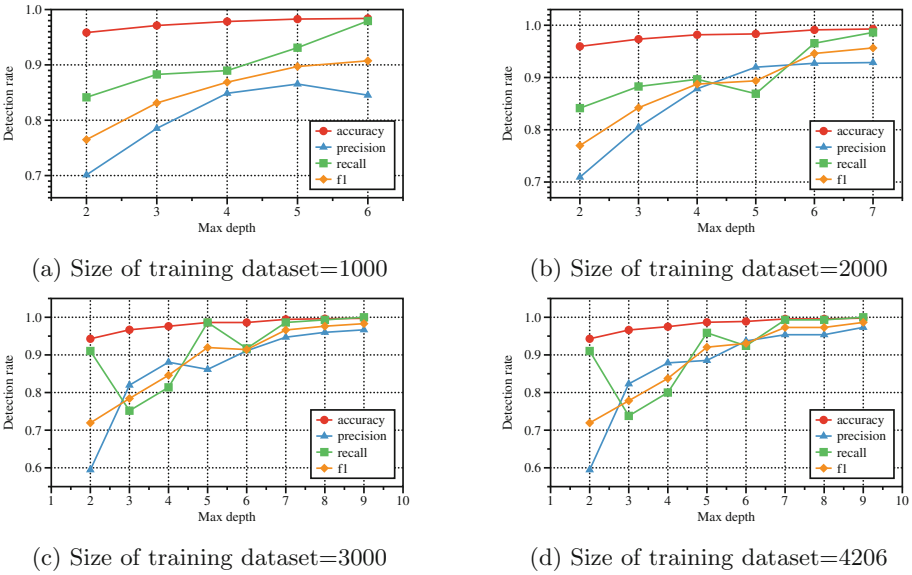
As for the decision table DT, each row in it indicates a boolean string of decision path, which is shuffled by TC with a random permutation π. As a result, the order of each dimension in the boolean string is disrupted in the new decision table DT'. Even if CS or another attacker obtains DT', they can only guess the value of original decision path with a probability of $\frac{1}{2^m}$. Moreover, the corresponding label of the boolean decision path is encrypted by TC who also holds the key of decryption. Thus, both the threshold information and the structure of decision tree are well protected and cannot be easily used by CS to deduce additional information.

7 Experiments and Evaluation

7.1 Effectiveness and Efficiency Experiments

We used a dataset that contains 6010 Bitcoin transaction records (including 454 theft-related records), where each record is depicted as a 9-dimensional feature. For more details, please refer to [17]. Our experimental setup consisted of two servers, both equipped with Intel i9-9980XE 36-core 3.00GHz processor and 128 GB memory, running Windows 10. One server acted as the transaction committer, while the other served as the cloud server. The implementation of our system was developed in `Python3`, using libraries such as `gmpy2`, `numpy`, and `pandas`. The decision tree model was trained non-privately using `scikit-learn`. Two sets of experiments were conducted to evaluate the detection effectiveness and efficiency of our proposed scheme `PPad`. The experiments were divided into 4 subgroups, each with a training dataset of size 1000, 2000, 3000 and 4206. In each subgroup, we varied the maximum depth of decision tree, which reflects the complexity of the model. Furthermore, we also compared our results to those presented by Song et al. in [17] (see Apeendix).

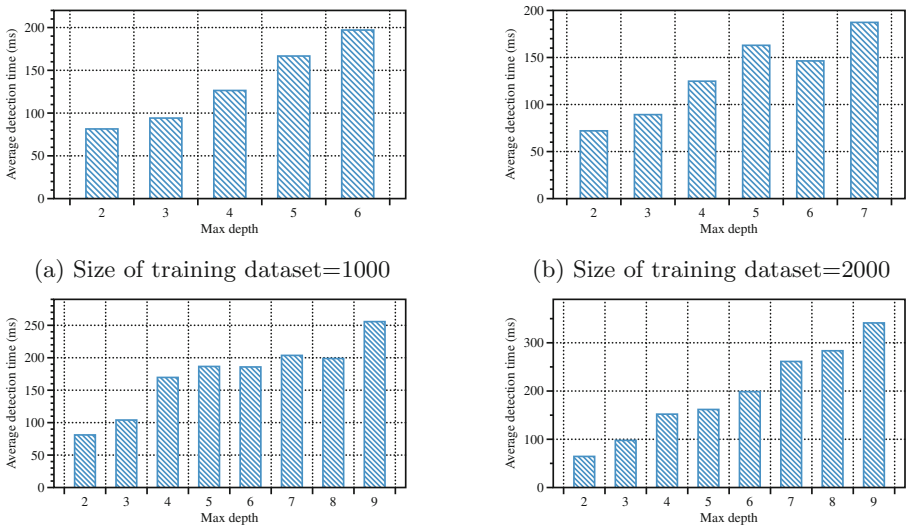
Figure 4 illustrates the effectiveness of `PPad` in anomaly detection with different sizes of training datasets. The accuracy, precision, recall, and F1 score are measured for 1803 randomly selected testing samples. It can be observed that these indicators increase with max depth in most cases. The detection accuracy stays above 95%, and as the max depth grows, it gradually approaches 100%. The detection precision, ranging from 59% to 97%, grows consistently with max depth. Additionally, for a given maximum depth, the model trained with more samples achieves a higher detection precision. The recall score shows several turning points in the plots when the size of the training set is 3000 and 4206, which means that it does not increase with max depth within certain ranges. However, for max depth bigger than 5, the recall score is close to 100%. In all of these four cases, the F1 score increases steadily with max depth. It should be noted that the maximum value of max depth for each training set varies since it depends on the minimum number of samples required to split an internal node.



(a) Size of training dataset=1000 (b) Size of training dataset=2000

(c) Size of training dataset=3000 (d) Size of training dataset=4206

Fig. 4. Detection effectiveness with different size of training dataset.



(a) Size of training dataset=1000 (b) Size of training dataset=2000

(c) Size of training dataset=3000 (d) Size of training dataset=4206

Fig. 5. T_{avg} of model trained with different size of training dataset.

To evaluate the efficiency of the `PPad`, we measure the average time for detecting a single transaction record. The average detection time T_{avg} is defined as the total running time divides the number of testing samples. The total running time

is the sum of initialization time, key generation time, model outsourcing time, and anomaly detection time. As is shown in Fig. 5, PPad only requires milliseconds of time to detect a newly-created transaction. Assume the average size for each transaction record in a bitcoin block is 550 bytes, a 1 MB block contains about 1818 transaction records. Since the block time on the bitcoin blockchain is roughly 10 min, the upper bound of T_{avg} is 330 ms. As is shown in Fig. 5, T_{avg} grows steadily with max depth and size of training set. The maximum value of T_{avg} is 341.14 ms when the size of training set is 4206 with max depth at 9. However, except for this point, all the other experimental results are below 330 ms. Therefore, our scheme is feasible for real-world scenarios in Bitcoin exchanges. We have observed that there is a trade-off between detection effectiveness and efficiency in our analysis. Better detection effectiveness is achieved at the cost of reduced detection efficiency. Hence, it is crucial to select suitable parameters that achieve a trade-off between effectiveness and efficiency to obtain an ideal detection model.

7.2 Complexity Analysis

In this part, we evaluate the computation and communication complexity of PPad scheme. With respect to computation cost, we focus on computationally expensive operations such as encryption and decryption, while omitting the cost of other operations such as matrix multiplication and permutation. During the **Model Outsourcing** phase, TC encrypts the inverse of each threshold at m internal nodes and uses matrix multiplication to randomize these ciphertexts. Hence, the computation complexity of TC in this phase is m *Paillier* encryption operations. During the **Anomaly Detection** phase, TC encrypts each dimension of an expanded feature vector. Since the number of testing samples is t, the computation complexity of TC in this phase is mt *Paillier* encryption operations. As for CS, it computes the product of perturbed detection query and then decrypts the eigenvalues. Therefore, the comutation complexity of CS during the **Anomaly Detection** phase is mt *Paillier* decryption operations. With respect to communication cost, we consider the bandwidth and communication rounds. For each query, the bandwidth is $O(m^2)$ and 2 communication rounds are

Table 1. Performance Comparison (m: Number of internal nodes, n: Number of leaf nodes, d: Max depth, f: Number of features, t: Number of detection queries.)

Schemes	Privacy Strategies	Communication Complexity	Rounds of Communication	Server Complexity	Client Complexity
[4]	HE+GC	$O(m+n)$	≈ 5	N/A	N/A
[3]	FHE/SWHE	$O(m)$	≥ 6	$O(mf)$	$O((m+t)f)$
[20]	AHE+OT	$O(m)$	6	$O(mf+2^d)$	$O((m+t)f+d)$
[18]	AHE	$O(m)$	4	$O(mf)$	$O((m+t)f))$
[5]	SS	$O(m+n)$	≈ 9	$O(mf+2^d)$	$O((m+t)f+d)$
Ours	AHE+Matrix Perturbation	$O(m^2)$	2	$O(mt)$	$O(mt)$

required. In Table 1, we compare the computation and communication complexities of PPad with those of other related works in PPDT. The results show that PPad has low computation complexity and communication rounds but requires more bandwidth due to the combination of matrix multiplication and homomorphic encryption. However, since m is usually a small number, our protocol achieves better computation efficiency with reasonable bandwidth.

8 Conclusion

Our paper presents an efficient privacy-preserving anomaly detection scheme for blockchain-based cryptocurrency transactions in a cloud outsourcing environment. The scheme is based on a decision tree model, which is pre-trained in plaintext and sent to the cloud server after decryption and perturbation processing to ensure the privacy of transaction data and the final detection result. Our design also prevents the cloud server from inferring additional information from the detection model, thereby protecting against potential attacks such as model extraction or interference. Future work will focus on enhancing privacy protection during tree model training by utilizing MPC techniques and exploring the integration of ensemble learning methods to further improve the performance and effectiveness of our scheme.

Acknowledgement. This work was supported by the National Key Research and Development Program of China (No. 2019QY1300), the National Natural Science Foundation of China (No. 61772548, No. 62102447), the Science Foundation for the Excellent Youth Scholars of Henan Province (No. 222300420099), and Major Public Welfare Projects in Henan Province (No. 201300210200).

A Appendix

In this part, we compare PPad scheme and ADaaS in [17] through theoretical analysis and experiments. From theoretical level, we analyze the detection model, privacy strategies, complexities, and contribution of these two schemes, which are summarized in Table 2. Generally speaking, *Paillier* operations take more time than *VHE* operations due to their bit-by-bit nature. However, in the context of this paper, the dimension of a transaction vector is 9, and the number of internal nodes, m, is much smaller than the number of training samples, n (where m is under 100 and n is over 1000). As a result, based on the real parameter settings, PPad scheme is more efficient than ADaaS, a fact which is later confirmed by experimental results.

Table 2. Overall comparison between `ADaaS` and `PPad`. (m: number of internal nodes, n: number of training samples, E_v: the execution time of one *VHE* encryption, IP_v: the execution time of one *VHE* inner product, E_p: the execution time of one *Paillier* encryption, D_p: the execution time of one *Paillier* decryption.)

Scheme	Detection Model	Privacy Strategies	Rounds of Communication	Computation Complexity	Contribution
`ADaaS`	kNN	VHE+Matrix Perturbation	2	$E_v + nIP_v$	General framework
`PPaD`	Decision Tree	AHE+Matrix Perturbation	2	$m(E_p + D_p)$	Practical for real-time detection

The comparative experiments of effectiveness and efficiency are divided into 7 subgroups by varying the size of training dataset from 1000 to 4206, while the number of testing samples is 1803. We set the maximum depth of decision tree in `PPad` scheme to 5,resulting the value of m ranging from 23 to 35, and we set the modulus number for *Paillier* to $N = 512$. As for `ADaaS`, we set the nearest neighbour parameter k to 5, with *VHE* parameters of $m' = 11$, $n' = 12$. In each subgroup, the effectiveness indicators such as accuracy, precision, recall and F1 score are measured. For assessing the detection efficiency performance, we measure the average detection time for each transaction record, T_{avg}.

Table 3. Effectiveness comparison between `ADaaS` and `PPad`.

Size of training dataset	Method	Accuracy(%)	Precision(%)	Recall(%)	F1 score(%)
1000	`ADaaS`	96.73	77.92	82.76	80.27
	`PPad`	98.28	86.54	93.10	89.70
1500	`ADaaS`	96.67	78.52	80.69	79.59
	`PPad`	98.28	89.58	88.97	89.27
2000	`ADaaS`	97.06	80.26	84.14	82.15
	`PPad`	98.34	91.97	86.90	89.36
2500	`ADaaS`	97.17	80.52	85.52	82.94
	`PPad`	98.67	89.03	95.17	92.00
3000	`ADaaS`	97.34	82.55	84.83	83.67
	`PPad`	98.61	86.14	98.62	91.96
3500	`ADaaS`	97.45	82.78	86.21	84.46
	`PPad`	98.34	93.89	84.83	89.13
4206	`ADaaS`	97.84	85.81	87.59	86.69
	`PPad`	98.67	88.54	95.86	92.05

The results presented in Table 3 indicate that our proposed scheme PPad, outperforms ADaaS in terms of effectiveness metrics across almost all subgroups, except for when the training dataset size is 3500, where ADaaS exhibits slightly higher recall. Regarding detection efficiency, as shown in Fig. 6, both schemes have similar trends where the average detection time T_{avg} increases with the size of the training dataset. However, the increase in T_{avg} for ADaaS is more rapid than that of PPad. In general, PPad requires significantly less time to detect a newly-created transaction in each subgroup. Therefore, it can be concluded that our proposed scheme PPad offers a more practical solution than ADaaS as it achieves better detection effectiveness and efficiency.

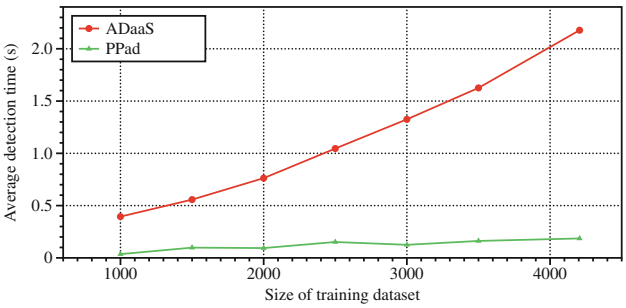


Fig. 6. Efficiency comparison between PPad and ADaaS

References

1. Awan, M.K., Cortesi, A.: Blockchain transaction analysis using dominant sets. In: Saeed, K., Homenda, W., Chaki, R. (eds.) CISIM 2017. LNCS, vol. 10244, pp. 229–239. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59105-6_20
2. Bartoletti, M., Lande, S., Pompianu, L., Bracciali, A.: A general framework for blockchain analytics. In: Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, SERIAL@Middleware 2017, pp. 7:1–7:6. ACM (2017). https://doi.org/10.1145/3152824.3152831
3. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015. The Internet Society (2015). https://www.ndss-symposium.org/ndss2015/machine-learning-classification-over-encrypted-data
4. Brickell, J., Porter, D.E., Shmatikov, V., Witchel, E.: Privacy-preserving remote diagnostics. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 498–507. ACM (2007)
5. Cock, M.D., et al.: Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. IEEE Trans. Dependable Secure Comput. **16**(2), 217–230 (2019). https://doi.org/10.1109/TDSC.2017.2679189

6. Hirshman, J., Huang, Y., Macke, S.: Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network, 3rd ed. Technical report, Stanford University (2013)
7. Jia, J., Salem, A., Backes, M., Zhang, Y., Gong, N.Z.: Memguard: defending against black-box membership inference attacks via adversarial examples. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pp. 259–274. ACM (2019). https://doi.org/10.1145/3319535.3363201
8. Khalilov, M.C.K., Levi, A.: A survey on anonymity and privacy in bitcoin-like digital cash systems. IEEE Commun. Surv. Tutor. **20**(3), 2543–2585 (2018). https://doi.org/10.1109/COMST.2018.2818623
9. Kumar, N., Singh, A., Handa, A., Shukla, S.K.: Detecting malicious accounts on the Ethereum blockchain with supervised learning. In: Dolev, S., Kolesnikov, V., Lodha, S., Weiss, G. (eds.) CSCML 2020. LNCS, vol. 12161, pp. 94–109. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49785-9_7
10. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_3
11. Loh, W.: Classification and regression trees. WIREs Data Min. Knowl. Discov. **1**(1), 14–23 (2011). https://doi.org/10.1002/widm.8
12. Monamo, P.M., Marivate, V., Twala, B.: A multifaceted approach to bitcoin fraud detection: Global and local outliers. In: 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, pp. 188–194. IEEE Computer Society (2016). https://doi.org/10.1109/ICMLA.2016.0039
13. Oosthoek, K., Doerr, C.: Cyber security threats to bitcoin exchanges: adversary exploitation and laundering techniques. IEEE Trans. Netw. Serv. Manag. **18**(2), 1616–1628 (2021). https://doi.org/10.1109/TNSM.2020.3046145
14. Paillier, P.: Public-Key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
15. Pham, T., Lee, S.: Anomaly detection in bitcoin network using unsupervised learning methods. CoRR abs/1611.03941 (2016). http://arxiv.org/abs/1611.03941
16. Pham, T., Lee, S.: Anomaly detection in the bitcoin system - A network perspective. CoRR abs/1611.03942 (2016). http://arxiv.org/abs/1611.03942
17. Song, Y., Wei, F., Zhu, K., Zhu, Y.: Anomaly detection as a service: an outsourced anomaly detection scheme for blockchain in a privacy-preserving manner. IEEE Trans. Netw. Serv. Manag. **19**(4), 3794–3809 (2022). https://doi.org/10.1109/TNSM.2022.3215006
18. Tai, R.K.H., Ma, J.P.K., Zhao, Y., Chow, S.S.M.: Privacy-Preserving decision trees evaluation via linear functions. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 494–512. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_27
19. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: USENIX Security Symposium, vol. 16, pp. 601–618 (2016)
20. Wu, D.J., Feng, T., Naehrig, M., Lauter, K.: Privately evaluating decision trees and random forests. Proc. Priv. Enhanc. Technol. **4**, 335–355 (2016)