Multimedia Tools and Applications (2022) 81:40343–40359
https://doi.org/10.1007/s11042-022-13163-w

ISC-MTI: An IPFS and smart contract-based framework for machine learning model training and invocation

Hao Lin[1,2,3] · Xiaolei Li[2,3] · Haoyu Gao[2,3] · Jie Li[2,3] · Yongsheng Wang[2,3]

Received: 1 April 2021 / Revised: 21 December 2021 / Accepted: 10 April 2022 /
Published online: 7 May 2022

Abstract

Due to centralized storage, centralization problems are common in machine learning model training and invocation, which makes train data and trained models extremely vulnerable to tampering and stealing. A safe framework for training and invoking models called ISC-MTI (IPFS (InterPlanetary File System) and Smart Contract-Based Method for Storage and Invocation of Machine Learning Mobel) is proposed in this paper. The framework uses IPFS as the storage solution, EOS (Enterprise Operation System) blockchain as the smart contract platform, RSA and AES as the implementation of encrypted communication. The Action responsible for invoking the training data and trained models in the smart contract and the model training, uploading, and invoking methods are designed. The experimental results demonstrate that ISC-MTI can improve the safety of model training and invocation with losing a little efficiency. Simultaneously, ISC-MTI can provide anti-theft model capabilities, traceability, tamper resistance, reliability, and privacy for the process.

Keywords Machine learning security · InterPlanetary file system · Smart contract · EOS · Model training · Model invocation

✉ Xiaolei Li
llxhappy@126.com

Hao Lin
suzukaze_aoba@foxmail.com

Haoyu Gao
1095055672@qq.com

Jie Li
uestc007@qq.com

Yongsheng Wang
517602864@qq.com

[1] School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China

[2] College of Data Science and Application, Inner Mongolia University of Technology, Inner Mongolia, China

[3] Inner Mongolia Autonomous Region Engineering and Technology Research Center of Big Data Based Software Service, Inner Mongolia, China

1 Introduction

With the development of machine learning, the performance of machine learning models is getting higher [1, 14, 17]. Machine learning models are constantly being deployed to personal computers and company servers to provide public access interfaces based on MLaaS (Machine Learning as a service) for users to predict and classify their test samples [16]. One of the main obstacles to the development of machine learning is the security issues facing machine learning. Without safeguarding machine learning, machine learning is vulnerable to model theft, adversarial (escape) samples, software system vulnerabilities, training data pollution, model tampering, etc [6, 7, 12, 28, 33]. At present, centralization problems are common in the training and storage process of machine learning models, which make train data and models extremely vulnerable to tampering and stealing. Hackers can use tampered machine learning models to generate more malicious software or more deceptive vulnerabilities [13]. More seriously, private information in training data may be reversed from the tampered machine learning models [3]. Once the models are stolen, the privacy-sensitive data collected and the models trained by companies or research institutes will lose their original value, which will cause substantial economic losses to the companies or research institutes.

Considering security issues facing machine learning models, we design a safe and reliable framework based on IPFS and smart contract for training and invoking of machine learning models. The framework has decentralization, resistance to deletion and tampering, traceability, and high reliability, which can effectively resist training data pollution, malicious theft to data and models, and model tampering.

1.1 Main contributions

The main contributions of this paper are summarized as follows:

- We propose a novel framework for training and invoking of machine learning called simply ISC-MTI.
- In the model training phase of ISC-MTI, IPFS is used as a storage solution for training data and models. The models are uploaded to IPFS after being trained in memory with the training data read from IPFS. In the model invocation phase of ISCMTI, EOS blockchain is used to build train data and model invocation contracts, which stipulate that the users must pay tokens to apply model invocations.
- Based on the custom hash algorithm provided by IPFS, we use RSA and AES for data encrypted transmission to prevent ISC-MTI from possible man-in-the-middle attacks. Moreover, we have designed the transmission rules of keys in ISC-MTI.
- We validate the efficiency of ISC-MTI and conduct multiple safety analyses.

1.2 Organisation of this work

The remainder of this paper is organized as follows: the related works are discussed in Section 2. Section 3 presents the selection of associated technologies we used in this paper. Section 4 describes the threat model. Section 5 describes the ISC-MIT framework. Section 6 presents the experimental results and safety analysis. Finally, Section 7 draws the conclusion and future research directions.

2 Related work

2.1 Training and invocation of machine learning model

When we need to use machine learning to train a model or use a trained model, the general process is shown in Fig. 1 [24].

The general process involves three roles: data owner, model trainer, and model user. Data owners provide data to the model trainers. According to the data, the model trainers select the algorithms, parameters, and other information to train models. The machine learning algorithms will generate some model files at the end of training. The model trainers will store the model files in the local hard disks or the third-party cloud platforms and provide the user interface for model users. Model users will be charged a certain fee when using the model, which will enable the model trainers to earn back the cost they invested in training the model. In some cases, a group of people are both data owners and data trainers. But in commercial scenarios, model users and model trainers will not be the same group of people.

2.2 Protection method of machine learning model

At present, there are two main types of research about protection methods for machine learning training and invocation, namely centralized protection methods and decentralized protection methods. The centralized protection methods focus on resisting adversarial sample attacks and utilizing encryption algorithms. Khoda et al. [9] noted that the common method for defending adversarial sample attacks is to train machine learning models with adversarial samples and normal samples simultaneously. They proposed two new sample combination methods—— the method based on the distance from malware cluster center and the method based on a probability measure derived from kernel-based learning (KBL). The results show that the KBL improves detection accuracy by 6%. Xu et al. [31] proposed VerifyNet. It uses key sharing protocols to protect the integrity of training samples, thereby preventing malicious adversaries from tampering with training samples and calculation results. Sun et al. [22] and Xu et al. [32] studied how to use the encrypted training data to train the models based on homomorphic encryption. The experiment verified the feasibility of their methods in terms of efficiency and accuracy. Centralized protection methods

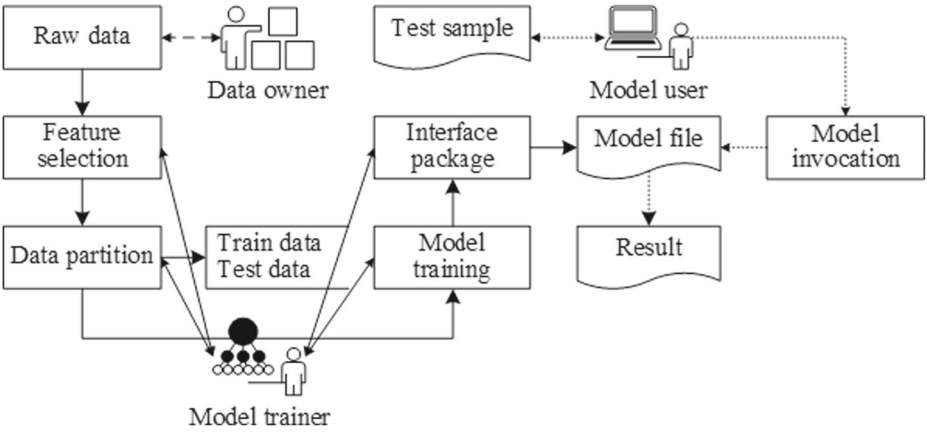

Fig. 1 General process of machine learning model training and invocation

are various and logically complex. These research assumes that hackers can easily steal the models and training data. As compared to defenders, hackers will always be in an advantageous position [8]. If attackers steal the models and training data, they will be able to research more novel, effective and targeted attack methods.

The decentralized protection methods store training data and model files in decentralized distributed storage [19, 25, 29]. In the traditional cloud environment, the methods of model storage rely on cloud service providers and TPAs (Third Party Auditors) [15]. Because cloud service providers and TPAs are not completely trusted, cloud servers are vulnerable to DDoS (Distributed denial of service attack), server node failures, and other reasons, the training data and trained models in the cloud have risks such as integrity destroyed, privacy leaked, and single node failure causing model unavailability. For solving these problems, scholars proposed that protect related data with blockchain. Blockchain has the characteristics of decentralization, audit-ability, and non-tampering, which can significantly improve train data and model security. Due to the need to reach a consensus on the onchain data within the entire network, blockchain is not suitable for storing large files [23]. In the production environment, data size of train data and models is vast. Enterprises cannot afford the high storage cost of blockchain. If the blockchain combines with traditional distributed file systems such as HDFS (Hadoop Distributed File System) and HBase (Hadoop Database), it will cause partial centralization and weaken the protection capabilities of the blockchain for machine learning [34]. Besides, in the Internet of Things(IoT), Internet of Vehicles(IoV), and Industrial Internet of Things, equipment security is worse and more vulnerable to attacks [11, 35]. However, calculation ability of equipment in these environments is low and blockchain platforms such as Bitcoin and Ethereum cannot work well.

2.3 Motivation

Therefore, there is a need to develop a framework for the protection of machine learning model as a benchmark to:

- safely store training data and trained models.
- protect training process and invocation process of machine learning.
- apply blockchain technology without high storage cost.
- do not require high calculation ability of the equipment.

The research and application of the framework can effectively protect machine learning related transactions. Furthermore, it is beneficial to improve data utilization and develop data sharing. In the big data era, we believe the framework is meaningful.

3 Selection of associated techologies

3.1 Train data and model document of machine learning

The training data involved in machine learning can be divided into structured data and unstructured data. Common algorithm libraries and frameworks for machine learning include Scikit-learn, TensorFlow, Keras, etc. According to algorithm libraries and frameworks, the trained model files can be saved as txt, Ckpt, pb, HDF5 (Hierarchical Data File 5) and json, which store the structure, index, internal parameters, and other data of the models. For compatibility with all file types, we must choose a decentralized file system that supports object storage to store training data and trained models.

3.2 Storage method

Since 2010, many file systems are developing towards decentralization, such as GPFS (General Parallel File System), Ceph, and IPFS [2, 5]. Table 1 presents the comparison of these decentralized file systems.

IPFS is superior to other file systems in terms of supporting multiple OSs, complete decentralization, automatic de-redundancy, and anti-DDOS ability. IPFS protocol stack is shown in Fig. 2.

IPFS has no storage capacity limitation and provides high-throughput content-addressed storage, which is perfectly suitable for protecting training data and model files involved in the machine learning training process. When data needs to be extracted, IPFS reads the data blocks in all nodes in parallel. Therefore, IPFS can meet the read and write performance requirements of the training process. IPFS nodes are located all over the world, which can guarantee enough backups. Hackers cannot attack all nodes simultaneously, enabling IPFS to store and share a large number of data and files that are secure. Moreover, IPFS has a version control function similar to Git, which facilitates management and utility of trained models. So we choose IPFS as the storage solution for ISC-MTI.

3.3 Blockchain and smart contract

Smart Contract is a computer protocol designed to spread, verify, and execute contracts in an information-based way. Smart contracts allow trusted transactions without TPAs, which are traceable and irreversible. Blockchain makes smart contracts have the advantages of programmability, decentralization, non-tampering, and traceability. In the era of blockchain 3.0, current smart contract platforms include Bitcoin, Ethereum, Hyperledger, and EOS [4, 18, 26]. Table 2 presents the comparison of these smart contract platforms.

Among papers about existing smart contracts, a good deal of equipment is used for mining [20, 30, 35], which is not suitable for the IoT, IoV, and Industrial Internet environments. EOS has low demand for equipment calculation ability and the TPS (Transaction Per Second) of EOS is among the best in current smart contract platforms. The structure of an EOS Blockchain node is shown in Fig. 3.

Cleos is client of EOS and supports interaction with users. Keosd is a wallet component used to save account keys. Nodeos is the daemon and core of EOS nodes, responsible for generating blocks. EOS can complete the generation of a block every 0.5 seconds. The operation of transactions and smart contracts on EOS does not need to consume tokens. Instead, EOS consumes resources including bandwidth, CPU, and RAM, which makes EOS more suitable for the IoT and IoV. EOS supports issuing new tokens with eosio.token contract (no need to script new contracts) and transferring tokens to another account by the issuer's account. So we choose EOS as the smart contract platform for ISC-MTI.

4 Threat model

Since third-party servers and third-party cloud storage are untrusted, the training data and trained models in the storage are not safe. In ISC-MIT, we assume data owners are credible, which means the data provided by the data owner are all trusted data. Model trainers and model users are not completely credible, who may maliciously use data and models without any cost. Man-inthe-MiddleAttack (MITM) may be encountered during the data upload and invocation phase.

Table 1 Comparison of current decentralized file systems

File system	OS	Decentralization	Retrospective file history	Automatic de-redundancy	Anti-DDOS ability
HDFS	Unlimited	low	not support	Partial support	Low
GPFS	Linux\Windows et al.	low	not support	not support	middle
Ceph	Linux	middle	not support	not support	middle
IPFS	Unlimited	Complete	Support	Support	high

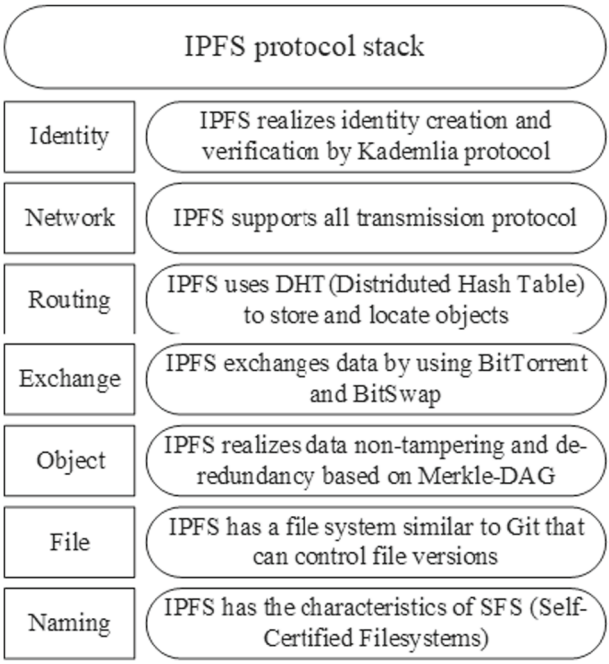

Fig. 2 IPFS protocol stack

Table 2 Comparison of current smart contract platforms

Platforms	Execution context	Programing language	Consensus algorithm	Ideal TPS	Calculate ability demand
Bitcoin	Null	Scripting language	Pow	Below 10	high
Ethereum	EVM	Solidity\LLL\Serpent	Pow\Pos	30~40	high
Hyperledger	docker	Go\Java	Multiple	3000	middle
EOS	WASM VM	C++	DPoS	Million	low

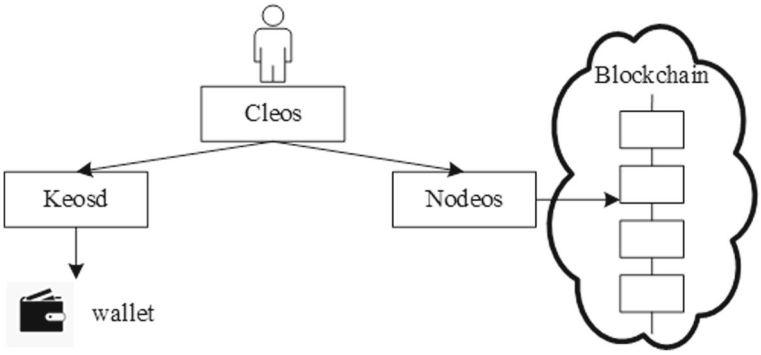

Fig. 3 IPFS protocol stack

The capability of adversaries is as follows:

- adversaries can infer the training data by invoking the trained model multiple times.
- adversaries can steal non-volatile data such as the data in hard disk.
- adversaries can intercept network data communicated between the users and IPFS.

5 ISC-MTI framework

This section presents the detailed description of ISC-MTI. Figure 4 illustrates the architecture of ISC-MTI.

ISC-MTI still involves three roles: data owner, model trainer, and model user. Data owners upload training data to IPFS, and IPFS will return the storage credentials of the training data (i.e., unique hash values). Model trainers read training data by smart contracts and train machine learning models in memory. After model training, data such as the structure and parameters of the models will be uploaded to IPFS. For defending MITM, model data will be encrypted by RSA encryption algorithm and AES encryption algorithm. After the model data is uploaded to IPFS, the storage credentials will also be returned. Model users read and decrypt trained models by smart contracts and private keys. Without saving the model file to the local disk, input the test samples prepared by yourself to implement the model's invocation.

In ISC-MTI, no trained model files will appear in the local disk or third-party cloud storage. The design maximizes the security of machine learning models by mandatory decentralization and non-tamperable features of IPFS [21]. If there is a need for higher security, the invocation results can be stored in the IPFS and the keys of encryption algorithms can be saved on the blockchain.

5.1 Design of smart contract

In ISC-MTI, the invocations of training data and models are completed by Transactions. Since we adopt EOS, a Transaction involving training data and trained model invocation consists of one or more Actions. After the user submits a transaction, the corresponding

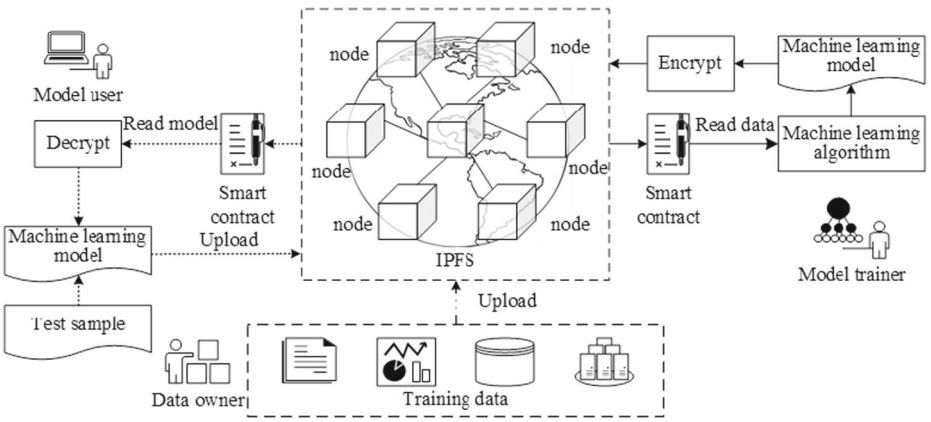

Fig. 4 Architecture of ISC-MTI

smart contract is responsible for processing the transaction in the WASM virtual machine. The execution process of the smart contract is shown in Fig. 5.

The enforceable and non-tampering feature of smart contracts can protect the invocation process of training data and models. These smart contracts will be deployed to an administrator account. Administrators need to use eosio.token to issue their own tokens. The standard for issuing tokens is ERC20. The model trainers and model users can purchase the invocation right of training data and trained model by transferring tokens. The pseudo code of Action responsible for invoking resources in the smart contract is shown in Algorithm 1.[1]

Algorithm 1 Resources Invocation Action.

Data: Resource name X, payment account Y
Result: Changes in transaction status and account status
Use X to get the resource price p_r and storage credential c_s;
Use Y to get account balance p_a;
if $p_a \geq p_r$ **then**
    Transfer from Y to the model trainers account with p_r tokens;
    if *Successful transfer* **then**
        return c_s encrypted by payer's public key;
    else
        Interrupt the transaction and throw a error message about transfer error;
    end
else
    Interrupt the transaction and throw a error message about insufficient balance;
end

When a user invokes resources, the user needs to send the Resources Invocation Action to the smart contract deployed on the administrator account. Resources Invocation Action stipulates that users must pay enough tokens to the model trainers to get the storage credentials of the resource. If the transfer fails, the corresponding error message will be thrown and the transaction will be interrupted.

5.2 Encrypted transmission

The storage credentials in IPFS can be generated by user-defined hash algorithms. But the hash value is still transmitted in plaintext. Attackers can steal or destroy the hash value by MITM to steal or destroy training data and trained models[32]. MITM for IPFS includes Network Sniffing, Domain Name System Attack, and Address Resolution Protocol Attack, as shown in Fig. 6.

To solve the problem, we use RSA encryption algorithm and AES encryption algorithm to encrypt training data and models. The key of RSA is long and has good security, but the computation of RSA is large. AES has low memory requirements and fast encryption speed. Considering the efficiency of model training, we use AES to encrypt data and use RSA to encrypt AES's key.

The storage credentials of resources were encrypted by payers' public key. Assuredly, payers can decrypt them by own private key when they use the resources.

[1]Resources refers to training data or trained models in ISC-MTI

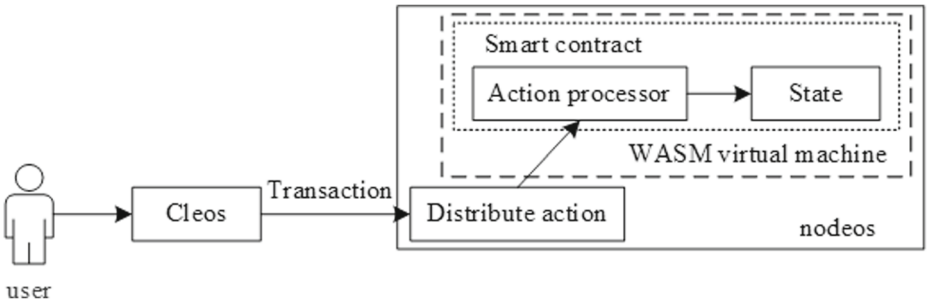

Fig. 5 Execution process of smart contract

5.3 Model training, model upload and model invocation

We all know that the data security in hard disk is much lower than that in memory. In order to prevent the model files from being saved to the local disk, different training, uploading, and invocation schemes of models need to be designed in the light of model file types. Combining the content of 4.2 sections and 4.3 sections, related pseudo code about training, uploading, and invocation of models is shown in Algorithm 2.

Algorithm 2 Model training, uploading and invocation.

Data: IP address, gateway, port number and other information API, resource name X, payment account Y, parameter of model training p_t, private key key, test sample T

Result: prediction results corresponding to test sample

Use X and Y to get storage credentials of training data c_s^1;
Read training data d_t by c_s^1 `// Start train models`
Train models M by using d_t `// Start upload models`
if *algorithm library used is Scikit-learn* **then**
    Convert M to bytes data, encrypt data using RSA and AES, upload encrypted data to IPFS and return storage credentials c_s^2;
end
if *algorithm library used is TensorFlow* **then**
    Convert computational Graph of M to bytes data , encrypt data using RSA and AES, upload encrypted data to IPFS and return storage credentials c_s^2;
end
if *algorithm library used is Keras* **then**
    Split M into two parts: model structure and model internal parameters, and convert them into json objects, encrypt data using RSA and AES, upload encrypted data to IPFS and return storage credentials c_s^2;
end
Use X and Y to get storage credentials of trained models c_s^3 `// Start invoke models`
Read encrypted trained models by c_s^3 and decrypt it key;
Assemble M in memory with the decrypted result according to algorithm library;
Use M with T return prediction results;

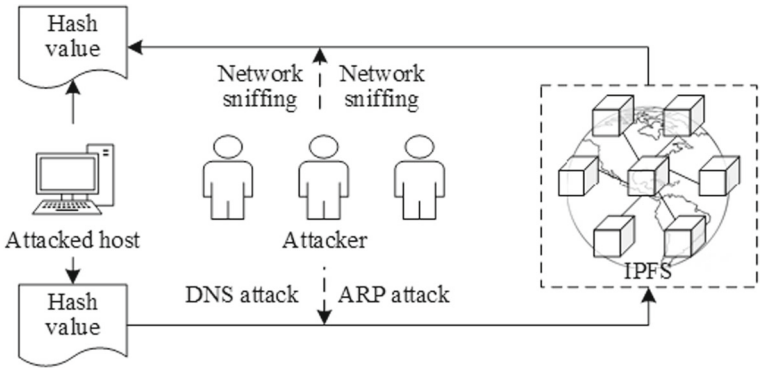

Fig. 6 MITM for IPFS

In particular, for TensorFlow, the convert method is designed according to the pb file, which is the way Google recommends to save the TensorFlow models.

Due to AES is used to encrypt data, the computation complexity of algorithm 2 mainly depends on AES. Obviously, the computation complexity of AES is $O(n)$. Where n is the length of the plaintext.

6 Experiment

6.1 Experimental environment and experimental data

We use Python and C++ to implement ISC-MTI and perform necessary performance analysis. The experiment computer has i7-8700 CPU, 16G memory, NVIDIA GT 730, 500M broadband and installed CentOS-6.10x64, python 3.7.1, go-ipfs 0.6.0. The smart contract platform is EOS test chain.

For verifying the availability of ISC-MTI comprehensively, we choose multiple data and algorithms to perform experiments. The data and algorithms used in the experiment are shown in Table 3[2]. For all parameters of machine learning, we set them as the default values in the algorithm libraries.

6.2 Performance experiment

In the experiment, we verify the influences of using IPFS and smart contract for the model training process on the trained model accuracy and training efficiency. We compare ISC-MTI with the ordinary training method based on the accuracy and calculation time. The calculation time includes the time consumed by all steps such as data reading, model training, model uploading, and model invocation3. The ratio of the training set to the test set is 7:3. The number of iterations is all set to 100. In the performance experiment, the calculation formula of the accuracy for classification is as follows:

$$accuracy = \frac{n_{ture}}{n}. \tag{1}$$

[2]The IPFS storage credential of dataset is QmeAKaopjP2g9Ni5jXqwpcv6BB4fnkMAZDLy6X2fpsPcQH

Table 3 Datasets, algorithms and models used in experiments

No.	Dataset detail	Dataset size	Algorithm	Function	Algorithm library	Model size
1	Sepal length, width and petal length, width data of Iris	4 kb	SVM	classification	Scikit-learn	5 kb
2	Time series data of electric power	48 kb	LSTM	regression	Keras	21 kb
3	Time series data of traffic volume	152 kb	RVM	regression	Scikit-learn	2294 kb
4	Text data of all kinds of news	236342 kb	GRU	classification	Keras	15465 kb
5	Image data of variety of vehicle	3780688 kb	CNN	classification	TensorFlow	62259 kb

Where n is the number of samples; n_{ture} is the number of samples correctly divided. The calculation formula of the accuracy for regression is as follow

$$accuracy = 1 - \left[\sum_{i=1}^{n} \left| (x_i^{pred} - x_i^{ture}) / x_i^{ture} \right| \frac{100\%}{n} \right]. \tag{2}$$

Where x_i^{pred} is predictive value; x_i^{ture} is ture value. Due to the exchange between data and IPFS as well as smart contracts in ISC-MTI, some training efficiency will be lost. For measuring the loss, we define an efficiency loss rate L, which is as follow

$$L = \frac{t_{ISC} - t_{ord}}{t_{ord}}. \tag{3}$$

Where t_{ISC} is the time to train a model using ISC-MTI; t_{ord} is the time to train a model using ordinary training method (see Fig. 1). The results of performance analysis experiment are shown in Table 4. The time difference between using original training method and ISC-MTI with the size of dataset and model is shown in Fig. 7.

In the case of small data size and model scale, using ISC-MTI will significantly reduce the training efficiency, simultaneously, L is very large. In the case of large data size and model scale, for example, the vehicle image data and 62259 kb model in test number 5, L will gradually decrease. This phenomenon is more obvious in Fig. 7. Due to the computation complexity of ISC-MTI mainly depends on the encryption algorithms, the larger the model size, the more additional training time. Besides, if the experiments are conducted on the EOS permissionless blockchain, L will have a certain degree of reduction because the performance of the EOS permissionless blockchain is significantly higher than the EOS test chain.

The second time-consuming factor in ISC-MIT is data uploading to IPFS. In terms of IPFS transmission speed, the maximum upload speed can reach 20.08 MB/S. The value is calculated when the file is uploaded to IPFS for the first time. Because of the de-redundancy function in IPFS, the first storage of data is the slowest. The time used to read data is very short and negligible. The result shows that ISC-MIT can meet user requirements for transmission speed.

In terms of accuracy, experimental results prove that using ISC-MTI will not affect the performance of the machine learning model. The change in accuracy does not exceed 0.2%. In general, ISC-MTI is a training method that trades training efficiency for training safety and data safety——the longer time using ordinary training, the smaller negative impact of using ISC-MTI on training efficiency, which is all within the acceptable range of model trainers and model users.

Table 4 Performance test results of ISC-MTI

No.	Ordinary training		ISC-MTI		L
	Accuracy	Time s	Accuracy	Time s	
1	85.82	0.13	85.82	9.84	7469
2	99.03	23.67	99.14	39.93	68.69
3	85.75	134.19	85.75	148.46	10.63
4	96	121611	96.3	121638	0.022
5	99.5	145716	99.7	145790	0.05

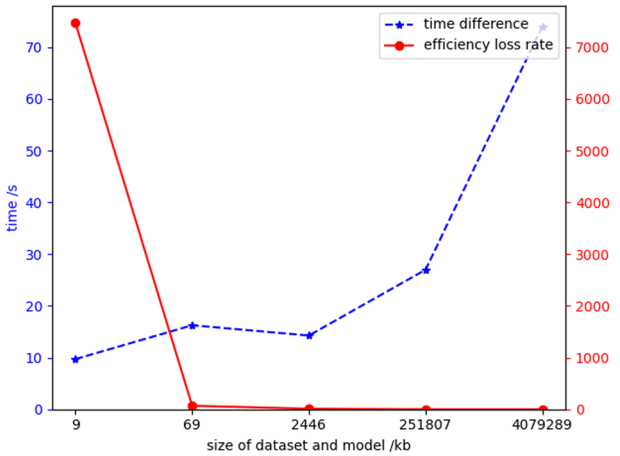

Fig. 7 Time difference between using the original training method and ISC-MTI

6.3 Smart contract experiment

This experiment verifies the usability of smart contracts designed by us. In the experiment, the token name is HAPY; the resource price p_r is 10 HAPY; the model invocation account is Tommy; the model training account is linhao; the account balance of tommy is 100 HAPY. After the smart contract is compiled and deployed, tommy sends Resources Invocation Action to the smart contract. The running result is shown in Fig. 8.

Since the account balance of tommy is greater than the p_r, the transaction should be successful. According to the smart contract in Algorithm 1, tommy needs to transfer 10 HAPY to linhao. The transfer record is shown in Fig. 9.

It can be seen from the above test results that the smart contract designed in this article has achieved the expected goal well.

6.4 Safety analysis

Ability to prevent model theft In ISC-MTI, no model files are saved in the local disk or traditional distributed cloud storage. Smart contracts are used for mandatory access control of resources. The attackers cannot directly steal the model files.

Tamper-proof In ISC-MTI, training data and trained models are stored in IPFS in blocks. IPFS nodes are located all over the world and IPFS can ensure that the number of data backups is sufficient. Unless attackers break all nodes simultaneously, training data and trained models cannot be deleted and tampered with.

```
executed transaction: 32a4e4a5c328d1d7560824524b93c62f35cb5fe85e1cc4b33a495effcf923c86  104 bytes  1626 us
#        linhao <= linhao::call                {"actor":"tommy"}
>> Pay token successfully. IPFS Storage token => QmSsw6EcnwEiTT9c4rnAGeSENvsJMepNHmbrgi2S9bXNJr
warning: transaction executed locally, but may not be confirmed by the network yet
```

Fig. 8 Running result of Resources Invocation Action

```
executed transaction: b72a23ea1ae8224186d8ae9c59a3fa53b1f31dbaefde7741fa56dee8a6c604e7  128 bytes  646 us
#   eosio.token <= eosio.token::transfer        {"from":"tommy","to":"linhao","quantity":"10.00 HAPY","memo":""}
#         tommy <= eosio.token::transfer        {"from":"tommy","to":"linhao","quantity":"10.00 HAPY","memo":""}
#        linhao <= eosio.token::transfer        {"from":"tommy","to":"linhao","quantity":"10.00 HAPY","memo":""}
```

Fig. 9 Transfer record of tommy

Traceability ISC-MTI uses the version control capability of IPFS to trace the full version of data deployed on the IPFS. The transactions involved in the smart contracts are also fully recorded on EOS.

Reliability and privacy The data in ISC-MTI is stored in completely decentralized storage, which can effectively resist DDOS attacks. Simultaneously, it effectively avoids the failure of single node. Since there is no need to introduce TPA for data auditing, data privacy can be guaranteed.

7 Conclude

We constructed a safe framework for machine learning training and invocation called ISC-MTI. We performed related experiments to analyze the impact of using ISCMTI on the model accuracy, training efficiency, and safety of model training. We concluded that, in the case of a slight loss of training efficiency, ISC-MTI can provide machine learning model training with anti-theft model capabilities, traceability, tamper resistance, reliability, and privacy.

Potential applications Our framework can come in handy in the applications of machine learning which relying on big data. We give a few examples heuristically, as shown below

- recognize COVID-19 based on Medical data sharing [27].
- road congestion prediction in IoV [10].
- user credit assessment using transaction data in the banks.

Limitations We did not consider the credibility of the resources in transaction, that is, did not design the pre-buy communication mechanism between data owners, model trainers and model users. Furthermore, there is a lack of credit evaluation and incentive mechanism for data owners, model trainers and model users in our ISC-MTI.

Future work Our further research will include:

- Study the application of ISC-MTI in large-scale networks and deep models. Reduce L in the case of small data size.
- Study the combination of ISC-MTI and swarm, which is a decentralised storage and communication system for a sovereign digital society.
- Consider the security of smart contracts and improve the security of ISC-MTI at the contract layer. Moreover, we will focus on the dynamic security of smart contracts.

Acknowledgements This work was supported in part by the Inner Mongolia Key Technological Development Program under Grant (2019ZD015, 2019ZD016), in part by the Key Scientific, Technological Research Program of Inner Mongolia Autonomous Region under Grant (2019GG273, 2020GG0094), and in part by the Inner Mongolia Autonomous Region Special Program for Engineering Application of Scientific and Technical Payoffs under Grant 2020CG0073. The authors thank Prof. Chundong Wang from Tianjin University of Technology for his comments and revision on the manuscript.

Declarations

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interests All authors declare that they do not have any conflict of interest.

References

1. Chatterjee I (2021) Artificial intelligence and patentability: review and discussions. Int J Modern Res 1(1):15–21
2. Elomari A, Maizate A, Hassouni L (2016) Data storage in big data context: a survey. In: Proceeding of 2016 Third International Conference on Systems of Collaboration (SysCo), pp 1–4. https://doi.org/10.1109/SYSCO.2016.7831344
3. Fredrikson M, Jha S, Ristenpart T (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of ACM Conference on Computer and Communications Security, pp 1322-1333, ACM. https://doi.org/10.1145/2810103.2813677
4. Haiwu H, An Y, Zehua C (2018) Survey of smart contract technology and application based on blockchain. J Comput Res Develop 55:2452–2466. https://doi.org/10.7544/issn1000-1239.2018.20170658
5. Haoyu G, Leixiao L, Hao L, Jie L, Dan D, Shaoxu L (2021) Research and application progress of blockchain in area of data integrity protection. J Comput Appl 41:745–755
6. Jian L, Rui SP, Min Y, Liang HE, Yuan Z, Yang ZX, Min LH (2018) Software, School Of and University, Fudan, software and cyber security——a survey. J Softw 29:42–68. https://doi.org/10.13328/j.cnki.jos.005320
7. Jiang W, Gong Z, Zhan J, He Z, Pan W (2020) A low-cost image encryption method to prevent model stealing of deep neural network. J Circ Syst Comput 2050252:29. https://doi.org/10.1142/S0218126620502527
8. Jin X, Huici W, Xiaofeng T (2020) 5G cyberspace security game. J Electron Inf Technol 42:2319–2329. https://doi.org/10.11999/JEIT200058
9. Khoda M, Imam T, Kamruzzaman J, Gondal I, Rahman A (2020) Robust malware defense in industrial IoT applications using machine learning with selective adversarial samples. IEEE Trans Ind Appl 56:4415–4424. https://doi.org/10.1109/TIA.2019.2958530
10. Li L, Lin H, Wan J, Ma Z, Hui W (2020) MF-TCPV: a machine learning and fuzzy comprehensive evaluation-based framework for traffic congestion prediction and visualization. IEEE Access 8:227113–227125. https://doi.org/10.1109/ACCESS.2020.3043582
11. Liu X, Li H, Xu G, Liu S, Liu Z, Lu R (2020) PADL privacy-aware and asynchronous deep learning for IoT applications. IEEE Inter Thing J 7:6955–6969. https://doi.org/10.1109/JIOT.2020.2981379
12. Lu J, Issaranon T, Forsyth D (2017) SafetyNet detecting and rejecting adversarial examples robustly. In: Proceedings of 2017 International Conference on Computer Vision (ICCV), pp 446-454, IEEE. https://doi.org/10.1109/ICCV.2017.56
13. Jinyin C, Yan Z, Xueke W (2020) a survey of attack, defense and related security analysis for deep reinforcement learning. Acta Automatica Sinica. https://doi.org/10.16383/j.aas.c200166
14. Pooya T, Mehran Y, Reza KM (2018) Robust cascaded skin detector based on AdaBoost. Multimed Tools Appl 78:2599–2620. https://doi.org/10.1007/s11042-018-6385-7
15. Rahman NHA, Choo K-KR (2015) A survey of information security incident handling in the cloud. Comput Secur 49:45–69. https://doi.org/10.1016/j.cose.2014.11.006
16. Ribeiro M, Grolinger K, Capretz MAM (2015) MLAas: machine learning as a service. In: Proceedings of the 14th International Conference on Machine Learning and Applications (ICMLA), pp 896-902, IEEE, Miami, FL, USA. https://doi.org/10.1109/ICMLA.2015.152
17. Ruchika, Purwar RK, Verma S et al (2021) Crowd abnormality detection in video sequences using supervised convolutional neural network. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-11781-4
18. Sharma A, Schuhknecht FM, Agrawal D, Dittrich J (2019) Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In: Proceedings of the 2019 International Conference on Management of Data, pp 105–122, ACM, New York, NY, USA. https://doi.org/10.1145/3299869.3319883

19. Shrivastava V, Kumar S (2019) Utilizing block chain technology in various application areas of machine learning. In: Proceeding of international conference on machine learning, big data, cloud and parallel computing (COMITCon), pp 167–171, 2019. https://doi.org/10.1109/COMITCon.2019.8862203
20. Suliman1 A, Husain1 Z, Abououf1 M, Alblooshi1 M, Salah K (2019) Monetization of IoT data using smart contracts. IET Netw 8:32–37. https://doi.org/10.1049/iet-net.2018.5026
21. Sun J, Yao X, Wang S, Wu Y (2020) Blockchain-based secure storage and access scheme for electronic medical records in IPFS. IEEE Access 8:59389–59401. https://doi.org/10.1109/ACCESS.2020.2982964
22. Sun LL, Li H, Yu SW, Wang YX (2020) A survey on encrypted image recognition models. J Cryptol Res 7:525–540. https://doi.org/10.13868/j.cnki.jcr.000387
23. Sun ZX, Zhang X, Xiang F, Chen L (2021) Survey of storage scalability on blockchain. J Softw 32:1–20. https://doi.org/10.13328/j.cnki.jos.006111
24. Tan ZW, Zhang LF (2020) Survey on privacy preserving techniques for machine learning. J Softw 31:2127–2156. https://doi.org/10.13328/j.cnki.jos.006052
25. ul Haque A, Ghani MS, Mahmood T (2020) Decentralized transfer learning using blockchain IPFS for deep Learning. In: Proceeding of 2020 International Conference on Information Networking (ICOIN), pp 170-177, IEEE. https://doi.org/10.1109/ICOIN48656.2020.9016456
26. Ur Rahman M, Baiardi F, ricci L (2020) Blockchain smart contract for scalable data sharing in IoT: a case study of smart agriculture. In: Proceedings of 2020 global conference on artificial intelligence and internet of things (GCAIoT), pp 1-7, IEEE. https://doi.org/10.1109/GCAIoT51063.2020.9345874
27. Vaishnav PK, Sharma S, Sharma P (2021) Analytical review analysis for screening COVID-19 disease. Int J Modern Res 1(1):22–29
28. Wei Z, Bing B, Hongwei W (2019) An intrusion detection method of data tampering attack in train control system based on KF. China Safety Sci J 29:32–37. https://doi.org/10.16265/j.cnki.issn1003-3033.2019.S1.007
29. Xiaofeng M, Lixin L (2021) Blockchain-based data transparency: issues and challenges. J Comput Res Develop 58:237–252. https://doi.org/10.7544/issn1000-1239.2021.20200017
30. Xu J, Ma L (2020) Application of blockchain technology in distributed energy transaction. Electr Power Autom Equip 40:17-22+30. https://doi.org/10.16081/j.epae.202008001
31. Xu G, Li H, Liu S, Yang K, Lin X (2020) VerifyNet secure and verifiable federated learning. IEEE Trans Inform Forensic Secur 15:911–926. https://doi.org/10.1109/TIFS.2019.2929409
32. Xu XW, Cai B, Xiang H, Sang J (2020) Multinomial logistic regression model based on homomorphic encryption. J Cryptol Res 7:179–186. https://doi.org/10.13868/j.cnki.jcr.000359
33. Yingchao Y, Lin D, Zuoning C (2018) Research on attacks and defenses towards machine learning systems. Netinfo Secur 18:10–18. https://doi.org/10.3969/j.issn.1671-1122.2018.09.002
34. Yujin Z, Jianguo Y, Haibing G (2020) Blockchain as a service: next generation of cloud services. J Softw 31:1–19. https://doi.org/10.13328/j.cnki.jos.005891
35. Zhang Y, Kasahara S, Shen Y, Jiang X, Wan J (2019) Smart contract-based access control for the internet of things. IEEE Inter Thing J 6:1594–1605. https://doi.org/10.1109/JIOT.2018.2847705