EasyChair Preprint

№ 764

Blockchain-Based platform for Distribution AI

Lifeng Liu, Chao Wu and Jun Xiao

February 2, 2019

Blockchain-Based platform for Distribution AI

Lifeng Liu
Zhejiang University
Zhejiang University, Zhejiang province, China
13588312894, 086
Liu_lf@zju.edu.cn

Chao Wu
Zhejiang University
Zhejiang University, Zhejiang province, China
17758020300, 086
chao.wu@zju.edu.cn

Jun Xiao
Zhejiang University
Zhejiang University, Zhejiang province, China
13867424906, 086
junx@cs.zju.edu.cn

ABSTRACT
In recent years, the current artificial intelligence exposed user data privacy during training and the high cost of training are getting more and more attention, which are becoming an obstacle to the development of AI. We identify the main issues as data privacy, ownership, and exchange and model privacy, which are difficult to be solved with the current centralized paradigm of machine learning training methodology or federating learning methodology. As a result, we propose a practical model training paradigm based on Blockchain, named Distributed AI, which aims to train a model with distributed data and to reserve the data ownership for their owners and the interest of trained model. In this new paradigm, we use Blockchain[3] as the base architecture in which we abstract different actors (i.e., model provider, data provider) taking different actions to archive own target, realize distributing encrypted model training by Federating Learning with different actors, set smart contract as model training infrastructure, set up notification server, pricing of training data is according to its contribution and therefore it is not about the exchange of data ownership.

Keywords
Blockchain · Distributed AI · Smart Contract · Actors · Encrypted

CCS Concepts
• Mathematics of computing~Distribution functions

1. INTRODUCTION
With the widespread of technology and the internet: AI is becoming the new engine for the rapid development of productivity, with (big) data as its fuel. With the expanding utility of deep learning [9, 10], large deep learning architectures are quite data hungry, and therefore the importance of data has grown even more. Data have become a type of "new money" in the digital world [8].

However, in the current centralized computation paradigm, where data are typically collected from the end users (and from various sensors) and uploaded to a remote server (or a cluster of servers) for data analysis and modelling, there exists a significant gap between data and model (i.e., a long distance before data can be utilized in application and decision making), because of the following issues:

- **Centralized Cost**: Modelling with a large amount of data incurs a high cost, even on a cloud-based infrastructure. Model developers need to afford the high cost for the computation and storage for the model training and its deployment. Although big companies can afford it, such high cost brings the barrier for individual developers and startups.
- **Security and privacy**: It is risky to host the data (especially user data) on the servers, and it also brings the cost for maintaining the security. More importantly, such centralized data storage and computation have severe privacy issue. Users need to give out their valuable and/or sensitive data to the third parties, and even to some malicious parties if the server was hacked.
- **Ownership**: As users give out their data to the central server, they lose the data ownership, because they cannot further control the usage of data. Therefore, it is difficult to establish a reasonable user incentive in such a paradigm.
- **Single point of failure**: Centralized modelling schema suffers the risk of eventually being litigated out of existence, or failing in crashes when the parent company liquidated abruptly.

Facing these challenges, the Federating Learning has been created to reserve the privacy of data from data owner. In that way, instead of collecting data in centralized context, data are remained at where they belong to, meanwhile we move modelling towards them. Therefore, users' privacy and data ownership are preserved.

However, we still couldn't protect the privacy of model and didn't take the high computation cost of data owner for training data into count. Therefore, we design the new paradigm which could solve above problems.

The main contributions of the paper can be summarized as follows:

- **Blockchain as the base infrastructure**: We use Blockchain as the base infrastructure where everyone can create Ethereum account. Realize highly decentralized machine learning on Blockchain and abstract three types actors to collaborate learning task. And we use Blockchain to secure the ownership of digital assets.
- **Blockchain with smart contracts as decentralized modelling infrastructure**: we regard smart contract as a modelling infrastructure to publish a modelling task, training and aggregated command, and rewarding strategy.
- **Moving encrypted model to data with protection**: By deploying the encrypted model to Blockchain, we move the model towards data, instead of moving data to model as in traditional approach. In addition, we shall further protect model by applying Homomorphic Encryption (i.e., HE [6]) and Multi-Party Computation (i.e., MPC [4]).
- **Lowering cost of training**: We can train model by Federating Learning[1,2] which was improved and further lower cost of model training. Cost of building a centralized computation infrastructure is distributed among participating nodes.

2. RELATED WORK

This section discusses with a short review of Blockchain and the peer-to-peer distributed file systems, which is crucial in our platform to tackle the data accessibility problem.

2.1 Blockchain

The blockchain is an implementation of the distributed ledger technology (DLT). It is tamper-proof with no central governance. Blockchain revolution can be divided into 3 stages [13]. Blockchain 1.0 is the cryptocurrency for economic transactions. Bitcoin is regarded as the most famous cryptocurrency at the moment [14]. Whereas, Blockchain 2.0 utilises smart contracts technology. Smart contracts can be seen as a programme that can act as a trusted third party authority during transactions. Successful projects include Ethereum [3] for open blockchain and Hyperledger Fabric for closed blockchain. Blockchain 3.0 is blockchain applications beyond currency, finance and markets.

2.2 P2P Distributed File System

Data storage is a challenging problem in Federating Learning. Data is usually large in size, sampled frequently in a distributed way. There have been many studies on global distributed file system since 2000s .

InterPlanetary File System(IPFS) is a P2P DFS system, which could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. There are several unique features of IPFS. Firstly, it uses the Merkle DAG data model, which can split large data files (> 256kB) into a list of links to file chunks that are < 256kB. This makes the system suitable for large data sets. Secondly, the hash of the root chunk can be used to both ensure the immutability of its content and retrieve the underlying data [15]. Therefore this hash can be used as our hash pointer on chain.

3. DESIGN OVERVIEW

In this section, we present DAI as a new paradigm of machine learning training. The ideas discussed in Section 1 will all be covered.

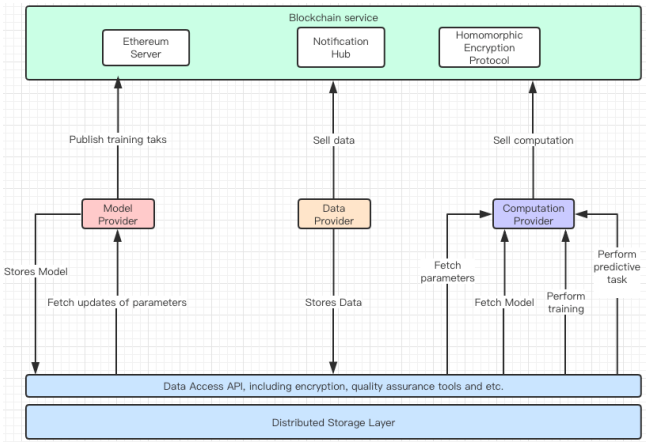

Figure1: new paradigm overview

3.1 Actors Design

First of all, from above Figure1, there are three types of actors within this distributed modelling paradigm:

1. **Data provider**: Data provider (shown in Figure 2) is typically the end user (individuals, companies, or any organizations having their own data), willing to utilize their data for exchanging services or other incentives. They collect data from sensors or other data sources. Before the data is contributed to modelling, data provider needs to evaluate and provide the quality measure of the data, as well its schema. We need to emphasize that when we say "data providers provide their data", that does not mean data is given out to some other parties. It means the data is granted for being used by modelling.
2. **Model provider**: Model provider (or data user, or data requester, as shown in Figure 3) develops and distributes machine learning models to utilize the data from data providers. It can be an initial model without any training, or an existing model pre-trained by the model provider. The model provider also acts as a training task provider, who initializes a model training task with test data to evaluate model update, schema for required training data, and reward plan for training data, in addition to the model itself. While in decentralized machine learning paradigm, they only need to provide their model as smart contracts to computation providers.
3. **Computation provider**: Computation provider (as shown in Figure 4) is a node in the network to run the smart contract for model training. It provides a secure and controlled training environment where both data and model are protected. A model training task is distributed among multiple computation providers as a federated learning task. Please note that any node in the Blockchain network can become a computation provider if it's willing to provide computation (ranging from high performance GPU cluster to mobile devices), even when it is also a data provider or model provider.

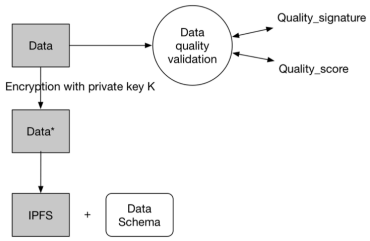

Figure 2: Data Provider.

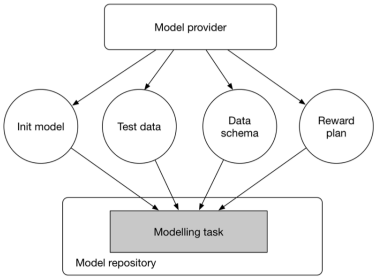

Figure 3: Model provider.

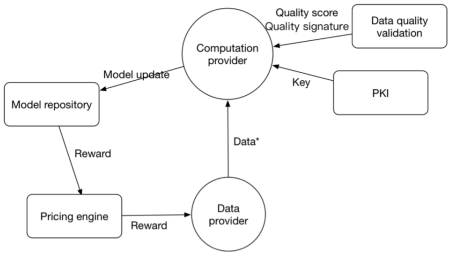

Figure 4: Computation provider.

3.2 Blockchain Network

In addition to these types of actors, there are base supporting nodes in the Blockchain network:

1) **Notification:** Notification takes the responsibility of communication and event triggers; when Blockchain generate a new block, notification Hub will generate a new goroutine to listen it. It will listen to all kinds of nodes in Blockchain communication and events.

2) **Blockchain:** Blockchain node records the addresses of participating nodes and their models/data.

3) **Smart Contract:** Smart Contract on Blockchain as model training infrastructure. A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties.

3.3 Training Process Design

Now we will propose the workflow of the model training on Blockchain-Based platform named Distribute AI where all kinds actors will take participate and collaborate with others to archive to train a secure model with encrypted data in Blockchain context.

1. **Blockchain Infrastructure：** In this system, we use blockchain as a communication bridge and context. There are a few factors to consider. Firstly we would like to ensure that all actors (every actor presents one Ethereum account) will freely join and exit the platform at any time, and equally participate in this data transaction network. Secondly as this is a public system with token trades, we would like to use PoW as our consensus to benefit from its highest level of security.

2. **Preparation:** Once actor enters network, it should generate one Ethereum account to execute actions. After that, instead of using sign with RPC, data provider choose to sign offline digital signature to protect privacy key. Then it will generate a data schema file to upload IPFS which will return hash string. Get hash, data provider will generate a offline transaction. Finally it will send transaction to the Ethereum and wait for mining. In addition, Notification hub will receive this action and record it. At this term, computation provider does the same way as data provider. Of course, it will send a computation specification file.

3. **Model Provisioning：** To protect model, model provider firstly encrypted initial (untrained or pre-trained) model weights and structure. Attention, this "initial model" also can refer to model which has been trained in other training context. Then, model provider will upload this encrypted model to IPFS and send hash to Ethereum. In addition, it will initiate smart contract which contains a modelling task. A modelling task has following item: 1) test data for evaluating performance; 2) evaluate performance (e.g., accuracy) of the trained model, with test data; 3) data schema for training data; 4) Rewarding strategy, a plan to determine the contribution of data providers and their rewards.

4. **Model Training:** Once a modeling task is published, a smart contract will be executed to match data providers. With N matched data providers, the task is distributed to N sub-tasks, with an identical initial model for different training data. The same methodology of federated learning is adopted here for distributed training, while the training task is taken on computation providers (e.g., EVM nodes in Ethereum network). The computation provider provides a secure sandbox for model training. Encrypted data and model is transferred to this computation provider. Then, computing provider will train model locally.

5. **Aggregate Model:** After locally training task finished, computation providers will send the update model to IPSF and send hash to Ethereum. Model provider will download the model from IPFS by hash. Then it will aggregate model. Doing some epochs, model provider get the steady global model.

6. **Assign Reward:** Finally, model provider will evaluate performance (e.g., accuracy) of the trained global model with test data and assign reward to data provider and computation provider, according to the contribution of actor.

4. PLATFORM IMPLEMENTATION

In this section, the details of the platform implementation will be discussed. The whole workflow of the system is illustrated in Figure 4. Whisper, which provides dark (plausible denial over perfect network traffic analysis) communications to two correspondents that know nothing of each other but a hash, is used to signal to each other in order to ultimately collaborate between nodes. All tokens used within the workflow are transferred according to the smart contract. As show in Figure5.

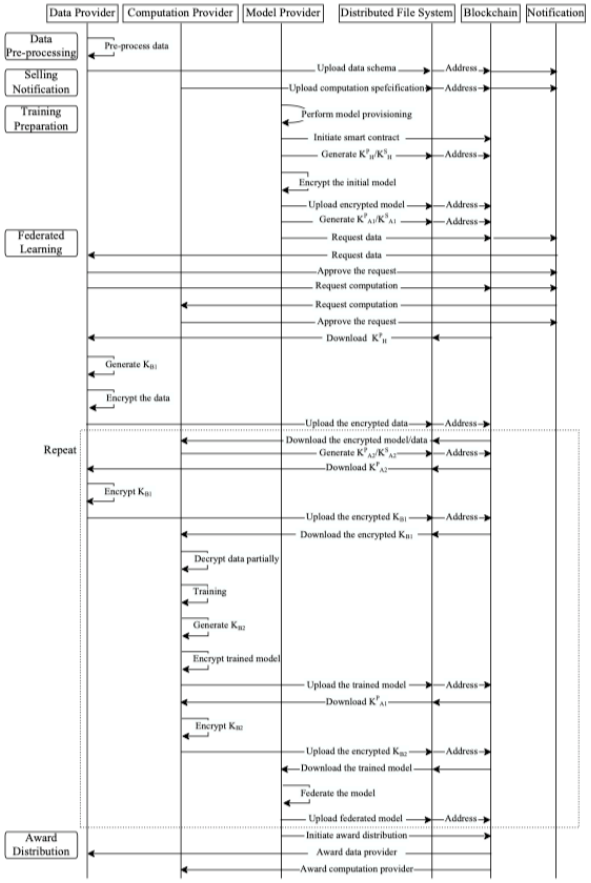

Figure 5: An illustration of system workflow. Herein, K_H^P /K_H^S represent the pair of homomorphic encryption keys, $K_{A_i}^P$ / $K_{A_i}^S$ represent the pair of asymmetric encryption keys, and K_{B_i}represents the symmetric encryption key.

4.1 Data pre-processing

Data pre-processing is performed offline locally by the data providers. There are several sub-tasks involved in this step:

1. Sharding is performed to divide data into small portions. There are mainly two reasons: firstly, it is difficult to scale with big chunks of data; secondly, some data consumers might only wish to purchase a small subset of data, instead of everything. Sharding enables micropayments in the system.
2. Description are calculated using a commonly agreed quality measurement function. The quality measurement function can take inputs from multiple parties. When the function is simple, methods such as HE and MPC can be used to preserve privacy.
3. Metadata (such as location, device id, data description, etc.) are prepared.

4.2 Selling notification

The Data/Computation Providers notify the network that they are willing to sell the data/computation.

1. Digital signature algorithms are widely used in Ethereum. At present, there are at least two known ones: one is to digitally sign the entire tx object when generating each transaction (Transaction, tx), and the other is the Clique algorithm in the consensus algorithm. We can signature offline prevent to expose our secret key to Ethereum.
2. Data providers firstly store data schema (including quality scores, metadata, and cost) on the distributed file system. The schema addresses are broadcasted to all nodes in the network. Any node can locate and view schema using the corresponding address.
3. Similarly, computation providers store the specification of their computing resources (including type of resource and cost) on the DFS. The specification addresses are broadcasted to all nodes in the network. Any node can locate and view the specification using the corresponding address.

4.3 Training preparation

Model provider prepares the training by performing the following steps:

1. We set up Ethereum context by forking Go Ethereum (Geth 1.8.13) and deployed a private testnet. In this version of Ethereum, it uses PoW as the consensus mechanism. Smart contracts in this study were written using Solidity Version > 0.4.0. On-chain data only includes three Ethereum addresses (authority, provider and consumer), four 32 bytes IPFS data addresses, a unsigned integer expiration time property, and two Boolean flags. Variable definitions and function signatures are shown in Appendix 1. Gas cost for the smart contract is shown in Table I. Note that functions calls the 'transfer()' function could not determine the maximum cost, but 21000 is commonly seen for payable transfers that do not have additional data. The gas prices comparably low as we store the minimum amout of data on-chain.
2. Model provider locally generates a pair of HE keys (K_H^P /K_H^S). This HE key pair is partially opened to the Blockchain network (i.e., the public key K_H^P is written into the smart contract and broadcasted to all nodes). Note that we will never reveal the HE private key K_H^S.
3. Model provider encrypts the initial model parameters M_0 using HE public key K_H^P. It then stores the encrypted model $f_H(M_0)$ on the distributed file system, and uploads its hash to the smart contract.
4. Model provider locally generates a pair of asymmetric encryption (e.g., RSA) keys (K_A^P /K_A^S). The asymmetric encryption key pair is also partially opened to the network.

Function	Gas
authority()	728
cancel()	877
dataConsumer()	816
dataProvider()	706
executeForcefulRefund()	commonly 21000
executeForcefulSettle()	commonly 21000
expirationTime()	724
initTransaction(address,bytes32,bytes32)	81112
ipfsConsumerPubKeyAddress()	500
ipfsDataAddress()	544
ipfsEncryptedAesKeyAddress()	632
ipfsSchemaAddress()	654
provideData(bytes32,bytes32)	61061
requestForcefulRefund()	20835
requestForcefulSettle()	21100
requestedForcefulRefund()	618
requestedForcefulSettle()	690
verifiedData()	commonly 21000

TABLE I: Gas cost for smart contract functions defined in dataTransfer.sol

4.4 Federated learning

The following sub-steps are executed in sequence. The training process will be repeated many times until the global model is converged.

1. The model provider finds a suitable data resource for training. It then asks for approval from the corresponding data provider. Note that the cost of data is specified in the corresponding schema.
2. If the data provider accepts the request, it will search for the most suitable computation provider on the network. Similarly, data provider will ask for approval from the computation provider. Note that for a rational data provider, it will only select computation provider with a cost less than the token received from the model provider.
3. If the computation provider also accepts the request, the training process starts.
   (a) Data provider downloads the HE public key K_H^P and locally generates a symmetric encryption (e.g., AES) keyK_{B_1}. Training data (D_i) is encrypted using HE public key and then further encrypted using the symmetric encryption key. Encrypted data $f_{B_1}(f_H(D_i))$ will be sent to the DFS. The generated symmetric encryption key will be sent to the distributed file system too, but encrypted with an asymmetric encryption key $K_{A_2}^P$ which we will explain later.
   (b) Computation provider downloads both the encrypted model $f_H(D_0)$ and encrypted data $f_{B_1}(f_H(D_i))$ The encrypted data can be decrypted to $f_H(D_i)$ as computation providers can obtain the symmetric encryption key $K_{A_2}^S$by decrypting the $f_{A_2}(f_H(K_{B_1}))$ with the self-generated asymmetric encryption key $K_{A_2}^S$. At this point, computation providers perform training and the results ($f_H(D_i)$) are encrypted using symmetric encryption. The symmetric encryption key K B 2 will be encrypted using public key $K_{A_1}^P$ The encrypted key

$f_H(K_{B_2})$, along with the encrypted model $f_{B_2}(f_H(D_i))$, will be saved in the distributed file system and uploaded to the smart contract.

(c) Model provider downloads all results $f_{B_2}(f_H(D_i))$ which can be decrypted to M_i. Then the results are aggregated into a global model for next round of federated learning.

4.5 Award distribution

When the global model finishes learning, model providers calculate the contribution of each data provider, and distribute the token in award pool to these providers. In addition, model provider might also reveal the asymmetric encryption key$K_{A_1}^S$. By revealing $K_{A_1}^S$, all the HE-encrypted results will be visible to all participating parties, and the correctness of distribution of contribution can be verified off the chain.

5. Conclusion and Future Work

As we've seen, what we proposed as Blockchain-based platform Distributed AI is a collection of concepts and components that form the basis of a new AI ecosystem which aims to train a model with distributed data and to reserve the data ownership for their owners and the interest of trained model. To more practical applying, our future work has key challenges:

- **Performance**: We need to make sure its performance to meet modelling requirement, especially considering the heavy modelling tasks with HE and MPC. However, the scalability of Blockchain (transactions per second) has become the key issue [12, 7] when the network is used in a real-world application. With various new consensus mechanisms proposed, such as DAG in IOTA and HashGraph [11], private/consortium Blockchain provides an alternative solution.
- **Decentralization and security**: In the current design of DAI, we still rely on some decentralized mechanism to maintain the platform. And can we make removal of 3rd-parties like notification?

6. REFERENCES

[1] Kone˘cný, Jakub, et al. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).

[2] McMahan, Brendan, and Daniel Ramage. "Federated learning: Collaborative machine learning without centralized training data." Google Research Blog (2017).

[3] Wood, Gavin. "Ethereum: A Secure Decentralised Generalised Transaction Ledger Final Draft." (2015).

[4] Du, Wenliang, and Mikhail J. Atallah. "Secure multi-party computation problems and their applications: a review and open problems." Proceedings of the 2001 workshop on New security paradigms. ACM, 2001.

[5] Papernot, N. (2018). A Marauder's Map of Security and Privacy in Machine Learning, (October 2018), 1–20. https://doi.org/10.1145/3270101.3270102

[6] Gentry, Craig, and Dan Boneh. A fully homomorphic encryption scheme. Vol. 20. No. 09. Stanford: Stanford University, 2009.

[7] Zheng, Zibin, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. "Blockchain challenges and opportunities: Asurvey." Work Pap.–2016 (2016).

[8] Shen, Yuncheng, Bing Guo, Yan Shen, Xuliang Duan, Xiangqian Dong, and Hong Zhang. "A pricing model for big personal data." Tsinghua Science and Technology 21, no. 5 (2016): 482-490.

[9] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521, no. 7553 (2015): 436.

[10] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[11] Baird, Leemon. Hashgraph consensus: fair, fast, byzantine fault tolerance. Swirlds Tech Report, 2016.

[12] Croman, Kyle, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller et al. "On scaling decentralized blockchains." In International Conference on Financial Cryptography and Data Security, pp. 106-125. Springer, Berlin, Heidelberg, 2016.

[13] M. Swan, Blockchain: Blueprint for a new economy. " O'Reilly Media, Inc.", 2015.

[14] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[15] J. Benet, "Ipfs-content addressed, versioned, p2p file system," arXiv preprint arXiv:1407.3561, 2014.

7. A PPENDIX: A SMART CONTRACTS

In this section we provide the function signatures and main variables of our smart contract.

```
pragma solidity ^0.4.25;
contract DataTransfer {
  address public authority = < Address 0x... > ;
  uint duration = < Duration here > ;
  address public dataConsumer = msg.sender;
  address public dataProvider;
  bytes32 public ipfsSchemaAddress;
  bytes32 public ipfsConsumerPubKeyAddress;
  bytes32 public ipfsEncryptedAesKeyAddress;
  bytes32 public ipfsDataAddress;
  uint public expirationTime;
  bool public requestedForcefulRefund = false;
  bool public requestedForcefulSettle = false;

  modifier onlyBy(address _account) {}
  modifier onlyAfter(uint _time) {}
  modifier onlyBefore(uint _time) {}
  modifier etherProvided() {}
  modifier dataProvidedByProvider(bool
    ↪ _isProvided) {}

  function initTransaction(address
    ↪ _dataProvider, bytes32
    ↪ _ipfsSchemaAddress, bytes32
    ↪ _ipfsConsumerPubKeyAddress) public
    ↪ payable onlyBy(dataConsumer)
    ↪ etherProvided {}

  function provideData(bytes32
    ↪ _ipfsEncryptedAesKeyAddress, bytes32
    ↪ _ipfsDataAddress) public
    ↪ onlyBy(dataProvider)
    ↪ onlyBefore(expirationTime) {}

  function cancel() public onlyBy(dataConsumer)
    ↪ dataProvidedByProvider(false)
    ↪ onlyAfter(expirationTime) {}

  function verifiedData() public
    ↪ onlyBy(dataConsumer)
    ↪ dataProvidedByProvider(true) {}

  function requestForcefulRefund() public
    ↪ onlyBy(dataConsumer) {}

  function requestForcefulSettle() public
    ↪ onlyBy(dataProvider)
    ↪ dataProvidedByProvider(true) {}

  function executeForcefulRefund() public
    ↪ onlyBy(authority) {}

  function executeForcefulSettle() public
    ↪ onlyBy(authority) {}
}
```

Listing 1: Smart contract defined in dataTransfer.sol. Variable 'authority' is the Ethereum address of the Authority. Variable 'duration' is the time duration permitted to perform the trade and call the corresponding functions. Both 'authority' and 'duration' need to be pre-defiend before compilation