A Privacy-Preserving Data Feed Scheme for Smart Contracts*

Hao WANG†††, Zhe LIU†, Chunpeng GE‡, Nonmembers, Kouichi SAKURAI††, Senior Member, and Chunhua SU†††‡‡, Member

SUMMARY  Smart contracts are becoming more and more popular in financial scenarios like medical insurance. Rather than traditional schemes, using smart contracts as a medium is a better choice for both participants, as it is fairer, more reliable, more efficient, and enables real-time payment. However, medical insurance contracts need to input the patient’s condition information as the judgment logic to trigger subsequent execution. Since the blockchain is a closed network, it lacks a secure network environment for data interaction with the outside world. The data feed aims to provide the service of the on-chain and off-chain data interaction. Existing researches on the data feed has solved the security problems on it effectively, such as Town Crier, TLS-N and they have also taken into account the privacy-preserving problems. However, these schemes cannot actually protect privacy because when the ciphertext data is executed by the contract, privacy information can still be inferred by analyzing the transaction results, since states of the contract are publicly visible. In this paper, based on zero-knowledge proof and Hawk technology, a on-and-off-chain complete smart contract data feed privacy-preserving scheme is proposed. In order to present our scheme more intuitively, we combined the medical insurance compensation case to implement it, which is called MIPDF. In our MIPDF, the patient and the insurance company are parties involved in the contract, and the hospital is the data provider of data feed. The patient’s medical data is sent to the smart contract under the umbrella of the zero-knowledge proof signature scheme. The smart contract verifies the proof and calculates the insurance premium based on the judgment logic. Meanwhile, we use Hawk technology to ensure the privacy of on-chain contract execution, so that no information will be disclosed due to the result of contract execution. We give a general description of our scheme within the Universal Composability (UC) framework. We experiment and evaluate MIPDF on Ethereum for in-depth analysis. The results show that our scheme can securely and efficiently support the functions of medical insurance and achieve complete privacy-preserving.

key words: data feed, smart contract, privacy-preserving, medical Insurance

1. Introduction

Due to its decentralization, trustlessness, security and fairness, traceability and verifiable characteristics, smart contracts [1]–[4] are becoming more and more popular in financial scenarios like medical insurance [5]. The traditional medical insurance system is not reasonable in some aspects, like arrear of claim or insurance fraud, which have led to many medical disputes in recent years. Smart contracts on blockchains can fix these problems effectively. Once the condition is triggered, smart contracts can spontaneously execute the compensation transaction and transfer the compensation to patients as soon as possible. Meanwhile, based on the open and transparent properties of blockchains, interaction between the participants and smart contracts can be clearly viewed and verified, which can also well prevent the occurrence of collusion to defraud insurance.

Unfortunately, there are still many problems with using smart contracts as a transaction medium. The medical insurance contracts need to input the patient’s condition information externally as the judgment logic to trigger the subsequent execution. Since the blockchain is a closed network environment [6], it is difficult for smart contracts to accurately and effectively access the data of the external world. The data feed [7]–[10] is designed to meet the needs of smart contracts on the blockchain to access external world data information. The data feed can be simply regarded as a way to transfer structured data from one system to another, but it is not just ordinary data transmission, it is a purposeful, structured, faster and more accurate data acquisition and transmission process. At present, smart contracts lack a substantial ecosystem of reliable and secure data feeds, and the data transmitted to the chain by data feed cannot guarantee authenticity and integrity. Meanwhile, at the process of data feed, privacy may be leaked. For medical insurance, after a patient sees a doctor, the hospital, as the data feed provider, uploads the patient’s medical condition, personal information and other private data to the smart contract. Patients may not be willing to send their private data to the blockchain due to privacy concerns. This seriously hinders the application and promotion of smart contracts in the medical insurance industry.

There are many related researches on smart contract data feed like Town Crier, TLS-N and PDFS [8]–[10]. They use different principles and technologies to solve data feed
security problems. Town Crier is based on Intel’s SGX hardware protection technology, which provides a secure and trusted execution environment to protect data security and privacy. TLS-N is an extension of the TLS protocol to generate verifiable non-interactive proof of TLS session content, which solves the problem of non-repudiation of data feed and trusted third parties. PDFS proposes a more practical authenticated data feed (ADF) scheme that takes into account the dishonesty of content providers and imposes restrictions. They all focus on the authenticity of data feeds and some of them also consider privacy and propose excellent solutions. However, we find that it is not enough to preserve privacy of data feeds only because although data sent to the contract is encrypted, adversaries can still analyze the execution consequence of contracts to infer the patient’s condition, which will be described in Sect. 3 in detail. Therefore, in this paper, we propose a smart contract data feed scheme that implements complete privacy-preserving on-and-off-chain. To show our scheme more clearly, we implement our scheme on medical insurance scenario, which is called MIPDF. In general, we implement smart contracts for medical insurance claims, and propose a on-and-off-chain model composed of patients, insurance companies, and a hospital, which plays a role as a data feed source. The idea behind our scheme is to use smart contracts and the blockchain to ensure the fairness and reliability of claim transactions, use zero-knowledge proof to realize the privacy-preserving of data feed and use Hawk to guarantee the privacy-preserving of on-chain data. Patients and insurance agencies negotiate the content of the insurance contract in advance and establish a smart contract to automatically generate a privacy-preserving contract through Hawk. After the patient sees a doctor, the hospital signs for the relevant data and generates a zero-knowledge certificate, which is sent to the contract as the data feed. The contract automatically parses the data and takes it as the subsequent judgment logic to determine subsequent transactions such as a transfer. Since all transactions happen on the blockchain, patients and insurance agencies cannot lie. Meanwhile, since insurance agencies have already deposited the indemnity into contracts in advance, the problem of arrear of indemnity can also be solved. Therefore, the rational choice for all parties is to honestly comply with the terms of the contracts. Patients do not have to worry about the leakage of their private data with the utilization of our scheme.

**Contributions.** Overall, contributions of our work are shown as follows:

1. For the first time, we consider the incompleteness privacy-preserving of data feed. We propose MIPDF, a decentralized and on-and-off-chain privacy-preserving smart contract scheme based on the medical insurance case. Our scheme is based on zero-knowledge signature algorithm and Hawk technology to preserve data feed and on-chain privacy respectively.

2. We formulate a security definition for MIPDF with an ideal functionality under the Universal Composability framework. We build a security model of the solution, and combining with our ideal program, we describe the security goals that our solution can achieve. We specifically describe how our scheme implements privacy-preserving in the on-chain part and the off-chain part respectively in detail.

3. We implement our scheme based on Ethereum and make comprehensive experiments to evaluate the performance. The results show that our scheme takes only $44$ to run the whole process. The time cost is also within a reasonable range. We consider some technical challenges like gas and bilinear mapping implementation and give our solutions to them.

2. **System Architecture and Model**

Medical insurance payment scenarios are very common in real life. Users sign a contract with an insurance agency. Once a certain medical condition occurs, a certain amount of money will be paid to users. The amount of compensation that may be paid for different degrees of illness is also different. However, during the execution of the contract, there may be some unfair incidents. For example, the patient is indeed ill while the insurance company is unwilling to pay, or the two parties have different perceptions of the condition and cannot reach a consensus. It is also possible the patient has some dishonest behaviours such as falsifying medical conditions and cheating insurance. Therefore, a trusted agreement or database record needs to be adopted to make the consensus reached by multiple parties irresistible. Therefore, we propose a medical insurance smart contract scheme called MIPDF and in this section, we mainly make a specific description of the system architecture of our scheme. Under the umbrella of our scheme, medical record information of patients will not leak from beginning to end.

2.1 On-and-Off-Chain Privacy-Preserving

Many of the current data feed security solutions also consider privacy protection issues. For example, in Town Crier, using the key $pk_{TC}$ bound to the SGX instance to encrypt data can ensure that no one other than the user can decrypt it, thus achieving privacy-preserving. However, in our medical insurance case, it is not enough to only ensure the privacy of the data feed. Adversaries can still analyze the results of the contract to infer data information. Upon receiving the trigger transaction, the contract chooses to send money to the patient or the insurance agency according to the decryption result of privacy-preserving level’ and res’ which represent the disease condition and the corresponding diagnosis respectively. At a first glance, the patient’s private data is protected, however, once the transfer transaction has been sent, the adversary can judge how the contract is executed based on the execution of the transaction and the transfer amount, thereby inferring the patient’s information.
Therefore, it is not enough to protect either data feed or on-chain privacy only. The two must be combined to achieve a full range of privacy protection on and off the chain to meet the demand.

2.2 System Architecture

MIPDF is based on the medical insurance case so that it involves three participants: a patient, an insurance agency and the trusted third-party, hospital and three kinds of entries: a blockchain, a data feed server and a Hawk server. In other words, from the application layer, our system architecture is composed of a patient, an insurance agency and the hospital while from the network layer, the architecture is composed of a blockchain, a data feed server and a Hawk server consists of three parties. Our system architecture is shown in Fig. 1. Our model includes three main participants:

- **Patient.** In our model, the patient is the most important part because he/she needs to negotiate the contract content with the insurance agency to ensure his/her profits and disease types and needs to see a doctor to provide personal information for the hospital. The patient needs to prepare an cryptocurrency address which supports the smart contract for sending and receiving transactions. The main insurance contract is uploaded by the patient in our model while we assume that it is guaranteed that the contract is definitely correctly set up. The contract is sent to the Hawk server and converted into a Hawk contract.

- **Insurance Agency.** The insurance agency is a participant of the Hawk contract that can receive the refund of the contract deposit. It also plays a role in jointly negotiating the construction of medical insurance contracts.

- **Hospital.** The hospital is the main algorithm executor in our model. It is the data source of our data feed demand. After the patient sees a doctor, the hospital processes the data to determine whether the patient meets the medical insurance payment conditions, signs for the data and outputs the judgment result. Then, it executes our privacy-preserving algorithm, so that the patient’s treatment result is zero-knowledge. Finally, the hospital transmits the processed data as a data feed to the chain. Generally speaking, the hospital in our model is the trusted third-party and we assume that it is definitely honest and trustful.

It involves three types of entries:

- **A blockchain.** A blockchain is used to implement smart contracts. It can be a public chain such as Ethereum, a private chain or a consortium chain, as long as it supports smart contracts. Meanwhile, The blockchain needs to be compatible with data feed and Hawk technology both, which means the blockchain must support the technical requirements of the current data feed services and also support the adaptation conditions of Hawk in

- **A data feed server.** A data feed server is used to provide data feed service obviously. In detail, the providers are devoted to supporting network services, finding the specified data and forwarding it to blockchains. Nowadays, there are already many data feed service providers for blockchains like Oracle. Without extra data feed protecting schemes, the security of data feed depends totally on the reputation of the providers themselves. In this paper, we also assume the data feed provider is reliable and honest for simplicity.

- **A Hawk server.** Because our model needs to convert a prime contract into a Hawk contract, a Hawk server is set to implement this function. For further details about specific the Hawk server, please see

2.3 Security Model

In our medical insurance case, we assume the third-party hospital is honest because hospital is a social welfare organization that provides services in accordance with the law. Hence, we do not concern the dishonest or false behaviours on it, which means the hospital will not help insurance fraud, nor will misdiagnosis or medical accidents occur. Then, we assume the patient and the insurance agency are both untrusted, which means their main purpose is to obtain more benefits but may even be possible to do something malicious. Meanwhile, we assume that the blockchain is secure enough that the adversary cannot break it on his/her own. Finally, we assume that the techniques, algorithms and cryptographic primitives used in our scheme are secure against the adversary.

From the specification of the ideal program Ideal-MIPDF, it captures the following security goals:

- **Security.** Ensure that after the contract is uploaded, all the items are locked and cannot be tampered with, and the feedback data from the hospital is authentic, integrable and credible. After the user initiates the data feed request, all the processes are automatically executed by the smart contract, out of reach of the user, the insurance company, and the adversary. Based on
the assumption that the hospital is a trusted third party and the data feed server is reliable, the entire system is secure.

- **Privacy.** The sensitive information, including the disease information, the diagnosis result and the insured amount, can never be disclosed to the adversary. Information mentioned above is sent to the contract directly while the adversary can only sense that someone has sent a request transaction. The whole calculation process is executed by the contract itself without leaking any information. Meanwhile, the contract executing results can also be protected and the adversary cannot find any valuable information.

2.4 Ideal Program

Therefore, with regard to these security requirements of our scheme, we propose the ideal program Ideal-MIPDF in Fig. 7 [15]. In our Ideal-MIPDF, Init initializes the program and Create makes the patient negotiate the contract items with the insurance agency and implements it with the smart contract codes. The insurance agency can previously make an estimate of the body situation of the patients and make a specific contract for them. Note that the contract is sent to the Hawk server and transformed into a Hawk contract, which means the sensitive information like the specific money amount or the disease items and data flow can be run in the private space and be protected. Meanwhile, the patient uploads his contract and deposits the money he wants to be insured. Then, the insurance agency sends the Deposit transaction to the contract to deposit the corresponding amount of funds. When some diseases have happened, the patient can call the SendRequest function to send a data feed request including some medical information like identity, time and other special details to the data feed server DF as a relay to connect to the hospital. After receiving the feedback transaction from the data feed server DF including the medical judgment from the hospital with the signature under the zero-knowledge proof, the contract verifies the proof and the signature and if true, the contract executes the claim process automatically and sends the transfer transactions to both participants.

3. MIPDF: Medical Insurance Privacy-Preserving Data Feed Scheme for Smart Contracts

In this section, we describe our scheme about privacy-preserving data feed for smart contracts on the medical insurance case. Actually, our scheme is mainly divided into two contributions. The first is to propose a smart contract scheme for the medical insurance case in order to address some dishonest and unfair behaviours. During this process, we find and consider the data feed problems on it and propose a privacy-preserving data feed scheme to strengthen our smart contract scheme. We firstly present our main idea and principles of this work. Then, we show the data encoding of our implementation. Afterwards, we specifically describe some structural details of our scheme.

3.1 Main Idea

The main idea of MIPDF is shown as follows. Medical insurance premiums are usually mainly determined by the user’s insured amount and the coefficient corresponding to the disease. The corresponding compensation coefficient of the disease is mainly evaluated by the severity of the disease and the amount of estimated treatment cost. These parameters are pre-arranged by the insured and the insurance agency and written into the smart contract to achieve non-tampering and non-repudiation. When a disease occurs, the user triggers the data feed request mechanism, and applies for medical record data to the hospital in which the patient visits to trigger the compensation mechanism. In this process, due to the numerous security problems of the data feed, the simple data feed data is not credible enough, and it will also reveal the user’s important medical privacy. A better way is for the hospital to determine the patient’s condition, sign the result, and transmit the zero-knowledge proof of the signed result through the data feed, rather than the plaintext data. Since the data transmitted to the contract is zero-knowledge, no patient information will be disclosed. However, this is not enough. In the previous section, we mentioned that the adversary will indirectly analyze the patient’s condition information through the execution of the contract. Therefore, we still have to protect the privacy of the execution on the chain. We use Hawk to ensure the privacy of key data and transactions. Through the combination of these two technologies, we ensure complete privacy protection throughout the execution process.

3.2 Data Encoding

Since the user information, medical conditions, and the judgment result of diagnosis in medical insurance are all text information, that is, strings, we will encode them uniformly to facilitate the implementation of our scheme. First of all, we ranked and digitized the user’s medical condition information according to its severity. From mild colds to life-threatening cancers, we present the results of coding in Table 1. We list several types of classical diseases and their corresponding compensation. Actually, the real medical insurance contract may contain thousands of items and more complicated protocols. For simplicity, we transform the six strings into the format of number 1 to 6. Information in this table is pre-written in the contract code. Then, since the hos-

| Table 1 Data encoding of symptoms |
|-----------------------------------|
| disease type | level | compensation amount |
| --------------|------|---------------------|
| Flu            | 1    | 30 dollars          |
| Pneumonia      | 2    | 150 dollars         |
| Fracture       | 3    | 300 dollars         |
| Diabetes       | 4    | 10000 dollars per year |
| Uremia         | 5    | 30000 dollars per year |
| Cancer         | 6    | 50000 dollars per year |
pital evaluates the patient’s symptom and returns the diagnosis result, we also convert the diagnosis result “Yes” or “No” into 1 or 0 in the digital form. Based on this kind of data encoding, the hospital converts the character string into a digital form to facilitate signature and zero-knowledge proof processing. Hence, the medical data $d$ can be encoded into a data string in the following format: $d = \langle \text{level}, \text{res}, T \rangle$ where level is the disease level defined in Table 1 and res is the hospital’s diagnosis result. Both level and res are integers. $T$ is the timestamp of this result. Once the hospital receives the data feed request, it collects these data, makes the zero-knowledge proofs of these commitments and sends them to the data feed server.

### 3.3 MIPDF Scheme

In this section, we will elaborate on our MIPDF scheme against the data feed privacy leakage problem. The MIPDF scheme is mainly divided into two key modules:

**Off-chain Part.** First of all, for the data information that will be transmitted on the chain, the most basic thing is to have the signature of the data to ensure the reliability of the source of the message and the integrity and authenticity of the data, hence the steps for signing and verifying the data are indispensable. However, this is still not enough. Because of the openness of transactions and data on smart contracts, for users and their data with privacy-preserving demands, it is necessary to implement privacy-preserving operations on the basis of signatures. This paper is based on processing the data feed private data through zero-knowledge proof cryptography to ensure that it is transmitted to the smart contract in a ciphertext state and verifiable correctness. However, it is not enough to solve the privacy protection of data when the data source is transmitted. After the data is on the chain, since the data during the execution of the smart contract is publicly visible, the execution result of the contract can be observed to determine the execution of the data situation, so there is still a need to protect the chain.

**On-chain Part.** This part is based on the on-chain data privacy-preserving of the Hawk system mentioned above, that is, the user contract is transformed into the Hawk contract in this paper, and the private data is passed to the private part to protect it. First of all, the contract party sure want to get the data, data sources and data forms of the basic requirements, such as and then set the corresponding smart contract uploaded to Hawk module, Hawk compiler will automatically compiled to block the chain between the user and encryption protocol, and sends a datagram request to contract content, users can verify the authenticity of the data contract, if it is true, then send the data to trigger subsequent Hawk contracts contract execution, The transactions involving relevant private data are all executed in the private part of the Hawk system, so as to ensure that the privacy of data is not leaked. After the execution of the contract, the outside world can only see the simple contents of some public parts, and the key transaction data cannot be seen.

Then, we describe the zero-knowledge proof algorithm of our scheme in Fig. 2. This algorithm comes from [16] and is based on the bilinear pairing. For simplicity, we call this algorithm as BPZK.

**BPZK: Zero-knowledge Proof Signature Algorithm**

**Setup:** On input the security parameter $k^*$, output two prime order groups $G = \langle g \rangle$ and $G = \langle g \rangle$, where the prime order $q = \theta(2^k)$ and there is a non-degenerate and computable linear pairing $e$, such that $e : G \times G = G$. In other words, $G$ and $G$ have the same prime order $q$ and $g$ is the generator of $G$ and $g$ is the generator of $G$. The Setup algorithm can be abbreviated as $(g, G, g, g, e) \leftarrow \text{Setup}(k^*)$.

**Key generation:** Upon the setup algorithm, input the $(q, G, g, g, e)$. The Key generation algorithm chooses $x \leftarrow \mathbb{Z}_q$, $y \leftarrow \mathbb{Z}_q$ and sets $sk = (x, y)$, $pk = (q, G, g, g, e, X, Y)$, where $X = g^x$, $Y = g^y$.

**Signature:**

1. Public input: the public key $pk = (q, G, g, g, e, X, Y)$ and the commitment $M$.
2. User input: the message $m$, which meets $M = g^m$.
3. Signer input: the signature secret key $sk = (x, y)$.
4. The user gives the zero-knowledge proof of the commitment $PK(\mu) : PK(\mu) = g^\mu$. Then, the signer calculates $\sigma = (a, a^g, a^{xmx+y}) = (a, b, c)$, where $a \leftarrow \mathbb{Z}_q$, $a \leftarrow g^x$. Afterwards, the user outputs the signature $\sigma$.

**Verification:**

1. Public input: the public key $pk = (q, G, g, g, e, X, Y)$.
2. The verifier input: the message $m$ and the signature $\sigma = (a, b, c)$.
3. Then the verifier calculates the blind signature $\sigma'$. Choose the random number $r$ and $r \in \mathbb{Z}_q^*$ and blind the signature $\sigma = (a^r, b^r, c^r) = (\bar{a}, \bar{b}, \bar{c})$. Let $V_a, V_b, V_s$ make $V_a = e(X, \bar{a})$, $V_b = e(X, \bar{b})$, $V_s = e(g, \bar{c})$. The prover and verifier will calculate these values (locally) and then execute the following zero-knowledge proof protocol: $PK(\mu, \rho)$ : $V_a^\rho = V_a V_s^\rho$. The verifier agrees with the above proof if and only if the proof protocol is true and $e(\bar{a}, Y) = e(g, Y)$. Fig. 2. BPZK: Zero-knowledge proof signature/verification algorithm based on bilinear pairing.

The MIPDF scheme involves smart contracts deployed on the blockchain, the data feed server and the Hawk server. The procedure of our scheme is presented in Fig. 3. Here are the details of each step in MIPDF:

- **Init:** In the initialization step, the hospital runs the Setup algorithm in BPZK and generates the bilinear pairing public parameter group $(q, G, g, g, e)$ as the master key.
- **Deposit:** The patient and the insurance agency deposit the certain amount of money into the smart contract as the prepayment of the medical insurance.
- **SendRequest:** When the patient has seen a doctor, he/she can send the data feed request transaction...
Protocol-MIPDF

The Patient \( \mathcal{P} \):

Setup: Upon receiving ("setup"):
\( \mathcal{P} \) sends it to the Hawk server and generates a Hawk contract.

Create: Upon receiving ("create", \( \text{Sdepot}, \text{T}_{\text{end}} \)), \( \mathcal{P} \) sends ("create", \( \text{Sdepot}, \text{T}_{\text{end}} \)) to Contract-MIPDF.

SendRequest: Upon receiving ("sendRequest"):
\( \mathcal{P} \) sends ("sendRequest") to Contract-MIPDF.

Abort: Upon receiving ("Abort"), \( \mathcal{P} \) sends ("abort") to Contract-MIPDF.

The Hospital \( \mathcal{H} \):

Setup: Upon receiving ("setup"):
\( \mathcal{H} \) runs the \( \text{BPZK.Setup} \) and generates the public parameters \((g, G, g, g, e)\).

Key Generation: Upon receiving ("keyGeneration"):
\( \mathcal{H} \) runs the \( \text{BPZK.Key Generation} \) and sets \( sk = (x, y), pk = (g, G, g, g, e, X, Y) \), where \( X = g^x, Y = g^y \).

Signature: Upon receiving ("Data Feed", \( \text{param}, \text{callback} \)) from the data feed server DP:
1. Parse \( \text{param} \), get the patient’s id and his/her medical id and search medical information in their local database;
2. Return diagnostic result \( \text{level and res} \) and summarize them into \( \text{data} \);
3. Run \( \text{BPZK.Signature} \) with the result \( \text{data} \) as input and outputs the signature \( \sigma \);
4. Send the signature \( \sigma \) ("Feedback", \( \sigma \)) to the Contract-MIPDF.

SendRequest to the medical smart contract. Then, the medical contract edits the information (\( \text{url}, \text{spec}, \text{T} \)) prepared for the data feed request and sends the request to the data feed contract. After receiving the request transaction, the data feed server parses the request demand and executes the data feed to the hospital.

Feedback: When receiving the data feed request, the hospital parses the request and gets the information of the patient and its specific situation, including the diseases, the treatment time and other information as a reference for comprehensive judgment. When the diagnostic result has been produced, the hospital

Claim: Upon receiving the feedback transaction from the data feed server, the medical contract parses the transaction and gets the zero-knowledge proofs

Figure 4 is a schematic diagram of the data and capital flow of this model. The blue line represents the flow of data and the red line represents the flow of money. Firstly, the flow of cash is introduced. After the user negotiates the corresponding contract regulations with the insurance company, the smart contract with privacy protection is realized by means of the Hawk system, and the corresponding advance payment is transferred into the contract, that is, the money of the user to buy the insurance and the money of the insurance company to advance the compensation is deposited in the contract first. After executing the whole process of data feed and compensation judgment, the contract will come to the result of whether to pay and how much to pay, and then the contract will execute the compensation function to send the money to the user or return the full payment or balance payment to the insurance company. Then we describe the data flow, which is essentially the execution process of data feed, consisting of four steps:

- Send a data request. The Hawk contract sends a datagram request with (param, callback) to the content contract, where the param contains the data source address URL as shown in the figure, the special form of the required data spec, and the contents of the time T. The special form of spec in this case includes the patient’s name, number, the number corresponding to the patient’s illness, etc. The spec part can be encrypted with the public key given by the hospital, which ensures the privacy of the data on the chain.

- Content contract After receiving the data request of M1, set an ID number for the request, and forward the request, including (ID, param), to the data source of the corresponding URL, that is, the hospital in this case.

- Hospital after receives the request data, using your private key to decrypt, get the corresponding patients and condition query request, given the corresponding query results, yes or no, and said with the data results, at the same time, in order to protect the privacy of the digital results hospital adopts zero knowledge proof for processing, the results stored in the data stream to content of the contract.

- After receiving the returned datagram, the content contract sends the data value to the corresponding user contract according to the ID. After receiving the data, the contract automatically verifies the correctness of the signature and the correctness of the zero-knowledge proof result, which is used to trigger the key judgment conditions for the final settlement. According to the conditions, the corresponding balance is sent to the user or the insurance company to complete the whole process. The Hawk contract divides the original contract into two parts. The parts of the contract that can be exposed are executed in the public part, such as the URL shown in the figure, the timestamp, and some public parameters. The contract part involving privacy content is implemented in the privacy part, in-
including the user’s identity information, disease level, data feed results, compensation amount and transaction involved, etc. Because the final settlement result is carried out in the privacy part of the Hawk system, the external personnel cannot judge the execution of the contract by observing the transaction result, and thus cannot know the condition information of the patient, hence achieving the privacy-preserving.

4. Implementation and Evaluation

4.1 Experiment and Evaluation

Our implementation of the MIPDF scheme consists of three components: the smart contract on the blockchain, the Hawk server, and the data feed server. We choose Ethereum for simulation experiments. The smart contract is written in Solidity language. The operating environment experiment uses Remix as our IDE which is an online compiler with a simple interface and simple operators based on browsers and does not need to be installed. The solidity language version used in this experiment is: 0.5.1+commit.c8a2cb62.Emscripten.clang. The Ethereum test network used is Rinkeby at https://www.rinkeby.io, using the PoA (Proof-of-Authority) consensus mechanism.

All the experiments have run on a PC with a CPU of Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz and 16 GB RAM executing on the Window 10 Professional operation system.

The following is the specific experimental process. The PBC library [17] is an open-source free cryptographic library that performs mathematical operations based on a paired cryptographic system. We use this PBC library to implement the bilinear mapping module of this paper and conduct experiments.

We use MetaMask, which is a browser-embedded Ethereum wallet and management tool, can easily help users view and manage their own currency on Ethereum, and can switch between various major blockchains like Ethereum main network and various test networks including Rinkeby. Afterwards, log in to Rinkeby’s official website and enter the Faucet page to apply for free test coins. This paper creates two accounts, one plays the role of a user and the other plays the role of an insurance company. The address of the medical insurance compensation contract, user address, and insurance company address are shown in Table 2. These addresses can be viewed on HTTPS://www.rinkeby.io [18].

**Money & Gas Cost.** Figure 5 is the money & gas flow of our MIPDF. We prove that our scheme is secure and sustainable at this aspect. It can be seen from Table 2 that under the current upper limit of 10000000 gasLimit, each transaction involved in this paper is far from reaching the threshold. In our testnet Rinkeby, our gasPrice of 5 transactions is set as 1 Gwei and it can work. While in Ethereum mainnet, currently the best gasPrice is about 50 Gwei. Due to the exchange rate of $600 per Ether, $1 can buy about 33000 gas. The total gas cost of our MIPDF including 5 transactions is 1462480 gas. According to our conversion formula above, our total money cost is 1462480/33000 = $44, which is a quite cheaper expenditure than manual insurance service.

**Time Cost.** Meanwhile, we also consider the time-consuming of our MIPDF. Since our scheme is completely based on the modelling of the smart contract of Ethereum, the time-consuming costs do not involve the initial parameters setup, writing Solidity contracts, and other pre-processes. We only consider the time-consuming process of transactions on Ethereum and data feed. Therefore, in summary, the time-consuming solution of this paper is only the time-consuming transaction execution on Ethereum. We consider the time cost of our scheme in each step. Table 3 shows the time situation of every transaction of the experiment we conduct on the Rinkeby testnet. The total time consumption of 5 transactions is 57s. This paper does not include the manual time costs of designing smart contracts, issuing transactions, etc., because these time costs are different from person to person and also needed to be consumed in offline contract negotiation. It can be seen that the time cost is also very low. Figure 6 shows the time cost of BPZK.Setup & Key Generation process and

---

| Function | Block | Gas | Txn Fee | Txn Hash     |
|----------|-------|-----|--------|--------------|
| Init     | 7764314 | 1089157 | 0.001089157 | 0xf4a3625346bf8762a5c070cc311fd97953db731b20a247c79d06fd6f99789ad |
| Create   | 7764398 | 42491 | 0.000042491 | 0xe7e3508c11576e467905962b85975b605c8da0b41c937848f99dca1f3f93c46 |
| SendRequest | 7764401 | 31855 | 0.000031855 | 0x10274902d1d9d93dbfa678c49df4c5e0b6b673afcc2d097e7d58a40268fa81 |
| GetRequest | 7764421 | 274272 | 0.000274272 | 0xd498dab9b2d25df6a06fc04798e17476277e008df6835ef301e331db8210fc6 |
| Claim    | 7764426 | 24705 | 0.000024705 | 0x11fc1e56c6d07f00d2ab507603c28678069174a691921a062db1a485a |
Table 3  Time cost situation on Rinkeby

| Function  | Start Time | End time | Difference | Txn Hash |
|-----------|------------|----------|------------|----------|
| Init      | 01:21:34   | 01:21:45 | 11         | 0xf4a3625346b8762a5c070cc31f6d97953db7d31b20a247c79d06fd97989ad |
| Create    | 01:42:40   | 01:42:53 | 13         | 0xe7e3508c11576b467905962b85975b05ccda84d149378948f99dca13f3f9c46 |
| SendRequest | 01:43:27   | 01:43:38 | 11         | 0x10274902d1d9d93db4a66b73afcb7d58a0268fa81 |
| GetRequest | 01:48:26   | 01:48:40 | 14         | 0xd498dab9b2d25d6a06c0479b8c17476277e00bd7f935ef1301e331db8210fc6 |
| Claim     | 01:49:48   | 01:49:56 | 8          | 0x11fc1e56c0d7ff0dd2ab50703c32c86780691a74cda9619d3ea1062aba1a85a |

Table 4  Functionality comparison with related works

| Schemes | No third trusted party? | Required changes on Privacy-preserving? |
|---------|------------------------|------------------------------------------|
| TLSnotary [7] | ✓                      | TLS                                      |
| Town Crier [8] | ✕                      | ✗                                        |
| TLS-n [9]   | ✓                      | TLS                                      |
| PDFS [10]   | ✓                      | App                                      |
| Our Scheme  | ✓                      | ✗                                        |

Fig. 6  Time cost of BPZK.Setup & BPZK.Key Generation and BPZK.Signature. We conduct 100 to 500 repeated experiments and get the average time cost of BPZK.Setup & BPZK.Key Generations is about 80ms and the cost of BPZK.Signature is about 27ms.

BPZK.Signature. The average time cost of one time is about 80ms and 27ms respectively, which is negligible for the coarse-grained medical insurance model. Therefore, our MIPDF privacy-preserving scheme has no problem in terms of efficiency.

Functionality Comparison. In Table 4, we propose the functionality comparison of our scheme with some related works. It shows that our scheme does not need a third party or a carrier and can achieve privacy-preserving properties.

4.2 Security/Privacy Issues

Privacy leakage. Patients want to preserve their medical privacy when executing the MIPDF protocol. For the proof of the diagnose, patients need to send their personal identity information to the trusted hospital for checking. MIPDF chooses to signs the result and commits the signature as a zero-knowledge proof to preserve the privacy. Because of the signature and the zero-knowledge property, it can guarantee that the result is sent by the hospital and the data is privacy-preserved. However, due to the transparency of the smart contract, the transaction details and results of smart contracts may be easily viewed, which can be combined with terms and details of the contracts to infer the patients’ medical information. In our example, if the transaction shows that the contract transfers 30000 dollars to the patient, we can infer that the patient gets cancer or uremia. Therefore, for higher privacy-preserving, the Hawk scheme is utilized to achieve these on-chain transactions cannot be viewed except these two participants because they are running on the private space created by Hawk. Due to the assumption of the security and privacy of the BPZK algorithm and Hawk, we can believe that privacy can be completely preserved.

Insurance Fraud. Considering the current unfair insurance fraud phenomenon, our model can effectively address them. Since the patient and the insurance agency have negotiated the contents of the contract in advance, which are written into a smart contract and sent to the chain, and a certain amount of deposit has been deposited onto it, based on the immutability of the blockchain and the automatic execution property of smart contracts, the insurance company cannot deny or default, because the contract will automatically execute the compensation operation. Meanwhile, for user fraud, since all transactions and data are stored on the chain and cannot be modified and we assume that the hospital is trusted, the users’ malicious fraud will be rejected by rational miners. Even if the private key data is preserved in zero-knowledge proof, it can still be verifiable.

5. Conclusion

The blockchain-based trust mechanism enables smart contracts to be executed securely and credibly without a third
The Medical Insurance Contract $MI$:

**Init:** Set $state := INIT$ and initialize $balance := 0$, $deposit[P] := 0$, $deposit[I] := 0$. Set the claim coefficient $mapping(level := money)$ as the highest amount of the upper limit. Set the current time as $timestamp$.

**Create:** Upon receiving ("create", $depositP$, $T_{end}$) from a patient $P$,

1. Set $state := INIT$.
2. Set $deposit[P]$ has been initialized.
3. $deposit[P] := deposit[P] + depositP$.
4. Set $state := CREATED$.

**Deposit:** Upon receiving ("deposit", $depositI$, $T_{end}$) from an insurance agency $I$,

1. Set $state := CREATED$.
2. Set $deposit[I]$ has been initialized.
3. Set $deposit[I] := deposit[I] + depositI$.
4. Set $state := DEPOSITED$.

**SendRequest:** Upon receiving ("sendRequest") from $P$.

1. Set $state := DEPOSITED$.
2. Send transaction ("Data Feed", $param$, callback) to the data feed server $DF$, in which $param := (url, spec, T)$.
3. Set $state := SENDREQUESTED$.

**Claim:** Upon receiving ("claim", $level$, $res$) including the encrypted data feed values $level$ and $res$ from the data feed server $DF$,

1. Decrypt $level$ and $res$ with the public key $pk$ into level and res respectively.
2. Set $res := 0$, send $depositI$ to the Insurance Agency $IA$, else $res := 1$, send $depositI$ to the patient $P$ and the rest $depositI$ to the Insurance Agency $IA$.

**Abort:** Upon receiving ("abort", $I$) from $I$.

1. Set $current time := T - timestamp >= validity period$.
2. Transfer $balance$ to $I$.

**Data Feed Contract $DF$:**

**Init:** Set $state := INIT$.

**Create:** Upon receiving ("create", $T_{end}$) from the data feed service provider,

1. Set $state := INIT$.
2. Set $state := CREATED$.

**GetRequest:** Upon receiving ("Data Feed", $param$, callback) from the medical insurance smart contract $MI$,

1. Set $state := CREATED$.
2. Parse $param$ and get (url, spec, T).
3. Execute data feed on the hospital interactive website with url.
4. Set $state := GETREQUESTED$.

**SendData:**

1. Set $state := GETREQUESTED$.
2. Set url := url.
3. Send ("claims", data) to medical insurance contract $MI$.
4. The user receives the zero-knowledge proof of the commitment $PK(\mu) : M = g^\mu$. Then, the signer calculates $\sigma = (a, a^b, a^{mxy}) = (a, b, c)$, where $a \leftarrow Z_\ell, b \leftarrow g^b$.
5. Afterward, the user outputs the signature $\sigma$.

Fig. 7 The ideal program for MIPDF. Operations are executed sequentially under the control of state.
cient protocols.” International Conference on Security in Communication Networks, pp.268–289, Springer, 2002.

[17] B. Lynn, “The pairing-based cryptography (pbc) library,” 2010.

[18] “Rinkeby explorer,” http://www.rinkeby.io/, accessed Dec. 20, 2020.

Hao Wang received the B.S. degree in Information Security from Lanzhou University, Lanzhou, China, in 2018 and the M.S. degree in Cyberspace Security from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2021. His research interests include blockchain and smart contract.

Zhe Liu is a professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. He received the B.S. and M.S. degrees from Shandong University, China, in 2008 and 2011, respectively, and the Ph.D. degree from University of Luxembourg, Luxembourg, in 2015. His research interests include security, privacy and cryptography solutions for the Internet of Things. He has co-authored over 80 research peer-reviewed journal and conference papers. He was a recipient of the prestigious FNR Awards—Outstanding Ph.D. Thesis Award in 2016, ACM CHINA SIGSAC Rising Star Award in 2017 as well as DAMO Academy Young Fellow in 2019. He serves as program committee member in more than 60 international conferences, including program chairs in INSCRYPT 2019, CRYPTOIC 2019 and ACM CHINA SIGSAC 2020.

Chunpeng Ge received the Ph.D degree in Computer Science from Nanjing University of Aeronautics and Astronautics in 2016. He is currently a research fellow at University of Wollongong and an associate professor at Nanjing University of Aeronautics and Astronautics. His current research interests include cryptography, information security and privacy preserving for blockchain. His recent work has focused on the topics of public key encryption with keyword search, proxy re-encryption, identity-based encryption.

Kouichi Sakurai received the B.S. degree in mathematics from the Faculty of Science, Kyushu University in 1986. He received the M.S. degree in applied science in 1988, and the Doctorate in engineering in 1993 from the Faculty of Engineering, Kyushu University. He was engaged in research and development on cryptography at the Computer and Information Systems Laboratory at Mitsubishi Electric Corporation from 1988 to 1994. From 1994, he worked for the Dept. of Computer Science of Kyushu University in the capacity of associate professor, and became a full professor there in 2002. Now he is also working also with Adaptive Communications Research Laboratories, Advanced Telecommunications Research Institute International (ATR) as a visiting researcher with information security. Professor Sakurai has published more than 400 academic papers around cryptography and information security (See: http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/s/Sakurai:Kouichi.html)

Chunhua Su received the B.S. degree for Beijing Electronic and Science Institute in 2003 and received his M.S. and PhD of computer science from Faculty of Engineering, Kyushu University in 2006 and 2009, respectively. He is currently working as a Senior Associate Professor in Division of Computer Science, University of Aizu. He has worked as a postdoctoral fellow in Singapore Management University from 2009–2011 and a research scientist in Cryptography & Security Department of the Institute for Infocomm Research, Singapore from 2011–2013. From 2013–2016, he has worked as an Assistant professor in School of Information Science, Japan Advanced Institute of Science and Technology. From 2016–2017, he worked as Assistant Professor in Graduate School of Engineering, Osaka University. His research interests include cryptanalysis, cryptographic protocols, privacy-preserving technologies in machine learning and IoT security & privacy. He has published more than 100 papers in international journals and conferences.