Internet Web Trust System Based on Smart Contract

Shaozhuo Li, Na Wang(✉), Xuehui Du, and Aodi Liu

He'nan Province Key Laboratory of Information Security,
Zhengzhou 450001, Henan, China
twftina_w@126.com

Abstract. The current Internet web trust system is based on the traditional PKI system, to achieve the purpose of secure communication through the trusted third party. However, with the increase of network nodes, various problems appear in the centralization system of public key infrastructure (PKI). In recent years, in addition to cryptographic problems, attacks against PKI have focused on the single point of failure of certificate authority (CA). Although there are many reasons for a single point of failure, the purpose of the attack is to invalidate the CA. Thus a distributed authentication system is explored to provide a feasible solution to develop distributed PKI with the rise of the blockchain. Due to the automation and economic penalties of smart contracts, a PKI system is proposed based on smart contracts. The certificate chain was constructed in the blockchain, and a mechanism was adopted for auditing access to CA nodes in the blockchain. Experimental results show that security requirements of CA are met in this system.

Keywords: Public key infrastructure · Blockchain · Smart contract

1 Introduction

The current Internet web trust system is based on the traditional PKI system. Public Key Infrastructure (PKI) is a key management platform that follows established standards. It provides cryptographic services such as encryption and digital signatures and the key and certificate management systems necessary for all web applications. PKI technology is the core of information security technology and the key and basic technology of e-commerce.

This third-party-based trust mechanism is now facing serious security challenges, resulting in frequent security incidents. For example, DigiNotar was invaded in 2011 [1]. DigiNotar is a Dutch company whose main business is to issue certificates to the public and is the first CA to be completely invaded. The forged certificate caused a very serious man-in-the-middle attack in Iran, collecting a large number of Gmail passwords, and it's root certificate was revoked.In December 2013, TurkTrust CA issued a false certificate [2]. TURKTRUST Inc. incorrectly created two CA branches (*.ego.gov.tr and e-islem.kktcmerkezbankasi.org). The CA of the.EGO.GOV.TR branch was subsequently used to issue a false digital certificate to *.google.com. This deceptive certificate may be used to perform phishing attacks or man-in-the-middle attacks on

X. Cheng et al. (Eds.): ICPCSEE 2019, CCIS 1058, pp. 295–311, 2019.
https://doi.org/10.1007/978-981-15-0118-0_23

several Google Webs. With the increase of network nodes, the problem that single-point CA is not credible has surfaced, and the PKI system has a crisis. There are several key problems that need to be solved:

Certificates Can Be Issued Without the Authorization of the Domain Name Owner [3]
In principle, the biggest problem we have now is that any CA can issue certificates to any domain name without the authorization of the domain name owner. The key problem here is that there is no effective technical means to ensure that CA does not produce omissions and security errors. In those days when there were only a few CAs, this may not be a problem, but now there are hundreds of CAs, which has become a big problem.

Lack of Trust Flexibility
Another theoretical problem is the lack of trust flexibility. The relying party maintains a root certificate store containing a certain number of CA certificates, so that the CA is either completely trusted or completely untrustworthy, with no intermediate conditions. In theory, the relying party can remove the CA from the root certificate store. But in fact, this can only happen if there is a serious incompetence or security breach, or if the CA is very small. Once a CA issues a large number of certificates, it will be a big deal.

Invalidity of Revocation
There are two main reasons, the first is that it takes time to transfer revocation information between systems, and the second is the current soft failure strategy implemented by all browsers.

Therefore, we urgently need to construct a distributed decentralized identity management system. The emergence of blockchain gives us an opportunity.

Blockchain technology is a new distributed infrastructure and computing method, which uses blockchain data structure to verify and store data, uses distributed node consensus algorithm to generate and update data, uses cryptography to ensure data transmission and access security, and uses smart contract composed of automated script code to program and operate data. Blockchain has the following characteristics: distributed, open consensus, autonomy, information not tampering, anonymity, traceability, and programmability.

However, the current research on distributed PKI based on blockchain is just in its infancy, and there are many shortcomings, such as using only the blockchain as a data storage tool or ignoring the reality to adopt a fully distributed structure.

Based on this, this paper proposes a PKI system construction based on smart contract. Its main principle is to use the characteristics of blockchain to solve many problems of existing PKI. Firstly, its open consensus and information not tampering guarantee the openness and transparency of the whole authentication process and the unforgeability of certificates. At the same time, traceability also ensures the realization of the accountability system. The open consensus ensures that the certificate revocation can take effect in time. We have adopted smart contracts in this article, which has the characteristics of automated execution and programmability. We solve the problem that any CA can issue certificates to any domain without consent by setting the conditions required for the smart contract automation execution in the program. At the same time,

we use its automated execution to realize the automatic reward and punishment mechanism for erroneous CA, and solve the problem of trust flexibility. Moreover, the existing PKI system does not have an effective audit access mechanism for the CA node, and cannot guarantee the security and reliability of the CA in the blockchain.

2 Research Status

2.1 Improvement Scheme of PKI

At present, the main problem with PKI is that it does not have the consent of the domain name owner to issue a certificate. At present, there are mainly the following solutions to this problem:

Certificate Transparency CT:
Certificate transparency is an open framework for auditing and monitoring certificates. The CA submits each certificate they issued to the public certificate log and obtains a certificate of encryption submitted this time. Anyone can monitor every new certificate issued by the CA. For example, a domain owner can monitor each certificate that has their domain name issued. If this idea is implemented, then the fake certificate can be quickly discovered. The proof of submission can be delivered to the client in a different way so that it can be used to confirm that the certificate has been made public. Through this transparent and open way to supervise CA, the problem of issuing certificates without the domain name owner's consent is solved.

Public Key Pinning:
Public key pinning generates a fixed hash value for the certificate used by the website, and then passes the hash value to the browser when the user visits it. The browser will check whether the certificate hash value is the same as the previous hash value when the user visits it next time. If it is different, the connection will be disconnected. With pinning, network owners can choose one or more CAs they trust to create their own trusted ecosystems that are much smaller than the global ecosystem. Public key pinning can now be implemented through Chrome's proprietary mechanism, and HTTP public key pinning standards are being developed.

Although these two methods can solve the untrustworthy problem of CA to a certain extent, they all have certain defects. For example, CT lacks a feedback mechanism for error certificates. The public key pinning does not solve the existing PKI system problem, but adopts a way to narrow the credibility range to reduce the probability that CA is not credible.

2.2 Research Status of Blockchain

Blockchain is a decentralized, non-trusted distributed ledger technology, which is composed of distributed data storage, point-to-point transmission, consensus mechanism, encryption algorithm and other technologies. Blockchain is a decentralized and trustless Distributed Accounting technology, which is composed of distributed data storage, point-to-point transmission, consensus mechanism, encryption algorithm and

other technologies. Blockchain was first proposed in 2008, which is the underlying technology of Bitcoin proposed by Zhongbencong [4]. Since its emergence, blockchain has been applied more and more.

In 2014, Ethereum founders Vitalik Buterin, Gavin Wood and Jeffrey Wilcke began researching a new generation of blockchain, trying to achieve a smart contract platform that does not require a trust base in general [5]. In the narrow sense, Ethereum refers to a series of protocols that define the decentralized application platform. Its core is the Ethereum Virtual Machine (EVM), which can perform the encoding of any complex algorithm. Smart contracts rely on blockchain on EVM bytecode to run. The default execution environment of Ethereum is no process, nothing happens, and the status of each account remains the same. However, each user can trigger an action by sending a transaction from an external account to launch Ethereum. If the destination of the transaction is another foreign account, then the transaction may transfer some Ethereum, otherwise nothing will be done. But if the destination is a contract, the contract will be activated and the code will run automatically.

Contracts usually serve four purposes:

Saving as a database represents something useful to other contracts or the outside world;

As a foreign account with a more complex access protocol, it is called a "forward contract";

Manage an ongoing contract or relationship between multiple users;

Provide functionality to other contracts, essentially as a software library.

2.3 Distributed PKI Based on BlockChain

The characteristics of blockchain technology make it an ideal technology for a variety of applications. In particular, these characteristics demonstrate the applicability of blockchain to PKI. Since blockchain-based PKI solutions are distributed, they do not have centralized failure points. More importantly, blockchain technology has several open source implementations that help build cost-effective and efficient solutions.

According to different trust systems, the main distributed PKI research based on blockchain can be divided into two categories:

The first category is to improve the existing PKI trust system. Its main idea is to make use of the information of blockchain that can not be tampered with to modify and open consensus, so as to achieve transparent and open certificate storage and authentication process, such as:

Instant Karma PKI (IKP) [6]

The Instant Karma PKI (IKP) framework extends the traditional CA approach by recording CA behavior into blockchain. In this way, the network can detect a misbehaving or attacked CA and must react. The blockchain's event logging feature helps blockchain users track and monitor CAs and helps detect misbehaving CAs. This method can reduce trust problems in traditional CA-based algorithms, because abnormal CA can be detected eventually.

Pemcor [7]
Pemcor uses the blockchain database as a distributed secure data store [7]. The idea is to have the CA issue an unsigned certificate, the hash of the certificate is stored in the blockchain, and the blockchain is controlled by an authority such as a bank or government. These organizations share two blockchain databases, one for the generated certificate and the other for the revoked certificate. At the time of verification, these agencies check the blockchain data storage they maintain. If the hash value of the certificate exists in the generated certificate blockchain and is not in the revoked certificate blockchain, the certificate is valid; otherwise it is invalid. The idea is simple, with verification with low latency guarantees.

Certcoin [8]
Certcoin is a completely decentralized PKI that relies on Namecoin [9] to build its platform. The core idea is to record the user certificate through the public general ledger, and associate the user identity with the certificate public key in an open manner to realize the decentralized PKI construction. Any user can query the certificate issuance process to solve the problem of certificate transparency and CA single point of failure faced by traditional PKI systems. The registration, updating and revocation of certificates are realized by issuing users and their public keys in the form of blockchain transactions, and the normal operation of PKI is guaranteed by the unalterable attributes of blockchain. Merkle root only records the hash value of the transaction. Users can complete the certificate verification without downloading all blockchain transaction data.

At present, typical applications in this mechanism are too limited for the use of blockchain characteristics, and can only be used for storage and public authentication, ignoring other characteristics of the blockchain. And there is no complete PKI system construction, and the problem it solves is very limited.

The second type of mechanism adopts a fully distributed trust system. Its main idea is to use the distributed nature of the blockchain to better implement existing distributed trust systems, such as pgp networks. Its typical application is SCPKI.

SCPKI [10]
SCPKI is a fully distributed PKI system that uses a network trust model (PGP) and a smart contract on the Ethereum blockchain. Using the design of the network trust model, an entity in the system can verify (or guarantee) the identity of another entity as a replacement for the CA.

This type of mechanism is an ideally perfect distributed trust system, but it faces great challenges in its practical application. The PGP trust model largely reflects the relationship between people in real life. When used in a wide range, it is difficult to trust other nodes unconditionally and trust any node through trust transmission. The lack of trust management is a fatal defect of PGP. It is easy to cause security risks [11]. In the PGP trust system, each node is very important and is an indispensable part of the trust network. Once a certain point or even more points are manipulated or bought, the transfer chain will be interrupted, thus losing trust to many nodes. This causes irreversible damage to the trust network.

3 Construction of PKI System Based on Smart Contract

The whole system is divided into five parts: admission mechanism for CA nodes, certificate chain structure, conventional certificate management function, error certificate feedback mechanism and automatic economic reward and punishment mechanism. These five functions are realized through contracts, and make full use of the characteristics of automatic contract execution to improve the existing mechanisms. Before introducing these five functions, we first make a description of the nodes in the system.

Node settings:
We classify the nodes into CA nodes, end-user nodes, and ordinary nodes. The CA nodes participate in the blockchain transaction for certificate authentication. The end-user node has no similar permissions and can only be used as an entity for certificate application and certificate query. Ordinary nodes, they can only view certificates and participate in the maintenance of the blockchain. The joining of a CA node with a certificate issuing function requires an admission mechanism.

3.1 Admission Mechanism for CA Nodes

Moreover, the current distributed PKI system based on blockchain is purely public, and any node can join the blockchain, which is a big security risk for CA. The system in this paper introduces an admission mechanism for the joining of CA nodes, which distinguishes the joining of CA nodes from other nodes. Other nodes, they can only have certificates or view certificates, and cannot issue certificates. The joining of a CA node with a certificate issuing function requires an admission mechanism.

The addition of a CA node requires auditing to ensure the security of the CA in the system, similar to the consortium blockchains structure. At the same time, other nodes join without auditing, maintain the characteristics of the public blockchains, and ensure large-scale application on the Internet.

The CA node wants to join the blockchain. First, you need to obtain the certificate issued by the advanced CA offline. After obtaining the certificate, the node joins the blockchain to upload its own certificate to the contract, and the advanced CA that issues the certificate to it also uploads the same certificate to the contract. At this point, the contract will automatically detect whether the two certificate information is consistent, and detect whether the two nodes meet the information in the certificate. After the contract detection information is correct, the node can successfully become a CA node. After the initial CA chain is constructed, the root CA is normally in the offline state. When there is a demand, the root CA goes online again. Thereby ensuring the security of the root CA.

The certificate uploaded by both parties includes two parts. The first part is the data required by the X.509 format certificate, including Sequence number, signature algorithm identifier, signer name, validity period, principal name, principal public key. The second part is the verification node information. The required two-party address, Nodeaddress and Signeraddress (Fig. 1).

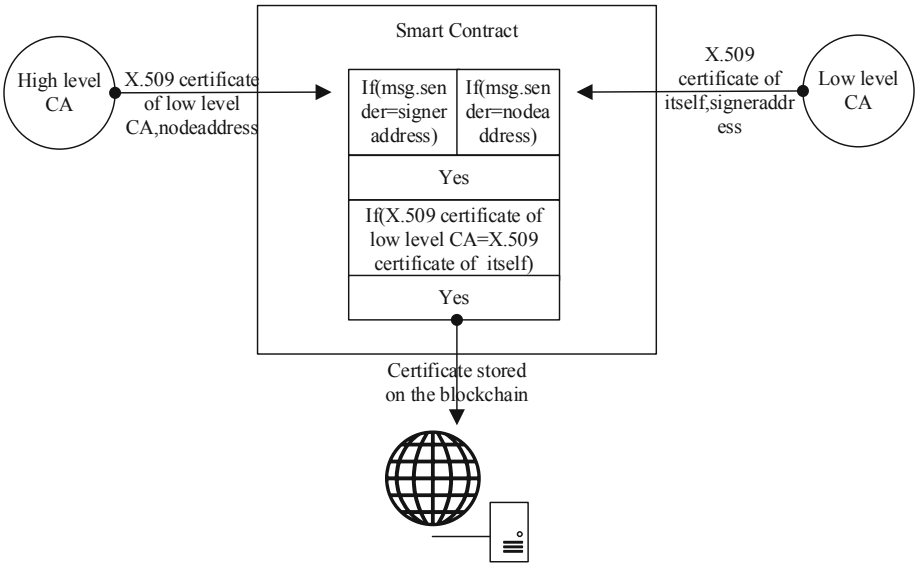

Fig. 1. Admission mechanism for CA nodes.

3.2 Construction of Certificate Chain

At present, the PKI system is centralized and hierarchical, and there are different levels among CAs. Except for root CA, every level of CA needs higher CA to issue certificates to it. Except for the root certificate, all the other certificates are intermediate certificates. We use intermediate certificates as proxies, because we must put the root certificate behind many security layers to ensure that its key is absolutely inaccessible. At the same time, in the real internet, the root CA is limited. If all authentication needs root CA to implement, its speed and efficiency can not be guaranteed. So we need to decentralize its operation and function, and finally establish trust relationship through certificate chain.

However, the current blockchain-based distributed PKI does not build such a certificate chain. Just use the blockchain for certificate storage, there is no logical relationship between certificates, they are independent of each other. As a result, the root CA can be freely accessed in the blockchain, without the protection of the security layer, and is easily attacked by malicious nodes. At the same time, the certificate stored in the blockchain cannot form a certificate chain. When the user views the certificate, the root CA trusted by the certificate cannot be determined, and the user cannot judge the trust level of the certificate. And it is also impossible to clearly define the level of CA in the blockchain, so it is impossible to determine who to apply for (Fig. 2).

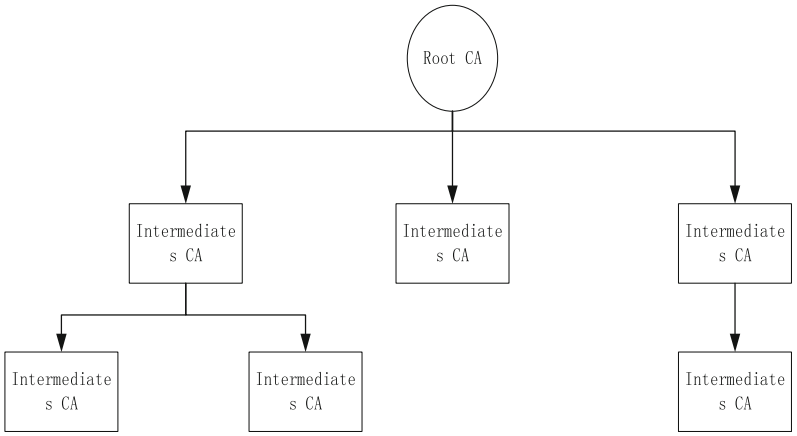

Fig. 2. Certificate chain structure

So we used the smart contract to construct the certificate chain. First, the certificates are divided into three types: root certificates, intermediate certificates, and end-user certificates. The organizational structure of CA is a tree structure. A root CAs contains multiple intermediates CAs, and intermediates can contain multiple intermediates. Both the root CAs and the intermediates CAs can issue certificates to users. The certificate that the end user uses to authenticate the public key is called end-user Certificates. The composition method of the certificate chain are shown in the Fig. 3 below.

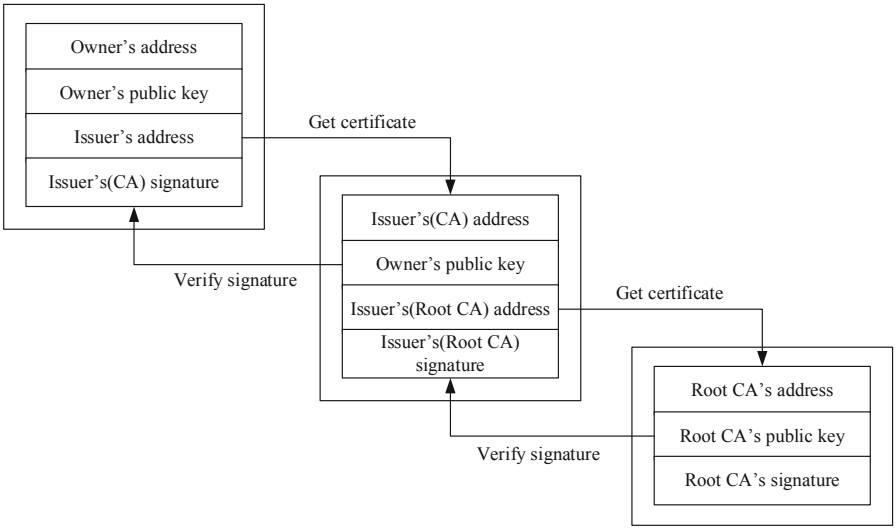

Fig. 3. The composition method of the certificate chain

The main method of forming a certificate chain in a contract is to map the CA's address to its certificate in the contract. Through its address, it can query the certificate it holds. The certificate records the public key of the signed CA as the signature. By analogy, we can form the entire certificate chain.

3.3 Conventional Certificate Management Function

As a PKI system, conventional certificate management is necessary, including the functions of issuing certificates, revoking certificates and updating certificates. The function of issuing certificates is limited to issuing end certificates. To achieve the functions of conventional certificate management, multiple functions in the contract need to work together. Take the complete step of adding a certificate to the blockchain as an example. First, the user submits an application through the Request function. After CA certification, it calls Add Certificate function to upload certificates. From the submission of applications by users to the final completion, two functions in the contract are invoked, namely Request function and Add Certificate function.

The process of revoking and updating the certificate function is slightly different from the above process. This process does not need the Request function, but only needs to verify whether the user of the function meets the requirements. The specific content will be described later.

Add Certificate Function

Adding certificate function is one of the main functions of PKI system. After a node applies for service, CA will authenticate and sign the certificate to the node, and then call adding certificate function to publish the certificate to the contract.

The parameters needed to add certificates are:

The parameters required here are divided into two parts. The first part is the data required for X.509 certificates.

Sequence number, signature algorithm identifier, signer name, validity period, principal name, principal public key.

The second part is the data needed for the contract.

Rank: The level of the requesting authentication entity in the blockchain is determined by the signature CA. And the level must be lower than the signature CA, because only the high-level node can sign the low-level node.

Nodeaddress: The address of the requesting authentication node. This address is used by CA for traceability authentication of the node. It is also used for transfer after certificate addition or for penalty after error.

Signeraddress: The address of the CA that signs the certificate. It is used to identify signature entity, transfer after certificate addition and punishment after error.

Addcertificates event records every call to the certificate addition function, and records the attributes of the certificate issued in the log. By indexing Nodeaddress, Signeraddress, principal name and principal public key, we can query the log with certificate function through these indexes, so that we can view the log in some cases. At the same time, we map Nodeaddress of each certificate, principal public key and node certificate. When we need to look at the relevant information of a node certificate, we can query it as long as we know its address or public key.

Revoke Certificate Function

Certificate revocation is used to revoke expired certificates and certificates with security problems. We stipulate here that only the node issuing the certificate can revoke the certificate. Other nodes have no authority to revoke the certificate. If the certificate revocation is successful, it will be put into the revocation certificate list.

The list of revoked certificates here stores key information for revoking certificates. Because of the inextricable modification of the blockchain, we can not delete the revoked certificate. Can only tell the user that the certificate has been revoked by recording its key information.

Certificate revocation function algorithm:

```
Function revokecertificate(address anodeaddress) public{
    Node  revokenode=list[anodeaddress];
    Require(msg.sender=revokenode.signeraddress);
    RevokeCertificates.push(Certificate({nodeaddress:anodeaddress,signeraddr
ess:msg.sender}));
    REVOKECERTIFICATES(anodeaddress,msg.sender);
    }
```

Update Certificate Function

Update certificate function is used to update certificate information. We stipulate here that only the node issuing the certificate can update the certificate. Other nodes do not have permission to update, and the certificate update will be put into the blockchain after success. Its algorithm is similar to certificate revocation function.

3.4 Error Certificate Feedback Mechanism

The feedback mechanism for discovering error certificates in existing PKI systems is limited. And there is no automated execution process. So we implement a reliable and secure error certificate feedback mechanism through two functions. As shown in Fig. 4, it is a process of error certificate feedback. The first function called is the Question function. The Question function is to upload the relevant information of the certificate to the contract, which is equivalent to proposing a proposal to vote. After uploading, each CA will see the certificate, and then the CA that meets the voting requirements will vote for the validity of the certificate according to the specific situation. After the voting time limit is reached, the contract will automatically terminate the Vote function and review the voting results. Check that the voting result meets the requirements and the contract will automatically add the certificate to the list of revoked certificates.

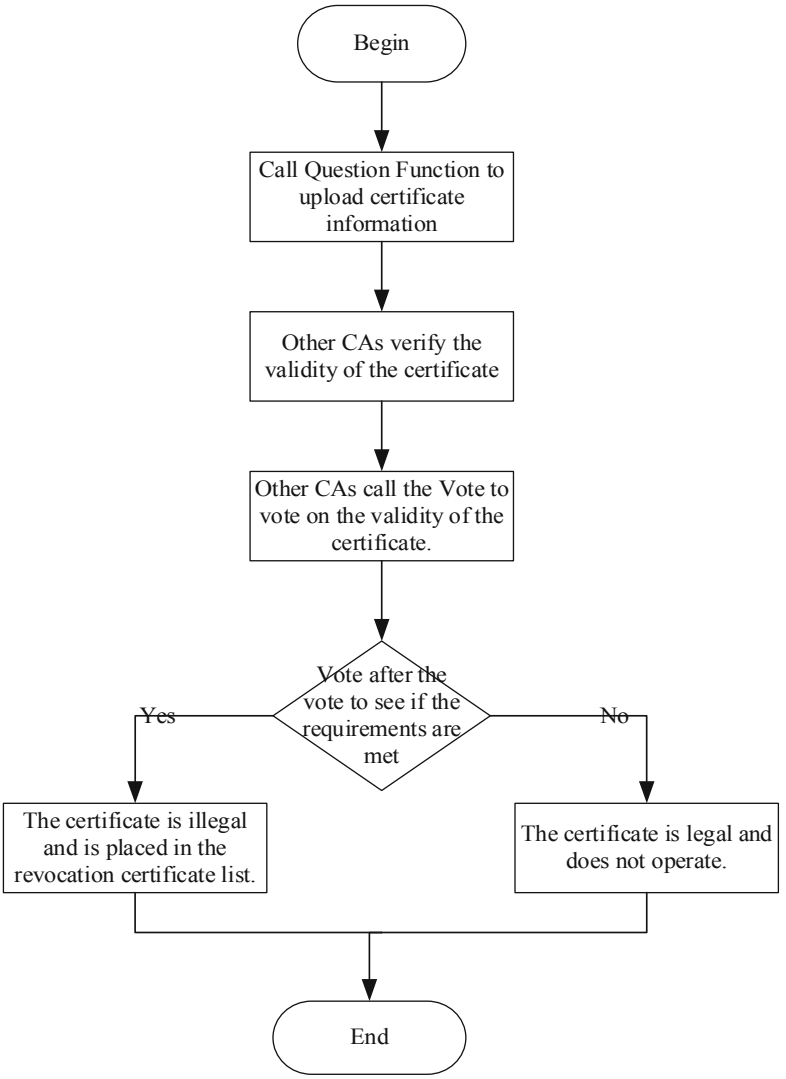

Fig. 4. Error certificate feedback mechanism

Question Function

The questioning function is based on the automation execution of smart contracts and effective economic rewards and punishments. The questioning function is not an essential part of the existing PKI system, but it is an important function in the PKI system based on smart contracts in this paper. The questioning function compensates for the lack of supervisors in the traditional PKI system, and also increases the economic penalties for the erroneous CA and the wrong certificate holder, so as to ensure the correctness of the nodes and certificates on the blockchain as much as possible.

Questioning function, as its name implies, is the function of questioning the wrong certificate. Nodes in the blockchain have the right to question any certificate. If a node thinks that a certificate is wrong, it can submit the certificate to the blockchain by calling the Questioning function, and then it needs to wait for the voting of other nodes.

Required parameters:

Questionaddress: The address of the node calling the query function. This address is recorded to facilitate economic rewards and penalties after voting.

Nodeaddress: The address of the holder of the questioned certificate. When voting, other nodes use this address to search for the certificate, so as to verify whether the certificate is in question, and then vote. After voting, the node is rewarded and punished according to its address.

Signeraddress: The address of the signer of the questioned certificate. After the voting, if there is a problem with the certificate, the node will be financially punished through this address. However, this parameter is not required because Signeraddress can be retrieved through Nodeaddress.

Vote Function

The voting function is a function of the smart contract in this paper that corresponds to the questioning function. The questioning function is simply to submit the proposal to the contract. If the proposal is approved, it must pass the voting mechanism. Only by getting enough votes can we do a series of operations such as revoking certificates and economic rewards and punishments in the blockchain.

In order to avoid voting time is too long, we will set a time limit. Within a certain time limit, the vote for a proposal will end and the number of votes will be counted. If the vote in favor exceeds half of the total number of votes, the proposal will be passed.

Vote function algorithm:

```
function vote(uint number) public{
    Node storage sender =list[msg.sender];
        require((!sender.voted)&&(sender.weight!=0)&&(sender.rank==delet
     enode[number].rank-1)&&(sender.trust>0)&&(deletenode[number].time<d
     eadline));
    sender.voted=true;
    sender.vote=number;
    deletenode[number].votecount += sender.weight;
}
```

3.5 Automatic Economic Reward and Punishment Mechanism

In order to maintain the high efficiency and stability of the whole system and ensure that each node can consciously maintain the entire blockchain, we use the smart contract to set up an automatic economic reward and punishment mechanism. Because it is executed automatically, it will not be disturbed by the outside world. As long as the node achieves the goal, it can harvest the set economic reward. Correspondingly, if the node is evil, it will automatically be punished economically, avoiding the trust flexibility problem in the traditional PKI.

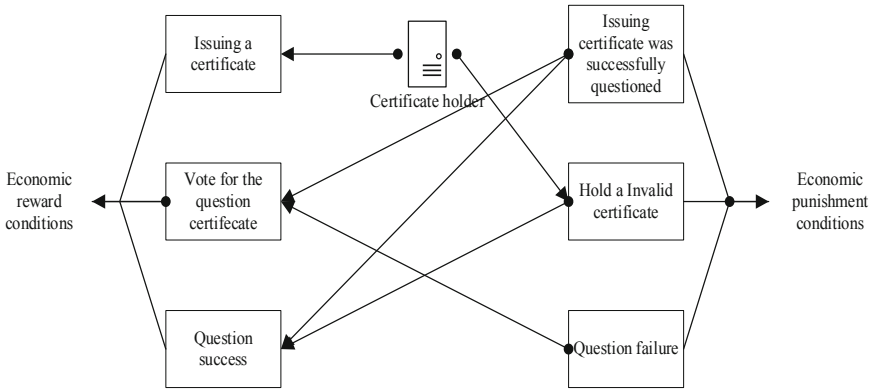

Fig. 5. The framework of the entire economic reward and punishment mechanism

As shown in Fig. 5, it shows the framework of the entire economic reward and punishment mechanism, including the conditions for economic rewards and punishments, as well as the flow of funds for penalties and rewards. The conditions required for economic rewards include: certificate certification, participation in certificate question voting, and questioning certificate success. The reward for certification comes from the node that obtains the certificate; the reward for participating in certificate questioning voting comes from the economic penalty for CA that issues and holds the wrong certificate or the economic penalty for CA that has invalid questioning. The reward for questioning the success of certificates comes from the economic penalties for CAs that issue and hold incorrect certificates. The conditions for economic punishment include: successful issuance of certificates, holding of invalid certificates and questioning invalidity. When successful issuance of certificates is questioned, not only economic punishment is imposed on the CA issuing certificates, but also all the economy of the holder of the certificates is liquidated. When the CA is questioned invalidity, the penalty amount is equally divided by the CA participating in the certificate questioning voting.

4 Feasibility and Safety Analysis

4.1 Feasibility Analysis

Lab environment: We wrote smart contracts by using remix and used a JavaScript VM to test the smart contract-based PKI system in this article. All testing processes were tested in the VM.

For the Ethereum system, the economic cost of contract construction and operation is the most important part of the feasibility. Therefore, we regard the post-deployment operating cost as the main content of the feasibility analysis.

After the deployment is complete, we tested each function in the contract, and then got the cost of each function call, we mainly analyze these costs. As of March 2019, 1 gas = 0.000000018ether [12], and 1 ether = $4.755 [13] (Table 1).

Table 1. The cost of deploying the contract and using each function.

Function	Gas	Gas in USD
Public contract	893002	$0.0763
CA certificate verification	225611	$0.0193
Add certificate	174828	$0.0149
Question	110138	$0.0094
Vote	22497	$0.0019
Revoke certificate	22001	$0.0018
Delete certificate	27099	$0.0021

Table 2. The cost of adding certificates.

Date(bytes)	Gas	Gas in USD
500	350081	$0.0299
600	415278	$0.0355
700	480231	$0.0411
800	575130	$0.0492
900	650001	$0.0556
1000	720526	$0.0616

This table shows the cost of deploying the contract and using each function, and the cost of deploying the contract requires −¥. In addition to the deployment contract, the most expensive cost are the CA certificate verification function and Add certificate function, because the two functions are the main functions of the contract, the most parameters are required to be passed in, and the node data required in the contract is passed through this function. The second is the challenge function. Because the node is to be challenged, it is necessary to upload the challenge certificate information, which also requires a relatively large overhead. Other functions are basically to operate on the data already in the blockchain, so the amount of money required is relatively small (Table 2).

The price of the certificates in the existing PKI is not uniform, but the cheapest dv certificate also needs about 100 yuan, and the general price of other certificates is about 10,000 yuan. We can see that all the costs of the contract are small compared to the fees for certification services in the PKI system, so the PKI system on smart contracts is realistic and feasible.

4.2 Safety Analysis

Previously, we mentioned several existing problems of PKI. Next, we make a security analysis of the system in this paper to solve these problems.

Certificates Can Be Issued Without the Authorization of the Domain Name Owner

We all know that smart contracts are only executed when certain conditions are met. Therefore, by setting conditions, we can realize that only when a node applies for authentication service, the corresponding certificate adding function in the contract can be invoked. And we add the process of authentication to ensure that the information of the application node and the certificate-holding node is the same. Through this series of settings, the problem of CA issuing arbitrary certificates directly without domain name owner's authorization is avoided.

Lack of Trust Flexibility

The error certificate feedback mechanism and economic reward and punishment mechanism in this paper are to solve this problem. When a CA is not credible, we use the error certificate feedback mechanism, and the certificate of an untrustworthy CA will be revoked. When a CA issues an error certificate, we use the economic reward and punishment mechanism to punish it economically, and the amount of money in the blockchain also represents its credibility. Through the cooperation of these two mechanisms, we can solve the problem of lack of trust flexibility.

Invalidity of Revocation

To solve this problem, we add a list of revoked certificates to the blockchain, which is realized by revoking certificates function in the contract. As long as the node participates in the blockchain, it needs to update the block. After updating the block, you can know the revoked certificate in the list of revoked certificates in the block. The main reason why the existing revocation does not take effect is that the revocation information can not be updated in time [14]. However, it only takes 10 s to update the block in Ethereum, so that the revocation certificate list can be updated quickly, thus solving the problem of revocation does not take effect.

4.3 Comparison with Current Research

In the current research situation, we analyze the advantages and disadvantages of each research. In view of the above research shortcomings, we propose a new smart contract-based PKI system in this paper.

Compared to IKP, Pemcor and Certcoin, these three improvements. We effectively use the automatic response and economic reward and punishment mechanisms of smart contracts to punish the issuing of wrong certificates and nodes that have experienced wrong behavior, and even revoke node certificates in the blockchain. It fills the gap of the existing distributed PKI for node management, and it is no longer simply a storage certificate.

Compared with SCPKI, we have not chosen a PGP network that cannot be used in a large number of nodes, but based on the traditional PKI to avoid the problem of trust chain breakage that occurs during large-scale application. And by setting the question function and vote function, it provides a error certificate feedback mechanism, not just offline feedback.

At the same time, we also set up a revocation certificate list, using the advantage of short blockchain update time to achieve faster update of the revocation certificate, avoiding the problem of revocation not effective.

5 Conclusions and Future Work

This paper describes a Internet web trust system based on smart contract. In the article, the specific content and implementation methods of each function are described, and the architecture is also elaborated. Finally, through experiments, the feasibility of this distributed PKI system is proved. At present, the smart contract-based PKI system of this paper has many advantages for the existing distributed PKI system, and it is very feasible, but it also has its own shortcomings. These shortcomings are mainly due to the disadvantages of the blockchain itself. The blockchain itself is difficult to complete the storage of big data, so the storage of a large number of certificates becomes the bottleneck of the system. At the same time, key recovery is also a problem that needs to be solved. In the identity management based on blockchain, we must first guarantee the security of the certificate. Secondly, the number of nodes needs to be guaranteed, and everyone can join the blockchain to view the certificate. So our consensus mechanism must ensure scalability and security. In blockchain-based identity management, it is inevitable to store some private identity information on the blockchain, but the blockchain itself is publicly accessible. Therefore, we must ensure the accessibility of the identity information on the blockchain and the privacy protection of the information [15]. The above questions are the next step for us.

Acknowledgements. This work is supported by the National Natural Science Foundations of China (grant No. 61802436 and No. 61702550) and the National Key Research and Development Plan (grant No. 2018YFB0803603 and No. 2016YFB0501901).

References

1. Is This MITM Attack to Gmail's SSL?(5). https://productforums.google.com/forum/#!msg/gmail/3J3r2JqFNTw/oHHZLJeed-HMJ. Accessed 20 Mar 2019
2. http://www.cnbeta.com/articles/tech/220690.htm. Accessed 20 Mar 2019
3. Ellison, C., Schneier, B.: Ten risks of PKI: What you're not being told about public key infrastructure. Comput. Secur. J. **16**(1), 1–7 (2000)
4. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Consulted (2008)
5. A next-generation smart contract and decentralizedapplication platform (5) (2016). https://github.com/ethereum/wiki/wiki/WhitePaper/784a271b596e7fe4e047a2a585b733d631fcf1d4. Accessed 20 Mar 2019
6. Matsumoto, S., Reischuk, R.M.: IKP: turning a PKI around with decentralized automated incentives. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 410–426. IEEE (2017)
7. Corella, F.: Implementing a PKI on a Blockchain. Pomcor Research inMobile and Web Technology (5). https://pomcor.com/2016/10/25/implementing-a-pki-on-a-blockchain/. Accessed 20 Mar 2019

8. Fromknecht, C., Velicanu, D., Yakoubov, S.: A decentralized public key infrastructure with identity retention. IACR Cryptology ePrint Archive 2014/803 (2014)
9. Wikipedia: Namecoin (5). https://en.wikipedia.org/wiki/Namecoin. Accessed 20 Mar 2019
10. Al-Bassam, M.: SCPKI: a smart contract-based PKI and identity system. In: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 35–40. ACM (2017)
11. Garfinkel, S.: PGP: Pretty Good Privacy. O'Reilly & Associates, Newton (1995)
12. https://ethstats.net/. Accessed 20 Mar 2019
13. https://coinmarketcap.co. 20 Mar 2019
14. Orman, H.: Blockchain: the emperors new PKI? IEEE Internet Comput. **22**(2), 23–28 (2018)
15. Jiang, W., Li, H., Xu, G., et al.: PTAS: Privacy-preserving Thin-client Authentication Scheme in Blockchain-Based PKI, Future Generation Computer Systems (2019). https://doi.org/10.1016/j.future.2019.01.026