Peer-to-Peer Networking and Applications (2023) 16:1211–1226
https://doi.org/10.1007/s12083-023-01455-3

A domain name management system based on account-based consortium blockchain

Genhua Lu[1] · Xiaofeng Jia[1] · Yi Zhang[1] · Jun Shao[1] · Guiyi Wei[2]

Received: 5 October 2022 / Accepted: 27 January 2023 / Published online: 13 March 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Due to the functionality for mapping domain names to IP addresses, the Domain Name System (DNS) is a critical component of the running of the Internet. However, the current DNS receives many criticisms. For instance, its centralized architecture may cause a single point of failure or power abuse. To solve this problem, many works suggest using the decentralization property of the blockchain. Nevertheless, the existing blockchain-based DNSs cannot support the whole life cycle of domain names or sealed-bid auctions for domain names. In this paper, aiming to address these problems, we propose a domain name management system by integrating some advanced cryptographic primitives, such as commitment and zero-knowledge proof, and an account-based blockchain system with the anonymous fund. The detailed security analysis indicates that our proposal holds fairness, fund-privacy, and payment-guarantee. We implement a prototype of our proposal, including the underlying account-based blockchain and the related smart contracts. The extensive experimental results conduct that our proposal is (relatively) efficient and effective.

Keywords DNS · Whole life cycle · Blockchain · Fund Privacy · Fairness

1 Introduction

Domain Name System (DNS) [1] is one of the crucial infrastructures of the Internet. In particular, people with the help from the DNS can conveniently use a human-readable domain name to visit websites instead of a hard-to-remember digital IP address. This process is usually called domain name resolution. One of the fundamental components of the DNS is the management of root domain names and top-level domain names (TLDs), which currently are maintained by the Internet Corporation for Assigned Names and Numbers (ICANN) [2]. This centralized architecture of the current DNS has received much criticism [3], such as the single point of failure [4] and power abuse. One of the famous incidents of a single point of failure on the current DNS is the DDoS attacks on Dyn [5]. The failure of the DNS provided by Dyn caused major websites, including Amazon.com, GitHub, Twitter, and Reddit, to convert unreachable via their corresponding domain names. As a result, it is natural to introduce the decentralized architecture into the current DNS to mitigate the above issues.

The past decade has witnessed a rapid development of blockchain, which has become the most famous and successful decentralized architecture. Recently, many works [6–12] have suggested introducing blockchain into DNS. Unfortunately, all of them suffer from the following problems, more or less.

- *Limited Functionality*: Most of the current solutions cannot support the whole life cycle of domain names [6, 9–12]. For example, the solution in [9] does not support the renewal process of the domain name, i.e., the domain name owner cannot prolong the use time of the domain name.

This article is part of the Topical Collection: *Special Issue on 2 - Track on Security and Privacy*
Guest Editor: Rongxing Lu

✉ Jun Shao
chn.junshao@gmail.com

Genhua Lu
genhualu22@gmail.com

Xiaofeng Jia
xiaofengjia668@gmail.com

Yi Zhang
zy03020@gmail.com

Guiyi Wei
weigy@zjgsu.edu.cn

[1] School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou 310018, China

[2] School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

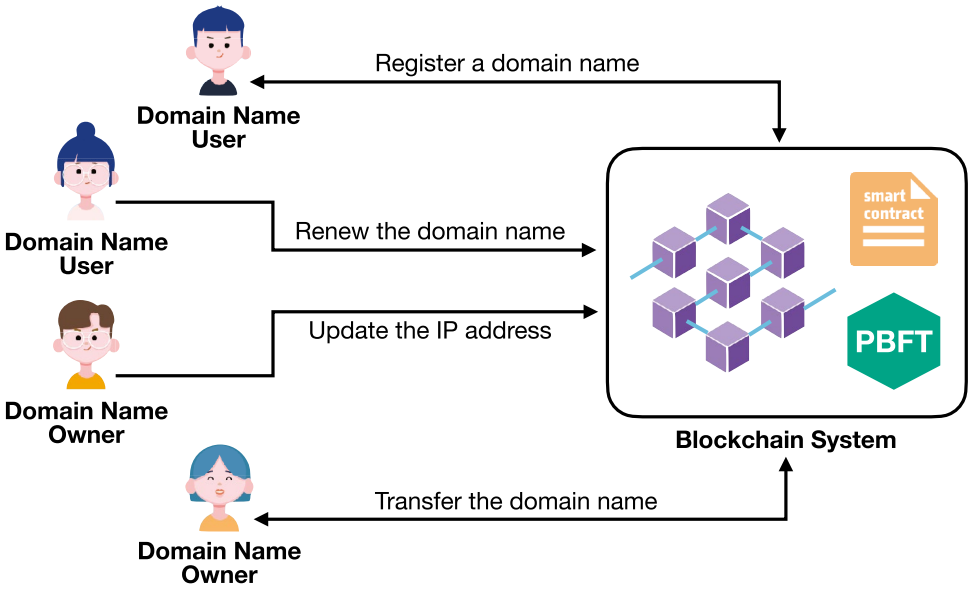


Fig. 1 The system model considered in this paper

- *Weak Security*: Some solutions [8] cannot provide fairness among the domain name users. Specifically, the domain name user bidding for the domain name later has more advantages to obtain the domain name.
- *Limited applicable objects*: Some solutions are only applicable to designated objects. For instance, the solution in [7] only supports mapping domain names to cryptocurrency addresses, but not to IP addresses.

As a result, it is desirable to design a new blockchain-based domain name management system without the above problems. In this paper, we propose a new domain name management system by employing some advanced cryptographic techniques, such as Pedersen commitment, zero-knowledge proof, and the account-based consortium blockchain with smart contract[1]. The main contributions of this paper can be summarized as follows:

- Our proposal supports the whole life cycle of domain names, including the registration, transfer, renewal, and update of domain names, in a fair and privacy-preserving way. The definition of fairness and privacy preservation can be found in Section 2.3.
- To realize the desired security properties of our domain name management system, we propose an account-based consortium blockchain system with anonymous funds and a blockchain-based Vickrey auction. To the best of our knowledge, the proposed blockchain-based Vickrey auction is the first one holding fairness and privacy preservation, which may be of independent interest.
- The security properties of our proposal can be easily obtained from those of the underlying cryptographic primitives. To show the efficiency and effectiveness of our proposal, we also provide a prototype implementation of our proposal in Python.

The remainder of the paper is structured as follows. We first formalize our system model, security model, and design goals in Section 2. In what follows, we recall some basic knowledge related to our proposal in Section 3. After that, we present our scheme in Section 4, followed by security analysis and performance evaluation in Section 5, respectively. Some related works are discussed in Section 6, and finally, conclusions are drawn in Section 7.

2 Models and design goals

In this section, we formalize our system model, security model and set out the design goals for our domain name management system.

2.1 System model

In our system model, as shown in Fig. 1, we mainly consider a typical blockchain-based domain name management scenario. We only have two kinds of entities in this paper: an account-based consortium blockchain with anonymous funds and many domain name users. The detailed descriptions of the two kinds of entities are as follows.

[1] This work extends the paper that was published at EAI AC3 2021 (Lu et al. [13])

- Blockchain system: Our system is heavily based on the blockchain system. Especially, we use the decentralization property of the blockchain to solve the problems due to the centralized architecture of the current DNS. In other words, all the nations together, instead of the single entity ICANN alone, maintain the mapping from domain names to IP addresses. To follow the fact that the internet in every country is usually controlled by network authority, we adopt the consortium structure for the underlying blockchain, and the consensus nodes in the blockchain are assumed to be the authorities. The blockchain is used to record the states of all the domain names, and everyone can map the domain name to an IP address by using the states. More details of our account-based consortium blockchain can be found in Section 4.1.
- Domain name users: Domain name users refer to those who possess domain names or are interested in registering/buying domain names in our system. They can register, sell/buy, renew, and update domain names with the help of our blockchain system. In particular, the user registers and transfers domain names in a sealed-bid auction way (i.e., Vickrey auction). More details of the Vickrey auction can be found in Section 3.1. Anyone who has enough funds in the consortium blockchain can prolong the use time of any domain name, no matter he/she is the owner of the domain name or not. In contrast, only the domain name owner can update the IP address corresponding to the domain name. In the rest of this paper, we simply say "user" instead of "domain name user".

2.2 Security model

This paper assumes that the underlying consortium blockchain is honest-but-curious as other blockchain-based systems [14–16]. In particular, the consensus nodes in the blockchain system will faithfully execute the specified steps in the system, such as verifying whether a user has the right to register a domain name, and exactly running the smart contract deployed in the blockchain. However, the consensus nodes may be curious about users' secret information, such as users' funds. On the contrary, the users are assumed to be malicious. They would try their best to register, buy/sell, renew, and update the domain name without any payment. For instance, a malicious user sells a domain name not belonging to him/her or obtain the coins from the domain name buyer without transferring the domain name.

2.3 Design goal

Given the above system and security model, our design goal is to present a blockchain-based domain name management system with the following properties.

- Fairness: In this paper, we mainly consider two kinds of fairness: transaction and bidding. The former means that the exchange between the domain name and the coins happens in a fair way. Once the owner obtains the coins, the domain name will be transferred to the buyer, and vice versa. The latter requires that all the users (bidders) have the same probability of obtaining the domain name in the Vickrey auction in terms of chronological order.
- Fund-Privacy: In our system, anybody holding a network link with any consensus node can access the data stored in the blockchain. Hence, the second security requirement of our proposal is to protect the user's privacy, such as the fund value. In particular, the adversary cannot deduce how many coins the user has by interacting with the user in domain name registration, update, transfer, and renewal processes.
- Payment-Guarantee: According to Fund-Privacy, we should protect the confidentiality of the user's fund. However, when users make payments, we need to guarantee that he/she has enough coins to cover the payment. Otherwise, the payment will fail. Therefore, in this paper, anyone can verify the sufficiency of the corresponding fund.
- Efficiency: The proposed blockchain-based domain name management system aims to relieve the problems of the current DNS. Hence, it also needs to be (relatively) efficient in terms of computational cost.

3 Preliminaries

In this section, we would like to review the concept of the Vickrey auction, Pederson commitment, and zero-knowledge proof, which will be applied in our proposal.

3.1 Vickrey auction

An auction is a process where people can buy items via the bidding method, and it usually involves an auctioneer and many bidders. The corresponding seller typically does not appear but delegates the auctioneer to sell the items. In this paper, we adopt the Vickrey auction in our proposal due to its properties. Specifically, bidders in the Vickrey auction submit their bids to the auctioneer only once in a private way. After all the bids are submitted in a preset period, the auctioneer will open the bids and decide who the winner is, and the winner only needs to pay with the second-highest bid instead of the highest one. It has been shown that the bidders in the Vickrey auction only need to focus on evaluating the value of the goods, which may bring more benefits to the bidders and the seller [17]. In our proposal, the role of the auctioneer is played by the blockchain system.

3.2 Pedersen commitment

All the parts related to payments in our proposal require Pedersen commitment, which contains stages `commit` and `reveal`. The committer commits a value to the verifier in the former stage and reveals it in the latter stage. It requires that the receiver cannot reveal the committed value before the latter stage, and the sender cannot change the committed value after the former stage.

The Pedersen commitment goes as follows. In the `commit` stage, the committer sends the commitment $\mathsf{fund} = g^x h^r$ to the verifier, where x is the committed value, r is a random number, g and h are two elements in the underlying finite cyclic group, and everybody knows g and h but no one knows $\log_h g$. Later on, the committer in the `reveal` stage sends (x', r') to the verifier. If $\mathsf{fund} = g^{x'} h^{r'}$ holds, the verifier accepts the committed value; otherwise, he/she rejects it. One of the properties of the Pedersen commitment is homomorphic addition. In particularly, given $\mathsf{fund}_1 = g^{x_1} h^{r_1}$ and $\mathsf{fund}_2 = g^{x_2} h^{r_2}$, we have that

$$\mathsf{fund}_1 \cdot \mathsf{fund}_2 = g^{x_1} h^{r_1} \cdot g^{x_2} h^{r_2} = g^{x_1+x_2} h^{r_1+r_2}.$$

The value of $\mathsf{fund}_1 \cdot \mathsf{fund}_2$ is essentially a commitment to $x_1 + x_2$.

3.3 Zero-knowledge proof

Our proposal applies the zero-knowledge proof to prove the sufficient fund for payment but without revealing the concrete fund of the user. In other words, the value of fund should be not less than the value of payment. The concrete scheme used in the proposal is original from [18].

Given $\mathsf{fund}_1 = g^{x_1} h^{r_1}$ and $\{\mathsf{fund}'_{2i} = g^{a_i \cdot 2^i} h^{r'_{2i}}\}_{i=0}^{\ell}$, the prover proves that $x_1 \geq \sum_{i=0}^{\ell} a_i \cdot 2^i$ with the knowledge of x_1, r_1, and $\{a_i \in \{0,1\}, r'_{2i}\}_{i=0}^{\ell}$, where g and h are the same as that in the Pedersen commitment, and ℓ is a positive integer larger than the bit-length of any possible value of the fund in our system. The whole process contains the following two parts.

In the first part, the prover proves that $x_1 = \sum_{i=0}^{\ell} a_i \cdot 2^i + \sum_{i=0}^{\ell} b_i \cdot 2^i$ with the conditions $\mathsf{fund}_1 = g^{x_1} h^{r_1}$, $\{\mathsf{fund}'_{2i} = g^{a_i \cdot 2^i} h^{r'_{2i}}\}_{i=0}^{\ell}$, and $\{\mathsf{fund}'_{3i} = g^{b_i \cdot 2^i} h^{r'_{3i}}\}_{i=0}^{\ell}$ by providing a signature corresponding to the public key $(h, \mathsf{fund}_1 / (\prod_{i=0}^{\ell} \mathsf{fund}'_{2i} \cdot \mathsf{fund}'_{3i}))$. Note that if $x_1 = \sum_{i=0}^{\ell} a_i \cdot 2^i + \sum_{i=0}^{\ell} b_i \cdot 2^i$, the prover knows the value of $\log_h \mathsf{fund}_1 / (\prod_{i=0}^{\ell} \mathsf{fund}'_{2i} \cdot \mathsf{fund}'_{3i}) = r_1 - \sum_{i=0}^{\ell} (r'_{2i} + r'_{3i})$, and hence he/she can generate the signature; otherwise, the prover without knowing it cannot generate the signature since $\log_h g$ is kept secret from everyone.

In the second part, the prover proves that all a_i's and b_i's belong to $\{0, 1\}$ as follows. Let us take a_i as an example. The prover only needs to provide a ring signature corresponding to the public keys (h, fund'_{2i}) and $(h, \mathsf{fund}'_{2i} / g^{2^i})$. The ring signature scheme [19] allows the verifier to check the signature's validity without revealing the public key corresponding to the real signing key. Note that if $a_i = 0$, the prover knows $\log_h \mathsf{fund}'_{2i} = r'_{2i}$; and if $a_i = 1$, the prover knows $\log_h \mathsf{fund}'_{2i} / g^{2^i} = r'_{2i}$ instead; for other cases, the prover does not know either $\log_h \mathsf{fund}'_{2i}$ or $\log_h \mathsf{fund}'_{2i} / g^{2^i}$ since $\log_h g$ is kept secret from everyone. As a result, the verifier can check whether $a_i \in \{0, 1\}$ via the ring signature.

4 Our proposal

All the steps in our proposal work for the state maintenance of domain names. Usually speaking, there are two kinds of states for one domain name: "NoOwner" and "Possessed". The former state means that no one possesses the domain name, and it can be transited into the latter by the Vickrey auction. The latter state refers to that someone owns the domain name. For this state, we have three variables, namely IP address, owner, and validity. Changing these variables may lead to a new "Possessed" state. The validity change even leads to the "NoOwner" state. We give the high-level description of the state change in Fig. 2. In the rest of this section, we will explain how the state does change. Before that, we would like to introduce an account-based blockchain system of which all the state changes run on top.

4.1 Account-based consortium blockchain

As we mentioned before, our blockchain is a consortium blockchain, where only a group of specific nodes could be the consensus ones that can add transactions to the blockchain. Our system assumes that the consensus nodes are internet authorities in each country. Furthermore, the consensus algorithm used in our blockchain system is the PBFT (Practical Byzantine Fault Tolerance) consensus algorithm, where the nodes can reach a consensus on some state via a three-round communication.

4.1.1 Account Structure

In our blockchain, there are three types of accounts, namely external account, contract account, and domain name account, as shown in Fig. 3. The external account contains two fields: $\mathsf{ID}_{\mathsf{EA}}$ and $\mathsf{EncFund}_{\mathsf{EA}}$. The former is the public key whose private key is only known to the corresponding user. The latter is with the format $g^c h^r$, where r is a random number, c is the amount of coin belonging to the account, g and h used by all users are two elements in the underlying finite cyclic group, but no one knows $\log_h g$. Similar to the Pedersen commitment, c cannot be

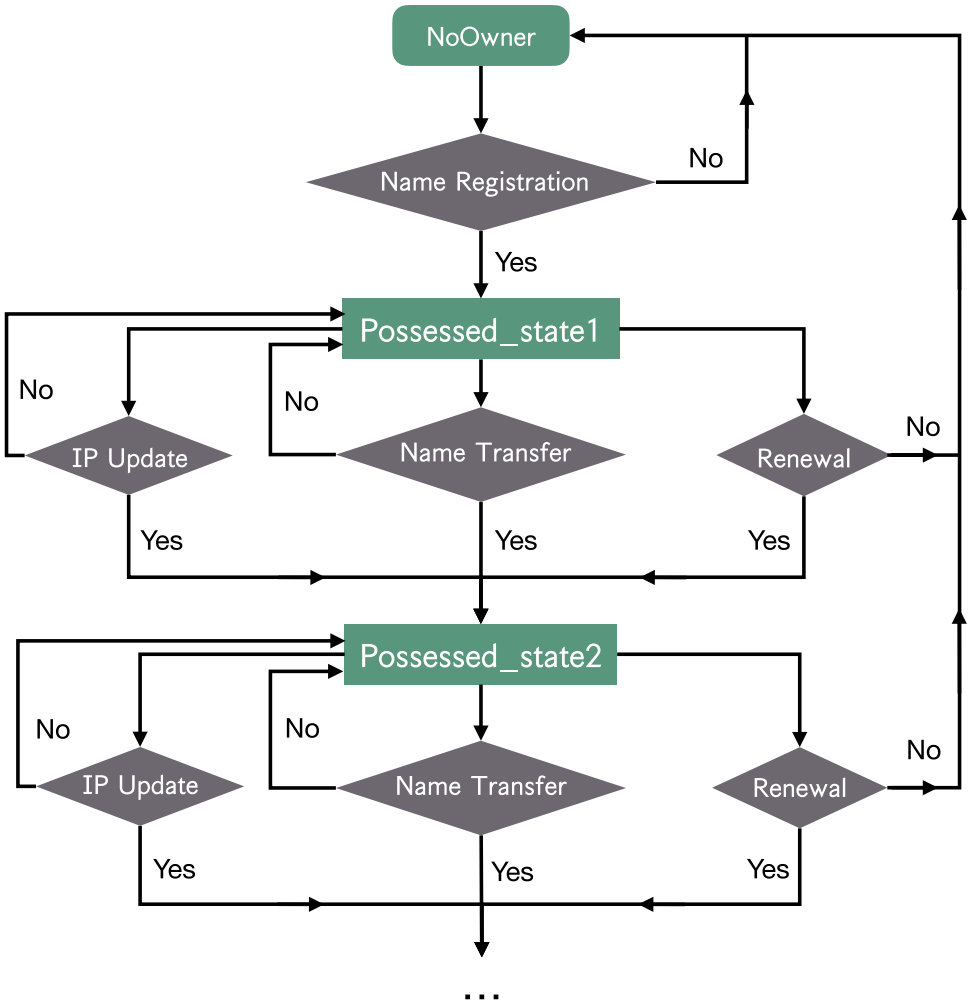


Fig. 2 State changes of a domain name

revealed by $g^c h^r$. In addition to sending and receiving coins, the external account can also establish and trigger smart contracts.

The contract account has three fields: ID_{CA}, $\mathsf{EncFund}_{CA}$, and HashCode, where $\mathsf{ID}_{CA} = H(\mathsf{ID}_{EA} \| \mathsf{HashCode})$, H is a cryptographic-secure hash function (such as SHA-256), $\mathsf{EncFund}_{CA}$ is of the same meaning with that in the external account, and HashCode is the hash of the corresponding smart contract code. The contract account can only trigger the smart contract corresponding to HashCode, and it cannot establish any smart contract.

The domain name account contains five fields, including ID_{DNA}, IP, ValidPeriod, Owner, and state as shown in Fig. 3. These five fields are the hash value, IP address, valid period, ID_{EA}, and state of the corresponding domain name, respectively. state $= 0$ denotes that the domain name is not in an active state, hence the domain name cannot be used. Otherwise, the domain name is in an active state.

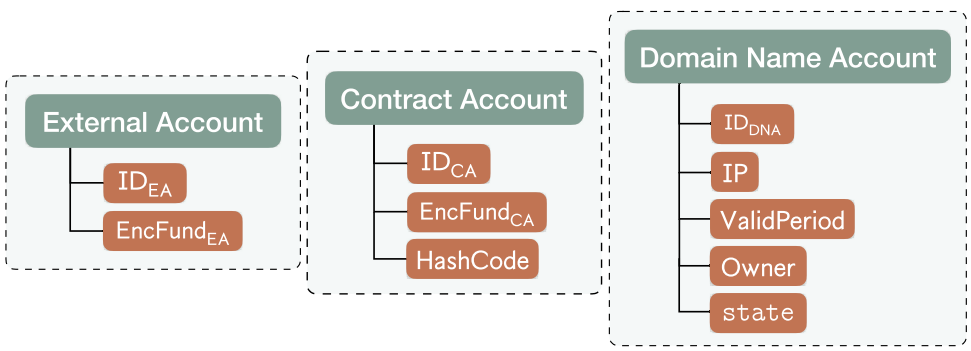


Fig. 3 The structure of three accounts considered in this paper

4.1.2 PBFT consensus algorithm

Practical Byzantine Fault Tolerance (PBFT) consensus algorithm was proposed by Castro et al. [20], aiming to improve the original BFT consensus algorithm in the context of distributed systems. Due to its low energy consumption, the PBFT has been adopted widely in the blockchain community, especially for private and consortium blockchains. In addition, there are many other variants, such as [21, 22], and [23]. However, in order to pursue universality, in this paper, we apply PBFT algorithm as the underlying consensus protocol of our blockchain system.

There are two kinds of roles in the PBFT: one master node and the other followers. They will all communicate with each other in three phases, named pre-preparation, preparation, and commitment. The ultimate goal is that honest nodes can reach a consensus on the data based on the principle of minority obeying the majority. In particular, the user initiates a transaction and sends it to the master node, who in the pre-preparation phase will store it and broadcast it to the followers if the transaction passes the verification. Upon receiving the transaction from the master node, the followers in the preparation phase will store it and broadcast it to other followers if the transaction passes the verification. Once the follower receives the same transaction from more than 2/3 of followers, it will send the confirmation message to other followers in the commitment phase. More details can be found in [20].

4.2 Description of our proposal

As shown in Fig. 3, the domain name has attributes: IP, ValidPeriod and Owner. The management of the first two attributes usually does not affect the domain name owner, which can be realized by the smart contract alone, as shown in the IP Update and Renewal phases in Sections 4.2.4 and 4.2.5, respectively. However, the change of the attribute Owner, happening in the Name Registration and Name Transfer phases, is more complex since it usually involves more users. In particular, we propose a new blockchain-based sealed-bid auction to realize the process. For clarification, we would like to present the notations involved in the whole life circle of the domain name in Table 1 and our sealed-bid auction protocol at first.

4.2.1 Blockchain-based sealed-bid auction

The underlying sealed-bid auction of our proposal is the Vickrey auction. As we introduced in Section 3.1, it contains four phases: launching, bidding, opening, and closing, which are implemented by using functions Create, Commit, Reveal and Finalize, respectively. The details of these phases can be found in the followings.

Launching phase In this phase, the auction is set up by deploying the smart contract and invoking the function Create by using (T_c, T_r, item, σ_o). Note that the function Create in each smart contract could have only one successful invocation, since the bidding item could be in only one auction at every moment. When the consensus nodes in the blockchain receive the function call, they check whether (T_c -now) $\in \mathbb{R}_c$, whether (T_r- T_c) $\in \mathbb{R}_{cr}$, whether the `item` has Owner, the validity of σ_o, and whether the `item` is in an active state. If all of them are checked successfully, the consensus nodes initialize the dictionary `bidders[ ][ ]` and set the `state` about the `item` is in an active state. Otherwise, they stop running the function. The pseudo-code of the function Create can be found in Fig. 4.

Bidding phase Once the auction is launched successfully, bidders, who are interested in the bidding item, will invoke the function Commit with (ID_b, $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$, σ_b, $\mathsf{ZKP}_{c,\mathsf{fund}}$). When the consensus nodes in the blockchain receive the function call, they check whether the bidder has submitted the bid by inspecting whether the dictionary `bidders[ ][ ]` contains ID_b or not, whether the call is still in the valid period according to T_c, and the validity of (σ_b, $\mathsf{ZKP}_{c,\mathsf{fund}}$). If all of them are checked successfully, the consensus nodes record the commitment of the bid ($\{\mathsf{fund}'_i\}_{i=0}^{\ell}$) into the dictionary `bidders`[ID_b]["*commit*"], reduce the committed bid $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$ from the corresponding $\mathsf{EncFund}_{\mathsf{EA}}$ in the external account, and add the committed bid $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$ into the $\mathsf{EncFund}_{\mathsf{CA}}$ in the contract account. Otherwise, they stop running the function. The pseudo-code of the function Commit can be found in Fig. 4.

Opening phase When the *bidding phase* stops according to T_c, the bidders can start to reveal their bids by invoking the function Reveal with (ID_b, $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$, σ_v). When the consensus nodes in the blockchain receive the function call, they check whether the call is still in the valid period according to T_r, whether the dictionary `bidders[][]` contains ID_b or not, whether σ_v is valid, and whether all the $\mathsf{fund}'_i = g^{\bar{c}'_i \cdot 2^i} h^{\bar{r}'_i}$ for $i = 0, \cdots, \ell$ hold. If all of them are checked successfully, the consensus nodes add the bid value $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$ into the dictionary `bidders`[ID_b]["bid"]. Otherwise, they stop running the function. The pseudo-code of the function Reveal can be found in Fig. 4.

Closing phase If the time arrives at T_r, the bidders can invoke the function Finalize to get the domain name and terminate the auction. Note that anyone can invoke the function Finalize and deduce who is the winner according to $\{\bar{c}'_i\}$ recorded in the dictionary `bidders[][]`. When the consensus nodes in the blockchain receive the function call, they check whether it is the time to finalize the auction according to T_r and whether the `state` about the `item` is in an active state. If both of them are checked successfully, they continue running the function. Otherwise, they stop running. The pseudo-code of the function Finalize can be found in Fig. 4.

Table 1 Notations used in the smart contract

Notations	Description	Notations	Description
$T_c, T_r, \mathtt{t}$	The respective deadlines for committing the bid, revealing the bid, and paying the transfer fee.	state	$\{0, 1\}$, 1 denotes that the domain name is in an active state, and 0 has the opposite meaning.
$\mathtt{item}$	The bidding item for the blockchain-based sealed-bid auction.	$\mathbb{R}_c, \mathbb{R}_{cr}$	The ranges for (T_c-now) and T_r-T_c, respectively.
$\{\mathsf{fund}'_i\}_{i=0}^{\ell}$	A commitment❷ for the bid value $\sum_{i=0}^{\ell} c'_i \cdot 2^i$.	$\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$	The values used to open the commitment $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$.
σ_i❶	A signature that is generated by user i, who could be the invoker of function CREATE, COMMIT, REVEAL, TRANSFER, RECEIVE, or UPDATE.	$\mathsf{ID_i}$	The $\mathsf{ID_{EA}}$ field in the external account of the user i. Specifically, $\mathsf{ID_b}$, $\mathsf{ID_o}$, and $\mathsf{ID_v}$ are the public keys of the bidder, the owner, and the recipient, respectively.
$\mathtt{bidders[][]}$	A dictionary with three objects: a key "$\mathsf{ID_{EA}}$", a value "commitment❷" for the bid value $\sum_{i=0}^{\ell} c'_i \cdot 2^i$, and a value "($\{(c'_i, r'_i)\}_{i=0}^{\ell}$)" for revealing the bid value $\sum_{i=0}^{\ell} c'_i \cdot 2^i$ corresponding to the commitment $\{\mathsf{fund}'_i = g^{c'_i \cdot 2^i} h^{r'_i}\}_{i=0}^{\ell}$.	$\mathsf{ZKP}_{c,\mathsf{fund}}$	A zero-knowledge proof for the statement $c \geq \sum_{i=0}^{\ell} c'_i \cdot 2^i \geq 0$, where $\mathsf{EncFund_{CA}} = g^c h^r$ in the contract account, and the commitment $\{\mathsf{fund}'_i = g^{c'_i \cdot 2^i} h^{r'_i}\}_{i=0}^{\ell}$ of the bid. And it is easy to see that we can obtain $\mathsf{ZKP}_{c,\mathsf{fund}}$ as the steps in Section 3.3.
$\mathsf{EncFund_{EA_i}}$	The $\mathsf{EncFund_{EA}}$ field in the external account associated with the user i. Specifically, $\mathsf{EncFund_{EA_b}}$, $\mathsf{EncFund_{EA_w}}$, $\mathsf{EncFund_{EA_o}}$, and $\mathsf{EncFund_{EA_v}}$ are the $\mathsf{EncFund_{EA}}$ in the external accounts are associated with the bidder, the winner, the owner, and the invoker, respectively.	c_p, c_t, c_r	The penalty fee, transfer fee, and renewal fee are in function FINALIZE (in Section 4.2.1), function TRANSFER (in Section 4.2.3) and RECEIVE (in Section 4.2.3), and function RENEWAL (in Section 4.2.5), respectively.
c'	The second-highest bid value $c' = \sum_{i=0}^{\ell} \bar{c}'_i \cdot 2^i$.		

❶ Similarly, we use sk_i to represent the signing key associated with user i. Therefore, $\sigma_o = sig(sk_o, T_c||T_r||\mathtt{item})$, $\sigma_b = sig(sk_b, \mathsf{ID_b}||\{\mathsf{fund}'_i\}_{i=0}^{\ell}||\mathsf{ZKP}_{c,\mathsf{fund}})$, $\sigma_v = sig(sk_v, \mathsf{ID_b}||\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell})$, $\sigma_t = sig(sk_t, \mathsf{ID_o}||\mathtt{name}||\mathsf{ID_v}||c_t||\mathtt{t})$, $\sigma_r = sig(sk_r, \mathtt{name}||c_t||\mathsf{ZKP}_{c,\mathsf{fund}})$, and $\sigma_u = sig(sk_u, \mathtt{name}||\mathtt{ip})$, where $sig()$ is a signature algorithm, such as ECDSA, sk_i is the signing key corresponding to the $\mathsf{ID_{EA}}$ of the user calling the function. Note that if there is no owner for the underlying $\mathtt{item}$, σ_o is a null value

❷ The "commitment" refers to $\{\mathsf{fund}'_i = g^{c'_i \cdot 2^i} h^{r'_i}\}_{i=0}^{\ell}$

function CREATE(T_c, T_r, item, σ_o)
  if (T_c-now) $\in \mathbb{R}_c$ and (T_r - T_c) $\in \mathbb{R}_{cr}$ **then**
    if item.Owner != null **then**
      if σ_o is valid **then**
        if item.state == 0 **then**
          Initialize bidders[][]
          Set item.state = 1
    else if item.state == 0 **then**
      Initialize bidders[][]
      Set item.state = 1
  return

function COMMIT(ID_b, $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$, σ_b, $\mathtt{ZKP}_{c,\mathsf{fund}}$)
  if (ID_b not in bidders[][]) and (now $\leq T_c$) **then**
    if (σ_b, $\mathtt{ZKP}_{c,\mathsf{fund}}$) is valid **then**
      bidders[ID_b]["*commit*"] = $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$
      $\mathsf{EncFund}_{\mathsf{EA}_b} = \mathsf{EncFund}_{\mathsf{EA}_b} / \{\mathsf{fund}'_i\}_{i=0}^{\ell}$
      $\mathsf{EncFund}_{\mathsf{CA}} = \mathsf{EncFund}_{\mathsf{CA}} \times \{\mathsf{fund}'_i\}_{i=0}^{\ell}$
  return

function REVEAL(ID_b, $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$, σ_v)
  if (ID_b in bidders[][] and σ_v is valid) **then**
    if $T_c <$ now $\leq T_r$ **then**
      if bidders[ID_b]["*commit*"] $== g^{\bar{c}'_i \cdot 2^i} h^{\bar{r}'_i}$ for $i = 0, \cdots, \ell$ **then**
        bidders[ID_b]["*bid*"] = $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$
  return

function FINALIZE( )
  if (now $> T_r$ and item.state == 1) **then**
Find the winner and the second-highest $\sum_{i=0}^{\ell} \bar{c}'_i \cdot 2^i$
    for each ID_b in bidders[][] **do**
      $\mathsf{EncFund}_{\mathsf{CA}} = \mathsf{EncFund}_{\mathsf{CA}} / \{\mathsf{fund}'_i\}_{i=0}^{\ell}$
      $\mathsf{EncFund}_{\mathsf{EA}_b} = \mathsf{EncFund}_{\mathsf{EA}_b} \times \{\mathsf{fund}'_i\}_{i=0}^{\ell}$
      if bidders[ID_b]["*bid*"] does not exist **then**
        Set $\mathsf{EncFund}_{\mathsf{EA}_b} = \mathsf{EncFund}_{\mathsf{EA}_b} / g^{c_p}$
      if (ID_b == winner) **then**
        Set $\mathsf{EncFund}_{\mathsf{EA}_w} = \mathsf{EncFund}_{\mathsf{EA}_w} / g^{c'}$
        if item.Owner != null **then**
          Set $\mathsf{EncFund}_{\mathsf{EA}_o} = \mathsf{EncFund}_{\mathsf{EA}_o} \times g^{c'}$
          Set IP = null
          Set Owner = ID_b
        else
          Set Owner = ID_b
        Set item.state = 0
  return

Fig. 4 Functions in the smart contract of the blockchain-based sealed-bid auction

In fact, we mainly deal with three cases for the bidders in the function FINALIZE. The first one is that the bidders follow the specific steps but losing this auction, the consensus nodes will deduct their funds from the contract account, and return to their external accounts. The second one is that if the bidders fail to open their bids in the *opening phase*, in addition to returning the corresponding fund from the contract account to their external account, the consensus nodes will also deduct a certain percentage of the penalty from the funds in the corresponding external accounts. The last one is for the winner whose money of the second-highest bid value will be destroyed in our account-based consortium blockchain, and the item will belong to the winner. However, if the item has owner, the winner's whole money of the second-highest bid value will be added to the owner's external account, and the item will belong to the winner.

4.2.2 Name registration

Now, we can utilize the sealed-bid auction above to register a domain name. In Name Registration, any user can only register a domain name he/she is interested in by deploying an auction smart contract. The relevant steps are as follows.

- Any user can use his/her external account to register a domain name by invoking the function CREATE in the corresponding smart contract with (T_c, T_r, name, null) attached. By doing so, if there are other interested users, they can join in this sealed-bid auction to compete for registering this domain name through anonymous bidding.
- After that, the interested users will use their external accounts to invoke the function COMMIT with (ID_b, $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$, σ_b, $\mathsf{ZKP}_{c,\mathsf{fund}}$) attached. Note that each user cannot change his/her bid after submitting his/her bid successfully in an auction. By doing so, it proves to other users in the blockchain that he/she has bid on the domain name.
- Then, if T_c now $\leq T_r$, the users can use their external accounts to invoke the function REVEAL in the smart contract with (ID_b, $\{(\bar{c}'_i, \bar{r}'_i)\}_{i=0}^{\ell}$, σ_v) to open their bids. By doing so, it proves to other users in the blockchain that their bids for the domain name have not been tampered with.
- Finally, if now $> T_r$ and the state about the domain name is in an active auction contract, any user can use his/her external account to invoke the function FINALIZE to get the domain name, get the punishment or return $\{\mathsf{fund}'_i\}_{i=0}^{\ell}$ to their $\mathsf{EncFund}_{\mathsf{EA}}$'s.

4.2.3 Name transfer

In Name Transfer, users can transfer domain names to other users. There are two methods of domain name transfer in our domain name management system: direct and

```
1: function TRANSFER(ID_o, name, ID_v, c_t, t, σ_t)
2:     if (ValidPeriod is valid and name.state == 0) then
3:         if (σ_t is valid and Owner == ID_o) then
4:             Set recipient = ID_v
5:             Set owner = ID_o
6:             Set transfer fee = c_t
7:             Set name.state = 1
8:     return
9: function RECEIVE(name, c_t, σ_r, ZKP_{c,fund})
10:    if t is valid then
11:        if (σ_r and ZKP_{c,fund} is valid) then
12:            Set EncFund_{EA_v} = EncFund_{EA_v} / g^{c_t}
13:            Set EncFund_{EA_o} = EncFund_{EA_o} * g^{c_t}
14:            Set IP = null
15:            Set Owner = ID_v
16:            Set name.state = 0
17:    return
```

Fig. 5 Direct transfer method in Name Transfer phase in our proposal

indirect. They are free to choose which method to use for domain name transfer.

Direct In the direct transfer method, the owner can transfer the domain name to the recipient directly, if they have reached an agreement on the transfer fee. It works as follows.

- Firstly, the owner can invoke the function TRANSFER (as shown in Fig. 5) with (ID_o, name, ID_v, c_t, t, σ_t) attached. When the consensus nodes in the blockchain receive the function call, they check whether the the name's ValidPeriod is valid, whether the state about the name is not in an active state, whether σ_t is valid, and whether the Owner of the name is equal to ID_o. If all of them are checked successfully, consensus nodes run the rest of the function.
- After that, any user can invoke the function RECEIVE with (name, c_t, σ_r, $\mathsf{ZKP}_{c,\mathsf{fund}}$) to pay the transfer fee. When the consensus nodes in the blockchain receive the function call, they check whether t and (σ_r, $\mathsf{ZKP}_{c,\mathsf{fund}}$) are valid. If all of them are checked successfully, the consensus nodes will deduct c_t from the $\mathsf{EncFund}_{\mathsf{EA}}$ of the user's external account, and add the same amount of money to the owner's external account. At the same time, the ownership of the domain name will be transferred to the recipient.

Indirect We illustrate the sealed-bid auction with which the indirect transfer method relies in Fig. 4. However, unlike in the Name Registration, during the Name Transfer, only the domain name owner can call the function CREATE to launch the auction.

```
1: function UPDATE(name, ip, σ_u)
2:     if (ValidPeriod, ip and σ_u are valid) then
3:         Set IP = ip
4:     return
```

Fig. 6 IP Update phase in our proposal

4.2.4 IP update

In this phase, when the user decides to change the mapping relationship between a domain name and its IP address in a domain name account, he/she only needs to invoke the function UPDATE with (name, ip, σ_u). When the consensus nodes in the blockchain receive the function call, they check the validity of (ValidPeriod, ip, σ_u). If all of them are valid, consensus nodes update the IP address of the name. The pseudo-code of the function UPDATE can be found in Fig. 6.

4.2.5 Renewal

In this phase, the owner or even anyone else can renew the corresponding ValidPeriod field in the domain name account by invoking the function RENEWAL with (name, c_r, $\mathsf{ZKP}_{c,\mathsf{fund}}$) at any time. When the consensus nodes in the blockchain receive the function call, they check whether the corresponding ValidPeriod and $\mathsf{ZKP}_{c,\mathsf{fund}}$ are valid. If all of them are verified successfully, consensus nodes will deduct c_r in the $\mathsf{EncFund}_{\mathsf{EA}}$ of the caller's external account, and extend a period of ownership for the name. The pseudo-code of the function RENEWAL can be found in Fig. 7.

5 Analysis of our proposal

In this section, we will analyze the security properties of our proposal one by one as listed in Section 2.3 and show the (relative) efficiency by experiments.

5.1 Security analysis

Fairness The fairness of our proposal contains the transaction fairness and bidding fairness. As other blockchain-based

```
1: function RENEWAL(name, c_r, ZKP_{c,fund})
2:     if (ValidPeriod and ZKP_{c,fund} are valid) then
3:         Set EncFund_{EA_v} = EncFund_{EA_v} / g^{c_r}
4:         Set ValidPeriod += a period of time
5:     return
```

Fig. 7 Renewal phase in our proposal

systems [14–16], we simply assume that our PBFT-based consortium blockchain is secure. In this case, the consensus nodes should faithfully execute smart contracts involved in the maintenance of domain names. Furthermore, the codes in lines 32-43 in the function FINALIZE, lines 12-16 in the function RECEIVE, and lines 3-4 in the function RENEWAL show that the transfer of the money and the domain name should happen or not at the same time. Hence, we have the transaction fairness immediately.

Regarding the bidding fairness of our proposal, we only need to show that no one can reveal the bid during the bidding phase, which is guaranteed by Theorem 1.

Theorem 1 *The bidding values are confidential in the Bidding Phase.*

Proof According to the description of proposed blockchain-based sealed-bid auction, only $\{\mathsf{fund}'_i, \sigma'_i\}_{i=0}^{\ell}$ contain the information of the bidding value, where $\mathsf{fund}'_i = g^{c'_i} h^{r'_i}$, and σ'_i is the ring signature corresponding to public keys (h, fund'_i) and $(h, \mathsf{fund}'_i/g^{2^i})$. In this proof, we only need to prove that these data won't leak any information about the bid $c' = \sum_{i=0}^{\ell} c'_i \cdot 2^i$.

For simplicity, we here only show that the pair $(\mathsf{fund}'_0, \sigma'_0)$ won't reveal any information on c'_0. We have the following steps.

- We randomly choose $\{0, 1\}$ as the challenge messages to the challenge oracle in confidentiality security of Pedersen commitment. The oracle will output a challenge commitment fund^*. After that, we set $\mathsf{fund}'_0 = \mathsf{fund}^*$. Note that, the committed value in fund^* is set as c'_0 implicitly.
- We send (h, fund'_0) and $(h, \mathsf{fund}'_0/g)$ as the challenge public keys to the challenge oracle in anonymity security of the underlying ring signature. The oracle will output a challenge ring signature σ^*. After that, we set $\sigma'_0 = \sigma^*$.
- Now, assume there exists an adversary that can get c'_0 from $(\mathsf{fund}'_0, \sigma'_0)$. With this, we can correctly response the guess oracles for Pedersen commitment and the underlying ring signature, which however is impossible.

Hence, we get the theorem.

Fund-Privacy Fund-privacy requires that the concrete value of user's fund won't be revealed during the maintenance process of domain names. This property can be easily obtained by using Theorem 2.

Theorem 2 *The fund value c won't be revealed in our proposal.*

Proof According to the description of our proposal, anyone can get the value of $g^c h^r$ and ring signature σ''_i of public keys (h, fund''_i) and $(h, \mathsf{fund}''_i/g^{2^i})$ for $i = 0, \cdots, \ell$, where $g^c h^r = \prod_{i=0}^{\ell} \mathsf{fund}''_i$ and $\mathsf{fund}''_i = g^{c''_i} h^{r''_i}$. Note that, one may argue that $g^c h^r = \prod_{i=0}^{\ell} \mathsf{fund}''_i$ should be $g^c h^r = \prod_{i=0}^{\ell} \mathsf{fund}'_i \mathsf{fund}''_i$. However, the value c_i in fund'_i will be revealed in the opening phase. Hence, we use $g^c h^r = \prod_{i=0}^{\ell} \mathsf{fund}''_i$ instead of $g^c h^r = \prod_{i=0}^{\ell} \mathsf{fund}'_i \mathsf{fund}''_i$. In this case, similar with the proof of Theorem 1, we can get the theorem.

Payment-Guarantee Payment-guarantee means that users have the ability to complete payment. This property can be obtained according to Theorem 3.

Theorem 3 *The auction winner always has the enough money to pay the wined domain name.*

Proof According to the description of our proposal, the bidder's fund c will be reduced by the bided value $c' = \sum_{i=0}^{\ell} c'_i \cdot 2^i$ in the bidding phase. Only if $c \le c' \le 0$, then we have the theorem.

For $c' \le 0$, we only need to show that $c'_i \in \{0, 1\}$. Assume that the malicious bidders generate a ring signature of public keys (h, fund'_i) and $(h, \mathsf{fund}'_i/g^{2^i})$, i.e., $c'_i \notin \{0, 1\}$. In other words, the malicious can generate a ring signature using a secret key whose public key is not in the public key set, which is against the unforgeability of the underlying ring signature.

Using the similar method, we can have $c \le c'$. Hence, we get the theorem.

5.2 Comparison and performance evaluation

In this part, we would like to give the comparison between our proposal and the previous works, along with an evaluation.

5.2.1 Comparison with existing schemes

Our proposal is related to two kinds of schemes, namely blockchain-based DNS and blockchain-based auction. Hence, we give two comparisons in Tables 2 and 3, respectively. From Table 2, we can see that only ENS [7], Handshake [8], and our proposal can support the auction method for domain name management. Furthermore, our proposal is the only one holding fairness, fund-privacy, payment-guarantee, and efficiency at the same time. From Table 3, we can see that only ShadowEth [24], the solution in [25], and our proposal do not require any (semi-) trusted auctioneer. Furthermore, ShadowEth and our proposal can simultaneously hold fairness, fund-privacy, payment-guarantee, and efficiency. However, ShadowEth makes use of the trusted

Table 2 Comparison between our proposal and existing blockchain-based DNS solutions

Schemes	Auction	Whole Life Cycle of Domain Names	Properties			
			Fairness	Fund-Privacy	Payment-Guarantee	Efficiency
Namecoin [28]	✗	✓	✗	✗	✓	✓
Blockstack [29]	✗	✓	✗	✗	✓	✓
ENS [7]	✓	✓	✓	✗	✓	✓
ConsortiumDNS [6]	✗	✗	✗	✗	✓	✓
TD-Root [12]	✗	✗	✗	✗	✗	✓
DNSonChain [30]	✗	✗	✗	✗	✗	✓
Handshake [8]	✓	✓	✗	✓	✓	✓
This paper	✓	✓	✓	✓	✓	✓

execution environment (TEE), whose security is still controversial [26, 27]. In summary, our proposal is the best one in terms of holding properties.

5.2.2 Performance evaluation

We evaluate the performance of our proposal in terms of off-chain cost and on-chain cost. In particular, we use the ECDSA with the secp256k1 curve as the underlying signature algorithm as that in most blockchain systems. In our python-type prototype[2], users can only invoke the specified smart contracts but cannot modify them. We test our prototype on an Ubuntu virtual machine with version 16.04, 4 processors, and 3GB of RAM allocated by the Windows 10 operating system. Additionally, the Ubuntu virtual machine runs on an Intel(R) Core(TM) i5-7300HQ CPU @2.5GHz Windows Platform with 16.0GB RAM. Note that, since PBFT is a well-recognized consensus mechanism for consortium blockchain. Therefore, in the following experiments, we only count the computational cost of off-chain and on-chain. All the numbers in Figures 8-10 are conducted 50 times and the averages are recorded. As we can see from the figures, our proposal is (relatively) efficient in terms of off-chain cost and on-chain cost.

Off-chain Cost The main computational cost for the off-chain part of our proposal is due to the generation of commitments and zero-knowledge proofs. In particular, in the function Commit, the bidder has to generate a commitment for the fund and a zero-knowledge proof for a statement. The statement shows that the bidder has enough money to cover his/her bidding. Similarly, in functions Receive and Renewal, users also have to generate zero-knowledge proofs off-chain for c_t and c_r. As illustrated in Fig. 8, we can see that the computational cost of the off-chain parts in "Renewal", "Receive",

Table 3 Comparison between our proposal and existing blockchain-based auction solutions

Schemes	Sealed-bid Auction	No (Semi-) Trusted Auctioneer	Properties			
			Fairness	Fund-Privacy	Payment-Guarantee	Efficiency
Hawk [31]	✓	✗	✓	✓	✓	✗
ShadowEth❶ [24]	✓	✓	✓	✓	✓	✓
Strain [32]	✓	✗	✗	✓	✓	✓
Galal and Youssef [33]	✓	✗	✓	✓	✗	✗
Nguyen and Thai [25]	✗	✓	✓	✓	✗❷	✓
Qusa et al. [34]	✓	✗	✗	✗	✗	✗
SSBAS-FA [35]	✓	✗	✓	✓	✓	✓
This paper	✓	✓	✓	✓	✓	✓

❶ The security of implementation of TEE is still controversial

❷ Authors assume that all parties have enough funds in their accounts for making deposits to the smart contract

[2] https://github.com/EmmaLu-ux/paperDemo

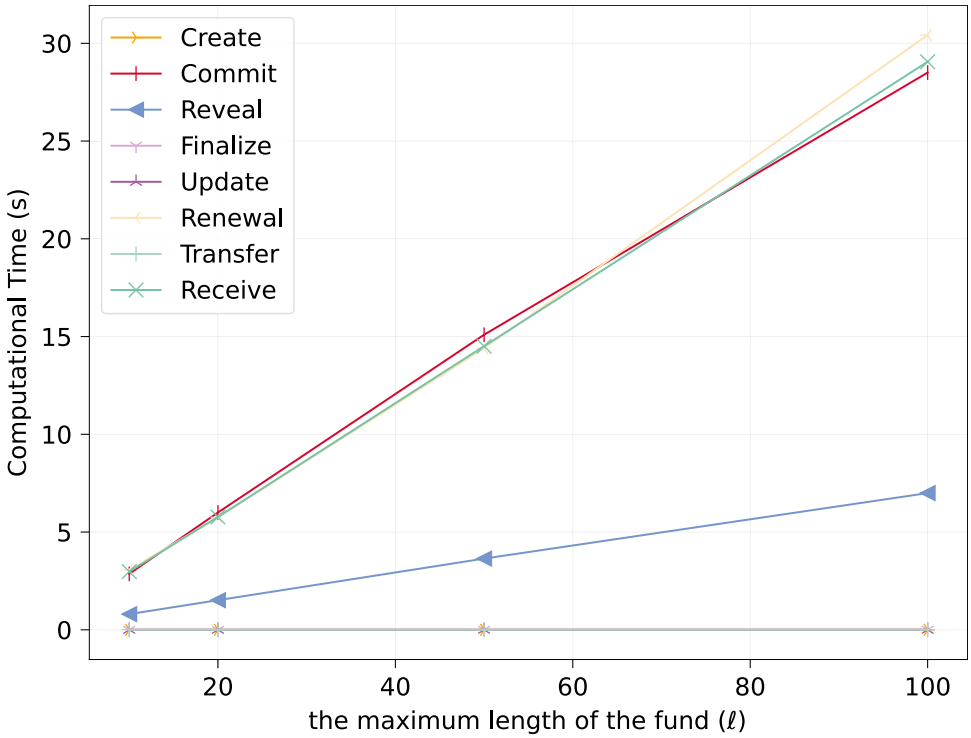


Fig. 8 The off-chain computational cost

"Commit" and "Reveal" is proportional to ℓ, while it is always acceptable. Furthermore, we can see that the computational cost of the off-chain parts in "Create", "Finalize", "Update" and "Transfer" are almost the same and computationally negligible. In addition, it is worth noting that, all the computational cost in "Finalize" is related to on-chain.

5.2.2.1 On-chain cost For the on-chain cost, which is mainly due to the computational cost for the on-chain part of our system, especially, the verification of zero-knowledge proofs in the functions Commit, Receive, and Renewal, and the verification of commitments in the function Reveal. As shown in Fig. 9, the trend of the computational cost of

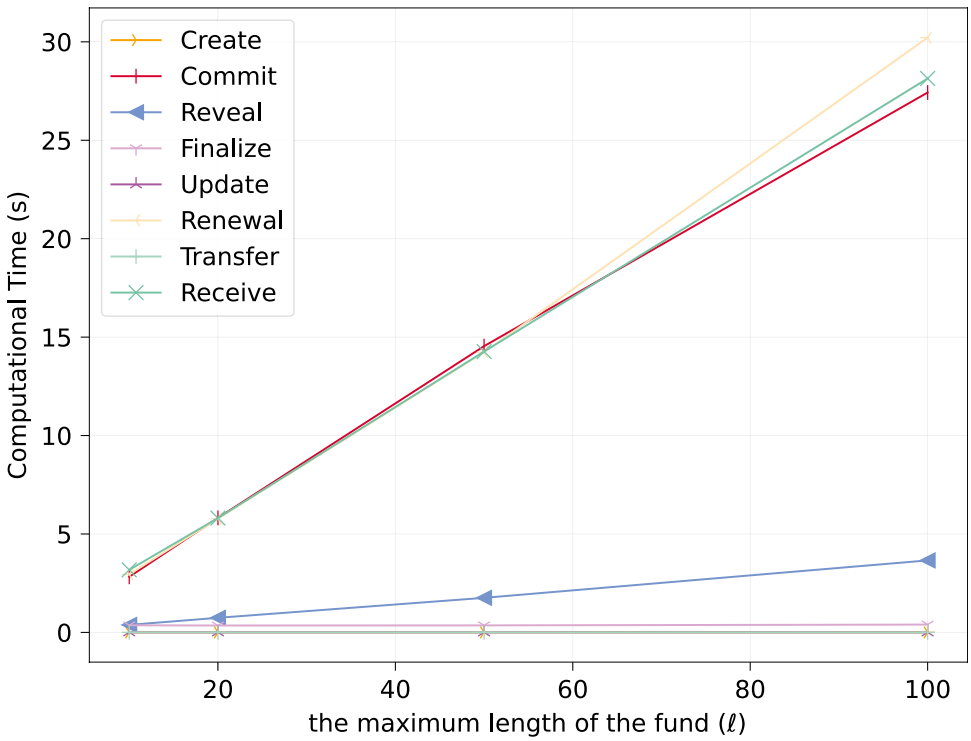


Fig. 9 The on-chain computational cost

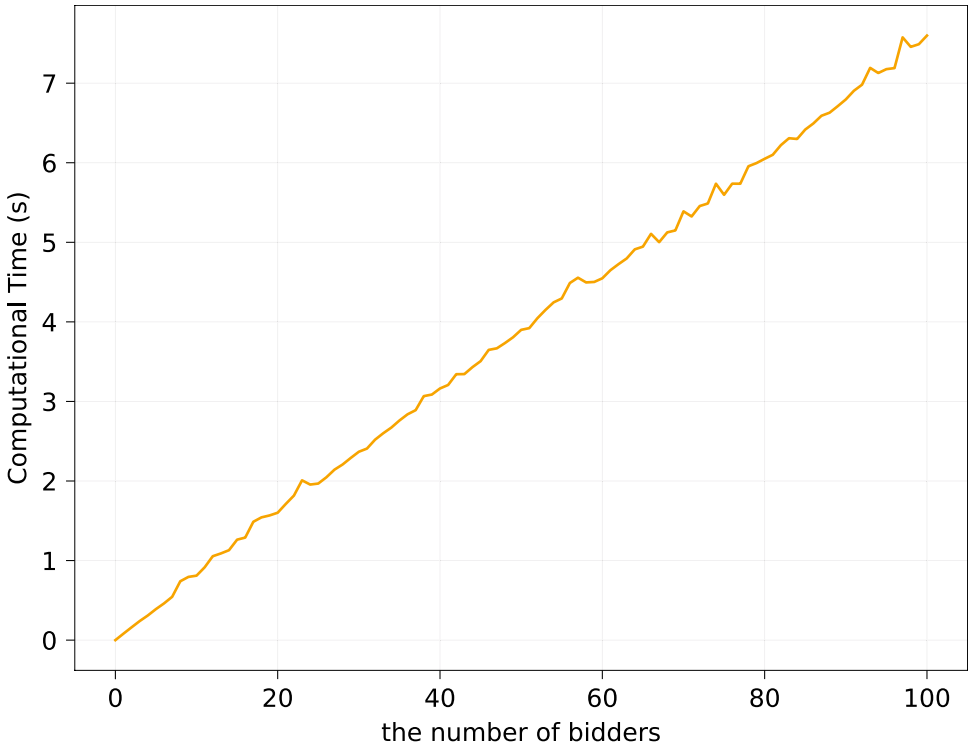


Fig. 10 The on-chain computational cost of the function FINALIZE

on-chain parts in "Renewal", "Receive", "Commit" and "Reveal" is similar to Fig. 8. Furthermore, "Finalize" produces a slightly higher than "Create", "Update" and "Transfer", since it needs to compute $\mathsf{EncFund}_{CA}$ in the contract account. To better evaluate the performance of the function FINALIZE, we generate different numbers of bidders in an auction to evaluate the corresponding on-chain cost. We can see from Fig. 10 that with the increase of the number of bidders, the computational cost generally shows an upward trend. When the number increases to 100 bidders, it requires around 7.59 seconds to complete the function "Finalize".

6 Related work

As mentioned before, our proposal is close to blockchain-based DNS and blockchain-based auction, hence we give related works in two parts respectively.

6.1 Blockchain-based DNS

To solve the traditional DNS' centralization problem, Namecoin [28], as the first project utilizing blockchain technology to solve the centralization problem of DNS, uses an additional preregistration method to avoid cybersquatting. Based on Namecoin, Ali et al. [29] proposed Blockstack Naming Service (BNS), which solved the security and performance issues for Namecoin. Unfortunately, both of them can't avoid damage caused by malicious cybersquatting to the current domain name owners. Compared with Namecoin and BNS, a recent study presented by Wang et al. [6] named ConsortiumDNS also supports the same registration method to prevent cybersquatting but cannot defend the single point of failure attack [36]. In addition, it mainly focuses on the optimization of storage performance for domain names' data, rather than the management of domain names' life cycle. Similar to the previous three solutions, TD-Root [12] mainly focuses on domain name registration and update but ignores domain name renewal and transfer. The DNSonChain scheme recently proposed by Jin et al. [30] only allows owners to claim their domain name ownership on the blockchain. As we know, the domain name auction is one of the typical ways to obtain the domain name [37]. However, none of the above blockchain-based systems provide such functionality. In [7], Ethereum officials proposed a domain name system based on the smart contract to resolve domain names into Ethereum addresses but not the digital IP address. The Namebase team recently proposed a blockchain-based DNS, named Handshake [8], with a domain name auction protocol. Their auction protocol follows the Vickrey auction, where the submitted bids are not revealed until all the bids are submitted, and the bidder with the highest bid wins the auction while paying with the second-highest bid. However, in their auction protocol, anyone can deduce the possible highest bid of the submitted bids, which may bring advantages over the latter bidders. In other words, the auction protocol cannot hold

the bidding-fairness property. To the best of our knowledge, it is still challenging to design a fair, secure, and privacy-preserving blockchain-based sealed-bid domain name auction protocol.

6.2 Blockchain-based auction

Using the secure multi-party computing (MPC) technique, Kosba et al. [31] proposed the first blockchain-based auction protocol named Hawk. However, the underlying secure MPC scheme is of a high level of interactivities, which leads to an inefficient solution, especially in the blockchain scenario. With the help of TEE, Yuan et al. [24] proposed an efficient blockchain-based auction protocol without sacrificing the smart contract's confidentiality. Nevertheless, the TEE technique usually demands users to update their hardware, which not everyone can afford. What's worse, the security of implementation of TEE is still controversial [26, 27]. Blass and Kerschbaum [32] proposed a new blockchain-based auction protocol still based on the secure MPC technique with a good performance. Nonetheless, the proposed protocol needs a pre-fixed bidding order and a semi-trusted auctioneer which is usually unnecessary in the blockchain scenario. Galal and Youssef [33] proposed an auction protocol over the Ethereum blockchain. However, it is not as efficient as expected due to the use of interactive zero-knowledge proof schemes. Furthermore, there is no mechanism to check whether the bidder has enough money to bid in the proposed auction protocol. Note that, although there is a deposit in their protocol, this deposit cannot be used as proof that the bidder has enough money to fulfill the bid. Recently, Nguyen and Thai [25] proposed a new efficient auction protocol based on multi-party state channels in terms of storage cost. However, the resulting protocol cannot support the sealed-bid auction. Qusa et al. [34] also proposed a new blockchain-based auction protocol, while the bidder colluding with some bidders can obtain some advantages over the rest of the bidders. Furthermore, Zhang et al. [35] proposed a time-released blockchain-based sealed-bid e-auction scheme, but the auctioneer has to initiate the auction and decide the time to terminate the auction, which cannot avoid the risk of ethical problems for the auctioneer. It is fair to say that a fair, secure, and privacy-preserving blockchain-based auction protocol is still desired. It is fair to say that a fair, secure, and privacy-preserving blockchain-based auction protocol is still desired.

7 Conclusion

To the best of our knowledge, the blockchain-based DNS is considered one of the most promising decentralized DNSs. However, the current blockchain-based DNSs fail to provide a management system of the entire life cycle of a domain name, and most of them are not secure or fair. To fill this gap, we in this paper have proposed a domain name management system based on an account-based consortium blockchain. Based on the security assumptions on the underlying blockchain system, the Pedersen commitment, and zero-knowledge proof, we have also shown that the proposed scheme is indeed privacy-preserving, secure, and fair. By running the extensive experiments, our proposal is indicated as (relatively) efficient in terms of computational cost.

Author contributions Miss. Genhua Lu participated in all processes of the paper, including participating in the original design and writing of the idea of the domain name management system based on an account-based consortium blockchain and participating in writing each part of the paper. Mr. Xiaofeng Jia is the leading performer in the experimental part of the paper. Furthermore, he also organized and analyzed the experimental data. Miss. Yi Zhang designed the Vickrey auction part. Prof. Jun Shao raised the problems of the existing domain name system. Prof. Guiyi Wei proposed the account-based consortium blockchain. In the final stage, all authors discussed and revised the content of each part of this paper.

Funding This work was supported by the National Key Research and Development Program of China [2019YFB1804500] and the National Natural Science Foundation of China [62272413].

Data availability The source code of the proposal can be found at https://github.com/EmmaLu-ux/paperDemo.

Declarations

Ethics approval This work does not involve any work related to ethics.

Consent to publish All authors consent to publication.

Conflicts of interest All authors declare that they have no potential competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Mockapetris PV. Domain names-concepts and facilities. https://www.ietf.org/rfc/rfc1034.txt. [Online; Accessed 21 Dec 2022]
2. Mockapetris PV. Domain names-implementation and specification. https://www.rfc-editor.org/rfc/inline-errata/rfc1035.html. [Online; Accessed 21 Dec 2022]
3. Khormali A et al (2021) Domain name system security and srivacy: A contemporary survey. Comput Netw 185:107699
4. Florian M. Dns single point of failure detection using transitive availability dependency analysis. https://www.sstic.org/media/SSTIC2018/SSTIC-actes/transdep/SSTIC2018-Article-transdep-maury.pdf. [Online; Accessed 21 Dec 2022]
5. Wikipedia. Ddos attacks on dyn. https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn#References. [Online; accessed 11 Feb 2023]
6. Wang X, Li K, Li H, Li Y, Liang Z (2017) Balaji, P. & Siegel, H.J. (eds) ConsortiumDNS: A distributed domain name service based on consortium chain. (eds Balaji, P. & Siegel, H.J.) HPCC/SmartCity/DSS pp 617–620

7. Johnson N, Griffith V (2017) Ethereum name service. https://ensuser.com/docs/readme.html. [Online; Accessed 21 Dec 2022]
8. Roquerre T. Handshake project paper. https://ensuser.com/docs/readme.html. [Online; Accessed 21 Dec 2022]
9. Benshoof B, Rosen A, Bourgeois AG, Harrison RW (2016) Abramson, D. & Acar, U. (eds) Distributed decentralized domain name service. (eds Abramson, D. & Acar, U.) IPDPS, pp 1279–1287
10. Shen Y et al (2021) Dns service model based on permissioned blockchain. Intell Autom Soft Comput 27:259–268
11. Liu W et al (2020) Montavont, N. & Douligeris, C. (eds) A secure domain name resolution and management architecture based on blockchain. (eds Montavont, N. & Douligeris, C.) ISCC, pp 1–7
12. He G, Su W, Gao S, Yue J (2020) TD-Root: A trustworthy decentralized DNS root management architecture based on permissioned blockchain. Futur Gener Comput Syst 102:912–924
13. Lu G, Zhang Y, Lu Z, Shao J, Wei G (2021) Shen, J. & Liu, J.K. (eds) Blockchain-based sealed-bid domain name auction protocol. (eds Shen, J. & Liu, J.K.) EAI AC3, pp 25–38
14. Guan Y, Zheng H, Shao J, Lu R, Wei G (2022) Fair outsourcing polynomial computation based on the blockchain. IEEE Trans Serv Comput 15:2795–2808
15. Zheng H, Shao J, Wei G (2020) Attribute-based encryption with outsourced decryption in blockchain. Peer-to-Peer Network Application 13:1643–1655
16. Lin C, He D, Huang X, Khan MK, Choo KR (2020) DCAP: A secure and efficient decentralized conditional anonymous payment system based on blockchain. IEEE Trans Inf Forensics Secur 15:2440–2452
17. Rosato A, Tymula A (2019) Loss aversion and competition in vickrey auctions: Money ain't no good. Games Econom Behav 115:188–208
18. Noether S (2015) Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive 2015:1098
19. Rivest RL, Shamir A, Tauman Y (2001) Boyd, C. (ed.) How to leak a secret. (ed.Boyd, C.) ASIACRYPT 2248:552–565
20. Castro M, Liskov B, Seltzer MI, Leach PJ (eds) (1999) *Practical byzantine fault tolerance*. (eds Seltzer, M.I. & Leach, P.J.) *OSDI*, pp 173–186
21. Xu G et al (2022) SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent internet of vehicles. J Parallel Distrib Comput 164:1–11
22. Li W et al (2021) A scalable multi-layer PBFT consensus for blockchain. IEEE Trans Parallel Distrib Syst 32:1146–1160
23. Lao L, Dai X, Xiao B, Guo S (2020) Chaudhary, V., Datta, S. & Ergun, F. (eds) G-PBFT: A location-based and scalable consensus protocol for iot-blockchain applications. (eds Chaudhary, V., Datta, S. & Ergun, F.) IPDPS, pp 664–673
24. Yuan R, Xia Y, Chen H, Zang B, Xie J (2018) Shadoweth: Private smart contract on public blockchain. J Comput Sci Technol 33:542–556
25. Nguyen TDT, Thai MT (2021) A blockchain-based iterative double auction protocol using multiparty state channels. ACM Trans Internet Technol 21:39:1–39:22
26. Shaun Davenport RF (2014) Sgx: The good, the bad and the downright ugly. https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly. [Online; Accessed 21 Dec 2022]
27. Luis Merino JA (2016) Sgx secure enclaves in practice security and crypto review. https://www.blackhat.com/docs/us-16/materials/us-16-Aumasson-SGX-Secure-Enclaves-In-Practice-Security-And-Crypto-Review.pdf. [Online; Accessed 21 Dec 2022]
28. Durham V (2014) Namecoin. https://www.namecoin.org. [Online; Accessed 21 Dec 2022]
29. Ali M, Nelson JC, Shea R, Freedman MJ (2016) Gulati, A. & Weatherspoon, H. (eds) Blockstack: A global naming and storage system secured by blockchains. (eds Gulati, A. & Weatherspoon, H.) USENIX pp 181–194
30. Jin L, Hao S, Huang Y, Wang H, Cotton C (2021) Hu, Y.C. & Yang, X. (eds) DNSonChain: Delegating privacy-preserved DNS resolution to blockchain. (eds Hu, Y.C. & Yang, X.) ICNP pp 1–11
31. Kosba AE, Miller A, Shi E, Wen Z, Papamanthou C (2016) Úlfar Erlingsson & Shmatikov, V. (eds) Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. (eds Úlfar Erlingsson & Shmatikov, V.) SP pp 839–858
32. Blass E, Kerschbaum F (2018) López, J., Zhou, J. & Soriano, M. (eds) Strain: A secure auction for blockchains. (eds López, J., Zhou, J. & Soriano, M.) ESORICS 11098:87–110
33. Galal HS, Youssef AM (2018) Zohar, A. et al. (eds) Verifiable sealed-bid auction on the ethereum blockchain. (eds Zohar, A. etal.) FCDS 10958;265–278
34. Qusa H, Tarazi J, Akre V (2020) Burmawi (ed.) Secure e-auction system using blockchain: UAE case study. (ed.Burmawi) ASET pp 1–5
35. Zhang M, Yang M, Shen G (2022) SSBAS-FA: A secure sealed-bid e-auction scheme with fair arbitration based on time-released blockchain. J Syst Archit 129:102619
36. Yin S, Teng Y, Hu N, Jia XD (2020) Tian, Z., Yin, L. & Gu, Z. (eds) Decentralization of DNS: old problems and new challenges. (eds Tian, Z., Yin, L. & Gu, Z.) CIAT pp 335–341
37. Wikipedia (2020) Domain name auction. https://en.wikipedia.org/w/index.php?title=Domain_name_auction&oldid=975293048. [Online; Accessed 21 Dec 2022]

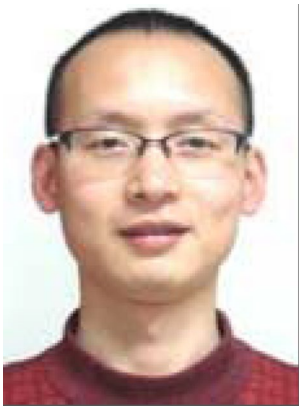
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Genhua Lu is a master student of the School of Computer Science and Technology at Zhejiang Gongshang University. Her research interests include applied cryptography and blockchain.

Xiaofeng Jia is a master student of the School of Computer Science and Technology at Zhejiang Gongshang University. His research interests include applied cryptography and blockchain.

Yi Zhang is a master student of the School of Computer Science and Technology at Zhejiang Gongshang University. Her research interests include internet applied cryptography and blockchain.



Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008. He was a Post-Doctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.

Guiyi Wei is a professor of the School of Information and Electronic Engineering at Zhejiang Gongshang University. He obtained his Ph.D. in Dec 2006 from Zhejiang University, where he was advised by Cheung Kong chair professor Yao Zheng. His research interests include wireless networks, mobile computing, cloud computing, social networks and network security.