Theme Article: AI and FinTech

Blockchain-Based Fair and Secure Electronic Double Auction Protocol

Lietong Liu, Mingxiao Du, and Xiaofeng Ma
Tongji University

***Abstract*—Double auction is an auction in which multiple buyers and sellers looking for a price where supply and demand balance. Since the electronic double auction based on secure multiparty computation (DABSMPC) cannot guarantee its fairness, we propose a blockchain-based fair and secure electronic double auction protocol (BFSDA). BFSDA modifies the data input and distribution mechanism of participants in DABSMPC, which improves the security of the protocol. Then, the BFSDA introduces and improves the blockchain-based fair and secure multiparty computation protocol (BFSMPC) to ensure fairness while increasing the success rate of secret recovery. In addition, BFSDA uses a fairer and more efficient protocol for secure two-party comparing to obtain the final marketing clearing price. The schema analysis result of BFSDA shows that: first, the private input data will not be revealed as long as data owner is not compromised, second, honest participants can get the result or economic compensation, and third, participants only need to pay the deposit once and a large amount of complicated verification operations are carried out off the chain, which ensures the efficiency of the protocol.**

The double auction is a variant of the first price sealed auction. There are multiple buyers and sellers. For the commodity to be traded, the buyer gives the quantity he wants to buy at each possible price and the seller gives the quantity he wants to sell at each price. The auctioneer collects all price data and computes the total demand and the total supply at each price in the market. There must be a price that makes the total supply amount at this price equals to the total demand amount. The price is the marketing clearing price (MCP) of each unit of the commodity to be traded.

Compared with other forms of electronic auctions, electronic sealed auctions have a better

Digital Object Identifier 10.1109/MIS.2020.2977896
Date of publication 3 March 2020; date of current version 6 July 2020.

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

performance in efficiency and privacy protection. Therefore, the electronic sealed auction scheme is an electronic auction protocol with relatively extensive research. Suzuki *et al.*[1] put forward a distributed auction system based on hash chain. However, these auction protocols rely on the existence of trusted third parties, which is difficult to put into practice. Brandt[2] proposed an electronic auction protocol based on homologous encryption mechanism without trusted third parties. However, the communication and computational costs of these protocols are very large and cannot be applied to large-scale utility systems.

Bogetoft *et al.*[3] introduced the idea of secure multiparty computation (SMPC) based on secret sharing, security comparison and other technologies. He proposed a set of electronic double auction schemes without relying on trusted third parties (DABSMPC). The set of electronic double auction schemes have high efficiency and can be applied in practice.

However, there are some drawbacks concerning DABSMPC: first, since DABSMPC uses noninteractive verifiable secret sharing (VSS) due to high efficiency, if any two servers are compromised, the opponent will obtain all the customers' private data, second, the protocol for secure comparing is a fundamental and time-consuming subprotocol in DABSMPC. In DABSMPC, the protocol for secure comparing has logarithmic efficiency. At present, the most efficient protocol for secure comparing has constant-round efficiency, and third, currently, the design of the SMPC protocol focuses more on security and correctness. It is difficult to guarantee protocol fairness in operation in reality because the protocol cannot prevent malicious participants from terminating the protocol in advance.[4]

This article modifies the data input and distribution mechanism of participants to improve the security of the protocol; it chooses fairer and more efficient protocol fair and efficient protocol for secure two-party comparing, (FEPSTC)[5] to make binary comparison on price chain to obtain MCP; it also introduces BFSMPC[6] and uses smart contracts to automatically judge participants' behaviors during secret reconstruction. The malicious nodes will have economic penalties so that all rational nodes tend to remain honest and implement the protocol fairly. For the multiple round characteristics of the electronic double auction protocol, this article improves the fair secret reconstruction protocol of BFSMPC to increase the success rate of secret recovery and ensure the efficiency of the protocol.

The rest of this article is organized as follows. An overview of DABSMPC and related parts of this scheme are stated in the "Background" section. We also describe the main basic principles and general steps of FEPSTC and BFSMPC. In the "Blockchain-Based Fair and Secure Electronic Double Auction Protocol" section, this article presents a blockchain-based fair and secure electronic double auction protocol (BFSDA), and describes the improved fair secret reconstruction, which is the subprotocol of BFSDA. The security, fairness, and performance are analyzed in the "Scheme Analysis" section. Finally, the "Conclusion" section concludes this article.

BACKGROUND

Electronic Double Auction Based on Secure Multiparty Computation

In the double auction scenario described in literature,[3] the trader is the farmer who produces agricultural products and the auctioneer is the company that acquires the products. The commodity to be traded is the production contract signed by the farmer and the company. The trade is carried out among all farmers. Because farmers need to redistribute the production contracts to maximize the production efficiency.

There are two reasons for applying SMPC. First, the quotation of counterparties undoubtedly reveals some private information. Farmers do not want the acquiring company to be the auctioneer because the acquiring company may take advantage of the information when they resign the contract. On the other hand, because the contract is managed by the company, thus the company will also ask for participation in the double auction to guarantee the fairness of the auction. As there is no full trust among farmers or between farmers and the acquiring company, it is reasonable to realize the electronic double auction through the mechanism based on SMPC without trusted third parties.

In DABSMPC, the role of the auctioneer is realized by the computation of multiple parties,

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

including the manufacturer association, the acquiring company and the project implementer. Each trader secretly distributes the price data to the three servers through the secret sharing scheme and obtains the current marketing clearing price through the secure computation between the three parties.

Traders use the noninteractive VSS method based on the public-private key pair and the pseudo-random number function to conduct the Shamir secret sharing of the demand and supply input at each possible price between servers.

The scheme performs a binary comparison on the price chain. For each comparison on the price chain, the server uses its own share to run the local comparison protocol and then reconstructs the protocol results to obtain the comparison result.

FEPSTC Under Semihonest Model

Literature[5] designed and modified the ElGamal algorithm (MEA) based on ElGamal's encryption algorithm. It also designed the multiplication and subtraction homomorphic encryption algorithms based on the MEA algorithm.

The range of the number compared by the protocol is real number $[-n, n]$, the computational complexity and the traffic are both $O(1)$. It is a fair protocol and can distinguish " $=$."

Blockchain-Based Fair and Secure Multiparty Computation Protocol

The fairness of the secure multiparty computation protocol means that all honest nodes should also get the output when the malicious node gets the output. However, Cleve has proved that fair multiparty computations are not available in majority dishonest situations.[4]

The most common SMPC security models are the semihonest model and the malicious opponent model. These two models assume that the behaviors of the participants follow the initial settings from beginning to end. The semihonest parties only infer the information of other participants, whereas the malicious parties will continue to destruct the execution of the protocol. However, in reality, participants are always rational, and they will take actions to maximize their own interests. Under the appropriate incentive mechanism, all people (or most people) will not destroy the protocol.

With the advent of the encrypted digital currency, Andrychowicz[7] and Kumaresan[8] began to study the economic penalty mechanism in the SMPC protocol to solve its fairness problem. They used Bitcoin to construct a time-limited commitment. Participants must announce their secrets within a limited time, or they will receive economic penalties. However, since the Bitcoin scripting language is not Turing-complete and cannot perform complex functions, the amount of deposit to be paid by the participants reached $O(N^2)$. Literature[9] designed the smart contract based on Ethereum and used the threshold secret sharing scheme to add fraudsters to the blacklist according to the secret reconstruction situation. The amount of deposit is reduced to $O(N)$. But each round of the SMPC protocol requires the participation of blockchain. Blockchain must verify the information announced by each participant and the computing cost is high.

The BFSMPC mainly consists of two phases. In the first phase, participants execute SMPC protocol based on the Gennaro scheme[10] off the chain. The secrets are shared in a t-order random polynomial expression. Afterward, participants will get their respective secret shares. The protocol combines the Gennaro VSS scheme with the Pedersen homomorphic commitment scheme[11] to achieve the verification of the correctness of the shares; in the second phase, participants jointly execute a fair secret reconstruction protocol on the blockchain. The smart contract determines whether participants are malicious or not based on the feedback timeout and feedback result. The honest party can recover the secret as long as it collects $t + 1$ correct shares. If the recovery fails, the deposit of the malicious party will be split equally among the honest parties.

BLOCKCHAIN-BASED FAIR AND SECURE ELECTRONIC DOUBLE AUCTION PROTOCOL

Contributes

- By introducing the BFSMPC, the multiparty computing process in BFSDA can be performed by all participants. Each trader is a node. In addition to the participants of the transaction as the compute nodes, the

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

manufacturer association and the acquiring company serve as the total demand comparison node ***DN*** and the total supply comparison node ***SN***, respectively, so as to achieve a secure comparison between the total demand and the total supply. BFSMPC uses the Ethereum smart contract to construct an economic penalty mechanism. During the execution of the secret reconstruction, the smart contract determines the malicious party. Finally, the malicious party is punished, and the honest party is compensated, thus the fairness of the protocol is guaranteed.

- BFSDA improves the price data input and distribution mechanisms in the double auction protocol. When participants input data, they blindly process their own data first, and then distribute the secret share of the blinded data to each compute node. The compute node completes the share computation of the total supply and the total demand. Then, they will recover the blinded total supply and total demand data along with the comparison nodes. The two comparison nodes obtain their blind random numbers from their respective participants and then recover the offset total supply and the offset total demand data.
- BFSDA introduces FEPSTC under the semi-honest model with the constant-round efficiency. For the two comparison nodes, they use FEPSTC to compare the offset supply with offset demand.
- BFSDA reduces round of VSS from $2N$ to at most $2\log_2 N$ by performing a binary comparison on the price chain, where N is amount of possible price.

Implementation Steps

Establishment of System n participants $I_1,\ldots,I_n$ are the blockchain nodes. The company and the manufacturer association are also nodes, which serve as the total demand node ***DN*** and the total supply node ***SN***, respectively. Participants deploy the agreed smart contracts on the blockchain and pay the deposit. Set p as a big prime number negotiated by n participants and it satisfies $p=2q+1$, q is also a prime number,g is the q-order element of $\mathbb{Z}_p^*$, h is a random element in the subgroup generated by g. $P_{\min}$ and $P_{\max}$ are the lower and upper limits of P,which is price of the product. Suppose the range of the current price P_{cur} is $[P_{\text{low}},P_{\text{high}}]$, the initial value of P_{low} is $P_{\min}$, and the initial value of P_{high} is $P_{\max}$.

Input Phase Current price $P_{\text{cur}}=[(P_{\text{low}}+P_{\text{high}})/2]$, and quantity of commodities that I_i is willing to purchase and sell at current price are x_i and y_i respectively. I_i first carries out the blind process to the input data by selecting $r_i\in_R\mathbb{Z}_p$, $r_i'\in_R\mathbb{Z}_p$, $t_i\in_R\mathbb{Z}_p$, $t_i'\in_R\mathbb{Z}_p$, which satisfy

$$r_i+t_i=r'_i+t'_i=\text{const} \tag{1}$$

where const is a constant determined by I_i privately. Let $x_i^*=x_i+t_i'$, $y_i^*=y_i+r_i'$. Then, I_i shares the blinded data x_i^* and y_i^* with other participants through the Shamir secret sharing scheme. I_i uses the Pedersen homomorphic commitment scheme to guarantee the verifiability of the secret share. Sharing x_i^* and sharing y_i^* have the same process and can be done simultaneously. For the sake of simplicity, only the process of sharing x_i^* is described here.

I_i selects two random t-order polynomial expressions, the secret value polynomial expression $a_i(x)=\sum_{l=0}^{t}\alpha_l x^l$ and the auxiliary polynomial expression $b_i(x)=\sum_{l=0}^{t}\beta_l x^l$, where α_0 is the secret value x_i^*. I_i distributes the share secret $s_{i,j}=(a_i(j),b_i(j))$ to other participants $I_j, j\neq i$. At the same time, I_i broadcasts the commitment value:

$$A_{i,k}=g^{a_i(k)}h^{b_i(k)}\text{modp}, k=0,1,\ldots n\,. \tag{2}$$

After I_j receives his share $s_{i,j}$, he need to verify whether the share is correct or not, that is, whether his share is on the polynomial expression with the same times t as others (called VSPS property). The methods are as follows:

Because$f(x)=\sum_{l=0}^{t}\alpha_l x^l$, there is

$$\begin{bmatrix}1 & 0 & \cdots & 0\\ 1 & 1 & \cdots & 1\\ \vdots & \vdots & \vdots & \vdots\\ 1 & t & \cdots & t^t\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_t\end{bmatrix}=\begin{bmatrix}f(0)\\ f(1)\\ \vdots\\ f(t)\end{bmatrix} \tag{3}$$

Record as

$$V\begin{bmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_t\end{bmatrix}=\begin{bmatrix}f(0)\\ f(1)\\ \vdots\\ f(t)\end{bmatrix} \tag{4}$$

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

So

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_t \end{bmatrix} = V^{-1} \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(t) \end{bmatrix} . \quad (5)$$

The participants select $\delta \in [1, n]$ randomly. A_δ is an announced commitment. Compute

$$A'_\delta = g^{f(\delta)} h^{r(\delta)} = g^{\sum_{l=0}^{t} \alpha_l \delta^l} h^{\sum_{l=0}^{t} \beta_l \delta^l} . \quad (6)$$

Record λ_{ji} as the value of V^{-1} in jth line and ith row. There is $\alpha_j = \sum_{i=0}^{t} \lambda_{ji} f(i)$. So

$$A'_\delta = g^{\sum_{j=0}^{t} \sum_{i=0}^{t} \lambda_{ji} f(i) \delta^j} h^{\sum_{j=0}^{t} \sum_{i=0}^{t} \lambda_{ji} r(i) \delta^j} = \prod_{i=0}^{t} \left(g^{f(i)} h^{r(i)} \right)^{\Delta_i} = \prod_{i=0}^{t} (A_i)^{\Delta_i} \quad (7)$$

where $\Delta_i = \sum_{j=0}^{t} \lambda_{ji} \delta^j$.

I_j first compares the computed A'_δ with A_δ announced previously. If they are equal, then he needs to check whether his share match the corresponding commitment. If they match, the received share is correct. If they do not match, then he can reapply to the distributor for share. Through this verification process, each participant can monitor the secret distributor's correct distribution of shares.

Computation Phase Assuming that I_j has a and b's share a_j and b_j, and the corresponding auxiliary value share ρ_j and σ_j. It needs to compute $c = a + b$, I_j will computes $c_j = a_j + b_j$ and sets the result as own secret share obtained. Meanwhile, I_j announces the commitment to the share $C'_j = g^{c_j} h^{\rho_j + \sigma_j} \bmod p$. Because the double auction protocol only involves additive operations, there is no need to announce the commitment after each addition.

I_j computes the secret share of the blinded total demand D^* to the certain price, which is

$$D_j^* = \sum_{i=1}^{n} x_{i,j}^* = \sum_{i=1}^{n} a_i(j) . \quad (8)$$

The commitment value of D_j^*

$$C'_j = g^{D_j^*} h^{\sum_{i=1}^{n} b_i(j)} \bmod p \cdot \quad (9)$$

Broadcast C'_j.Other participants can compute $C_j = \prod_{i=1}^{n} A_{i,j}$. If it is equal to C'_j, it indicates that the commitment is correct.

Output Phase After the computation phase is completed, $I_i (i = 1, \ldots n)$ will hold the share D_i^* and S_i^*. The public information obtained is the commitment value $C'_j, j = 1, 2, \ldots, n$ to the secret share D_j^*, and commitment value C'_0 to D^*.

As mentioned before, sharing x_i^* and sharing y_i^* have the same process and can be done simultaneously. After performing same process, I_i can also hold the share S_i^* and commitments to all shares S_j^* and S^*.

During the output phase, the fair secret reconstruction protocol is realized through blockchain, which is described in section III.C. All participants announce the secret share and the share commitment auxiliary value. Secret can be recovered correctly as long as there are $t + 1$ correct shares. All participants worked with ***DN*** and ***SN*** to reconstruct D^* and S^*, respectively.

Recovery of Blind Data After the output phase, ***DN*** has blinded total demand D^* of current price P_{cur}. ***SN*** has blinded total supply S^*.

I_i will send t'_i and r'_i to ***DN*** and ***SN***, respectively. They will compute the offset total demand D' and the offset total supply S' of current price

$$D' = D^* + \sum_{1}^{n} t'_i = D + \sum_{1}^{n} \text{const}_i \quad (10)$$

$$S' = S^* + \sum_{1}^{n} r'_i = S + \sum_{1}^{n} \text{const}_i . \quad (11)$$

Comparison Phase Two comparison nodes will be performing the FEPSTC.

If $D' = S'$, it means the double auction protocol runs successfully, the P_{cur} is MCP and the deposit will be returned; if $D' > S'$, $P_{\text{low}} := P_{\text{cur}}$, back to Input Phase, the protocol continues; if $D' < S'$, $P_{\text{high}} := P_{\text{cur}}$, back to Input Phase, the protocol continues.

Fair Secret Reconstruction Protocol

Fair secret reconstruction protocol in BFSMPC includes the local protocol LocPro(***LP***) and the smart contract ConContract (***CC***). ***LP*** is executed by the participant locally, its complete description is presented in Figure 1. ***CC*** is executed by the blockchain nodes and responsible for the deposit storage and judges the behaviors of the participants during the secret reconstruction phase.

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

```
Protocol LocPro
The protocol owner is $I_i$. The amount of deposit is $\$dep$, verification array $J_i$ is all 0.

Deposit: $I_i$ send (Deposit, $\$dep$) to CD.
    - Once $T \geq over(1)$, then send (GetD) to CD.
After $D$ is received .
    -If $D \neq 1$, then refund the deposit and terminated the
        protocol.
Compute: Once $T \geq over(2)$, then execute the data input and computation phases. $I_i$ can obtain the output $OUT_i = (y_i, \varepsilon_i, C_0, C_1 \dots, C_n)$, where $y_i$ is the secret share, $\varepsilon_i$ is the commitment auxiliary value of the secret share, $C_{j\in[n]} := g^{y_j} h^{\varepsilon_j} \bmod p$.
Commit: Once $T \geq over(3)$, send (Commit, $I_i$, $C_i$) to CC.
Open: Once $T \geq over(4)$, send (Verify, $I_i$ , $y_i, \varepsilon_i$) to other participants $I_{j(j\neq i)}$ and SN/DN.
Verify: Once the received $I_{j(j\neq i)}$'s (Verify, $y_i, \varepsilon_i$), verify $T < over(5)$.
    -If $g^{y_j} h^{\varepsilon_j} \bmod p = C_j$, then $J_i[j] := 1$.
After all the shares are verified, take $t + 1$ correct shares randomly to recover the secret.
    -If it succeeds, then $J_i[0] := 1$, send (Verify, $I_i$, $J_i$) to
        CC.
Check: Once $T \geq over(5)$, send (Check, $I_i$) to CC.
After $U$ is obtained.
    -If $J[0] = 1$, once $T \geq over(6)$, send (GetR) to CC.
        -If $R$ = False, send (Pay, $I_i$) to CC, the protocol is
        terminated.
        -Otherwise, back to Compute, the protocol
        continues.
    -If $J[0] \neq 1$ and $J[i] \neq 1$ , then send (ReVerify,
        $I_i, y_i, \varepsilon_i$) to CC.
ReCheck: Once $T \geq over(6)$, send (ReCheck, $I_i$) to CC.
After ($U$, $Sh$) is obtained.
    -If $J[0] = 1$, secret is successfully recovered.
        -If $I_i$ cannot recover the secret, then use $Sh$ to recover.
            Once $T \geq over(7)$, send (GetR) to CC.
            -If $R$ = False ,then send (Pay, $I_i$ ) to CC, the
                protocol is terminated
            -Else, back to Compute, the protocol continues.
    -If $J[0] \neq 1$, once $T \geq over(7)$, send (RePay, $I_i$) to CC
        , the protocol is terminated.
```

Figure 1. LocPro protocol.

```
Contract ConDeposit
The set of protocol participants is $I := \{I_1, \dots, I_n\}$, the deposit flag bit is $D := 0$, the amount of deposit is $\$dep$, the deposit record array $M$ is 0.

Deposit: Once the received (Deposit, $\$dep$, $I_i$ ) and $T < over(1)$,
    -If $Ledger[I_i] \geq \$dep$ then
    $Ledger[I_i] := Ledger[I_i] - \$dep$;
    $Ledger[CD] := Ledger[CD] + \$dep$;
    $M[i] := 1, D := M[1] \& \dots M[n]$;
    -Else, terminated the protocol.
GetD: Once the received (GetD, $I_i$) and $T < over(2)$,
    -If $D \neq 1$, then refund the deposit.
    -Else, return $D$.
Pay: Once the received (CC, Pay, $I_i$, $f$), then
$Ledger[I_i]$
$:= Ledger[I_i] + \$dep + (f \times \$dep)/(n - f)$
RePay: Once the received (CC, RePay, $I_i$, $f$), then
$Ledger[I_i] := Ledger[I_i] + \$dep + (f \times \$dep)/(n - f)$.
```

Figure 2. ConDeposit contract.

During each output phase, $I_i (i = 1, \dots n)$ will reconstruct D^* with ***DN*** and S^* with ***SN***. Since BFSDA needs to reconstruct at most $2\log_2[P_{\max} - P_{\min}]$ secret data, it needs to perform secret reconstruction multiple times. To reduce the number of deposit payment, this article separates ***CC***'s deposit payment and judgment of participants' behaviors, dividing ***CC*** into two smart contracts, including ConDeposit(***CD***) and ConContract. Their complete descriptions are presented in Figures 2 and 3, respectively. The running of protocol was controlled by ***DN***/***SN***, even if it needs to perform multiple rounds of secret reconstruction, the deposit only needs to be paid once. In addition, in the BFSMPC, only when all nodes confirm the secret recovery can the protocol be successfully executed. If there is a malicious node concealing the fact that the secret is recovered successfully, the protocol will be terminated and loses its fairness. BFSDA adds a recheck procedure. After all the secret shares and commitments are collected, the smart contract will judge whether there are enough correct shares. If there are enough correct shares, shares will be returned and the protocol continues. Even if there is a malicious node lying, the protocol can also run normally.

The sequence diagram of the fair secret reconstruction protocol is shown in Figure 4.

A fair secret reconstruction protocol is divided into the following phases, including the preparation phase, the commitment and verification phase, and the check and recheck phase.

Preparation Phase In the beginning, each participant pays a deposit to ***CD***. Record $\mathrm{over}(r)$ as the end of the *r*th round. The participant pays the deposit before $\mathrm{over}(1)$. Once $T > \mathrm{over}(1)$, if someone does not pay the deposit, the deposit is refunded and the protocol is terminated, otherwise the protocol continues. The deposit only needs to be paid first time.

Commitment and Verification Phase Each participant performs the data input and computation

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

Contract *ConContract*

The set of protocol participants is $I := \{I_1, \ldots, I_n\}$, ***DN***/***SN***. The set of malicious participants is $F := \emptyset$. The number of malicious participants f, the initial value is 0. The array *Ch* and *Sh* are used to compute the commitments and shares released by the participants, tthe running flag R is True.

Commit: Once the received $(I_i$, **Commit**, $C_i)$,

- -If $T < over(4)$, then record I_i trigger action, $Ch[i] := C_i$.
- -Else, add the participants who did not trigger Commit into F.

Verify: Once $(I_i$, **Verify,** $J_i)$ is received,

- -If $T < over(5)$, then record I_i trigger action, $J = J\&J_i$.
- -Else, add the participants who did not trigger Verify into F.

Check: Once $(I_i$, **Check**) is received, $U := (J, F)$, return (U).

- -If $J[0] = 1$, secret is successfully recovered, ***SN*** and ***DN*** will perform FEPSTC. If comparison result shows equality. ***SN***/***DN*** will set R False.

GetR: Once the received (**GetR**, I_i), return R.

ReVerify: Once $(I_i$, **ReVerify**, $y_i, \varepsilon_i)$ is received, check whether $g^{y_i}h^{\varepsilon_i} mod p = C_i$, if so, $Sh[i] := (y_i, \varepsilon_i)$, $J[i] := 1$.

ReCheck: Once $(I_i$, **ReCheck**) is received, then add the participants whose shares have not been verified into F.

- -If *Sh* has enough shares to recover secret data, $J[0] := 1$, secret is successfully recovered, ***SN*** and ***DN*** will perform FEPSTC. If comparison result shows equality. ***SN***/***DN*** will set R false.

$U := (J, F)$, return (U, Sh).

Pay: Once $(I_i$, **Pay**) is received,

- -If R = False and $I_i \in I - F$, then refund the deposit and compensation, send (**Pay**, I_i, f)to ***CD***.

RePay: Once $T \geq over(7)$, then $F := F \cup \{I_i | i \in [1, n] \&\& J[i] \neq 1\}$

- -If the received $(I_i$, **RePay**) , if $J[0] \neq 1$ and $I_i \in I - F$, then send (**RePay**, I_i, f) to ***CD***.

Figure 3. ConContract contract.

phase (**Compute**) of the blockchain. I_i submits the commit C_i to ***CC*** and ***CC*** will save C_i into $Ch[i]$. Once $T \geq \text{over}(4)$, ***CC*** will add the participant who did not submit the commitment into the malicious set *F*. Before $\text{over}(5)$, I_i will send the secret share y_i and the commitment auxiliary value ε_i to other participants and and ***SN***/***DN*** (**Open**) to verify. Participants will send the verification result to ***CC*** (**Verify**). ***CC*** will compare all the verification arrays received bit by bit and get the state array J. Once $T \geq \text{over}(5)$, participants who did not submit the verification array will be added to the malicious set *F*.

Check Phase I_i query (J, F) (**Check**) from ***CC***. If $J[0] = 1$, it indicates that all participants successfully have recovered the secrets. At this time, ***SN*** and ***DN*** will perform FEPSTC. If comparison result shows equality, it means current price P_{cur} is MCP, double auction runs successfully. ***SN***/***DN*** will set R False. Once $T \geq \text{over}(6)$, I_i will get R(**GetR**) from **CC**. If $R = \text{False}$, I_i will send a repayment request (**Pay**) to ***CC*** and ***CD*** returns the refund and compensation; otherwise, back to commitment and verification phase, the protocol continues.

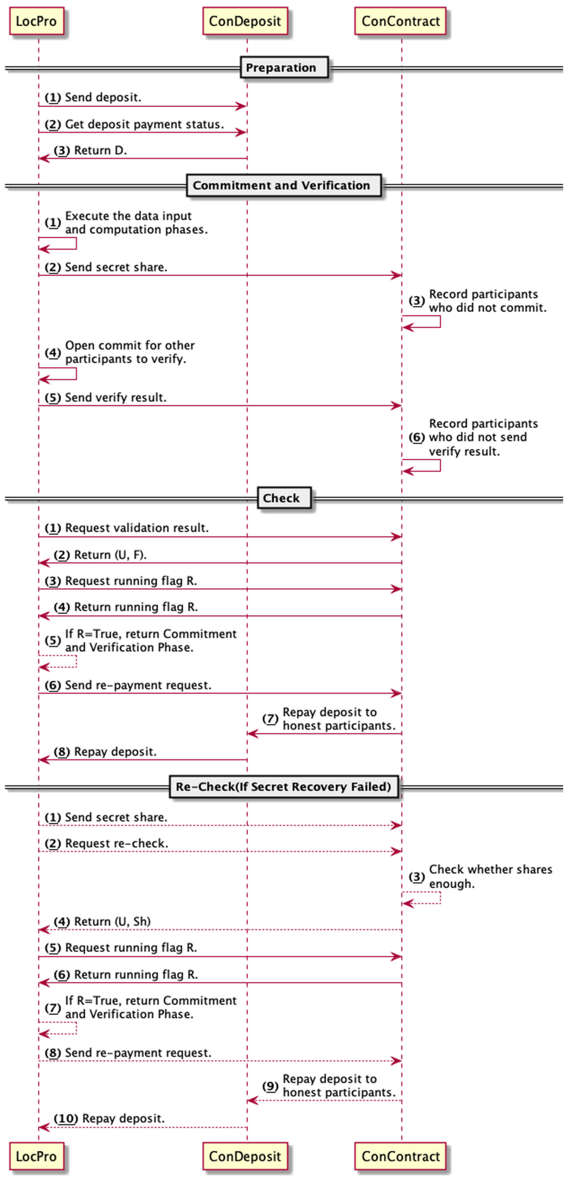
LocPro
ConDeposit
ConContract
Preparation
(1) Send deposit.
(2) Get deposit payment status.
(3) Return D.
Commitment and Verification
(1) Execute the data input and computation phases.
(2) Send secret share.
(3) Record participants who did not commit.
(4) Open commit for other participants to verify.
(5) Send verify result.
(6) Record participants who did not send verify result.
Check
(1) Request validation result.
(2) Return (U, F).
(3) Request running flag R.
(4) Return running flag R.
(5) If R=True, return Commitment and Verification Phase.
(6) Send re-payment request.
(7) Repay deposit to honest participants.
(8) Repay deposit.
Re-Check(If Secret Recovery Failed)
(1) Send secret share.
(2) Request re-check.
(3) Check whether shares enough.
(4) Return (U, Sh)
(5) Request running flag R.
(6) Return running flag R.
(7) If R=True, return Commitment and Verification Phase.
(8) Send re-payment request.
(9) Repay deposit to honest participants.
(10) Repay deposit.
LocPro
ConDeposit
ConContract

Figure 4. fair secret reconstruction protocol.

Recheck Phase If $J[0] \neq 1$, it indicates that some participant did not successfully recover

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

secret data. It is necessary for I_i to send a reverification request (**ReVerify**) to ***CC*** before $\text{over}(6)$ and submit its own share to ***CC***. ***CC*** extracts C_i from $Ch[i]$ for verification. If it passes, the share will be saved into $Sh[i]$, and $J[i] := 1$. When $T \geq \text{over}(6)$, I_i will send a recheck request (**ReCheck**) to ***CC***. ***CC*** checks whether the current share array is sufficient to recover the secret, if yes, then $J[0] := 1$, ***SN*** and ***DN*** will perform FEPSTC. ***CC*** returns (U, Sh) to I_i. If $J[0] = 1$ and I_i has not recovered the secret before, the secret can be recovered by *Sh*. Once $T \geq \text{over}(7)$, I_i will get R(**GetR**) from **CC**. If $R = \text{False}$, sends a repayment request (**Pay**) to ***CC*** and ***CD*** returns the refund and compensation; otherwise, back to commitment and verification phase**,** the protocol continues. If $J[0] \neq 1$, the secret recovery fails. After $T \geq \text{over}(7)$, I_i will send a repayment request (**RePay**) to ***CC***. If the deposit is refunded correctly, the protocol is terminated.

SCHEME ANALYSIS

Simulating Experiments

In order to compare the security and fairness of BFSDA and DABSMPC more intuitively. This article simulates the execution of the two protocols through scripts. The performances of two protocols in security and fairness are compared in the following security analysis and fairness analysis.

The following three points need to be explained about the simulation experiment.

- All participants are controlled by the script to participate in the execution of the protocol.
- There is a particular script called malicious script simulating the opponent's behavior. It can randomly compromise a given number of participants. If a participant is compromised, the input data will be revealed, and the channel will be monitored. Besides, the opponent can also control its behavior.
- The fair secret reconstruction protocol in BFSDA is built on Truffle, which is a popular testing framework of Ethereum.

Security Analysis

The performances of two protocols in security from simulating experiment are compared in Table 1.

Table 1. Comparison between two protocols in security.

Protocol	Number of malicious node	Information reveled	State of secret reconstruction
DABSMPC	$\leq n-t-1$	None	Success
	$> n-t-1$	All participants' private data	Failure
BFSDA	$\leq n-t-1$	Compromised participants' private data	Success
	$> n-t-1$	Compromised participants' private data.	Failure

In DABSMPC, SMPC is executed by three servers, so n is 3, t is 1. If more than one server is compromised, the opponent will obtain all the participants' private data, and honest participants can't get enough secret shares to recover secret. So, the maximum number of malicious parties that the protocol can bear is 1.

In BFSDA, n is the number of participants and t is the number chosen to balance robustness and security of protocol. If more than $n-t-1$ participants are compromised, honest participants can't get enough secret shares to recover secret. So, the maximum number of malicious parties that the protocol can bear is $n-t-1$. At the meantime, the private input data of the participants not compromised is still safe, because its blind random number is not revealed. Even if two comparison nodes are compromised, all information opponent can get is offset total supply and demand data. The real total supply and demand data are still safe as long as one participating node is not compromised.

The security of BFSMPC has been proven in.[6] During the whole process of the SMPC off the chain, the parties involved ensure that the commitments computed in the computation phase based on VSPS detection and Pedersen homomorphic commitment are correct.

Fairness Analysis

In DABSMPC, whether the three servers recover bid data or compare results, there is no guarantee of fairness.

In BFSDA, the fairness of secret recovery is guaranteed by smart contracts that run automatically.

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

Table 2. BFSDA's measures to different malicious behaviors.

Malicious behavior	Consequence	Measure
Refusing to submit commitment	The protocol cannot be executed properly.	Recording and punishing
Refusing to submit secret share	Other participants cannot recover secret.	Recording and punishing
Submitting wrong secret share	Other participants cannot recover secret.	Recording and punishing
Refusing to submit verify result	The protocol cannot be executed properly.	Recording and punishing
Submitting wrong verify result	Early termination of protocol.	Recheck phase added

As simulating experiment result shown in Table 2, facing different malicious behavior, the smart contract has corresponding measure.

Refusing to submit commitment and verify result causes the protocol not to be executed properly. Honest participants fail to recover secrets when malicious participant submits no share or wrong share. All these behaviors are detected and verified by smart contract, and all malicious participants have their deposits confiscated eventually. Hence, the rational participants will choose to behave.

Submitting wrong verify result causes early termination of the protocol without being punished. To solve this problem, BFSDA adds a recheck procedure in the fair secret reconstruction protocol. Even if the malicious party displays fraudulent behaviors, the smart contract can still judge that all the participants recover the secrets after collecting all the secret shares, so that the protocol can run normally.

Performance Analysis

BFSDA and DABSMPC are both composite protocol with subprotocols, so subprotocols should be analyzed before. BFSDA and DABSMPC both mainly composed of VSS and secure comparison. The comparison of round and complexity of these subprotocols in BFSDA and DABSMPC are shown in Table 3.

Table 3. Comparison of round and complexity of subprotocols In BFSDA and DABSMPC.

	BFSDA	DABSMPC
Round of VSS[a]	$2N$	$2\log_2 N$
Communication complexity of VSS[b]	$O(N)$	$O(N^2)$
Rounds of secure comparison protocol[a]	$\log_2 N$	$\log_2 N$
Computational complexity of secure comparison protocol[c]	$O(\log N)$	$O(1)$

a. N is the amount of possible price.
b. N is the amount of participants.
c. N is the number of binary bits of comparison data.

Compared with DABSMPC, BFSDA is more sensitive to the number of participants. At the meantime, DABSMPC is more susceptible to price ranges.

Beyond that, BFSDA also needs to perform fair secret reconstruction protocols at most $2\log_2 P$ times on blockchain. BFSDA not only makes time consuming and compute-intensive verification operation off the chain, but also makes sure that participant only needs to pay the deposit once at the beginning. These improvements significantly increase the efficiency of the protocol.

Ethereum can produce one block every 15 s. In addition to Ethereum, BFSDA can also run on any blockchain that supports smart contracts and economic systems. The efficiency can be further improved.

From what has been discussed above, BFSDA is more efficient than DABSMPC in application scenarios with a small number of participants and a wide range of bids.

CONCLUSION

BFSDA not only modifies the data input and distribution mechanism of the protocol participants to improve the security of private data but also chooses a fairer and more efficient protocol for secure two-party comparing. To obtain the fairness of the secure multiparty computation protocol, BFSDA introduces BFSMPC. This article modifies the fair and secret reconstruction protocol in BFSMPC for multiple rounds of secret reconstruction requirements of the electronic double auction protocol.

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.

This article simulates the execution of the DABSMPC and BFSDA to compare two protocols' security and fairness. The security analysis shows that the private input data is still safe as long as data owner is not compromised, and the maximum number of malicious parties that BFSDA can bear is $n - t - 1$. The fairness analysis shows that participants in BFSDA will be identified by the smart contract if they try to prevent other participants from recovering secret or terminate the protocol in advance, their deposit will be confiscated eventually. Honest participants will get their deposit back and receive compensations whether the secret is eventually recovered or not. The performance analysis shows that participant only needs to pay deposit once during multiple fair secret reconstruction protocol. A large number of complicated verification operations are carried out off the chain, which ensures the efficiency of the protocol. Compared to DABSMPC, BFSDA reduces the round of VSS but increases the communication complexity of it. When there are too many participants, the communication cost will make the protocol run too slowly. Therefore, this protocol is more suitable for application scenarios with a small number of participants and high protocol security requirements.

REFERENCES

1. K. Suzuki, K. Kobayashi, and H. Morita, "Efficient sealed-bid auction using hash chain," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2000, pp. 183–191.
2. F. Brandt, "Fully private auctions in a constant number of rounds," in *Proc. Int. Conf. Financial Cryptography*, 2003, pp. 223–238.
3. P. Bogetoft *et al.*, "Secure multiparty computation goes live," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, Feb. 2009, pp. 325–343. Revised Selected Papers.
4. R. Cleve, "Limits on the security of coin flips when half the processors are faulty (extended abstract)," in *Proc. 18th ACM Symp. Theory Comput.*, 1986, pp. 364–369.
5. L. Chen, "Fair and efficient protocol for secure two-party comparing under semi-honest model," *Comput. Eng. Appl.*, vol. 46, pp. 126–132, 2010.
6. H. Jianhua, J. Huiya, and L. Zhongcheng, "Constructing fair secure multi-party computation based on blockchain," *Appl. Res. Comput.*, vol. 37, pp. 225–244, Jan. 2020.
7. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 443–458.
8. R. Kumaresan, V. Vaikuntanathan, and P. N. Vasudevan, "Improvements to secure computation with penalties," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 406–417.
9. G. Zyskind, *Efficient Secure Computation Enabled by Blockchain Technology*. Cambridge, MA, USA: Massachusetts Inst. Technol., 2016.
10. R. Gennaro and M. O. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography," in *Proc. 17th Annu. ACM Symp. Princ. Distrib. Comput.*, 1998, pp. 101–111.
11. T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. Int. Cryptol. Conf. Adv. Cryptol.*, 1991, pp. 129–140.

Lietong Liu is currently a postgraduate with the Department of Control Science and Engineering, Tongji University, Shanghai, China. His research interest focuses on the application of SMPC in blockchain. Contact him at liulietong@tongji.edu.cn.

Mingxiao Du is currently working toward the Ph.D. degree with the Department of Control Science and Engineering, Tongji University, Shanghai, China. His research interest focuses on the decentralized consensus mechanism in blockchain Contact him at 1410449@tongji.edu.cn.

Xiaofeng Ma is currently an associate professor with the Department of Control Science and Engineering, Tongji University, Shanghai, China. His research interests include blockchain technologies and application innovation. He is the corresponding author of this article. Contact him at xiaofengma@tongji.edu.cn.

Authorized licensed use limited to: FUDAN UNIVERSITY. Downloaded on September 09,2020 at 15:43:53 UTC from IEEE Xplore. Restrictions apply.