Received May 30, 2021, accepted June 11, 2021, date of publication June 15, 2021, date of current version June 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3089492

A Secure and Fair Double Auction Framework for Cloud Virtual Machines

KE CHENG[1], WEI TONG[2], LELE ZHENG[1], JIAXUAN FU[1], XUTONG MU[1], AND YULONG SHEN[1], (Member, IEEE)
[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[2]School of Cyber Engineering, Xidian University, Xi'an 710071, China

Corresponding author: Yulong Shen (ylshen@mail.xidian.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFE0207600, in part by the Key Research and Development Program of Shaanxi Province under Grant 2021KWZ-04, and in part by the Innovation Fund of Xidian University under Grant YJS2103.

ABSTRACT Double auction is one of the most promising solutions to allocate virtual machine (VM) resources in two-sided cloud markets, which can increase the utilization rate of VM resources. However, most cloud auction mechanisms simply assume that the auctioneer is fully trusted while ignoring bid-privacy preservation and trade fairness in the process of auction. Previous studies have indicated that some cryptographic tools can be used to resolve the above issues, but the poor performance makes those techniques difficult to practice. In this paper, we propose a Secure and Fair Double AuCtion framework (named SF-DAC) for cloud virtual machines, which performs cloud auction efficiently while guaranteeing both bid privacy and trade fairness. We design secure 3-party computation protocols that support secure comparison and secure sorting, which enable us to construct a secure double auction scheme that outperforms all prior comparable solutions. Furthermore, we propose a fair trading mechanism based on smart contracts to prevent the bidders from halting the auction without financial penalties. The extensive experiments demonstrate that SF-DAC achieves an order of magnitude reduction in computation and communication costs than prior arts.

INDEX TERMS Privacy preservation, secure double cloud auction, secure three-party protocol, trade fairness.

I. INTRODUCTION

Auction-based resource allocation mechanism is increasingly adopted for cloud virtual machine (VM) markets to cut costs and increase revenues. For example, there are a large number of VMs transactions through auction-style trading in Amazon EC2. This trading mechanism is more common in complex two-sided cloud markets. A two-sided cloud market means multiple sellers provide various types of cloud VMs for rental services, while multiple buyers can purchase VMs from multiple sellers to implement their respective tasks. Double cloud auction is specifically designed for resource allocations in the two-sided cloud markets [1].

Recently, various cloud auction schemes [1], [2] have been proposed for VMs transactions with considering economic characteristics such as truthfulness, budget balance, and social welfare maximization. The existing cloud auction schemes [1], [2] implicitly assume that the auctioneer is fully trusted, and all bidders (including all sellers and buyers) send their bids to the auctioneer unreservedly. However, the auctioneer is not always trustable in practice, which would lead to potential security risks and an absence of fairness. For example, the auctioneer can simply adapt auction rules to obtain extra profit by monitoring the bids from the sellers and buyers. A dishonest bidder can select a bid untruthfully to snatch maximum profit through prying into other participants' private bids. More severely, by studying the historical bids, external attackers can send the auctioneer hostile bids to disrupt the normal order of the auction. In terms of auction fairness, a dishonest bidder may drop out of the auction after learning other participants' bids, harming the interests of other bidders. In short, the lack of bid-privacy protection and trade fairness would lead to alarming economic losses during the auctions.

To prevent bid information leakage, Chen *et al.* [3] design the first privacy-preserving cloud auction. A crypto-service provider is introduced in this solution and works with the auctioneer to constitute a two-party computation (2PC) model [4]. In this model, bidders send encrypted bids to two parties (the auctioneer and crypto-service provider) who

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Aljawarneh.

perform privacy-preserving auction protocol based on the garbled circuit (GC) [5], to provide a privacy guarantee that no bid information is revealed to two parties except the auction results. Since this scheme fails to work for the more complicated two-sided cloud markets, a secure double auction mechanism PDAM is proposed [6]. Based on the multiple cryptographic primitives, PDAM designs a hybrid auction protocol in the 2PC model. Unfortunately, these auction schemes bring considerable end-to-end (including offline stage and online stage) computation time and communication costs, making the systems very hard to deploy into realistic environments. To achieve fair trading, a recent work [7] proposes a decentralized cloud auction and trading framework based on smart contracts [8]. However, the protection of the bid privacy is not considered in this work. In summary, designing a practical cloud auction scheme with bid-privacy protection and trade fairness remains a challenge.

Inspired by the above efforts, we propose a Secure and Fair Double AuCtion framework (named SF-DAC) for the two-sided cloud markets. By following the SF-DAC framework, the bid privacy of auction participants can be protected against adversaries, and every auction participant will receive the profits they deserve as long as they faithfully follow the auction protocols without an abort. To achieve the design goal, we are facing two main challenges. The first challenge originates from the conflicting requirements of high efficiency and strong security. Existing secure cloud auctions [3], [6] involve plenty of time-consuming encryptions/decryptions or large-scale circuit evaluations, which lead to considerable computation and communication costs. The second challenge is how to ensure fairness in the entire auction at low system overheads. To address these challenges, our work makes efforts in the following areas. We first introduce two auction agents who collaborate with the auctioneer to construct a three-party computation (3PC) model [9]. All bidders send encrypted bids to the auctioneer and one agent, and then the two parties collectively run the secure interactive auction protocol with the help of another agent. Then, we design 3PC-based protocols for secure comparison and secure sorting, and construct a secure and efficient double auction protocol based on that, leading to an order of magnitude speedup compared with the 2PC-based solutions in [3] and [6]. Furthermore, SF-DAC leverages the smart contract to motivate each auction participant to faithfully follow the auction protocols, and successfully accomplish the VM resource trading. In summary, we make the following contributions.

- To the best of our knowledge, we are the first to design a secure and fair auction framework towards double cloud auctions by leveraging the 3PC model, which can achieve bid-privacy preservation, trading fairness, and high efficiency simultaneously.
- We propose secure 3PC-based protocols for performing various secure arithmetic operations, secure comparison, and secure sorting based on the lightweight additive secret sharing, which are well applicable in other auctions, such as spectrum auctions [10], [11], advertising auctions [12]. Moreover, to address the fairness issue in secure cloud auctions, we design a fair and efficient trading scheme based on smart contracts to incentivize each auction participant to faithfully follow the auction protocols.
- We present a thorough theoretical analysis of SF-DAC in terms of both security and fairness. To demonstrate its practicality, we implement the prototype of SF-DAC and evaluate its performance by comparative experiments. Concretely, SF-DAC incurs 69× lower total computation time and 108× lower communication costs than the state-of-the-art double cloud auction [6].

The rest of the paper is organized as follows. Section II reviews the related works. Section III gives an overview of our system. Section IV presents the preliminaries. A set of secure sub-protocols are provided in Section V. In Section VI, the design of the SF-DAC framework is explained in detail. We analyze the security and the trade fairness of SF-DAC in Section VII. The proposed protocols and framework are evaluated through extensive experiments in Section VIII. We make a conclusion in Section IX.

II. RELATED WORK

A. CLOUD VIRTUAL MACHINE AUCTION

There are plenty of works [2], [13]–[18] using auction-based pricing models to balance users' and cloud VM provider's benefits. Shi *et al.* [13] propose the first online combinatorial auction mechanism to optimize system efficiency across the temporal domain and model heterogeneous VMs in practice. Some works [14], [15] use game theory to optimize users' profits in auction-based cloud VM trading. Li *et al.* [2] take the heterogeneous demands into consideration and propose a truthful auction mechanism in IaaS clouds to maximize the cloud provider's profit. Zheng *et al.* [16] design an online auction mechanism for service clouds, with unique features of job-oriented users and soft deadline constraints. Parida *et al.* [17] propose a dynamic cloud auction scheme with providing a flexible mechanism to change the VM cost dynamically during the auction. Li *et al.* [18] formulate the VM resource pricing and auction problem as a bin packing problem, which yields high efficiency. However, all of the above works mainly focus on cloud VM auctions in a single-sided market, while the two-sided cloud market is more common and more complicated in practice.

With cloud integration services increasingly popular, multiple users and multiple VM providers are involved in cloud trading. There has been a flurry of recent works [1], [19]–[22] in the area of the double auction for the two-sided cloud market. Li *et al.* [19] propose a double auction mechanism for a two-sided cloud market, which enables the users to have the choice of purchasing resources from multiple providers to achieve higher cost-efficiency. Samimi *et al.* [20] present a combinatorial double cloud auction, which allows buyers to submit package bids for various VM resources.

Singhal and Singhal [21] propose a feedback-based combinatorial double cloud auction, which allows the users to access resources from different providers at optimal prices, by prioritizing genuine providers with good feedback over in-genuine providers with bad feedback. Gao *et al.* [22] present a VM resource auction mechanism to allocate VMs in geo-distributed edge cloud nodes to users and design a greedy approximation algorithm to determine the winners of the auction. Lu *et al.* [1] propose a double auction mechanism (denoted as DAM), which is individual-rational, truthful, and budget-balanced. In this mechanism, they use the method of the second-price auction [23] for the pricing and allocation of cloud virtual machines. However, none of the above works take privacy preservation and trade fairness into consideration.

Among all of the above cloud auctions, we select the DAM scheme [1] as the underlying scheme of our secure solution. There are two main reasons for this choice. Firstly, the DAM scheme is an efficient trading mechanism for the two-sided cloud markets, while being proved to be individually rational, truthful, and budget-balanced. But the other works [19]–[22] do not provide a complete proof for all the above properties. Secondly, a second-price method is employed in the DAM scheme to determine the winners and their clearing price, which is more tractable in a secure manner. Because implementing the second-price method requires few loop statements. Some works like [22] use a greedy algorithm to determine the winners, which need to invoke a significant number of loop statements. Obviously, it is difficult to construct an efficient cloud auction in a secure manner based on these works.

B. SECURE AND FAIR AUCTION

In recent years, some works [7], [24], [25] consider the fairness of the auction-based resource allocation. The works [24], [25] reduce the probability of user quit-auction problems based on the reputation system. Liu *et al.* [26] propose a blockchain-based fair and secure double auction protocol. But it involves a large number of complicated verification operations, which makes it more suitable for application scenarios with a small number of participants. To ensure trade fairness in the cloud auction, the work [7] uses the smart contract to charge a penalty for dishonest bidders.

As for bid privacy, Chen *et al.* [3] first propose a privacy-preserving cloud auction scheme for a single-sided market, which develops a data-oblivious cloud auction algorithm and then employs garbled circuits to implement a secure solution. Further, by employing homomorphic encryption (HE), Xu *et al.* [27] design a privacy-preserving double auction (denoted as HE-based auction) for the more complicated two-sided market. However, this solution is based on expensive cryptographic primitives and incurs high computation and communication overheads. Recently, by combining two-party secret sharing and garbled circuits together, a privacy-preserving double auction mechanism (PDAM) [6] is proposed under the 2PC model. PDAM optimizes the 2PC auction protocol into the offline-online setting so as to improve the online performance, but the offline stage requires considerable computation and communication costs.

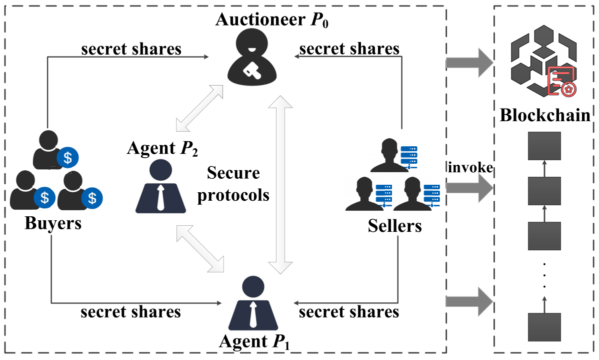


FIGURE 1. System architecture of SF-DAC.

A related research topic that we need to mention is secure spectrum auctions, which usually employ a similar technical route as secure cloud auctions. Chen *et al.* [28] first present an information-theoretically secure framework ITSEC for truthful spectrum auctions. The following work [10] has improved the efficiency of ITSEC. The recent work [11] proposes a privacy-preserving and truthful double auction mechanism PS-TAHES for heterogeneous spectrum. Wang *et al.* [29] design a secure cloud auction scheme PROST to provide comprehensive protection for users' location privacy and time dynamics. Based on the trusted processors and the smart contracts, a recent work [30] designs a general secure and fair auction framework named SAFE for wireless markets. To summarize, Table 1 makes a comparison of the property differences between SF-DAC and the above secure and fair auctions.

III. SYSTEM OVERVIEW

In this section, we present the system model, threat model, and design goals.

A. SYSTEM MODEL

As illustrated in Figure 1, SF-DAC involves an auctioneer (P_0), two auction agents (P_1 and P_2), sellers, and buyers. In the two-sided cloud market, N_b buyers purchase M types of cloud VMs from N_s sellers. In our privacy-preserving solution, buyers and sellers send the secret shares of the bids to the auctioneer P_0 and the auction agent P_1, respectively. To achieve the secure auction, P_0 and P_1 execute secure interactive protocols with the help of P_2 on the input shares, and invoke smart contracts to complete the trading automatically.

We emphasize that the three parties P_0, P_1, and P_2 do not collude to break the protocols. There are some ways to realize it. First, these parties are typically managed by different companies. Therefore, collusion among them is highly unlikely as it will damage their reputation, which affects their revenues. Second, with blockchain technology, we can use an economic approach to bound the probability of collusion [31], [32].

TABLE 1. Comparison of SF-DAC with existing secure and fair auctions.

Properties	Truthfulness	Double Auction	Bid Privacy	Fairness	High Efficiency
[2], [13]–[18]	✓	×	×	×	✓
[1], [19]–[22]	✓	✓	×	×	✓
[7], [24], [25]	✓	×	×	✓	✓
[3]	✓	×	✓	×	×
[6], [10], [11], [27]–[29]	✓	✓	✓	×	×
[26], [30]	✓	✓	✓	✓	×
Our Work	✓	✓	✓	✓	✓

Third, we observe that the servers are now increasingly equipped with a set of trusted computing mechanisms powered by software and hardware designs, such as Intel SGX [33] and Arm TrustZone [34], which make the computations on the servers more trustworthy. Actually, this model is not new, which is a widely used assumption in previous works [35]–[38]. Similar to the second method [31], we also employ blockchain technology to guarantee that each participant will perform rational behaviors. So our work can adopt the second method [31] to restrain the three parties from colluding. We refer readers to [31] for details.

B. THREAT MODEL AND DESIGN GOALS

The main goal of SF-DAC is to guarantee bid privacy and trade fairness for the cloud auction. As mentioned above, following many popular 3PC-based security works [35]–[38], we assume that only one party among P_0 to P_2 is corrupted by the semi-honest adversary, who performs the protocol faithfully but tries to infer private information from the interactive messages. To demonstrate the security of SF-DAC, we resort to the security proof method in the real-ideal paradigm [39]. The security of a protocol is modeled by defining two interactions: a real interaction where the parties run a protocol π and an ideal interaction where parties send their inputs to an ideal functionality $\mathcal{F}$ completed by a trusted party. A protocol π is said to realize a functionality $\mathcal{F}$ securely if for every adversary $\mathcal{A}$ in the real interaction, there is a simulator $\mathcal{S}$ in the ideal interaction, such that the ideal world adversary's view is indistinguishable from the real world adversary's view.

It is worth mentioning that, malicious adversary model [40] is a stronger threat model where the adversary is able to launch an attack actively to break secure multi-party computation (MPC) protocols. Although there are some general techniques [40], [41] to convert any semi-honest secure MPC protocol into a secure MPC protocol against malicious attacks, these methods introduce a large amount of overhead or the trusted computing base. Designing tailored MPC protocols under the malicious adversary model for the double cloud auctions is an interesting research direction, and we leave this as future work.

Furthermore, our system uses smart contracts to complete the trading for ensuring fairness. The trade fairness ensures that 1) it is impossible for a seller to drop out of the auction without financial penalties, or to get fees without providing requested VM resources; 2) it is impossible for a buyer to drop out of the auction without financial penalties, or to obtain VM resources without paying fees; 3) the auctioneer and the agents should be given the corresponding reward.

TABLE 2. Notations and definitions.

Notations	Definitions
N_b, N_s, and M	numbers of buyers, sellers, and VM types
$b_i = \langle x_i, c_i \rangle$	buy-bid of buyer i
$x_i = (x_i^1, \ldots, x_i^M)$	buyer i's purchasing need
x_i^m	desired quantity of type-m VMs
c_i	payment for the purchasing need $x^{(i)}$
ϕ_i	bid density of buy-bid b_i
$\mu = (\mu_1, \ldots, \mu_M)^{\mathrm{T}}$	capacity utilization for each type of VMs
$s_j^m = \langle X_j^m, \theta_j^m \rangle$	sell-bid of seller j
X_j^m	number of type-m VMs provided by the seller j
θ_j^m	unit price of type-m VMs required by the seller j

IV. PRELIMINARIES

In this section, we review a plaintext algorithm for double cloud auction and introduce secure cryptographic primitives and smart contracts that are used in the paper. Table 2 summarizes the notations and definitions used in this paper.

A. DOUBLE AUCTION FOR CLOUD VMs ALLOCATION

We introduce an advanced scheme DAM for double cloud auction [1]. Initially, all buyers submit the buy-bids $b_i = (x_i, c_i)(1 \leq i \leq N_b)$ to the auctioneer, where $x_i = (x_i^1, \ldots, x_i^M)$ is the buyer i's purchasing need, x_i^m is the number of type-m VMs,[1] c_i is the payment for the purchasing need. At the same time, all sellers submit the sell-bids $s_j^m = (X_j^m, \theta_j^m)(1 \leq m \leq M, 1 \leq j \leq N_s)$ to the auctioneer, where X_j^m is the number of type-m VMs held by seller j, and θ_j^m is the unit price of type-m VMs required by the seller j. After that, the auctioneer will execute the following steps to determine the auction results.

1) WINNER MATCHING

The auctioneer generates the bid density for the buyer i by

$$\phi_i = c_i/(x_i \cdot \mu), \tag{1}$$

where $\mu = (\mu_1, \ldots, \mu_M)^{\mathrm{T}}$ is the capacity utilization for each type of VMs, and is public to all participants.

The auctioneer determines the winning sellers and buyers according to second-price auction rules [23]. That is, the buy-bids are sorted in non-ascending order according to bid densities, and the sell-bids are sorted in non-descending

[1] In the original work, x_i is obtained by a cost-aware resource provisioning algorithm with the input of the users' task requirements, the payments, and some public parameters. Since it is independent of the auction process, we omit this step.

order according to the quoted price of type-m VMs. The auctioneer obtains the following ranked bid lists L_1 and L_2.

$$L_1 : \phi_{\alpha_1} \geq \phi_{\alpha_2} \geq \cdots \geq \phi_{\alpha_{N_b}}. \tag{2}$$
$$L_2 : \theta^m_{\beta_1} \leq \theta^m_{\beta_2} \leq \cdots \leq \theta^m_{\beta_{N_s}}, \forall m \in \{1, \ldots, M\}. \tag{3}$$

Note that α_i is the index of i-th largest bid density and β_j is the index of j-th smallest quoted price. After that, the auctioneer finds the index (j^*) of the least profitable transaction by:

$$j^* = \arg\max\{\sum_{m=1}^{M} x^m_{\alpha_1} \cdot \theta^m_{\beta_{j^*}} \leq \phi_{\alpha_2} \cdot x_{\alpha_1} \cdot \mu\} \tag{4}$$

The first buyer in L_1 (with largest bid density) and the top $j^* - 1$ sellers in L_2 (with sell-bids $\theta^m_{\beta_1}, \ldots, \theta^m_{\beta_{j^*-1}}$) are the auction winners.

2) PRICING AND VMs ALLOCATION

The auctioneer calculates the payment of the winning buyer win_b is

$$\widehat{c}_{win_b} = \phi_{\alpha_2} \cdot x_{win_b} \cdot \mu; \tag{5}$$

The unit price of type-m VM charged by the winning seller win_s is

$$\widehat{\theta}^m_{win_s} = \theta^m_{\beta_{j^*}}; \tag{6}$$

Each winning seller win_s sells the same amount ($\widehat{X}^m_{win_s}$) of type-m VMs to the winning buyer, where

$$\widehat{X}^m_{win_s} = \left\lceil \frac{x^m_{win_b}}{(j^* - 1)} \right\rceil \tag{7}$$

At this point, the auctioneer has finished one round of the double auction, then remove the winning buyer and the winning sellers from the auction.

B. ADDITIVE SECRET SHARING

Given an ℓ-bit value x, a 2-out-of-2 additive secret sharing of x (denoted by $\langle x\rangle$) is a pair $(\langle x\rangle_0, \langle x\rangle_1) = (x-r, r)$ (such that $x = \langle x\rangle_0 + \langle x\rangle_1$) where r is a random number from the ring $\mathbb{Z}_{2^\ell}$ [42]. Additive secret sharing can perfectly hide value x as long as no party obtains both $\langle x\rangle_0$ and $\langle x\rangle_1$.

To compute the sum of two shared values $\langle x\rangle$ and $\langle y\rangle$, party P_i locally computes $\langle x+y\rangle_i = \langle x\rangle_i + \langle y\rangle_i$. To perform secure multiplication $\langle x \cdot y\rangle = \langle x\rangle \cdot \langle y\rangle$, we use Beaver's multiplicative triples (MTs) [42] of the form $\langle ab\rangle = \langle a\rangle \cdot \langle b\rangle$. Party P_i locally computes $\langle e\rangle_i = \langle x\rangle_i - \langle a\rangle_i$ and $\langle f\rangle_i = \langle y\rangle_i - \langle b\rangle_i$. To recover e and f, P_0 and P_1 exchange $\langle e\rangle_i$ and $\langle f\rangle_i$, and compute $e = \langle e\rangle_0 + \langle e\rangle_1$ and $f = \langle f\rangle_0 + \langle f\rangle_1$. At last, P_0 sets $\langle ab\rangle_0 = f \cdot \langle a\rangle_0 + e \cdot \langle b\rangle_0 + \langle c\rangle_0$ and P_1 sets $\langle ab\rangle_1 = e \cdot f + f \cdot \langle a\rangle_1 + e \cdot \langle b\rangle_1 + \langle c\rangle_1$. For division over the secret-shared data, we invoke a state-of-the-art cryptographic protocol [43], denoted as SecDiv(). On input $\langle x\rangle$ and $\langle y\rangle$, SecDiv outputs secret shares of $\langle x/y\rangle$ with the required accuracy. We only introduce the function of this protocol; readers may refer to the original study [43] for details.

C. SMART CONTRACT

Smart contracts in Ethereum [8] are transaction-driven, state-based code contracts that extend blockchain applications beyond monetary transactions to more practical functions. Each smart contract contains a transaction processing and storage mechanism, and a Turing-complete state machine. This contract is signed and verified by multiple nodes just like a normal transaction to ensure its validity, and the valid contract is successfully executed. The whole process is completed automatically by Ethereum with the characteristics of transparency and immutability. The contract's code cannot be modified forever even by its creator.

V. CRYPTOGRAPHIC BUILDING BLOCKS

Before describing the specifics of the SF-DAC Framework, we construct some cryptographic building blocks, including secure comparison protocol and secure sorting protocol, by using additive secret sharing under the 3PC model. Recall that we assume the existence of three parties P_0, P_1, and P_2, such that secure interactions happen among them. P_2 is an assist party in these secure protocols by providing relevant randomness and assistant parameters for P_0 and P_1. All operations to be performed are on the ring $\mathbb{Z}_{2^\ell}$.

A. SECURE COMPARISON PROTOCOL

To support the sort over two secret-shared data, we propose a secure comparison (SC) protocol. Assume that P_0 and P_1 hold the shares of x, y over $\mathbb{Z}_{2^\ell}$. On input $\langle x\rangle$, $\langle y\rangle$, SC outputs $(\langle \min(x,y)\rangle, \langle \max(x,y)\rangle)$ without revealing any information about x, y to P_0, P_1, and P_2. The rationale behind SC protocol is as follows:

$$\begin{cases} \min(x,y) = x + f(y-x) \\ \max(x,y) = y + f(x-y) \end{cases} \quad f := x \geq y. \tag{8}$$

The detailed procedure of SC protocol is described in Algorithm 1. To begin with, our protocol securely computes the flag $\langle f\rangle$ with the help of P_2, where f specifies whether x is greater than y ($f = 1$) or not ($f = 0$). Concretely, for $\forall(x,y)$, $(x-y)r \geq 0 \Rightarrow x \geq y$ with $r > 0$. For that, P_0 and P_1 generate random positive numbers $\langle r\rangle_0$ and $\langle r\rangle_1$, respectively. P_2 generates the secret shares of value 0 and 1, i.e., lets $\langle u\rangle_i = \langle 0\rangle_i$ and $\langle v\rangle_i = \langle 1\rangle_i$, $i \in \{0,1\}$ (Line 1). Then P_0 and P_1 compute $\langle t_2\rangle = \langle (x-y) \cdot r\rangle$ based on Beaver MTs (Line 2 & 3). P_0 and P_1 send $\langle t_2\rangle_0$ and $\langle t_2\rangle_1$ to P_2, who recovers t_2 by $t_2 = \langle t_2\rangle_0 + \langle t_2\rangle_1$. If $t_2 \geq 0$, P_2 sends secret shares of value 1 to P_0 and P_1, respectively. Otherwise, P_2 sends secret shares of value 0 to them (Line 4 & 5). After that, P_0 and P_1 compute the minimum and maximum of x and y based on Equation (8) cooperatively (Line 6 & 7). At the end of SC protocol, P_0 outputs the sorted pair $(\langle \min(x,y)\rangle_0, \langle \max(x,y)\rangle_0)$, and P_1 outputs $(\langle \min(x,y)\rangle_1, \langle \max(x,y)\rangle_1)$.

B. SECURE SORTING PROTOCOL

The underlying scheme in Section IV-A shows that the sorting process is one of the most crucial operations during the

Algorithm 1 Secure Comparison (**SC**) Protocol

```
Input: $P_0$ inputs $(\langle x\rangle_0, \langle y\rangle_0)$, $P_1$ inputs $(\langle x\rangle_1, \langle y\rangle_1)$
Output: $P_0$ outputs $(\langle \min(x,y)\rangle_0, \langle \max(x,y)\rangle_0)$
        $P_1$ outputs $(\langle \min(x,y)\rangle_1, \langle \max(x,y)\rangle_1)$
1: $P_0$ and $P_1$ generate random positive numbers $\langle r\rangle_0$ and $\langle r\rangle_1$, $P_2$ generates the secret shares of value 0 and 1, i.e., $\langle u\rangle_i = \langle 1\rangle_i$ and $\langle v\rangle_i = \langle 0\rangle_i$, $i \in \{0,1\}$.
2: $P_i$ computes $\langle t_1\rangle_i = \langle x\rangle_i - \langle y\rangle_i$, $i \in \{0,1\}$.
3: $P_0$ and $P_1$ compute $\langle t_2\rangle = \langle t_1\rangle \cdot \langle r\rangle$ by Beaver MTs.
4: $P_i$ $(i \in \{0,1\})$ sends $\langle t_2\rangle_i$ to $P_2$.
5: $P_2$ recovers $t_2$ by computing $t_2 = \langle t_2\rangle_0 + \langle t_2\rangle_1$.
   If $t_2 \geq 0$, $P_2$ lets $\langle f\rangle = \langle u\rangle$. Otherwise $P_2$ lets $\langle f\rangle = \langle v\rangle$.
   $P_2$ sends $\langle f\rangle_0$ and $\langle f\rangle_1$ to $P_0$ and $P_1$, respectively.
6: $P_0$ and $P_1$ computes $\langle t_3\rangle = \langle x\rangle + \langle f\rangle \cdot (\langle y\rangle - \langle x\rangle)$ and $\langle t_4\rangle = \langle y\rangle + \langle f\rangle \cdot (\langle x\rangle - \langle y\rangle)$.
7: $P_0$ outputs $(\langle \min(x,y)\rangle_0, \langle \max(x,y)\rangle_0) = (\langle t_3\rangle_0, \langle t_4\rangle_0)$,
   $P_1$ outputs $(\langle \min(x,y)\rangle_1, \langle \max(x,y)\rangle_1) = (\langle t_3\rangle_1, \langle t_4\rangle_1)$.
```

auction. Therefore, we implement a secure sorting (**SST**) protocol on the foundation of additive secret sharing and **SC** protocol. Consider a array of values $X = [x_1, \ldots, x_n]$ is secret-shared into two shares $\langle X\rangle_0 = [\langle x_1\rangle_0, \ldots, \langle x_n\rangle_0]$ and $\langle X\rangle_1 = [\langle x_1\rangle_1, \ldots, \langle x_n\rangle_1]$, stored in P_0 and P_1 respectively. Let α_i is the index of the i-th largest element in the array X. In **SST** protocol, with the help of P_2, P_0 and P_1 obtain a new sorted secret-shared array $\langle X_\alpha\rangle = [\langle x_{\alpha_1}\rangle, \langle x_{\alpha_2}\rangle, \cdots, \langle x_{\alpha_n}\rangle]$.

The security requirements imply that a secure sorting protocol should be input-independent, i.e., the execution path of **SST** protocol does not rely on the input data. However, most of the conventional sorting algorithms are input-dependent, which would cause the leakage of data access patterns (e.g., the order among the elements in X). In consideration of security and performance, the construction of **SST** protocol is built on the *Shuffle-then-Compute* strategy. We first shuffle the input array and then use the disturbed data as the input to the conventional sorting process. Since the shuffled array is disconnected from the original input, the adversary cannot infer the access patterns during the sorting process. Concretely, we use a *permutation matrix* to permute elements to obfuscate their orders. For example, given the original array $X_a = [x_{a_1}, x_{a_2}, \cdots, x_{a_n}]$, we can get a new array $X_b = [x_{b_1}, x_{b_2}, \cdots, x_{b_n}]$ of the same elements with different orders, using an $n \times n$ sorting matrix W. For moving the element x_{a_i} to the new position x_{b_j}, we set $w_{i,j} = 1$ and $w_{i,k} = 0, \forall k \neq j$. The following equation illustrates the shuffle operation by permutation matrix with $n = 3$.

$$[x_1, x_2, x_3] \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = [x_3, x_1, x_2]$$

Algorithm 2 shows the steps of **SST** protocol. Party P_2 uses pseudo random generator (PRG) to generate a random permutation matrix W and the assistant matrices A, B, C where $C = A \cdot B$. Then P_2 sends the secret shares of these matrices to P_0 and P_1, respectively. P_0 and P_1 execute the

Algorithm 2 Secure Sorting (**SST**) Protocol

```
Input: $P_0$ inputs $\langle X\rangle_0 = [\langle x_1\rangle_0, \ldots, \langle x_n\rangle_0]$
       $P_1$ inputs $\langle X\rangle_1 = [\langle x_1\rangle_1, \ldots, \langle x_n\rangle_1]$
Output: $P_0$ outputs $\langle X_\alpha\rangle_0 = [\langle x_{\alpha_1}\rangle_0, \ldots, \langle x_{\alpha_n}\rangle_0]$
        $P_1$ outputs $\langle X_\alpha\rangle_1 = [\langle x_{\alpha_1}\rangle_1, \ldots, \langle x_{\alpha_n}\rangle_1]$
1: $P_2$ generates a random permutation matrix $W \in \mathbb{Z}_2^{n\times n}$ and two random matrices $A \in \mathbb{Z}_{2^\ell}^{1\times n}$, $B \in \mathbb{Z}_{2^\ell}^{n\times n}$ by PRG. $P_2$ gets a $1 \times n$ matrix $C$ by $C = A \cdot B$.
2: $P_2$ gets the secret shares of $W, A, B, C$, and sends $(\langle W\rangle_0, \langle A\rangle_0, \langle B\rangle_0, \langle C\rangle_0)$,$(\langle W\rangle_1, \langle A\rangle_1, \langle B\rangle_1, \langle C\rangle_1)$ to $P_0$ and $P_1$, respectively.
3: $P_0$ computes $\langle E\rangle_0 = \langle X\rangle_0 - \langle A\rangle_0$ and $\langle F\rangle_0 = \langle W\rangle_0 - \langle B\rangle_0$. $P_1$ computes $\langle E\rangle_1 = \langle X\rangle_1 - \langle A\rangle_1$ and $\langle F\rangle_1 = \langle W\rangle_1 - \langle B\rangle_1$.
4: $P_0$ and $P_1$ exchange the shares of $E, F$, then recover $E, F$ by $E = \langle E\rangle_0 + \langle E\rangle_1$ and $F = \langle F\rangle_0 + \langle F\rangle_1$.
5: $P_0$ computes $\langle X'\rangle_0 = \langle X\rangle_0 F + E\langle W\rangle_0 + \langle C\rangle_0$, $P_1$ computes $\langle X'\rangle_1 = \langle X\rangle_1 F + E\langle W\rangle_1 + \langle C\rangle_1 - EF$.
6: for $i = 0$ to $n$ do
       for $j = 0$ to $n - i - 1$ do
           $P_0$ and $P_1$ invoke SC($\langle X'_j\rangle, \langle X'_{j+1}\rangle$).
       end for
   end for
7: $P_0$ outputs $\langle X_\alpha\rangle_0 = \langle X'\rangle_0$, $P_1$ outputs $\langle X_\alpha\rangle_1 = \langle X'\rangle_1$.
```

matrix multiplication to obtain $\langle X'\rangle = \langle X \cdot W\rangle$ (Line 3-5), i.e., each party receives one share of the shuffled array X'. After that, P_0 and P_1 invoke **SC** protocol to execute bubble sort [44]. Note that, we can also choose other classical sorting algorithms (e.g., quick sort and Shell sort [45]) in this step, and have no effect on the security of **SST**. At the end of the protocol, P_0 and P_1 output the ranked array X_π in the secret-shard form.

The above section demonstrates that **SST** protocol can sort a secret-shared one-dimensional array. We now describe how to upgrade this protocol to make it work on a secret-shared key-value array. To this end, we begin by upgrading **SC** protocol, which is the cornerstone of **SST** protocol. To avoid repetition, we only describe the main changes of the upgraded **SC** protocol. On input the secret-shared key-value array $[(\langle k_1\rangle, \langle x_1\rangle), \langle k_2\rangle, \langle x_2\rangle)]$, the upgraded **SC** protocol outputs $[(\langle k_{\alpha_1}\rangle, \langle x_{\alpha_1}\rangle), \langle k_{\alpha_2}\rangle, \langle x_{\alpha_2}\rangle)]$, where α_1 is the index of minimum value between x_1 and x_2, α_2 is the index of maximum value between x_1 and x_2. P_0 and P_1 first compute $\langle f\rangle$ according to x_1 and x_2, and then execute the same computations on $\langle x_i\rangle$ and $\langle k_i\rangle$, such that the keys can be swapped according to their values. For the upgraded **SST** protocol, the input is the secret-shared key-value array $(\langle k_1\rangle, \langle x_1\rangle), (\langle k_2\rangle, \langle x_2\rangle), \cdots, (\langle k_n\rangle, \langle x_n\rangle)$, and similarly for the output. In addition, we use the same permutation matrix W to shuffle x_i and k_i simultaneously, and use the upgraded **SC** protocol to swap the key-value pairs. That is, if the values are reordered so do their corresponding keys. In a similar way, **SST** protocol can be extended to support secure sorting over the secret-shared multi-dimensional array.

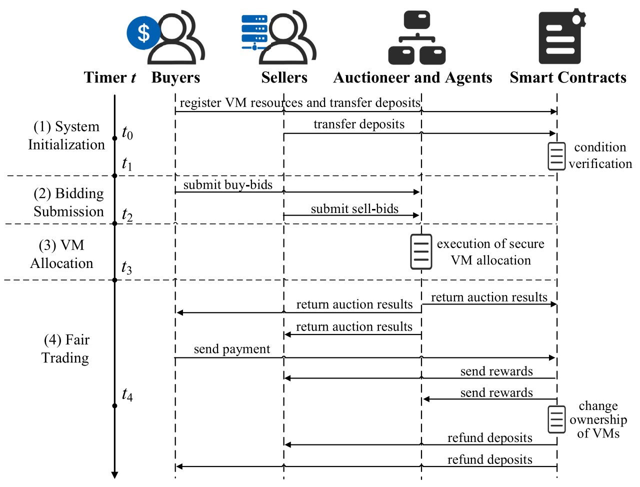


FIGURE 2. **SF-DAC overview.**

VI. DESIGN OF SF-DAC FRAMEWORK

In this section, we specify the design details of the SF-DAC framework. As mentioned before, our design rationale is based on the auction mechanism DAM described in Section IV-A. The technical difficulty is to execute the auction on the secret-shared data in a data-oblivious manner. To achieve this, we improve the DAM auction mechanism to make it data-oblivious, and refactor it by using secure sub-protocols presented in Section V. Moreover, we ensure the trade fairness of our auction scheme by the smart contract.

As illustrated in Figure 2, the SF-DAC framework consists of the following four phases: system initialization, bidding submission, secure auction, and fair trading. First, the system initialization establishes system parameters and gathers deposits from all auction participants. Then, the sellers and the buyers submit the secret-shared bids to the auctioneer and the agent. After that, the main auction protocol is executed among the auctioneer and the two agents. Finally, fair trading is completed to reach a fair transaction.

A. SYSTEM INITIALIZATION

In this phase, the auctioneer sets some system parameters, and the sellers and the buyers transfer a deposit to the smart contract. Algorithm 3 shows the main steps in this phase. The auctioneer sets the deposit-transfer end time t_0, the initialization end time t_1, the bid-submission end time t_2, the payment start time t_3, and the payment end time t_4 to ensure that the auction can be achieved in a limited time. Then, the auctioneer sets the deposit dpt_{seller} that each seller needs to pay, where $dpt_{seller} = pledge_S + GL$, $pledge_S$ denotes the pledge of each seller and GL denotes the max cost for executing the smart contracts. Similarly, the auctioneer sets the deposit dpt_{buyer} that each buyer needs to pay, where $dpt_{buyer} = pledge_B$, $pledge_S$ denotes the pledge of each seller. After that, each seller registers VM resources with the smart contract SC_{init}, and transfers to SC_{init} a deposit $dpts_{S_j}$. Analogously, each buyer transfers to SC_{init} a deposit dpt_{B_i}. When the current time $t \geq t_1$, SC_{init} verifies the following

Algorithm 3 System Initialization
```
1: The auctioneer $P_0$ sets the deposit-transfer end time $t_0$,
   the initialization end time $t_1$, the bid-submission end
   time $t_2$, the payment start time $t_3$, and the payment end
   time $t_4$, the deposit $dpt_{buyer}$ that each buyer needs to
   pay, the deposit $dpt_{seller}$ that each seller needs to pay.
2: **if** the current time $t < t_0$ **then**
3:    **for** $1 \leq j \leq N_s$ **do**
4:       Seller $j$ registers VM resources with the smart
         contract $SC_{init}$, transfers to $SC_{init}$ a deposit $dpts_{S_j}$.
5:    **end for**
6:    **for** $1 \leq i \leq N_b$ **do**
7:       Buyer $i$ transfers to $SC_{init}$ a deposit $dpt_{B_i}$.
8:    **end for**
9: **else**
10:   $SC_{init}$ verifies the following conditions:
      $\forall j, dpts_{S_j} \geq dpt_{seller}$ and $\forall i, dpt_{B_i} \geq dpt_{buyer}$.
      If the conditions are not satisfied, abort the protocol.
11: **end if**
```

conditions: $\forall j, dpts_{S_j} \geq dpt_{seller}$ and $\forall i, dpt_{B_i} \geq dpt_{buyer}$. If the conditions are not satisfied, the initialization protocol is terminated. Otherwise, we can move to the next phase.

B. BIDDING SUBMISSION

In the bid-submission phase (the current time $t < t_2$), all sellers first divide their ID ID_j^s and sell-bids $s_j^m (1 \leq m \leq M, 1 \leq j \leq N_s)$ locally by additive secret sharing, and then send the two shares of the IDs and buy-bids to the auctioneer (P_0) and the auction agent (P_1), respectively. To be specific, for a seller's ID ID_j^s and a sell-bid $s_j^m = (X_j^m, \theta_j^m)$, three random numbers r_1, r_2, r_3 are chosen randomly in $\mathbb{Z}_{2^\ell}$. Then, the seller sets $\langle ID_j^s\rangle_0 = r_1$ and $\langle s_j^m\rangle_0 = (\langle X_j^m\rangle_0, \langle\theta_j^m\rangle_0)$ where $\langle X_j^m\rangle_0 = r_2$, $\langle\theta_j^m\rangle_0 = r_3$, and computes $\langle ID_j^s\rangle_1 = ID_j^s - r_1$, $\langle s_j^m\rangle_1 = (\langle X_j^m\rangle_0, \langle\theta_j^m\rangle_1)$ where $\langle X_j^m\rangle_1 = X_j^m - r_2$, $\langle\theta_j^m\rangle_1 = \theta_j^m - r_3$. After that, the shares $(\langle ID_j^s\rangle_0, \langle s_j^m\rangle_0)$ are submitted to P_0 and the other shares $(\langle ID_j^s\rangle_1, \langle s_j^m\rangle_1)$ are submitted to P_1. In a similar fashion, the buyers send the secret shares of the IDs ID_i^b and buy-bids $b_i = (x_i, c_i)(1 \leq i \leq N_b)$ to P_0 and P_1, respectively. So P_0 and P_1 receive the following secret-shared bids from the buyers and sellers:

Buyers: $(\langle ID_i^b\rangle, \langle b_i\rangle), i = 1, \ldots, N_b$
Sellers: $(\langle ID_j^s\rangle, \langle s_j^m\rangle), j = 1, \ldots, N_s$ and $m = 1, \ldots, M$.

C. SECURE CLOUD VM AUCTION

The security requirements imply that the cloud VM auction protocol ought to be executed in a data-independent manner while the underlying insecure algorithm is executed depending on the layout of the input data. So we modify the underlying auction scheme to be an input-independent version by introducing a set of binary flags, and then construct a secure solution by the proposed secure sub-protocols. In the auction phase (the current time $t < t_3$), with the help of the auction agent P_2, the auctioneer P_0 and the auction agent P_1 execute the secure cloud VM auction protocol to implement

Algorithm 4 Secure Cloud VM Auction (SCA) Protocol

```
Input: The secret-shared bids from the buyers and sellers.
Output: The secret-shared winning bidders' IDs and final
        prices, and the number of the sold VMs.
 1: for $1 \leq i \leq N_b$ do
 2:     for $1 \leq m \leq M$ do
 3:         $P_0$ computes $\langle U_i^m \rangle_0 = \langle x_i^m \rangle_1 \cdot \mu_m$,
            $P_1$ computes $\langle U_i^m \rangle_1 = \langle x_i^m \rangle_1$.
 4:     end for
 5:     $P_0$ and $P_1$ compute $\langle V_i \rangle = \langle U_i^1 \rangle + \langle U_i^2 \rangle + \cdots + \langle U_i^K \rangle$ and $\langle \phi_i \rangle = \text{SecDiv}(\langle c_i \rangle, \langle V \rangle)$.
 6: end for
 7: $P_0$ and $P_1$ Bind $\langle V_i \rangle$, $\langle \phi_i \rangle$ with $(\langle ID_i^b \rangle, \langle b_i \rangle)$ to get tuple
    array $\mathbb{B} = (\langle ID_i^b \rangle, \langle b_i \rangle, \langle U_i \rangle, \langle \phi_i \rangle)$, $i \in [1, N]$.
 8: $P_0$, $P_1$, and $P_2$ invoke SST protocol to sort $\mathbb{B}$ in
    non-ascending order according to $\phi_i$, then gain the
    ranked array $\langle \phi_{\alpha_1} \rangle, \cdots, \langle \phi_{\alpha_{N_b}} \rangle$, the buyer's ID $\langle ID_{\alpha_1}^b \rangle$,
    and the purchasing need $\langle x_{\alpha_1} \rangle$.
 9: $P_0$ and $P_1$ set $\mathbb{S} = (\langle ID_j^s \rangle, \langle s_j^m \rangle)$, $j \in [1, N_s]$, $m \in [1, M]$.
10: $P_0$, $P_1$, and $P_2$ invoke SST protocol to sort $\mathbb{S}$ in
    non-descending order according to $\theta_j^m$, then gain the
    ranked arrays $\langle \theta_{\beta_1}^m \rangle, \cdots, \langle \theta_{\beta_{N_s}}^m \rangle$, $m \in [1, M]$.
11: for $1 \leq m \leq M$ do
12:     $P_0$ computes $\langle \gamma_{\alpha_1}^m \rangle_0 \leftarrow \langle x_{\alpha_1}^m \rangle_0 \cdot \mu_m$,
        $P_1$ computes $\langle \gamma_{\alpha_1}^m \rangle_1 \leftarrow \langle x_{\alpha_1}^m \rangle_1$.
13: end for
14: $P_0$ and $P_1$ compute
    $\langle \Phi \rangle = \langle \phi_{\alpha_2} \rangle \cdot (\langle \gamma_{\alpha_1}^1 \rangle + \langle \gamma_{\alpha_1}^2 \rangle + \cdots + \langle \gamma_{\alpha_1}^M \rangle)$.
15: for $1 \leq j \leq N_s$ do
16:     $P_0$ and $P_1$ compute
        $\langle \delta_j \rangle = \langle x_{\alpha_1}^1 \rangle \cdot \langle \theta_{\beta_j}^1 \rangle + \cdots + \langle x_{\alpha_1}^M \rangle \cdot \langle \theta_{\beta_j}^M \rangle$.
17:     $P_0$, $P_1$ and $P_2$ compute $\langle \lambda_j \rangle$ = SCMP$(\langle \Phi \rangle, \langle \delta_j \rangle)$,
        $\langle ID_{\beta_j}^s \rangle = \langle \lambda_j \rangle \cdot \langle ID_{\beta_j}^s \rangle$, and
        $\langle \widehat{\theta}_{\beta_j}^m \rangle = \langle \lambda_j \rangle \cdot \langle \theta_{\beta_j}^m \rangle$, $m \in [1, M]$.
18:     $P_0$ and $P_1$ add $\langle ID_{\beta_j}^s \rangle$ into the list $\langle Sellers \rangle$.
19: end for
20: $P_0$ and $P_1$ add $\langle ID_{\alpha_1}^b \rangle$ into the list $\langle Buyers \rangle$.
21: for $1 \leq j \leq N_s - 1$ do
22:     $P_0$ and $P_1$ compute $\langle \eta_j \rangle \leftarrow \langle \lambda_j \rangle - \langle \lambda_{j+1} \rangle$.
23: end for
24: $P_0$ and $P_1$ set $\langle \eta_{N_s} \rangle = \langle \lambda_{N_s} \rangle$, $\langle \widehat{c}_{\alpha_1} \rangle = \langle \phi_{\alpha_2} \rangle \cdot \langle V_{\alpha_1} \rangle$,
    compute $\langle j^* \rangle = \langle \eta_1 \rangle \cdot \langle 1 \rangle + \cdots + \langle \eta_{N_s} \rangle \cdot \langle N_s \rangle$.
25: for $1 \leq m \leq M$ do
26:     $P_0$ and $P_1$ compute
        $\langle \widehat{X}^m \rangle = \text{SecDiv}(\langle x_{\alpha_1}^m \rangle, \langle j^* \rangle \ominus \langle 1 \rangle)$.
27: end for
28: $P_0$ and $P_1$ output $\langle Buyers \rangle$, $\langle Sellers \rangle$, $\langle Buyers \rangle$, $\langle \widehat{c}_{\alpha_1} \rangle$,
    $\langle \widehat{\theta}_{\beta_j}^m \rangle$, $\langle \widehat{X}^m \rangle$, $m \in [1, M]$, $j \in [1, N_s]$.
```

1) winners matching and 2) pricing and VMs allocation, as shown in Algorithm 4.

1) WINNERS MATCHING

P_0 and P_1 calculate the buy-bid density by inputting $\langle b_i \rangle = (\langle x_i \rangle, \langle c_i \rangle)$ with two looping statements according to Equation (1) (Line 1-6). Then, P_0 and P_1 bind $\langle V_i \rangle$, $\langle \phi_i \rangle$ with $(\langle ID_i^b \rangle, \langle b_i \rangle)$ to get the multi-dimensional array $\mathbb{B} = [(\langle ID_i^b \rangle, \langle b_i \rangle, \langle V_i \rangle, \langle \phi_i \rangle)]$, $i \in [1, N_b]$. Then, with the help of P_2, P_0 and P_1 invoke SST protocol with inputting $\mathbb{B}$ to obtain a ranked $\mathbb{B}_\alpha$ according to ϕ_i in non-ascending order. After sorting, P_0 and P_1 obtain a ranked bid-density array $\langle \phi_{\alpha_1} \rangle, \cdots, \langle \phi_{\alpha_{N_b}} \rangle$, the buyer's ID $\langle ID_{\alpha_1}^b \rangle$ and the purchasing need $\langle x_{\alpha_1} \rangle$ corresponding to the largest bid density (Line 7 & 8). Next, by the same way, P_0, P_1, and P_2 invoke SST protocol to sort the arrays $\mathbb{S} = (\langle ID_j^s \rangle, \langle s_j^m \rangle)$, $j \in [1, N_s]$, $m \in [1, M]$ in non-descending order according to θ_j^m, then gain the ranked arrays $\langle \theta_{\beta_1}^m \rangle, \cdots, \langle \theta_{\beta_{N_s}}^m \rangle$, $m \in [1, M]$ (Line 9 & 10).

Afterward, P_0 and P_1 find the winning buyer and sellers by Equation (4) (Line 11-20). To obtain the critical index j^* without leaking the order of the bids, we introduce two arrays of binary flags λ_j and η_j with $j \in [1, N_s]$ to aid the secure computation. Let λ_j indicate whether j is less than or equal to the critical index j^* ($\lambda_j = 1$) or not ($\lambda_j = 0$). Let η_j indicate whether j is equal to the critical index j^* ($\eta_j = 1$) or not ($\eta_j = 0$). The numerical relationship between the two flag arrays is as follows:

$$\left| \begin{array}{ccccccc} j: & 1 & \cdots & j^*-1 & j^* & j^*+1 & \cdots & N_s \\ \lambda_j: & 1 & \cdots & 1 & 1 & 0 & \cdots & 0 \\ \eta_j: & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{array} \right|$$

According to Equation (4), the flag λ_j can be computed as $\lambda_j = (\phi_{\alpha_2} \cdot x_{\alpha_1} \cdot \mu \geq \sum_{m=1}^{M} x_{\alpha_1}^m \cdot \theta_{\beta_j}^m)$, $j \in [1, N_s]$ (Line 15-19). In particular, we invoke SCMP protocol to execute the comparison operation over the secret-shared data. SCMP protocol is implemented similarly to SC protocol shown in Algorithm 1, and the only change is to make P_0 and P_1 output the secret-shared flag $\langle f \rangle$. From the above pattern, we also compute the flag η_j by $\eta_j = \lambda_j - \lambda_{j+1}, j \in [1, N_s - 1]$ (Line 21-24). Then, P_0 and P_1 compute the ID of winning sellers by $ID_{\beta_j}^s = ID_{\beta_j}^s \cdot \lambda_j$. Finally, P_0 and P_1 add $\langle ID_{\beta_j}^s \rangle$ into the winning-seller list $\langle Sellers \rangle$, and add $\langle ID_{\alpha_1}^b \rangle$ into the winning-buyer list $\langle buyers \rangle$.

2) PRICING AND VMs ALLOCATION

To compute the wining bidders' final prices and the number of the sold VMs, P_0 and P_1 compute the index j^* by $j^* = \sum_{j=1}^{N_s} j \cdot \eta_j$. Then, P_0 and P_1 obtain the final price for winning buyer $\widehat{c}_{\alpha_1}$ (Line 24), the income of type-m VM for the winning seller $\widehat{\theta}_{\beta_j}^m$ (Line 17) and the amount of sold type-m VMs $\widehat{X}_m$ (Line 25-27) based on Equation (5)-(7).

D. FAIR TRADING

Algorithm 5 shows the main steps in the phase of fair trading (the current time $t \geq t_3$). After executing the privacy-preserving auction protocol, the smart contract SC_{ft} receives the auction results $Buyers$, $Sellers$, $\widehat{c}_{\alpha_1}$, $\widehat{\theta}_{\beta_j}^m$, $\widehat{X}^m$, $m \in [1, M]$, $j \in [1, N_s]$. SC_{ft} first refunds the deposits to all failed sellers and the failed buyers. Then, each winning buyer i pays $cost_{B_i}$ to SC_{ft}. SC_{ft} verifies that whether $cost_{B_i}$ is greater than the payment of the winning buyer $\widehat{c}_{\alpha_1}$. If $cost_{B_i} \geq \widehat{c}_{\alpha_1}$, the deal

Algorithm 5 Fair Trading

```
1:  if $t_3 \le t < t_4$ then
2:    The smart contract $SC_{ft}$ receives the auction results
      $Buyers, Sellers, \widehat{c}_{\alpha_1}, \widehat{\theta}^m_{\beta_j}, \widehat{X}^m, m \in [1, M], j \in [1, N_s]$.
3:    $SC_{ft}$ refunds the deposit $dpt_{S_j}$ and $dpt_{B_i}$ to the failed
      sellers and the failed buyers.
4:    $SC_{ft}$ computes the payment $cost_{S_j}$ of the winning
  seller
      should receive by $cost_{S_j} = \sum_{m=1}^{M} \widehat{X}^m \cdot \widehat{\theta}^m_{\beta_j}$.
5:    Winning buyer $i$ pays $cost_{B_i}$ to $SC_{ft}$.
6:    if $cost_{B_i} \ge \widehat{c}_{\alpha_1}$ then                % deal
  succeeds
7:      $SC_{ft}$ makes the ownership of required VM resources
        belongs to winning buyer $i$. $SC_{ft}$ transfers to the
  winning
        seller $cost_{S_j}$ as a reward.
8:      $SC_{ft}$ transfers to the auctioneer $cost_{B_i} - cost_{S_j}$
        as a reward.
9:    else                                                % deal fails
10:     $SC_{ft}$ transfers to the winning seller $dpt_{B_i}$ as
        financial penalties.
11:   end if
12: end if
13: if $t \ge t_4$ then                                   % timeout
14:   $SC_{ft}$ transfers to the winning seller $dpt_{B_i}$ from the
      winning buyers without paying on time. $SC_{ft}$ refunds
      $dpt_{B_i}$ to the winning seller with paying on time.
15:   $SC_{ft}$ gets commission from all winning sellers'
  deposits,
      and refunds the left deposits to them.
16: end if
```

succeeds, such that SC_{ft} makes the ownership of required VM resources belong to winning buyers, and transfers the rewards to the winning sellers and the auctioneer. If the deal fails, SC_{ft} transfers the deposit of the winning buyer (dpt_{B_i}) to the winning seller as financial penalties. When the current time $t \ge t_4$, SC_{ft} transfers the deposit of the winning buyer without paying on time to the winning seller as financial penalties, and refunds dpt_{B_i} to the winning seller with paying on time. Finally, SC_{ft} gets commission from all winning sellers' deposits, and refunds the left deposits to them.

VII. THEORETICAL ANALYSIS

A. SECURITY ANALYSIS

We analyze the security of SF-DAC with the security goals described in Section III-B. As can be seen from our auction scheme, the sellers and buyers do not take part in the auction computations, such that what they can get from the auction is only the auction outcome. Thus, we mainly prove that the auctioneer and the agents cannot get anything about the sensitive inputs except for the auction results.

We state that the cryptographic primitives (described in Section IV-B) involved in our auction protocol are secure under the semi-honest adversaries model [40], whose formal security proofs of them can be found in [46]. Based on the above definition and security properties, Theorem 1 and Theorem 2 give the security proofs of SC protocol and SST protocol. Then, combining with the composition theory [40], we prove the security of the secure cloud VM auction (SCA) protocol in SF-DAC against semi-honest adversaries, which is stated as Theorem 3.

Theorem 1: As long as secure computation primitives are secure under the semi-honest adversary model, SC protocol (as shown in Algorithm 1) is secure under the semi-honest adversary model.

Proof: To prove the security of SC protocol, we construct simulators in three distinct cases depending on which party is corrupted. Case 1: P_0 is corrupted by an adversary, we construct a simulator S_0 to simulate P_0's view. For P_0 receives $\langle f \rangle_0$, S_0 randomly selects a value l_0 from $\mathbb{Z}_{2^\ell}$. Since l_0 and $\langle f \rangle_0$ are all uniformly random chosen from $\mathbb{Z}_{2^\ell}$, any PPT adversary cannot distinguish l_0 from the interactive message $\langle f \rangle_0$. Case 2: P_1 is corrupted by an adversary, we can construct a simulator S_1 as the same as S_0, because SC protocol is symmetric for two parties. Case 2: P_2 is corrupted by an adversary, we construct a simulator S_2 to simulate P_2's view. For P_2 receives $\langle t_2 \rangle_0$ and $\langle t_2 \rangle_1$, S_2 randomly select two values l_1 and l_2 from $\mathbb{Z}_{2^\ell}$. Since $\langle t_2 \rangle_0$ and $\langle t_2 \rangle_1$ are the secret shares of $t_1 r$ over $\mathbb{Z}_{2^\ell}$, any PPT adversary cannot distinguish them from l_1 and l_2 due to the security of the additive secret sharing. Putting the above results together, we can claim that SC protocol is secure under the semi-honest adversary model.

Theorem 2: As long as SC protocol is secure against semi-honest adversaries, SST protocol (as shown in Algorithm 2) is secure under the semi-honest adversary model.

Proof: SST protocol makes $O(n^2)$ calls to SecSC protocol. Hence, for the interactive messages in these calls, the simulators for SST protocol can be trivially constructed by calling corresponding simulators of SC protocol. Except for the above interactive messages, there are some transmitting secret-shared matrices $\langle W \rangle_0$, $\langle A \rangle_0$, $\langle B \rangle_0$, $\langle C \rangle_0$ (resp. $\langle W \rangle_1$, $\langle A \rangle_1$, $\langle B \rangle_1$, $\langle C \rangle_1$) between P_0 and P_2 (resp. P_1 and P_2) in SST protocol. Since the matrices W, A, B, C are all generated randomly, the simulators can generate random matrices with the same size to simulate these interactive messages. The transmitting secret-shared matrices $\langle E \rangle$ and $\langle F \rangle$ are masked by the random matrices $\langle A \rangle$ and $\langle B \rangle$, respectively. The simulators can also generate random matrices to simulate these messages. Therefore, any PPT adversary cannot distinguish the simulator's views from these interactive messages. We can claim that SST protocol is secure under the semi-honest adversary model.

Theorem 3: As long as the secure computation primitives and SST protocol are secure against semi-honest adversaries, SCA protocol in SF-DAC (as shown in Algorithm 4) is secure under the semi-honest adversary model.

Proof: The exchanges data in SCA protocol consist of the interactive messages while executing the secure arithmetic operations and SST protocol. The secure arithmetic operations over the secret shares and SST protocol are proved to be secure against semi-honest adversaries, so any PPT adversary cannot distinguish the simulator's views from the interactive messages. According to the composition theory [40], we can declare that SCA protocol is secure against semi-honest adversaries as all secure computation steps are composed sequentially.

B. TRADE FAIRNESS ANALYSIS

In this subsection, we explain how SF-DAC guarantees trade fairness. Our system requires the sellers to registers their VM resources with the smart contract and to send deposits to the smart contract before the auction. That is, the ownerships of these resources are transferred to the smart contract temporarily, which can prevent a seller from dropping out of the auction without financial penalties, or getting fees without providing requested VM resources. SF-DAC also requires the buyers to send deposits to the smart contract before the auction, such that a buyer cannot drop out of the auction without financial penalties or obtain VM resources without paying fees. Since the rewards of the auctioneer and the agents are from the winner sellers' deposits, the deliveries of the corresponding rewards can be guaranteed. In conclusion, SF-DAC is able to guarantee trade fairness.

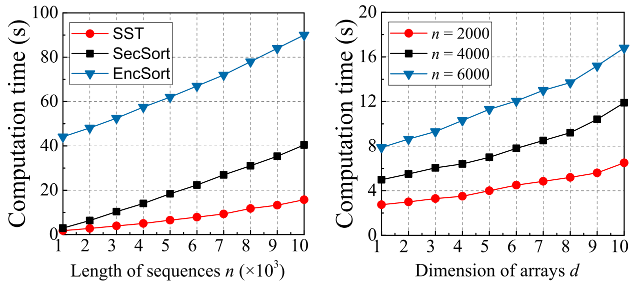
VIII. EXPERIMENTAL EVALUATION

A. EXPERIMENT SETUP

We implemented SF-DAC by using C++ and Solidity. Specifically, all secure protocols are constructed by C++, and the smart contracts are constructed by Solidity 0.4.24. The implementations of all secure protocols are multi-threaded. We set the bit length for an integral part $\ell = 32$ and the bit length for the fractional part $\ell_F = 16$ to achieve the same level of accuracy as the underlying auction scheme. We use three PCs with a 3.60-GHz Intel i7-4790 CPU and 8 GB of memory to act as the auctioneer and the two auction agents. The communication bandwidth among these machines for the LAN setting is set to 1 GB/s. In our experiment, we assume that each seller holds 1000 VMs in five types of VMs ($M = 5$). To implement the smart contracts, we build an Ethereum test network on a local server and use multiple EVM to set up Ethereum nodes. Since a small number of transactions and blocks are generated during the auction, we do not consider the impact of the world state on the invocation of the smart contracts.

B. MICROBENCHMARKS OF BUILDING BLOCKS

We evaluate the performance of all secure arithmetic operations over secret-shared data, SC Protocol and SST Protocol. Table 3 reports the benchmarking result (including computation time and communication cost) of the above building blocks. Specifically, the secure addition only takes 1 μs, and the secure multiplication takes 15 μs. Although the secure division takes about 1.5 ms, this operation is called in our auction protocol only once. Thanks to the assistant of a



(a) Computation time for different sizes of the arrays (b) Computation time for different dimensions of the arrays

FIGURE 3. Comparison of the computation time between SST protocol and prior works.

TABLE 3. Benchmarking results of different secure sub-protocols on secret-shared data.

Sub-protocols	Comp. time (in μs)	Comm. cost (in bytes)
Secure addition	1	0
Secure multiplication	15	16
Secure division	1536	1448
SC protocol	34	32

third party, SC protocol takes 34 μs to achieve the compare and swap. In short, these operations can work at extremely high efficiency, and establish the foundation for an efficient auction framework.

Recall that secure sorting takes up the bulk of the entire auction time. We plot the computation time of SST protocol for different sizes and different dimensions of the arrays in Figure 3(a) and Figure 3(b), respectively. As a point of reference, we also include the corresponding results of the prior works SecSort [6] and EncSort [47], which are under the two-party-computation setting. Note that, SecSort and EncSort protocols are designed in the offline-online mode, we only list the online computation time for them in Figure 3(a). For a varying size of arrays, the computation performance of our work is superior to that of SecSort and EncSort. In particular, when the size $n = 10000$, the computation time of SecSort and EncSort is 40s and 91s, respectively. By comparison, the computation time of our protocol is 15.7s, which improves performance by 60% and 82% percent, respectively. The underlying reason is that our method is 3PC-based protocol and does not involve any time-consuming operations such as garbled circuits [5] or homomorphic encryptions [48], while SecSort invokes a large number of garbled circuits and EncSort mainly depends on the homomorphic encryptions. By contrast, SST protocol is able to finish all secure operations by using the lightweight additive secret sharing, which benefits from the help of the third party.

C. PERFORMANCE OF SF-DAC FRAMEWORK

1) PERFORMANCE OF SECURE CLOUD VM AUCTION

To demonstrate the performance superiority of SF-DAC framework, we compare the secure cloud VM auction (SCA) protocol (as shown in Algorithm 4) with the advanced secure auction schemes including HE-based auction [27], PS-TAHES [11], PROST [29], and PDAM [6]. Recall that the

TABLE 4. Computation time (in second) and communication cost (in MB) of different auction schemes (the number of buyers N_b = 200 and the number of sellers N_s = 20).

Solutions	Comp. time (in second)		Comm. cost (in MB)	
	Online	Total	Online	Total
HE-based [27]	114	114	148	148
PS-TAHES [11]	2.51	145	2.86	128
PROST [29]	2.85	161	3.12	136
PDAM [6]	2.17	136	1.96	121
GC-based [49]	4.11	125	8.24	254
Our work	**1.97**	**1.97**	**1.12**	**1.12**

HE-based auction performs secure double auction based on homomorphic encryption. PS-TAHES and PROST execute the secure auction protocol by combining with homomorphic encryption and garbled circuits (GC). PDAM alternately uses additive secret sharing, garbled circuits, and homomorphic encryptions to achieve secure auction. In addition, we also make a comparison with the GC-based solution by using a state-of-the-art GC framework EMP-toolkit [49]. Note that, although PS-TAHES and PROST are designed for spectrum auctions, we can slightly modify the original protocols to make them achieve the same functions. In this experiment, we fix the number of sellers $N_s = 20$ and the number of buyers $N_b = 200$.

Table 4 shows the computation time and the communication cost for secure cloud VM auction. Compared with these works, the results in Table 4 demonstrate that SF-DAC achieves an order of magnitude speedup in both total computation and communication costs. In particular, compared with HE-based auction, PS-TAHES, PROST, PDAM, and GC-based auction, SF-DAC attains the shortest total computation time and reduces it by 57×, 73×, 81×, 69× and 63×, respectively. Furthermore, SF-DAC incurs 132×, 114×, 121×, 108× and 226× lower total communication cost than these compared works. Next, we analyze the reasons for the above results. Secure auction schemes with a single technique (e.g., homomorphic encryption and garbled circuits) incur a longer computation time in general. HE-based auction involves expensive cryptographic primitives and does not optimize it to offline-online mode, which both result in considerable overheads. GC-based auction needs to execute large-scale circuits for the secure auction, meaning that it is more computationally intensive. Prior arts, including PS-TAHES, PROST, and PDAM, design mixed protocols by multiple techniques to perform secure auction. And these works optimize their 2PC protocols into the offline-online mode so as to improve the online performance, but there are many time-consuming and high-traffic operations required for generating randomness and assistant parameters during the offline stage. By contrast, SF-DAC uses the lightweight technique (i.e., additive secret sharing) under 3PC model to vastly reduce the computation time for relevant randomness and assistant parameters. In addition, SF-DAC does not have to pre-generate assistant parameters in the offline stage due to the assistant of a third party. Hence, SF-DAC outperforms the above works in both the computation time and the communication cost for the secure auction.

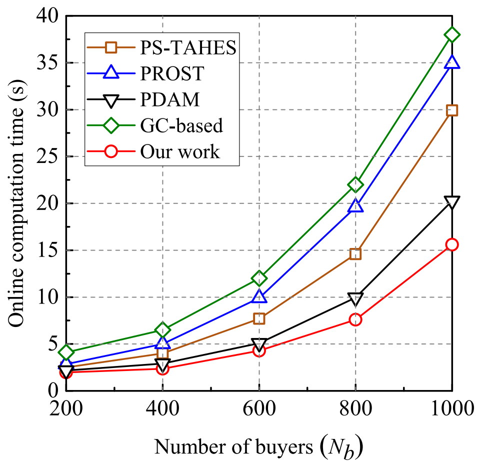


FIGURE 4. Online computation time of cloud auction for different the number of buyers (the number of sellers N_s = 20).

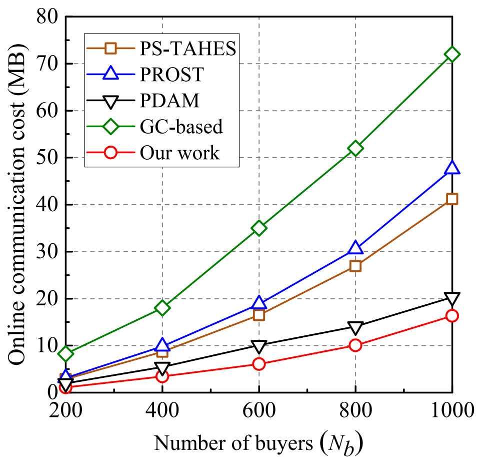


FIGURE 5. Online communication cost of cloud auction for different the number of buyers (the number of sellers N_s = 20).

In the following experiments, we further show the online performance for different secure auctions. The results in Table 4 show that the online performance of HE-based auction is considerably lower than that of other works, and HE-based auction is not further discussed. Figure 4 shows the online computation time of secure auction protocols with different numbers of buyers (N_b). We demonstrate that the online computation time of SF-DAC are $1.2\times - 1.9\times$, $1.4\times - 2.5\times$, $1.1\times - 1.3\times$ and $2.1\times - 2.9\times$ faster than PS-TAHES, PROST, PDAM and GC-based auction, respectively. Due to the introduction of the third party, our auction protocol can achieve all the secure operations over secret-shared data with few interactions. In contrast to our scheme, during the online stage, PS-TAHES, PROST, and PDAM still involve secure operations based on the homomorphic encryptions or garbled circuits, while GC-based auction needs to execute the complex garbled circuits. Figure 5 shows the online communication overhead of secure auction protocols with different numbers of buyers (N_b). Similarly, SF-DAC also offers significant savings over prior works in terms of online communication costs. Benefited from the third party,

TABLE 5. Running time (in second) of the smart contracts.

Number of Ethereum nodes		5	10	15	20	25
Running time	SC_{init}	1.62	1.77	1.81	2.02	2.27
	SC_{ft}	2.04	2.21	2.29	2.41	2.50

the expensive cryptographic primitives can be avoided in our protocol. However, other solutions inevitably invoke homomorphic encryptions or garbled circuits, which lead to a lot of communication costs.

2) PERFORMANCE OF THE SMART CONTRACTS

SF-DAC framework uses two smart contracts SC_{init} (for system initialization) and SC_{ft} (for trading) to guarantee trade fairness. To show the performance of these smart contracts, we list the exact values of running time in Table 5. We can find that the running time of both smart contracts is linearly related to the number of Ethereum nodes and increases slowly. Specifically, SF-DAC framework always takes less time to execute these contracts, which is typically less than 2.5s. Overall, we can implement fair trading efficiently by smart contracts.

IX. CONCLUSION

In this paper, we proposed a secure and fair double auction framework SF-DAC in two-sided cloud markets, which achieves bid-privacy protection and trade fairness. At the core of SF-DAC, we design a set of efficient 3PC protocols based on additive secret sharing to achieve security and efficiency simultaneously. We also use smart contracts to avoid trade disavowing and to ensure trade fairness. Formal security analysis demonstrates that the protocols in SF-DAC are secure under the semi-honest adversary model. Finally, the experimental results demonstrate that SF-DAC allows us to achieve an order of magnitude reduction in computation time and communication cost compared to the state-of-the-art prior works. As for future work, we plan to extend our framework to defend against malicious attackers.

REFERENCES

[1] L. Lu, J. Yu, Y. Zhu, and M. Li, "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided cloud markets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 720–733, Apr. 2018.

[2] J. Li, Y. Zhu, J. Yu, C. Long, G. Xue, and S. Qian, "Online auction for IaaS clouds: Towards elastic user demands and weighted heterogeneous VMs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2075–2089, Sep. 2018.

[3] Z. Chen, L. Chen, L. Huang, and H. Zhong, "On privacy-preserving cloud auction," in *Proc. IEEE 35th Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2016, pp. 279–288.

[4] C. Hazay and M. Venkitasubramaniam, "On the power of secure two-party computation," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2016, pp. 397–429.

[5] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *Proc. Int. Colloq. Automata, Lang., Program.* Berlin, Germany: Springer, 2008, pp. 486–498.

[6] K. Cheng, Y. Shen, Y. Zhang, X. Zhu, L. Wang, and H. Zhong, "Towards efficient privacy-preserving auction mechanism for two-sided cloud markets," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[7] Z. Chen, W. Ding, Y. Xu, M. Tian, and H. Zhong, "Fair auctioning and trading framework for cloud virtual machines based on blockchain," *Comput. Commun.*, vol. 171, pp. 89–98, Apr. 2021.

[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[9] S. G. Choi, J. Katz, A. J. Malozemoff, and V. Zikas, "Efficient three-party computation from cut-and-choose," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2014, pp. 513–530.

[10] K. Cheng, L. Wang, Y. Shen, Y. Liu, Y. Wang, and L. Zheng, "A lightweight auction framework for spectrum allocation with strong security guarantees," in *Proc. INFOCOM IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1708–1717.

[11] Q. Wang, J. Huang, Y. Chen, X. Tian, and Q. Zhang, "Privacy-preserving and truthful double auction for heterogeneous spectrum," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 848–861, Apr. 2019.

[12] M. Ostrovsky and M. Schwarz, "Reserve prices in Internet advertising auctions: A field experiment," in *Proc. 12th ACM Conf. Electron. Commerce*, 2011, pp. 59–60.

[13] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2060–2073, Aug. 2016.

[14] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 78–89, Jan. 2018.

[15] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2348–2362, Aug. 2016.

[16] B. Zheng, L. Pan, S. Liu, and L. Wang, "An online mechanism for purchasing IaaS instances and scheduling pleasingly parallel jobs in cloud computing environments," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 35–45.

[17] S. Parida, B. Pati, S. C. Nayak, and C. R. Panigrahi, "Offer based auction mechanism for virtual machine allocation in cloud environment," in *Advanced Computing and Intelligent Engineering*. Berlin, Germany: Springer, 2020, pp. 339–351.

[18] S. Li, J. Huang, and B. Cheng, "A price-incentive resource auction mechanism balancing the interests between users and cloud service provider," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 2030–2045, Jun. 2021.

[19] H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Virtual machine trading in a federation of clouds: Individual profit and social welfare maximization," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1827–1840, Jun. 2016.

[20] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Inf. Sci.*, vol. 357, pp. 201–216, Aug. 2016.

[21] R. Singhal and A. Singhal, "A feedback-based combinatorial fair economical double auction resource allocation model for cloud computing," *Future Gener. Comput. Syst.*, vol. 115, pp. 780–797, Feb. 2021.

[22] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Trans. Services Comput.*, early access, Mar. 4, 2019, doi: 10.1109/TSC.2019.2902549.

[23] B. Edelman, M. Ostrovsky, and M. Schwarz, "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords," *Amer. Econ. Rev.*, vol. 97, no. 1, pp. 242–259, Mar. 2007.

[24] J. Murillo, B. López, V. Muñoz, and D. Busquets, "Fairness in recurrent auctions with competing markets and supply fluctuations," *Comput. Intell.*, vol. 28, no. 1, pp. 24–50, 2012.

[25] G. Baranwal and D. P. Vidyarthi, "A fair multi-attribute combinatorial double auction model for resource allocation in cloud computing," *J. Syst. Softw.*, vol. 108, pp. 60–76, Oct. 2015.

[26] L. Liu, M. Du, and X. Ma, "Blockchain-based fair and secure electronic double auction protocol," *IEEE Intell. Syst.*, vol. 35, no. 3, pp. 31–40, May 2020.

[27] Y. Xu, Z. Chen, and H. Zhong, "Privacy-preserving double auction mechanism based on homomorphic encryption and sorting networks," 2019, *arXiv:1909.07637*. [Online]. Available: http://arxiv.org/abs/1909.07637

[28] Z. Chen, L. Huang, and L. Chen, "ITSEC: An information-theoretically secure framework for truthful spectrum auctions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2065–2073.

[29] Q. Wang, J. Huang, Y. Chen, C. Wang, F. Xiao, and X. Luo, "PROST: Privacy-preserving and truthful online double auction for spectrum allocation," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 374–386, Feb. 2019.

[30] Y. Chen, X. Tian, Q. Wang, J. Jiang, M. Li, and Q. Zhang, "SAFE: A general secure and fair auction framework for wireless markets with privacy preservation," *IEEE Trans. Dependable Secure Comput.*, 2020.

[31] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, "Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 211–227.

[32] J. Chen and S. Micali, "Algorand," 2016, *arXiv:1607.01341*. [Online]. Available: http://arxiv.org/abs/1607.01341

[33] V. Costan and S. Devadas, "Intel SGX explained," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[34] T. Alves and D. Felton, "TrustZone: Integrated hardware and software security," *ARM Inf. Quart.*, vol. 3, no. 4, pp. 18–24, 2004.

[35] L. Shen, X. Chen, J. Shi, Y. Dong, and B. Fang, "An efficient 3-party framework for privacy-preserving neural network inference," in *Proc. Eur. Symp. Res. Comput. Secur.* Berlin, Germany: Springer, 2020, pp. 419–439.

[36] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-Party secure computation for neural network training," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, Jul. 2019.

[37] S. Faber, S. Jarecki, S. Kentros, and B. Wei, "Three-party oram for secure computation," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2015, pp. 360–385.

[38] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur.*, May 2018, pp. 707–721.

[39] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, 2001, pp. 136–145.

[40] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[41] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 336–353.

[42] D. Demmler, T. Schneider, and M. Zohner, "ABY—A framework for efficient mixed-protocol secure two-party computation," in *Proc. NDSS*, 2015, pp. 1–15.

[43] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-Party secure inference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 325–342.

[44] O. Astrachan, "Bubble sort: An archaeological algorithmic analysis," *ACM Sigcse Bull.*, vol. 35, no. 1, pp. 1–5, 2003.

[45] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Commun. ACM*, vol. 23, no. 11, pp. 620–624, Nov. 1980.

[46] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2004, pp. 103–114.

[47] F. Baldimtsi and O. Ohrimenko, "Sorting and searching behind the curtain," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2015, pp. 127–146.

[48] P. Paillier *et al.*, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt*, vol. 99. Berlin, Germany: Springer, 1999, pp. 223–238.

[49] X. Wang, A. J. Malozemoff, and J. Katz. (2016). *EMP-Toolkit: Efficient MultiParty Computation Toolkit*. [Online]. Available: https://github.com/emp-toolkit

KE CHENG received the B.S. and M.S. degrees from Anhui University, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, China. His research interests include data security and privacy protection technology.

WEI TONG received the B.E. degree from the Jiangsu University of Science and Technology, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University, China. His research interest includes the performance optimization of blockchain.

LELE ZHENG received the B.S. degree from Xidian University, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include differential privacy and the IoT data security.

JIAXUAN FU received the B.S. degree from Xidian University, China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include the IoT data security and machine learning security.

XUTONG MU received the B.S. degree from the North University of China, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, China. His research interests include machine learning security and differential privacy.

YULONG SHEN (Member, IEEE) received the B.S. and M.S. degrees in computer science and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively. He is currently a Professor with the School of Computer Science and Technology, Xidian University, where he is also the Associate Director of the Shaanxi Key Laboratory of Network and System Security and a member of the State Key Laboratory of Integrated Services Networks. His research interests include wireless network security and cloud computing security. He has also served on the technical program committees of several international conferences, including ICEBE, INCoS, CIS, and SOWN.

• • •