Formalization and Verification of Data Auction Mechanism Based on Smart Contract Using CSP

Yingjia Du[1,*], Yuan Fei[2,*], Sini Chen[1], Huibiao Zhu[1]
[1]Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
[2]College of Information, Mechanical and Electrical Engineering,
Shanghai Normal University, Shanghai, China

***Abstract*—Nowadays, the utilization of online auction platforms is becoming increasingly prevalent. Online auction provides a common and practical way for global buyers to compete fairly. Nevertheless, the anonymous environment may bring collusion among entities with effects on results. Compared with traditional mechanisms which rely on third-party platforms, the data auction based on smart contract can create a decentralized environment to avoid the occurrence of collusion. Meanwhile, there exists few research on the verification of its reliability and safety which is worth investigating from the perspective of formal methods.**

In this paper, we apply Process Algebra CSP in modeling the data auction communicating system among five key entities. In addition, we use Process Analysis Toolkit (PAT) to realize the mechanism and verify five crucial properties, including deadlock freedom, data reachability, data correctness, anti-collusion capability and data security. The verification results indicate that the architecture of data auction based on smart contract can satisfy all the above requirements. Especially, the design of asymmetric encryption for the fundamental information ensures the non-occurrence of collusion in the auction. Additionally, the digital signature generated by private key attached to the message guarantees the safety of the interaction.

***Index Terms*—Data Auction Mechanism; Smart Contract; Process Algebra CSP; Modeling; Verification**

I. INTRODUCTION

In the era of information, the demand for high-quality data, resistant to conventional pricing methods [1] has brought enormous popularity and prosperous landscape in the field of data auction.

The online data auction mechanisms have been proliferating [2]. Despite that, existing mechanisms, which excessively rely on third-party platform, can lead to information leakage [3], challenges of distrust between entities and other issues urgently to be addressed.

Meanwhile, the blockchain technology has obtained growing attention [4] as a promising solution for secured and traceable record which can be applied in the auction process. Xiong et al. [5] proposed the decentralized anti-collusion data auction mechanism based on smart contract, which enables secured and immutable record of the process [6] and invokes automated condition evaluation program.

As demonstrated in Fig.1, the mechanism based on smart contract is decentralized in data exchange. The smart contract replaces the role of auctioneers and result announcement in the traditional system [7]. Data providers can upload data to the smart contract and potential buyers can make confidential bids. Once the auction is over, smart contract will make the result public to participants and the buyer with the highest bid will gain ownership of the auctioned data. Details of the transaction are recorded and traceable [8]. Additionally, anyone involved in the auction can access the results. Thus, security and truthfulness of the data auction mechanism based on smart contract are crucial and formal methods can provide a systematic and rigorous approach to analyzing it.

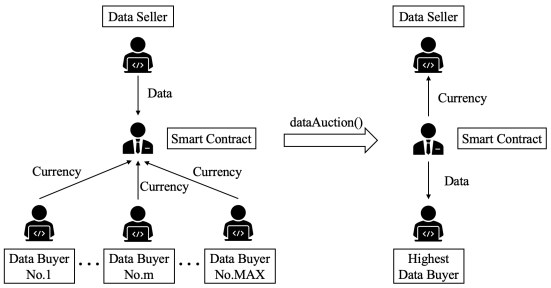

Fig. 1. Data Auction Mechanism Based on Smart Contract

In this paper, we formalize the model of the anti-collusion data auction mechanism [5] based on smart contract utilizing Communicating Sequential Process (CSP) [9]. Besides, with the aid of Process Analysis Toolkit (PAT) [10], we verify the following crucial properties, including *Deadlock Freedom*, *Data Reachability*, *Data Correctness*, *Anti-collusion Capability*. Meanwhile, we simulate the scenario with the presence of an intruder and establish a model to express its behavior of decrypting and faking messages. We can infer from the verification results that the mechanism is stable and reliable.

The remainder of the paper is organized as follows. We deliver a brief introduction to the architecture of the data auction and the basic knowledge of CSP in Section II. The subject of Section III is the formalized model of the system. Moreover, Section IV presents verification process and results. We discuss the conclusion and future work in Section V.

II. BACKGROUND

In this section, we start with an overview of the proposed data auction system based on smart contract by Xiong et al. [5]. We also give a brief introduction to process algebra CSP.

*Corresponding Authors. E-mail address: yuanfei@shnu.edu.cn (Y. Fei).
yingjiadu@stu.ecnu.edu.cn (Y. Du).

DOI reference number: 10.18293/SEKE2023-172

A. Data Auction Mechanism Architecture

As demonstrated in Fig.2, the framework of the mechanism based on smart contract is decentralized to improve the efficiency and is comprised of five entities as below.

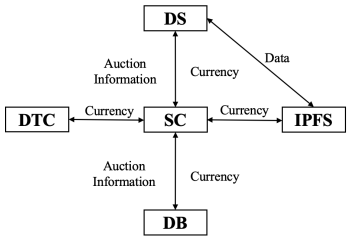

Fig. 2. The Data Auction Framework

- **Data Seller (DS):** It owns the data and the revenue.
- **Data Buyer (DB):** It intends to acquire auctioned data.
- **Data Trading Centre (DTC):** It is the institution where copies of transaction can be properly stored.
- **File System (FS):** It provides safe storage for the data.
- **Smart Contract (SC):** It plays a crucial role in recording the information and managing the transaction.

The data auction system based on smart contract adopts a single-round sealed-bid auction architecture. Only one data seller and multiple data buyers are allowed to take part in the same auction. The auction process can be divided into three steps: **Preparation**, **Process** and **Conclusion**, which are illustrated in Fig.3. The definitions of key variables and functions are given as follows.

- **Sealed Price** sp_i**:** It is the price proposed by the seller as the minimum transaction price and i stands for its id.
- **Sealed Bid** sb_j**:** The variable shows true bid of the buyer sealed in the process and j stands for the buyer's id.
- **Bid Mask** bm_j**:** It is required to be larger than the corresponding sb and is public to disguise the true bid.
- **paySDeposit():** The function records the information of the seller and deduct a deposit from its account.
- **payBDeposit():** It is activated when a buyer is willing to take part in the auction. Moreover, it depicts the behaviour of paying deposit to get the admission to the auction.
- **dataAuction():** It checks whether the transaction meets the condition $sp_i \leq sb_j \leq bm_j$.
- **payBid():** This function withdraws corresponding bid price from the account of the winning buyers.
- **verifyInfo():** It is used to check whether a certain buyer is on the winning buyers' list to authenticate its identity.

Next, we will present detailed steps of the data auction.

1. The Data Seller i sends the reserved price sp_i to the Smart Contract with some supplementary basic information.

2. The Data Seller informs the Data Trading Centre and Data Buyer of the auctioned data information.

3. The Data Buyer decides whether to participate in the current auction. If the answer is yes, it pays the deposit to the Smart Contract. Otherwise, it quits.

4. The Data Seller and Data Trading Centre receive the list of participants in the auction from the Smart Contract.

5. The qualified Data Buyer j submits a sealed bid sb_j, which is based on evaluation of the auctioned data to Smart Contract and broadcasts a bid mask bm_j, which is a disguise of the true bid.

6. The Smart Contract forwards the same bid information to the Data Trading Centre.

7. The Smart Contract executes the auction program and make the highest Data Buyer accessible to the auctioned data.

8. The Smart Contract broadcasts the list of winner(s) to the Data Seller, the Data Trading Centre and all Data Buyer.

9. The winning buyer pays for the bid to the Smart Contract.

10. The Smart Contract confirms that it has received the currency from the Data Buyer.

11. The Smart Contract generates a token randomly.

12. The Smart Contract provides the winning Data Buyer with a created token to download the auctioned data.

13. The Data Buyer shows its id for File System to validate.

14. The File System downloads the auctioned data by token.

15. Data is accessible to the winning Data Buyer.

16. The File System sends confirmation to the Data Trading Centre and Data Seller.

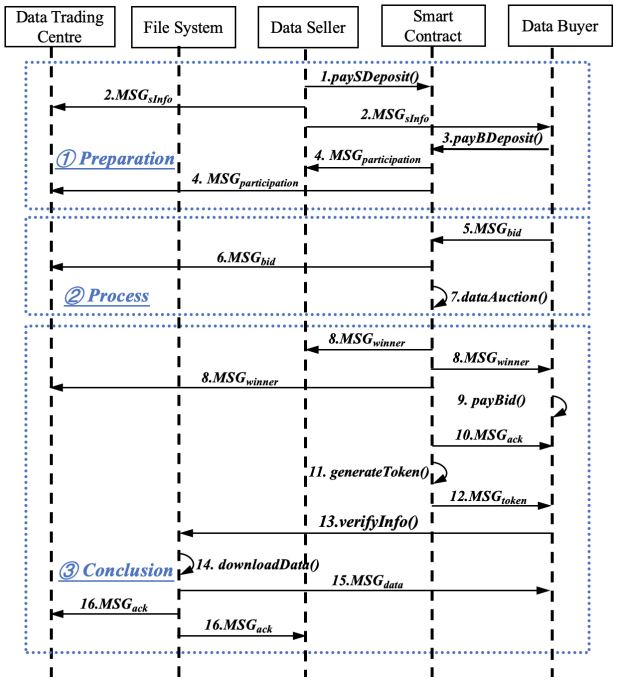

Fig. 3. The Data Auction Flow Chart (adapted from [5])

B. CSP

Communicating Sequential Processes (CSP) is a representative process algebra based on logic [9]. The definition of the syntax of CSP used in this paper is given as below.

$$P, Q ::= SKIP \mid c?u \rightarrow P \mid c!v \rightarrow P \mid P \Box Q \mid P||Q \mid P[|X|]Q \mid P \triangleleft B \triangleright Q$$

- $SKIP$ stands for that the process which does nothing, but terminates immediately.
- $c?u \rightarrow P$ denotes that the process receives a value through the channel c, assigns it to variable u and behaves as P.
- $c!v \rightarrow P$ indicates that the process sends message v through the channel c. Then it performs as process P.

- $P \square Q$ represents that the process behaves like either process P or Q, determined entirely by the environment.
- $P||Q$ refers to the concurrent execution of P and Q. Same events in a common alphabet require synchronization.
- $P[|X|]Q$ describes the concurrent execution of P and Q on the set of channels X.
- $P \triangleleft b \triangleright Q$ stands for the condition. When Boolean variable b is true, it will perform as P. Otherwise, it will behave like Q.

III. MODELING

In this section, we give the formalized model of the system of data auction [5] using CSP. We begin by the definition of sets, messages and channels. Then, we formalize the architecture described in Section II by modeling entities respectively.

A. Sets, Messages and Channels

The whole architecture is rather complicated. Thus, we divide the involved elements into sets as the basic components of the model according to their types and functionalities. We give an introduction to the sets as below.

- **Entity** involves Data Seller (**DS**), Data Buyer (**DB**), Data Trading Centre (**DTC**), File System (**FS**) and Smart Contract (**SC**) mentioned in Fig.2.
- **ID** defines serial number of **DB** and **DS**. $ID{=}BID \cup SID$.
- **Account** consists **DB** and **DS**'s funding. $Account = BAccount \cup SAccount$.
- **Price** includes sealed price *sp* of **DS**, sealed bid *sb* and bid mask *bm* of **DB**. $Price{=}SPrice \cup SBid \cup BMask$.
- **Key** contains public and private keys belonging to **Entity**. $Key = k_{pub} \cup k_{pri}$
  $k_{pub} =_{df} \{bk_{pub}, sk_{pub}, sck_{pub}, fk_{pub}, tk_{pub}\}$.
  $k_{pri} =_{df} \{bk_{pri}, sk_{pri}, sck_{pri}, fk_{pri}, tk_{pri}\}$.
- **Sig** is composed of the identification of **Entity**. $Sig =_{df} \{bsig, ssig, scsig, fsig, tsig\}$.
- **Par** is composed of **DB**'s participations.

In addition, **E** and **D** describe message encryption and decryption while **G** and **V** refer to digital signature generation and verification using k_{pub}, k_{pri} in blockchain.

- **E** (k_{pub}, msg) signifies k_{pub} is used to encrypt msg.
- **D** $(k_{pri}, E(k_{pub}, msg))$ shows that k_{pri} is able to decrypt msg encrypted by corresponding k_{pub}.
- **G** (k_{pri}, sig) denotes k_{pri} can be used for generating sig.
- **V** $(k_{pub}, E(k_{pri}, sig))$ shows that k_{pub} can be utilized to recognize identification sig generated by k_{pri}.

Moreover, we classify and abstract the messages transferred between entities as tuples to reduce redundancy.

$Tuple_{s1} = [sid.sname.stype.slink]$
$Tuple_{b1} = [bid.par]$ $Tuple_{b2} = [bid.baddress]$
$Tuple_{b3} = [bid.baddress.bm]$
$Tuple_{b4} = [bid.baddress.baccount.par]$
$\boldsymbol{MSG_{reqs}}{=}_{df}\{Msg_{reqs1}.Tuple_{s1}.G(k_1, sig),$
$Msg_{reqs2}.Tuple_{s1}.saccount.G(k_2{,}sig).E(k_3, sp)|$
$sid \in SID$, $sname \in Name$, $stype \in Type$,
$slink \in Link$,$sp \in Price$,$saccount \in SAccount$
$k_1, k_2 \in k_{pri}$, $sig \in Sig$,$k_3 \in k_{pub} sp \in Price\}$

e.g. $Msg_{reqs1} = sid.sname.stype.slink.G(k_{pri}, sig)$.
Similarly, we omit the remaining conversion process.

$\boldsymbol{MSG_{reqb}}{=}_{df}\{Msg_{reqb1}.Tuple_{b1}.G(k_1, sig),$
$Msg_{reqb2}.Tuple_{b3}.G(k_2, sig).E(k_3, sb),$
$Msg_{reqb3}.bid.G(k_4, sig),$
$Msg_{reqb4}.Tuple_{b4}.G(k_5, sig)|$
$bid \in BID$, $baccount \in BAccount$,$par \in Par$,
$baddress \in BAdress$, $bm, sb \in Price$,
$k_1, k_2, k_4, k_5 \in k_{pri}$, $sig \in Sig$,$k_3 \in k_{pub}\}$

$\boldsymbol{MSG_{reqc}}{=}_{df}\{Msg_{reqc}.G(k_1, sig), E(k_2, token)|$
$k_2 \in k_{pri}$, $sig \in Sig$, $k_2 \in k_{pub}\}$

$\boldsymbol{MSG_{reqd}}{=}_{df}\{Msg_{reqd}.data.G(k_1, sig)|$
$data{\in}Data$,$k_1{\in}k_{pri}$,$sig{\in}Sig\}$

$\boldsymbol{MSG_{check}}{=}_{df}\{Msg_{check}.Tuple_{b2}.G(k_1{,}sig).E(k_2, token)|$
$bid \in BID$, $k_1 \in k_{pri}$, $sig{\in}$ Sig,
$k_2 \in k_{pub}$, $baddress \in BAddress\}$

$\boldsymbol{MSG_{ack}} =_{df}\{Msg_{ack}.content \mid content \in ACK\}$
$\boldsymbol{MSG_{req}}{=}_{df} MSG_{reqs} \cup MSG_{reqb} \cup MSG_{reqc} \cup MSG_{reqd}$
$\boldsymbol{MSG} =_{df} MSG_{req} \cup MSG_{check} \cup MSG_{ack}$

Furthermore, we use channels for communication, as depicted in Fig.4 and Fig.5. Each channel in the model is bidirectional and can only transfer one message at a time.

- Channels between truthful components are called ***Com_Path:***
  ComDSS, ComDBS, ComDSF, ComDST, ComDBT
  ComTS, ComTF, ComDBF, ComDBDS

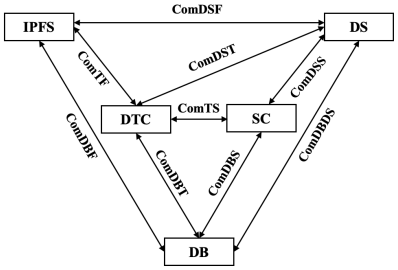

Fig. 4. Internal Channels of the Data Auction Architecture

- Channels involving interception and messages faking are named ***Intruder_Path:*** *InterceptDB, FakeSC*

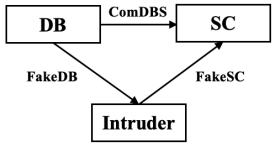

Fig. 5. Internal Channels of Faking Messages

B. Overall Modeling

We first formalize the model $System_0$ of the proposed mechanism without interception and forgery. We then take the situation in which intruder exists into account to establish a complete model. Due to the page limitation, we omit the modeling process of DataTradingCentre and FileSystem.

As illustrated in Fig.3, the system has three stages: **Preparation, Process and Conclusion** executed sequentially. Furthermore, in accordance with the base of data auction, the message transmission process needs to comply with rules of blockchain. The sender should encrypt message with k_{pri} and generate

a digital signature by $\mathbf{G}(k_{pri}, sig)$ as mentioned in Part A. Similarly, the receiver can call **verifySig** $(k_{pub}, E(k_{pri}, sig))$ to verify the sender's identity by k_{pub}.

$$System_0 =_{df} DataBuyer \;||\; DataSeller \;||\; FileSystem \;||\; SmartContract \;||\; TradingCentre$$
$$System =_{df} System_0[|Intruder\_Path|]Intruder$$

C. DataSeller Modeling

DataSeller is responsible for providing data information and proposing the sealed price sp as the reserved auction price which is not exposed to others until the auction ends.

DataSeller $=_{df}$

$$\left(\begin{array}{l} encryptInfo(sck_{pub}, sp) \rightarrow ComDSS!Msg_{reqs2} \rightarrow \\ paySDeposit(sid) \rightarrow ComDST!Msg_{reqs1} \rightarrow \\ ComDBDS!Msg_{reqs1} \rightarrow ComDSS?Msg_{reqb1} \rightarrow \\ verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow DataSeller \\ \lhd(sig == scsig) \rhd fail \rightarrow DataSeller \end{array}\right)$$
$$\lhd (state == Preparation) \rhd$$
$$\left(\begin{array}{l} ComDSS?Msg_{reqb3}.bid.G(sck_{pri}, sig) \rightarrow \\ verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow \\ (\; ComDSF?MSG_{ack} \rightarrow DataSeller \;) \\ \lhd(sig == scsig) \rhd fail \rightarrow DataSeller \end{array}\right)$$
$$\lhd (state == Conclusion) \rhd fail \rightarrow DataSeller$$

In the preparation stage, we define $encryptInfo(k, sp)$ to disguise the lowest price. $paySDeposit(sid)$ is designed to deduct the deposit from the DataSeller's account to avoid bought-in. We express the behaviour of delivering basic information Msg_{reqs1}, Msg_{reqs2} to SmartContract and DataTradingCentre respectively. $verifySig(k, sig)$ returns the decrypted sig using the corresponding public key. We compare the result with the original sig from the expected SmartContract to check the sender's identity. If the result is $true$, the system performs smoothly. Otherwise, the identity verification fails and the transaction terminates. The variable par is defined to record the participation of DataBuyer. In the conclusion step, DataSeller is informed of the winner from SmartContract and obtains the revenue as Step 8 in Fig.3. The specific contents are given as below.

Msg_{reqs1} =[$sid.sname.stype.slink.G(sk_{pri}, sig)$].
Msg_{reqs2} =[$Tuple_{s1}.saccount.G(sk_{pri}, sig).E(sck_{pub}, sp)$].
Msg_{reqb1} =[$bid.par.G(sck_{pri}, sig)$].

D. DataBuyer Modeling

DataBuyer is the crucial component of the data auction architecture. The set of buyers make decisions on whether to take turns to participate and make public bid mask bm to hide true bid sb. We firstly model the behaviour of a single DataBuyer and integrate multiple buyers as a whole. The variable DB stands for the number of participants.

In the first period, in Step 2 in Fig.3, DataBuyer receives Msg_{reqs1} from DataSeller and checks its signature to confirm that the message is from the specified sender. DataBuyer decides individual participation which is defined by the variable par. If the value equals to $true$, DataBuyer shows its willingness to make bid on the data. Then, it pays for the required deposit. In the second step, DataBuyer takes part in broadcasting bm and applies $encryptInfo(k, sb)$ in disguising sb from entities except for the SmartContract in Step 5. Finally, it gets the result of the data auction in Msg_{reqb3}. We define the comparison between DataBuyer's id is with the winner's as a condition statement. If it is true, winning buyer pays for the bid according to sb and receives the token to download data encrypted by its public key. $decryptInfo(k, token)$ serves to obtain the token generated by SmartContract and encrypts it to FileSystem to obtain the data in Steps 11-13. Winning DataBuyer receives Msg_{data} in the end if the result of verification is $true$.

Msg_{reqb21} =[$bid.address.bm.G(bk_{pri}, sig).E(sck_{pub}, sb)$].
Msg_{reqb4} =[$bid.address.account.par.G(bk_{pri}, sig)$].
Msg_{reqc} =[$G(sck_{pri}, sig).E(bk_{pub}, token)$].
Msg_{check} =[$bid.G(bk_{pri}, sig).E(fk_{pub}, token)$].
$msg_1 = E(bk_{pub}, token)$. $msg_2 = E(G(fk_{pri}, sig))$.

DataBuyer'$_i$ $=_{df}$

$$\left(\begin{array}{l} ComDBDS?Msg_{reqs1} \rightarrow \\ verifySig(sk_{pub}, G(sk_{pri}, sig)) \rightarrow \\ \left(\begin{array}{l} ComDBS!Msg_{reqb4} \rightarrow \\ payBDeposit(bid, baccount) \rightarrow \\ DataBuyer'_i \lhd (par == true) \rhd \\ fail \rightarrow DataBuyer'_i \end{array}\right) \\ \lhd(sig == ssig) \rhd fail \rightarrow DataBuyer'_i \end{array}\right)$$
$$\lhd (state == Preparation) \rhd$$
$$\left(\begin{array}{l} encryptInfo(sck_{pub}, sb) \rightarrow \\ ComDBS!Msg_{reqb21} \rightarrow DataBuyer'_i \end{array}\right)$$
$$\lhd (state == Process) \rhd$$
$$\left(\begin{array}{l} ComDBS?Msg_{reqb3}.bid.G(sck_{pri}, sig) \rightarrow \\ verifySig(sck_{pub}, G(sck_{pri}, sig)) \rightarrow \\ \left(\begin{array}{l} payBid() \rightarrow ComDSS?MSG_{ack} \rightarrow \\ ComDBS?Msg_{reqc} \rightarrow \\ verifySig(sk_{pub}, G(sk_{pri}, sig)) \rightarrow \\ \left(\begin{array}{l} decryptInfo(bk_{pri}, msg_1) \rightarrow \\ ComDBF!Msg_{check} \rightarrow \\ encryptInfo(fk_{pub}, token)) \rightarrow \\ ComDBF?Msg_{data}.data.msg_2 \rightarrow \\ verifySig(fk_{pub}, G(fk_{pri}, sig)) \rightarrow \\ DataBuyer'_i \lhd (sig == fsig) \rhd \\ fail \rightarrow DataBuyer'_i \end{array}\right) \\ \lhd(sig == ssig) \rhd fail \rightarrow DataBuyer'_i \end{array}\right) \\ \lhd(sig == scsig \wedge bid \in winner) \rhd \\ \quad Skip \rightarrow DataBuyer'_i \end{array}\right)$$
$$\lhd (state == Conclusion) \rhd fail \rightarrow DataBuyer'_i$$

DataBuyer' $=_{df}$ $|||i : \{0..DB-1\}$@***DataBuyer'***$_i$

Allowing for the existence of intruder, shown in Fig.5, we use symbol $\{|c|\}$ to denote the set of all interaction over the channel c. ***DataBuyer*** is defined via renaming as below.

DataBuyer $=_{df}$ ***DataBuyer'***[[
$ComDBS?\{|ComDBS|\} \leftarrow ComDBS?\{|ComDBS|\}$,
$ComDBS?\{|ComDBS|\} \leftarrow InterceptDB?\{|InterceptDB|\}$]]

E. SmartContract Modeling

SmartContract is the basis of the mechanism. It defines rules for other entities to follow, selects winning buyer(s) and broadcasts the list of winner(s).

At the beginning, as Step 1 in Fig.3 denoted, DataSeller sends Msg_{reqs2} to inform it of the overview of the auctioned data. SmartContract then conducts validation of the identity of sender in $verifySig(k, sig)$. It collects the deposit and shows the auction participation to DataSeller.

In the Process stage, it gets the specific data of bid in Msg_{reqb21}. It also forwards Msg_{reqb22} to DataTradingCentre. Then, it decrypts the message with its private key in $decryptInfo(k, msg)$. It calls the $dataAuction()$ program. Msg_{reqb22} =[$bid.address.bm.G(sck_{pri}$,$sig).E(sck_{pub}, sb)$].

$$\textbf{\textit{dataAuction}}(bid, baccount, bm, sp, sb, par) =_{df}$$
$$\left(\begin{array}{l} winner.append(bid) \\ \lhd(sb == high\_bid) \rhd withdrawBDeposit(bid) \\ winner.empty() \rightarrow winner.append(bid) \\ \lhd(sb > high\_bid) \rhd withdrawBDeposit(bid) \end{array}\right)$$
$$\lhd (condition) \rhd withdrawBDeposit(bid)$$
$$condition : (par == true \wedge sb \leq baccount \wedge sp \leq sb \wedge sb \leq bm)$$

The function ***dataAuction****(bid, baccount, bm, sp, sb, par)* checks whether the decrypted sb meets the rules and compares it with the recent highest bid $high\_bid$.

At last, in Step 8 in Fig.3, it broadcasts winning buyers' id. When it confirms the payment in Msg_{ack}, it randomly generates a token and encrypts it to ensure confidentiality.

$$\textbf{\textit{SmartContract'}} =_{df}$$
$$\left(\begin{array}{l} ComDSS?Msg_{reqs2} \rightarrow \\ verifySig(sk_{pub}, G(sk_{pri}, sig)) \rightarrow \\ \left(\begin{array}{l} ComDBS?Msg_{reqb4} \rightarrow \\ verifySig(bk_{pub}, G(bk_{pri}, sig)) \rightarrow \\ ComDSS!Msg_{reqb1} \rightarrow \\ ComTS!Msg_{reqb1} \rightarrow SmartContract' \\ \lhd(sig == bsig) \rhd fail \rightarrow SmartContract' \end{array}\right) \\ \lhd(sig == ssig) \rhd fail \rightarrow SmartContract' \end{array}\right)$$
$$\lhd (state == Preparation) \rhd$$
$$\left(\begin{array}{l} ComDBS?Msg_{reqb21} \rightarrow \\ verifySig(bk_{pub}, G(bk_{pri}, sig)) \rightarrow SmartContract' \\ \left(\begin{array}{l} ComTS!Msg_{reqb22} \rightarrow \\ decryptInfo(sck_{pri}, E(sck_{pub}, sp)) \\ decryptInfo(sck_{pri}, E(sck_{pub}, sb)) \\ dataAuction() \rightarrow SmartContract' \end{array}\right) \\ \lhd(sig == bsig) \rhd fail \rightarrow SmartContract' \end{array}\right)$$
$$\lhd (state == Process) \rhd$$
$$\left(\begin{array}{l} ComDSS!Msg_{reqb3}.bid.G(sck_{pri}, sig) \rightarrow \\ ComDBS!Msg_{reqb3}.bid.G(sck_{pri}, sig) \rightarrow \\ ComTS!Msg_{reqb3}.bid.G(sck_{pri}, sig) \rightarrow \\ ComDBS!Msg_{ack}.ack \rightarrow generateToken() \rightarrow \\ EncryptInfo(bk_{pub}, token) \rightarrow \\ (\ ComDBS!Msg_{reqc} \rightarrow SmartContract'\ ) \\ \lhd(bid \in winner) \rhd Skip \rightarrow SmartContract' \end{array}\right)$$
$$\lhd (state == Conclusion) \rhd fail \rightarrow SmartContract'$$
$$\textbf{\textit{SmartContract}} =_{df} SmartContract'[[$$
$$ComDBS!\{|ComDBS|\} \leftarrow ComDBS!\{|ComDBS|\},$$
$$ComDBS!\{|ComDBS|\} \leftarrow FakeSC!\{|FakeSC|\}]].$$

F. Intruder Modeling

The intruder causes unexpected accidents by eavesdropping and forging messages. We define a set **Fact** including facts it might obtain from the interaction between truthful entities.

$$\textbf{\textit{Fact}} =_{def} \{bid, address, bm, G(bk_{pri}, sig), E(sck_{pub}, sb) |$$
$$bid \in BID,\ baddress \in BAdress,\ bm, sb \in Price,$$
$$bk_{pri} \in k_{pri},\ sig \in Sig, sck_{pub} \in k_{pub}\} \cup Entity.$$

The intruder can deduce new facts from the accessible **Fact**. We describe the progress of deducing content of fact f from the origin fact F by the symbol $F \mapsto f$.

We can learn from the first rule that the intruder can decrypt message in asymmetric encryption.

The second rule shows that when set F is the subset of F', intruder can deduce f not only from F but also from F'.

$$(1)\{k^{pri}, E(k^{pub}, c)\} \mapsto c \qquad (2)F \mapsto f \wedge F \subseteq F' \Longrightarrow F' \mapsto f$$

We define the function $getInfo(msg)$ to imply the knowledge which the intruder intercepts from the process.

$$getInfo(msg_{reqb2}.bid.address.bm.G(bk_{pri}, sig).E(sck_{pub}, sb)$$
$$=_{def} \{bid, address, bm, G(bk_{pri}, sig), E(sck_{pub}, sb)\}$$

We define a channel $Deduce$ to deduce new facts:

$$Channel \quad Deduce : Fact.P(Fact)$$

Then, we give a formalized model of $Intruder'$ as follows.

$$Intruder'(F) =_{df}$$
$$\Box_{msg \in MSG} FakeDB?msg \rightarrow Intruder'(F \cup getInfo(msg))$$
$$\Box\Box_{msg \in MSG, getInfo(msg) \subseteq F} FakeSC!msg \rightarrow Intruder'(F)$$
$$\Box\Box_{new \in Fact, new \notin F, F \mapsto new} \rightarrow Deduce.new.F$$
$$\rightarrow \left(\begin{array}{l} \{Data\_Faking\_Success == true\} \rightarrow \\ Intruder'(F \cup \{new\}) \\ \lhd(new == sb) \rhd Intruder'(F \cup \{new\}) \end{array}\right)$$

The intruder adds $getInfo(msg)$ to its current knowledge when intercepting msg. It can decrypt msg from DataBuyer and change the content of msg to send it to SmartContract. If the msg cannot be identified from an unauthorized user and is received, it means that the intruder successfully modifies the content of the message and completes an attack. Then, we assign $true$ to $faking\_flag$. Now we give the model of $Intruder$. The parameter BK is its basic knowledge.

$$Intruder =_{df} Intruder'(BK),$$
$$BK =_{df} Entity \cup k_{pub} \cup \{sck_{pri}\}$$

IV. VERIFICATION AND RESULTS

In this section, we analyze and verify five properties of the constructed model in the model checker PAT [10] to evaluate the reliability of anti-collusion data auction mechanism.

A. Properties and Analysis

- **Deadlock Freedom:**
  Deadlock represents the situation when no entity can interact with each other normally in the system. PAT has a primitive to describe it.
  $$\#assert\ System()\ deadlockfree;$$
- **Data Reachability:**
  Reachability ensures that all valid data can be transferred between subjects in the process until the end. We assign the defined Boolean variable *data_reachable_flag* to *true* when the process obtains the expected target data.
  $$\#define\ Data\_Reachable\ reachable\_flag\ ==\ true;$$
  $$\#assert\ System()\ reaches\ Data\_Reachable;$$
- **Data Correctness:**
  Correctness describes the ability ensuring that the final displayed price is consistent with the price proposed by the buyer. It also indicates that the mechanism can draw the correct conclusion. We assign the defined Boolean

variable *data_correct_flag* to *true* when receiving the true result with no mistakes.

$$\#define\ Data\_Correct\ data\_correct\_flag\ ==\ true;$$
$$\#assert\ System()\ reaches\ Data\_Correct;$$

- **Anti-collusion Capability:**
  Collusion stands for the situation in which data buyers can observe others' prices to make adjustment to their own bid price to keep winning price as low as possible. We should avoid its occurrence to make the auction fair. In the supposed situation, the buyer intercepts the message in order to get sealed bid to influence the result. The variable *current_bid* represents the data transferred on the channel, which has the risk of being intercepted. Moreover, the variable *true_bid* stands for the true bid price from the corresponding buyer. We focus on comparing *current_bid* with *true_bid* to verify the property.

  $$\#define\ Anti\_Flag\ current\_bid\ !=\ true\_bid;$$
  $$\#assert\ System()\ reaches\ Anti\_Flag;$$

- **Data Security:**
  Security refers to the situation where the intruder cannot forge data successfully. Supposing there exists an intruder, we define the Boolean variable *faking_flag* to record the effect of attack. We assign it to *true* when the intruder successfully changes the value and has influence on the auction result to check whether the system is safe.

  $$\#define\ Data\_Faking\_Success\ faking\_flag\ ==\ true;$$
  $$\#assert\ System()\ reaches\ Data\_Faking\_Success;$$

B. Verification Results

We use PAT to simulate the behaviour of the constructed model. Then, we verify the above properties. The invalid result of *Data_Faking_Success* equals to the confirmation of *Data Security*. The result is illustrated in Fig. 6, implying that the system can always satisfy five properties, including *Deadlock Freedom*, *Data Reachability*, *Data Correctness*, *Anti-collusion Capability* and *Data Security*.

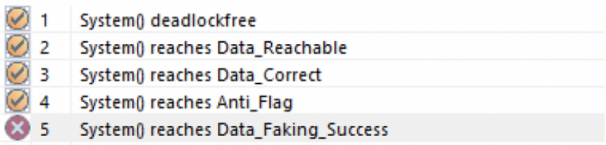

Fig. 6. Verification Results of the Model

The verification results indicate that the auction mechanism can avoid deadlock. At the same time, it can obtain the target data from an authorized user and achieve proper result. Moreover, it can protect messages from interception and tampering in the process by encrypting bid price and generating digital signatures which intruder is unable to fabricate.

Furthermore, the system can provide users with a way of recording the origin data and maintain data copies permanently which cannot be modified by intruders owing to digital signature authentication. The intruder can decrypt sealed sb and modify it by acquired sck_{pri} when the data leakage occurs. However, it cannot imitate the signature of the DataBuyer. As a result, SmartContract receives msg from Intruder and verifies its certification which msg cannot pass. Hence, the attack is invalid and has no effect on the auction result.

Thus, we can come to the conclusion that anti-collusion auction mechanism based on smart contract can guarantee high truthfulness, safety and reliability.

V. CONCLUSION AND FUTURE WORK

In this paper, we adopted process algebra CSP to model the communication process of this architecture of the proposed [5] decentralized data auction scheme. It is based on smart contract which can solve the problem of collusion. We verified five properties: *Deadlock Freedom*, *Data Reachability*, *Data Correctness*, *Anti-collusion Capability* and *Data Security* with the assistance of the model checker PAT. As demonstrated in the verification results, it satisfies all properties, indicating the distributed mechanism is of great reliability, confidentiality and security under any circumstances. It guarantees to return the valid result of the data auction by encrypting vital information. Besides, it creates digital signatures attached to the message transferred in the system to ensure the safety.

In the future, we will put forward analysis on security strategies of smart contract and take the system of multiple sellers into consideration. Meanwhile, we will also introduce more types of attack into authentication process, such as replay attack and reentrancy attack.

VI. ACKNOWLEDGEMENTS

This work was partially supported by the "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (No. 22510750100), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing (East China Normal University).

REFERENCES

[1] Jun Du, Erol Gelenbe, Chunxiao Jiang, Zhu Han, Yong Ren: Auction-Based Data Transaction in Mobile Networks: Data Allocation Design and Performance Analysis. IEEE Trans. Mob. Comput. 19(5): 1040-1055 (2020).

[2] Wolfgang Jank, Shu Zhang: An Automated and Data-Driven Bidding Strategy for Online Auctions. INFORMS J. Comput. 23(2)CCF B: 238-253 (2011).

[3] Shashank Pandit, Duen Horng Chau, Samuel Wang, Christos Faloutsos: Netprobe: a fast and scalable system for fraud detection in online auction networks. WWW 2007: 201-210.

[4] Anup Kumar Paul, Xin Qu, Zheng Wen: Blockchain-a promising solution to internet of things: A comprehensive analysis, opportunities, challenges and future research issues. Peer-to-Peer Netw. Appl. 14(5): 2926-2951 (2021).

[5] Wei Xiong, Li Xiong: Anti-collusion data auction mechanism based on smart contract. Inf. Sci. 555: 386-409 (2021).

[6] Bhabendu Kumar Mohanta, Soumyashree S. Panda, Debasish Jena: An Overview of Smart Contract and Use Cases in Blockchain Technology. ICCCNT 2018: 1-4.

[7] Baoyi An, Mingjun Xiao, An Liu, Yun Xu, Xiangliang Zhang, Qing Li: Secure Crowdsensed Data Trading Based on Blockchain. IEEE Trans. Mob. Comput. 22(3): 1763-1778 (2023).

[8] Shuangke Wu, Yanjiao Chen, Qian Wang, Minghui Li, Cong Wang, Xiangyang Luo: CReam: A Smart Contract Enabled Collusion-Resistant e-Auction. IEEE Trans. Inf. Forensics Secur. 14(7): 1687-1701 (2019).

[9] C. A. R. Hoare: Communicating Sequential Processes. Prentice-Hall, 1985.

[10] Jun Sun, Yang Liu, Jin Song Dong: Model Checking CSP Revisited: Introducing a Process Analysis Toolkit. ISoLA 2008: 307-322