Say No to Price Discrimination: Decentralized and Automated Incentives for Price Auditing in Ride-Hailing Services

Youshui Lu, Yong Qi, *Member, IEEE*, Saiyu Qi, *Member, IEEE*, Yue Li, Hongyu Song, and Yuhao Liu

Abstract—As the most successful application of the sharing economy, ride-hailing service is popular worldwide and serves millions of users per day worldwide. Ride-hailing service providers (SPs) usually collect users' personal data to improve their services via big data technologies. However, SPs may also use the collected user data to apply personalized prices to different users, which raises price fairness concerns. In this paper, we propose a smart price auditing system named Spas. Spas allows a user to purchase *Fair Price Insurance* in the form of *Price Auditing Contract*, then the price of ride-hailing service (RHS) order will be audited automatically once completed. According to the auditing result, the contract punishes misbehaving SPs and also compensates affected users automatically. By replacing an untrustworthy centralized auditor with carefully designed smart contracts, we construct a decentralized price auditing system which is trustworthy and transparent. We demonstrate a theoretical model for practical payment flows based on real RHS user data and we implement Spas in Hyperledger Fabric to show that decentralizing and automating price auditing for RHS with financial incentives is technically feasible.

Index Terms—Auditing, blockchain, smart contract, ride-hailing, game theory, insurance scheme, decentralized system

1 Introduction

Ride-hailing service (RHS), such as Didi Chuxing, Uber, and Lyft, serve millions of users per day to set up rides by their smartphones. RHS allows a user to easily hail a ride and track the driver's location in real-time [3]. Their advantage over traditional taxi services is due to the convenience of hailing services [1], e.g., ride requests at the touch of a button, price estimation, and recommendation of frequent destinations, etc. These services are available because the RHS providers can collect a vast amount of user personal data (such as location traces, identity information).

Unfortunately, malicious RHS providers may also take advantage of the collected data to acquire illegal benefits [2] via price discrimination ("personalized pricing"), in which most customers regard as unfair or manipulative [25]. There is evidence that RHS users are discriminated based on the racial and/or gender information specified in their profiles [27]. Despite racial and gender discrimination, some RHS providers such as Didi Chuxing [22] and Uber [2] have been exposed to "big data killing" behavior-a typical strategy of price discrimination which is also the primary motivation for our paper.

From an economic view, the term "price discrimination" [5] means that the service providers(SPs) build portraits for individual customers based on their personal information, travel habits and destinations and purchase habits, and then provide the same commodity or service at different prices to different users. This price is based on the seller's estimation of the price a buyer might be willing to pay for that good or service. In this paper, we regard the order with personalized pricing as a rogue order. For example, the SPs may charge a higher price from a loyal user than a newly registered user, or it may charge more from a user with high consumption power than a user with low consumption power, or even it may charge more when the users' phone batteries were low. The price discrimination will cause the affected users financial losses in a long run. Also, it will make the users have to compare the prices among different platforms to get the best deal, which will eventually increase the search cost for online users. From a macro-economic perspective, it would be bad for the economy if personalized pricing causes users to lose trust in online sellers in general [25], [29].

Recently, new data protection laws such as the European Union's General Data Protection Regulation (GDPR) [30] have been proposed to prevent the abuse of users' personal data. In the RHS context, however, how to detect price discrimination is still a challenging issue. For one reason, it is difficult to detect whether SPs manually adjust prices according to individuals' characteristics because prices may fluctuate based on multiple travelling factors such as timing, starting location, destination and so on. For another, the non-transparency nature of online RHS platforms does not allow their users to compare prices with others' easily.

• *Youshui Lu, Yong Qi, Hongyu Song, and Yuhao Liu are with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China. E-mail: lucienlu@me.com, qiy@mail.xjtu.edu.cn, {hysong0101, lyuhao}@stu.xjtu.edu.cn.*
• *Saiyu Qi is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: syqi@connect.ust.hk.*
• *Yue Li is with the Department of Economics, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands, and also with Risk Modelling Department, ABN AMRO Bank N.V, The Netherlands. E-mail: yue.li@vu.nl.*

Manuscript received 28 Jan. 2020; revised 3 July 2020; accepted 6 July 2020. Date of publication 9 July 2020; date of current version 7 Jan. 2022.
(Corresponding authors: Youshui Lu and Saiyu Qi.)
Digital Object Identifier no. 10.1109/TMC.2020.3008315

To mitigate price discrimination, a direct technical solution is to stop the SP from collecting the users' personal data. However, it will also prevent the SP to provide desirable and convenient services brought by the data at the same time. Therefore, current RHS users face a trade-off between convenience and fair-trading. In this paper, we aim to answer the following question: *is it possible to build a RHS system that provides automated personalized pricing auditing and ultimately build a system that can offer both convenience and fair-trading to the user?* The answer is positive. Our high-level idea is to allow the SP using users' personal data, but enable pricing auditing at the same time.

However, we still encounter two fundamental problems. First, how can we automatically audit if the price follows an agreed price policy? To manually audit the price, a user has to check a proper price policy, collect all the input variables, and then compare the computation results. This procedure is not only cumbersome but also error-prone.

Second, how can we automatically compensate affected users when price personalization is detected? The existing system allows a user to pursue legal action by reporting a rogue order to a centralized fair trading authority. However, it could take days or even longer for the authority to investigate. Moreover, the investigation process is time and labour consuming due to the large volume of orders every day. As a result, users often cannot get fruitful results when they suffer price discrimination issues.

To address these problems, we propose to resort to the smart contract technology [6], which is a newly emerging blockchain-based computing paradigm that allows defining and executing self-enforcing contracts on the blockchain. In this paper, we build a smart price auditing system (namely Spas), which enables automated order price auditing and compensation without relying on a centralized authority. Meanwhile, since all price auditing results are recorded in Spas's blockchain, Spas holds ride-hailing SPs accountable for any of their offered prices. Such built-in accountability effectively deters misbehaving SPs from offering personalized pricing, which eventually benefits the entire RHS ecosystem.

Instead of relying on a centralized auditor, Spas introduces a decentralized auditor named Spas authority which is made up with lists of smart contracts including *Price Policy Contract (PPC)* and *Price Auditing Contract (PAC)*. Spas authority is responsible for the core functionalities of the system. The SP publishes its current price policies through a PPC. A PAC can be then invoked to use the corresponding price policy to calculate the actual price during the price auditing process. We carefully design the incentive mechanism among these contracts and combine the techniques of *Fair Price Insurance* and two-factor authentication to prevent malicious behaviors of SP and users.

We generate real order price datasets by using the trajectory datasets and the corresponding price policies sourced from Didi Chuxing GAIA Intitutive [9]. With the datasets, we first conduct experiments for comprehensive performance evaluation via a proof-of-concept implementation on Hyperledger Fabric [26]. The experiment results show the practically affordable performance of our design. In addition, we estimate realistic payoff amounts by referring to laws of Consumer Rights and Online Taxi Booking Business Operations [7], [8] as well as the results of the RHS's user survey. Moreover, by formulating Spas as a non-cooperation dynamic game, we use game theory to analyze the design of Spas. Based on the estimated payoff values, we show that even though the user does not purchase insurance, the SP is impossible to apply personalized pricing. At last, we discuss a generic application of price discrimination detection in online services fields.

Our main contributions are summarized as follows:

- We first give a detailed definition of price discrimination, and provide an analysis on how online price discrimination violates data privacy laws, and the lists of challenges faced under RHS context. We propose a decentralized smart price auditing system (Spas) which offers automated order price auditing and compensation upon order completion. At the core of Spas is a set of smart contracts including *Price Policy Contract* and *Price Auditing Contract*. We also carefully design the incentive mechanism behind the contracts to regulate the behaviors of the involving parties.
- To prevent users to launch *Sybil attack*, we introduce a new concept named *Fair Price Insurance* to ensure that a user who purchased price insurance can have his RHS order audited. Also, to prevent users to launch *Spoofing attack*, we use two-factor authentication when using the trajectory data collected from the user's mobile devices during the auditing process.
- We formalize Spas as a non-cooperation dynamic game and use game theory as a tool to analyze the viability of using Spas in the real world. Based on the payout constraints, we show that the incentive mechanism of Spas will lead to independent, self-interested participants toward outcomes that are desirable from a system-wide perspective.
- We implement a prototype of Spas and deploy it on Hyperledger Fabric [26]. We also generate real RHS order datasets by using the trajectory datasets sourced from Didi Chuxing [9], we evaluate the performance of Spas, the example payoff settings, and demonstrate the technical feasibility of Spas.

Paper Organisation. In the rest of the paper, we provide an overview of the background of the RHS, price discrimination, motivation and the preliminaries in Section 2, and then we introduce the problem definition and an adversary model in Section 3. Section 4 provides a detailed explanation of Spas system design which includes the architecture, the events, the workflow of Spas, User Registration, Price Policy Registration and Update, Insurance Purchase and Termination, Price Auditing and Incentive Mechanism and Payout Settings. Next, we provide an analysis of our proposed model in Section 5. Our experiment is presented in the evaluation Section 6. Finally, we survey related work in Section 7, discussion and limitations in Section 8 and conclude the paper in Section 9.

2 Background and Preliminaries

2.1 Background

2.1.1 Price Discrimination

To illustrate the price discrimination phenomena (Fig. 1), image a SP who supplies service with a constant marginal

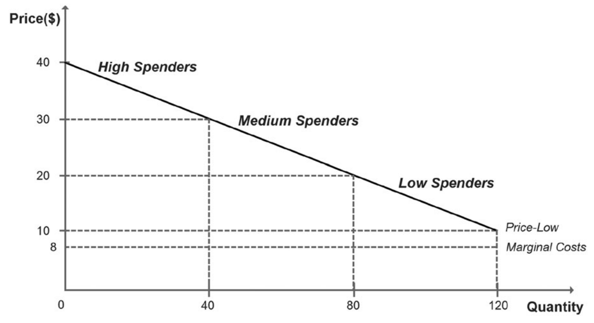


Fig. 1. Stylized example of SP applying price discrimination.

cost of \$8 to a market of 120 customers. One-third of these customers, the high spenders, are willing to pay an amount between \$30 and \$40 for this service; another one third are the medium spenders who are willing to pay an amount between \$20 and \$30; the remaining are the low spenders between \$10 and \$20. Under uniform pricing, the SP would consider the lowest price point. At $\texttt{Price}_{Low}$ = \$10, all the customers are buying. The SP sells 120 units ($\texttt{Quantity}_{Low}$ = 120), making a profit of \$2 per unit, \$240 in total. Low spenders have a user surplus [25] (the difference between their willingness to pay and what they pay) of between \$2 and \$12, medium spenders between \$12 to \$22 and high spenders between \$22 and \$32. Total user surplus at this price is 40 $\times$ (\$15–8) + 40 $\times$ (\$25–8) + 40 $\times$ (\$35–8)= \$2040. This example illustrates that when all the users buy the service, personalized pricing will bring significantly more profits to the SP. However, it might cause the medium and the high spenders feel unfair when their prices are personalized.

2.1.2 Motivation

Although personalized pricing is a marketing strategy, many people regard personalized pricing as unfair or manipulative. In a US survey, Turow *et al.* [13] "found that [American adults] overwhelmingly object to most forms of behavioural targeting and all forms of price discrimination as ethically wrong." In another US survey, 78 percent of the respondents did not want "discounts (...) tailored for you based on following (...) what you did on other websites you have visited"[14]. In addition, personalized pricing may also lead to the users worrying about being personalized, such suspiciousness therefore increases their search costs with a need to search around to make sure they get the best deal [12]. From a macro-economic perspective, it would be bad for the economy if personalized pricing causes users to lose trust in online sellers in general [25], [29].

Moreover, with data protection laws providing general principles of data transparency, however, online users must be able to understand when and which personal data is collected, used, consulted or otherwise processed, as well as the purpose of such processes. Data protection law will be triggered when "personal data" are processed. Under RHS context, when a SP provides service to its user, almost anything that can be done with personal data, such as storing and analysing its user's data, which falls within the definition of "processing" [30]. Therefore, to comply with the data protection laws, the SPs must, for instance, provide information regarding its user's identity and "the purposes of the processing" and must provide more information when necessary to guarantee fair processing. Hence, a SP must inform customers if it processes personal data to personalize prices.

However, current SPs do not give enough consent notices to their users about how they will process their data. The reason is that the price of a RHS order is determined by many variables such as trip distance, duration, and so on. The SP can bravely manipulate those variables and display them on the user's app because no user can easily detect SP's misbehavior since two different orders can rarely share the same variables. Therefore, the SP can simply apply personalized pricing to its users without worrying about being punished. For instance, the SP may pay less to a driver with higher loyalty than the newly registered driver. The SP may also charge more to high spenders than low spenders. (To simplify the illustration of Spas, in this paper, we assume that the SP can use personalized pricing to the rider only, and we use user to denote rider in the rest of this paper.)

Therefore, in this paper, we are motivated to foster fairness in RHS settings which not only enables data processing happens transparently on the user level but also increase the health of the online market ecosystem in general.

2.2 Preliminaries

In this section, we provide an overview of blockchain-based cryptocurrencies, which we use to instantiate Spas.

2.2.1 Blockchain

A blockchain [7] may be defined as a database that is shared among its users and allows them to transact valuable assets in a public and pseudonymous setup without the reliance on an intermediary or central authority [11]. There are three core elements included in a blockchain system: the block, the chain, and the activity.

Specifically, a block is the storage carrier based on a consensus agreement by all stakeholders or validators. The storage content also captures the interactions between the different parties, such as transactions in Bitcoin [19]. An activity in a blockchain system can be represented in a service manner. For example, a digital transaction can be deemed the service content in Bitcoin. Also, a chain is a method of connecting all blocks in a one-way growing manner. The one-directional chain growth is a critical element of determining its tamper-resistant characteristic.

2.2.2 Smart Contracts

The term smart contract was initially introduced in the 90s by N. Szabo [6], stemming from the idea that a technological legal framework would help commerce, reduce costs and disputes. A smart contract allows users to define and execute contracts on the blockchain [6], and maintains its own data storage and balance, access to both of which is entirely governed by its code (though all contract data and balances can be publicly read on the blockchain). The program code captures the contractual logic clauses between multiple parties and pre-defines the trigger conditions and response actions. The execution of the functions in a smart contract is triggered by times or events, e.g., transactions added to the blockchain. Contracts allow for the creation of autonomous agents whose behavior is entirely dependent on their code and the transactions sent to them, thus providing functionality comparable to that of a centralised party in a transparent, decentralised manner. With the help of smart contracts, the rules of financial transactions can be enforced without trusted third-parties.

Hyperledger Fabric [26], which supports smart contract, is a project hosted by the Linux Foundation as a cross-industry collaborative project. The system was designed with the enterprise architecture in mind with customized networking rules that help different consensus protocols operate. It borrows the Unspent Transaction Output(UTXO) and script-based logic from Bitcoins [19], and uses Practical Byzantine Fault-tolerant (PBFT) [24] consensus protocol instead of the PoW algorithm. PBFT is known to process thousands of requests per second with a latency increase of less than a millisecond. In this paper, we implement a proof of concept prototype of Spas by using Fabric, which we will discuss in the latter part of this paper.

2.2.3 *Game Theory*

Game theory [31] is a field of applied mathematics that describes and analyzes interactive decision situations. It provides analytical tools to predict the outcomes of the complex interactions between rational parties, where rationality demands strict adherence to a strategy based on perceived results. We limit our discussion to non-cooperative models that address the interaction between individual rational decision-makers. Such models are called "games" and the rational decision-makers are referred to as "players." In the most straightforward approach, players choose a single action from a set of possible actions. Interaction between the players is represented by the influence that each player has on the resulting outcome after all players have chosen their actions.

3 Problem Definition and Adversary Model

The goal of this paper is to efficiently detect price discrimination by automatizing price auditing process and compensating affected users, thus ensuring fair and transparent trading in RHS. In order to achieve this goal, we must design a system which can 1) Define the format of personalized pricing and specify reactions and payments that will take place when personalized pricing presents; 2) Process audit requests regarding personalized price automatically; 3) Execute reactions and payments automatically once personalized pricing was detected; 4) An incentive mechanism to punish misbehaving SP and compensate affected users.

3.1 Desired Properties

A system that aims to achieve the above goals should have at least the following features:

- Trustfulness: the audit processes should be executed in a trusted way.
- Correctness: the audit results should be correct and traceable.
- Automation: the auditing process should be executed automatically upon the order completion and the procedure should be executed automatically without requiring additional information or authorisation.
- Compensatory: the affected user should be compensated once the rogue order is detected.
- Punishment and encouragement: the SP has an expected negative return on investment for offering personalized pricing.

3.2 Adversary Model

The adversary's goal is to offer personalized pricing by analysing users' personal data or even to collude with some users while maintaining an expected positive expected return on investment. The adversary can take any action allowable in Spas to improve its benefits. We assume that the location data collected from the driver app is accurate and tamper-proof, however, the adversary user can forge his location data. We further assume that the communication between different entities is reliable and trusted. Also, the security of the standard cryptographic primitives used in Spas cannot be broken, such as finding hash collisions, forging digital signatures or compromise the private keys of arbitrary entities. We further assume that the adversary cannot control over half of the hashing power in the blockchain network in our instantiation.

3.3 Design Challenges

We encounter the following technical challenges in the design of Spas:

Challenge 1: Sybil Attack by the User. Malicious users may abuse their rights in auditing all of their RHS order to increase the number of auditing operations, thus prolonging the latency of the system and making the system less efficient.

Challenge 2: Spoofing Attack by the User. Malicious users may transmit tampered trajectory information to the decentralized auditor to get compensated for their RHS orders, it therefore leads to a huge loss to the affected SP.

Challenge 3: Collusion Attack by Different Entities. Different entities may collude to obtain a positive return on investment. For instance, SP and user may collude to profit through auditing.

4 Spas System Design

In this section, we introduce the framework of Spas. We first introduce Spas's architecture. We then describe the workflow and events in Spas. Next, we focus on the detailed explanation on the designs of the price policy registration process and the price auditing process. Also, we give an in-depth discussion on the process of insurance purchase and termination and the two-factor authentication of the trajectory information.

4.1 Architecture

Fig. 2 shows an overview of the entities and functions in Spas framework. Similar to current RHS systems, Spas mainly includes three types of parties: users, drivers, and SPs. Spas relies on centralized SPs to bridge RHS between users and drivers. Differently, Spas introduces a distributed authority, namely Spas authority, which consists of a list of smart contracts including *Pricing Policy Contract*, *Insurance Purchasing Contract(IPC)*, and *Price Auditing Contract*. In

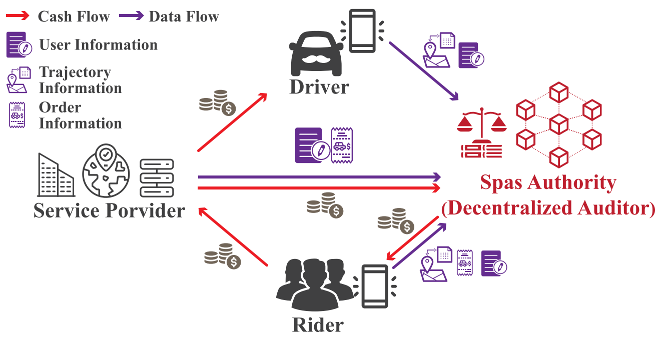


Fig. 2. Overview of the entities and their interactions in Spas.

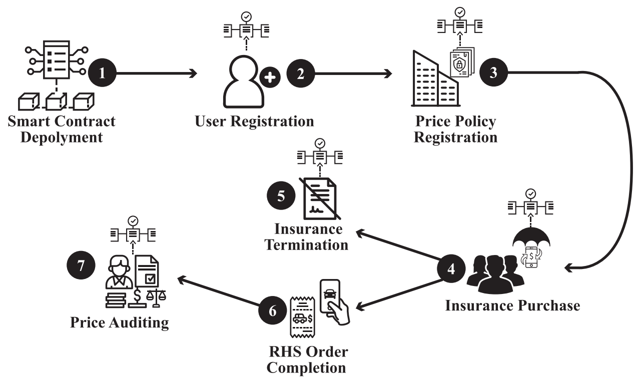


Fig. 3. Workflow and Events in Spas.

addition, the Spas authority also manages the *Global Account* which maintains a balance called the global fund to send and receive payments in Spas.

Moreover, entities in the standard RHSs have additional responsibilities. SP can sell *Fair Price Insurance* to its user, and eventually the SP will act as an insurer against personalized pricing. SPs also register and update their price policies with the Spas authority. In addition, a user can purchase *Fair Price Insurance* via *IPC*.

4.2 Workflow and Events

Spas authority, as the core entity in our system, plays an important part in the workflow of Spas. The key events in Spas (shown in Fig. 3) consists of the registrations, auditing, purchase or termination insurance. We introduce these events as follows.

User Registration is designed for the parties to register in Spas. It maintains information of each party such as identifiers (e.g., user IDs), and financial account information at which to receive payments.

For any SP to register as an legitimate SP, the SP has to deposit *Deposit* to the *Global Fund Account*. It is reasonable as in many countries to obtain a business license in certain industry, the company has to deposit a quality of service guarantee to the administrative authorities. In Spas, we regard this deposit as a statutory reserve for any punishment fees incurred in the future.

Price Policy Registration and Updates is designed for the SP to register and update its price policies which are the references for future auditing. The logic of price policies is determined by machine understandable policies. We will discuss in details in Section 4.4.

Purchase or Terminate Insurance. A user can purchase a *Fair Price Insurance* through the *IPC*, only insured user can have his order audited upon order completion. Moreover, the insured user can choose to prematurely terminate the insurance once the SP is found to have misbehaved. The detailed insurance scheme will be discussed in the Section 4.5.

Price Auditing. For an insured user, the price auditing process will be triggered by executing the *PAC* upon the order completion. The *PAC* defines the auditing procedure and lists the financial reaction policies when a rogue is detected. The logic of reaction policies are well written in the reaction programs, bringing in flexibility in addition to the automation and financial incentives. We will discuss further in Section 4.6.

4.3 User Registration

For the entities in Spas, such as user or SP, who by our design choices, do not need any publicity. For example, when the SP makes physical interaction with a user, the SP may wish to query the driver's driving license before agreeing to the transaction. The SP, on the other hand, needs a public identity(pseudonym) that others could refer to for payouts and punishments. The detailed registration transaction is as follows.

- Inputs:
  - Identity information $\mathrm{Id_u}$
  - Financial account information: user's account address $\mathrm{Acc_u}$ or SP's account address $\mathrm{Acc_{SP}}$
- Outputs:
  - Registration transaction **rt**

Entities in Spas register through *User Registration Contract*. They generate new, unique, public-private key pair (u_{pk}, u_{sk}) in the underlying system that will be used to process payouts.

$$\mathbf{rt} = (\mathrm{Timestamp, TxID, EnType, Info_u, u_{pk}}).$$

TxID is a unique identifier for the transaction which in practice could be hash of all the other values in **rt** (including the unique public key u_{pk}). EnType is the type of the registered entity, it could be a user or a SP. $\mathrm{Info_u}$ is the description of the registered entity, the detailed profile such as the driver's license information for a driver, and any other information needed, and u_{pk} is the public key uniquely associated with this entity.

4.4 Price Policy Registration and Update

The SP must publish its up-to-date pricing policy to the public, therefore the auditor could take the updated policy as a reference to process price auditing operation. The price policy should always be accessible by any users in the system since it defines the detailed legitimated pricing clauses which should obtain its users' consent. In Spas, the SP updates its latest price policy with the Spas authority through *Price Policy Contract* which also converts those polices into machine understandable language at the same. The complete price policy registration transactions are as follows.

- Inputs:
  - Type (e.g., Express) Type
  - Policy version VerNum

 - Pricing clauses information PPInfo
 - Update keys UpKey
- Outputs:
 - Price policy registration transaction **pprt**

Each **pprt** contains enough information to allow *PAC* to take as a reference when auditing an order. Also, a price policy has a staring time, from which the policy is effective. A price policy transaction is in the following form:

$$\mathbf{pprt} = (\mathrm{Timestamp}, \mathrm{TxID}, \mathrm{Type}, \mathrm{PPInfo}, \mathrm{VerNum}, \mathrm{Upkey}).$$

Where Type is the service type information, such as Express, Comfort, Luxury, etc. PPInfo is the price policy information which lists the details clauses such as the definition of the Peak hour and Off-peak hour intervals during the day. VerNum is the version number. And UpKey is an update key which is required to authorize changes to the price policies. Price policies serve an important role in determining a rogue order because price polices of a particular user type is tied to the reactions polices in the *PAC*.

4.5 Insurance Purchase and Termination

4.5.1 Purchasing Insurance

As aforementioned, Spas introduces a novel *Fair Trade Insurance* to prevent *Sybil attack* launched by malicious users. To purchase insurance, a user transfers a fee f_{Ins} to the *Global Account* automatically. Only by purchasing the insurance, the user can have his order audited at the later stage. In Spas, a user purchases the insurance through *IPC*, and the detailed transaction is as follows.

- Inputs:
 - Registration transaction **rt**
 - Payment in currency f_{Ins}
- Outputs:
 - Insurance payment transaction **ipt**

Each **ipt** contains enough information to allow *PAC* to verify the validity of insurance. Also, insurance has a staring time, an expiry time, insurance price f_{Ins}, coverage (such as Express order), SP's identity and any other information which is included in the $\mathrm{InsInfo}$. An **ipt** transaction is in the following form:

$$\mathbf{ipt} = (\mathrm{Timestamp}, \mathrm{TxID}, \mathrm{Id_u}, \mathrm{InsInfo}).$$

The insurance validity period can be a day, a week, a month or even longer, it depends on what plan the user will purchase.

4.5.2 Terminating Insurance

Spas also supports insurance termination operation. Therefore, the insured user can choose to prematurely terminate the insurance once the SP is found to have misbehaved. It is reasonable because the RHS user will always choose a trustworthy SP, once the user loses trust in a particular SP, he is probably not using this SP any more. If a user wishes to terminate his insurance, he can request through the *ITC* as follows.

- Inputs:
 - Registration transaction **rt**
 - Insurance payment transaction **ipt**
- Outputs:
 - Insurance termination transaction **itt**

An **itt** transaction is in the following form:

$$\mathbf{itt} = (\mathrm{Timestamp}, \mathrm{TxID}, \mathrm{Id_u}, \mathrm{PayInfo}).$$

Where $\mathrm{PayInfo}$ contains the detailed payouts information between each entity. As insurance has a limited validity period. If insurance is terminated for any reason, the specified amount of funds is split between the user and the *Global Account* based on the fraction of the reaction policies validity period that has passed. The detailed payouts settings will be discussed in Section 4.7.

4.6 Price Auditing

In this subsection, we will mainly focus on *Price Auditing Contract*. Also, we will discuss in details about the key components during the auditing process including the trajectory two-factor authentication and the reaction policies of the auditing results.

4.6.1 Price Auditing Contract(PAC)

The *PAC* is automatically triggered once the insured user's order is completed and begins the price auditing process.

The detailed price auditing algorithm is shown in Algorithm 1.

Algorithm 1. Price Auditing

```
Input: 1) $MSG_R$:Message received from the rider
       2) $MSG_D$:Message received from the driver
       3) $MSG_{SP}$:Message received from the SP
       4) Rider's ipt
       5) Rider's rt
       6) SP's updated pprt
Output: Price auditing transaction pat
Procedure: Price Auditing Process
 1: Check the validity of the rider's registration through rt;
 2: Check the validity of the rider's insurance through ipt;
 3: Process two-factor authentication of the messages transmitted from different entities:
 4: if verified then
 5:   $Price$ ←get charged price from OrderInfo;
 6:   $Time_{PK}, Time_{OK}, Dist_{PK}, Time_{OK}$ ← get travel information from TraInfo;
 7:   PPInfo ←get price policy from pprt;
 8:   $Price'$ ← calculate order correct price based on the information above;
 9:   if $Price > Price'$ then
10:     ReactionProgram.trigger;
11:   else
12:     end process.
13:   end
14: else
15:   end process.
16: end
```

4.6.2 Two-Factor Authentication

During the process of auditing a particular order, the *PAC* will use the order's information as well as the trajectory data collected from the user's mobile device. However, malicious users may launch *Spoofing attack* by tampering the

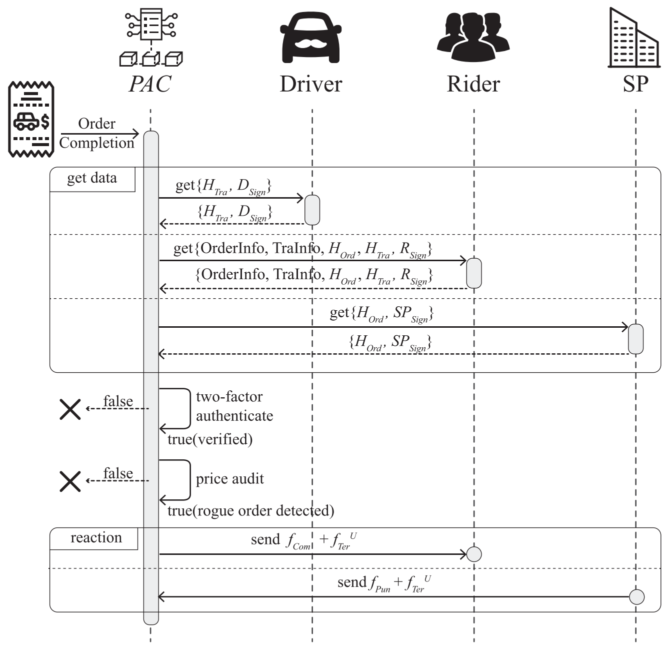

PAC
Driver
Rider
SP
Order Completion
get data
get{H_{Tra}, D_{Sign}}
{H_{Tra}, D_{Sign}}
get{OrderInfo, TraInfo, H_{Ord}, H_{Tra}, R_{Sign}}
{OrderInfo, TraInfo, H_{Ord}, H_{Tra}, R_{Sign}}
get{H_{Ord}, SP_{Sign}}
{H_{Ord}, SP_{Sign}}
false
two-factor authenticate
true(verified)
false
price audit
true(rogue order detected)
reaction
send $f_{Com}+f_{Ter}^{U}$
send $f_{Pun}+f_{Ter}^{U}$

Fig. 4. Sequence diagram of price auditing process(*PAC*).

necessary information. To prevent *Spoofing attack*, we design a two-factor authentication scheme to ensure that the information used during the auditing process is correct and tamper-proof. Fig. 4 shows that all the involving entities will interact with the *PAC* during the price auditing process. The rider will transmit MSG_R, the driver will transmit MSG_D, and the SP will transmit MSG_{SP} through a secure channel to the *PAC*. Note that the trajectory information of a same RHS order is collected from different sources, the SP's application and the user's mobile devices (ride-hailing service and smart audit are two different applications, during auditing process, we assume that the smart contract can collect information from user's devices in addition to the RHS software operated by the SP).

$$MSG_R = \{\mathrm{OrderInfo}, \mathrm{TraInfo}, H^R_{Ord}, H^R_{Tra}, R_{Sign}\}$$

$$MSG_D = \{H^D_{Tra}, D_{Sign}\}$$

$$MSG_{SP} = \{H^{SP}_{Order}, SP_{Sign}\}$$

$$H^R_{Ord} = Hash(\mathrm{OrderInfo})$$

$$H^R_{Tra} = Hash(\mathrm{TraInfo})$$

$$R_{Sign} = Sign(\mathrm{OrderInfo} \parallel \mathrm{TraInfo} \parallel H^R_{Ord} \parallel H^R_{Tra}),$$

where $\mathrm{OrderInfo}$ is the order's information, $\mathrm{TraInfo}$ is the trajectory information collected from the rider's mobile device, it consists of the peak hour travel time $Time_{PK}$ and travel distance $Dist_{PK}$, off-peak hour travel time $Time_{OK}$ and distance $Dist_{PK}$, and so on. H^R_{Ord} is the hash value of the order's information, H^R_{Tra} is the hash value of the trajectory information, and R_{Sign} is the rider's signature that is calculated with rider's secret key R_{sk}.

In addition, the driver will only transmit the hash value of the trajectory information $H^D_{Tra} = Hash(\mathrm{TraInfo}')$ and D_{Sign} which is the driver's signature signed over $\mathrm{TraInfo}'$ by the driver's secret key D_{sk}.

$$D_{Sign} = Sign(\mathrm{TraInfo}', D_{sk}).$$

Similarly, the SP will transmit the trajectory information's hash value $H^{SP}_{Ord} = Hash(\mathrm{OrderInfo}')$ and SP_{Sign} which is the SP's signature signed over $\mathrm{OrderInfo}'$ by the SP's secret key SP_{sk}.

$$SP_{Sign} = Sign(\mathrm{OrderInfo}', SP_{sk}).$$

When the *PAC* collects these information above, it first compares the hash values of the rider's and the driver's trajectory information,

$$H^R_{Tra} \overset{?}{=} H^D_{Tra},$$

if the result is equal, it then compares the hash values of both the rider's and the SP's order information,

$$H^R_{Ord} \overset{?}{=} H^{SP}_{Ord},$$

if the result is equal, it means that the rider's information is correct and not tampered, and then the *PAC* will use $\mathrm{OrderInfo}$ and $\mathrm{TraInfo}$ in the later stage of the auditing process.

4.6.3 Reaction Policy

The reaction policy in Spas refers to the insurance claim which is the financial reaction transaction defined in the *Fair Price Insurance*. The reaction policy contains (1) the user's identifier, (2) a reference to the user's price policy, (3) a checker program that ensures the pre-defined trigger condition has met and (4) a reaction program which contains the payments that occur in response to a rogue order.

We first explain the design principles of the reaction policies.

- *P1. User-independence.* User has an option to purchase insurance from a SP before an order started. However, insurance is independent: any registered user can purchase insurance from any SP.
- *P2. Policy-adherence.* A reaction policy should binding to a specific type of pricing policy. Therefore it ensures the consistency between the policies for price auditing and the financial reactions for breaching those policies.
- *P3. Single-use.* A reaction policy should be limited to a single rogue order. Once detecting a rogue order, the reaction program will be executed and the insurance will terminate automatically. Because the *PAC* may execute financial payments, enforcing single-use reaction policy helps to ensure the availability of such one-time resources for each rogue order.

When an order is determined as a correct order, it will not trigger the reaction programs. However, we will focus on discussing the case when a rogue order is detected. We now define the contents and format of reaction policies. Fig. 5 shows the format of a sample reaction policy, and

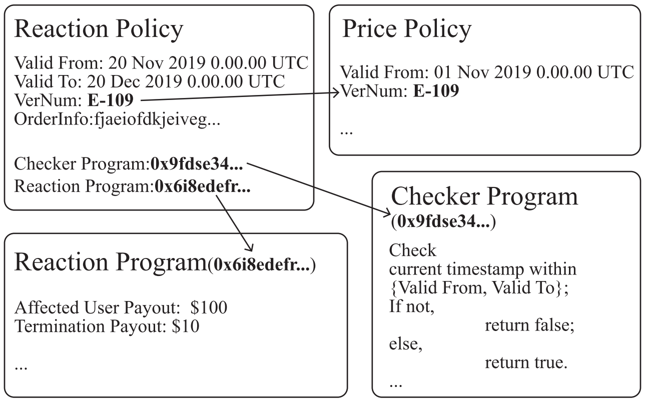

Reaction Policy
Valid From: 20 Nov 2019 0.00.00 UTC
Valid To: 20 Dec 2019 0.00.00 UTC
VerNum: E-109
OrderInfo:fjaeiofdkjeiveg...
Checker Program:0x9fdse34...
Reaction Program:0x6i8edefr...
Price Policy
Valid From: 01 Nov 2019 0.00.00 UTC
VerNum: E-109
...
Checker Program
(0x9fdse34...)
Check
current timestamp within
{Valid From, Valid To};
If not,
return false;
else,
return true.
...
Reaction Program(0x6i8edefr...)
Affected User Payout: $100
Termination Payout: $10
...

Fig. 5. Sample reaction policy.

Table 1 describes each field of a reaction policy. A reaction policy contains the order information including the user's account addresses extracted from the earlier steps. It also specifies a validity period and identifies the version of the price policy for which it is active, all the required information is extracted from the user's binding insurance.

Reaction Program. As shown in Algorithm 2, the *PAC* contains the reaction program to execute financial transactions when a rogue order is detected. When an order's reaction program is triggered, this user's insurance is terminated as one reaction program can only be used once in life. This mechanism not only avoids the users to manually terminate his insurance, but also has the insurance proportionally refunded to the user as soon as the reaction program is executed, thus guaranteeing the users have the minimum loss on the insurance cost.

Algorithm 2. Reaction Program

```
Input: 1) OrderInfo: order's information;
       2) SP's updated pprt;
       3)$Acc_F$: Global Fund's account address;
Output: Payouts distribution
Procedure: Reaction Process
1: $Acc_R \leftarrow$ get affected user's account from OrderInfo;
2: $Acc_{SP} \leftarrow$ get misbehaving SP's account from OrderInfo;
3: $Id_u \leftarrow$ get user's identity information from OrderInfo;
4: $RP_{id} \leftarrow$ look up from reaction programs list RPL
5: Send $fCom + f_{Ter}^U$ from $Acc_F$ to $Acc_R$;
6: Send $f_{Pun} - f_{Ter}^U$ from $Acc_{SP}$ to $Acc_F$;
7: Delete $RP_{Id}$ from RPL.
8: end Procedure
```

A reaction program defines three methods as follows:

1) trigger, which executes when a rogue order is detected
2) terminate, which executes upon request from a user, and
3) expire, which executes automatically via *PAC* after the reaction policy has expired.

We note that a reaction policy has a start and end time for its validity, rather than only a start time as a price policy does. A reaction policy has a limited validity period. If a reaction policy is terminated for any reason, the specified amount of funds is split between the user and the SP based on the fraction of the reaction policies validity period that has passed.

TABLE 1
Explanation of Reaction Policy Fields

Field	Use
Valid From	specify start period of reaction policy's Validity
Valid To	specify end period of reaction policy's Validity
VerNum	version of user's price policy used to trigger reaction policy
OrderInfo	the order's information including the account informaiton of each entity
Reaction Program	implement a response to rogue order

4.7 Incentive Mechanism and Payout Settings

We now discuss the incentive mechanism that we designed for punishing misbehaving SP that generates rogue orders and compensating victims. We first provide a framework for payout reaction programs, which specify a set of financial payments that execute in response to a rogue order.

As Fig. 5 shows, a payout reaction program specifies two payouts: affected-user payouts and termination payouts. To ensure that the *Global Fund Account* has a sufficient balance for these payouts, a security deposit $Deposit$ is sent to the global fund by SP when registering.

Affected-User Payouts. The affected-user payout (f_{Com}) is paid to the users involved in a rogue order. The payout compensates the users for the risk it incurs by having an incorrect fare paid. Meanwhile, the punishment fee for a misbehaved SP would be f_{Pun}, and we have

$$f_{Com} < < f_{Pun}, \tag{1}$$

f_{Pun} is a fixed large amount of money.

Termination Payouts. The termination payout (f_{Ter}) is split between the user and the SP, if the user terminates the insurance. The split of the termination payout between the user and SP is proportional to the amount of time left in the insurance validity. Letting $\alpha \leqslant 1$ denote the fraction of the reaction policy's remaining validity, we then have

$$f_{Ter}^U = \alpha \cdot f_{Ins}. \tag{2}$$

Because $0 \leqslant \alpha \leqslant 1$, we see that f_{Ter}^U is bounded by

$$0 \leqslant f_{Ter}^U \leqslant f_{Ins}. \tag{3}$$

We note that although SP does receive funds from the termination payout despite its misbehavior, we show in Section 5.2 that SP loses more funds compared to if it had behaved correctly.

We next describe proper payout settings for a reaction policy in Spas. Particularly, for a reaction policy with a payout reaction program, Spas presets the payouts f_{Com}, $Deposit$ and f_{Pun}, as well as the price f_{Ins} of the insurance. Spas sets the constraint on the amounts that must hold

$$f_{Ins} < f_{Com} < f_{Pun} < Deposit. \tag{4}$$

These constraints are justified in Section 5.2.

In addition, the reaction program also considers the following scenarios including trigger, terminate, and expire accordingly.

- *trigger.* When an order is determined as rogue, the reaction program is triggered and executed automatically. The reactions include user U receives the affected-user payout f_{Com} and its share termination payout f^U_{Ter}, and SP receives its share of the termination payout $f_{Ins} - f^U_{Ter}$.
- *termination.* If SP is found to have misbehaved, any insured user U that has a valid issuance can prematurely terminate the insurance. To do so, the user terminates through the *Insurance Termination Contract*. The *Insurance Termination Contract* automatically checks the validity of the insurance. If that insurance has not expired, U receives its share of the termination payout f^U_{Ter}, and the SP receives its share $f_{Ins} - f^U_{Ter}$.
- *expiration.* Once the validity period for insurance belonging to a user U has ended, the reaction program linked to the insurance will be automatically removed from the list of U's reaction programs. By doing so, it can reduce the computation cost of the insurer SP.

5 Analysis

In this section, we make the security analysis and theoretical analysis for the ensured security goals and the expected incentives of Spas. Also, we make a feasibility analysis of Spas by using game theory.

5.1 Security Analysis

Security Against Sybil Attacks. Spas supports personalized price detection once a RHS order completed. However, users may launch *Sybil attacks* to increase the computational overheads of the blockchain network, thus making the system inefficient when detecting personalized pricing orders. By introducing an insurance scheme in Spas, the user has to purchase insurance from a SP before his orders to be audited, which increases the costs for the users in case of order auditing. Therefore, Spas can mitigate *Sybil attacks*.

Security Against the Malicious User. Spas introduces a new type of entity named decentralized detector. Thanks to the decentralized property, it ensures no centralized authority will control the auditing process. However, during the auditing process, a malicious user may launch *Spoofing attack* by transmitting manipulated order information at this point. In Spas, we design the two-factor authentication scheme which enables *PAC* to verify the order information from SP's server, driver's mobile device, and the rider's mobile device. The scheme compares the hash values of collected information from different entities, such as the hash values of the rider's and the driver's trajectory information, $H^R_{Tra} \overset{?}{=} H^D_{Tra}$, and the hash values of the rider's and the SP's order information, $H^R_{Ord} \overset{?}{=} H^{SP}_{Ord}$. Spas will not allow manipulated information to proceed to the next auditing step, therefore it ensures that malicious user will not succeed.

Security Against Collusion Attacks. Spas provides resistance against the collusion of entities, which can be launched by an insured user and a SP. The collusion attacks may be launched for the following purposes: i) decreasing punishments for SP's price discrimination misbehavior; ii) obtaining more compensations from a rogue order detection. Spas designs an incentive mechanism scheme and the payout constraints to ensure that even different entities collude, they will not achieve a positive return on investment. (detailed in Section 5.2). More importantly, Spas provides decentralized and automated incentives/punishments that can help to regulate the behaviors of entities and make them tend to behave normally.

Security Against SP's Repudiating Behavior. The SP can refuse to accept punishment by transferring no incentive to affected users, disabling the incentives allocation. When a SP first registers in Spas, it has to put a deposit *Deposit* in the *Global Fund Account*, this deposit serves as a statutory reserve for any punishment fees incurred in the future. If a misbehaved SP refuses to transfer the punishment fees to the affected user, the fee will be automatically deducted from this deposit by smart contract. Therefore, Spas can defend against SP's repudiating behavior.

TABLE 2
List of Payments Send for Each Event

Event	From	To	Amount
Sell Insurance	U	S	f_{Ins}
SP Registration	F	S	$Deposit$
Terminate Insurance	F	U	f^U_{Ter}
	F	S	$f_{Ins} - f^U_{Ter}$
Report Correct Order	-	-	-
Report Rogue Order	F	U	$f_{Com} + f^U_{Ter}$
	F	S	$f_{Ins} - f^U_{Ter}$
	S	F	$-Deposit$

U represents the user, and F is the global fund, Deposit represents the amount sent to the global fund by S.

5.2 Theoretical Analysis on the Incentive Mechanism

We model each party's incentives in the Spas by considering the flow of payments among entities for each operation. By using this model, we demonstrate several important guarantees that hold in Spas. Table 2 summarises the payout amounts for each action in Spas. For most of the time, we consider a single reaction program lifetime, and use the following notations:

- U denotes the user for whom a (possibly rogue) order is incurred,
- S denotes the SP that sells the insurance to U,
- F denotes the global fund.

To demonstrate the core properties above and simplify our analysis, we only consider the scenario of whether the SP uses personalized price or not. For this case, we take into account the payments made in the series of events which must have occurred and can determine each entity net reward by summing its received payments and subtracting the sum of the outgoing payments. We acknowledge that we would not consider payments made outside Spas, as we cannot track or govern those transactions.

For this scenario, we consider whether S misbehaves by applying personalized prices or not. We assume that the insurance is valid and the proper payments have been made. We observe that if no rogue order is detected, then the insurance will eventually expire, regardless of whether

TABLE 3
Rewards For Each Entity In Different Scenarios

Entity	No Rogue Order Detected	Rogue order Detected
U	0	$-f_{Ins} + f_{Com} + f_{Ter}^{U}$
S	0	$-f_{Pun} + f_{Ins} - f_{Ter}^{U}$
F	0	$f_{Pun} - f_{Com}$

SP misbehaves. So we consider only two cases: 1) S behaves in full compliance with price policies, and 2) S misbehaves by not following the tied price policies.

Table 2 presents the amount that is paid out to the involved parties. According to Table 3, we aggregated into the two cases.

Regarding the affected user U, we observe that in the case of personalized price being detected, U receives an additional $f_{Com} + f_{Ter}^{U}$ than it would if no personalized price was detected. In order for U to profit, we require $f_{Ins} < f_{Com} + f_{Ter}^{U}$. Since we know from Equation (3) that $0 \leqslant f_{Ter}^{U}$ and Equation (4) that $f_{Ins} < f_{Com}$, it ensures a positive compensation for affected users.

Regarding S, we observe that for the S to lose money due to possible misbehavior in the same case, we require $f_{Ins} < f_{Pun} + f_{Ter}^{U}$. Again, since $0 \leqslant f_{Ter}^{U}$ and we want a loss of money for all possible values of f_{Ter}^{U}, we obtain the stronger inequality $f_{Ins} < f_{Pun}$ which ensures the deterrence of misbehavior for this scenario. However, this constraint is subsumed by Equation (4), which sets a tighter bound on f_{Ins}.

Finally, to avoid collusion attacks in this scenario, we consider that the parties except S receive a positive reward. We observe that although U profits in the case of a rogue order being detected, if we sum the rewards of U, and S, the result is $-f_{Pun} + f_{Com}$. From Equation (4), we also know that $f_{Com} < f_{Pun}$, and the result < 0, thus a collusion between S and other related parties does not profit.

5.3 Feasibility Analysis

Since game theory is the logical analysis of situations of conflict and cooperation, it is appropriate to use game theory to analyze the feasibility in our proposed RHS settings. We construct an economic model to analyze the acceptance of Spas. The interactions among SPs and users are modeled as a non-cooperative game since the objective of each party is to maximize its own profit. We assume that all players have preferences, beliefs and common knowledge about the world (including the other players), and try to optimize their individual payoffs. Also, players are aware that other players are trying to optimize their payoffs. The payoff functions of all system players are specified based on their interactions. The pricing strategies applied in our analysis is pay-per-use, as often adopted in practice.

In Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2020.3008315, we consider the possibility that Users may have private information about their own payoffs and their type (incomplete information game). We introduce two different types of User, namely Price Insensitive User and Normal User. We find that the analysis based on incomplete information game shows similar and consistent conclusions to those in the main analysis.[1]

For easy presentation and understanding, we summarize the main notations used in this paper in Table 4 with nomenclature. We now explain the fundamental elements as follows.

TABLE 4
Notation With Nomenclature

Field	Use
$s, \boldsymbol{S}$	The parameters related to SPs;
$u, \boldsymbol{U}$	The parameters related to users;
$PO_{x}^{0,1}$	The payoff function of x, where x can be s or u;
IN_{s}^{0}	The income that s get from the per RHS order when using uniform pricing;
IN_{s}^{1}	The income that s get from the per RHS order when using personalized pricing;
CO_{s}	The cost that s generated per RHS order;
LO_{u}	The difference between the personalized price and the uniform price for a same order, or the additional loss for a user receiving personalized pricing compared to receiving uniform pricing;
f_{Ins}	The cost for purchasing a *Fair Price Insurance*;
f_{Pun}	The punishment charges for an s caught misbehaving;
f_{Com}	The compensation received by the affected s once a rogue order is detected;
β	$0 \leqslant \beta \leqslant 1$, the probability of u plays No Insurance strategy, or the proportion of orders in which u purchases *Fair Price Insurance* within Spas;
θ	$0 \leqslant \theta \leqslant 1$, the probability of s plays Personalized Pricing strategy, or the proportion of orders in which s applies Personalized Pricing strategy.

5.3.1 Player

We term the players in Spas as $\mathbb{R} = \{\boldsymbol{S}, \boldsymbol{U}, \boldsymbol{A}\}$, where $\boldsymbol{S}, \boldsymbol{U}$, and $\boldsymbol{A}$ denote the sets of SPs, users, and authorities, respectively. We only consider SPs and users in our game because the purpose of the authorities is mainly to monitor and regulate the market but not to look for profits. To implement the game theory analysis, we assume that all SPs and all users are rational decision makers to increase their profit. In the game theory model, the first player is s, which is a random selected SP from $\boldsymbol{S}$. The second player is u, which is a random selected users from $\boldsymbol{U}$.

5.3.2 Strategy

SPs. There are two pricing strategies for each SP, personalized pricing or uniform pricing. Any SPs can choose to either behave correctly by following the agreed uniform pricing policies updated by *PPC*, or apply personalized

1. The conclusion in the Appendix, available in the online supplemental material, shows: if the Price Insensitive User entirely could not feel the difference in utility of different monetary amount, in a long-run when there is an equilibrium, SP plays Fair Pricing, Price Insensitive User purchases the insurance and Normal User does not purchase the insurance. If the Price Insensitive User could still feel the difference in utility of different monetary amount (but less sensitive as Normal User), in the equilibrium, the preference of Normal User to purchase the insurance is extremely low, the preference of SP to apply personalized pricing is approximately zero.

pricing to its users. In the later case, if the SPs were detected, they would get punished and lose a large sum of money.

User. For the users in RHS, since they regard price discrimination as unfair or manipulative, we assume that all the users do not like or accept personalized pricing strategies. The users may also choose to purchase a *Fair Price Insurance* within Spas or not. If so, the user will surely detect personalized pricing when the corresponding SP misbehaved, and get compensation for that affected order. Nevertheless, if a user does not purchase the insurance, even though SP uses personalized pricing, the user has no efficient way to detect personalized pricing and might pay more money for that order if he is categorized as a high spender.

5.3.3 Payoff

Payoff of SPs. We employ $PO_s^{0,\gamma}$ to present the payoff function of s that applies uniform pricing strategy. γ can be 0 or 1. If γ = 0, it means that the user does not purchase *Fair Price Insurance* within Spas. Else when γ = 1, the user does. In general, if the user does not choose to purchase insurance, the payoff of s is the difference between the total income it receives and the total cost. We denote IN_s^0 as the income unit when using a uniform pricing strategy, and CO_s as the cost unit. Then

$$PO_s^{0,0} = IN_s^0 - CO_s. \tag{5}$$

If u does purchase insurance, and the insurance fee is f_{Ins}, then we have

$$PO_s^{0,1} = IN_s^0 - CO_s + f_{Ins}. \tag{6}$$

When personalized pricing is applied, once personalized pricing is detected, the affected SP will eventually get punished by losing f_{Pun} amount of money per order. We denote that IN_s^1 as the income unit when using personalized pricing strategy. Then, we conclude the payoff function of s with personalized pricing strategy in the following equations, when u does not choose to purchase insurance, we have

$$PO_s^{1,0} = IN_s^1 - CO_s. \tag{7}$$

When u does purchase insurance, we have

$$PO_s^{1,1} = IN_s^1 - CO_s + f_{Ins} - f_{Pun}. \tag{8}$$

Payoff of Users. u uses RHS at s. Let $PO_u^{\lambda,0}$ denotes its payoff function when u does not purchase the *Fair Price Insurance* within Spas. λ can be 0 or 1. When λ = 0, it means that s does not apply personalized pricing strategy to the user. Else when λ = 1, it means that s applies personalized pricing to u. We also assume that the additional loss for u receiving personalized pricing compared to receiving uniform pricing is LO_u, we have

$$LO_u = IN_s^1 - IN_s^0. \tag{9}$$

When s does not apply personalized pricing but use uniform pricing strategy to the user, then we have

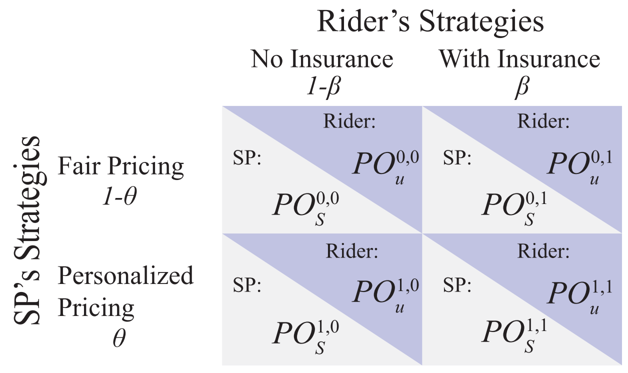


Fig. 6. The payoff bi-matrix of Game Theory in Spas.

$$PO_u^{0,0} = -CO_u. \tag{10}$$

When s applies personalized pricing to the user, we have

$$PO_u^{1,0} = -(CO_u + LO_u). \tag{11}$$

When u purchased the insurance, if s applies personalized pricing on u, that rogue order will be detected and then u will get compensated of f_{Com} amount of money per order. Let $PO_u^{\lambda,1}$ denote its payoff function when u does not purchase the *Fair Price Insurance* within Spas. Therefore, we have

$$PO_u^{0,1} = -(CO_u + f_{Ins}) \tag{12}$$

$$PO_u^{1,1} = -(CO_u + LO_u + f_{Ins} - f_{Com}). \tag{13}$$

5.3.4 Which Strategy to Choose

In our RHS settings, the SPs and Users are always looking for utility maximization. We assume linear utility function ($U(x) = x$) applied for both SPs and Users. We will demonstrate the possible optimal approaches by using the game bi-matrix (shown in Fig. 6). The approaches allow the players to decide which strategy they will choose.

A pure strategy Nash Equilibrium is a strategy profile in which no player would benefit from a change of pure strategy, given that the other participant's strategy remains unchanged. Given the payoff matrix in Fig. 6, in a single-stage game, there does not exist a pure strategy Nash Equilibrium. There is no stable combination of choice for both players.[2] A mixed strategy Nash equilibrium exists when at least one player uses a randomized strategy and the other player would not benefit from playing an alternate (randomized) strategy. in a long-run repeated games, when both players apply random strategies to their choices, there will exist a mixed strategy Nash Equilibrium.

In a mixed strategy Nash Equilibrium, s plays the Fair Pricing and Personalized Pricing with probability (1-θ, θ) $(0 \leqslant \theta \leqslant 1)$, respectively. u plays No Insurance and

2. The payoff matrix in Fig. 6 shows that, if the user chooses No Insurance, SP will choose Personalized Pricing. When SP chooses Personalized Pricing, the user will prefer With Insurance. When the user's choice is With Insurance, then the SP will choose Fair Pricing. If the SP chooses Fair Pricing, the user chooses No Insurance. Therefore, in this case, there does not exist a pure Nash Equilibrium.

Insurance with probability (1-β, β) ($0 \leqslant \beta \leqslant 1$), respectively. These probabilities could also be interpreted as follows. s applies personalized pricing to θ proportion of the orders, so the uniform pricing orders' proportion will be 1- θ. In addition, u purchases insurance for β proportion of his/her orders, while not for the rest 1 - β proportion of orders.

Theorem 1. *Suppose both s and u are rational players and they know that the other player is rational (rational player assumption), $f_{Pun} \gg LO_u$ (the punishment charges for s caught misbehaving is far greater than the difference between the personalized price and the uniform price for the same order), then in a long-run when there is an equilibrium, the preference of user to purchase the insurance is extremely low. When rational players assumption holds, and $f_{Com} \gg f_{Ins}$ (the compensation received by the affected s once a rogue order is detected is far larger than the cost for purchasing a* Fair Price Insurance*), then in a long-run when there is an equilibrium, the preference of s to apply personalized pricing is extremely low.*

Proof. The proof is based on the definition of mixed strategy Nash Equilibrium. In this equilibrium, given the strategy of u, s should be indifferent between playing Fair Pricing and Personalized Pricing. In other words, the s's expected payoff of playing Fair Pricing must be equal to that of playing Personalized Pricing. We have

$$(1-\beta) \times PO_s^{0,0} + \beta \times PO_s^{0,1} = (1-\beta) \times PO_s^{1,0} + \beta \times PO_s^{1,1}.$$

By substituting the Equations (5), (6), (7), (8) and (9) into the above equation, we obtain

$$\beta = \frac{LO_u}{f_{Pun}}.$$

When $f_{Pun} \gg LO_u$ (in reality, SPs always have to pay hefty fines when they are found misbehaving which is far greater than difference between the personalized price and the uniform price for the same order), $\beta \approx 0$. It can be interpreted as in a long-run when there is an equilibrium, the preference of user to purchase the insurance is extremely low, and the theorem is proved. □

In this equilibrium, given the strategy of s, u should be indifferent between playing No Insurance and With Insurance. The u's expected payoff of playing No Insurance must be equal to that of playing With Insurance. We have

$$(1-\theta) \times PO_u^{0,0} + \theta \times PO_u^{1,0} = (1-\theta) \times PO_u^{0,1} + \theta \times PO_u^{1,1}.$$

By substituting the Equations (9), (10), (11), (12) and (13) into the above equation, we obtain

$$\theta = \frac{f_{Ins}}{f_{Com}}.$$

When $f_{Com} \gg f_{Ins}$ (in reality, insurance compensation is far greater than the cost of purchasing insurance), $\theta \approx 0$. It can be interpreted as in a long-run when there is an equilibrium, the preference of s to apply personalized pricing is approximately zero, and the theorem is proved.

Theorem 2. *When Theorem 1 holds, then the expected payoff for s and u is close to that of the theoretical best situation in terms of social welfare.*

Proof. In terms of social welfare, the theoretical best situation is that s do not apply personalized pricing and u do not purchase a *Fair Price Insurance* within Spas. Put differently, it is ideal to prevent s from applying personalized pricing without any additional insurance premium cost paid by u. The payoff under this best situation for s and u is

$$(IN_s^0 - CO_s, CO_u).$$

The expected payoff of s and u in the long-run equilibrium is

$$(IN_s^0 - CO_s + \beta \times f_{Ins}, CO_u + \theta \times LO_u).$$

Since $\beta \approx 0$ and $\theta \approx 0$, the expected payoff of long-run equilibrium is quite close to that of the theoretical best situation. The differences are that in the expected payoff, s would gain an additional extremely small amount profit from insurance sales ($\beta \times f_{Ins}$), and u would bear an additional negligible cost due to seldom personalized pricing ($\theta \times LO_u$). □

In conclusion, our theoretical analysis implies that in a long-run, although the preference of users to purchase the insurance is quite low, the implementation of Spas in RHS still acts as a great deterrence against possible SP misbehavior of applying personalized pricing to its users. The expected payoff for both SPs and users is close to that of the theoretical best situation in terms of social welfare.

6 Evaluation

In this section, we first determine reasonable values for system-wide parameters based on the analysis of the current legislation in the related area. We also generate real order dataset based on the real trip dataset provided by Didi Chuxing GAIA Initiative [9], according to which we analyze the trip duration and price in the different time frame in a day. Finally test the performance of Spas in Hyperledger Fabric [26].

6.1 System Parameter Values

To get an estimate of sample reaction program payout values, we take Law of the Peoples Republic of China on the Protection of Consumer Rights and Interests (LPPCRI) [7] as the primary reference and Interim Measures for the Administration of Online Taxi Booking Business Operations and Services (IMAOTBBOS) [8] as the secondary reference.

In respect of the 55th clause in LPPCRI, if the operator provides fraudulent acts on services, it shall increase the compensation amount for the losses suffered by the users. The amount of compensation shall be three times the price that the services received by the customer. Moreover, the compensation amount should be no less than RMB 500. Also, we analysed the order dataset and found that the average fare for a user is around RMB 20 (Fig. 7 shows the order price distribution from the datasets).

Further, we surveyed 50 persons on the price of their willingness to pay for *Fair Price Insurance*, and the results showed that the average amount for each order they can afford is around RMB 0.5, so the price of the insurance f_{Ins} we would set it to RMB0.5.

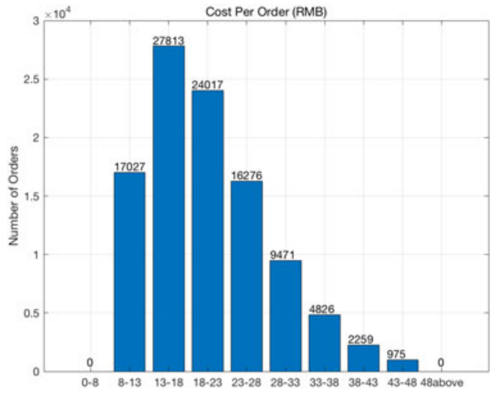


Fig. 7. Cost distribution.

Moreover, the total amount of the compensation for the affected users would be RMB500 or 3 times of the correct fare whichever is greater. For example, if the correct fare for the user is RMB240 and this order was found rogue, the affected users would get compensation of RMB720 (3 times of RMB240). So the payout value for the affected users f_{Com} is RMB500 or 3 times of the correct fare amount whichever is greater.

In accordance with IMAOTBBOS, if the SP were found fraudulent, the government would impose a fine of RMB5000 to RMB10000 on the affected SP. So in order to better regulate SP's behavior, we would set the punishment fee f_{Pun} for SP's misbehavior to RMB10000.

Also, we found the reasonable price f_{Ins} for the *Fair Price Insurance* is RMB1 after referring to the Grab applications policy [23],according to which, the registered driver has to pay for an SGD0.4 per order to the government.

Moreover, to ensure that the *Global Account* has sufficient funds, each registered SP should put RMB1, 000,000 to *Global Fund Account* as a deposit *Deposit* when registered. The reason for setting *Deposit* to this amount is because being a ride-hailing SP, it should be solvent.

The list all the preset parameter values in Spas are $Deposit$ = RMB1, 000,000, f_{Com} = RMB 500, f_{Ins} = RMB1, and f_{Pun} = RMB10, 000. From which we can see that the constraints from Sections 4.7 and 5.2 are easily satisfied. In addition, the insurance literature is a well-studied research area, and a variety of tools can be used in Spas to calibrate insurance prices. For example, Spas could delegate a hired actuarial consulting firm to design the punishment and the rewarding of the insurance, and determined the premiums by using the Generalized Linear Models (GLM) [21]. This technique is widely used in non-life insurance pricing.

6.2 Experiment and Performance Evaluation

We instantiated Spas prototype by using Hyperledger Fabric [26], a wildly used alliance blockchain framework on which we can deploy smart contract(as known as *Chaincode* in Fabric). We notice that Spas is not constrained to Fabric, and it can also be deployed on other types of blockchains such as Ethereum [20] and so on. Fabric is a complex distributed system. Its performance depends on many parameters including the complexity of the distributed application, transaction size, consensus implementation and their parameters, the network parameters and topology of nodes in the network and many other factors. In our experiment, we mainly focus on testing the viability of Spas to be applied in the real world. We consider two variables:(1) number of transactions per block and (2)number of v-CPUs to test on the impact on the throughput and latency.

TABLE 5
Modified Data Structure

Raw Data	Example Format
Driver ID	glox7rrlltBMv
Order ID	jkkt8kxniov
Order Starting Time	1501584540
Order Ending Time	1501584540
Total Distance(km)	15.4
Total Fare(RMB)	35.5
Time Traveled in 0:00-9:00 & 17:00-24:00(Off-Peak)	10.2
Time Traveled in 9:00-17:00(Peak)	10.2
Distance Traveled in 0:00-9:00 & 17:00-24:00(Off-Peak)	5.6
Distance Traveled in 9:00-17:00(Peak)	5.6
Extra Fee	10

Datasets. Since there is no real order dataset that we can obtain from online, we generate an order dataset from the historical trajectory dataset provided by Didi Chuxing GAIA Initiative [9]. The original dataset consists of carpool service's trajectory data of Xi'an and ChengDu city covering October 1, 2018 to December 1, 2018. The interval of each labelled data is 3 seconds, and the information of the data includes Driver ID, Order ID, Timestamp, Latitude, Longitude. From the raw dataset, we generated the modified dataset which will be used in our experiment. The structure of modified data are shown in Table 5.

As shown in Figs. 8a and 8b, we first analyze the trip duration and price in a different time frame in a day. By choosing three different trip distances, 10 km, 20 km and 30 km respectively, we found that when testing on trips with the same distance, the trip duration varies when starting a trip at different timing of a day, which indicates the complexity of pricing under the RHS context.

Setup. We use the following setup in our experiment: (1) nodes run on Fabric version v2.0 instrumented for performance evaluation through local logging, (2) nodes are hosted locally as dedicated VMs interconnected with 10-Gbps-throughput network, (3) all nodes are Intel(R) Core (TM) i5-4460 CPU @ 3.2 GHz with 4-vCPU VMs running Ubuntu 18.04 with 8 GB of RAM and SSDs as local disks, (4) two different organizations (orgs) to represent SPs or other government agencies, each organization consists of one Certificate Authority(CA) node and two peers, one is endorser node, and the other is orderer node which runs a typical Kafka orderer with a ZooKeeper service.

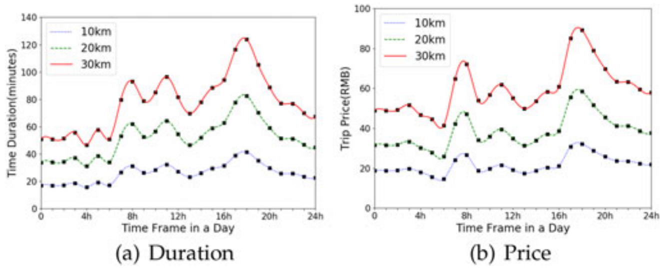
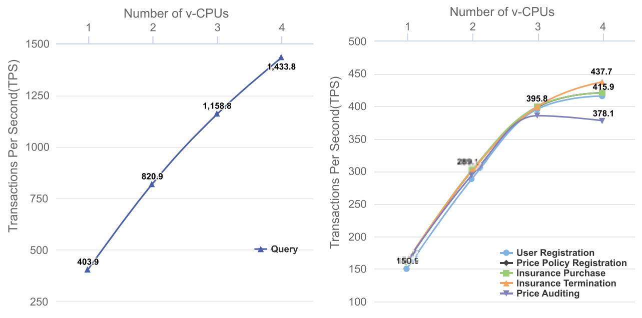
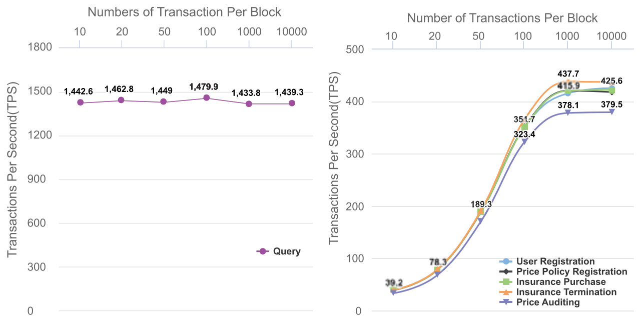


Fig. 8. Trip during and price in different time frame in a day.



(a) Query operation (b) Main operations

Fig. 9. Throughput varies by number of v-CPUs.



(a) Query operation (b) Main operations

Fig. 10. Throughput varies by number of transactions per block.

In real-world scenario, there could be many organizations in the network, for simplicity, we use two different organizations during the experiments. CA nodes are responsible for issuing certificates to the participants, therefore, the clients can access to the peers. Peers are responsible for interacting with the clients, when a peer receives a proposal from the clients, an endorse peer will executed the corresponding chaincode and verify the endorsement policy on its ledger, then the endorsement result and proposal will be packaged and sent to the orderer service. After ordering via Kafka consensus between orderers across all organizations, the proposal will be written into the block as a transaction. When the block is ready to be accepted by blockchain network, it will be broadcasts to all peers in the network and then be executed on each peer.

6.2.1 Experiment 1: Impact of Peer CPU

Spas peers run many CPU-intensive cryptographic operations. To estimate the impact of CPU power on throughput and latency, we performed a set of experiments in which 2 peers run on 1 to 4 vCPU VMs. Fig. 9 shows the evaluation results, for blocks containing query (Fig. 9a) and various operations in Spas (Fig. 9b). For those operations, the measured throughput and latency scales in the same way with the number of vCPUs increases. We can observe that the query throughput reaches over 1,430 transactions per second(tps) when there are 4 v-CPUs. For the various operations in Spas, Price Auditing has the lowest throughput because the complexity of *PAC* is higher than any other operations in Spas. However, it can still achieve around 400 tps, which performs much better than the popular blockchain network Bitcoin (7 tps) or Ethereum (40 tps). Furthermore, the validation performance scales quasi-linearly with the number of vCPUs increases, as the endorsement policy verification is parallel.

6.2.2 Experiment 2: Impact of the Number of Transactions Per Block

To estimate the impact of the number of transactions per block on throughput and latency, we use 4 v-CPUs and performed a set of experiments in which we set the number of transaction in each block to 10, 20, 50, 100, 1,000, 10,000 and measure the performance of the network.

Fig. 10 shows the evaluation results, for blocks containing query (Fig. 10a) and multiple operations in Spas (Fig. 10b). For those operations, the measured throughput and latency scale in the same way with the number of transactions per block (tpb) increases. We can observe that as the number of tpb increases, the query throughput maintains a high level around 1,450 tps. However, when executing the listed operations, they all increase sharply from around 40 tps to around 400 tps as the number of tpb increases from 10 to 1,000, but as the number of tpb increase from 1,000 to 10,000, the throughput does not change much. It is because that the initiation phase has a fixed time overhead, as the number of tpb increases, the impact raised by the initiation phase will get smaller to each transaction. In addition, we also observe that price auditing operation is the lowest in throughput as it is the most complex one among all the operations as discussed earlier.

For the latency, as shown in Table 6, there is no much difference between different operations except of the query operation which is within 0.1s. For most operations in Spas the latency is around 27s at 20 tpb which reaches the maximum, however, as the number of tpb increases from 20 to 10,000, the latency decreases from 27.06s to 1.24s, which can be regarded as viable in practice since the bitcoin network takes 10 minutes to confirm a transaction.

7 Related Work

In this section, we discuss work related to Spas. We cover three main areas: insurance schemes, blochchain-based ride-hailing services and incentive mechanism.

TABLE 6
Latency(second) Comparison Between Different Operations When Number of Transactions per Block(TPB) Changes

Number of TPB	User Registration	Insurance Purchase	Insurance Terminate	Price Policy Registration	Price Auditing	Query
10	12.14	11.73	11.73	11.74	11.52	0.01
20	27.06	26.98	26.59	26.91	26.26	0.01
50	8.46	8.43	8.26	8.48	8.39	0.01
100	2.47	2.45	2.15	2.49	2.54	0.01
1000	1.29	1.27	1.00	1.33	1.27	0.01
10000	1.24	1.31	1.06	1.39	1.30	0.01

7.1 Insurance Schemes

Insurance schemes are broadly used around our life, both in traditional and modern settings. Traditional applications such as life insurance and car insurance are ubiquitous in one's life. Recently, some insurance applications such as flight delay insurance, delivery food delay insurance have already entered into everyone's life. For example, Eleme [16] allows customers to purchase a delivery delay insurance at placing order stage with a little money and get compensated if the delivery was delayed. Other types of insurance including electronic insurance policies for service evaluation in distributed systems [17] and authentication metric that followed good design principles [18]. However, both of these proposals offer little accountability and automation. Due to the complexity of the RHS systems pricing algorithms, the insurance scheme mentioned above could not directly apply to the price auditing as these methods do not allow auditing publicly.

Spas is the first to introduce the idea of insurance into price policies as a means of balancing SP control and liability. Spas also utilised smart contract technology to ensure auditing publicly and executing transactions automatically.

7.2 Blockchain versus Ride-Hailing Services

In the past decades, many scholars have proposed blockchain-based ride-hailing applications. Among those works, most are focusing on providing a decentralized RHS platforms in a privacy preserving manner which enable the rider and the driver interact with each other without a centralized service provider. Z. Li *et al.* [42] proposed a privacy-preserving collaborative ride-hailing service named CRide. CRide uses private proximity test, privacy-preserving query processing to protect users' privacy, it also constructs a consortium blockchain among different RHS platforms to record collaborative rides. Similarly, Y. Kanza *et al.* [43] proposed a decentralized RHS that allows the ride requests, matching a driver with a ride, the payment are all via blockchain, it also does not reveal any private information of the user. Moreover, M. Li *et al.* [41] proposed FICA using blockchain-assisted vehicular fog computing, which supports conditional privacy, one-to-many matching, destination matching, and data auditability. Furthermore, there are several cryptocurrency-based decentralized ride-sharing efforts either currently in development or that have been developed and are in the market as of right now, such as [33], [34], [35], [36], [37], [38], [39], [40]. These projects are similar in design to the related works described above. The main difference between Spas and the above mentioned related work is that Spas is aiming to address the price discrimination issue, while the aforementioned works are aiming to construct a decentralized ride-hailing service platform. Also, Spas only introduces a decentralized auditor but not make the entire ride-hailing service in a decentralized way.

Meanwhile, some scholars have proposed works which are focusing on the fair-trading. M. Baza *et al.* [44] proposed to decentralize ride sharing services using the revolutionary public blockchain named B-Ride. Despite that B-Ride guarantees its users privacy, it also proposed a reputation management system to track drivers' behaviour which allows them to behave honestly in the system. Otherwise, they will not be selected for future trips. Also, the rider will have a trip and the driver get the fare in a trust-less environment using the pay-as-you-drive methodology. In addition, Zhang, H. *et al.* [45] presented a secure billing protocol which allows a driver and a passenger to count the fare together by arranging them to reach an agreement on the route. If one party tries to cheat the other party will be informed. As all the ride information are storing in the blockchain. The ride information can be provided as a proof to governmental agencies in case of reputation ratings or criminal investigations. Although works [44] and [45] addressed billing issues, they still cannot solve the price discrimination behaviour. Therefore, among all the blockchain-base ride hailing service related works, Spas is the only approach to efficiently address price discrimination issue.

7.3 Incentive Mechanism

Incentive mechanism has traditionally been one of the essential dimensions of information systems design, together with software engineering and user-acceptance [46]. In most cases, the proposed system incentivized agents to improve their performance to achieve the desirable goal of the system.

Among all related approaches, Vafeas *et al.* [47] focus on investigating incentive mechanism for rational miners to purchase the computational resources. To find the optimal incentive for the edge service provider(ESP) and miners to choose auto-fit strategies, Vafeas *et al.* formulate a two-stage Stackelberg game between the miners and ESP, they also investigate Stackelberg equilibrium of the optimal mining strategy under two different mining schemes. Also, Ma *et al.* [48] proposed a blockchain-based mechanism for fine-grained authorization in data crowd-sourcing (BC-FGA-DCrowd). To attract the data trading platform users, the authors use a public blockchain to implement cryptocurrencies and payment services as the incentive mechanism. Meanwhile, to break islands of knowledge and make knowledge tradable in edge-AI enabled IoT, Lin *et al.* [49] proposed a blockchain-based P2P knowledge market for knowledge paid sharing. They also proposed a non-cooperative game based optimal knowledge pricing strategy as incentives to encourage sellers to learn more data under a fixed budget. Moreover, Chen *et al.* [50] proposed a secure and efficient blockchain-based data trading approach for the Internet of Vehicles (IoV). To encourage data exchange and sharing, the proposed work give an extra reward to the broker with the greatest contribution of data sharing during a certain period as an incentive to solve PoW. In addition, another work named DeepChain [51] provided a value-driven incentive mechanism based on Blockchain to force the participants to behave correctly. Furthermore, Zhou *et al.* [52] developed a consortium blockchain-based secure energy trading mechanism for V2G. Under distributed deep learning context, to attract more parties actively involving in collaborating training. They proposed an efficient incentive mechanism based on contract theory.

Similarly, Spas's goal of using incentive mechanism is pretty much the same as the listed approaches above. Although the mechanism designs are varying from one to another, similar to Spas's, they all defined the incentive value to a certain range by setting constraints of the correlating factors.

8 Discussion and Limitations

We next discuss the generic application of Spas and the fundamental limitations that Spas does not adequately address or that present room for improvement.

8.1 Generic Application of Spas Framework

Owing to the rapid development of internet technology, people are spending increasingly large portions of their time on cyberspaces (e.g., online services, e-commerce, online games, etc.). As mentioned at the beginning of this paper, it is because the nature of RHS's high complexity during price generating and its non-transparency to audit the price, we propose Spas to address the price issues in RHS. With online platforms such as Didi Chuxing [22], Ctrip, and Amazon been exposed to "big data killing" behavior, the Spas framework can also be implemented into the applications listed above, such as e-commerce, online games etc.

8.2 Scalablity of Spas

For scalability, Spas can achieve over 400 transactions per second. Since Spas uses blockchain to process registration transactions and insurance transactions, we will focus on the scalability for these two types of transactions. For registration transactions, although there are huge number of users, the number of new registered users per day is not many for a city with millions population. As a result, we believe that the performance of our scheme is sufficient to handle the registration transactions. For the insurance transactions, it depends on the number of users that have actually purchased the insurance. We consider the worst-case that all the users purchased the insurance. Considering a large city where two million orders are generated per day (according to Didi Chuxing market report), the average insurance transactions that need to be processed is less than 30 per second. As a result, we believe that the performance of our scheme is sufficient to handle insurance transactions. Moreover, as the blockchain technology develops, many scholars have proposed approaches to address the scalability issue, these approaches include sharding [53], side chains [54], algorand [55], plasma [56], stella [57], and so on. Therefore, the scalability would not be a bottleneck in our scenario.

In addition, if Spas is deployed in public blockchain settings, for example Ethereum [20], the worst case of processing insurance transactions requires 30 transactions per second. Since Ethereum can achieve 40 transactions per second, it also can meet the performance requirement of Spas. However, as the Ethereum requires its participants to consume gases to process the different operations, it increases the user's costs. Moreover, public blockchain allows anyone to join the network which also brings data privacy risks. In summary, we suggest to use permissioned blockchain in Spas.

8.3 Limitations

The Correctness of the Location Data. The accuracy of the sensors can affect the correctness of the location data. In Spas, we assumed that all the sensing data obtained from mobile devices were correct and accurate, however, in a real situation, these data are not always accurate because of the bias and noises.

Driver and Rider's Collusion. The *PAC* will verify the correctness of the trajectory data collected from the driver's and the rider's mobile devices, however, if the driver and the rider colludes and transmits the manipulated trajectory information to the *PAC*, the rider will get compensated and eventually profit. It seems a limitation for current Spas system, but in fact, the matching process between a driver and a rider is randomized which makes the driver and the rider entering a temporary employment relationship, therefore the probability for them to collude is extremely low. In addition, as electrical vehicles (EVs) are getting popular, their built-in GPS system enables the users can tracking their EVs at any time. Spas can also further enhance the two-factor authentication to three-factor by adding the EV's built-in GPS sensor in addition to the driver's and rider's mobile devices. Therefore, the difficulty of collusion becomes even higher.

Other Forms of Misbehavior. In this work, we only consider the misbehavior raised by price discrimination. SP could also misbehave from many other aspects such as issuing a driving permit to an unqualified driver and so on. We will consider other misbehaviors in our future work.

Privacy of the Users. Though Spas offers privacy and anonymity for the users, it only offers limited protection for users' name or ID. However, the transparency of Spas makes it possible for the user to get *inference attacks*[4], where an adversary analyses a user's public accessible order data to gain knowledge about his personal data illegitimately. In addition, in the existing permissioned blockchain settings, it is securer than the public blockchain because not every entity can join the network, so the private data of riders and drivers can only be accessed by the authorized parties. Although the sensitive data can be processed before storing (such as use identifies instead using user's real identity, adding noise to the current data, and etc.), it can mitigate the negative effects if the information is leaked. Also, there are also many scholars who have many efficient solutions mainly focus on privacy-preserving and information leakage issues, for example the redactable blockchain supports rewriting operations of the blockchain which enables sensitive data removal in the later stage, all those works [41], [42], [43] are orthogonal to our approach and they also can be integrated into Spas. Current Spas system makes no effort to address this, and we may explore encrypted data processing in our future work. Futhermore, Spas does not protect the SPs' privacy either. One could argue that such transparency and auditability of the SPs may be useful for the market as a whole, but this may not be desirable for SPs who want to conceal their order volume. We defer this potential improvement to future work.

9 Conclusion

In this paper, we propose Spas, a platform for price auditing in responding to SP's price discrimination misbehavior in a decentralized and automated setting. We described the full process from registering a user to claiming reaction payouts. We introduce a novel type of insurance to the user, with which the insured user can have his RHS order automatically audited upon the order completion. We also develop a model to explain reaction payouts, which help us discover the constraints to formulate the reasonable reaction policies better. We further analyze the security guarantee of Spas and demonstrate the feasibility of Spas through game theory analysis. Moreover, we discussed the deployability incentives and created a decentralised instantiation of Spas authority based on Hyperledger Fabric. Although our work does not stop all misbehaving SPs, we believe only by punishing SP's misbehavior, should the users have their financial interests better protected in a more transparent RHS industry. Moreover, we argue that Spas is the first concrete step towards that final goal.

Acknowledgments

This work was supported by the National key R&D Program of China under Grant No.2018YFB1402700. It was also partially supported by the National Natural Science Foundation of China under Grant No. 61672412 and No. 61602363.

References

[1] A. Pham, I. Dacosta, G. Endignoux, J. R. Troncoso-Pastoriza, K. Huguenin, and J.-P. Hubaux, "ORide: A privacy-preserving yet accountable ride-hailing service," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 1235–1252.

[2] Is your friend getting a cheaper Uber fare than you are?. Apr. 13, 2018. [Online]. Available: https://www.theguardian.com/commentisfree/2018/apr/13/uber-lyft-prices-personalized-data

[3] The ride hailing trend: Past, present, and future overview of ride hailing. Nov. 26, 2018. [Online]. Available: http://movmi.net/ridehailing-trend-overview/

[4] C. Li, H. Shirani-Mehr, and X. Yang, "Protecting individual information against inference attacks in data publishing," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2007, pp. 422–433.

[5] A. Acquisti, "Price discrimination, privacy technologies, and user acceptance," in *Proc. CHI Workshop Personalization Privacy*, 2006, pp. 1–3.

[6] N. Szabo, "Smart contracts: Building blocks for digital markets," 1997. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/ CDROM /Literature/LOTwinterschool2006/szabo.best.vwh.net/ smart_contracts_2. html

[7] Law of the people's Republic of China on the protection of consumer rights and interests (LPPCRI) (2013 Amendment). Mar. 15, 2014. [Online]. Available: http://en.pkulaw.cn/display.aspx?cgid=211792&lib=law

[8] Interim measures for the administration of online taxi booking business operations and services (IMAOTBBOS). Jul. 27, 2016. [Online]. Available: http://lawinfochina.com/display.aspx?id=22963&lib=law

[9] DiDi brain intelligent driving smart transportation AI labs. Accessed: Jan. 15, 2020. [Online]. Available: https://outreach.didichuxing.com/appEn-vue/Register?path=register

[10] Ethereum homestead documentation. Mar. 01, 2017. [Online]. Available: http://www.ethdocs.org/en/latest/

[11] M. Risius and K. Spohrer, "A blockchain research framework," *Bus. Inf. Syst. Eng.*, vol. 59, pp. 385–409, 2017. [Online]. Available: https://doi.org/10.1007/s12599-017-0506-0

[12] T. Miettinen and R. Stenbacka, "Personalized pricing versus history-based pricing: Implications for privacy policy," *Inf. Econ. Policy*, vol. 33, pp. 56–68, 2015.

[13] J. Turow, L. Feldman, and K. Meltzer, "Open to exploitation: America's shoppers online and offline," Report of the Annenberg Public Policy Center, University of Pennsylvania, Philadelphia, 2005.

[14] J. Turow, J. King, C. J. Hoofnagle, A. Bleakley, and M. Hennessy, "Americans reject tailored advertising and three activities that enable it," 2009. [Online]. Available: https://ssrn.com/abstract=1478214

[15] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI around with decentralized automated incentives," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 410–426.

[16] Eleme mobile application. Accessed: Jan. 16, 2020. [Online]. Available: https://www.ele.me/home/

[17] C. Lai, G. Medvinsky, and B. C. Neuman, "Endorsements, licensing, and insurance for distributed system services," in *Proc. 2nd ACM Conf. Comput. Commun. Secur.*, 1994, pp. 170–175.

[18] M. K. Reiter and S. G. Stubblebine, "Authentication metric analysis and design," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 138–158, May 1999.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Whitepaper*, 2008.

[20] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *White Paper*, 2015.

[21] E. Ohlsson and B. Johansson, *Non-Life Insurance Pricing With Generalized Linear Models*. vol. 2. Berlin, Germany: Springer, 2010.

[22] Didi chuxing application. Accessed: Jan. 15, 2020. [Online]. Available: https://www.didiglobal.com/

[23] Grab mobile application. Accessed: Jan. 15, 2020. [Online]. Available: https://www.grab.com/sg/

[24] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Des. Implementation*. 1999, vol. 99, pp. 173–186.

[25] F. Z. Borgesius and J. Poort, "Online price discrimination and EU data privacy law," *J. Consum. Policy*, vol. 40, no. 3, pp. 347–366, 2017.

[26] Hyperledger fabric. Accessed: Jan. 16, 2020. [Online]. Available: https://www.hyperledger.org/projects/ fabric

[27] Y. GE, C. R. Knittel, D. Mackenzie, and S. Zoepf, "Racial and gender discrimination in transportation network companies," Tech. Rep., National Bureau of Economic Research, Last visited: Jan. 2017. 2016. [Online]. Available: http://www.nber.org/papers/w22776

[28] K. J. Strandburg, "Free fall: The online market's consumer preference disconnect," University Chicago. Legal Forum, p. 95, 2013.

[29] Online targeting of advertising and prices. A market study. Accessed: Jan. 15, 2020. [Online]. Available: http://webarchive.nationalarchives.gov.uk/20140402142426/

[30] General data protection regulation (GDPR)–Official legal text, Accessed: Jan. 12, 2020. 2020. [Online]. Available: https://gdpr-info.eu/

[31] E. Prisner, "Game theory through examples," *Math. Assoc. America*, 2014. Accessed: Jan. 12, 2020. [Online]. Available: http://www.jstor.org/stable/10.4169/j.ctt6wpwgj

[32] M. G. M. Mehedi Hasan, A. Datta, and M. A. Rahman, "Chained of things: A secure and dependable design of autonomous vehicle services," in *Proc. IEEE/ACM 3rd Int. Conf. Internet-of-Things Des. Implementation*, 2018, pp. 298–299.

[33] W. G. III, "Dacsee whitepaper," White Paper, Oct. 2017. [Online]. Available: https://dacsee.io/dacsee-whitepaper.pdf

[34] T. iRide, "iride : Ridesharing powered by ethereum blockchain," White Paper, Sep. 2018. [Online]. Available: https://iride.io/whitepaper/iRide-whitepaper.pdf

[35] Bitcab whitepaper, White Paper, 2017. [Online]. Available: https://bitcab.io/BitCab WhitePaper.pdf

[36] Ridecoin, White Paper, Feb. 2018. [Online]. Available: https://static1.squarespace.com/static/5aa1f1de0dbda36d6bc88b4e/t/5ab19b17f950b7dcfa8e0450/1521589015691/Ridecoin+Whitepaper.pdf

[37] La-zooz white paper, White Paper, Jun. 2015. [Online]. Available: https://www.weusecoins.com/assets/pdf/library/LaZooz

[38] Arcade city, Accessed: Jan. 23, 2019. [Online]. Available: https://arcade.city/

[39] Chasyr, Accessed: Jan. 23, 2019. [Online]. Available: https://chasyr.com/

[40] Y. Yuan and F.-Y. Wang, "Towards blockchain-based intelligent transportation systems," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst.*, 2016, pp. 2663–2668.

[41] S. Qi, Y. Lu, Y. Zheng, Y. Li, and X. Chen, "Cpds: Enabling compressed and private data sharing for industrial IoT over blockchain," *IEEE Trans. Ind. Informat.*, early access, May 28, 2020, doi: 10.1109/TII.2020.2998166.

[42] L. Zhu, K. Gai, and M. Li, "Blockchain-enabled ride-hailing services," in *Blockchain Technology in Internet of Things*, Berlin, Germany: Springer, 2019.

[43] K. Yaron and E. Safra, "Cryptotransport: Blockchain-powered ride hailing while preserving privacy, pseudonymity and trust," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2018. pp. 540–543. doi: https://doi.org/10.1145/3274895.3274986

[44] M. Baza, N. Lasla, M. Mahmoud, G. Srivastava, and M. Abdallah, "B-Ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain," *IEEE Trans. Netw. Sci. Eng.*, early access, Dec. 23, 2019, doi: 10.1109/TNSE.2019.2959230.

[45] H. Zhang *et al.*, "Smart contract for secure billing in ride-hailing service via blockchain," *Peer-to-Peer Netw.*, vol. Appl. 12, pp. 1346–1357, 2019. [Online]. Available: https://doi.org/10.1007/s12083-018-0694-5

[46] S. Ba, J. Stallaert, and A. B. Whinston, "Research commentary: Introducing a third dimension in information systems design–The case for incentive alignment," *Inf. Syst. Res.*, vol. 12, no. 3, pp. 225–239, 2001.

[47] N. Vafeas and J. F. Waegelein, "The association between audit committees, compensation incentives, and corporate audit fees," *Rev. Quantitative Finance Accounting*, vol. 28, pp. 241–255, 2007. [Online]. Available: https://doi.org/10.1007/s11156–006-0012-9

[48] H. Ma, E. X. Huang, and K. Y. Lam, "Blockchain-based mechanism for fine-grained authorization in data crowdsourcing," *Future Gener. Comput. Syst.*, vol. 106, pp. 121–134, 2020.

[49] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, "Making knowledge tradable in Edge-AI enabled IoT: A consortium blockchain-based efficient and incentive approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6367–6378, Dec. 2019.

[50] C. Chen, J. Wu, H. Lin, W. Chen, and Z. Zheng, "A secure and efficient blockchain-based data trading approach for internet of vehicles," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9110–9121, Sep. 2019.

[51] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, early access, Nov. 08, 2019, doi: 10.1109/TDSC.2019.2952332.

[52] Z. Zhou, B. Wang, M. Dong, and K. Ota, "Secure and efficient vehicle-to-grid energy trading in cyber physical systems: Integration of blockchain and edge computing," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 1, pp. 43–57, Jan. 2020.

[53] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 583–598.

[54] P. Gazi, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," *IACR Cryptology ePrint Archive*, vol. 2018, 2018, Art. no. 1239.

[55] Y. Gilad *et al.*, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 51–68.

[56] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

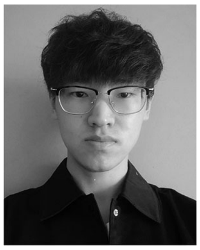
[57] Stella Project Report by European Central Bank and Bank of Japan. Accessed: Apr. 13, 2019. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/ecb.stella_project_report_september_2017.pdf

Youshui Lu received the BS degree from The Australian National University, Australia, in 2013, and the MS degree from the University of Sydney, Australia, in 2015. He is currently working toward the PhD degree at the School of Computer Science and Technology, Xi'an Jiaotong University, China. His research interests include blockchain technology, distributed systems, trusted urban computing, and computational social systems.

Yong Qi (Member, IEEE) received the PhD degree from Xi'an Jiaotong University, China. He is currently a full professor with Xi'an Jiaotong University, China. His research interests include operating systems, distributed systems, and cloud computing.

Saiyu Qi (Member, IEEE) received the BS degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2008, and the PhD degree in computer science and engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2014. He is currently an assistant professor with the School of Cyber Engineering, Xidian University, China. His research interests include applied cryptography, cloud security, distributed systems, and pervasive computing.

Yue Li received the BS and MS degrees in econometrics from the University of Groningen, the Netherlands, in 2010 and 2012 respectively. He is currently working toward the PhD degree in economics in the School of Business and Economics, Vrije Universiteit Amsterdam, the Netherlands and working as a quantitative risk analyst at the ABN AMRO bank, The Netherlands. His research interest includes applied economics, economics models, financial risk models, and macroprudential policies.



Hongyu Song received the bachelor of engineering degree in the Internet of Things engineering from Xi'an Jiaotong University, China, in 2019. He is currently working toward the master's degree in the School of Computer Science and Technology, Xi'an Jiaotong University, China. His research interest includes cyber security, Internet of Things, and blockchain.

Yuhao Liu received the BS degree in software engineering from Northwestern Polytechnic University, China, in 2018. He is currently working toward the master's degree in the Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, China. His research interests include cyber security, Internet of Things, and blockchain.