Assuring Non-fraudulent Transactions in Cash on Delivery by Introducing Double Smart Contracts

Ngoc Tien Thanh Le[1], Quoc Nghiep Nguyen[2], Nguyen Ngoc Phien[3], Nghia Duong-Trung[4], Thai Tam Huynh[5], The Phuc Nguyen[6], Ha Xuan Son[7]

CanTho University of Technology, Can Tho city, Viet Nam[1,2,4,7]
Center for Applied Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam[3]
Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam[3]
FPT University, Can Tho City, Viet Nam[4,7]
Transaction Technologies PTE. LTD., Singapore[5]
University of Trento, Trento, Italy[6]

***Abstract*—The adoption of decentralized cryptocurrency platforms is growing fast, thanks to the implementation of Blockchain technology and smart contracts. It encourages the novel frameworks in a wide range of applications including finance and payment methods such as cash on delivery. However, a large number of smart contracts developed for cash on delivery suffer from fraudulent transactions which enable malicious participants to break the signed contracts without sufficient penalties. A shipper will involve in the system and place a mortgage to ensure reliability. A buyer also pledges an amount of money when making the order. Our process not only ensures the interests of a seller but also prevents a fraud shipper. The penalties will be made in two scenarios: (i) the buyer refuses to receive the commodities without any reliable reasons; and (ii) the shipper attempts to make any modification on the delivered goods during transportation. To help developers create more secure and reliable cash on delivery system, we introduce double smart contracts, a framework rooted in Blockchain technology and Ethereum, to tackle those mentioned problems. We also contribute our solution as an open source software that developers can easily add to their implementation to enhance functionality.**

***Keywords*—Cash on Delivery (COD); Blockchain; smart contract; Ethereum; e-commerce; online payment**

I. Introduction

A basic problem for e-commerce is the exchange of digital goods for payment. The earliest solutions to this problem come from at least on the first day of the world wide web [1], e.g. online stores accept credit card payments. Because goods exchange and payment cannot happen simultaneously, there is an inherent tension and consequently, trust in the transaction is required. The seller must trust that the buyer will pay and the buyer must believe that the seller must deliver the goods. Traditionally, this necessity for trust has been solved by introducing a third party, e.g. a credit card.

Cash on Delivery (COD) allows customers to pay in cash when the products are delivered to their home or a location they choose. This is sometimes called a payment system because customers receive goods before making a payment. COD has become increasingly popular in recent years and been considered one of the main payment methods in many countries [2], [3], [4]. However, most published documents about COD have appeared in reports or magazines and/or on the web, with a few scientific studies to date. Among research articles, most investigated payment methods is in general, rather than focusing on COD in particular. Transfer agents are often used as postal services, but usually, consumer and business shipments will be sent to COD by courier companies, commercial truck forwarders or organizations own delivery services. COD sales usually involve a delivered fee charged by the shipping agents and is usually paid by the buyer. In retail and wholesale transactions, shipments rely on COD-based payment method when the buyer does not have a credit account and the seller does not choose a payment method in advance. COD postal services [5] were first introduced in Switzerland in 1849, India and Australia in 1877, the United States in 1913, Canada in 1922 and the United Kingdom in 1926.

In a contrary direction to previous work, the authors propose an information contract implemented by a seller. A smart contract requires both the shipper and the buyer to place a deposit into an account. In the proposed protocol, the shipper first sends the deposit to get the seller's goods. Then the buyer sends the payment as well as the deposit itself. The seller then sends the key to open the digital commodities. The buyer verifies that the commodities are well received. If the commodities are in good condition, they will send a notification to a smart contract. Deposits made by both parties are only returned after successful transactions have been made, e.g. shippers successful delivered the commodities which are also successfully verified by buyers. To the best of our knowledge, this novel idea is firstly investigated and implemented by the authors.

II. Related Work

One of the major problems of e-commerce globally is buying and selling between parties on the Internet. Krishnamachari *et. al.* [6] proposed a mechanism to implement a transaction with any asset by using digital keys and these processes did not require a reliable third party. In addition, the authors described a deposit transaction method for fraudulent and delivery transactions between the two parties in which the seller can use digital signatures to verify a transaction. Sellers and buyers use a pair of keys to verify goods. A smart contract is utilized to decide and handle seller-buyer relations

by increasing deposits. But the above article has not analyzed the shipping issue. More specifically, if the delivery does not comply with the commitment, the system cannot resolve.

Other researchers [7], [8] has proposed a mechanism based on the Ethereum Blockchain [9] or Son *et. al.* has introduced a mechanism [10] based on Hyperledger Fabric flatform [11] that relates to product transportation between sellers and buyers. In their approach, the carrier plays an important role. The transportation process consists of 2 steps: (i) a key is shipped with the product and handed over to the buyer, and (ii) the buyer will enter the key to confirm the reception in Smart Contract. Ether [12] will only be placed in the seller's account if the key entered by the buyer matches the key in the Smart Contract. Ether will be forwarded to the seller after successful confirmation. This solution is easy to implement because it is quite simple and depends on the key that the seller gives to the carrier. However, this leads to dependence on the belief that the shipper will not take advantage of that key before handing it to the buyer. Therefore, this solution is not recommended.

Hasan and Salah [13] has introduced a delivery process including buyers, sellers and carriers. If a carrier wants to deliver the commodities of a seller, he/she must place a mortgage payment in advance. This amount is usually double the value of the commodities. If the goods are successfully shipped, the money is paid to the parties. If it fails, the system will resolve the dispute by relying on delivery time, from which the system will make a decision without human intervention. Doubling the mortgage price of the products not only increases the transaction cost but also prevents mortgagees from cheating to avoid loss of money.

In this article, we look at several COD related issues and solve them by Blockchain technology. We introduce double smart contracts to address non-fraudulent transactions. A shipper will involve in the system and place a mortgage to ensure reliability. A buyer also pledges an amount of money when making the order. Our process not only ensures the interests of a seller but also prevents a fraud shipper. The penalties will be made in two scenarios: (i) the buyer refuses to receive the commodities without any reliable reasons and (ii) the shipper attempts to make any modification on the delivered goods during transportation.

III. Materials and Technical Background

In this section, the authors summarize the most related technical background that we implement in this work. Readers might also look at some interesting materials in the literature [14], [15], [16].

A. Blockchain and Blockchain-based Smart Contracts

Blockchain is a list of developing logs, called blocks, linked by encryption. Each block contains the previous block's cryptographic hash function, timestamp, and transaction data. Each block has a block header and a body containing data and hash values of the previous block. The hash value is the result of a hash function. The hash function transforms data of any length into a fixed length string or numeric value, such as 256 bits (32 bytes) with SHA256. Blockchain is a technology that allows secure data transmission based on an extremely complex encryption system, similar to accounting books of a company where cash is closely monitored. In this case, the blockchain is an accounting ledger [17] that works in the digital field. A special feature of blockchain is that transactions are done at a high level of trust without disclosing information.

Blockchain-based smart contracts are proposed contracts that could be partially or fully executed without human interaction [18], [19], [20]. One of the main objectives of a smart contract is an automated escrow. An IMF (International Monetary Fund) staff discussion reported that smart contracts based on Blockchain technology might reduce moral hazards and optimize the use of contracts in general, but "no viable smart contract systems have yet emerged". Due to the lack of widespread use, their legal status is unclear [21]. Smart contract based on blockchain is being considered for many different types of transactions, from ubiquitous devices to real-time operational management structures for industrial products and data transfer in some applications including transaction finance. All types of business and management can participate in the network and use the properties of the Blockchain system to ensure transparency of stakeholders.

B. Ethereum

Ethereum (ETH) [9] is an open-source, and a blockchain-based distributed computing platform. Ethereum uses functions of smart contracts. Smart contract of Ethereum written by Solidity [22] programming language. A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. A Smart contract allows the performance of credible transactions without third parties involved. These transactions are trackable and irreversible. A transaction in a smart contract will use Ether [23] unit to pay for the transaction. Like Bitcoin money has Satoshi and Kilobyte, USD money has dollars and cents, Ether is the currency of Ethereum's internal network. Nevertheless, Wei is the smallest unit changed from Ether unit used in a smart contract normally. In an Ethereum network, there are two addresses that we should be noted: an account address and a contract address. Each account address which is an external account has a corresponding personal key (private key). We can treat the private key as a password that we are the only ones who know. We need the address and private key pair to interact with the Blockchain. A contract address is also called a contract account which is controlled by the code stored together with the account. The contract address is determined at the time the contract is created. It is derived from the creator address and the number of transactions sent from that address, the so-called "nonce". Furthermore, every account has a balance in Ether, e.g. in Wei to be exact, 1 ether is 10**18 wei, which can be modified by sending transactions that include Ether. Furthermore, one unit may be confused with Ether unit which is gas unit. When creating, each transaction is charged by a certain amount of gas, its purpose is to limit the amount of work needed to execute the transaction and pay for this execution at the same time.

C. Smart Contracts

A cryptocurrency is a decentralized platform that a distributed ledger is used to interact with virtual money. A contract is an instance of a computer program that executes

on the Blockchain. Users transfer money by publishing transactions and interacting with contracts in the cryptocurrency network where information is propagated, data is stored among miners or network's nodes. An underlying cryptocurrency system supports the utilization of smart contracts. A smart contract contains program code, a stored file and an account balance. Any user can submit a transaction to an append-able-only log. When the contracted is created, its program code cannot be changed. An append-able-only log, called a blockchain, which imposes a partial or total arrangement on submitted transactions is the main interface provided by the cryptocurrency. Fig. 1 presents the idea of a decentralized cryptocurrency system and its components.

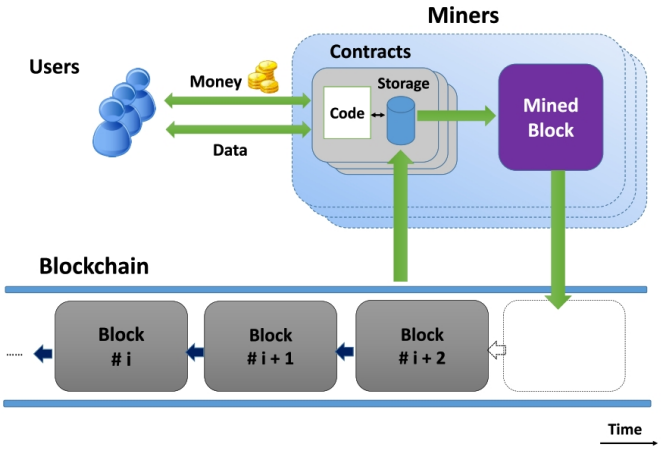

Fig. 1. An illustration of smart contracts and Blockchain in a decentralized cryptocurrency system [24].

IV. Proposed Methodology

In this section, the authors introduce a general overview of the architecture with a highlight of the idea of double smart contracts. Then, we discuss the proposed architecture is more details. Finally, we present several important algorithms that serve as the backbone of our proposed architecture.

A. General Architecture

Our proposed COD system consists of several components, e.g. see Fig. 2. A seller gives package information into the system (step 1). In step 2, a buyer sends the request to buy this package. Then, the seller and buyer will sign a smart contract, called smart contract 1, in step 3. In smart contract 1, the buyer must mortgage a price to ensure that if the buyer is attempted to fraudulent or a shady businessman, the money will be sent to the seller. In step 4, the shipper sends a request to deliver the package. After the shipper sends the request, the seller and shipper must sign the smart contract 2 (step 5). In smart contract 2, the shipper deposits an amount of money that is equal to the price of the package to avoid shipper's lost or modification. In step 6, the shipper will send the package to the buyer and confirm a successful transaction. Next, the money deposited in the smart contract 2 will be sent automatically to the seller in step 7. Similar to step 7, smart contract refunds money to the buyer in step 8.

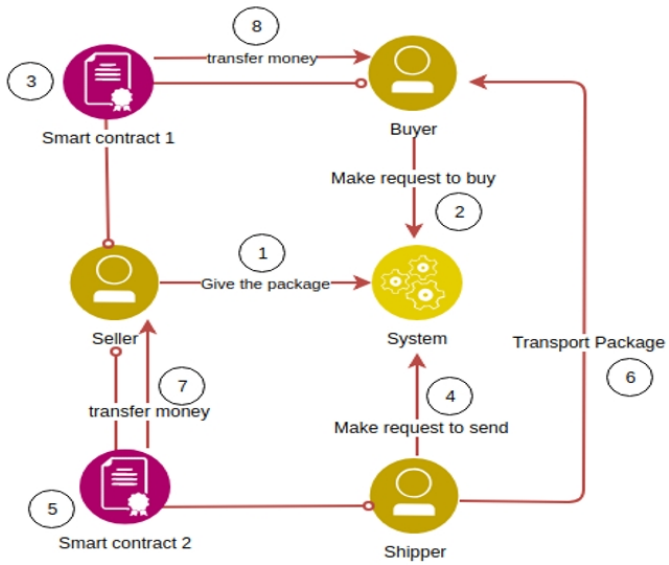

Fig. 2. General design of our proposed architecture.

B. Detailed Design

In this subsection, the authors discuss the proposed COD architecture in more fine-grained details. An illustration of the detailed design is presented in Fig. 3. In step 1, a seller gives a package into the Blockchain system with the information about the package's name and its price. In step 2 and 3, a buyer sent a request to buy the package and deposit the money into the smart contract 1 to ensure personal interests. Next, in step 4 and 5, the seller sends the request to receive the package together with its information and the buyer's address. The seller and shipper will sign the smart contract 2. In this smart contract 2, the shipper must deposit an amount of money equal to the price of the package to ensure that the interest for the seller. Next, the shipper will send this package, step 6, to the buyer and then the buyer must pay money to the shipper in step 7. If the transaction goes well and the shipper receives cash from the buyer, the deposit amount of money in the smart contract 1 will be sent to the buyer in step 8, and the amount of deposit money in the smart contract 2 will be transferred to the seller in step 9. So the delivery has been successful. However, if transportation fails due to the shipper problems such as lost or item damaged during delivery in step 10. The money balance in the smart contract 2 will be transferred to the seller in step 11 and the money balance in the smart contract 1 will be transferred to the buyer in step 12. Furthermore, in step 13, if the buyer does not pay for the shipper, the deposit money in the smart contract 1 will be sent to the seller in step 14. Due to the errors generated by the buyer, deposit money in the smart contract 2 will be sent to the shipper in step 15. The steps that will be discussed in the paper hereafter are referred to these steps in the detailed design.

C. Algorithms

The main purpose of the algorithm (1) is to create a package with the request that the seller is the package's owner. In step 2, the package's information will be collected including *id*, *name* and *price*. The information is stored with an auto-id increase in step 3. An exemplification of Algorithm (1) is

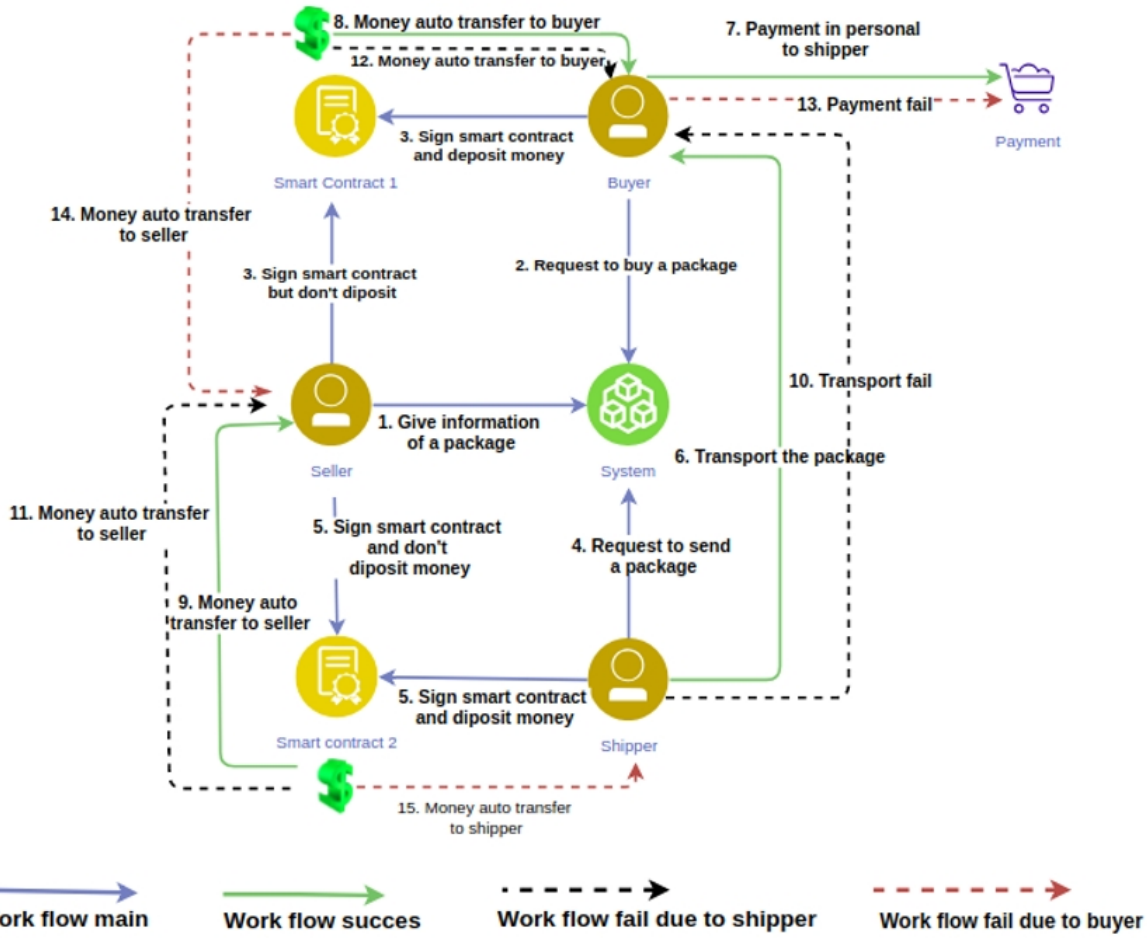

Fig. 3. Detailed Design of our proposed architecture.

Algorithm 1 Create Package

1: **if** the seller is an owner and he/she creates the requirement **then**
2: Set information: *id*, *package name*, and *price*.
3: Save the information into an array with id package increasing one unit.
4: **else**
5: Show an error message to customer.
6: **end if**

presented in Table I.

Algorithm 2 Money Deposit

1: Get *package id*.
2: **if** caller address != owner address of the package **then**
3: set price to deposit.
4: **if** price == price of the package **then**
5: set price success.
6: **else**
7: show an error message.
8: **end if**
9: save *balance* into a smart contract.
10: **return** value of *balance* equal to price deposited
11: **end if**

TABLE I. AN EXEMPLIFICATION OF ALGORITHM 1.

Transaction hash	0x55593a8aa7aaee08a6ae3f354bd8eb7ebf8ace5ea 326c48eaf4fe6367bdee36b
From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Sell.AddPackage(string,uint256) 0xbbf289d846208c16edc8474705c748aff07732db
Gas	3000000
Transaction cost	104565 gas
Execution cost	82397 gas
Hash	0x55593a8aa7aaee08a6ae3f354bd8eb7ebf8ace5ea 326c48eaf4fe6367bdee36b
Input	0xb3a...00000
Decoded input	{ "string _name": "car", "uint256 _price": "10" }
Call to Sell.idPackage	{"0": "uint256: 1"}

This main purpose of the algorithm (2) is to get *id* of the package in step 1. Step 3 sets the deposit money with a condition that the price must be equal the price of the package in the algorithm (1). In Table I, the price of the package is 10 Ether. In Table II the amount of deposit money with 10000000000000000000 Wei equal 10 Ether is transferred into a smart contract.

In algorithm (3), the system gets the value of balance in made in algorithm (2). In step 2, the caller of this function confirms the balance transfer to the specified address in step 3. In Table II, readers can see

TABLE II. AN EXEMPLIFICATION OF ALGORITHM 2.

Transaction hash	0x21c1c983b876728c9607c47ee9c633907714ae4 99e9e077e06a2590f255fbe1e
From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Sell.ApplyBuy(uint256) 0xbbf289d846208c16edc8474705c748aff07732db
Gas	3000000
Transaction cost	64531 gas
Execution cost	43076 gas
Hash	0x21c1c983b876728c9607c47ee9c633907714ae4 99e9e077e06a2590f255fbe1e
Input	0x114...00001
Decoded input	{ "uint256 id": "1" }
Decoded output	{ "0": "uint256: 10000000000000000000" }
Logs	[]
Value	10000000000000000000 wei

Algorithm 3 Money Transfer

1: get *balance* of the smart contract.
2: **if** caller address confirmed **then**
3: transfer *balance* to specified address.
4: **end if**

the money deposit equal to 10 Ether unit from address *0x14723a09acff6d2a60dcdf7aa4aff308fddc160c*. So in Table III, the system will transfer money to various address and the balance will be return 0. An instance running of Algorithm (3) is presented in Table III.

V. EXPERIMENTAL SETUP

The smart contracts are created and tested using the Remix IDE [25], [26] which provides necessary tools for developing and debugging. As explained in the previous sections, gas is a measure of the amount of expenditure used to calculate the cost need to perform certain activities [24]. Each and every line of code will definitely require a certain amount of gas to calculate. We illustrate three different cases in this article and compare the total gas amount in three cases afterwards. In case 1, we illustrate a normal process and call it Transport Success. Next, in case 2, we discuss transaction errors due to shipper problems. Finally, we present how the system handles transaction errors because the buyer does not accept commodities.

TABLE III. AN EXEMPLIFICATION OF ALGORITHM 3.

Transaction hash	0x075ea20859f1a0bf554341efc62f48c60010ae14 cf7c536879e94168a86f0d4c
From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Sell.ConfirmToTransferBuyer() 0xbbf289d846208c16edc8474705c748aff07732db
Gas	3000000
Transaction cost	30279 gas
Execution cost	9007 gas
Hash	0x075ea20859f1a0bf554341efc62f48c60010ae14 cf7c536879e94168a86f0d4c
Input	0x11f...36795
Decoded input	{ }
Decoded output	{ "0": "uint256: 0" }
Logs	[]
Value	0 wei

A. Case 1: Transport Success

1) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c creates a package on the smart contract 1: see Table IV.

TABLE IV. CASE 1: STEP 1

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Sell.AddPackage(string,uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	104693 gas
Execution cost	82397 gas

2) Buyer 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 agrees to deposit the amount of money equal to 10 Ether: see Table V.

TABLE V. CASE 1: STEP 2

From	0xdd870fa1b7c4700f2bd7f44238821c26f7392148
To	Sell.ApplyBuy(uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	64531 gas
Execution cost	43067 gas

3) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c gives the information of the package as well as the buyer's address to the shipper in the smart contract 2: see Table VI.

TABLE VI. CASE 1: STEP 3

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Cod.AddPackage(string,uint256,address,uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	187265 gas
Execution cost	163369 gas

4) Shipper 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c agrees to deposit price and deliver the package: see Table VII.

TABLE VII. CASE 1: STEP 4

From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Cod.ApplyDeliver(uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	44397 gas
Execution cost	22933 gas

5) Transport success: The buyer pays in cash directly to the shipper. Then teh deposit money in smart contract 1 will be transferred to the buyer *0xdd870fa1b7c4700f2bd7f44238821c26f7392148*, see Table VIII.

TABLE VIII. CASE 1: STEP 5

From	0xdd870fa1b7c4700f2bd7f44238821c26f7392148
To	Sell.ConfirmToTransferBuyer() 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	30279 gas
Execution cost	9007 gas

6) The deposit money in smart contract 2 is transferred to the seller: see Table IX.

TABLE IX. CASE 1: STEP 6

From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Cod.ConfirmSuccess() 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	29790 gas
Execution cost	8518 gas

B. Case 2: There are transaction errors due to shipper problems

1) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c creates a package on smart contract 1: see Table X.

TABLE X. CASE 2: STEP 1

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Sell.AddPackage(string,uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	104693 gas
Execution cost	82397 gas

2) Buyer 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 agrees to deposit the amount of money equal to 10 Ether: see Table XI.

TABLE XI. CASE 2: STEP 2

From	0xdd870fa1b7c4700f2bd7f44238821c26f7392148
To	Sell.ApplyBuy(uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	64531 gas
Execution cost	43067 gas

3) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c gives the information of the package as well as the buyer's address to the shipper in the smart contract 2: see Table XII.

TABLE XII. CASE 2: STEP 3

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Cod.AddPackage(string,uint256,address,uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	187265 gas
Execution cost	163369 gas

4) Shipper 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c agrees to deposit price and deliver the package: see Table XIII.

TABLE XIII. CASE 2: STEP 4

From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Cod.ApplyDeliver(uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	44397 gas
Execution cost	22933 gas

5) The shipper 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c has a problem: the buyer *0xdd870fa1b7c4700f2bd7f44238821c26f7392148* does not receive the package. The seller *0xca35b7d915458ef540ade6068dfe2f44e8fa733c* will receive money from the balance of smart contract 2, see Table XIV.

TABLE XIV. CASE 2: STEP 5

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Cod.ErrorSuccess() 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	30066 gas
Execution cost	8794 gas

6) The money of balance in smart contract 1 will be transferred to the buyer 0xdd870fa1b7c4700f2bd7f44238821c26f7392148: see Table XV.

TABLE XV. CASE 2: STEP 6

From	0xdd870fa1b7c4700f2bd7f44238821c26f7392148
To	Sell.ConfirmToTransferBuyer() 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	30279 gas
Execution cost	9007 gas

C. Case 3: There are problems with transactions because the buyer does not accept commodities.

1) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c creates a package on smart contract 1: see Table XVI.

TABLE XVI. CASE 3: STEP 1

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Sell.AddPackage(string,uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	104693 gas
Execution cost	82397 gas

2) Buyer 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 agrees to deposit the amount of money equal to 10 Ether: see Table XVII.

TABLE XVII. CASE 1: STEP 2

From	0xdd870fa1b7c4700f2bd7f44238821c26f7392148
To	Sell.ApplyBuy(uint256) 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
Gas	3000000
Transaction cost	64531 gas
Execution cost	43067 gas

3) Seller 0xca35b7d915458ef540ade6068dfe2f44e8fa733c gives the information of the package as well as the buyer's address to the shipper in the smart contract 2: see Table XVIII.

TABLE XVIII. CASE 3: STEP 3

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Cod.AddPackage(string,uint256,address,uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	187265 gas
Execution cost	163369 gas

4) Shipper 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c agrees to deposit price and deliver the package: see Table XIX.

TABLE XIX. CASE 3: STEP 4

From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Cod.ApplyDeliver(uint256) 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	44397 gas
Execution cost	22933 gas

5) The buyer 0xdd870fa1b7c4700f2bd7f44238821c26f7392148 avoids responsibility and does not receive goods without any acceptable reasons: The deposit money of the buyer will be transferred to the seller *0xca35b7d915458ef540ade6068dfe2f44e8fa733c* after the seller confirms that the problem is from the buyer, see Table XX.

TABLE XX. CASE 3: STEP 5

From	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
To	Sell.ErrorSuccess() 0xbbf289d846208c16edc8474705c748aff07732db
Gas	3000000
Transaction cost	30044 gas
Execution cost	8772 gas

6) The shipper takes back its money in smart contract 2: see Table XXI.

TABLE XXI. CASE 3: STEP 6

From	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
To	Cod.ConfirmToTransferShipper() 0x0dcd2f752394c41875e259e00bb44fd505297caf
Gas	3000000
Transaction cost	30135 gas
Execution cost	8863 gas

VI. FINAL REMARKS

In Fig. 4, we observe that the total transaction gas and execution gas are similar in three investigation scenarios. In Ethereum, gas is the concept to discourage over-consumption of resources. The user who creates a transaction must purchase gas by spending currency. Every program instruction consumes some amount of gas during a transaction is executed. Consequently, if a transaction fails in case of the failure caused by other partners, the total gas spent should not be so high. The double smart contracts have successfully penalized undesired failures.

VII. CONCLUSION

In this paper, we propose a new framework that uses smart contracts, blockchain and Ethereum to assure non-fraudulent transactions in cash on delivery and enhance the reliability of distributed cryptocurrency platforms. Although there is a lot of thoughtful discussion on the use of smart contracts in distributed cryptocurrency, there have not been many frameworks that would address the non-fraudulent transactions caused by buyers and shippers. The penalties will be made in two circumstances: (i) the buyer refuses to receive the commodities without any reliable reasons and (ii) the shipper attempts to make any modification on the delivered goods during transportation. Our approach is a practical implementation which has been developed and evaluated empirically.

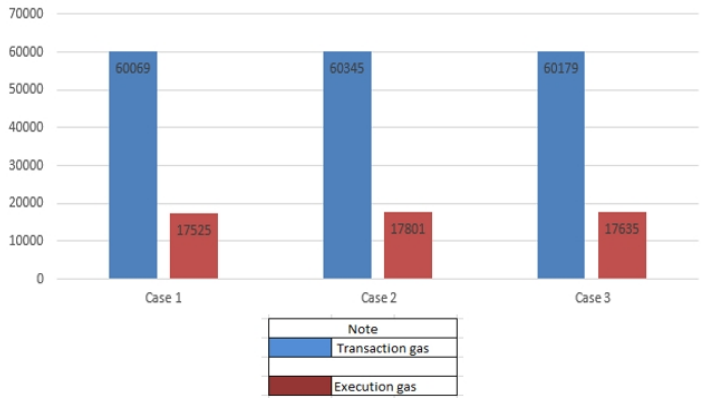

Fig. 4. Gas consumption in three investigation scenarios.

To facilitate future research endeavors on COD and smart contract programming, we have released the open source codes of our implementation. The materials are available at https://github.com/nghiepnguyen520/Cash-on-delivery.

REFERENCES

[1] P. Timmers, *Electronic commerce*. John Wiley & Sons, Inc., 1999.

[2] M. Halaweh, "Cash on delivery (cod) as an alternative payment method for e-commerce transactions: Analysis and implications," *International Journal of Sociotechnology and Knowledge Development (IJSKD)*, vol. 10, no. 4, pp. 1–12, 2018.

[3] ——, "Intention to adopt the cash on delivery (cod) payment model for e-commerce transactions: An empirical study," in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2017, pp. 628–637.

[4] U. Tandon and R. Kiran, "Study on drivers of online shopping and significance of cash-on-delivery mode of payment on behavioural intention," *International Journal of Electronic Business*, vol. 14, no. 3, pp. 212–237, 2018.

[5] J. D. Alie and P. E. Vliek, "International cash-on-delivery system and method," Jul. 24 2007, uS Patent 7,249,069.

[6] A. Asgaonkar and B. Krishnamachari, "Solving the buyer and seller's dilemma: A dual-deposit escrow smart contract for provably cheat-proof delivery and payment for a digital good without a trusted mediator," *arXiv preprint arXiv:1806.08379*, 2018.

[7] "Two party contracts," Feb 2015. [Online]. Available: https://dappsforbeginners.wordpress.com/tutorials/two-party-contracts/

[8] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps*. O'Reilly Media, 2018.

[9] "Ethereum project." [Online]. Available: https://ethereum.org/

[10] H. X. Son, M. H. Nguyen, N. N. Phien, H. T. Le, Q. N. Nguyen, V. D. Dinh, P. T. Tru, and P. Nguyen, "Towards a mechanism for protecting seller's interest of cash on delivery by using smart contract in hyperledger," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, 2019. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2019.0100405

[11] "Hyperledger fabric." [Online]. Available: https://hyperledger-fabric.readthedocs.io/

[12] G. Hileman and M. Rauchs, "Global cryptocurrency benchmarking study," *Cambridge Centre for Alternative Finance*, vol. 33, 2017.

[13] H. R. Hasan and K. Salah, "Blockchain-based solution for proof of delivery of physical assets," in *International Conference on Blockchain*. Springer, 2018, pp. 139–152.

[14] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[15] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[16] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 528–547.

[17] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[18] T. Hamid, "Cash on delivery the biggest obstacle to e-commerce in uae and region," May 2014. [Online]. Available: https://www.thenational.ae/business/technology/cash-on-delivery-the-biggest-obstacle-to-e-commerce-in-uae-and-region-1.604383

[19] F. Idelberger, G. Governatori, R. Riveret, and G. Sartor, "Evaluation of logic-based smart contracts for blockchain systems," in *International Symposium on Rules and Rule Markup Languages for the Semantic Web*. Springer, 2016, pp. 167–183.

[20] M. Alharby and A. van Moorsel, "Blockchain-based smart contracts: A systematic mapping study," *arXiv preprint arXiv:1710.06372*, 2017.

[21] D. He, K. F. Habermeier, R. B. Leckow, V. Haksar, Y. Almeida, M. Kashima, N. Kyriakos-Saad, H. Oura, T. S. Sedik, N. Stetsenko *et al.*, "Virtual currencies and beyond: initial considerations," 2016.

[22] "Solidity." [Online]. Available: https://solidity.readthedocs.io/

[23] E. U. C. Team, "Ethereum unit converter." [Online]. Available: https://etherconverter.online/

[24] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 79–94.

[25] "Solidity ide." [Online]. Available: https://remix.ethereum.org/

[26] Ethereum, "ethereum/remix-ide," Apr 2019. [Online]. Available: https://github.com/ethereum/remix-ide