World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:16, No:11, 2022

A Blockchain-Based Privacy-Preserving Physical Delivery System

Shahin Zanbaghi, Saeed Samet

Abstract—The internet has transformed the way we shop. Previously, most of our purchases came in the form of shopping trips to a nearby store. Now, it is as easy as clicking a mouse. We have to be constantly vigilant about our personal information. In this work, our proposed approach is to encrypt the information printed on the physical packages, which include personal information in plain text using a symmetric encryption algorithm; then, we store that encrypted information into a Blockchain network rather than storing them in companies or corporations centralized databases. We present, implement and assess a blockchain-based system using Ethereum smart contracts. We present detailed algorithms that explain the details of our smart contract. We present the security, cost and performance analysis of the proposed method. Our work indicates that the proposed solution is economically attainable and provides data integrity, security, transparency and data traceability.

Keywords—Blockchain, Ethereum, smart contract, commit-reveal scheme.

I. Introduction and Background

THIS work aims to improve existing physical assets delivery systems by leveraging Blockchain technology and focusing on privacy-preserving of personal information. We propose a method for online buyers to have a trusted, decentralized, privacy-preserving physical assets delivery solution. By using this framework, online shoppers would be able to hide and protect their personal information in both real and digital world. Physical delivery of assets is a time-consuming and costly process. The current physical delivery system involves various intermediaries to help facilitate the process. Problems faced in this process include: high cost, poor certainty, low transparency, fraud and high dependence on a single party.

In this section, we review the terminology and main concepts we use in this paper. The concepts include blockchain, Ethereum smart contracts, and a commit-reveal scheme.

A. Blockchain

Blockchain is a distributed ledger technology that enables transactions to be recorded securely and permanently on an immutable ledger. Blockchain records transactions within blocks of information, with each block connected to the next one. This means that once data have been entered into the blockchain, it cannot be edited or deleted. Although this may seem like a disadvantage at first glance, there are many ways this helps solve some of today's biggest problems that exist in many industries. At their basic level, they enable a community of users to record transactions in a shared ledger within that community, such that under normal operation of the blockchain network no transaction can be changed once published.

Shahin Zanbaghi, and Saeed Samet, Associate Professor, are with School of Computer Science, Faculty of Science, University of Windsor, Canada (e-mail: zanbagh@uwindsor.ca, saeed.samet@uwindsor.ca).

B. Ethereum

Ethereum is a public, open-source, blockchain-based distributed computing platform that features smart contract functionality. It delivers a decentralized virtual machine. The Ethereum Virtual Machine, also called EVM, can execute scripts using an international network of public nodes. Ethereum was proposed in late 2013 by Vitalik Buterin [5] and has been live since July 2015. The EVM is an international network that can be used by anyone, not just those who are using it. It represents a blockchain with a built-in Turing complete programming language. [13] It delivers an abstract layer allowing anyone to make their own rules for ownership, formats of transactions, and state transition functions. This is accomplished by applying smart contracts, a collection of cryptographic rules executed only if specific conditions are met [5].

C. Smart Contracts

Ethereum allows us to create a contract with someone and set rules for behaving to receive money from us. This contract is called a smart contract, and it is finalized once both parties have agreed on all the details of the contract. Once this has happened, then no one can change it because everything has been written into the blockchain and cannot be changed again or hacked. In other words, smart contracts as shown in Fig. 1 are self-executing contracts where the terms of the agreement between multiple parties are directly written into lines of code. The code and the agreements contained therein exist across a blockchain network. Smart contracts allow trusted transactions and agreements to be carried out among disparate, anonymous parties without needing a central authority, legal system, or an external enforcement mechanism.

D. Commit-Reveal Scheme

A commit-reveal scheme, also known as commitment scheme, is a cryptographic algorithm utilized to let someone commit to a value while holding it concealed from others with the capability to reveal it later. The values in a commitment scheme are binding, indicating that no one can modify them once committed. The scheme has two stages: 1. The commit

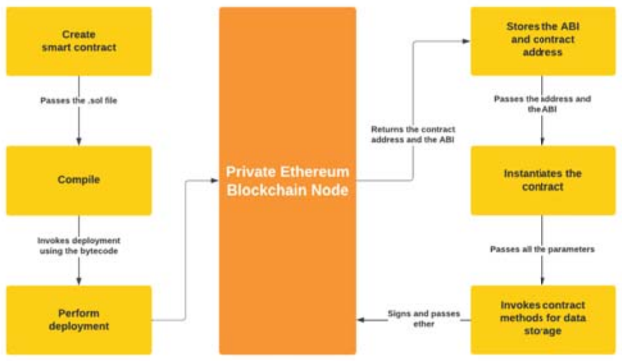
Create smart contract
Passes the .sol file
Compile
Invokes deployment using the bytecode
Perform deployment
Private Ethereum Blockchain Node
Returns the contract address and the ABI
Stores the ABI and contract address
Passes the address and the ABI
Instantiates the contract
Passes all the parameters
Invokes contract methods for data storage
Signs and passes ether

Fig. 1 Smart contract development

stage: where a value is selected and specified. 2. The reveal stage: where the selected value is revealed and confirmed. To get a better understanding, consider this simplified instance. Suppose Alice, the sender, places a note in a sealed package and hands it to Bob, the receiver. Bob cannot access the note because it is sealed in the package, and Alice cannot alter the message because it is in Bob's control, but when Alice likes to reveal the note, she can open the package and show it to Bob [2].

The rest of this paper is organized as follows. Section II provides the related work. Section III presents the methodology of the proposed method. Section IV provides the experiments and results. Section V concludes the paper and provides future work.

II. Related Works

Blockchain technology has been redefining the way we deal with physical assets. It has enabled a more efficient and accurate way of transferring ownership rights of physical assets. Blockchain-based systems have been proposed to help alleviate the many problems faced in physically delivering assets. This section will discuss some works related to blockchain technology and its use in delivering physical assets.

A. *Untraceable Payments Using Cryptography*

The idea of untraceable payments using cryptography was first introduced by David Chaum [6], who used digital blind signature schemes to achieve the desired level of anonymity.

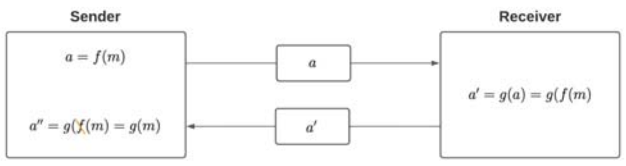

Fig. 2 An example of blind signature in works

Bitcoins first came about in 2009 when it was proposed by Satoshi Nakamoto (a pseudonym) and presented the blockchain technology which offered a practical model for anonymous monetary transactions using public key and hashing primitives [9]. At the beginning, Bitcoin technologies gained popularity among those who were tech-savvy. Later on, academic researchers became interested in them and started researching various features of the blockchain. Specifically, the issue of anonymity on blockchain emerged as an interesting area for research where published works either try to undermine it or offer solutions that strengthen it. The work on the analysis of the Bitcoin blockchain anonymity includes the results presented by Ron and Shamir [11] where they were able to group transactions that shared specific patterns and assign them to various entities with the help of the Bitcoin blockchain graph. They showed how Ross Ulbricht, operator of drug-selling site Silk Road, could not be identified by FBI agents despite their efforts [10]. Another paper that confirmed the difficulty of de-anonymizing multiple mix transactions on the blockchain is presented in [8].

B. *Secure Hash and Key*

"Two party contracts" [1] proposes using a secure hash and a key that the seller gives to the transporter along with an item. Once on the arrival of the transporter to the destination, the buyer would enter the key and then compare the hash of the key against what is already recorded in this contract. This method is straightforward and simple to execute. It uses the contract as an escrow, meaning that you would only pay for goods once they have been verified with a hash. However, it involves trusting the transporter not to manipulate the key which could lead disastrous effects on all parties involved in this transaction. Furthermore, its success depends on all of those involved acting honest and trustworthy- something that cannot be guaranteed at any point in time.

C. *Ethereum Market Places*

LocalEthereum [4] began in 2017 as the first peer-to-peer marketplace for Ethereum. The intent is to use smart contracts and in-browser cryptography to purchase or trade Ether for offline currencies. LocalEthereum is set up so that users consistently manage their funds; however, the currency in which it is denominated. Depositing Ether utilizing LocalEthereum will transfer funds to a client-side encrypted wallet. This wallet lives in the browser independently and is not maintained by the platform operators. All communication between customers and vendors utilizes end-to-end encryption for extra privacy and security. Both parties can also consent to use an escrow system that uses smart contract features. The group will be able to read messages just when a disagreement occurs. That approach involves both parties willingly offering the decrypted versions of their communications to a mediator. Furthermore, BitBay [3] is a decentralized marketplace that was founded by Sylwester Suszek in 2014 which is a doubledeposit escrow mechanism provided by a decentralized online marketplace. With BitBay, the need for a trusted third party is eliminated because the contract will act as an escrow. It takes all investments until the transaction is completed. With BitBay, it's possible to have transactions in which physical items are being delivered without any guarantee that they will not be tampered with or delayed by a transporter during transit time.

D. Blockchain-Based Proof of Delivery

One of the recent work done by Hasan and Salah [7] presents a blockchain based Proof of Delivery (POD) solution of shipped physical items that uses smart contracts of Ethereum blockchain network, in which tracking, and tracing activities, logs, and events can be done in a decentralized manner. The proposed solution uses a smart contract attestation authority to confirm that the code follows the terms and conditions signed by the participating entities. Furthermore, it permits the transaction cancellation by the vendor, customer and transporter established on the contract state. Similarly, in particular justifiable cases, the customer can even ask for a refund.

III. Methodology

This section proposes a solution that utilizes the Ethereum blockchain to create a decentralized system with trust, immutable logs, and events. This system protects the personal information of online buyers in both the physical and digital worlds by using smart contracts that automate this process.

A. Motivation

Privacy is not given; it can only be taken. No one has the right to violate an individual's privacy without consent. Individuals should control how much data they want to share with others and for whom they are willing to reveal those data. Privacy becomes even more critical in a society where so many things are recorded digitally without permission from individuals, even if the information was previously public knowledge or accessible by anybody online for years prior. Privacy and security solutions for the physical world are evolving. We need to keep up with what's happening in our digital lives so that we can protect ourselves in the physical world, too. It is time for us to evolve these privacy and security solutions into even more effective than before.

B. Problem Statement

Traditional delivery systems do not have acceptable protection for the personal information printed on the packages and the privacy and transparency of the same information stored in the centralized databases of tech corporations. Those data include personal information like full name, phone number, email address, home address, etc. Blockchain technology is very transparent. All transactions are visible to all participants from the beginning of its creation until now, making it a very open system. There's less chance for any discrepancies in the network due to nothing being hidden away or cloaked. Due to the high transparency of transactions in blockchain technology, any kind of fraud can be easily identified. On the other hand, a smart contract is a contract where specific situations and conditions are specified, which helps execute a predefined task automatically. Blockchain technology is helpful in the invention of automating predefined action execution. The intent of smart contracts is to decline the cost of the transaction, enhance the execution speed, and provide security of a higher level when compared with traditional law contracts. Furthermore, the commit-reveal scheme can help us securely hold some information on the Blockchain network and reveal it at the proper time for the proper party or agent.

Because of all the reasons mentioned, we thought the blockchain, smart contracts and commit-reveal scheme were the best options to solve the mentioned problem.

C. Proposed Method

The proposed solution focuses on privacy-preserving personal information on the delivery of physical assets between a seller and a buyer. Agents are also part of the contract to ensure the asset delivery is carried out and each of the mentioned participants possesses an Ethereum address. In our proposed system, we use Blockchain technology, smart contracts to increase the transparency, accessibility, and integrity of the data and the commit-reveal scheme to preserve the confidentiality of the data. We assume all of the mentioned participants are joined in a blockchain network channel and maintain a shared and distributed ledger. All of the participants are connected to the smart contract proposed in the method to either commit or reveal a piece of information from the Blockchain network. Lastly, we agreed that agents would act honestly and not share information.

The various roles of the Ethereum entities in our smart contract are outlined below:

Seller: The seller creates the contract, provides keys and QR code to agents, and the seller is the party who owns and sells the asset.

Buyer: The buyer is the entity that commits to buying an item, while they are responsible for entering their shipping address into a smart contract.

Delivery Agency: Delivery Agencies or agents typically receive an item from the seller and deliver it to the buyer at the shipping address agreed on the smart contract. This proposed smart contract follows a certain algorithm that flows in sequence so the participating entities know what to do. If actions take place off the blockchain, they are tracked by functions in the contract that trigger logged events. The contract holds the shipping address, so the contract acts as an escrow until the package reaches the last agent responsible for the last step of the delivery. The smart contract includes the following:

Modifiers: Modifiers guarantee that the appropriate legitimate commodities execute transactions and functions. Modifiers modify the function that uses them to permit it to execute based on the impact of another code first executed inside the modifier.

Require: Require can be used to check for conditions and throw an exception if the condition is not met.

Events: If a function is executed, an event could be used to create notifications and log the activity.

Variables: Variables help you save important values internally that preserve the state of the contract as it changes along with the functions. Variables used in the contract hold the shipping address of the buyer as well as the Ethereum addresses of agents.

Fig. 3 shows the system architecture of the proposed blockchain solution that shows the transactions between the seller, buyer and agents. The buyer first submits its shipping address using the front-end created for buyers. Then the framework encrypts the shipping address using AES-256 symmetric encryption before committing it to the blockchain network. In the next step, the seller adds the list of agents who can send a reveal request by accessing the front-end created for sellers; after adding the first agent, the framework sends back a QR code. The seller is responsible for transferring that QR code to agents by printing the QR code on the package. Agents scan the QR code to access the agent's front-end to send a reveal request; if the request came from an agent the seller added, the framework reveals the portion of the address is necessary for the agent to process to the next delivery step,which another agent does. The reveal process would be repeated until the last piece of data is revealed, then the smart contract would reject any new reveal request. Moreover, the seller could control the agents access to the data by implementing any access control lists (ACLs). Fig. 4 shows the sequence diagram that demonstrates the flow of the reveal process.

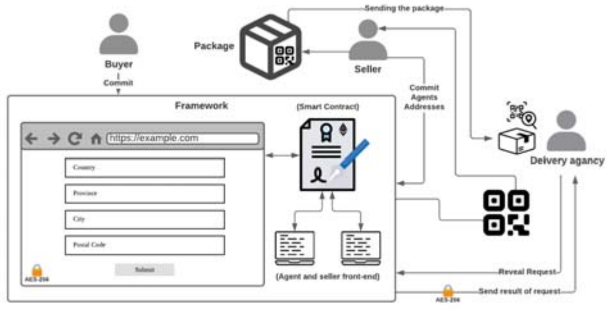

Fig. 3 The system architecture of the ethereum-based solution showing the main participating entities that participate in a successful transaction

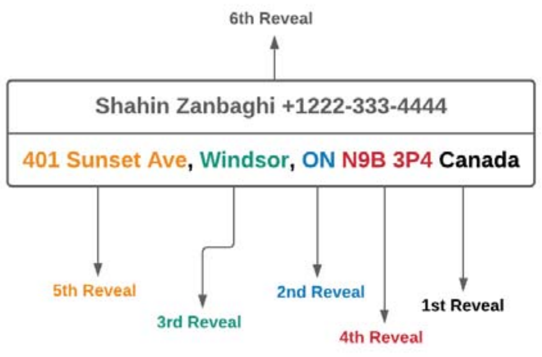
6th Reveal
Shahin Zanbaghi +1222-333-4444
401 Sunset Ave, Windsor, ON N9B 3P4 Canada
5th Reveal
3rd Reveal
2nd Reveal
4th Reveal
1st Reveal

Fig. 4 Example of reveal process on an address

Only the last agent who will deliver the physical assets to the buyer would be able to guess the full address because of their physical presence at the final address location. In our scheme, agents are expected not to reveal any data between each other.

D. Implementation

Our framework has four main sections shown in Fig. 5 which are listed as:

1) Smart contract.
2) Front-end for buyers.
3) Front-end for sellers.
4) Front-end for delivery agency.

The smart contract code is written in Solidity using the Truffle Suite [12]. The code focuses on three main entities, the seller, agent and buyer to acquire the delivery of a physical asset. Pseudocode of the smart contract is shown in Algorithm 1. Blockchain network is an open way to record transactions. Because of that, before committing information provided by the buyer (full name, phone number, country, city, Address and Postal code) on the front-end created for them, We encrypt them on the same front-end using AES-256 symmetric encryption. After finishing this process, we commit them to the Blockchain network as encrypted data. If someone checks the transaction, they will only see encrypted data rather than plain text.

Our framework is not limited to AES encryption; we can easily implement any other encryption algorithms based on need. We used this encryption algorithm because it is secure enough and has acceptable resource usage and performance.

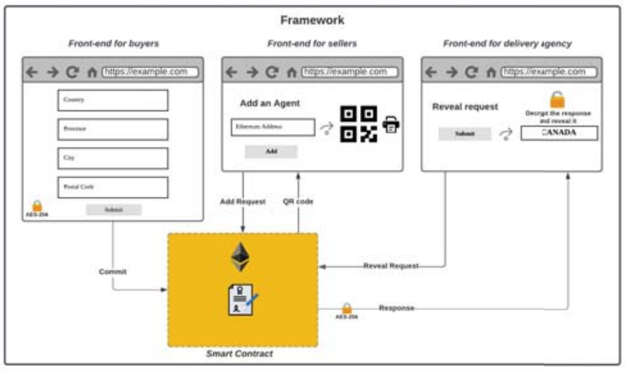

Fig. 5 Framework overview

After the buyer commits to the shipping information, the seller should add the list of agents who can reveal information. As we mentioned, sellers can set different ACL types based on their needs. In our smart contract seller should add six unique Ethereum addresses in total by accessing to the seller front-end. Each Ethereum address is for an agent. Each agent can reveal only one part of shipping information, which means if an agent has already released a piece of information, any request from them will be rejected by smart contact.

Unlike shipping addresses, the list of Ethereum addresses is not encrypted. Because of that, anyone, including the buyer, can check the Ethereum transactions to find agents list who will deliver their package.

After the seller adds the first agent, the framework will generate the QR code that the seller should print on the package. The information stored in the QR code is shown in Fig. 6. By using this QR code, each agent would be able to access the agent front-end.

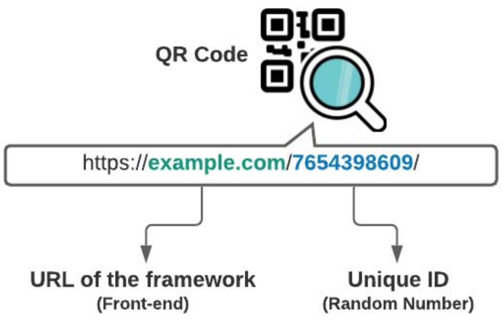

Fig. 6 Content of QR code

When the package is handled to the first agent by the seller of the product, the first agent will scan the QR code printed on the package, and they will be able to access the agent's front-end. Based on different ACL conditions, we can implement different front-ends for agents as illustrated in Fig. 7; for example, we can add 2FA (Two Factor Authentication). Agent sends a reveal request to the smart contract, and if their Ethereum address is on the list of the agents added by the seller, our smart contract will send the first piece of shipment address shown in Fig. 4 which is the name of the country. At the first step of the delivery process, the country name is the only required part of the shipment address that the agent needs to process the package.

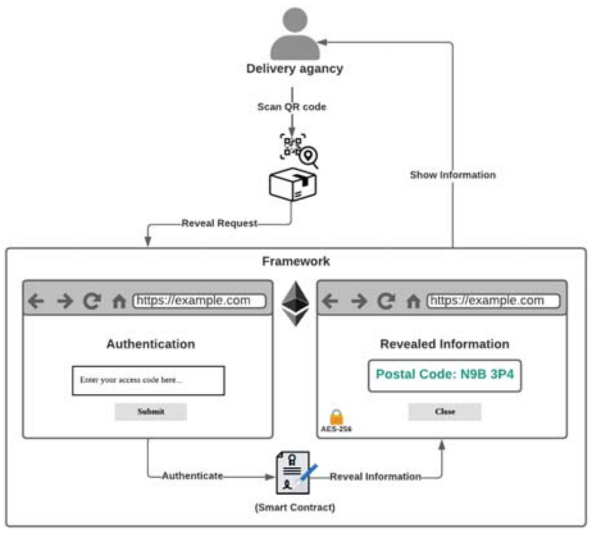

Fig. 7 Front-end of the framework for delivery agents with 2FA implemented

The revel request result is encrypted data that will be decrypted on the agent's front-end. If someone checks the Ethereum transaction, they see only encrypted data rather than plain text.

If agents reveal all shipment pieces, any incoming reveal requests will be rejected after that by the smart contract.

Lastly, we would like to mention that we can change the information requested on the buyer front-end based on seller needs. Also, our framework act as an escrow to store the AES-256 key. We will provide some solutions in Section V to remove this responsibility from our proposed framework.

Algorithm 1 Pseudocode of the smart contract

```
procedure SMARTCONTRACT
    $agentAddr$ = Addresses of agents;
    $Stages$ = Stages of the reveal proccess;
    $CommitChoice$;                                ▷ Struct data type
    function ADDAGENT($address$)
        modifier(isOwner);
        $agentAddr[address]$ = true;
        return true;
    end function
    function COMMITADDRESSES($dH1$,$dH2$,...,)
        modifier(isNotAgentandOwner);
        $stage$ = Stages.InitialCommit;
        $commitIndex$ = 0;
        while $dH$ do
            commit($dH$);
        end while
        $stage$ = Stages.FirstReveal;
        $orderID$ = random();
        $committed$ = true;
    end function
    event Commit(bool);
    function COMMIT($dataHash$)
        require(stage == Stages.InitialCommit);
        $commitChoice$ = dataHash;
        commits.push($commitChoice$);
    end function
    event Reveal(bool);
    function REVEALADDRESS
        modifier(isAgent);
        Check stage status;
        $agentAddr[msg.sender]$ = false;
        require($commitIndex < commits.length$);
        require(!commits[commitIndex].revealed);
        $commits[commitIndex].revealed$ = true;
        Change stage status;
    end function
    function RANDOM
        Generate a random number;
    end function
end procedure
```

E. Method Comparison

Following, we discuss some of the important features of the proposed method compared to related works. A summary of comparison with one of the recent related works [7] is provided in Table I.

One of the biggest differences between the proposed method and similar works is that other methods manage the online market, payment and shipping under one big platform. This has some cons, for example: centralizing all of the buyer information under one platform is very costly to integrate with

TABLE I
COMPARISON OF THE PROPOSED METHOD WITH A SIMILAR WORK

Features	[7]	Proposed Method
Availability	N	Y
Auditability	Y	Y
Decentralized	Y	Y
Flexibility	N	Y
Independent	N	Y
Privacy-Preserving of PII	N/A	Y

current systems (because the whole system should change), and most of those methods need third-parties to work properly. Fig. 8 shows the proposed method, and Fig. 9 shows how other methods work.

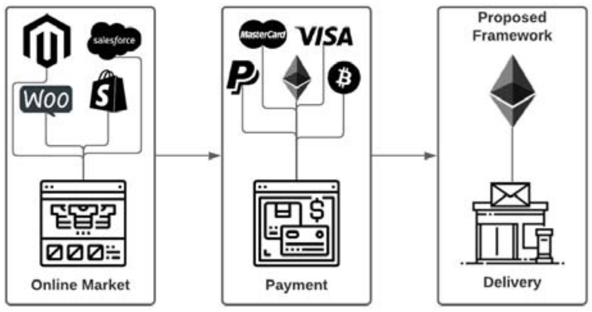

Fig. 8 Proposed method (each section is handled in separate platforms)

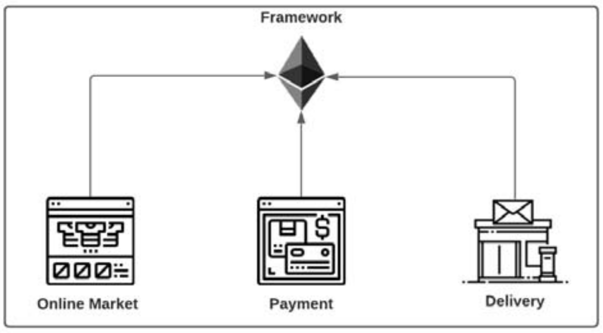

Fig. 9 Compared methods (everything handled in one platform)

IV. EXPERIMENTS AND RESULTS

In this section, we will have a look at the proof of concept of the proposed method. Also, we evaluated the performance of our proposed method to show its viability in a distributed environment. The performance is evaluated by studying the time and space complexity, security analysis and the gas cost estimates for deploying/running the smart contract.

Several tools have been used in our work. Fig. 10 shows tools that we used on each section of the proposed framework.

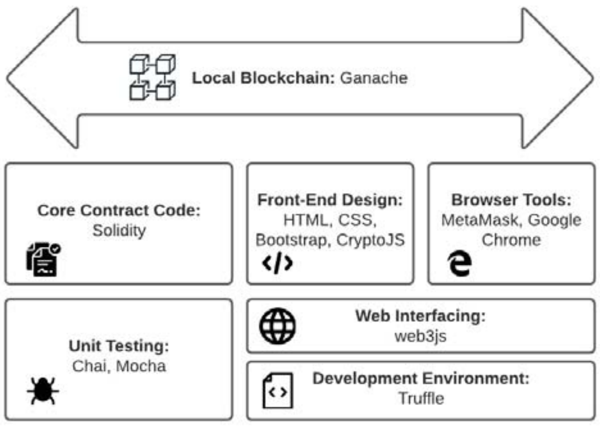
Local Blockchain: Ganache
Core Contract Code: Solidity
Front-End Design: HTML, CSS, Bootstrap, CryptoJS
Browser Tools: MetaMask, Google Chrome
Unit Testing: Chai, Mocha
Web Interfacing: web3js
Development Environment: Truffle

Fig. 10 Tools used to create each section of the proposed framework

A. System Configuration

We used the following described system as a server, seller view, and buyer view during the experiments.

TABLE II
SYSTEM SPECIFICATIONS

Operating System	MacOS Monterey v12.2.1
Computer Model	MacBook Pro
Processor	Apple M1 Pro
Memory	16 GB
Personal Ethereum	Ganache v2.5.4
Browser	Google Chrome v99.0.4844.51
Gateway to Blockchain	MetaMask v10.10.2
Development Environment	Truffle v5.4.2

B. Experiments

As a proof of concept, we have implemented a working prototype of smart contract which is available as an open source project at [14]. The smart contract is implemented in 147 lines of Solidity.

Proposed smart contract has seven functions in total. *addAgent* function is responsible from adding a list of agents who they can send reveal requests. *getOrdernumber* is responsible from returning a unique order number generated by *random* function. *commitAddresses* function gets the buyer shipment information and calls the *commit* function to commit information on the blockchain network.
textitrevealAddress is responsible for getting reveal requests from agents and sending back the desired data to them. At the end we have *isContract* function that is responsible from checking the Etherum address type of sender. Generating QR code and all of the cryptography happens on the front-end.

C. Time and Space Complexity

Table III shows the time and space complexity of each function that we used in the proposed smart contract. In Table III n indicates the input size.

As shown in Table III most of our functions have $O(1)$ which means constant space and time complexity. The worst

TABLE III
TIME & SPACE COMPLEXITY (WORST CASE)

Function	Time complexity	Space complexity
$addAgent$	$O(1)$	$O(1)$
$getOrdernumber$	$O(1)$	$O(1)$
$commit$	$O(1)$	$O(n)$
$commitAddresses$	$O(n)$	$O(n)$
$revealAddress$	$O(1)$	$O(1)$
$random$	$O(1)$	$O(1)$
$isContract$	$O(1)$	$O(1)$

time and space complexity is $O(n)$ which is often called linear complexity. This means that the running time increases at most linearly with the size of the input.

D. Response Time

Average response times of each function on the smart contract shown in Table IV.

TABLE IV
AVERAGE RESPONSE TIMES OF EACH FUNCTION ON THE SMART CONTRACT

Function	Avg Response Time(s)
Deployment	5.7897 sec
Commit	969.3791 msec
Reveal	778.6979 msec
AddAgent	792.6899 msec

E. Cost Analysis

The gas cost estimates for deploying and running our smart contract are provided using truffle Solidity framework in TableV. These estimates are evaluated as of February 2022, 1 unit of gas = 18 * 10^{-9} ether, and 1 ether = 2,519.78 USD.

TABLE V
ESTIMATES FOR GAS COST IN USD FOR DEPLOYMENT AND DIFFERENT FUNCTIONS OF THE SMART CONTRACT

Function	Gas units	Gas cost (USD)
Deployment	684771	1.6125
Commit	376085	0.885629
Reveal	43175	0.101671
AddAgent	27160	0.063958

F. Number of Agents

Our framework's number of participating agents is another important variable impacting the required calculations and gas usage. In this experiment, we compare two different cases. In the first case, we assume that we do not know the total of agent's numbers involved in the delivery process, and we assume that we will add them ongoing. We assume we know the exact number of delivery agents in the second case at the first step. In our main experiment, we used case one, but as we showed in Fig. 11 we can reduce the cost by 81.63% but time/space complexity changes to $O(n)$.

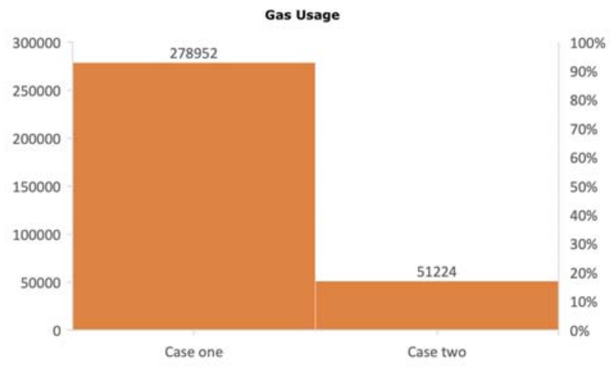

Fig. 11 Gas usage comparison for adding six agents on two different case

G. Security Analysis

In this section, given our operation and adversarial assumptions, we reason about our security claims for the properties provided by our method. Note that we do not intend to give formal proofs for such properties, mainly because our system relies heavily on the physical world and modelling the physical world is outside this work's scope.

We assessed the proposed smart contract against common solidity vulnerabilities listed below. The proposed smart contract is protected against these attacks.

Accessing Private Data: When a contract uses the private modifier on a function or field, it does not mean that these variables cannot be read. An attacker can look at the transactions related to this contract on the public blockchain to figure out the state of all variables. We encrypt all sensitive information using AES encryption algorithms on the client-side, which means even attackers could access any private modifier; those data will be useless for them.

Arithmetic Overflow and Underflow: An overflow happens when a number reaches incremented beyond its maximum value. Assume we declare an uint8 variable, an unsigned variable that can accept up to 8 bits. This indicates that it can have decimal numbers between 0 and $2^8 - 1 = 255$. $uint8a = 255$; $a++$; this will lead to an overflow because a's maximum value is 255. All input from buyers is a string type and none of the inputs is integer type.

Arbitrary data write: Similar to how in a standard buffer overflow, an attacker can overwrite critical data such as a function return address, there exists a similar problem where anyone may overwrite the owner of a contract. To ensure that there are no out-of-bounds accesses, we implemented a couple of *modifier*.

Re-Entrancy: A reentrancy attack can occur when you create a function that makes an external call to another untrusted contract before it resolves any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the effects were resolved. To protect our smart contract against this vulnerability, we implemented the *isContract* function that makes sure only wallets can access our smart contract.

V. Conclusions and Future Work

This paper has presented a blockchain solution that privacy-preserving shipment information from sellers, delivery agents servers, and physical packages by dividing and storing shipment information on the blockchain network by revealing only required shipment information to agents and removing shipment labels that have some personal information in plaintext. We have presented architecture details, system components and algorithms, and their implementation. We have also analyzed costs associated with executing functions within our smart contracts to show that it is a solution that is affordable for users. We conducted a security analysis for the smart contract to check its robustness against well-known vulnerabilities.

This framework is also applicable in other industries such as finance, military, aerospace, automotive, and education. It can be applied to any industry or government organization where there is a need to protect confidential information. This framework can secure sensitive data by providing only specific access to authorized users.

In the future, the proposed solution can be improved by: first, building a private permissioned blockchain can be configured to eliminate gas prices. Second, we can use a zero-knowledge proof or zero-knowledge protocol to improve privacy further.

References

[1] Dapps for Beginners. *Two party contracts*. https://dappsforbeginners.wordpress.com/tutorials/two-party-contracts/. 2018.

[2] Abdeljalil Beniiche. "A study of blockchain oracles". In: *arXiv preprint arXiv:2004.07140* (2020).

[3] BitBay. *Double Deposit Escrow – BitBay*. https://bitbay.market/double-deposit-escrow. 2018.

[4] localethereum.com's official blog. *How Our Escrow Smart Contract Works*. https://blog.localethereum.com/how-our-escrow-smart-contract-works/. 2018.

[5] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". In: *white paper* 3.37 (2014).

[6] David Chaum. "Security without identification: Transaction systems to make big brother obsolete". In: *Communications of the ACM* 28.10 (1985), pp. 1030–1044.

[7] Haya R Hasan and Khaled Salah. "Blockchain-based solution for proof of delivery of physical assets". In: *International Conference on Blockchain*. Springer. 2018, pp. 139–152.

[8] Malte Möser, Rainer Böhme, and Dominic Breuker. "An inquiry into money laundering tools in the Bitcoin ecosystem". In: *2013 APWG eCrime researchers summit*. Ieee. 2013, pp. 1–14.

[9] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[10] Dorit Ron and Adi Shamir. "How did dread pirate roberts acquire and protect his bitcoin wealth?" In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 3–15.

[11] Dorit Ron and Adi Shamir. "Quantitative analysis of the full bitcoin transaction graph". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 6–24.

[12] *Truffle suite*. 2016. URL: https://trufflesuite.com.

[13] Dejan Vujičić, Dijana Jagodić, and Siniša Ranić. "Blockchain technology, bitcoin, and Ethereum: A brief overview". In: *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE. 2018, pp. 1–6.

[14] Shahin Zanbaghi and Saeed Samet. *A Blockchain-based Privacy-Preseving Physical Delivery System*. Version 1.0.4. 2022. URL: https://github.com/ShahinZa/PPPDA.