Received July 13, 2018, accepted August 16, 2018, date of publication August 21, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2866512

Blockchain-Based Proof of Delivery of Physical Assets With Single and Multiple Transporters

HAYA R. HASAN AND KHALED SALAH, (Senior Member, IEEE)
Electrical and Computer Engineering Department, Khalifa University, Abu Dhabi 127788, UAE

Corresponding author: Khaled Salah (khaled.salah@kustar.ac.ae)

This work was supported by ICT Fund, UAE.

ABSTRACT With the widespread of E-commerce, the need of a trusted system to ensure the delivery of traded items is crucial. Current proof of delivery (PoD) systems lacks transparency, traceability, and credibility. These systems are mostly centralized and rely on trusted third parties (TTPs) to complete the delivery between sellers and buyers. TTPs can be costly, a single point of failure, and subject to hacking, privacy evasion, and compromise. The blockchain is an immutable, trusted, and decentralized ledger with logs and events that can be used for transparency, traceability, and tracking. In this paper, we present a solution and a general framework using the popular permissionless Ethereum blockchain to create a trusted, decentralized PoD system that ensures accountability, auditability, and integrity. The solution uses Ethereum smart contracts to prove the delivery of a shipped item between a seller and a buyer irrespective of the number of intermediate transporters needed. In our proposed solution, all participating entities are incentivized to act honestly by using a double deposit collateral. Automated payment in ether is an integral part of a solution to ensure that every entity gets its intended share of ether upon successful delivery. An arbitration mechanism is also incorporated if a dispute arises during the shipping process. In this paper, we show how we implemented, verified, and tested the proper functionality of our PoD solution. We also provide security analysis and give estimates of the cost consumption in ether gas. We made the full code of the Ethereum smart contracts publicly available at Github.

INDEX TERMS Blockchain, Ethereum, smart contracts, cyber security, security analysis, decentralized management.

I. INTRODUCTION

Online shopping has become the most convenient way for most people to shop and buy goods these days. Online shopping offers consumers the ability to enormously save time, compare prices, check reviews, similar products, or current trends. E-commerce shopping is getting more favored with time especially with the widespread of smart phones and the ease of accessibility to the Internet [1]. According to the UPS, the percentage of smart phone online purchasers has increased to 77% compared to 55% in 2015 [2]. Consequently, in order to meet the increase in demand, delivery services are now offered by vendors to empower the shopper and elevate their experience. Therefore, proof of delivery of a traded physical items and products is immensely needed to facilitate the shipment in a way that is trusted, transparent, and cost-efficient, especially if the the seller, buyer, and transporters are located globally in different countries and can not be trusted. Proof of delivery ensures that the shipped item has reached its entitled destination, with the ability of providing evidence to all involved parties of the state of shipment as it moves from the seller, intermediate transporters, and buyer.

Current Proof of Delivery (PoD) systems lack transparency, traceability and credibility. For instance, a large number of PoD services depend on signed papers and documents as a way of authentication and hence, proving the delivery to the recipient. Those papers are typically carried along with the transporter. However, there are other PoD systems which rely on hand-held electronic devices for the authentication procedure. Hence, the recipient would provide an electronic signature and a valid identification card (ID). Then it would depend on the transporter to validate the provided documents and signature and ensure that the item is given to the right intended recipient. Such method relies on the honesty of the transporter and the recipient as the ID can be faked. On the other hand, online retailers may not always have their own shipping services. Some companies would rely on a trusted third party for managing the delivery. For example, Amazon depends on different courier services for their national and regional delivery services. FedEx, UPS and DHL are some of the companies that Amazon relies on for its regional shipments [3].

Furthermore, current systems are mostly centralized, relying on trusted third parties (TTPs) to complete the delivery between a seller and a buyer. Such systems are hard to manage and are costly as they involve TTPs. Not only this, but TTPs can be a single point of failure, and are subject to hacking, privacy evasion and compromise. Thus, making them unreliable and not trustworthy [4]. As a result, there is an immense need for a decentralized solution that provides PoD of physical items as well as traceability in a secure, and trusted method without relying on a third party.

Blockchain is an immutable, tamper-proof, decentralized distributed ledger [5], [6] with ample security features [7] that make it possible to create the needed PoD solution. Blockchain uses ordered logs and events which are used to achieve traceability and auditability. In addition, using Ethereum makes blockchain programmable. Ethereum smart contracts empowered blockchain by allowing the execution of code [8].

A sound PoD solution for traded physical assets, items, products, or goods must fulfill the following key requirements [4]. **Accountability**, which is the ability to attribute certain actions to a particular actor. Accountability is similar to non-repudiation where the actor can not deny a committed transaction. **Penalty and Incentivization**, which ensure that the participating entities have the incentives to act honestly; otherwise, a penalty will be incurred. **Auditability**, where the system provides a mechanism to trace and track back events and actions, in a way that is completely secure and trusted. **Integrity**, where all transactions, logs and events are timestamped and tamper-proof. **Authentication and Authorization** which ensure that certain functions and operations can only be carried out by specific actors. **Timebound** which ensures that the item reaches its final destination within a specific time frame. Furthermore, another important requirement is **Off-chain Arbitration** which allows for a judging entity to settle any dispute in case of false claims by any actor. The chosen arbitrator has full access control to assembled funds which can be dispersed by the arbitrator after examining the evidence on the ledger.

A preliminary short conference paper on PoD for physical assets has been published [4]. In sharp contrast, this article is different in so many substantial and significant ways. Firstly, in this work the proposed PoD solution is for single and multiple transporters giving high importance to multiple transporters where as in [4], the solution is devised to only fit a single transporter between a seller and a buyer. Secondly, the implementation in [4] uses two keys which is cumbersome for multiple transporters. Hence, in this article, the implementation is optimized to utilize the security features of blockchain in a better way and use only one key for the verification. Furthermore and this is one of the major and important differences, the solution in [4] uses only one simple contract. In sharp contrast, the proposed solution in this paper is made from three different types of contracts. The contracts are linked to each other like a chain to account any number of intermediate transporters. In this paper, we also highlight the gas consumption as well as the cost in USD dollars. Furthermore, this paper provides thorough security analysis of the proposed solution.

In this paper, we propose a blockchain-based solution and general framework for the proof of delivery of physical items involving single and multiple transporters. Our solution focuses on solving the problems that the current centralized systems suffer from such as the centralization caused by the use of TTPs. Thus, our solution is decentralized, eliminates the trust issue and the need for a TTP, and provides the ability to trace logs without the need of a trusted third party and utilizes off the chain arbitration to handle disputes. The main contributions of this paper can be summarized as follows:

- We propose a PoD system and framework between a seller and a buyer irrespective of the number of transporters.
- We automate payments in Ether tokens and employ incentives and penalties to force involved actors to act honestly.
- We make use of InterPlanetary File System (IPFS) to ensure integrity for the agreed-by terms and conditions, with a link of the IPFS hash to the Ethereum smart contracts.
- We address cancelation as well as refund in case of delay. We also incorporate an off-chain arbitration mechanism where a chosen arbitrator is given control over funds to settle a dispute.
- We discuss and detail out key aspects related to implementation algorithms, testing and validation.
- We provide cost analysis of the solution in terms of Ether gas, and we discuss how the solution meets the key security objectives and how it is robust against common cyber security attacks.

The remainder of this paper is organized as follows. Section II presents the related work. Section III details out the proposed blockchain Ethereum solution implementation. Section IV provides the testing details and a discussion on the security and cost analysis. Section V concludes the paper, and discusses future work.

II. RELATED WORK

In this section, we review and summarize work related to proof of delivery algorithms and techniques that make use of blockchain. We also survey decentralized blockchain-based marketplaces.

The authors in [9] proposed a simple scheme based on Ethereum blockchain which involves transporting a product between two parties. The scheme depends on a single key that is given to the transporter by the seller [9]. The key is transported along with the item and is handed over to the receiver who is the buyer. The buyer then needs to enter the key for verification. The key hash would already be available in the smart contract which acts as an escrow and holds the buyer's Ether. The Ether would only be placed in the seller's account if the hash of the key entered by the buyer matches the existing hash in the smart contract. A successful verification leads to the transfer of the Ether to the seller [9]. This solution is easy to implement as it is simple and depends on only one key. The method however, depends on trusting the transporter completely that no manipulation of the key would take place before reaching the buyer. It also lacks incentives to keep all parties involved honest, although this approach fails with a malicious act from any participating entity especially the transporter.

A solution that utilizes the blockchain technology to create an online decentralized peer to peer marketplace for trading Ether is called 'localEthereum' [10]. LocalEthereum does not depend on the contract to act as an escrow [10]. There technique relies on a trusted third party which is a funded escrow agreed upon by both the seller and the buyer. This method requires trusting the third party and costs more as it requires paying the escrow. Furthermore, the seller places the Ether with the funded escrow and the buyer provides the payment directly to the seller. If the seller confirms the payment, the trade is complete and the seller would release the escrow. However, if a dispute arises, an arbitrator (which is localEthereum) is setup to settle the dispute. Clearly, this solution is costly as it depends on a TTP to act as an escrow, also it does not give the buyer and the seller, the ability to choose their trusted arbitrator in the case of any dispute. It also does not show ways of incentivizing the participating entities and the transporter is not involved in any on the chain logs.

A similar concept to localEthereum is 'OpenBazaar' which relies on a funded escrow that is agreed upon by the buyer and seller or traders which is knows as a 'Multisignature escrow' [11]. The funded escrow would also act as a moderator in the case of dispute. Therefore, there are three parties involved in the process of selling an item. The seller, buyer and the moderator. The buyer would place the Bitcoin (which is the most commonly used currency in OpenBazaar) in the funded escrow. The payment will only be released to the agreed upon destination based on the major votes of the participants. Two out of the three votes would determine the destination of the payment transfer [11]. This method does not involve the transporter in any on the chain procedures. The transporter is trusted completely without any incentives to take the item from the seller and deliver it to the buyer off the chain. Hence, the transporter is not tracked on the chain.

'SOMA' is another decentralized marketplace which facilitates trade with the use of the social platform [12]. SOMA uses blockchain to add a social aspect to a decentralized marketplace which was inspired from the created groups on Facebook for trading. SOMA provides users the ability to promote their items. Each physical item possesses a digital representation using an Interactive Item Card (IIC). The owner chooses to promote the IIC of it's item when they decided on selling it. The owner can also allow other SOMA users to promote the item. The IIC has all the history of the item, this includes ownership history, authenticity and its social value. Promoters are paid a reward of SOMA Community Tokens (SCT) if the item was sold successfully. This reward is held with the 'prize contract' and it would return to the owner after a compensation if the item does not get sold. This reward system creates an incentive for the platform users to collaborate with one another. SOMA uses Solidity for writing its smart contracts and uses a server to store the item's information. Only the data signature, key and database link are stored in the blockchain. However one major drawback is that the transportation is done off the chain and is not part of the reward system [12]. Hence, tracing and tracking cannot be done using the blockchain logs.

'Syscoin Blockmarket' employes blockchain to offer another decentralized marketplace for trading of goods including small digital documents [13]. The seller and buyer agree on a trusted third party arbitrator to act as an escrow. The arbitrators receive fees for their contribution. Each of the platform users has a reputation to create an incentive for everyone to act honest. Also, the system enables notifications to keep the users updated about the shipment state. In addition, a proof of shipment is done by allowing the seller to take a video of the sent package and hashing it. The hash is included as part of the transaction details. This is made to help in resolving disputes faster. The system also encourages the use of aliases for Ethereum addresses. Lastly, the system allows the trade of small size digital documents by using digital certificates [13]. Although the system creates an incentive for the seller, buyer and arbitrator to act honest through a reputation based solution, the system ignores the transporter and the importance of including it on the chain to have better transparency and traceability. Also, the arbitrator acts as an escrow and is always available to hold the funds even of there is no dispute. Hence, this solution can be a single point of failure if the arbitrator decides to be dishonest.

'BitBay' is another decentralized marketplace that allows trading as well as buying and selling [14]. BitBay uses blockchain to create a decentralized marketplace and relies on a smart contract to implement a 'double-deposit escrow' to incenitvize all parties to act honestly. This requires the buyer and seller to deposit twice the item price to the contract. If anything goes wrong, everyone loses the deposited collateral. There is no arbitrator to verify and resolve disputes [14]. Also, the transporter is not included as a key participant in the shipping process.

All of these existing blockchain decentralized approaches have clear limitations that need improvement to have a more efficient and complete system that includes a PoD solution. Firstly, in [9]–[11] there is no incentive to any of the participating entities to act honest. The seller, buyer and transporter are all trusted completely. Secondly, [10], [11], and [13] depend on a trusted TTP or arbitrator to act as an escrow and hold all the funds from the beginning of the selling process till its end. Having a TTP that holds the funds can be a single centralized point of failure and is also costly. Furthermore, [9], [12], and [14] do not have a mechanism to resolve disputes if any occur. Hence, there will be a lost to the seller, buyer or both for any act of dishonesty. Additionally, all of the above solutions with the exception of the first one [9] are decentralized markets that do not provide decentralized PoD solutions. References [10]–[14] keep the transporter off chain and do not involve the transporter in any incentives or logs that take place on the chain. Hence, this makes tracing and tracking more complex and does not utilize the blockchain to prove the delivery of the goods. In our solution, we create a decentralized PoD solution that not only involves arbitration in the case of disputes and incentivizes all the participating parties to act honest without the use of a TTP, but also involves single and multiple transporters to be on the chain to ease tracing and tracking as well as the resolving of disputes.

III. PROPOSED BLOCKCHAIN SOLUTION

In this section, we present our Ethereum blockchain solution that utilizes the security features of the technology to create a solution for PoD between a seller and a buyer involving a single or multiple transporters. The seller and buyer could be located a few miles a part or in different countries. Our solution can be extended to as many transporters (or courier services) as required and works in an adequate, reliable and secure way similar to working with only one transporter. Our solution incentivizes all the parties to act honest by ensuring a collateral is deposited by each participating entity. Also, unlike other solutions, the transporters are part of the on chain system and play a role in the PoD solution proposed. This increases trust and security and helps in resolving disputes accurately.

A. SYSTEM OVERVIEW

The proposed blockchain solution focuses on the proof of delivery of traded physical assets between two parties. Figure 1 shows the main participating entities of the system, the seller, buyer, courier service(s), arbitrator and the smart contract attestation authority (SCAA). Each of the entities has an Ethereum address and interacts with the smart contracts created throughout the process based on permissions. As the item gets handed over between two entities a chain of contracts is created based on the number of courier services. However, at least two contracts are required between a seller and a buyer. Moreover, the terms and conditions of the agreement between the seller, buyer and courier service(s) should be signed and agreed upon before starting the transaction. Therefore, the InterPlanetary File System (IPFS) hash of the terms and conditions agreement is part of the contracts created. If all entities agree, the transaction starts and the agreed upon collateral is withdrawn from the main entities i.e. the seller, buyer and the transporter. The roles of the participants can be summarized as follows:

- **Seller:** The seller has the item to be packaged for transfer to the interested buyer. The seller creates the first contract in the chain. Therefore, the seller is the owner of the first contract created.

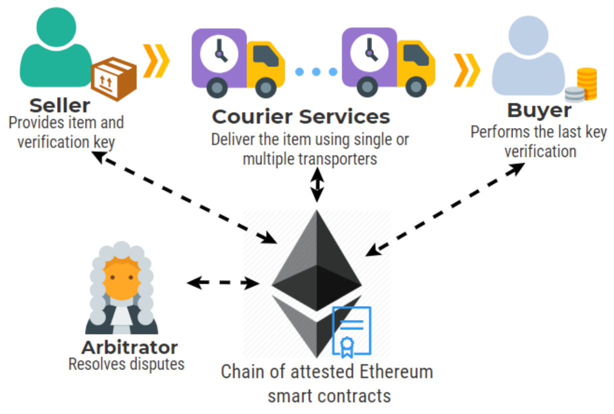

Seller
Provides item and verification key
Courier Services
Deliver the item using single or multiple transporters
Buyer
Performs the last key verification
Arbitrator
Resolves disputes
Chain of attested Ethereum smart contracts

FIGURE 1. **The different participating entities and their interaction with the attested smart contracts in the chain.**

- **Buyer:** The buyer would like to spend Ether and buy the item from the seller.
- **Courier Service(s):** Multiple couriers are available to deliver the item from the seller to the buyer if needed based on the geo-location of the seller and buyer. The transporter creates the next contract in the chain.
- **Arbitrator:** The arbitrator is a trusted entity by the seller, courier services and buyer. Its main task is to ensure that the rights of each participating entity is preserved in the case of dispute. The blockchain-solution is completely arbitrator independent during normal transactions. The arbitrator involvement is minimal, and is not involved in every transaction under normal behavior. Furthermore, the arbitrator has no ability to reverse, alter or fake the order and actions of the buyer sellers, or transporters. Any Ether that was deposited to the contract(s) gets transferred to the arbitrator only if the transaction fails and will be then redistributed based on the results of the off-chain arbitration.
- **Smart Contract Attestation Authority (SCAA):** SCAA attests each contract in the chain to ensure that the code satisfies the agreed upon terms and conditions. If a contract is attested by the SCAA, the contract address would be part of the SCAAs contract and the SCAAs address would be included in the contract. Therefore, the attested contract and the SCAAs contract are pointing to each other.

One of the benefits of the blockchain technology is transparency and the ability to log everything on the public ledger. Therefore, the Ethereum smart contracts used in the chain utilize this property to create events and logs that help in tracking the item as it gets packaged, delivered and passed from one entity to another. Furthermore, a key hash is used to confirm the receiving of an item by the transporters and buyer. Hence, the key is given physically to the next courier or finally to the buyer and then its hash is created and compared to the hash already available in the smart contract. This verification is done to ensure that the item has been truly received by a certain entity. In order to achieve the needed functionality with transparency and tracing the item as it moves through the chain of contracts, the smart contracts contain the following:

- **Methods:** Methods are used in smart contracts to create function calls. Each function is responsible for executing and implementing a desired action. Hence, in this work, some of the important functions we have created include methods to deposit the collateral, perform the key verification between any two parties as well as settle the payment and handle the dispute. All public variables have automatic getter functions created for them. However,setters have to be created as required. Hence, to change the state of a contract a setter function was created to allow only its parent or its own child to alter its state.
- **Modifiers:** Modifiers are used in the smart contracts to create a requirement before the execution of a function. For instance, the collateral should be a certain agreed upon amount. This is checked using a modifier. Other modifiers were also used to restrict the execution of a function based on the Ethereum address of the function caller. Therefore, certain functions can only be executed by the seller, others by the transporters and buyer respectively.
- **Events:** Events act as notifications and are used as logs which can help in tracing back in case of dispute. Therefore, any function that is executed creates an event that updates all entities about the status of the item and contract until now.
- **Variables:** Variables are used to store information that might change as the transaction progresses or that are needed for certain checks and functionalities. Therefore, the main variables in the contracts are used to store Ethereum addresses of the participating entities, the key hash that is used in the key verification comparison, the item price, the contract state, IPFS hash and the address of the child contract for each parent contract in the chain.

B. SYSTEM DESIGN

In order to deliver the item between he seller and the buyer and to create a solution that adapts to the number of courier services required, three types of contracts are designed. The contracts are created based on the need, and togEther they make a chain of contracts. Each contract points to the next contract. Therefore, every parent contract has the address of its child contract and every child has the address of its parent contract. In addition, all contracts have the address of the first main contract that started the chain. The main contract has an additional address as well, which is the address of the last contract in the chain.

The chain should have at least two contracts. Therefore, this indicates that if only one transporter is needed, two contracts will be created. However, if more than one transporters are needed, then at least 3 contracts are created. It also always starts with a contract of the type Proof of Delivery (PoD) and ends with a contract of the type Buyer Transporter (BT). Therefore, PoD is the main contract and BT is the end of chain contract. In the middle, if the number of transporters is greater than one then contracts of the type Courier Service (CS) are created as needed.

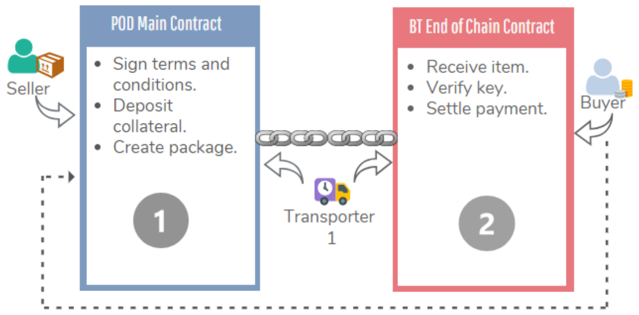


FIGURE 2. A chain of two contracts showing the interaction among actors involving a single transporter.

Figure 2 shows the chain of contracts when there is one transporter only. Therefore, the chain is made of only two contracts, the PoD main contract and the BT end of chain contract. Figure 2 also illustrates the entities that interact with each contract. The seller interacts with the PoD contract only, while the transporter and the buyer are part of both the contracts. Moreover, all of them deposit their collateral in the PoD contract.

On the other hand, Figure 3 illustrates a chain of three transporters. Hence, there are two other contracts of type Courier Service between the PoD and the BT contracts. The number of Courier Service contracts required is always less than the number of transporters by 1.

Figure 3 also shows the entities participating in each contract across the chain. In the PoD contract, the seller, buyer and first transporter sign the terms and conditions and deposit the agreed upon collateral. Later, the seller would create the package and physically hand it over to the transporter along with a key. The transporter would then create the next CS contract and Transporter 2 agrees to the terms and conditions and deposits a collateral which is held by the CS contract. Therefore, every contract acts as an escrow to the Ether deposited to it. Transporter 2 would then receive the packaged item and would notify everyone that Transporter 1 has arrived. This is an important step that will allow Transporter 1 to then confirm that it has reached and that the key is now with Transporter 2. Transporter 2 then enters the key which is hashed and compared to the key hash already available in the contract. If the verification is successful, the next CS contract is created by Transporter 2 and the chain goes on until the destination address is the address of the buyer. When the destination is the same as the buyers address, a BT contract is created, and the final key verification is done.

IV. IMPLEMENTATION DETAILS

The contracts are created and tested using the Remix IDE which provides the necessary tools for testing and debugging.

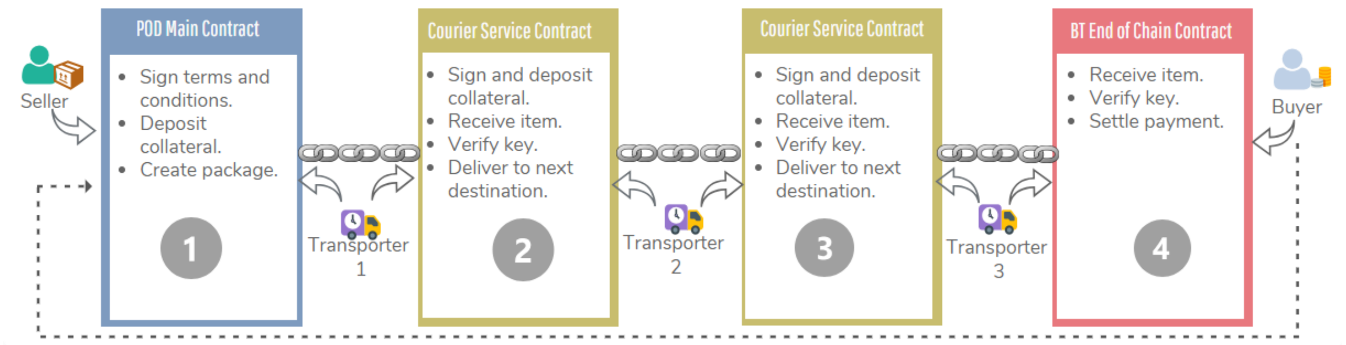


FIGURE 3. A chain of four contracts showing the interaction among actors involving multiple transporters.

Hence, making it easier to modify the code as needed while programming. This section focuses on the implementation algorithms used in the smart contracts.

The language used for writing the smart contracts is Solidity. Three types of contracts are created as mentioned in the Design section which are the PoD main contract, CS contracts that depend on the number of transporters and the BT end of chain contract. One of the essentials aspects in the implementation is that each contract across the chain has the contract addresses of both the parent and the child contract. This allows the transfer of funds to the intended parties during the automated payment settlement and dispute handling.

Moreover, every time the item is handed over in the chain, the receiver would first acknowledge the arrival of the transporter, then the transporter is allowed to confirm the arrival and provide the key for verification. This kind of ''handshake'' is of great significance as it helps in keeping both parties in need for each other, thus act honestly. Figure 4 illustrates a flowchart that presents the complete logic behind the chain of contracts created using the code.[1] The flowchart shows the full process cycle, commencing with the PoD main contract and ending with the BT end of chain contract.The rest of this section discusses the details of the important algorithms used in the code.

A. SIGNING TERMS AND CONDITIONS

Algorithm 1 shows the algorithm for the signing terms and conditions. For each contract created, new participants that have not signed the terms and conditions should do so in the newly created contract. Hence, in the first contract for instance, all participants i.e. the seller, buyer and Transporter 1 would sign the terms and conditions. In the next CS contract only Transporter 2 would do so and so on. In the last BT contract there is no signing of any terms and conditions.

It is to be noted that an IPFS hash is available in each contract that requires a signature. The participants are required to access the terms and conditions form off the chain and if they agree to its terms, sign on the chain. By signing the terms and conditions, the agreed upon collateral is deducted from

[1]https://github.com/smartcontract694/PoD_MultipleTransporters

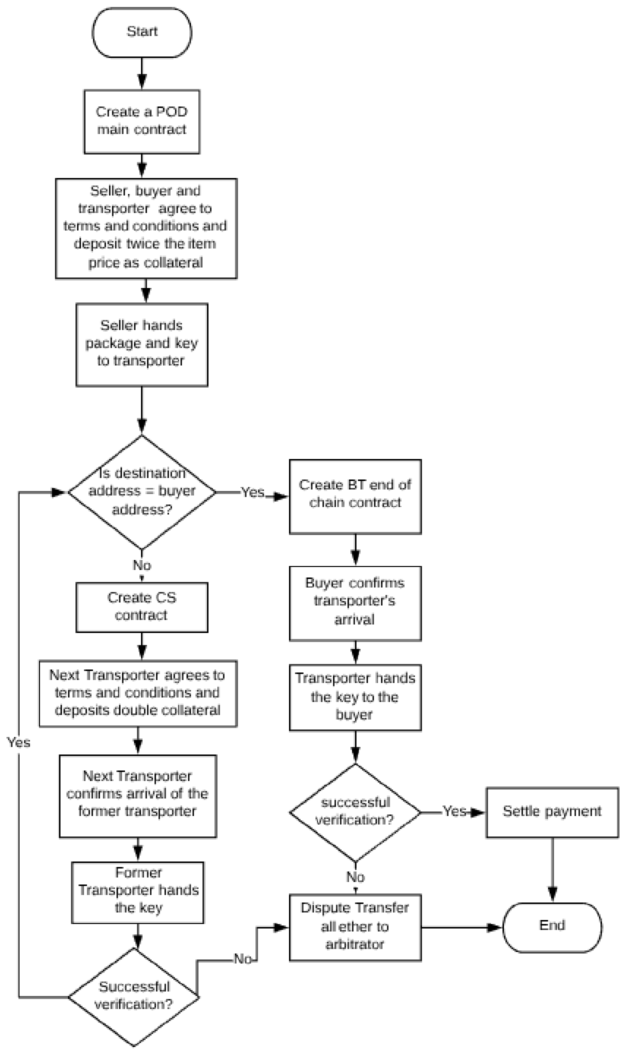

Start
Create a POD main contract
Seller, buyer and transporter agree to terms and conditions and deposit twice the item price as collateral
Seller hands package and key to transporter
Is destination address = buyer address?
Yes
Create BT end of chain contract
No
Create CS contract
Buyer confirms transporter's arrival
Next Transporter agrees to terms and conditions and deposits double collateral
Transporter hands the key to the buyer
Yes
Next Transporter confirms arrival of the former transporter
successful verification?
Yes
Settle payment
Former Transporter hands the key
No
Dispute Transfer all ether to arbitrator
End
No
Successful verification?

FIGURE 4. Flowchart exhibiting the workflow logic for chaining contracts.

the signing entity. The terms and conditions agreement form contains all the needed information about the item details, price, collateral and the process of transportation between he seller and buyer.

Algorithm 1 Signing Terms And Conditions

```
   Input : $E$, collateral, item price, contract state, IPFS
           Hash
1  $E$ is the set of all Ethereum addresses participating in
   this contract.
2  Restrict access to only $e \in E$ who did not sign yet.
3  if $collateral = item\ price$ then
4      foreach $e \in E$ in the smart contract do
5          if $contract\ state = waiting\ for\ signature\ of\ e$
           then
6              Withdraw required collateral from $e$.
7              Create a notification about the successful
               signature and deposit withdrawal.
8              Change state of the contract to the next
               state.
9          end
10         else
11             Revert contract state and show an error.
12         end
13     end
14 end
15 else
16     Revert contract state and show an error.
17 end
```

Algorithm 2 Canceling From PoD Main Contract

```
   Input : $E$, caller, cancelation state
1  $E$ is the set of all Ethereum addresses participating in
   the PoD contract.
2  if $caller \in E$ then
3      Check cancelation state of caller.
4      if $cancelation\ state = true$ then
5          Change contract state to Cancelation and
           Refund.
6          foreach $e \in E$ do
7              Return back the collateral.
8          end
9          Create a notification about the cancelation and
           refund of the deposited collateral.
10         Change state of the contract to Aborted.
11         Self destruct the contract.
12     end
13     else
14         Cancelation of transaction is denied.
15         Revert contract state and show an error.
16     end
17 end
18 else
19     Revert contract state and show an error.
20 end
```

B. CANCELATION AND REFUND

A transaction can be canceled by the seller, buyer and or the transporter only under certain conditions. Algorithm 2 shows the details of the cancelation algorithm that takes place from the main PoD contract. The input needed for the algorithm includes the address of the *caller*, and the cancelation state. The *cancelationstate* is a boolean which is initialized to true at first and only becomes false if the item is handed to the transporter or is on the way to the next destination. Any of the mentioned entities can decide on canceling the transaction. For instance, the seller and transporter can cancel the transaction, if the item has not yet been handed over to the transporter. However, the buyer has a chance to cancel as long as the item is not on its way to the next destination. If the item is getting delivered, no one can cancel the transaction. Therefore, the cancelation can only be called from the first contract and a refund of the deposited collateral takes place if the cancelation is accepted.

C. KEY VERIFICATION

Key verification happens every time the item is handed over to another participant. The receiving entity confirms first that the delivering transporter arrived, which creates a notification and a change in the contract state to "*Confirmation Done*". This change in state is a requirement for the delivering transporter to be able to hand in the item and the key to the receiver. Once the key is handed over, the contract state changes to "*Arrived To Destination*". Consequently, the receiving entity whEther the buyer or another transporter would then enter the key where the $keccak256$ hash is generated and compared to the existing hash in the contract. Algorithm 3 shows the full details of how the function that does the key verification works in the code. The input for the algorithm includes the *authorizedcaller* which should be the address of the buyer or next transporter and the *keyhash* which is the right hash of the key stored in the contract.

D. EXCEEDING DELIVERY TIME

Punctuality is crucial when dealing with delivery services. Therefore, since delivery time contributes in leaving a good impression on the customer and providing a good customer service, the buyer has the right to refuse taking the item if the delivery exceeded the expected delivery time. Algorithm 4 shows the details of the exceed delivery time algorithm that can only take place from the BT end of chain contract. Only the buyer is allowed to call the function and the function will only execute if the expected delivery time has been exceeded.

E. PAYMENT SETTLEMENT

When all the key verifications are successful throughout the chain, the item reaches the buyer by the last transporter. In the BT contract, the buyer confirms the arrival of the transporter and is handed over the item and the key. The buyer performs the last key verification. If the key verification is successful, only then, the payment is settled. This successful key verification automatically generates a call to the PoD

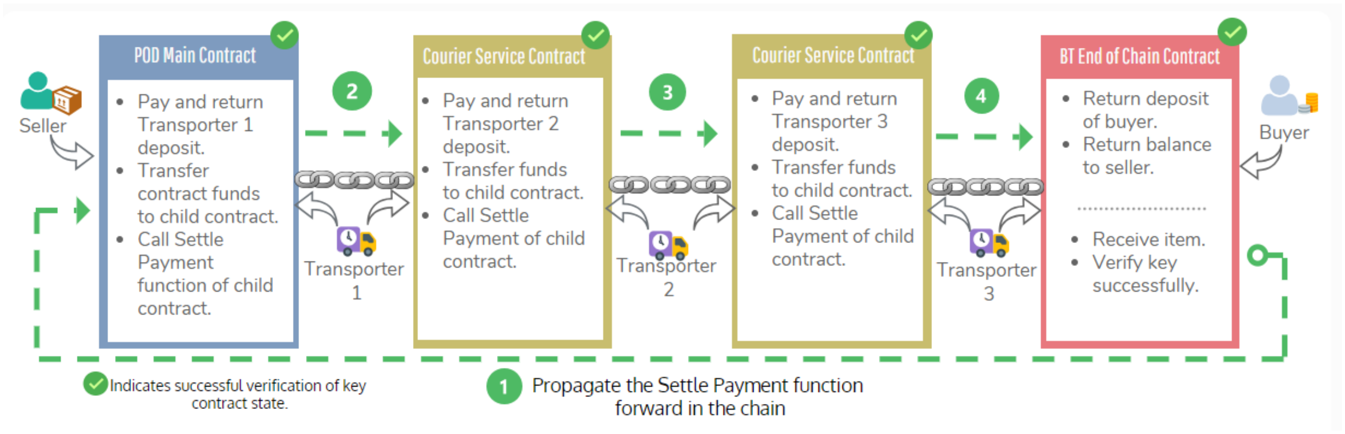

Seller
POD Main Contract
Pay and return Transporter 1 deposit.
Transfer contract funds to child contract.
Call Settle Payment function of child contract.
2
Transporter 1
Courier Service Contract
Pay and return Transporter 2 deposit.
Transfer funds to child contract.
Call Settle Payment of child contract.
3
Transporter 2
Courier Service Contract
Pay and return Transporter 3 deposit.
Transfer funds to child contract.
Call Settle Payment of child contract.
4
Transporter 3
BT End of Chain Contract
Return deposit of buyer.
Return balance to seller.
Receive item.
Verify key successfully.
Buyer
Indicates successful verification of key contract state.
1 Propagate the Settle Payment function forward in the chain

FIGURE 5. The process flow during payment settlement.

Algorithm 3 Verifying the Key

```
Input : authorized caller, caller, contract state, key,
        keyHash
1  if $caller = authorized\ caller \wedge contract\ state = Arrived\ to\ Destination$ then
2      Compute the $keccak256$ hash of the key.
3      $x \leftarrow keccak256(key)$
4      if $x = keyHash$ then
5          Create a notification about the successful
           verification.
6          Change state of the contract to Successful
           Verification by caller.
7      end
8      else
9          Create a notification about the verification
           failure.
10         Change state of the contract to verification
           failure.
11         Handle Dispute as shown in Algorithm 6.
12     end
13 end
14 else
15     Revert contract state and show an error.
16 end
```

main contract that will propagate a settle payment function in each contract in the chain as shown in Figure 5.

Consequently, each contract in the chain pays the deposit of each transporter (which is twice the item price) along with the transporter fees, then transfers the Ether held by the contract to the child contract. At the end, the last BT contract pays back the buyer the collateral, and the balance is returned to the seller. The state of each of the contracts is also changed along the way before executing its settle payment function.

Furthermore, in order for a child contract to receive Ether from another contract, a fall back payable function is available in the CS and BT contracts. The fall back functions are

Algorithm 4 Exceeding Delivery Time

```
Input : buyer, caller, contract state, current time,
        delivery time
1  if $caller = buyer \wedge contract\ state = Item\ on\ the\ way\ to\ buyer$ then
2      if $current\ time > delivery\ time$ then
3          Set state of the contract to Dispute Due to
           Exceeding Delivery Time.
4          Create a notification about the cancelation by
           the buyer.
5          Execute the dispute algorithm (Algorithm 6).
6      end
7  end
8  else
9      Revert contract state and show an error.
10 end
```

functions that have no name and their main functionality is to allow the contract to receive Ether. They also have an event to create log about the payment received. It is important to note that the address of the BT end of chain contract must be updated in the PoD main contract once created. This is required to restrict the access of the PoD settle payment function to the address of the BT contract. Therefore, any other address trying to execute the settle payment function will result in an error and a revert in state. Algorithm 5 shows the complete details of the payment settlement. In the algorithm, authorized caller is used to restrict access to only the BT contract or the parent contract. In addition, the input caller is the Ethereum address who created the transaction.The parent contract refers to the address of the parent of the calling contract and main contract is the PoD contract.

F. DISPUTE HANDLING

A dispute can arise at any point in the chain where the item has been handed over to another entity and key verification

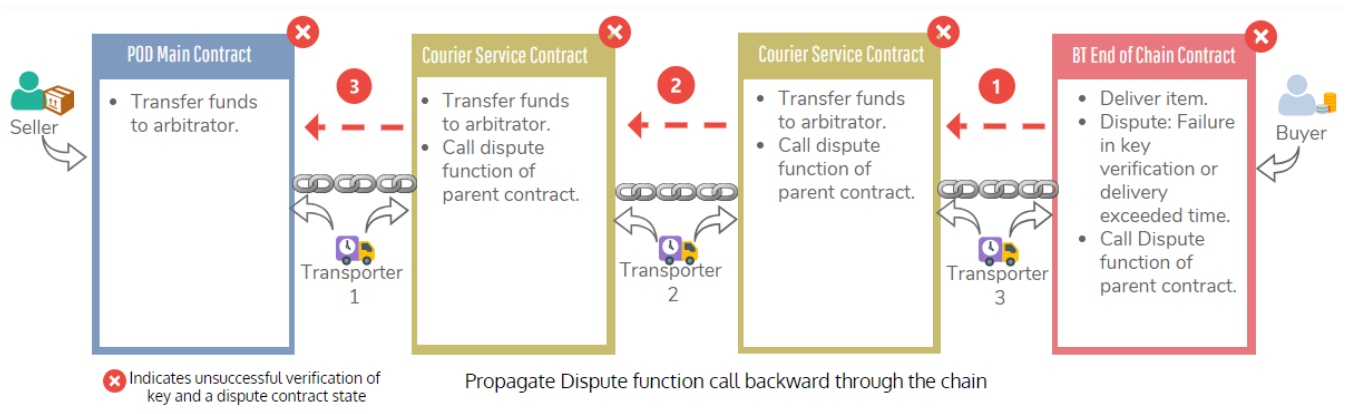

POD Main Contract
Transfer funds to arbitrator.
Courier Service Contract
Transfer funds to arbitrator.
Call dispute function of parent contract.
Courier Service Contract
Transfer funds to arbitrator.
Call dispute function of parent contract.
BT End of Chain Contract
Deliver item.
Dispute: Failure in key verification or delivery exceeded time.
Call Dispute function of parent contract.
Seller
Buyer
Transporter 1
Transporter 2
Transporter 3
1
2
3
Indicates unsuccessful verification of key and a dispute contract state
Propagate Dispute function call backward through the chain

FIGURE 6. **The process flow during dispute.**

Algorithm 5 Settling Payment

```
Input : authorized caller, caller, BTcontract state,
        parent contract, main Contract, transportation
        fees, item price, buyer, seller
1  if $BTcontract\ state = Successful\ Key\ Verification$ then
2      Settle payment starting from the main contract.
3      foreach $contract \in chain$ do
4          Set state of contract to Successful Key
           Verification.
5          if $contract\ is\ a\ BTcontract$ then
6              Transfer item price to buyer.
7              Transfer the contract balance to the seller.
8              Set contract state to Settle Payment Success.
9              Create a notification.
10         end
11         else
12             if $caller = authorized\ caller \wedge contract$
               $state = Successful\ Key\ Verification$ then
13                 $payment \leftarrow$
                   $(2 * item\ price) + transportationfees$
14                 Transfer $payment$ to contract's next
                   transporter.
15                 Transfer the contract balance to the
                   child contract.
16                 Create a notification.
17             end
18             else
19                 Revert contract state and show an error.
20             end
21         end
22     end
23 end
```

is about to take place. The result of the key verification can result to a case of dispute if the computed *keccak*256 hash of the key entered does not match the key hash already available in the contract. Therefore, a dispute can happen between any two transporters or between a transporter and a buyer. Furthermore, if a dispute happens due to key verification failure, it is difficult to know whose fault exactly it was. As a result, all the Ether deposited in the chain should be transferred to the arbitrator. The arbitrator would then off chain examine the logs and the situation well and distribute the Ether accordingly.

Figure 6 shows the flow of function calls when there is a dispute at the end of chain, in the BT contract. In the BT contract, a dispute can also occur if the item has exceeded the delivery time and the buyer refuses to receive the item when knowing that the item is on its way. The reason for exceeding the delivery time cannot be determined on the chain. Hence, the dispute procedure, Algorithm 6 is followed and the arbitrator would then solve the issue off the chain. Figure 6 shows the calls in order as the handling of the dispute takes place through the chain. As can be seen, the calls propagate backwards until reaching the main PoD contract. In each contract, the state is changed to *Dispute* and then the Ether held by that contract is transferred to the arbitrator. Algorithm 6 illustrates the algorithm in details. The input for the algorithm include the *authorizedcaller* which in this case is the buyer, or child contract or the next transporter.

V. TESTING AND VALIDATION

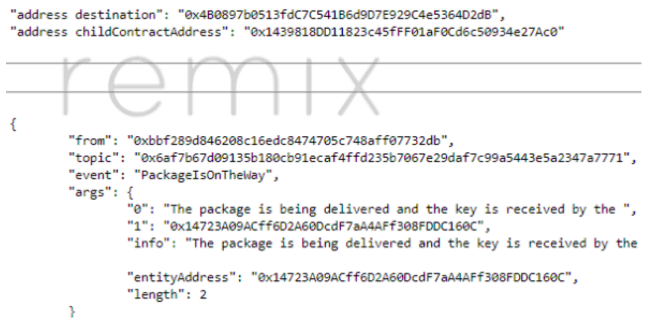
This section describes the details of testing the smart contracts' code. Moreover, it also provides a discussion on the gas cost analysis of the code as well as its security analysis. Two main functionalities are tested which are the dispute handling at the BT contract and payment settlement after a successful verification. A successful verification in the BT end of chain contract indicates a successful key verification across the chain. Also, the testing is done for one transporter and for multiple transporters.

1) CHILD CONTRACT ADDRESS ADDITION

All parent contracts have the addresses of their child contracts. The PoD main contract possesses the child contract address just like all other contracts across the chain and the

Algorithm 6 Handling Dispute

```
   Input : authorized caller, caller, contract state, parent
           contract, main PoD Contract
1  foreach parent contract ∈ chain do
2      if caller = authorized caller ∧ contract state =
       Dispute Verification Failure then
3          x ← DepositsofCurrentContract
4          if x > 0 then
5              Transfer Ether to the arbitrator.
6          end
7          if parent contract = main PoD contract then
8              Set state of the main PoD contract to
               Dispute Verification Failure.
9              Transfer Ethers of the main PoD contract to
               the arbitrator.
10             Create a notification that states all Ether has
               been deposited to the arbitrator.
11         end
12         else
13             Create an instance of courier service using
               the parent contract CS address.
14             Set state of the parent contract to Dispute
               Verification Failure.
15             Transfer Ethers of the parent contract to the
               arbitrator.
16             Create a notification which states that Ether
               has been deposited to the arbitrator.
17         end
18     end
19     else
20         Revert contract state and show an error.
21     end
22 end
```



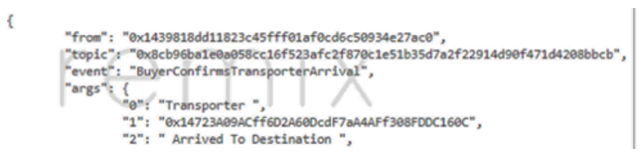
```
"address destination": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
"address childContractAddress": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0"
```

```
{
        "from": "0xbbf289d846208c16edc8474705c748aff07732db",
        "topic": "0x6af7b67d09135b180cb91ecaf4ffd235b7067e29daf7c99a5443e5a2347a7771",
        "event": "PackageIsOnTheWay",
        "args": {
                "0": "The package is being delivered and the key is received by the ",
                "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                "info": "The package is being delivered and the key is received by the

                "entityAddress": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                "length": 2
        }
```

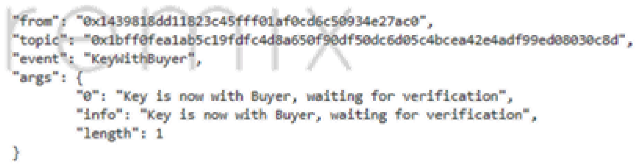
FIGURE 7. Logs showing the setting of child contract address in PoD contract by the transporter.

BT end of chain address. Figure 7 shows the child contract address added successfully to the PoD contract. This step is executed after the creation of the child contract. The transporter sets the child contract address in the parent contract, in this case the PoD contract and an event that says the package is on the way is created in the logs.



```
{
        "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
        "topic": "0x8cb96ba1e0a058cc16f523afc2f870c1e51b35d7a2f22914d90f471d4208bbcb",
        "event": "BuyerConfirmsTransporterArrival",
        "args": {
                "0": "Transporter ",
                "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                "2": " Arrived To Destination ",
```

FIGURE 8. Logs showing buyer confirming transporter arrival.



```
 "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
 "topic": "0x1bff0fea1ab5c19fdfc4d8a650f90df50dc6d05c4bcea42e4adf99ed08030c8d",
 "event": "KeyWithBuyer",
 "args": {
         "0": "Key is now with Buyer, waiting for verification",
         "info": "Key is now with Buyer, waiting for verification",
         "length": 1
 }
```

FIGURE 9. Logs of transporter arrival confirmation and key handing.

2) BUYER-TRANSPORTER HANDSHAKE

The buyer has to announce that the transporter has arrived so the transporter can confirm the arrival and hand in the item and key. This handshake is tested successfully. Figure 8 shows the successful execution of the function "confirmTransporterArrival" executed by the buyer. This changes the state of the contract and the transporter can only confirm the arrival after the buyer has done so. Figure 9 shows the logs after the transporter confirms the arrival. The logs indicate that the key is now with the buyer for verification.

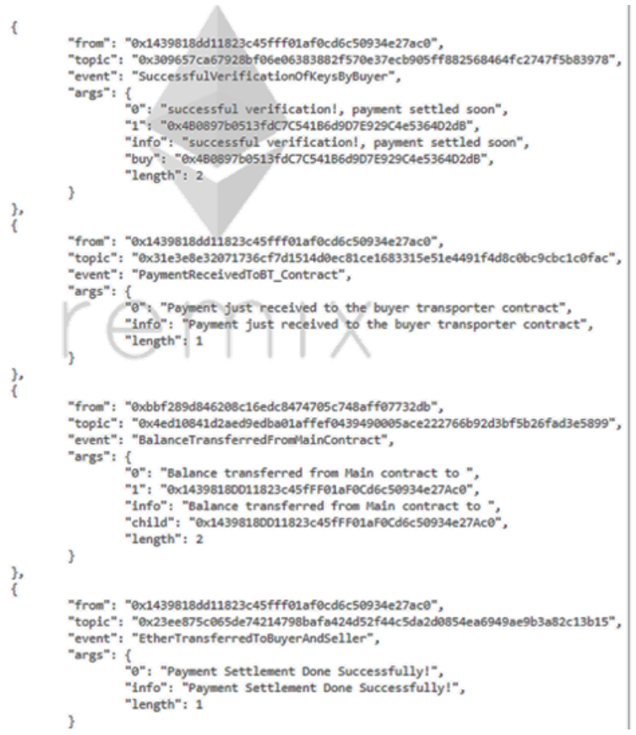
3) SUCCESSFUL KEY VERIFICATION

A successful key verification leads to a payment settlement. The logs in Figure 10 show the events after a successful key verification in a chain of two contracts with only one transporter. As can be seen, a successful key verification, automatically calls the settle payment function in the BT contract, which calls the function of the PoD contract. The transporter is then paid from the PoD contract and then the balance is transferred to the BT contract where the buyer and seller are given their deposits and payment.

Furthermore, a successful key verification in a chain of three transporters, thus, one PoD contract, one BT contract and two CS contracts leads to a payment settlement as shown in Figure 11. Each participant at the beginning had 100 Ether in their account. Three transporters at the end received 0.2 Ether as transportation fees along with their collateral. Thus ended up having almost 102 Ether. The buyer received the collateral back and paid 2 Ether for the item, hence the buyer has almost 98 Ether in the account at the end. Finally, the seller receives back the balance which is almost 1.4 Ether. Therefore, the seller at the end has almost 101.4 Ether as a result of the addition of 2 Ether from the buyer and paying the transporters 0.6 Ether.

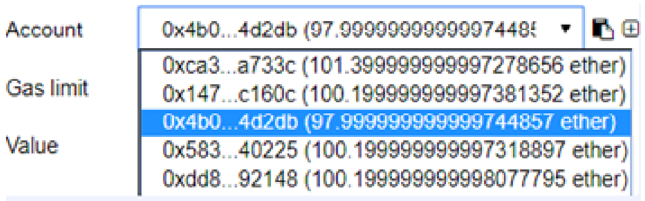
4) UNSUCCESSFUL KEY VERIFICATION

After an item is handed over, the recipient enters the key, which is hashed and compared with the hash already available in the contract. A mismatch in the hashes causes an



```
{
        "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
        "topic": "0x309657ca67928bf06e06383882f570e37ecb905ff882568464fc2747f5b83978",
        "event": "SuccessfulVerificationOfKeysByBuyer",
        "args": {
                "0": "successful verification!, payment settled soon",
                "1": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                "info": "successful verification!, payment settled soon",
                "buy": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                "length": 2
        }
},
{
        "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
        "topic": "0x31e3e8e32071736cf7d1514d0ec81ce1683315e51e4491f4d8c0bc9cbc1c0fac",
        "event": "PaymentReceivedToBT_Contract",
        "args": {
                "0": "Payment just received to the buyer transporter contract",
                "info": "Payment just received to the buyer transporter contract",
                "length": 1
        }
},
{
        "from": "0xbbf289d846208c16edc8474705c748aff07732db",
        "topic": "0x4ed10841d2aed9edba01affef0439490005ace222766b92d3bf5b26fad3e5899",
        "event": "BalanceTransferredFromMainContract",
        "args": {
                "0": "Balance transferred from Main contract to ",
                "1": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0",
                "info": "Balance transferred from Main contract to ",
                "child": "0x1439818DD11823c45fFF01aF0Cd6c50934e27Ac0",
                "length": 2
        }
},
{
        "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
        "topic": "0x23ee875c065de74214798bafa424d52f44c5da2d0854ea6949ae9b3a82c13b15",
        "event": "EtherTransferredToBuyerAndSeller",
        "args": {
                "0": "Payment Settlement Done Successfully!",
                "info": "Payment Settlement Done Successfully!",
                "length": 1
        }
```

FIGURE 10. Logs after successful key verification of one transporter.



```
Account     0x4b0...4d2db (97.9999999999997448!  ▾
Gas limit   0xca3...a733c (101.399999999997278656 ether)
            0x147...c160c (100.199999999997381352 ether)
Value       0x4b0...4d2db (97.99999999999744857 ether)
            0x583...40225 (100.199999999997318897 ether)
            0xdd8...92148 (100.199999999998077795 ether)
```

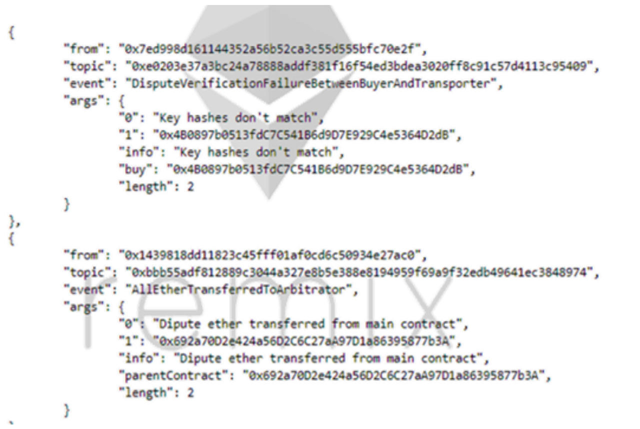
FIGURE 11. Ether balance of participants after successful key verification.

unsuccessful verification which leads to dispute. In order to handle the dispute, all the Ether is transferred to the arbitrator. The flow propagates backwards through the chain. In Figure 12, part of the logs of the handled dispute that took place in the BT end of chain contract can be seen. The events indicate a mismatch in the key hashes and a transfer of the funds to the arbitrator.

The same procedure applies in the case of multiple transporters. Figure 13 illustrates the changes that take place after a dispute is handled. In this case, there were two transporters and the item price was 2 Ether. Therefore, all the participants except the arbitrator have almost 96 Ether in their balance. This is because the collateral deducted is twice the item price. When the dispute occurred, the arbitrator who had initially 100 Ether like all the other participants received the funds from all the contracts and thus ended up with almost 116 Ether.

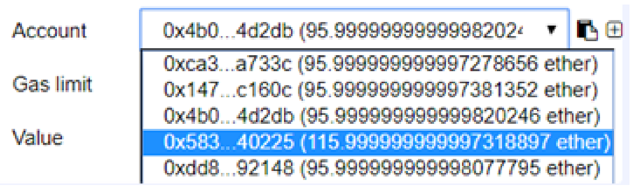
A. COST ANALYSIS

Gas is used in the Ethereum blockchain as a measure of the computational work a transaction needs.



```
{
        "from": "0x7ed998d161144352a56b52ca3c55d555bfc70e2f",
        "topic": "0xe0203e37a3bc24a78888addf381f16f54ed3bdea3020ff8c91c57d4113c95409",
        "event": "DisputeVerificationFailureBetweenBuyerAndTransporter",
        "args": {
                "0": "Key hashes don't match",
                "1": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                "info": "Key hashes don't match",
                "buy": "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                "length": 2
        }
},
{
        "from": "0x1439818dd11823c45fff01af0cd6c50934e27ac0",
        "topic": "0xbbb55adf812889c3044a327e8b5e388e8194959f69a9f32edb49641ec3848974",
        "event": "AllEtherTransferredToArbitrator",
        "args": {
                "0": "Dipute ether transferred from main contract",
                "1": "0x692a70D2e424a56D2C6C27aA97D1a86395877b3A",
                "info": "Dipute ether transferred from main contract",
                "parentContract": "0x692a70D2e424a56D2C6C27aA97D1a86395877b3A",
                "length": 2
        }
```

FIGURE 12. Logs of handled dispute in the BT contract.



```
Account     0x4b0...4d2db (95.9999999999998202‹  ▾
Gas limit   0xca3...a733c (95.999999999997278656 ether)
            0x147...c160c (95.999999999997381352 ether)
Value       0x4b0...4d2db (95.999999999999820246 ether)
            0x583...40225 (115.999999999997318897 ether)
            0xdd8...92148 (95.999999999998077795 ether)
```

FIGURE 13. Ether balance of participants after dispute handling.

Different transactions require different amounts of gas and the transaction fee is calculated as gas but the actual cost is paid in Ether. Hence, the 'gas cost' is how much gas is required for a transaction and the 'gas price' is the unit price of the gas in Ether. Moreover, the limit is set for each transaction to avoid running out of gas if there are any bugs in the code. Therefore, the gas limit provides a safety mechanism. It is also possible to speed up the transaction depending on the amount of Ether spent per unit of gas "Gwei". The higher the amount of Gwei, the higher the priority so it is a trade off between priority and cost. Miners would be more incentivized to execute the transactions that offer a higher gas price. Choosing the right gas price is a bit difficult as it requires continuous monitoring of the network. However, the ETH Gas Station made this easier by providing three different categories on their website [15].

- **SafeLow:** This is a gas price that can be chosen to go for a cheap and safe alternative. It is affordable, at the same time the transaction would be mined promptly. This price is determined after at least 50 transactions being executed at this price in the last 24 hours.
- **Average:** This gas price is accepted by the top miners and is usually quite close to the default wallet price.
- **Fast:** This price is the least price required to be chosen in order to be favored by the all the top miners. Choosing a price higher than this price will most probably not yield to a better speed.

In the cost analysis used a gas price of 4 Gwei was used which is the current Average price based on the ETH Gas Station [15]. A higher gas price means the cost of the transaction is higher as well. Table 1 shows the gas cost in

TABLE 1. Successful verification and payment settlement costs.

Number of Transporters	Transaction Gas	Execution Gas	Total Gas
1	53171	30747	83918
2	94998	72830	167828
3	101590	79422	181012

TABLE 2. Failure in verification and dispute handling costs.

Number of Transporters	Transaction Gas	Execution Gas	Total Gas
1	91045	68941	159986
2	122500	100396	222896
3	153719	131615	285334

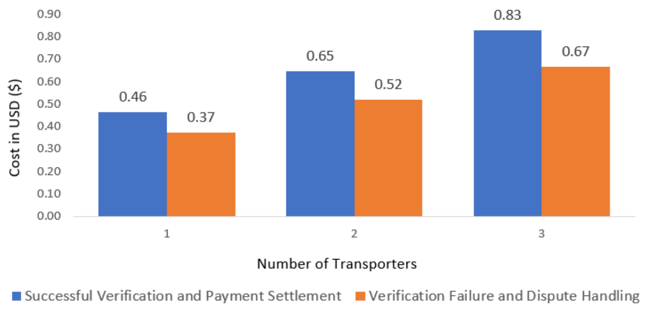


FIGURE 14. The cost in USD for the successful and unsuccessful verification in the BT contract.

Gwei for the execution of a successful key verification in the BT end of chain contract, which yields to a payment settlement. The gas cost is compared for different number of transporters. As the number of transporters increase, the gas cost increases. Regardless of the number of transporters, any chain has one PoD contract and one BT contract. However, what differs and creates the difference in the cost is the number of CS contracts, which is always one less than the number of transporters. Hence, one transporter means the chain has only two contracts and no CS contracts. In addition, two and three transporters, result in one and two CS contracts respectively. Hence, the increase in gas consumption.

On the other hand, Table 2 shows the gas cost of a failure in key verification in the end of chain BT contract, which yields to a dispute and transfer of funds to the arbitrator. Just like in a successful key verification, an increase in the number of transporters results in an increase in the gas. However, as can be seen in Figure 14 the cost for the payment settlement is more than the cost of dispute by an amount less than $0.2 This increase is justified because during payment settlement a call is first made to the PoD contract, then the call is propagated forwards through the chain. However, during dispute the calls are propagated backwards directly from the BT end of chain contract.

B. SECURITY ANALYSIS

In this subsection, we describe potential threats and attacks on the overall system, and discuss handling techniques to ensure satisfying the security goals. By design, our solution inherits key security features of the blockchain. These features include decentralized trust, integrity, non-repudiation, and availability. Our system also handles authentication and access control via smart contract features by employing restrict modifiers which allow execution of functions by certain actors. Also by design, the system protects against **Man-In-The-Middle (MITM)** attacks as well as **replay attacks** as every message exchange is cryptographically signed and time stamped. Even if the attacker replaced the actor's EA with his EA and public key, the attacker will not be able to correctly sign it as the private key is only known by the legitimate actor. Hence, if an attacker fakes an IP address or an EA address of a legitimate actor, the transaction would be denied and the contract state would be reverted.

Integrity is an important feature that ensures no data modification can occur to vital information. The PoD system provides the ability to trace back events in history using logs. The immutability of blockchain ensures the integrity of all the exchanged messages between the participating parties as well as the created logs and generated events. **Non-repudiation** provides verification for the identity of the sender. The Ethereum address of the initiator for all function calls is recorded and is always part of the logs. All function calls are signed by the actors. This ensures that no actor can deny their actions.

As for ensuring **availability**, it is to be noted that our smart contracts that get deployed to the blockchain would always be available for the participating entities to execute their functions. This makes the system services always readily available for all users. Historical data including initiated transactions, logs, and events are also available to all actors. Also, the system is protected against **Denial of Service (DoS)** attacks as all transactions are recored and stored on the public Ethereum ledger in a decentralized and distributed manner, and not subject to failure, hacking, or compromise. The Ethereum public ledger is highly robust and resistant to DDoS attacks as it is distributed globally, and protected by the tens of thousands of mining nodes which host duplicated records and data with high integrity and consistency.

The **confidentiality** requirement can be achieved by using a private or permissioned blockchain, as that of multichain, hyperledger, or private Ethereum. In our system, and considering the fact that buyers and sellers are unknown, we used the public Ethereum blockchain in which transactions are transmitted and stored in the clear. For a private Ethereum, transactions can be encrypted and keys can be shared only by participating miner nodes and actors. It is worth noting that our Blockchain-based design relieves the use of the expensive Public Key Infrastructure (PKI) for key distribution. As explained in [16], unique 20-byte Ethereum Addresses (EA) can be assigned instantly to participants with almost zero collision probability-which is a powerful feature of blockchain. EA comes with asymmetric public key pairs, which can be subsequently used for encrypting all data and transactions.

VI. CONCLUSION

In this paper we have presented a blockchain-based solution for the Proof of Delivery (PoD) of traded physical assets. Our solution utilizes the features of the Ethereum blockchain to automate payment and provides tamper-proof logs and events for trusted tractability and transparency. The implemented solution works for multiple transporters as well as single transporter, with a penalty and incentivization mechanism to force all participants to behave honestly. The solution also eliminates the need of a trusted third party and utilizes the smart contracts as an escrow to automatically settle payments, even under dispute, and give each participants its agreed upon share once the item is successfully delivered to the buyer. Our decentralized PoD solution uses a chain of contracts, with no cyclic dependencies, to satisfy the need of delivering among multiple transporters. We tested key functionalities, and demonstrated the correct behavior and outcomes considering multiple test case scenarios. We demonstrated and discussed that our solution meets the requirements for a sound PoD system. Moreover, we analyzed the security of our solution and concluded that the solution is resilient to known security attacks and does satisfy cyber security features and objectives. Our cost analysis reveals that the overall system cost is minimal, and increases marginally in the range of \$0.1 - \$0.2 with the increase in the number of transporters who can be involved in global shipment. The cost of both transactions is proportional to the current value of the gas price used. However, the expected cost difference between a successful and unsuccessful transaction is expected to be minor. This demonstrates that our solution is cost efficient and can be adopted as a generic solution for the sales and trades of physical assets locally and globally. As for future work, we are currently working on a solution for PoD of the sale of digital assets (such as as those of online books, documents, photos, movies, music, etc.), to ensure decentralized, trusted and secure delivery and automated payment for all types of traded assets whether they are physical or digital.

REFERENCES

[1] *Consumers are Now Doing Most of Their Shopping Online*. Accessed: Jun. 13, 2018. [Online]. Available: http://fortune.com/2016/06/08/online-shopping-increases/

[2] *UPS Study: Purchases From Marketplaces Nearly Universal Retail now Global as E-Commerce Shoppers Cross Borders*. Accessed: Jun. 13, 2018. [Online]. Available: https://www.comscore.com/Insights/Press-Releases/2018/4/UPS-Study-Purchases-From-Marketplaces-Nearly-Universal-Retail-/Now-Global-As-E-Commerce-Shoppers-Cross-Borders

[3] *About Shipping Carrier Contacts*. Accessed: Jun. 13, 2018. [Online]. Available: https://www.amazon.com/gp/help/customer/display.html/ref=hp_ss_qs_v3_rt_ci?ie=UTF8&nodeId=201117350

[4] H. R. Hasan and K. Salah, "Blockchain-based solution for proof of delivery of physical assets," in *Proc. Int. Conf. Blockchain*, Seattle, WA, USA, 2018, pp. 139–152.

[5] K. Toyoda, P. T. Mathiopoulos, I. Sasase, and T. Ohtsuki, "A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain," *IEEE Access*, vol. 5, pp. 17465–17477, 2017.

[6] K. Biswas and V. Muthukkumarasamy, "Securing smart cities using blockchain technology," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun.; IEEE 14th Int. Conf. Smart City; IEEE 2nd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2016, pp. 1392–1393.

[7] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *Proc. 2nd Int. Conf. Contemp. Comput. Inform. (IC3I)*, Dec. 2016, pp. 463–467.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[9] *Two Party Contracts*. Accessed: Jun. 13, 2018. [Online]. Available: https://dappsforbeginners.wordpress.com/tutorials/two-party-contracts/

[10] *How Our Escrow Smart Contract Works*. Accessed: Jun. 14, 2018. [Online]. Available: https://blog.localethereum.com/how-our-escrow-smart-contract-works/

[11] *Truly Decentralized, Peer-to-Peer Ecommerce Features*. Accessed: Jun. 14, 2018. [Online]. Available: https://www.openbazaar.org/features/

[12] *Soma the Social Market Place*. Accessed: Jun. 14, 2018. [Online]. Available: https://soma.co/documents/

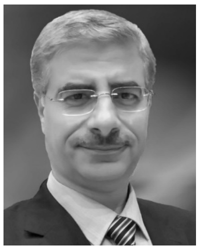
[13] *Syscoin 3.0: A Peer-to-Peer Electronic Cash System Built for Business Applications*. Accessed: Jun. 14, 2018. [Online]. Available: https://syscoin.org/Syscoin_3.0_Whitepaper___Condensed.pdf

[14] *Double Deposit Escrow—Bitbay*. Accessed: Jun. 14, 2018. [Online]. Available: https://bitbay.market/double-deposit-escrow/

[15] *Eth Gas Station*. Accessed: Jun. 14, 2018. [Online]. Available: https://ethgasstation.info/

[16] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17315765

HAYA R. HASAN received the B.S. degree in computer engineering from the American University of Sharjah, UAE, in 2014. She is currently pursuing the M.S. degree in computer engineering with Khalifa University, UAE. She is currently a Full Time Researcher and a Graduate Student with the Department of Electrical and Computer Engineering, Khalifa University. She is also a Research and Teaching Assistant with Khalifa University. She has authored or co-authored a number of conference and journal publications. Her research interests include cybersecurity, blockchain, Internet of Things, and secure mobile agents.



KHALED SALAH (SM'88) received the B.S. degree in computer engineering with a minor in computer science from Iowa State University, USA, in 1990, and the M.S. degree in computer systems engineering and the Ph.D. degree in computer science from the Illinois Institute of Technology, USA, in 1994 and 2000, respectively. He was with the Department of Information and Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, for 10 years. In 2010, he joined Khalifa University, UAE, where he is currently teaching graduate and undergraduate courses in the areas of cloud computing, computer and network security, computer networks, operating systems, and performance modeling and analysis. He is a Full Professor with the Department of Electrical and Computer Engineering, Khalifa University. He has authored or co-authored over 170 publications, holds three patents, and has given a number of international keynote speeches, invited talks, tutorials, and research seminars on the subjects of blockchain, Internet of Things, fog and cloud computing, and cybersecurity. He was a recipient of the Khalifa University Outstanding Research Award in 2014/2015, the KFUPM University Excellence in Research Award in 2008/2009, the KFUPM Best Research Project Award in 2009/2010, and also the departmental awards for Distinguished Research and Teaching in the prior years. He serves on the editorial boards for many WOS-listed journals including the *IET Communications*, the *IET Networks*, Elsevier's JNCA, Wiley's SCN, Wiley's IJNM, J.UCS, and AJSE. He serves as the Track Chair for the IEEE Globecom 2018 on Cloud Computing.