Lelantos: A Blockchain-based Anonymous Physical Delivery System

Riham AlTawy*, Muhammad ElSheikh†, Amr M. Youssef†, and Guang Gong*
*Electrical and Computer Engineering Department, University of Waterloo, Ontario, Canada.
†Concordia Institute for Information Systems Engineering, Concordia University, Québec, Canada.

***Abstract*—Real world physical shopping offers customers the privilege of maintaining their privacy by giving them the option of using cash, and thus providing no personal information such as their names and home addresses. On the contrary, electronic shopping mandates the use of all sorts of personally identifiable information for both billing and shipping purposes. Cryptocurrencies such as Bitcoin have created a stimulated growth in private billing by enabling pseudonymous payments. However, the anonymous delivery of the purchased physical goods is still an open research problem.**

In this work, we present a blockchain-based physical delivery system called *Lelantos*[1] that within a realistic threat model, offers customer anonymity, fair exchange and merchant-customer unlinkability. Our system is inspired by the onion routing techniques which are used to achieve anonymous message delivery. Additionally, Lelantos relies on the decentralization and pseudonymity of the blockchain to enable pseudonymity that is hard to compromise, and the distributed consensus mechanisms provided by smart contracts to enforce fair irrefutable transactions between distrustful contractual parties.

I. Introduction

Cryptocurrencies such as Bitcoin enable digital monetary transactions to be carried out without the presence of a trusted intermediary [1]. Additionally, transacting parties get to keep their real entity private by using pseudonyms [2] which are very hard to link to their actual identities. While the use of cryptocurrencies is very attractive for individuals who want to keep their anonymity, this attractive feature soon disappears when transactions involve physical goods, particularly, because purchasers are required to provide their address information for shipping the purchased goods. The issue of anonymous physical delivery has been an open problem for all the use cases of cyptocurrencies. Fortunately, the current rich and broad transactional platform offered by blockchains such as Ethereum [3] which support the distributed execution of smart contracts enables us to offer a solution to this problem.

Lelantos is a blockchain-based system that offers the service of anonymous delivery of physical goods. This system allows the customer, merchant, and a set of customer chosen delivery companies to engage in a delivery smart contract [4]. Such a contract acts as a trusted intermediary to enforce fair monetary transactions and to enable the contractual parties to communicate. Lelantos is composed of an onchain part which is the smart contract on the blockchain, and offchain components which are user-side applications and a web server. The core functionality of the system is executed in an onchain smart contract. The web server advertises and certifies all the delivery companies offering the service, and the user-side applications are used by different contractual parties to communicate with the smart contract. Our system also borrows concepts from Crowds [5] and onion routing techniques [6]. Specifically, in the route of the package delivery, every delivery company has knowledge of only two other locations which are the one before it and the one after it. Thus, linking a specific customer to an individual merchant is very hard unless all the contractual delivery companies were compromised or are collaborating with the merchant. Additionally and inspired by the delivery process in Crowds [5], the customer can dynamically decide, in real time, to pick the package up from any location on the chosen route, so none of the delivery companies knows a priori that its location is the chosen pickup one.

[1] *Lelantos* is the Titan god of the unseen in Greek mythology, and the name means to move unobserved.

A. Motivation

While Bitcoin at the first glance may seem to provide e-commerce anonymity, this anonymity completely vanishes if delivery of physical goods is required. Regardless of how the legitimacy of this degree of anonymity might be perceived by the public [7], [8], protecting one's personally identifiable information is essential in many cases. In fact, over the past two decades, there has been a rise in businesses offering privacy protection services in the real world. Specifically, one of the first such businesses that operated in the late 1990s is a company called iPrivacy LLC [9] whose services included obfuscation of one's street address that can only be translated by the delivery company software when it reaches the local postal area. Alternatively, and similar to our proposal, this company offered a service in which the recipient would go to a local delivery depot to pick the package up herself after being anonymously verified. Also, there are some patented protocols [10], [11] aiming to hide the identity of the customers from the merchants by utilizing a proxy service that makes the shipping information available only to the delivery company. However, these patents do not provide unlinkability because the delivery company knows both the customer and the merchant. Currently, there are commercial mail-receiving agencies such as *Private Box* in New Zealand [12] which offers the service of having an actual mailbox that do not carry one's home address. Examples of actual businesses that offer to receive packages, repackage them and then forward

them to addresses that are chosen by customers include *Mail ghost* [13] in the UK, and *Snail mail* [14] and *Rapid Remailer* [15] in the U.S. where one can even use their services to send a package as if it is originating from another address. It has been reported that *Snail mail* is used by Jimmy Carter for communication with other leaders and politicians to avoid the NSA surveillance programs [16]. Even Amazon provides the option of depo delivery in some parts of the U.S. via its locker service, where one may ship Amazon purchases to the locker address and then go pickup the package using a secret code after it is delivered. All these services offer a weak level of anonymity because linking senders to receivers or vice versa is simply accomplished by compromising the proxy service [17].

Strong anonymity is desirable for legitimate uses in many scenarios, some of which are listed below:

- Wrong profiling based on reading habits: It has been known that security agencies are conducting mass surveillance on the public in terms of their Internet browsing and telecommunication patterns. One can simply imagine a scenario where an academic conducting research on terrorism, war machinery, or any topic beyond the socially accepted norm, and thus buying specialized kind of books, to be wrongfully profiled and end up in some sort of a watch list. Profiling people based on the material they read has been used since World War I, where librarians were asked and sometimes volunteered to provide information about individuals who read communist material [18]. Currently, under the Patriot Act, the U.S. government, without a probable cause, can acquire library records through a secret court order. In fact, last year, the U.S. government issued a subpoena to Amazon.com to obtain the identities of customers purchasing books through the Amazon marketplace [19]. This subpoena was a step towards an attempt to profile the psychology of the U.S. citizens by data mining their book purchasing habits .
- Merchandise that makes an individual a target for burglary: People often worry about letting a stranger into their houses because then, they will be exposing their belongings and can possibly make them a valuable target for theft. The purchase of high value merchandise can trigger the same effect without the need to physically step into the house. One might imagine a scenario where an individual is buying an expensive safe from a compromised merchant or using a corrupted shipping agency to be highly susceptible to burglary which can further endanger the lives of his/her family.
- Sensitive purchases of important individuals: Top government and highly influential individuals, and celebrities are high value targets for snooping on their personal lives. The knowledge of some personal vulnerabilities maybe used for blackmailing or even causing them to lose their jobs. Consequently, if such an important individual is buying medications for HIV or any other medical condition that is crucial to be kept private, regular shipping or even weak anonymous services are not satisfactory.

Privacy is a fundamental right for all individuals and it is only them who decide what to share and what to keep secret. Following the techniques available for anonymizing one's electronic communication and transactions, we believe such techniques should be extended to our physical world, and our system is a step in that direction.

Our Contribution: In this paper, we propose Lelantos, a blockchain based anonymity-preserving physical delivery system which employs package routing through multiple delivery companies. Lelantos combines a blockchain smart contract interface to fairly and anonymously intermediate the delivery process without the need of a trusted third party, a web service to advertise and register delivery companies that offer the requested service, and contractual party-side applications to monitor the state of the smart contract and interact with it based on the role of the contractual party. We define the functionality of our system's smart contract and offchain components keeping in mind a lightweight implementation of the onchain operations to minimize the onchain code execution and thus *gas* expenditure [3]. Moreover, we analyze the security of the basic properties of the systems in terms of anonymity and unlinkability, fair exchange, and authorized pickup. As a proof of concept, we have implemented a working prototype of the Lelantos smart contract and it is available as an open-source project[2]. Our anonymous delivery system is built upon a realistic operational and threat model, and it offers the following features:

- Fair exchange: The package delivery is moderated by a decentralized smart contract which ensures fair transfer of funds to both merchants and delivery companies, and that the package is delivered to the intended customer.
- Customer anonymity: No private information related to the customer who is using a pseudonym is revealed to any of the contractual parties.
- Customer-merchant unlinkability: the scope of package routing knowledge of any contractual party except the customer is limited to a maximum of two hops.

II. Background

In this section, we give a brief overview on the technologies used to construct Lelantos.

Blockchain-enabled cryptocurrencies and smart contracts: Next generation blockchain-enabled cryptocurrencies such as Ethereum [3] builds on top of Bitcoin's blockchain technology a broad alternative platform which not only moderates monetary transactions but also extends to building decentralized applications. Ethereum implements a blockchain with a built-in Turing-complete programming language which enables writing smart contracts which are programs that autonomously execute the terms of an agreement. Smart contracts were first proposed in [4] as a way to make legal agreements fair and

[2]https://github.com/mhgharieb/Lelantos-Smart-Contract

precisely executed. On Ethereum, smart contracts are executed on the network nodes, also known as miners, and their results are enforced by a consensus protocol implemented by the netwrok [20]. These results are used to update the states of the contracts on the blockchain. Contract states are actual part of the blocks that are continually appended to the blockchain and entities can send or receive money and data to a contract. The open code nature and network consensus on the output of contracts execution enable smart contracts to build applications that allow mutually distrustful parties to transact safely without trusted third parties.

Entities on Ethereum transact using pseudonyms where each pseudonym is associated with a public key whose corresponding private key is owned by this entity. However, unlike public key infrastructure (PKI), the link between a specific public and secret key pair to a real world identity is not important unless a given entity willingly chooses to advertise its pseudonym (e.g., entities offering public services). The state of the Ethereum blockchain is made up of accounts, where each account has its own address, balance, storage, and code (if present). Accounts can either be *externally owned accounts*, also sometimes known as wallets, or contract accounts. The state of a wallet is controlled by its owner's private key which is used to transfer funds by digitally signing transactions to another wallet.

A contract account is controlled by its code, although it is incepted by an externally owned account. Once on the blockchain, a smart contract behaves autonomously and its code cannot be modified unless it is wiped by its owner using the *suicide* opcode. A contract implements one or more functions that represent its execution entry points which are determined by its creator. These functions accept messages as inputs in the form of function calls. Accordingly, a specific contract code executes in response to either receiving a message from another contract or a transaction from an externally owned account. Also, a contract has the ability to change the state of the ledger by enforcing monetary transactions in response to certain messages or transactions. Both terms "message" and "transaction" are sometimes used interchangeably, but a message usually refers to transactions generated by smart contracts, and a transaction refers to that originating from an externally owned account.

Smart contracts are executed on the Ethereum network nodes which exert computational power for running its code. Accordingly, to mitigate DoS attacks where an attacker can keep calling functions within contracts and thus crippling the network by aimlessly using its resources, Ethereum enforces the purchase of a resource called *gas* to power the execution of contracts or mining of transactions. So *gas* is considered the price one pays for mining nodes to execute code on the contract or to verify and commit a transaction on the blockchain. Every operation requires a fixed amount of *gas* units and accordingly transactions include a parameter "gaslimit" which is set by the sender to specify the maximum amount of *gas* units she is willing to spend so that her transaction and its subsequent computations are verified by the network nodes. The price of *gas* unit is determined in the transaction as well and it is expressed in Ethers (Ethereum's cryptocurrency unit). Nevertheless, a transaction goes through if the initiating account has enough funds to cover the price of the "gaslimit" and if miners accept the set price in exchange for their resources.

Onion routing: Onion routing [6] is a technique inspired by Chaum's MixNets [21] with the aim of building anonymous connections within a network of onion routers. In an onion network, a sender who wants to maintain her anonymity runs an onion client application to encapsulate her messages in layers of encryption, similar to the layers of an onion. The encrypted message is then forwarded to its destination through a series of network nodes called onion routers, each of which can only decrypt one layer of encryption thus, revealing the next destination to which the message is to be forwarded. After the last layer is decrypted, the message arrives to its intended recipient. For a given receiver, a message is received from the last onion router, and as long as the number of hops is greater or equal to two, the sender is anonymous because each onion router knows only the address of the immediately preceding and following nodes. While in onion routing, the nodes on the path are deterministically chosen, Crowds [5] aims to hide the identity of the sender by employing a real time randomly selected path of nodes where the message can be delivered to the recipient by any node on this path based on a coin flip. Accordingly, in our system we borrow ideas from both onion routing and Crowds to enable the real-time dynamic pickup of the package from any delivery company on a deterministically selected path.

III. System Architecture

The main functionality of Lelantos is implemented in an onchain smart contract L_{sc} that fairly intermediates the delivery process between an anonymous customer (C), a merchant (M), and a set of n delivery companies $(DC_1, DC_2, \cdots, DC_n)$ chosen by the customer. Our system also runs an offchain web server (L_{ws}) that advertises and certifies the public keys of the delivery companies and merchants that want to offer anonymous delivery and sale services, respectively. A delivery company operating under Lelantos is required to further run a courier side application App_{dc} the monitors the state of contract and interacts with it through messages from its wallet W_{dc}. A merchant operating under Lelantos is expected to advertise its wallet account W_m and run a merchant side application App_m to monitor the state of W_m, and when prompted with an order from L_{sc}, App_m also monitors the state of the smart contract. Finally, a customer who wants to use Lelantos to anonymize the delivery of the purchased goods is required to run a customer side application App_c to communicate with both the merchant and delivery companies through L_{sc}, and to direct and monitor the delivery progress of the package. The general architecture of Lelantos depicting how the onchain smart contract interacts with the rest of the system's components is shown in Figure 1. In what follows, we give a more detailed description of the

functionality of each component and how they interact with each other.

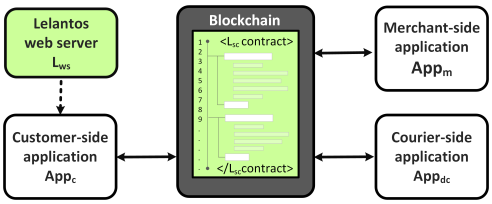

Fig. 1. Interaction between the onchain and offchain components of Lelantos. The dotted arrow denotes an optional interaction. Green components are trusted by the contractual parties

Lelantos smart contract (L_{sc})***:*** The onchain part of Lelantos that mediates the interaction between the contractual parties which are: a customer C, a merchant M, and a set of chosen delivery companies DCs. L_{sc} is designed with multiple functions that enable parties to relay messages to each other and to allow an anonymous C to monitor and direct the delivery route. Particularly, a customer first uses L_{sc} to place an order and relay the prepared encrypted addresses of the DCs on the route of her choice to M and other DCs who use them for package labeling. In other words, the merchant is required to place the first prepared ciphertext in a barcode readable format labels on the package so that first DC can read and decrypt its contents. Similarly, each DC is supposed to prepare a new lable for the following DC. The customer also includes an encrypted message for the merchant to privately know where to drop off the package. L_{sc} also implements functions to (i) Update the package tracking information, and (ii) Let the client decide whether to allow a given delivery company to forward the message to the next one, or to physically go and pick the package up. Finally, L_{sc} is responsible for fairly distributing the payments among the merchant and delivery companies.

Lelantos web server (L_{ws})***:*** The web server is an offchain part of Lelantos that advertises registered merchants and delivery companies. In other words, every merchant and delivery company offering anonymous delivery under Lelantos is required to register its locations and a long term public-key to be used by the customer in the preparation of the cipher layers. In this sense, one may think of L_{ws} as an entity that vouches for the advertised merchants and delivery companies to make sure that they are not frauds. However, L_{ws} has no means to know which set of delivery companies the customer already chose and once the cipher layers are prepared, L_{ws} cannot track a package to a specific destination. The only way that L_{ws} can monitor packages on a delivery route is if it advertises made-up delivery companies with public keys which L_{ws} owns the corresponding private keys, thus, it can decrypt all the leyered ciphers and know the final destination. We consider this scenario highly unrealistic because in this case, Lelantos has also to physically run different shipping locations and we assume that the advertised delivery companies are well known reputable ones. Nevertheless, a customer not wanting to trust L_{ws}, is free to pickup her own trusted merchant and delivery companies, and still use Lelantos's smart contract and applications to mediate the delivery process.

Customer-side application (App_c)***:*** This is one of the user-side offchain applications that can optionally connect anonymously (e.g. using Tor [22]) to L_{ws} to select the delivery companies on the chosen delivery route, and then App_c generates one encrypted message for each one using the corresponding public key. App_c also monitors the state of the smart contract and accordingly, based on the customers decisions forwards messages to L_{sc}. Particularly, acting as W_c, App_c initially creates the contract, places the order, then monitors the progress of the package delivery through its tracking information as the package moves from one delivery company to the other.

Merchant-side application (App_m)***:*** This application is run by a merchant and it monitors the associated wallet account W_m and delivery contracts that the merchant is engaged in. Once a merchant's W_m receives a transaction/message from L_{sc} indicating the requested product, App_m forwards the merchants response to L_{sc}, and monitors its state to get the uploaded ciphertext for labeling the package.

Courier-side application (App_{dc})***:*** This application is run by every delivery company to act as W_{dc} and forward messages containing tracking information to L_{sc} on the blockchain. Afterwards, App_{dc} monitors the state of L_{sc} to determine if it is going to forward the package to the next delivery company or if the package is going to be picked up from its current location.

IV. Operational and Adversarial Assumptions

The main objective of Lelantos is to mimic the regular shipping process in the real world while at the same time providing customer anonymity and fair exchange of funds for merchants and delivery companies. In our model, we do not consider external attacks where an individual or a GPS device can be used to track the package or the use of cameras at pickup locations which can be utilized to identify customers. Additionally, we do not consider the case of international shipping because it involves regulations that are beyond the scope of this paper. We assume that appropriate packaging of the goods is applied in order to obfuscate the contents. Also, we assume that our system has a large number of users and that delivery companies also offer regular shipping services, otherwise it is trivial to track packages if delivery companies are dedicated to this anonymous delivery business. Accordingly, we can reason about our contractual parties, system components, and operational environment using the following assumptions:

- ***A customer*** is interested in getting the purchased goods and maintaining her anonymity. However, due to the use of unadvertised pseudonyms, there is no actual customer accountability. Accordingly, a customer might act as an adversarial entity where she attempts to abort the protocol amid delivery, thus causing financial loss for other contractual parties.
- ***A merchant*** is keen to keep clients satisfied in order to

grow the business. Also, although the client is anonymous, the merchant is not. In fact, merchants are vouched for by Lelantos where it certifies their registered public keys. Accordingly they can be rated based on a reputation system by pseudonyms that have been in contracts with, thus can lose business with Lelantos in the case of repeated complaints. Nevertheless, it is reasonable to assume that a given merchant can be curious to know the identity of a given client and may collaborate with the drop off delivery company for information exchange.
- ***A delivery company*** is mostly interested in maximizing its profit and thus shipping the package is its first priority. Although, we require that delivery companies use different pseudonyms for different contracts, they remain identifiable to the customer who deals with them using their published public keys which are vouched for by Lelantos. Accordingly, we assume that there is some degree of accountability. We also assume that delivery companies do not depend on specific merchants because in our protocol, they are independently selected by the customer. Consequently, there is no obvious motivation for sharing information except for curiosity.
- ***Lelantos smart contract (L_{sc})*** is publicly exposed on the blockchain and its code is openly executed on the network, and hence, we assume that it behaves as expected.
- ***Offchain applications*** App_c, App_m, App_{dc} are assumed to relay authenticated publicly visible transactions/messages from the wallets of the contractual parties. Accordingly, we assume that the integrity of messages is protected by the digital signature of the originating wallet.
- ***The blockchain*** is trusted for correctness, availability, and integrity but not confidentiality as its state is publicly visible by everyone.
- ***Network communication attacks*** are assumed to affect the timely execution of the system but not the correctness. More specifically, an active adversary may tamper, drop, or reorder messages from different parties to the blockchain but cannot forge them.

A. Security Properties

The main aim of Lelantos is the protection of the privacy of the customer. Such a property is realized by the use of the blockchain which allows transactions using pseudonyms and the confusion generated by the multiple layers of encryption associated with package hops. Our system also guarantees fairness for other contractual parties when a customer is behaving in an adversarial manner. Also, the blockchain model [23] allows an entity to create an unrestricted number of pseudonyms when interacting with the onchain accounts. Accordingly, our system leverages all the features provided by the blockchain model in addition to the logic of our smart contract to provide the following properties

- Customer anonymity: The identity of a customer who engages in a delivery smart contract is guaranteed to be kept private. Moreover, all the information regarding the shipping route is also kept hidden.
- Fair exchange of services: Merchants and delivery companies are paid when the package is dropped off and validated by the chosen next destination and the customer is guaranteed to receive a package if all the employed delivery companies are paid.
- Protection against customer early protocol aborts: At any time during the package delivery process, parties who did their job are guaranteed to get paid even if the customer decides to abort the protocol.
- Authorized pickup: The package is delivered to the intended customer and no other entity can successfully claim it.

In what follows, we give a detailed description of our system, formal description of the flow of the messages message, and formal abstraction of the proposed smart contract L_{sc}.

V. Conventions and Protocol Description

We adopt the same notational conventions for writing contracts as described in [23], specifically, the following notation is used in our contract.

- A given entity can generate many pseudonyms by generating many public keys, where each pseudonym is the result of hashing the corresponding public key. In the contract description, we denote a given entity by X, where X is used as its corresponding pseudonym. The adopted blockchain model [23] provides a wrapper for smart contracts which handles pseudonym generation and the message signing for sending transactions so as to abstract all these details when writing the contract program.
- Transfer of cryptocurency takes place when operations involve $ledger[X]$ are invoked, where $ledger[X]$ denotes the balance of X in the global ledger. Variables that are preceded with the \$ sign denote a monetary value and do not affect a specific entity's balance unless an operation takes place on the $ledger$.
- Functions defined in the contract execute when they receive messages of a corresponding type. Generally, these functions may accept messages from any entity, but when a function is written as "upon receiving a message from party X", it is assumed that X is already added (known) in the contract. If the entity's pseudonym is preceded by the word "some" then this enables the addition of a new entity to the contract.
- A contract may have a $Timer$ function that is invoked at the beginning of each round. The blockchain timer advances in rounds whenever a new block is mined. The current time is encoded in the variable T.

This adopted blockchain model which is formalized in [23], does not only offer convenience when writing contract description but is also backed up by exact and formal definitions based upon the Universal Composability framework [24]. For more details on the blockchain and smart contracts formal modeling, the reader is referred to [23].

A. Lelantos protocol description

From a high level perspective, the Lelantos protocol description proceeds as follows. First, the customer C picks the desired product and records its identifier P_{id} from the merchant's online store. Then using her application App_c, the customer connects anonymously to Lelantos's web server L_{ws} and selects the merchant's identity and a set of $n \geq 2$ delivery companies ordered by a desired sequence of their locations. Unlike the conventional onion routing, we do not have a message to protect/hide, so we do not need the encapsulation. We only want to have the addresses of the chosen n delivery companies along with other information encrypted in a specific order. Accordingly, we form a set of ciphertext marked by their order. More specifically, App_c forms $n+1$ ciphertexts for all the n delivery companies and the merchant using their registered public keys. Each ciphertext for a delivery company contains the contract address, a tracking number, and the address of the next drop-off location masked by a unique masking value. The ciphertext for the merchant includes the address of the drop-off delivery company. Next, the client creates the contract by uploading all the hash commitment of all tracking numbers, the hash of secret to be verified on pickup and further sends the merchant a blockchain *order* message as an invitation to engage in the created delivery contract to acquire the first generated ciphertext and prepare the package label.

After the merchant drops off the package at DC_1, each delivery company starts to sequentially join the contract by uploading the tracking number and running App_{dc} to monitor its state. Now, once a contracts's state shows that a tracking number is uploaded, a user can either upload the next ciphertext for labeling and masking function so that the current delivery company can reveal the address where it must ship the package to, or she may go pick it up by letting the contract verify her pseudonymous identity through answering a committed challenge.

Figure 2 depicts our proposed contract for mediating the delivery protocol. The functions of the offchain applications (APP_c), (APP_m), and (APP_{dc}) which monitor the state and communicate with L_{sc} as W_c, W_m, and W_{dc}, respectively are explained in what follows.

Informally, the Lelantos smart contract is initialized by defining a hash function and the maximum fee a delivery company can charge. The contract steps and functions are described as follows:

- Contract creation: A customer C creates a delivery contract L_{sc} by invoking the `Create` function and uploading a commitment denoted by com, where $com = h(secret)$ where $secret$ is chosen by C and the customer further uploads the hash of all tracking numbers $\{h_i = h(tn_i)\}_{i=1}^{n}$ which she has selected for the delivery companies so that the contract can verify them when a new tracking number is uploaded. This step is essential to deter anyone from uploading a random tracking number and invoking the `receive` function which triggers the release of funds to a party that may had not done a drop off yet. Formally, for each delivery company DC_i in the ordered set of

```
Init: Set state := init
      Set fee_info := {}
      Set Commitment := ''
      Set tracking_comm := {}
      Set next_label := ''
      Def h(.) := SHA-3
      Set $max_hop_fee := $max_fee
      Set i := 1

Create: Upon receiving ("create", comm,{h(tn_i)}_{i=1}^n ) from some
customer C
      Set state := created
      Set commitment := comm
      Set tracking_comm := {h(tn_i)}_{i=1}^n
      Set $max_del_fee := n x $max_hop_fee

Order: Upon receiving ("order", M, p_id, $price, mc_0, c_1, T_end)
from C
      Assert state = created
      Assert ledger[C] >= $price + $max_del_fee
      Set ledger[C] := ledger[C] - ($price + $max_del_fee)
      Set next_label := c_1
      Send ("order", p_id, $price, next_label, mc_0) to M
      Set state := ordered

Accept: Upon receiving ("accept", _p_id) from M
      Assert state = ordered
      Assert p_id = _p_id
      Set state := accepted

Receive: Upon receiving ("receive", tn_i, $fee_i) from some
delivery company DC_i
      Assert state = accepted or next
      Assert h(tn_i) = tracking_comm[i]
      Assert $fee_i <= $max_hop_fee
      If (state = accepted)
         Set ledger[M] := ledger[M] + $price
      Else
         Set ledger[DC_i-1] = ledger[DC_i-1] + $fee_i-1
      Set fee_info[i] := (DC_i, $fee_i)
      Set state := received

Next: Upon receiving ("next", mask_i, c_i+1) from C
      Assert state = received
      Set next_label := c_i+1
      Set i := i + 1
      Set state := next

pickup: Upon receiving ("pickup", secret) from DC_i
      Assert state = received
      Assert Commitment = h(secret)
      Set ledger[DC_i] := ledger[DC_i] + $fee_i
      Set $diff := $max_del_fee - sum($fee_i)
      If ($diff > 0)
        Set ledger[C] := ledger[C] + $diff
      Set state := verified

Timer : If (state = ordered and T > T_end)
        Set ledger[C] := ledger[C] + ($price + $max_del_fee)
        Set state := aborted
```

Fig. 2. The pseudocode for the proposed smart contract L_{sc}

chosen n delivery companies where $i = 1, 2, \cdots, n$, the customer's application App_c generates the following ciphertext: $c_i = Enc_{pk_i}(tn_i, L_{sc}, add_{next_{dc}} \oplus mask_i)$, where pk_i is the long term public key of DC_i which is vouched for by Lelantos, tn_i is a random tracking number generated by C, L_{sc} denotes the address of the smart contract account, $add_{next_{dc}}$ denotes the address of DC_{i+1} on route, and $mask_i$ is a the i^{th} random masking value that conceals the value of $add_{next_{dc}}$ from DC_i until C decides either to invoke `Next` to upload $mask_i$ or go to DC_i where `Pickup` is invoked by App_{dc} to verify C (note that for the last delivery company DC_n, the $next_{dc}$ value is all zeros).Now that the contract gets n hashes, the `Create` function computes the maximum delivery fee $\$max\_del\_fee = n \times \$max\_hop\_fee$. The amount $\$max\_hop\_fee$ is set by Lelantos and advertised on L_{ws}, and thus it is agreed upon by all delivery companies offering the service. However, according to the size or weight of the package delivery companies may charge lesser fees and at the end, the contract returns the difference to the customer's balance again.

- Place an order: A customer C invokes the `Order` function by sending a message containing the pseudonym of the merchant M, the requested product identifier p_{id}, the product price $\$price$ which C is willing to pay for the product, the maximum allowed delay for a customer to wait for a response from the merchant to accept an order T_{end} after which the order is aborted, and the address of the first delivery company add_{DC_1} where the merchant should drop off the package at. However, the address is uploaded to the contract as mc_0 which is the result of encrypting the address probabilistically by the public key of the merchant pk_m so as to keep all the physical addresses private on the blockchain. C further sends the first encrypted label c_1 so M places it on the package to be decrypted by DC_1. After asserting that funds equal to $\$price + \$max\_del\_fee$ are transfered from $ledger[C]$ to the contracts balance to ensure fair payments to the contractual parties, the contract then sends an `order` message to M whose account state is monitored by App_m and accordingly, M can either ignore the order request or engage in the contract by accepting the order through invoking the `Accept` function.
- Accept an order: A merchant M willing to engage in L_{sc} invokes the `Accept` function by sending a message indicating the $\_p_{id}$. Now M decrypts mc_0 to reveal the address of DC_1 and places the cipher label c_1 on the package and drops it off at DC_1
- Receive a package: When M or DC_i drops off the package at DC_1 or DC_{i+1}, respectively, the respective c_i label is decrypted and the current DC_i uploads tn_i to the provided address L_{sc}. Now the contract verifies the uploaded tn_i against the one committed by C and only upon successful verification, L_{sc} transfers the owed fees to the balance of the entity which executed the drop off. In the meantime, App_{dc} keeps monitoring the state of L_{sc} waiting for either $mask_i$ to unmask the address of DC_{i+1} and the next label c_{i+1} so it can ship the package or the customer physically going to its location where she provides $secret$ to invoke the function `Pickup` which verifies her identity for pickup.
- Forward the package: A customer C invokes the function `Next` when she wants the package to be forwarded to the next delivery company in the sequence. `Next` is invoked by a message from App_m which using W_m, sends a message to L_{sc} containing $mask_i$ and c_{i+1} so that DC_i labels and ships the package to DC_{i+1}.
- Picking up the package: The function `Pickup` is invoked by a given delivery company to verify a customer who claims that she is the owner of a given package. The customer provides $secret$ which is uploaded by App_{dc} to L_{sc} through the message to the function `Pickup`. This function then verifies $secret$ against the commitment that was used during creation of the contract to verify the customer. A delivery company only dispenses the package to a customer when the state of L_{sc} changes to `verified`.

Figure 3 shows how the currency flows between the wallets of the contractual parties. In all the depicted currency transactions, we assume that gas fees are accounted for in the requested product price and delivery fees, and thus we omit the need to deal with them separately. However, all the code in the smart contract is executed by every miner in the network and hence to minimize the gas expenditure, we adopt a lightweight onchain code and delegate all the heavy execution involved with public key encryption to the offchain applications. Our protocol runs a web server where merchants

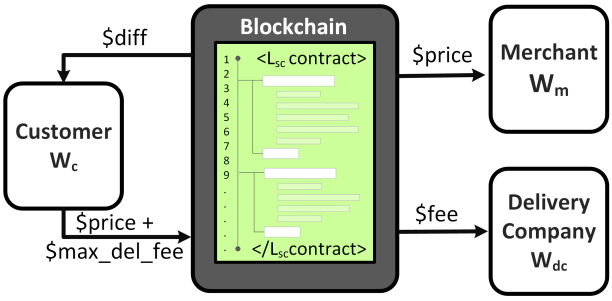

Fig. 3. Currency flow between the wallets of the contractual parties.

and delivery companies are registered and are required to engage in the smart contract with customers to mediate all the purchase and delivery processes. One can also think of an alternative procedure where customers create only purchase contracts with merchants and let the delivery companies handle package tracking and forwarding on their own websites (e.g. UPS and DHL). However, smart contracts lack network access outside of the blockchain so it cannot get tracking updates and thus the protocol will not work. A solution for this is the use of an authenticated data feed such as the recently proposed *Town Crier* [25] which uses Intel's SGX extensions [26] to relay authenticated https traffic to contracts in the blockchain via an onchain contract interface. Nevertheless, in

our protocol and for a given customer, the tracking numbers are already authenticated because they are transfered to the delivery companies encrypted with their public keys in a challenge response manner. Hence, if a tracking number is successfully uploaded to the contract, then it means that the intended delivery company received the package, if it is tampered with, it will simply get rejected by the contract because it will fail being verified against its committed value. Figure 4 depicts a general overview of the protocol flow. Note that Figure 4 shows messages that invoke respective functions in the contract after it has been created.

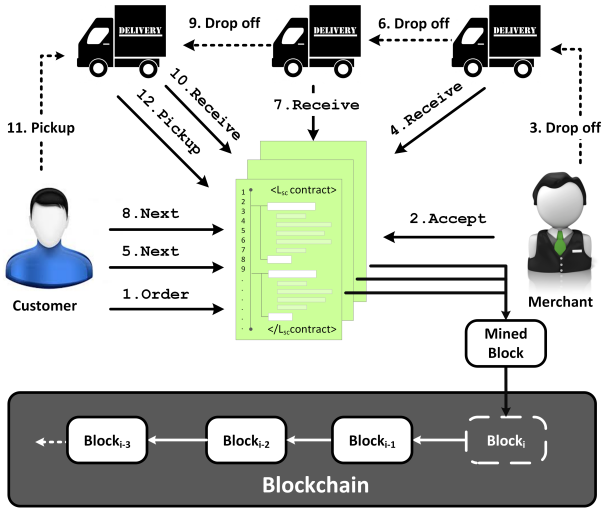

Fig. 4. An illustration of the Lelantos protocol steps. Messages and physical actions are numbered by the order in which they take place. Solid arrows denote onchain communications and dotted arrows denote physical actions such as drop off and pickup.

VI. Security Analysis

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

Anonymity claim: Intuitively, and given our threat model, in our protocol, anonymity means that an adversary including a corrupt merchant or delivery company cannot infer the real identity of the customer from monitoring the state of the contract. Assuming that a customer uses a different pseudonym for different deliveries and her transactions goes through multiple mixes [8] before cashing the funds, Lelantos provides anonymity. Formally, given a global passive adversary and the set of all blockchain pseudonyms Ψ, if the customer real identity u uses $C \in \Psi$ as a pseudonym for sending or receiving a message m, then we claim that $U_{(C,m)}(u) \approx 1/|\Psi|$, where $U_{(C,m)}(u)$ denotes the attacker's a posteriori probability of $C \in \Psi$ having the identity u with respect to message m.

Proof (sketch): According to the above claim, an a posteriori knowledge may be acquired by an adversary that is monitoring a specific actual identity using either a pre-known pseudonym or a pseudonym that is being associated with frequent change-transactions to a known pseudonym, or a customer who is known to pick a specific delivery route each time. Particularly, we refer to accounts that are advertised or used to deal with other real life entities that enforce the *Know Your Client* principal [8] as known pseudonyms and change-transactions are transactions that are used to send the change remaining from an actual transaction to another account associated with the sender. Subsequently, we argue that if the customer follows the operation assumption defined in the claim, then such a posteriori knowledge is infeasible to acquire. Also, delivery companies are required to generate different pseudonyms for new contracts. Accordingly, even if a given delivery company has a long term public key that identifies it on Lelantos's web server, its temporary blockchain entity cannot be linked to this public key except by the customer and the two delivery companies preceding and following it on the route. Subsequently, even if a specific real life identity is known to use the same delivery companies on a given route, such a route is freshly pseudonymized with each new contract.

Customer-merchant unlinkability claim: In our protocol, we informally describe unlinkability [17] as the inability of an adversary to trace a given customer back to the merchant or the other way around either by passively observing the state of the contract or by being a collaborating proper subset of the chosen set of delivery companies. More specifically, given the customer chosen set of n delivery companies, from the perspective of an adversary including at most $(n-1)$ corrupt delivery companies, the customer and merchant involved in the smart contract are no more or less related after observing the contract states than they are related given the regular a priori knowledge.

Proof (sketch): For an observing adversary, Lelantos lets the customer select the route where the package to be shipped through a set $\mathcal{N}$ of n delivery companies of her choice. For each delivery company, the customer prepares an encrypted message containing a tracking number of her choice and the address of the next delivery company masked by a number of her choice too. All the ciphertexts are sent encrypted to the contract, and tracking and masking numbers are meaningful only to the customer. Accordingly, all the information related to the package route, addresses of all delivery companies are confidential except for the customer and each pair of consecutive delivery companies. Therefore, if the set $\mathcal{K}$ of all delivery companies registered at Lelantos has k members, then an observer has a success probability of $1/\binom{k}{n}$ to link a merchant to a specific customer. For a set $\mathcal{S}$ of s corrupted collaborating delivery companies, the proof can be inferred from onion routing techniques where unlinkability is compromised when all the onion routers in the selected path are compromised. In our case, unlinkability is compromised *iff* the customer chose $\mathcal{N} \subseteq \mathcal{S}$ companies from the set $\mathcal{K}$ for her route. Assuming that $s << k$ because delivery companies are vouched for by Lelantos, the probability of getting a full compromised route $\approx (s/k)^n$. Accordingly, we can reason that getting a full compromised route has negligible probability.

Fair exchange of services claim: In our protocol, we define *fair exchange* [27] as the guarantee of getting paid the expected fee when the job is done correctly. We should note that *fair exchange* does not imply *commission fairness* [7]. In other words, our protocol guarantees that if all the delivery companies operate correctly, the customer receives a package for what she paid for. However, it does not guarantee that the contents of the package is what she is expecting. We reason about this using our realistic operational assumptions that both sets of merchants and delivery companies are not anonymous and that their best interest is to maximize their profit through maintaining their customers satisfied. We claim that Lelantos smart contract (L_{sc}) ensures fair exchange of services where given an adversarial customer, the merchant and the n chosen delivery companies are guaranteed to get paid what they ask for when a successful drop off is executed. Also, a customer is guaranteed to receive her package if all the delivery companies are paid

Proof (sketch): When a merchant accepts an order, the contract withdraws funds equal to $\$price + \$max\_del\_fee$, where $\$price$ denotes the price of the product as accepted by the merchant and $\$max\_del\_fee$ is the max delivery fee a given delivery company can charge multiplied by n. Indeed, upfront, the fees for the selected route are guaranteed and are out of the customers balance in the global ledger. Now, both the merchant and delivery companies get paid only when they successfully drop off the package at the following delivery company which decrypts its corresponding ciphertext and sends the tracking information to the smart contract. We assume that given that both the merchant and delivery companies are publicly known and registered in Lelantos's system, then they can exchange physical receipts between each drop-off, so if a delivery company did not upload its tracking info, the preceding entity can hold it accountable because of payment loss.

Authorized pickup claim: Our protocol ensures that the package is delivered to the customer C who invokes the `Create` function in the smart contract or someone that is delegated by C.

Proof (sketch): When a customer first creates the contract, she commits to a $secret$ that only she should know. This commitment is provided in the form of a 256-bit hash of $secret$ using SHA-3. At the pickup delivery company, the customer is asked to provide $secret$ which is uploaded to the contract and hashed for verification against the stored commitment. Accordingly, for an adversary who only knows the commitment to correctly claim and pickup the package, it has to successfully launch a preimage attack on SHA-3 which is completely infeasible. Formally, the adversary must find $secret$ or $x \neq secret$ such that $h(x) = h(secret) = commitment$, and the success probability of this preimage attack is $= 2^{-256}$.

VII. Implementation

As a proof of concept, we have implemented a working prototype of Lelantos smart contract which is available as an open source project[3]. The smart contract is implemented in 92 lines of Solidity. The full contract code is available in Appendix A. Note that because Ethereum does not support timer activated functions, we have implemented `Timer` as a withdraw function that is triggered by an explicit withdrawal request from the customer. Using this implementation, the gas cost estimates for deploying/running the Lelatos smart contract are provided using Remix Solidity IDE in Table I. These estimates are evaluated based on the following parameter byte sizes: tn_i = 4bytes, $next\_label$ = 96 bytes, $mask_i$ = 64 bytes, and $secret$ = 16 bytes. As of May 2017, 1 unit of gas = 18×10^{-9} ether, and 1 ether = 86.24 USD.

TABLE I
ESTIMATES FOR GAS COST IN USD ASSOCIATED WITH DEPLOYING AND RUNNING DIFFERENT FUNCTIONS OF THE LELANTOS CONTRACT‘ .

Function	Gas units	Gas cost (USD)
Deployment	1200583	1.86
Create	202808	0.31
Order	210102	0.33
Accept	68952	0.12
Receive	47138	0.07
Next	129350	0.20
PickUp	18617	0.02
Withdraw	10921	0.02

VIII. Related work

The idea of untraceable payments using cryptography was first introduced by Chaum [17] where digital blind signature schemes were used to achieve the desired level of anonymity. In early 2009, Bitcoin was proposed by Satoshi Nakamoto (a pseudonym used by the inventor(s) of Bitcoin) and presented the blockchain technology which offered a practical model for anonymous monetary transactions using public key and hashing primitives [1]. While initially gaining popularity among tech-savvies, Bitcoin technologies soon caught up the interest of academic researchers who were motivated to analyze various features of the blockchain. Specifically, investigating the blockchain anonymity emerged as an interesting area for research, where published works were either trying to undermine it or offer solutions to strengthen it. The work on the analysis of the Bitcoin blockchain anonymity includes the results presented by Ron and Shamir [29] where they analyzed the whole Bitcoin blockchain graph in an effort to group transactions which share certain patterns and assign them to specific entities. Ron and Shamir also showed that the FBI could not identify all the Bitcoins of Ross Ulbricht (the operator of the drug selling site Silk Road) [30]. Another work that confirmed the difficulty of de-anonymizing multiple mix transactions on the blockchain is presented in [8].

Our work leverages the pseudonymity and fair monetary exchange features offered by the blockchain [27] to present a system for the anonymous purchase and delivery of physical goods. In particular, we were inspired by two previous proposals that deal with the anonymous delivery of physical goods. The first proposal is presented by Androulaki and Bellovin [31], where they proposed a system that employs onion routing and blind group signatures to build a chain of blind tickets

[3]https://github.com/mhgharieb/Lelantos-Smart-Contract

that the customer initially gets form the merchant to use with the delivery company and then the system's central authority. This proposal differs from ours in that they employ a central authority where the chosen hops at the route are known by the central authority. However, the order in which they will be used or if all of them will be included in the route or not is not revealed to the central authority. Another difference is that the system in [31] does not ensure fair exchange for the delivery companies. In other words, even though the customer commits the total delivery fee upfront, payment distribution of these fees on the employed delivery companies is controlled by the customer. Whereas in our system, delivery fee payments are controlled by the smart contract which guarantees that each delivery company is fairly paid what it asked for. Furthermore, unlike our system, the pickup location is determined before the package is shipped from the merchant so the last delivery company always knows that the customer is picking up the package from its location. Also, the system in [31] offers tracking capabilities to the merchant as well as the customer which is accomplished by the presence of the central authority and the chain of trust due to the group signature scheme. The second system is proposed by Aïmeur *et al.* [32] where they propose an anonymous delivery system based on Chaum's MixNets [21]. Their system assumes that the package passes between a fixed ordered set of delivery companies on what they call a *horizontal rotary surface*. Each package on this rotary is examined by each delivery company it passes by, and according to the encapsulated layers of encryption, only those companies with the right public key are able to peel one encryption layer until the package reaches the pickup delivery company which removes it from the revolving rotary until the customer arrives for delivery. Similar to our proposal, the system's central authority does not know which delivery companies are chosen by the customer. However, the last delivery company always knows that it is the pickup point. Other than the presence of a central authority that manages the delivery process, the most important difference between Lelantos and both of the proposals in [31], [32] is that they do not deal with the anonymous payment, they only assume an anonymous payment of funds that cover the price of goods and the maximum delivery fee is made to the merchant who can be charged by the delivery central authority afterwards. On the other hand, our blockchain-based solution using smart contract integrates the whole anonymous purchase and delivery process in a decentralized, correct and always available environment and also, guarantees fair exchange.

IX. Conclusion

In this paper, we have proposed a blockchain-based system for the delivery of physical good. Our system employs techniques from Crowds and onion routing and further, leverages the decentralization, correctness, availability, and pseudonymity features of the smart contract enabled blockchain technology to offer anonymity, customer-merchant unlinkability, and fair exchange. Our system is composed of an onchain contract, a web server, and a set of offchain applications. In our proposal, we report the description of a smart contract that mediates the process of anonymously purchasing and delivering physical goods between a customer, a merchant, and a set of delivery companies. Moreover, we define the security properties of the system and reason about their proofs. We have also, implemented the Lelantos smart contract as an open source prototype and reported the estimated gas costs for its different functions. As compared to existing proposals, our protocol eliminates the need of a trusted third party and thus grants service availability, integrates the whole purchase and delivery process in one trusted smart contract which ensures fair exchange between contractual parties, and finally the offered anonymity is piggybacked from both the blockchain and onion routing protocols.

References

[1] S. Nakamoto, "Bitcoin: A peer to peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[2] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems," in *International Workshop on Selected Areas in Cryptography*. Springer, 1999, pp. 184–199.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014, http://gavwood.com/paper.pdf.

[4] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[5] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.

[6] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[7] A. Juels, A. Kosba, and E. Shi, "The ring of Gyges: Investigating the future of criminal smart contracts," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. ACM, 2016, pp. 283–295.

[8] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the Bitcoin ecosystem," in *APWG eCrime Researchers Summit*, 2013, pp. 1–14.

[9] B. Rosenberg, *Handbook of financial cryptography and security*, 1st ed. CRC Press, 2011.

[10] S. Stolfo, J. Smith, and J. Chung, "Method and system for private shipping to anonymous users of a computer network," 2001, uS Patent App. 09/754,897.

[11] R. Johnson, "eDropship: Methods and systems for anonymous ecommerce shipment," 2011, uS Patent App. 12/910,952.

[12] Privatebox, "Manage your PO box online and mail forwarding," https://www.privatebox.co.nz/.

[13] Mail ghost, "Anonymous mail forwarding," http://mail-ghost.com/.

[14] Snail mail, "Anonymous snail mail - real world anonymity," http://www.ultimate-anonymity.com/snail-mail.htm.

[15] Rapid remailer, "Anonymous letter and package remailing service," http://rapidremailer.com/.

[16] Reuters, "Jimmy Carter sticks to 'snail mail' in missives to world leaders," http://www.reuters.com/article/us-usa-security-carter-idUSBREA2M0QI20140323.

[17] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[18] P. Zwerling, *The CIA on campus: Essays on Academic Freedom and the National Security State*, 1st ed. McFarland & Company, 2011.

[19] Biggovernmentnews, "Big brother U.S. government subpoenaed Amazon.com to obtain book purchasing records of customers," 2015, http://www.biggovernment.news/2015-12-03-big-brother-u-s-government-subpoenaed-amazon-com-to-obtain-book-purchasing-records-of-customers.html.

[20] A. Miller and J. J. LaViola Jr, "Anonymous Byzantine consensus from moderately-hard puzzles: A model for Bitcoin," *http://nakamotoinstitute.org/research/anonymous-byzantine-consensus*, 2014.

[21] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[22] The Tor Project, Inc., "Tor project: Anonymity online," torproject.org https://www.torproject.org/F. [Online]. Available: https://www.torproject.org/
[23] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE Symposium on Security and Privacy*, 2016, pp. 839–858.
[24] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.
[25] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts."
[26] "Intel® Software Guard Extensions," 2017, Software.intel.com. [Online]. Available: https://software.intel.com/en-us/sgx
[27] I. Bentov and R. Kumaresan, "How to use Bitcoin to design fair protocols," in *International Cryptology Conference*. Springer, 2014, pp. 421–439.
[28] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 254–269. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978309
[29] D. Ron and A. Shamir, "Quantitative analysis of the full Bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.
[30] ——, "How did Dread Pirate Roberts acquire and protect his Bitcoin wealth?" in *Financial Cryptography and Data Security*. Springer, 2014, pp. 3–15.
[31] E. Androulaki and S. Bellovin, "APOD: Anonymous Physical Object Delivery," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2009, pp. 202–215.
[32] E. Aïmeur, G. Brassard, and F. S. M. Onana, "Secure anonymous physical delivery," *IADIS Int. J. WWW/Internet*, vol. 4, no. 1, pp. 55–69, 2006.

APPENDIX

```
pragma solidity ^0.4.0;
contract Lelatos {

//State Machine
enum States {Init, Created, Ordered, Accepted,
    Received, Next, Verified}

//Variables
States public state;
uint public maxHopFee;
address customer;
address merchant;
uint public pID;
uint public price;
bytes public nextLabel;
bytes32[] trackingComm;
bytes32 public commitment;
uint expirationTime;
uint maxDelFee;
uint8 index;
address currentHop;
uint currentHopFee;
bytes public mask;

//Checks as Modifiers
modifier checkState(States _state) {
if (state != _state) throw;
_;
}

modifier CheckCustomer() {
if (customer != msg.sender) throw;
_;
}

modifier CheckMerchant() {
if (merchant != msg.sender) throw;
_;
}

//Events
event NewOrder(address merchant, uint pID, uint
    price, bytes label0, bytes nextLabel);
event NewAccept(address merchant, uint pID);
event NewReceiver(address currentHop);
event NextHop(bytes mask, bytes nextLabel);
event Pickup(States state);

function Lelatos(uint _maxHopFee) {
maxHopFee = _maxHopFee;
state = States.Init;
index = 0;
}

function create(bytes32 _commitment, bytes32[]
    _trackingComm) checkState(States.Init) {
customer = msg.sender;
state = States.Created;
commitment = _commitment;
uint nHop = _trackingComm.length;
for (uint i; i < nHop; i++)
    trackingComm.push(_trackingComm[i]);
maxDelFee = nHop * maxHopFee;
}

function order(address _merchant, uint _pID, uint
    _price, bytes label0, bytes _nextLabel, uint
    validityTime) payable CheckCustomer
    checkState(States.Created) {
if (_price + maxDelFee > msg.value) throw;
merchant = _merchant;
pID = _pID;
price = _price;
nextLabel = _nextLabel;
expirationTime = now + validityTime;
state = States.Ordered;
NewOrder(merchant, pID, price, label0, nextLabel);
}

function accept(uint _pID) CheckMerchant
    checkState(States.Ordered) {
if (pID != _pID) throw;
currentHop = merchant;
currentHopFee = price;
state = States.Accepted;
NewAccept(merchant, pID);
}

function receive(bytes trackingNum, uint _fee) {
if (state != States.Accepted && state !=
    States.Next) throw;
if (trackingComm[index] != sha3(trackingNum)) throw;
if (_fee > maxHopFee) throw;
if (!currentHop.send(currentHopFee)) throw;
currentHop = msg.sender;
currentHopFee = _fee;
state = States.Received;
NewReceiver(currentHop);
}

function next(bytes _mask, bytes _nextLabel)
    CheckCustomer checkState(States.Received) {
index++;
mask = _mask;
nextLabel = _nextLabel;
state = States.Next;
NextHop(mask, nextLabel);
}

function pickup(bytes secret)
    checkState(States.Received) {
```

```
if (currentHop != msg.sender) throw;
if (commitment != sha3(secret)) throw;
if (!currentHop.send(currentHopFee)) throw;
state = States.Verified;
Pickup(state);
selfdestruct(customer);
}

function withdraw() CheckCustomer
    checkState(States.Ordered) {
if (now < expirationTime) throw;
selfdestruct(customer);
}
}
```