P2T: pay to transport

Fadi Barbàra[1] and Claudio Schifanella[1]

University of Turin, Italy
{fadi.barbara,claudio.schifanella}@unito.it

Abstract. We present Pay To Transport (P2T), a protocol that lets customers buy an item remotely in an atomic, privacy preserving and trustless manner. P2T needs only basic features of a blockchain scripting language and does not need any tracking systems, arbitrator or deposit to preserve its security properties. For this reason the protocol can be implemented on any permissionless blockchain, regardless of its scripting language, without additional trust. Merchants' and transporters' addresses are public, but in P2T the parties never pay those addresses directly. Therefore P2T maintains the privacy of customers, merchant and transporters.

Keywords:blockchain transportation p2sh bitcoin privacy

1 Introduction

As humans, we exchange value for goods since the specialization of labor. Value has been represented in many forms, for example using gold, money and even rocks. We started doing it physically in markets, and recently we moved to remote exchanges using online web-stores. Remote exchanges are undoubtedly useful, less physically hazardous and more convenient and efficient with respect to the previous in-person method, but the shift to remote exchanges had undesired consequences.

One of those consequences is the *de facto* loss of privacy and security. Today, services that centrally collect and store users' data have a far wider reach for sharing that data than they had before the internet. Third-party services can use data both for legitimate and non-legitimate purposes, and an aggressive sharing of data increases the probability of non-legitimate uses. Examples of non-legitimate uses are unfair prices or insurance premiums, stigmatization of people and, in the worst case, the unfair punishment of people in non-honest states [1]. Furthermore, any service that stores data centrally is potentially the target of malicious attacks. In this regard, then, private data is a liability for both the user giving it and the service collecting it. We claim that it would be better for all the parties involved not to have the data in the first place.

Nowadays, people can use blockchains to exchange value in a more (but not completely) private manner to defend themselves from some of these attacks, but there is not a private or anonymous method or protocol for the shipping of

some good from the merchant to the customer: generally the shipping still forces customers to share their personal data (e.g. their address or their identity) with the service they are buying from. The result is that people still have to trust services not sharing their data with other data collectors even though they use a blockchain based payment system.

Proposed methods to exchange goods using a blockchain for both the exchange of value and the shipment agreement still require private data sharing or additional trust. For example, some methods rely on tracking and the result of tracking is posted on a blockchain [2]. Generally, this involves external objects (typically a GPS) to signal the position of the package. In those settings, the GPS operates like some form of trusted oracle. This is both a trust and a privacy problem. In fact, both the company supplying the product and the client receiving it have to trust that nobody tampered with the GPS. Furthermore, a throwaway GPS sensor can be more expensive than the purchased item the transporter is carrying: this makes transportation costs higher than the item's cost. Therefore GPS-based tracking are not feasible for inexpensive items.

Our contribution To solve the current lack of privacy, security and trust in delivery systems, we present a protocol that doesn't require sharing private data but is still secure against non-honest participants. More specifically, our contributions are:

- We present Pay to Transport, denoted as P2T, a protocol that lets a merchant M and a customer C to remotely exchange value (*coins*) for goods using any permissionless blockchain;
- We analyze P2T and informally prove how the protocol respects the properties presented in Section 3, including privacy, atomicity and trustlessness, even without any arbitrator or deposit;
- We present a proof-of-concept implementation[1] which uses the Bitcoin blockchain.

The paper proceeds as follows. In Section 2 we present the literature on the topic of delivery transportation using a blockchain. In Section 3 we introduce the concepts needed to understand the protocol. In Section 4 we introduce the protocol using only one transporter. In Section 5 we analyze P2T and then we conclude.

2 Related Works

While there are multiple papers about the use of a blockchain system on a supply chain (see e.g. [3] for a survey), we decided to analyze only those papers that explicitly study the use of a transporter.

[1] See code at https://gitlab.com/disnocen/pay-to-transport

2.1 Proof of Delivery

In [4], Hasan et al. analyze what they call a Proof of Delivery system to trade and track sold items between two parties. The system relies on five agents. The first three agents are directly involved with the shipment of the item and they are the Seller, the Buyer and the Transporter. The others are external parties not directly involved in the exchange, they overview the process. Those are the Arbitrator and the Smart Contract Attestation Authority. The Arbitrator is a trusted third party involved in case of a dispute and solve the issues off the chain. The smart contract authority is responsible to attest that the smart contract complies with the terms and conditions signed by the involved parties in the agreement form. Each involved party puts an equal deposited collateral which he risks to lose if he behaves maliciously.

In the solution proposed by the authors the third parties (the arbitrator and the SC authority) are not prevented from colluding with one of the parties. In particular, the arbitrator handles the data off chain, so there is no transparent way to inspect his judgment. Furthermore, both the systems require that all parties, arbitrator and the smart contract authority included, know both the physical address and blockchain address of the buyer, so privacy is not guaranteed.

2.2 Lelantos

The solution proposed by Al Tawi et al. [5] also uses a smart contract deployed by the Lelantos system itself to manage the shipment. A single smart contract is used by all customers, merchants and couriers. A customer C is able to redirect shipment between different couriers by using a specific smart contract function. C sends new delivery addresses in encrypted form using the long term public key of the currently designated delivery courier. The public keys are vouchered by Lelantos itself.

The customer C does not declare in advance which couriers he will use. Furthermore, C will not contact any Currier before the shipment. While this process achieve anonymity for the customer C, the Lelantos protocol is interactive and requires both C and all the delivery couriers to pay attention to the delivery smart contract.

3 Preliminaries

Labeled Wallets and Derived Blockchain Addresses It is possible to create multiple public/secret key pair (and therefore addresses) starting from a single secret, called *base*. An example of this behavior is given, e.g. in the Bitcoin blockchain, from BIP32 address generation format [6]. The wallet generates new addresses starting from the base and a label. Therefore it is possible to index those addresses via the label.

In this paper we use the label format of [7]. Given a secret (private) key $\mathbf{sk}$ and a generation point g in an elliptic curve, the public key $\mathbf{pk}$ is computed as

$\mathbf{pk} = g^{\mathbf{sk}}$. The couple $(\mathbf{sk}, \mathbf{pk})$ is the base in our wallet. The derived key pair with label x from base $(\mathbf{sk}, \mathbf{pk})$ is written as $(\mathbf{sk}[x], \mathbf{pk}[x])$ an it is given by

$$\mathbf{sk}[x] := \mathbf{sk} + H_1(x) \quad \mathbf{pk}[x] := \mathbf{pk} + g^{H_1(x)}$$

where H_1 is (the numerical representation of) the hash function implemented in the blockchain. For example H_1 in Bitcoin is the SHA256, while H_1 in Tezos is Blake2b[2]. We call $\mathbf{sk}[x]$ the *derived secret key* and $\mathbf{pk}[x]$ the *derived public key*. Blockchain addresses are derived in a deterministic way using particular encoding $Enc(\cdot)$ of the hash of the public key. So if P is the address generated by the public key $\mathbf{pk}$, then the *derived address* is

$$d_{addr}(P, x) = Enc(H_2(\mathbf{pk}[x]))$$

where H_2 is generally different from H_1.

Conditional Transactions It is possible to make transactions that include conditional statements based on time (or block numbers): one branch of the transaction is used if it is redeemed before a certain time (or a certain block); the other branch is used if it is redeemed later. Throughout the paper we denote these as *conditional transactions*. It is possible to build conditional transactions for the majority of the blockchains, including Bitcoin and derivatives, Ethereum or Tezos. In the repository of this paper, we put a way to build conditional transactions in the case of Bitcoin transactions.

In P2T we use two kinds of conditional transaction constructions between non-trusting parties A and B: conditional transactions with secrets and without secrets. One way to build transactions that use secrets is through hash-lock contracts [8]. In short, given pre-computed secret s and hash $h = H(s)$, where $H(\cdot)$ is a hash function, the party who builds the transaction (say A) adds the the presentation of the preimage of h among the conditions to redeem that transaction. The other participant (say B) must then reveal the secret s (such that $h = H(s)$), in addition to putting his signature, to redeem the transaction. A hash-lock contract can be put in either (or both) of the branches of a conditional transaction.

More formally, let v be the value (sometimes called *amount*) of a transaction and let the string x be a particular encoding of the order placed by the customer C. Then, given $i = 1, 2$, s and h as above we denote with the ordered 5-tuple $(A, B, v, t, (h, i))$ the conditional transaction toward the address $d_{addr}(P_A, x)$ with secret s which is redeemable by:

1. a *multisig*[3] between parties (A, B), i.e. by using a joint signature with the keys related to addresses $d_{addr}(P_A, x)$ and $d_{addr}(P_B, x)$, before time t, or

[2] Although there are multiple hash functions implemented in Tezos, including SHA256, the Blake2b function is used for the most important cryptographic operations (such as signature checks). See the code at the URL https://gitlab.com/tezos/tezos/-/blob/master/src/lib_crypto/secp256k1.ml

[3] Even though in some blockchains such as Ethereum, there is no concept of multisignature, it is still possible to build smart contracts that have functions behav-

2. the signature of party A alone using the private key relative to address $d_{addr}(P_A, x)$ after time t .

If $i = 1$, then the hash-lock contract is put in the first branch of the transaction. Otherwise, if $i = 2$, the hash-lock contract is in the second branch. Finally, note that not all conditional transactions use secrets in P2T. In those cases we will write $(A, B, v, t, \texttt{null})$ or more simply (A, B, v, t)

Transportation Protocol Properties Using the terminology explained in Table 1.1 of [10] we want P2T to satisfy the following cryptographic properties:

- **Privacy**: keeping information secret from all but those parties that are authorized to see it
- **(Customer) Anonymity**: the customer is able to conceal his identity
- **Entity Authentication**: corroboration of the identity of an entity
- **Receipt**: acknowledgment that information has been received;
- **Confirmation**: acknowledgment that services have been provided
- **Plausible Deniability**: an external party can not prove that a customer C bought an item I from a merchant M and it has been delivered by a transporter T without the active collaboration of at least one of those parties.

Similarly to [4] we also focused on these logistic properties:

- **Punctuality**: every action and deliver has a maximum allowed time
- **Honesty**: following the protocol is the most rewarding behavior
- **Atomicity**: no party involved can lose money even though other parties misbehave

Finally we want P2T to satisfy this property:

- **Trustlessness**: it is not necessary for one participant to trust the others

In Section 5 we show how the P2T protocol satisfies all these requirements.

Notation Parties are addressed with their initial letter, e.g. the merchant is denoted by M. In the previous sections we already introduced the notation for keys, addresses and conditional transactions. We explain here the notation used for the time constraints.

Given an ordered couple of parties (P, Q) and a transporter T, we denote with δ_T^{PQ} (an estimate of) the time that T needs to go from the pick up point of P to the pick up point of Q. Note that in principle going from P's pick up point to Q's one can be different from going from Q's pick up point to P's. For this reason the couple (P, Q) is ordered and $\delta_T^{PQ} \neq \delta_T^{QP}$. Because the transporter has to wait at his pick up point for the customer or others to take the package, we

ing like a multisignature. See for example https://github.com/unchained-capital/ethereum-multisig. Furthermore, in other blockchains it is possible to create *aggregate signatures* that act as a multisignature but leaving only one signature as blockchain footprint, further increasing privacy. See for example [9]

put ϵ_T the maximum time that T can wait before he returns the package. We also put $\tilde{\delta}_T^{PQ} = \delta_T^{PQ} + \delta_T^{QP} + \epsilon_T$ as the whole time that T needs to go from P's pick up point to Q's and back plus the waiting time. Of course $\tilde{\delta}_T^{PQ} = \tilde{\delta}_T^{QP}$, but we will use both notations. We use $\tilde{\delta}_T^{PQ}$ when T goes from P to Q and then he goes back to P and vice versa. This lets us be more explicit in the description of all the time constrained payments needed to support the fact that C can refuse the delivered package in the end.

4 Transportation Protocol

The P2T protocol involves three parties: a merchant M, a customer C and a transporter T. Public keys $\mathbf{pk}_M$ and $\mathbf{pk}_T$ and blockchain addresses P_M and P_T of M and T respectively are public and known in advance to all the parties (e.g. those information are in the contact internet pages of the parties). Note that a protocol for the shipment of a product from M to C is different from a protocol for returning that product *after* the acceptance of the delivered package: in this paper we focus only on the shipment protocol. Therefore the shipped package can be accepted or refused by C on the spot only. Of course, it is possible to adapt this protocol in case of a product return, treating it as a shipment from C to M, but this is not discussed here.

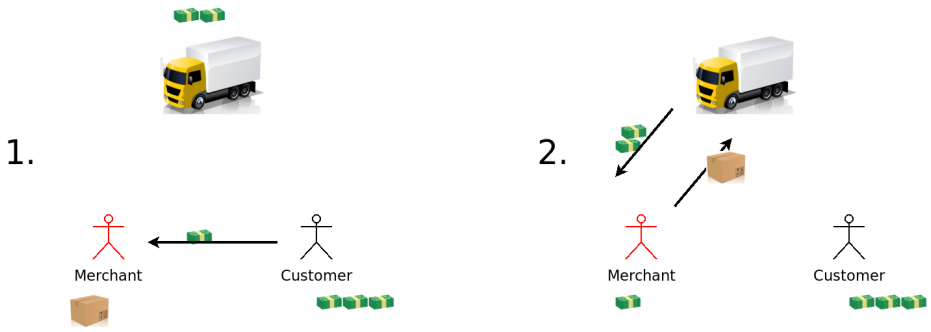

Fig. 1. Phases one and two of the Basic Protocol with One Transporter. **1.** C pays M the transportation costs, **2.** T physically goes to M, pays M and receive the package

Broadly speaking, the P2T protocol works as follows. The transporter T goes to merchant M, pays M the cost of the item (ad interim payment) and takes charge of the package. The transporter then brings it to his own pick up point (e.g. T's company headquarters). Finally, customer C goes to T's pick up point, pays T the cost of the item plus transportation costs and takes his package. C does not have to reveal his own physical address nor his identity to perform these actions.

Since a system of "simple" transactions (e.g. P2PKH in Bitcoin or transactions between Externally Owned Accounts in Ethereum) would not give sufficient

Algorithm 1 Basic protocol

1: C decides on I and M
2: C and M engage T, C accepts c_T
3: T generates r, T computes $R := H(r)$
4: T sends R to C
5: C pays c_T to M
6: CT1: C sends conditional transaction $(C, T, c_I, t_1 + \tilde{\delta}_T^{TM}, (R, 1))$
7: CT2: T sends conditional transaction $(T, M, c_I - c_T, t_1 + \delta_T^{TM}, \texttt{null})$
8: M gives package to T and *at the same time* CT3: T and M send conditional transaction $(M, T, c_I - c_T, t_1 + \tilde{\delta}_T^{TM} + \delta_T^{TM}, (R, 2))$
9: **if** C accepts the package **then**
10: C and T spend CT1 (so T releases r)
11: M spends CT3 using r
12: **else**
13: T brings package back to M
14: T and M send money from CT3 to T's address
15: **end if**

guarantees to any of the parties involved, we based every passage of coins and product on time constraints and spendability conditions (see Section 3), coded in the transactions. Building transactions this way, we accomplish two things. On the one hand we accomplish a traceable coordination of multiple parties without using external tracking devices. On the other hand, C doesn't need to be on line after the first payment to M and from that payment on the P2T protocol is non interactive from his point of view. This is a huge advantage for C since in this way he can use a device once (for example a public computer in a library) without the need to subsequently check the status of his order.

Fundamental steps of the P2T protocol are summarized in pseudo-code in Algorithm 1 and it works as follows. Customer C decides to buy an item I at time t_0 from the webstore of the merchant M, and he needs I to be shipped to a place which is more close to him. We assume that the cost for the item is c_I, and throughout the protocol the cost of the transportation is assumed to be c_T for each chosen transporter T. An order from C can have multiple information, such as the item's identification number, the item's quantity or the maximum date of delivery which we denote as t_2 (see below for a constraint on t_2). C is required also to provide a blockchain address P_C both as the (only) identification for that shipment and to prove he has enough coins to pay for the item I. On the other hand, C is *not* required to provide a delivery address nor any other identifying information. We emphasize that a blockchain address created specifically for this trade cannot be considered as an identifying element for the customer C. In fact, assuming that C is keen on maintaining its privacy, this address can be funded using the particular technologies of blockchain projects. Examples are the z-shielded transactions in Zcash, the use of a large number of *mixins* in Monero, CoinJoins or similar technologies in Bitcoin or, in general, the use of mixers. Furthermore, this address is used directly only once, so there is no risk of address reuse.

Based on the information provided by C, M and C agree on a transporter T[4]. During the agreement, C specifies what we call the *minimal required zone* (MRZ). A MRZ is the minimal information needed by T to estimate delivery costs and date of delivery. For example, if T charges the same delivery costs and estimates the same delivery time for a whole country, then C communicates only the country where he intends to pick up the item I. Therefore by the agreement T must take the item I from M and take it to his pick up place before date t_2. Here $t_1 + \tilde{\delta}_T^{TM} - \epsilon_T \leq t_2 \leq t_1 + \tilde{\delta}_T^{TM}$ where t_1 is the occurrence of the first payment from C to M (see below for details). Of course, if C and M agree on T, they also agree on the additional costs c_T. All parties M, C and T provide to each other some contact information for possible notifications, e.g. for the arrival of the package at T's pick up place[5]. As soon as T has been decided and engaged in transport, T generates a random number r and creates $R = H(r)$ where H is a hash function. T sends R to C using the contact information provided before. R is the puzzle for the secret, as described in Section 3.

After this step, M checks that there are at least $c_I + c_T$ funds in P_C[6] and if so M creates a bill contract x that he sends to C. In x there are some static information about M, i.e. information that persists for more than one order, and some dynamic information regarding the specific order. In particular, the address P_T of T is included in x. C verifies that the information on x are sound and if he agrees on them he sends the equivalent of c_T to address $d_{addr}(P_M, x)$. We call this transaction the *non-redeemable commitment transaction of* C and it is done at time t_1. This payment represents three things about C: it is a proof that C controls the funds in address P_C, it is a proof that C accepts all terms written in x and, being non redeemable by C, it is an incentive not to spam M with fake requests which would result in a DoS attack.

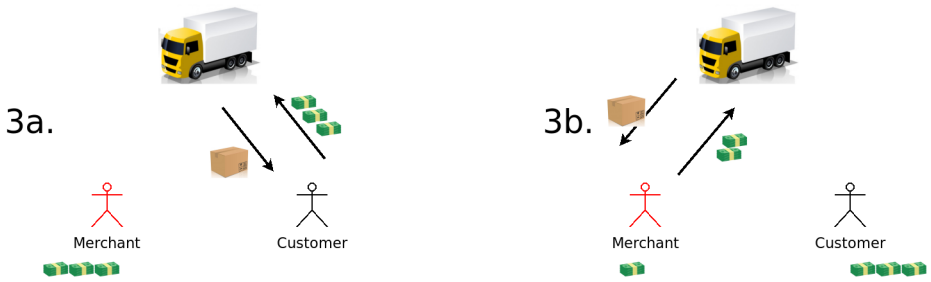

Fig. 2. Possible endings of the Basic Protocol with One Transporter. **3a.** C physically goes to T and if the package is intact, then C accepts it and pays T, **3b.** otherwise, if C refuses the package, T gives M the package back while M returns T his money

[4] T can be chosen from a billboard or by some convention between the merchant M and T himself

[5] The contact information should not reveal identity of C. For example C can use a throwaway e-mail address or a burner phone

[6] The merchant M can do that because the blockchain is public

When the merchant M receives the payment from customer C, he is sure that C has serious intentions in buying the object, but he does not know anything about T. For this reason M sends $H(x)$ to T waiting for his commitment transaction.

Before doing that, T needs a commitment from C, so C sends another commitment transaction, this time to T. This commitment transaction is different from the previous one because it is redeemable by C. This is a $(C, T, c_I, t_1 + \tilde{\delta}_T^{TM}, (R, 1))$ conditional transaction with secret R in the first branch of the transaction (see Section 3) and we call it CT1. C has to do this payment before time $t_1 + \delta_T^{TM}$, otherwise T cannot go to M in time and T risks to delay the whole shipment process. In case C is too slow to pay, T decides to abort the protocol and notifies other parties. In case T can commit to M, he sends $c_I - c_T$ coins to address $d_{addr}(P_T, x)$ doing a redeemable conditional commitment transaction without secret $(T, M, c_I - c_T, t_1 + \delta_T^{TM})$ which we call CT2. M considers valid T's coin transfer only if the transaction is built in the way described above, otherwise M aborts the protocol and notifies C of that.

After the merchant M saw the payment, he produces and physically prints a visual representation V (e.g. a QRcode) of $H(x)$ and use it to seal the package. At time $t_1 + \delta_T^{TM}$, T can take this package with item I inside it from M. If M is not malicious, the package of item I is in perfect conditions and T verifies that V is the visual representation of $H(x)$ (recall that T has received $H(x)$ before to create address $d_{addr}(P_T, x)$), both M and T sign the transaction from $d_{addr}(P_T, x)$ to $d_{addr}(P_M, x)$. This is a $(M, T, c_I - c_T, t_1 + \tilde{\delta}_T^{TM} + \delta_T^{TM}, (R, 2))$ conditional transaction CT3 with secret R in the second branch. At this stage, M has received (but cannot use yet) $c_T + (c_I - c_T) = c_I$ coins, so the merchant has received the full price of the item I and the item is shipped. The second conditional transaction with a secret is done to account for the case in which C could refuse the package.

T takes the package to his pick up point. When T and C physically meet at T's pick up point, C checks that the package is intact, that the seal V is not broken and that it represents $H(x)$. If that is case, then C and T spends their conditional transaction CT1 sending funds to an address belonging to T. This way T has to reveal r such that $R = H(r)$ and M can use it to spend CT3. On the other hand, if there is some problem with the package, C refuses the package and T has to bring it back to M. T is sure he can have his coins back because of the conditional transaction CT3.

5 Analysis

Privacy and Anonymity From C's point of view, P2T is highly private. In fact, the customer C provides to the merchant M and the transporter T only a public key with funds and a geographical zone (the MRZ) where he intends to pick the package. Depending on the blockchain method used, the source of funds can be obfuscated in a way to detach it from the real identity of C (see Section 3). Therefore P2T satisfy also customer anonymity. Note that privacy and

anonymity comes at a cost for C. M could steal funds of C and never give him any product, gaining c_T. While this is the case, we assume C won't use merchants or transporters that have any or a bad reputation for big payments. On the other hand, even though C loses funds, he only loses transportation costs. Still, this is better than today's policy for which C must pay the whole cost in advance, and therefore he risks losing both the cost of the item c_I and the delivery cost c_T.

Authentication and Deniability In the P2T protocol there is an intrinsic authentication method. In fact, the public addresses and keys of the merchant M and the transporters T are public and the payments are made to addresses deriving from those public keys using the homomorphic properties of the construction of the addresses. The fact that the entities are able to spend these funds in the derived addresses is proof of their identity. Furthermore, since all entities use derived addresses and not their publicly accessible addresses, an external observer cannot prove that the parties involved have completed a particular exchange thanks to the hardness of the discrete logarithm problem assumption[7]. Therefore all parties can plausibly deny their involvement in an order.

Confirmation and Receipt The use of the blockchain and the particular way parties following the protocol build the addresses and transactions gives both the receipt and confirmation of orders and payments. Furthermore, by following the state of the blockchain the entities involved can track the state of the order by seeing which payments have been already done.

Punctuality This also provide the punctuality property of P2T: time constraints in the transactions force parties to respect all prearranged times or they risk losing funds.

Atomicity and Honesty Transactions are constructed taking into account the possible dishonesty of each participant. Since every transaction is atomic, if a participant does not respect the protocol (that is, he is not honest), he does not receive the coins that would be due to him. Each participant is therefore encouraged to be honest. In other words, following the protocol is the most rewarding behavior.

Trustlessness In P2T, participants do not have to trust the others. This is due to the particular constructions of the transactions. We analyze the protocol from the point of view of each of the participants.

From the point of view of the merchant M, there is no way to lose both the money and the product. In fact, once the product has been given to the transporter T, the transaction CT3 assures the merchant that (if customer C accepts the package) he will be able to spend his coins. This is because M

[7] This is the underlining assumption for the construction of public keys on all blockchain projects.

supervises the creation of this transaction (M and T are in the same place at the same time) and can verify that the hash placed by T is the same as the one in transaction CT1. Furthermore M will know about the preimage of the hash the moment T redeems CT1. If, on the other hand, C does not accept the product, theoretically T may decide not to return the package to M. But this would not be a rational choice for T. The transporter, in fact, does not know what is contained in the package (therefore a priori may not be interested in the article) and he is therefore encouraged to return it in order to redeem the money stuck in the multisig with M.

As far as the transporter T is concerned, he is interested in not losing the money invested to earn the transport commissions. T cannot lose money in CT2 (its first transaction) since it is atomic and T only executes it after CT1 has been confirmed. T risks losing money in CT3 if C doesn't accept the package and M doesn't show up for the return. In this case the time-lock would expire and M could redeem the transaction. This is not possible because M must also solve the hash-lock contract, and to solve it M needs the preimage revealed by T. T reveals this secret only if C accepts the package. In this regard, note that if the secret had been created by C, T would not have had the same assurances. In fact, given the anonymity of C, he and M could be the same entity, or colluding. If that were the case, then C could refuse the package and M could still redeem the coins in CT3 because he is aware of the preimage.

Finally, C doesn't need to trust anyone too. Once the shipping costs have been paid (which C agrees to lose if the package is refused) C creates and sends the atomic transaction CT1 . On the scheduled date, C goes to the pick up point of T and decides whether to accept the package and sign the transaction with T or to refuse the package. In this latter case, C only has to wait for the time-lock to expire in order to use its coins again: in the meantime T cannot spend them because they are in a multisig.

6 Conclusion and Future Works

We present P2T, a payment protocol for the exchange of coins and physical goods. P2T is trustless, privacy preserving and preserves the anonymity of customers without using external tracking systems, arbitrators or deposits. The protocol uses mechanisms common to all blockchain protocols, so that it is possible to implement it in all these projects. We implemented a proof of concept that uses the Bitcoin blockchain and that can be found online. In addition to privacy and anonymity, the protocol satisfies other properties such as plausible deniability and encourages participants' honesty by using atomic transactions. In the future we intend to extend the treatment of the P2T protocol by giving a more formal analysis of these properties. We also plan to extend the protocol to use more than one transporter and to include the inverse case of the return of a product. In fact, a protocol for the return of the product would mirror the proposed one by exchanging the roles of the merchant and the customer, keeping that of the transporter the same. In the future version of P2T, payments

can make use of the payment channel such as Lightning Network (on Bitcoin) or Raiden Network (on Ethereum). The implementation on payment channels would make payments faster and increase privacy since most transactions would never appear on blockchain.

References

1. Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In *Advances in Cryptology – EUROCRYPT 2020*, pages 373–402. Springer International Publishing, 2020.
2. Steve Ellis, Ari Juels, and Sergey Nazarov. ChainLink,A Decentralized Oracle Network, September 2017.
3. Hussam Juma, Khaled Shaalan, and Ibrahim Kamel. A survey on using blockchain in trade supply chain solutions. *IEEE Access*, 2019.
4. Haya R. Hasan and Khaled Salah. Blockchain-based solution for proof of delivery of physical assets. In *Lecture Notes in Computer Science*, pages 139–152. Springer International Publishing, 2018.
5. Riham AlTawy, Muhammad ElSheikh, Amr M. Youssef, and Guang Gong. Lelantos: A blockchain-based anonymous physical delivery system. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, aug 2017.
6. Pieter Wuille. Hierarchical deterministic wallets. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, 2012.
7. Ilja Gerhardt and Timo Hanke. Homomorphic payment addresses and the pay-to-contract protocol. *ArXiv*, abs/1212.3257, 2012.
8. Hashlock - Bitcoin Wiki. https://en.bitcoin.it/wiki/hashlock.
9. Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, September 2019.
10. Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.