UNIVERSITY of York

This is a repository copy of *Refund Attacks on Bitcoin's Payment Protocol*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/144143/

Version: Submitted Version

Conference or Workshop Item:
McCorry, Patrick, Shahandashti, Siamak F. orcid.org/0000-0002-5284-6847 and Hao, Feng (2017) Refund Attacks on Bitcoin's Payment Protocol. In: Financial Cryptography and Data Security, 22-26 Feb 2016.

https://doi.org/10.1007/978-3-662-54970-4_34

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Refund attacks on Bitcoin's Payment Protocol

Patrick McCorry, Siamak F. Shahandashti, Feng Hao

School of Computing Science, Newcastle University UK
(patrick.mccorry, siamak.shahandashti, feng.hao)@ncl.ac.uk

Abstract. BIP70 is a community-accepted Payment Protocol standard that governs how merchants and customers perform payments in Bitcoin. This standard is supported by most major wallets and the two dominant Payment Processors: Coinbase and BitPay, who collectively provide the infrastructure for accepting Bitcoin as a form of payment to more than 100,000 merchants. In this paper, we present new attacks on the Payment Protocol, which affect all BIP70 merchants. The *Silkroad Trader* attack highlights an authentication vulnerability in the Payment Protocol while the *Marketplace Trader* attack exploits the refund policies of existing Payment Processors. Both attacks have been experimentally verified on real-life merchants using a modified Bitcoin wallet. The attacks have been acknowledged by both Coinbase and Bitpay with temporary mitigation measures put in place. However, to fully address the identified issues will require revising the BIP70 standard. We present a concrete proposal to revise BIP70 by providing the merchant with publicly verifiable evidence to prevent both attacks.

1 Introduction

Bitcoin [20], the world's first successful crypto-currency, is increasingly becoming a popular method of payment for e-commerce due to low transaction fees and the ease of use supplied by third party Payment Processors. BitPay and Coinbase are currently the two dominant Payment Processors that handle Bitcoin payments for more than 100,000 merchants. Their customers include Dell, Microsoft, Overstock, Shopify, Paypal and CeX. This demonstrates that large organisations are placing trust in Bitcoin as a viable form of payment. In fact, Overstock claimed to have made $3 million worth of Bitcoin sales in 2014 [12]. These Payment Processors are attractive due to their ability to convert bitcoins into fiat currency instantly which removes the risk involved in Bitcoin's price volatility on behalf of the merchant.

Both Payment Processors and all merchants are recommended to follow the community accepted BIP70: Payment Protocol standard that was proposed by Andresen and Hearn [5] to be used with Bitcoin. The motivation for this protocol is to reduce the complexity of Bitcoin payments as customers are no longer required to handle Bitcoin addresses[1]. Instead, the customer can verify the merchant's identity using a human-readable name before authorising a payment.

[1] A form of identity (26–35 alphanumeric characters) that is related to a public-private key pair and is used to send/receive bitcoins.

At the time of a payment authorisation, the customer's wallet will also send a refund Bitcoin address to the merchant that should be used in the event of a future refund.

The Payment Protocol provides two pieces of evidence that can be used in case of a dispute with an arbitrator. The customer has evidence that they were requested to authorise a payment if they keep a copy of the signed *Payment Request* message. This can be considered evidence as the customer could not have produced the signature without the co-operation of the merchant. The second piece of evidence for both the customer and the merchant is the payment transaction as the payment is signed by the customer and time-stamped on Bitcoin's Blockchain, that is stored by most users of the network. In this paper, we argue that a third piece of evidence is required to authenticate the refund address sent from the customer as the protocol recommends the customer's payment and refund addresses not be the same. This flexibility leads to the following attacks:

- The *Silkroad Trader* attack relies on a vulnerability in the Payment Protocol as the customer can authenticate that messages originate from the merchant, but not vice-versa. This allows a customer to route payments to an illicit trader via a merchant and then plausibly deny their own involvement.
- The *Marketplace Trader* attack focuses on the current refund policies of Coinbase and BitPay who both accept the refund address over e-mail [26][9]. This allows a rogue trader to use the reputation of a trusted merchant to entice customers to fall victim to a phishing-style attack.

Without the knowledge of our attacks, Schildbach asked the Bitcoin-Development mailing list why the refund address in the Payment Protocol was currently unprotected [24] and one of the original authors responded:

> *We talked about signing it with one of the keys that's signing the Bitcoin transaction as well. But it seems like a bit overkill. Usually it'll be submitted over HTTPS or a (secured!) Bluetooth channel though so tampering with it should not be possible. - Mike Hearn [13]*

As seen above, a solution may involve the customer endorsing a refund address using one of the public keys that authorised the transaction. However, the author stated this solution was an overkill as the refund address is currently protected in an HTTPS communication channel. We demonstrate that the HTTPS communication channel cannot protect the refund address as it only provides one-way authentication, the customer can authenticate messages originated from the merchant, but not vice-versa. At first glance, the 'overkill' solution suggested by Hearn could provide the evidence required for the merchant. Unfortunately, this solution opens the door to another attack which allows a malicious co-signer the sole authority to endorse the refund address used by the merchant and thus steal the bitcoins of other co-signers. We will discuss this in detail in Section 5.

Contributions. Our contributions in this paper are summarised below:

- We present new attacks on Bitcoin's Payment Protocol and the current practice of both the Payment Processors,

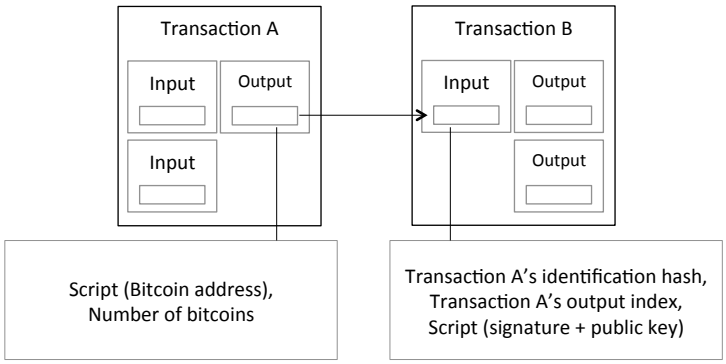

Fig. 1. Information stored in the inputs and outputs of Bitcoin transactions

- We present real-world experiments that demonstrate how merchants today are vulnerable to both attacks using a modified Bitcoin wallet.
- We propose a solution that removes the incentive to perform both attacks as the merchant is provided with publicly verifiable evidence whose origin can be verified by an arbitrator.

2 Background

We discuss background information about Bitcoin before presenting the community accepted Payment Protocol standard.

2.1 Bitcoin

We discuss three concepts that are needed to understand Bitcoin's Payment Protocol. These include Bitcoin addresses that act as a form of identification, Transactions that are used to send/receive bitcoins and the Blockchain that stores all transactions on the network.

A **Bitcoin address** is a form of identification in the Bitcoin community that is used to receive bitcoins and authorise payments. An address can be described as the hash of an EC (Elliptic Curve) public key and the accompanying private key is used to produce ECDSA (Elliptic Curve Digital Signature Algorithm) signatures to authorise payments.

A **Transaction** consists of one or more inputs and one or more outputs as seen in Figure 1. Briefly, an input specifies the source of bitcoins being spent (the previous transaction's identification hash and an index to one of its output) and is accompanied with signature(s) and public key(s) of the sender to authorise the payment. An output specifies the new owner's Bitcoin address and the number of bitcoins being sent. Strictly, these inputs and outputs are controlled using a Forth-like scripting language to dictate the conditions required to claim the bitcoins. The dominant script today is the 'pay-to-pubkey-hash' which requires

a single signature from a Bitcoin address to authorise the payment. On the other hand, the 'pay-to-script-hash' approach enables a variety of transaction types and was introduced as a soft-fork in BIP16 [4]. In practice, this 'pay-to-script-hash' script is widely used[2] to enable escrow services and multi-signature authorisation (k of n keys required to claim bitcoins).

The **Blockchain** is responsible for storing the entire network's transaction history with a relatively secure time-stamp [20]. This ledger is an append-only data structure and is stored by most users of the network. To append new transactions to this ledger requires a computationally difficult proof of work puzzle to be solved. 'Miners' are responsible for computing this proof of work and are rewarded in bitcoins for appending a new 'block' of transactions to the ledger.

2.2 Payment Protocol

Andresen and Hearn proposed the Payment Protocol which has been accepted as a standard in BIP70 [5] and is supported by several prominent wallets. The goal of this protocol is described in the standard as the following:

> *"This BIP describes a protocol for communication between a merchant and their customer, enabling both a better customer experience and better security against man-in-the-middle attacks on the payment process."*

Communication between the customer and merchant is sent over HTTPS[3] and importantly, the customer is also responsible for broadcasting the payment transaction to the Bitcoin network. In this HTTPS setting, the merchant must have an X.509 certificate issued by a trusted Certificate Authority. This is necessary to let the customer authenticate messages from the merchant.

Figure 2 outlines the messages exchanged and actions performed for the protocol. To initiate, the customer clicks the 'Pay Now' button on the merchant's website to generate a Bitcoin URI. This URI opens the customer's Bitcoin wallet and downloads the *Payment Request* message from the merchant's website. The wallet verifies the digital signature for the message using the public key found in the merchant's X.509 certificate (and checks the merchant's certificate for authenticity using the operating system's list of root certificate authorities). A human-readable name for the merchant[4] and the number of requested bitcoins is displayed on-screen and the customer must check this information before clicking 'Send'. Upon authorisation, the wallet performs two actions:

1. The customer's wallet sends one or more payment transactions to the Bitcoin network.

[2] Currently 8.9% of all bitcoins are stored using the 'pay-to-script-hash' approach [1].
[3] The protocol specification allows messages to be sent over HTTP and for the merchant not to have an X.509 certificate, but this is not considered secure.
[4] URL from the the X.509 certificate's 'common name' field.

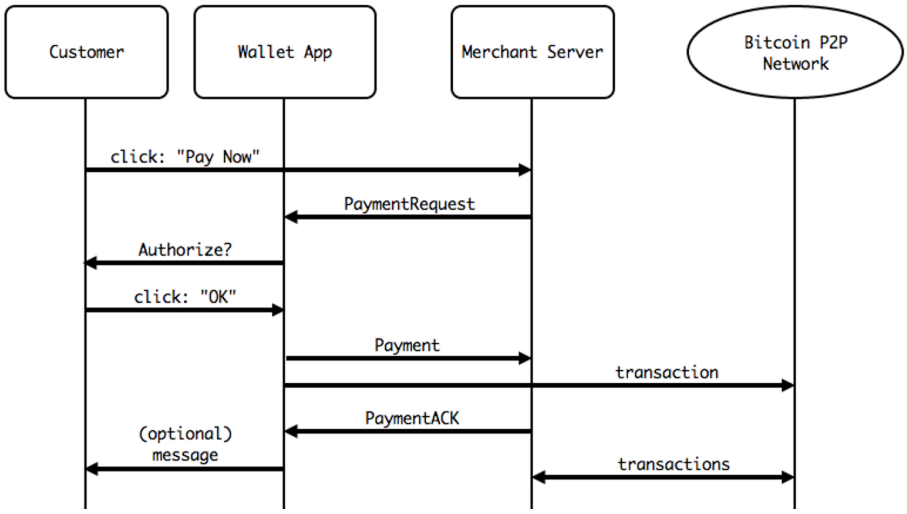

Fig. 2. Overview of the Payment Protocol [5]

2. The *Payment* message which includes the payment transactions and refund addresses is sent to the merchant's website.

The merchant responds to the customer's *Payment* message with a *Payment Acknowledgement* message which notifies the customer's wallet to display a confirmatory 'Thank you' message. Furthermore, once the merchant has detected the payment transaction on the Bitcoin network, the customer's web browser is refreshed to display a confirmation page. For the rest of this paper, we focus on messages sent over the HTTPS communication channel as seen in Figure 2 and for simplicity we assume the customer only sends a single payment transaction. The content for each message is the following:

- The **Payment Request** message contains a unique payment address M, requested number of bitcoins ฿, creation time for request t_1, expiry time for request t_2, a memo message m_{M}, a payment URL u_{M} and some merchant-specific data to link any future payments z_{M}. The contents of this message is signed using the private key xsk_{M} that corresponds to the merchant's X.509 certificate public key such that $\sigma_{\mathrm{M}} = S_{xsk_{\mathrm{M}}}(\mathrm{M}, ฿, t_1, t_2, m_{\mathrm{M}}, u_{\mathrm{M}}, z_{\mathrm{M}})$ where S is the signature algorithm.
- The **Payment** message contains a repeat of the merchant-specific data z_{M}, a payment transactions τ_{C}[5], a list of refund addresses $(R_{\mathrm{C}_1}, ..., R_{\mathrm{C}_n})$ and the number of bitcoins ฿ that should be refunded to each address such that $((R_{\mathrm{C}_1}, ฿_1), ..., (R_{\mathrm{C}_n}, ฿_n))$ and a memo from the customer m_{C}. There is no restriction to the number of refund addresses sent to the merchant and the

[5] A single payment transaction τ_{C} is considered for simplicity. The protocol supports one or more payment transactions, and our results still apply in this case.

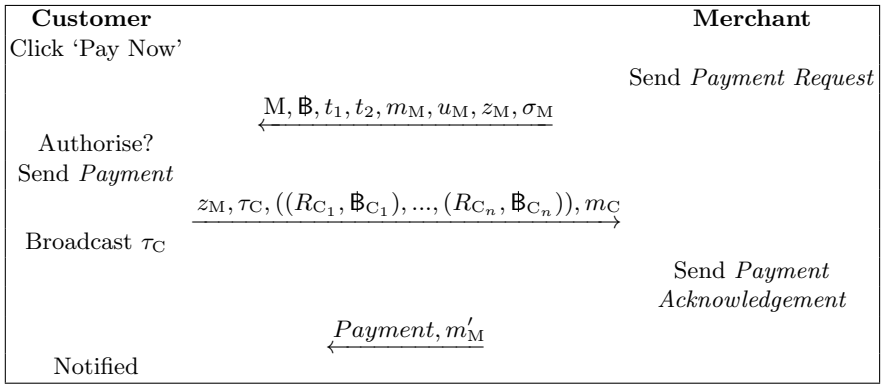

Fig. 3. Message contents for the Payment Protocol

customer is responsible for deciding how the bitcoins are refunded amongst the refund addresses provided. Merchants expect one or more *Payment* messages until all requested bitcoins have been received.

– The **Payment Acknowledgement** message is a repeat of the customer's *Payment* message and includes an optional memo m'_M from the merchant.

As seen in Figure 2, the refund address sent in the *Payment* message is not digitally signed by the customer and its integrity relies on the HTTPS communication channel established between the customer and the merchant to prevent man-in-the middle attacks. This lack of mutual authentication in the Payment Protocol and the refund policy of both Payment Processors to accept refund addresses over e-mail enables the attacks outlined in the next section.

3 Attacking the Payment Protocol

In this section, we outline attacks which are feasible due to an authentication vulnerability in the Payment Protocol and the refund policy of both Payment Processors. Fundamentally, our attacks rely on the merchant's inability to distinguish if the refund address originated from the same pseudonymous customer that authorised the payment. As well, these attacks are successful even when all messages are sent over an HTTPS communication channel.

3.1 Silkroad Trader Attack

This attack allows a customer to route payments to an illicit trader via an honest merchant and then plausibly deny their own involvement in the refund transaction. The main idea behind this attack relies on the customer's ability to swap the refund address in their *Payment* message with a Bitcoin address under the control of an illicit trader. Most importantly, the customer is not required to endorse the illicit trader's Bitcoin address with a digital signature.

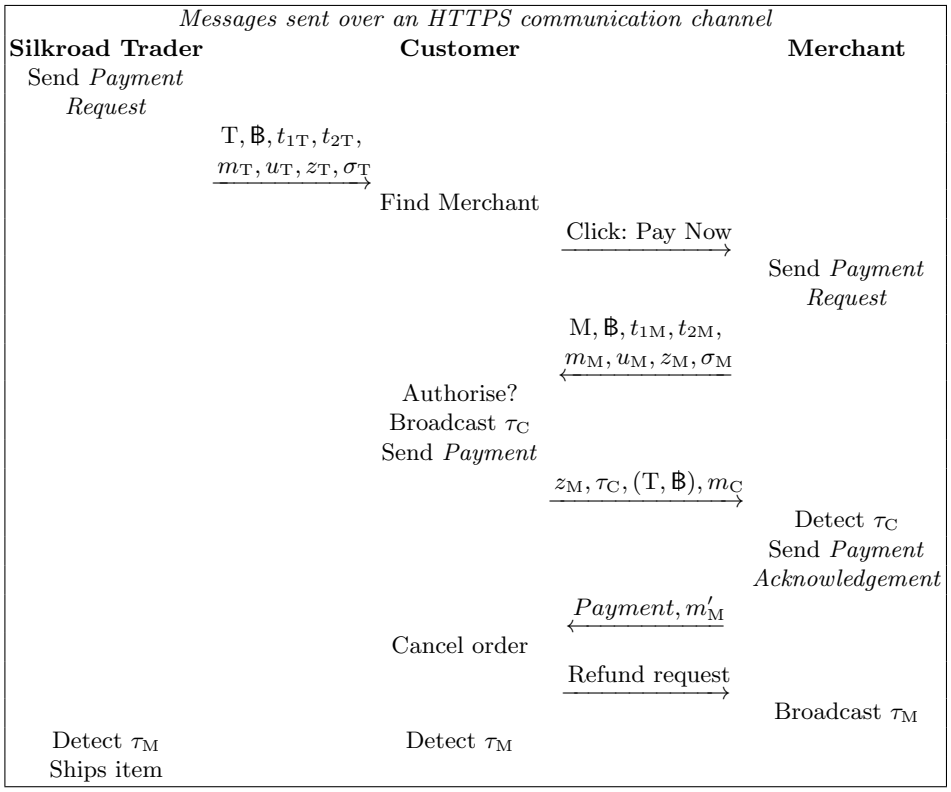

Fig. 4. *Silkroad Trader* attack allows a customer to route bitcoins to an illicit trader via an honest merchant and then plausibly deny their involvement.

Figure 4 is an outline of the Silkroad Trader attack. It begins with the customer visiting the website of an illicit trader. The customer downloads a *Payment Request* message for the desired 'illicit goods' they wish to purchase before searching for a merchant who supports the Payment Protocol and is selling an item approximately equal (or greater) in price. Once a merchant and item has been found, the customer clicks 'Pay now' to start the payment process and downloads a *Payment Request* message from the merchant's website. To commence the attack, the customer's wallet authorises the payment transaction τ_C and inserts the illicit trader's payment address T in the *Payment* message as the refund address (instead of their own refund address) and then sends the message to the merchant.

The customer must request a refund to finish the attack once their Bitcoin wallet has received the *Payment Acknowledgement* message alongside a confirmation e-mail from the merchant. Assuming the merchant follows the Payment Protocol faithfully, the refunded bitcoins in τ_M are sent to the illicit trader's payment address T. Also, the customer can detect the refund transaction (mer-

chant sending bitcoins to the illicit trader) τ_{M} on the Bitcoin network before contacting the illicit trader for an acknowledgement that the 'illicit goods' have been dispatched.

Ideally, if this attack happened in practice, the merchant could provide the *Payment* message as publicly verifiable evidence that the bitcoins were sent to the refund address provided by the pseudonymous customer. Unfortunately, the customer may plausibly deny having supplied the illicit trader's payment address due to their lack of endorsement, and hence claim that the merchant has forged the message single-handedly.

3.2 Marketplace Trader attack

In practice, the policy of Coinbase and BitPay encourages customers to provide refund addresses using an external method of communication such as e-mail [9][26] which ignores the refund address sent in the Payment Protocol. This deviation from the protocol is the basis of a new phishing style attack as a 'rogue trader' can use the reputation of a 'trusted' merchant to encourage potential customers to purchase an item from their website.

Figure 5 outlines this attack. It begins with the rogue trader establishing a website that sells the latest products well below the market rate to attract customers to their store. Most customers may be suspicious that the rogue trader can offer these prices and may wisely think it is a scam. To encourage customers to proceed with a purchase, the rogue trader can advertise that all payments are sent to a trusted merchant such as CeX and there is little reason not to trust them. When a customer proceeds to checkout on the rogue trader's website and clicks 'Pay now', the rogue trader's website can automatically fetch a *Payment Request* message from the trusted merchant's website and forward this to the customer. The customer's wallet opens the genuine *Payment Request* message and displays a human-readable name for the trusted merchant alongside the number of requested bitcoins. This can boost the customer's confidence that the rogue trader is legitimate as the payment is sent to the 'trusted' merchant.

Unfortunately, the customer falls victim to the attack upon authorising the payment as they are unwittingly paying for a purchase on behalf of the rogue trader to the trusted merchant. The rogue trader detects the payment[6] transaction on the Bitcoin network and refreshes the victim's web browser to display a fake confirmation page (remember, the customer's web browser is connected to the rogue trader's website). The rogue trader can proceed to cancel the order and send a new refund address over e-mail to the trusted merchant. As the merchant's policy is to use an external method of communication to authenticate customers and deviate from the Payment Protocol standard - then the refund address sent by the rogue trader over e-mail should receive the bitcoins.

Furthermore, the customer cannot be aware this attack has occurred as they lack enough information to identify the refund transaction on the Bitcoin network. More importantly, this attack is deployable single-handedly by a rogue

[6] Currently 50% of nodes on the network receive a new transaction within 5 seconds [2].

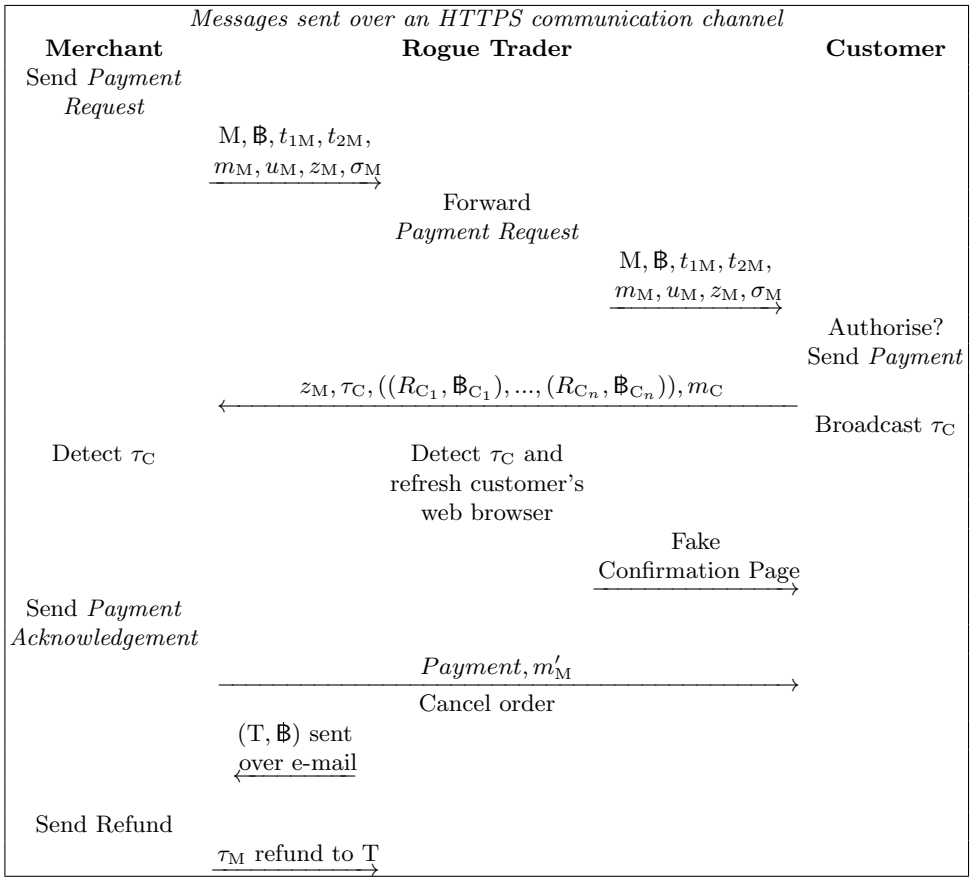

Fig. 5. *Marketplace Trader* attack involves a rogue trader using the reputation of a 'trusted' merchant to encourage customers to fall victim to a phishing-style attack.

trader and does not require the co-operation of a 'trusted' merchant. In fact, the trusted merchant may only become aware of this scam if contacted in the future by the customer.

4 Real-world experiments

Our experiments aim to verify the current practice of processing refunds by merchants, and assess the feasibility of the attacks. We purchased items from real-life merchants using a modified Bitcoin wallet before requesting for the order to be cancelled and a refund processed. The attack is considered successful if the refunded bitcoins are received by the adversary's wallet. The merchants used during these experiments are based in the UK and are supported by BitPay or Coinbase. The bitcoins used for the experiments are owned by the authors

and no money is sent to any illicit trader. All experiments have been ethnically approved by Newcastle University's ethical committee.

4.1 Proof of concept wallet

We have developed a wallet which supports the Payment Protocol and automates the *Silkroad Trader* attack. We explain how our wallet works step-by-step:

1. The customer inserts the illicit trader's *Payment Request* URI into the wallet which stores both the request and Bitcoin address for later use.
2. The customer finds an item equal (or greater) in value as the 'illicit goods' and inserts the merchant's *Payment Request* URI into their wallet.
3. The wallet provides a list of refund addresses that can be chosen for the *Payment* message that is sent to the merchant and the customer can choose the illicit trader's Bitcoin address.
4. Assuming a refund has been authorised by the merchant, the wallet can detect the merchant's refund transaction on the network and include it in a *Payment* message that is sent to the illicit trader.
5. The wallet is notified by a *Payment Acknowledgement* message from the illicit trader that the payment has been received.

4.2 Simulation of attacks

We discuss our experience carrying out a simulation of both attacks against real world merchants using arbitrary identities (i.e., random name, e-mail address, telephone number, delivery/billing addresses created for experiments only). Only e-mail is used to communicate with each merchant. Our results for the *Silkroad Trader* attack are as follows:

Cex refunded the bitcoins within 3 hours of cancelling the order and used the refund address from the Payment Protocol.

Pimoroni Ltd refunded the bitcoins within a single business day and used the refund address from the Payment Protocol.

Scan refunded the bitcoins after 26 days and used the refund address from the Payment Protocol. The delay was due to Scan initially requesting us to provide a refund address over e-mail, but we insisted using the one specified in the original payment message.

Dell were unable to process the refund due to 'technical difficulties' and requested our bank details. We informed them that we did not own a bank account and Dell suggested sending the refund as a cheque. While not the experiment's aim, this potentially opens Dell as an exchange for laundering tainted bitcoins.

To simulate the *Marketplace Trader* attacks we sent the refund address in an e-mail to the merchants. Assuming the merchants accepted e-mail as a good form of authentication and ignored the refund address sent in the Payment Protocol, then the phishing-style attack we described earlier could happen in practice. Our results were the following:

Something Geeky refunded the bitcoins within a single business day to a refund address sent over e-mail.

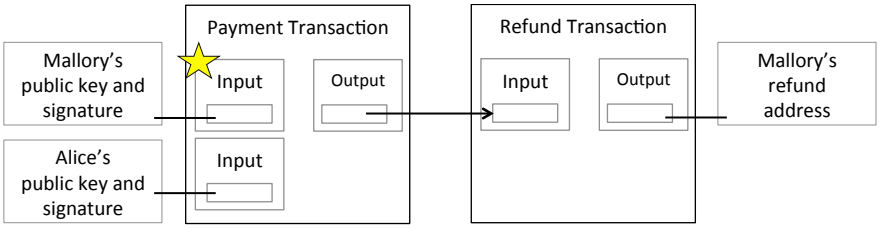

Fig. 6. The malicious co-signer attack allows a co-signer the sole authority to endorse the refund address used by the merchant and thus steal the bitcoins of other co-signers

Girl meets dress refunded the bitcoins within 11 business days to a refund address sent over e-mail. The delay was due to the merchant initially thinking we paid using a bank transfer.

BitRoad refunded the bitcoins within a single business day to a refund address sent over e-mail. In this experiment, we registered using a non-existing e-mail address and requested for the order to be cancelled using a variant of the e-mail address. This demonstrates that even the registered e-mail address to initiate the purchase is not being used to authenticate the customer.

5 Solution

We propose providing the merchant with publicly verifiable evidence that can cryptographically prove the refund address received during the protocol was endorsed by the same pseudonymous customer who authorised the payment.

A solution proposed by Hearn [13] assumes the payment transaction is authorised by a single customer and recommends endorsing the refund address using any key which authorised the transaction. However, it is not valid to assume that a transaction has been authorised by a single customer due to the nature of a Bitcoin transaction. If the adversary is responsible for sending the *Payment* message to the merchant, then they have the sole authority to endorse the refund address used by the merchant as seen in Figure 6.

Our proposed solution prevents this attack by requiring each key that authorised the transaction to also endorse its own refund address. In the event of a refund the merchant sends the same (or less) number of bitcoins received[7] from each transaction input to an associated refund address.

5.1 Proposed Solution

To achieve a signature solution requires changes to each message sent as part of the protocol. We outline these changes in Figure 7 and explain each message separately before discussing their implications.

The **Payment Request** message considers the memo m_{M} as a mandatory parameter and should contain enough information for the customer(s) to be

[7] A transaction input does not record the number of bitcoins 'sent' and instead references an output from a previous transaction which specifies the bitcoins.

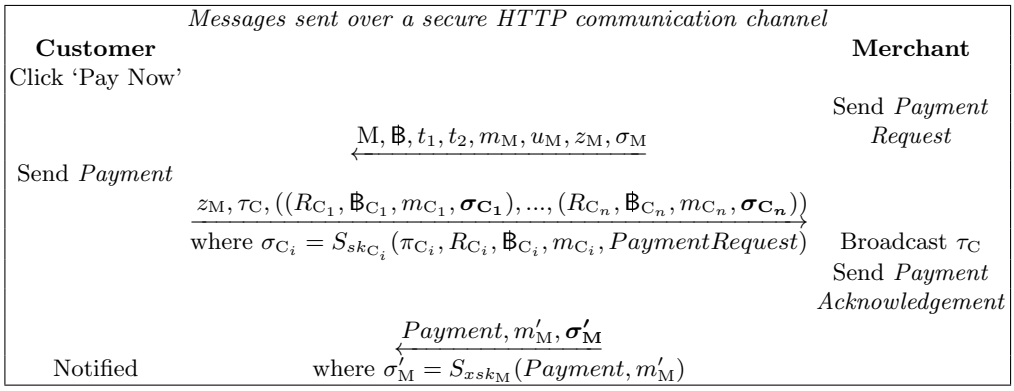

Fig. 7. A single customer sends a payment to the merchant

aware that this payment request is only for them, e.g. the registered e-mail address, delivery address, product information, etc. This memo field should also include customer-specified instructions to provide evidence that the merchant followed any instructions provided by the customer. The payment address M should be unique for each *Payment Request* and like before, there should be no restriction on the number of times a customer can download the same *Payment Request* to support paying from multiple devices or sharing with others.

The **Payment** message aims to associate each transaction input π_{C_i} with a refund address R_{C_i} by endorsing the refund address using the same keys that authorised the transaction input. We assume the customer is no longer responsible for broadcasting the payment transaction τ_C to the Bitcoin network; instead, the responsbility of broadcasting the payment transaction should fall on the merchant (as recommended by one of the original authors of the Payment Protocol[8]). For simplicity, we describe our solution using a single *Payment* message and payment transaction τ_C[9].

Each refund address endorsement signature is $\sigma_{C_i} = S_{sk_{C_i}}(\pi_{C_i}, R_{C_i}, Ƀ_{C_i}, m_{C_i}$ *Payment Request*), where S is the signature algorithm, sk_{C_i} is the private key which corresponds to the key that authorised the transaction input, π_{C_i} is a concatenation of the elements that constitute the signed transaction input, R_{C_i} is the refund address, $Ƀ_{C_i}$ is the number of bitcoins to refund, m_{C_i} is an additional memo from the customer and *Payment Request* is the signed message from the merchant. These parameters were chosen to clarify which transaction input is associated with the endorsed refund address and to ensure this endorsement is only valid for this *Payment Request* message. The concatenated information π_{C_i} is the data stored inside the transaction input and includes: the previous transaction identification hash, an index for the output in the referenced trans-

[8] `https://groups.google.com/forum/#!msg/bitcoinj/ymFRupTSRJQ/zANj2RpslCcJ`

[9] Our solution continues to allow customers to send one or more *Payment* messages to the merchant until all requested bitcoins have been received. Furthermore, these messages can contain a list payment transactions.

action and the script which contains a signature to authorise the payment and its corresponding public key.

A new refund policy for the merchant is required as each transaction input is responsible for endorsing a refund address. We propose the bitcoins associated with each refund address must be equal to (or less than) the number of bitcoins sent in the respective transaction input. The merchant must also check that the total number of bitcoins associated with all refund addresses in this message is equal to (or less than) the number of bitcoins he received in the payment transaction. These checks are necessary as the payment transaction can have additional outputs for change and the merchant needs to ensure he does not refund more bitcoins than he received from each transaction input. For example, if the transaction has a single input of ฿5 (from the customer) and two outputs: ฿4 (to the merchant) and ฿1 (to the customer as change), then the merchant must ensure the customer is only refunded ฿4 (and not ฿5).

The message content sent to the merchant is outlined in Figure 7 and includes: the merchant-specific data z_{M}, the complete transaction τ_{C} and a list of refund addresses alongside their associated endorsement signatures and the number of bitcoins to refund $((R_{\mathrm{C}_1}, ฿_{\mathrm{C}_1}, m_{\mathrm{C}_1}, \sigma_{\mathrm{C}_1}), ..., (R_{\mathrm{C}_n}, ฿_{\mathrm{C}_n}, m_{\mathrm{C}_n}, \sigma_{\mathrm{C}_n}))$.

The **Payment Acknowledgement** message is signed using the merchant's X.509 private key and repeats the customer's *Payment* message alongside an additional memo m'_{M}. The signature is $\sigma'_{\mathrm{M}} = S_{xsk_{\mathrm{M}}}(Payment, m'_{\mathrm{M}})$ where S is the signature algorithm and xsk_{M} is the private key that corresponds to the merchant's public key in their X.509 certificate.

We simplified the notation for σ_{C_i} to only show the case when the customer endorses the ith refund address using a *single* signing key. However, if the multi-signature approach is used to authorise the transaction input, then a threshold of k signing keys should be used to endorse the respective refund address. Each customer with control of 1 of k keys can independently authorise the transaction input and the corresponding refund address. Both signatures which endorse the refund address and authorise transaction input are sent to the other co-signers to be included in the *Payment* message.

5.2 Discussion

Our solution provides a **proof of endorsement** as the refund address received by the merchant is signed using the same set of keys used to authorise the transaction. This evidence removes the customer's plausible deniability for their involvement in the *Silkroad Trader* attack and provides an incentive for the merchant to use the refund address sent during the Payment Protocol which prevents the *Marketplace Trader* attack. The merchant does not need to distinguish whether or not the payment has been split which preserves the customers privacy and prevents the malicious co-signer attack as co-signers cannot endorse the refund addresses of others. These additional signatures are handled by the wallet on behalf of the user. Also, **no connection to the peer to peer network** is required for the customer as the merchant is responsible for broadcasting the payment transaction τ_{C} and this prepares the Payment Protocol to **support off-chain transactions** such as the Lightning Network [21].

Furthermore, we explored other potential solutions such as requesting the customer to provide a signature from the refund address at the time of payment (instead of using the same keys that authorised the transaction) or including secret data inside the merchant-specific data field z_{M}. The former is not satisfactory as proving ownership of the refund address does not necessarily link the refund address to the same keys that authorised the transaction and the latter remains vulnerable to the *Marketplace Trader* attack as the rogue trader has access to the *Payment Request* message. As well, the attacks introduced in this paper also stem from the fact that merchants have no community-accepted refund protocol today. While researchers have proposed secure post-payment communication protocols [15] in the past which could conceivably be used to support arranging refund in a private and authenticated manner, this remains a subject for further investigation in future work.

Payment Processors are expected to perform anti-money-laundering policies on behalf of their merchants [10]. The state of New York recently released Bitlicense [25] to outline regulation for Bitcoin businesses. Our solution enhances the book-keeping required for this license as the signed *Payment* message is cryptographic evidence that the pseudonymous customer has endorsed the transaction-related information required for auditing by investigators. We improve the mandatory customer receipt which is currently a static web-page or an e-mail by using the *Payment Acknowledgement* message as a cryptographic receipt as it is signed by the merchant's X.509 private key. Similar cryptographic evidence has been explored in the Bitcoin research community and two examples include: providing a warrant to hold a mixer accountable in the event of any wrongdoing with Mixcoin/Blindcoin [8][27], and to compute a proof of solvency [11] that demonstrates the business is financially in good standing to customers.

Clustering techniques have been demonstrated to link a group of Bitcoin addresses to a single pseudonymous user [6]. Meiklejohn et al. [17] identified that 374.49 BTC stolen from Betcoin in April 2012 and 4,588 BTC from the Bitcoinica theft in May 2012 were sold at Bitcoin-24, Mt Gox, BTC-e, CampBX and Bitstamp. Also, Reid et al. [22] tracked 25k stolen bitcoins and deduced LulSec's involvement in the theft. These analysis techniques using the Blockchain are currently supporting criminal charges in the Silkroad Trial [3]. However, privacy-enhancing protocols [14][23][16] and altcoins [18][19] are actively reducing the effectiveness of these analysis techniques. Nevertheless, these techniques provide a platform for the *Silkroad Trader* attack as independent observers may discover merchants sending bitcoins to an illicit trader and then publicly release the 'evidence'. Our solution provides the merchant with publicly verifiable evidence to demonstrate a customer's deception.

5.3 Inherent issues due to Bitcoin

We outline four issues that are inherent due to Bitcoin that need to be considered for our solution:

First, the *proof of endorsement* evidence can only authenticate pseudonymous customers as the Payment Protocol lacks the type of *real-life identity endorsement* that comes with banks. While protecting an honest merchant, our

Step	Description	Time
	Customer in the current protocol	
1	Verify merchant's certificate and chain of certificates authenticity	0.83 ms
2	Verify merchant's signature on the Payment Request message	0.08 ms
3	Sign a single transaction input	0.08 ms
4a	Fetch a list of previously generated refund addresses $R_{C_1}, ..., R_{C_k}$	0.72 ms
4b	Generate a new refund address R_C from the wallet's key pool	110.55 ms
5	Update wallet's address book with the refund address R_C	72.68 ms
	Total without 4b:	*74.39 ms*
	Total with 4b:	*184.94 ms*
	Merchant in the current protocol	
6	Verify the customer's payment transaction	0.29 ms
	Total:	*0.29 ms*
	Additional changes proposed for the customer	
7	Produce endorsement signature σ_C using the private key sk_C	0.11 ms
	New Total without 4b:	*74.49 ms*
	New Total with 4b:	*185.04 ms*
	Additional changes proposed for the merchant	
8	Fetch the transaction input's referenced transaction output	0.01 ms
9	Verify the transaction input's endorsement signature σ_C	0.13 ms
	New Total:	*0.43 ms*

Table 1. Time performance for proposed changes to the Payment Protocol

solution cannot prevent a malicious merchant simulating both attacks and insisting they were tricked.

Second, in a similar way to the original protocol, an observer of the Blockchain may be able to link the payment and refund transactions using the denominations of bitcoins sent and received.

Third, customers can re-sign the transaction to change the identification hash and broadcast it to the network. We recommend the merchant keeps a copy of the payment transaction received in the *Payment* message as a re-signed transaction cannot be used to verify the endorsement in the future.

Fourth, we assume merchants maintain the UTXO (Unspent Transaction Output) set to participate in the Payment Protocol. Without this list of spendable outputs, the merchant cannot independently verify transactions or calculate the number of bitcoins to refund for each transaction input. On the other hand, customers do not require the UTXO set and can continue to use SPV (Simplified Payment Verification) wallets for the Payment Protocol.

5.4 Solution performance

All tests are carried out on a MacBook Pro mid-2012 running OS X 10.9.1 with 2.3GHz Intel Core i7 and 16 GB DDR3 RAM. Time performance in Table 1 represents both the current Payment Protocol implementation and our proposed modifications for the Bitcoin Core Client while utilising 1 core. Furthermore, both signing operations in steps 3 and 8, and the verification operation in step 9, are performed using the Secp256k1 implementation which has recently re-

placed OpenSSL in Bitcoin Core [28]. Each step was executed 100 times and the reported times represent the average.

Steps 1–5 represent the customer's perspective in the current Payment Protocol's implementation. The wallet verifies the merchant's certificate authenticity using the chain of certificates that lead to a trusted root authority and verifies the merchant's signature on the *Payment Request* message before authorising at least one transaction input to authorise the payment. Then, the wallet fetches a list of pre-generated refund addresses and Step 4b only occurs if this list is empty as a new refund address must be generated. This refund address is associated with the payment for future reference. These steps require 74.39 ms if the list of pre-generated refund addresses is not empty, otherwise 184.94 ms is required. Our proposed change in Step 7 takes 0.11 ms and requires the customer's wallet to sign an endorsement message for the refund address, obtaining the signature σ_{C}. In total, the time required for the customer is 185.04 ms with Step 4b, and 74.49 ms without Step 4b.

Step 6 represents the merchant's perspective in the current Payment Protocol's implementation and requires 0.29 ms to check if the payment transaction with a single input is valid. We propose in Steps 8–9 that the merchant fetches the transaction output referenced in the payment transaction's input to let the merchant check the number of bitcoins associated with each refund address. Then, the transaction input's public key C is used to verify the endorsement signature. These proposed changes require 0.14 ms, and in total the time required for the merchant is 0.43 ms.

6 Payment Processors Response

We privately disclosed our attacks to the Payment Processors and received the following response:

BitPay acknowledged "the researchers have done their homework" and that "refunds are definitely a significant money laundering attack vector". They are now actively monitoring for refund activity on behalf of their merchants. Furthermore, after we disclosed our results, BitPay released a new refund flow [7] that recommends using the refund address provided in the Payment Protocol rather than the one supplied by email.

Coinbase acknowledged the 'Silkroad Trader' attack as a good example of an authentication vulnerability in the Payment Protocol. To prevent the *Marketplace Trader* attack, Coinbase no longer provides merchants the API to change the refund address if it has been supplied by the Payment Protocol. Also, they have updated their user documentation to discourage merchants sending the refund using their own bitcoins to bypass the API changes.

Bitt is preparing to launch merchant services for the Caribbean and acknowledged both attacks. They believe the endorsement evidence may support Payment Processors become more 'airtight' for future regulation.

These temporary mitigation measures help to address the *Marketplace Trader* attack, but not the *Silkroad Trader* attack. To fully address the latter, the BIP70 standard would need to be revised, as we have discussed in Section 5.

7 Conclusion

This paper presented two attacks that leverage an authentication vulnerability in Bitcoin's Payment Protocol and the refund policies of the two largest Payment Processors: Coinbase and BitPay. We experimentally demonstrated both attacks on real-life merchants using a proof of concept wallet before proposing a solution that provides the merchant with cryptographic evidence that the refund address received during the Payment Protocol has been endorsed from the same pseudonymous customer who authorised the transaction. Both Payment Processors have acknowledged our attacks and have implemented mitigation measures.

8 Acknowledgements

The second and third authors are supported by the European Research Council (ERC) Starting Grant (No. 306994). We would like to thank the original authors of the Payment Protocol; Mike Hearn for his constructive feedback on our proposed solution and recommendation to include customer-specified instructions and Gavin Andresen for reviewing this paper and giving feedback. Also, we thank the anonymous reviewers for their very good feedback.

References

1. Alcio. Monitor pay to script hash adoption. May 2015. http://p2sh.info/, Accessed on 2015-05-21.
2. S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao. Zombiecoin: Powering next-generation botnets with bitcoin. In *Bitcoin Workshop at Financial Cryptography and Data Security*. Springer, 2015.
3. I. Allison. Silk Road prosecutors talk about Bitcoin, Ripple and money laundering. *International Business Times*, Aug. 2015. http://www.ibtimes.co.uk/silk-road-prosecutors-talk-about-bitcoin-ripple-money-laundering-1517414.
4. G. Andresen. Pay to Script Hash. *Bitcoin Improvement Process*, 2012. https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki, Accessed on 2015-12-07.
5. G. Andresen and M. Hearn. BIP 70: Payment Protocol. *Bitcoin Improvement Process*, July 2013. https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki, Accessed on 2015-01-15.
6. E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.
7. BitPay. New Invoice Adjustment and Refund Flow. Aug. 2015. https://blog.bitpay.com/new-refund-flow/, Accessed on 2015-09-20.
8. J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.
9. Coinbase. How do I do a customer refund with the API? May 2015. https://support.coinbase.com/customer/portal/articles/ 1521752-how-do-i-do-a-customer-refund-with-the-api-, Accessed on 2015-05-15.
10. Fincen. Request for Administrative Ruling on the Application of FinCENs Regulations to a Virtual Currency Payment System. 2015. http://www.fincen.gov/news_room/rp/rulings/pdf/FIN-2014-R012.pdf, Accessed on 2015-09-07.

11. G. Dagher and B. Bunz and J. Bonneau and J. Clarke and D. Boneah. Provisions: Privacy-preserving Proofs of Solvency for Bitcoin Exchanges. In *The 22nd ACM Conference on Computer and Communications Security*. 2015.
12. B. Geiger. Overstock.com offers its staff the option of being paid in Bitcoin. 2015. http://fortune.com/2015/01/09/overstock-com-offers-its-staff-the-option-of-being-paid-in-bitcoin/ Accessed on 2015-02-26.
13. M. Hearn. Re: [Bitcoin-development] BIP 70 refund field. *Bitcoin-Development*, Mar. 2014. http://sourceforge.net/p/bitcoin/mailman/message/ 32157661/, Accessed on 2015-02-01.
14. G. Maxwell. CoinJoin: Bitcoin privacy for the real world. 2013. https://bitcointalk.org/index.php?topic=279249, Accessed on 2015-05-20.
15. P. McCorry, S. F. Shahandashti, D. Clarke, and F. Hao. Authenticated key exchange over bitcoin. In *Security Standardisation Research*, pages 3–20. Springer, 2015.
16. S. Meiklejohn and C. Orlandi. Privacy-enhancing overlays in bitcoin. In *Bitcoin Workshop at Financial Cryptography and Data Security*. Springer, 2015.
17. S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.
18. I. Miers, C. Garman, M. Green, and A. Rubin. Zerocoin: Anonymous Distributed E-cash from Bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.
19. Monero. Monero is a secure, private, untraceable currency. It is open-source and freely available to all. 2015. https://getmonero.org/home, Accessed on 2015-12-08.
20. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. November 2008. https://bitcoin.org/bitcoin.pdf, Accessed on 2015-01-01.
21. Y. Perez. Could the Bitcoin Lightning Network Solve Blockchain Scalability? 2015. http://www.coindesk.com/could-the-bitcoin-lightning-network-solve-blockchain-scalability/, Accessed on 2015-05-15.
22. F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Privacy, security, risk and trust (passat), 2011 IEEE Third International Conference on and 2011 IEEE third international conference on social computing* , pages 1318–1326, Oct 2011.
23. T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *Computer Security-ESORICS 2014*, pages 345–364. Springer, 2014.
24. A. Schildbach. Re: [Bitcoin-development] BIP 70 refund field. *Bitcoin-Development*, Mar. 2014. http://sourceforge.net/p/bitcoin/mailman/message/ 32157651/, Acccessed on 2015-02-1.
25. N. Y. State. Chapter i regulations of the superintendent of financial services, part 200. virtual currencies. *Department of finance services*, Feb. 2015.
26. M. Tur. Can BitPay refund my order? 2015. https://support.bitpay.com/hc/en-us/articles/203411523-Can-BitPay-refund-my-order-, Accessed on 2015-04-07.
27. L. Valenta and B. Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In *Bitcoin Workshop at Financial Cryptography and Data Security*, 2015.
28. P. Wuille. Switch to libsecp256k1-based ECDSA validation. *Bitcoin Github Repository*, Nov. 2015. https://github.com/bitcoin/bitcoin/pull/6954, Accessed on 2015-12-31.