Countering Double-Spend Attacks on Bitcoin Fast-Pay Transactions

John P. Podolanko, Jiang Ming
Department of Computer Science and Engineering
University of Texas at Arlington
Email: {john.podolanko | jiang.ming}@uta.edu

Matthew Wright
Center for Cybersecurity
Rochester Institute of Technology
Email: matthew.wright@rit.edu

***Abstract*—Bitcoin is a Proof-of-Work-based payment system that does not rely on a trusted authority. As its popularity grows, more businesses are starting to accept it as payment, including services like fast food restaurants and vending machines that must deliver goods soon upon payment. Unfortunately, these *fast pay* scenarios are vulnerable to fraudulent double-spending of Bitcoins. To protect consumers from having to cover the costs of these attacks, there is a growing need to find countermeasures to such attacks that are scalable and realistic to deploy. Several possible countermeasures have been proposed in prior work but not evaluated carefully. In this paper, we analyze these countermeasures and simulate their effects in a scaled Bitcoin P2P network using the Shadow framework. We find that integrating several of these countermeasures into an *enhanced observer* can be effective in detecting and alerting a vendor to an incoming double-spend attack in less than 28 seconds on average in our model.**

1. Introduction

Bitcoin is a digital currency with over US $16.4 billion in circulation and over $57 million in daily exchanges as of Jan. 1, 2017 [1]. It offers significant benefits to consumers and vendors, such as privacy and low cost for processing compared with credit cards. Bitcoin removes the reliance on a trusted third party to conduct transactions by implementing a Proof-of-Work (PoW) concept that is integrated into a public peer-to-peer (P2P) network [2]. It allows users to *mine* digital coins by performing calculations on block chains and to exchange digital coins through the use of digital signatures and time stamps aimed at preventing double-spending.

Bitcoin is particularly suited to *slow-pay* scenarios, in which the vendor delivers the goods only after the network confirms the transaction as correct. As Bitcoin continues to grow, however, so does its practical use in *fast-pay* scenarios, like ATM withdraws, vending machines, and other point-of-sale transactions where a fast delivery of goods is expected after payment. Unfortunately, it takes 10 minutes on average [3] to confirm that a Bitcoin transaction is not a double-spending attack, in which the attacker spends the same Bitcoins repeatedly and only one vendor actually gets the money. Recent research [3], [4], [5], [6], [7] has shown that double-spending attacks on fast payments are not only possible, but also significantly less costly to the attacker than originally thought. These attacks cost businesses money, and such costs are very likely to be passed down to consumers. Additionally, businesses are also likely to limit the maximum amount for each fast-pay transaction, and some potential fast-pay vendors will not accept Bitcoin at all, further harming consumers who seek the benefits of Bitcoin.

A few countermeasures to these attacks have been proposed by Karame et al. [3], [4] and Bamert et al. [5], such as communicating double-spending alerts among peers, inserting observers into the network, implementing a listening period, blocking incoming connection requests, and inspecting propagation depth of transactions. We know of no work, however, to analytically or experimentally validate or compare these ideas.

The purpose of our research is to investigate the effectiveness of these proposed double-spending countermeasures and their impacts on the performance, anonymity, decentralization, and other aspects of Bitcoin. We also explore whether some of the proposed ideas can be combined for a more effective solution.

This paper makes the following contributions:

1) We perform an analysis of representative countermeasures proposed in related works.
2) We use the modified Bitcoin code to conduct simulations in the Shadow framework [8] to determine the effectiveness of these defenses and the impact that the implementations have on individual client and network performance. We find that these modifications to the individual clients drive up CPU utilization by 63%.
3) We propose using a hybrid variation of the observer countermeasure together with deeper inspection of transaction details for greater effectiveness against double-spending attacks. We find that introducing new, special nodes called *enhanced observers* (ENHOBS) into the network so that they make up 2% of the total number of online nodes can alert vendors to a double-spending attack in 22 seconds on average.

2. Background

Bitcoin is an electronic P2P payment system that removes the need for trusted third parties [2]. This is ac-

complished through the use of cryptographic Proof-of-Work (PoW) in the conduction of financial transactions. The currency exchanged between peers is Bitcoins (BTCs). Each BTC has a unique Bitcoin address stored in the owner's digital wallet. These addresses are run through a hashing function that incorporates the BTC owner's private key and the public key of the recipient of the BTCs to produce a new Bitcoin address designating the recipient as the new owner. In other words, the coin's owner digitally signs the hash of the previous transaction through which the coin was received combined with the next owner's public key.

Every Bitcoin transaction is broadcasted to all peers, and each transaction is verified upon receipt [2]. Verification involves ensuring that the transaction is properly formed. Peers must also verify that the BTCs have not been previously spent. That requires checking the transaction against the blocks in the block chain. Once verified, the transaction is locally stored in the peers' memory pools, and all peers work on constructing a block.

Bitcoin blocks contain several transactions that get broadcast to all peers in the network [2]. Blocks are generated by incorporating a random nonce value into the hash function along with a Merkle hash of all previous transactions [3]. If the resulting hash does not begin with a special number of zero bits, the function is repeated with a different nonce value each time until a valid hash is generated. The number of leading zero bits is determined by the required difficulty of the next transaction block which is determined by the difference in timestamps between the current block and its predecessor. This difficulty requirement is used to maintain an average of 10 minute intervals between confirmed transaction blocks, and it is also the most significant reason Bitcoin is resistant to rogue block chains. Once a peer on the Bitcoin network finds a valid hash, it gets broadcast with the nonce to all other peers and is publicly verifiable. As of January 2017, the finder mines 12.5 new BTC as a result of producing a valid hash [9], which provides the incentive to validate the transaction.

Once a transaction appears in a valid block, it is confirmed in all but a few cases. Those cases are known as forks in the block chain. In the event two different mining pools simultaneously discover valid block hashes for a different set of transactions, they will both propagate them to the peers. Some peers will receive the block from one mining pool first and others will receive the other mining pool's block first. As the peers receive the other block, they see that it doesn't extend their current block chain and disregards it. This is resolved by whichever side of the fork happens to discover the hash for the next block first. As that new block is broadcast to all peers, the other side accepts the new block because it extends their block chain, and then they discover the fork which is quickly remedied. When that happens, some transactions from the dead side of the fork may be lost. It is for this reason, Bitcoin recommends waiting for a few blocks to be added to the block chain before delivering goods or services.

2.1. Bitcoin and Double-Spending

In traditional slow-pay Bitcoin, a double-spending attack would fail, because honest peers would only accept the first transaction it received. To overcome this, and get previously spent payments to be confirmed, the attacker would need to have more computational power than the rest of the P2P network combined.

Due to the increasing popularity of Bitcoin and the demand for a larger scope of Bitcoin-supported transactions, fast payment services are now being used worldwide. This poses a risk to fast-pay vendors, however, since these vendors generally need to provide their goods and services before receiving confirmation which takes 10 minutes on average but has been known to take upwards of 40 minutes [3]. Since they provide goods and services without getting confirmation, this opens them up to double-spending attacks.

Fast-pay, zero-confirmation transactions fundamentally clash with the very nature of Bitcoin by creating the need to trust a third party in order to conduct business. Namely, the vendor must trust that the customer isn't about to rip them off. It is for primarily this reason why Bitcoin advocates and researchers continue to recommend waiting for confirmation [4], [5], [6]. Still, Bitcoin is flexible and isn't limited to just slow-pay transactions. As such, many vendors feel that the benefits of conducting fast-pay Bitcoin transactions outweighs the risk of being robbed. To mitigate potential losses, fast-pay vendors generally adhere to recommendations and only offer Bitcoin as a payment option for low-value transactions.

To double spend Bitcoins, the attacker first creates two transactions. The first transaction, T_V, lists the vendor as the recipient of the BTCs, and the second transaction, T_A, lists another address owned by the attacker as the recipient of the BTCs. The attacker's goal is to have the vendor accept T_V long enough to deliver the goods or services and have the rest of the network accept T_A so that the attacker keeps the money. The attacker then sends out both transactions. The first transaction, T_V, is transmitted directly to the vendor, while T_A is broadcasted to the rest of the network. For this to happen, the IP address of the vendor must be known in order for the attacker to form a direct connection with the vendor, and the vendor must receive T_V before T_A arrives [5] ensuring that T_A will be automatically dropped when the vendor eventually receives it.

The other thing that must happen in a successful double-spend attack is that T_A must be confirmed in the block chain first or else T_V will actually be confirmed and that block will become the accepted block in the network. Given an equal propagation of both transactions, there is a 50 percent chance for either transaction to be confirmed. In summary, this attack requires more nodes working on T_A than on T_V to increase T_A's likelihood of being accepted into the block chain and requires that the vendor only sees T_V. Note that neighbors of the vendor will likely get T_V first (directly from the vendor) and thus drop T_A rather than propagate it to the vendor.

Additionally, there exists another form of double-spend attack known as the block withholding attack [10], [11] in which the attacker pools resources to create a block, B_V, which contains T_V. In this attack, the attacker blocks all other connections to the vendor and prevents the vendor from ever receiving all other blocks confirming T_A while sending B_V the moment it is calculated. The attacker essentially creates a fork in the block chain that will eventually be disregarded since no other mining pools work to extend this side of the fork [4]. Unlike the previous method, this method of double-spending can succeed in slow-pay transactions in which the vendor awaits confirmation.

3. Analysis of Countermeasures

Several prior works have described potential countermeasures to double-spending attacks against fast pay. In this section, we briefly analyze these countermeasures.

There is an existing metric for propagation depth built into Bitcoin transaction messages [12], and Bamert et al. [5] propose requiring the vendor to wait for a transaction to propagate a number of steps before accepting it. The notion is that if few nodes have seen the transaction, it could be untrustworthy, so greater depth is assumed to be better. With a chain of malicious nodes, however, an attacker could simply move T_V along until the propagation depth reaches the required threshold.

Bamert et al. also propose blocking of incoming connection requests as a countermeasure [5]. This essentially prevents the attacker from achieving one of the necessary conditions required to successfully perform a double-spend attack which is to directly connect to the targeted vendor. By blocking incoming connection requests, the attacker cannot establish a direct connection to the vendor from which to send the vendor T_V unless the vendor actually sends the attacker a connection request upon joining the Bitcoin network. To overcome this, the attacker can target newly joining vendors given that these vendors must still request connections to other peers to ensure it has the latest block chain information. The attacker would create several malicious nodes distributed throughout the network making it more likely for the vendor to randomly connect to one of them.

Karame et al. [3] propose a listening period countermeasure, in which honest nodes check all incoming transactions against T_V for a specified period of time once T_V has been received. If T_A is discovered within the window, then it alerts the vendor to the double-spend. This is not likely to be effective in detecting attacks in the short time it could take to complete a fast-pay transaction. In particular, the attacker could delay releasing T_A to other nodes long enough to avoid detection by the vendor within the window, and the attacker could effectively use multiple malicious nodes to inject T_A simultaneously and quickly enough that T_A reaches more mining nodes.

Karame et al. also propose the use of *observers* [3], nodes randomly distributed across the P2P network that forward all transactions to the vendor, thereby increasing the scope of transactions that can be detected in a listening period. In other words, if T_A is released in a part of the network that is far from the vendor but contains an observer, then the vendor could essentially know about T_A within a few seconds. This approach is somewhat effective, but it does not directly prevent the double-spend attack nor the propagation of T_A. Also, observers can easily be detected in the Bitcoin network by an attacker analyzing traffic patterns [13]. This could then allow the attacker to carry out targeted attacks such as DoS against the observers, re-enabling double spending.

Finally, Karame et al. propose a *peer alert* countermeasure [3], [4], in which peers conduct a deeper investigation of conflicting transactions and broadcast alerts to all peers if a double-spend attack is detected. While Bitcoin has alert mechanisms in place [14], they are currently unused. This approach may help to catch double-spenders, but it only catches them after the act and does nothing to prevent the attack once detected. Even if the attacker is discovered and its pseudonym blacklisted, the attacker could effortlessly create a new pseudonym and try again.

4. Proposal

To build a more effective countermeasure against double-spending attacks in the fast-pay setting, we propose introducing *enhanced observers* (ENHOBS), a hybrid of observers and the peer alert system. In this scheme, the ENHOBS will conduct deeper inspections of all transactions received and compare outputs as well as inputs. Once a double-spend attack is detected, an alert message that contains both transactions as evidence is sent through the P2P network. Once an alert is received and verified, any transactions matching the same inputs are dropped from the memory pool immediately.

This is different from the observers proposed by Karame et al. because those observers simply forward all transactions they receive to the vendor, leaving the vendor to do all the detective work [3]. In our approach, we have the observers do the investigation and only broadcast alerts through the network when a double-spend attack is discovered. This allows the other non-observer clients to better utilize their resources for mining and to respond to the alerts in a much more timely manner. The cost of this approach is the additional overhead of investigation. The observers proposed in [3] also generate unique traffic patterns that make them detectable. Specifically, they only unicast alert messages to the vendor as opposed to forwarding all incoming broadcasts. Our ENHOBS forward all incoming transactions and blocks as would any other client in the Bitcoin P2P network as well as alert messages so that their traffic patterns are indistinguishable from other peers in the network.

All other Bitcoin peers only require minimal modification to their client source code so that they accept and process alert messages. In the event an alert message is received, a peer verifies that the two transactions are indeed part of a double-spend attack by comparing the inputs and outputs of both transactions contained in the alert. Once

the alert has been validated, the peer then runs a one-time scan of all transactions in its memory bank, looking for the transaction that contains the input that matches the alert and removes that transaction. If no matching transaction is found, then nothing changes. This only adds an additional computing cost of $O(T)$, where T is the number of transactions that currently reside in the memory bank.

Afterwards, the peer keeps the alert message in memory for until two new blocks have been appended to the block chain. We keep this alert in memory in the event that one of the double-spend attempts happens to arrive after the alert has been processed. This adds a minimal $O(2a)$ to the overhead computing cost where a represents the number of alerts in memory.

To pay for the overhead of ENHOBS, we consider a subscription-based system to justify their costs of operation. The ENHOBS will operate as a collective on the Bitcoin P2P network with all standard clients, and alert messages will be sent only to the peers who have paid for a subscription to receive alerts for a low monthly fee paid to the collective in BTCs. The collective would then fairly distribute the BTCs to the enhanced observer clients. It is important that the collective must maintain a minimum number ENHOBS so that they may detect double-spends and alert the subscriber base quickly. The only other option would be to have a small number of altruistic parties absorb these costs for the benefit of all Bitcoin users.

To summarize our proposal, here is a list of benefits and potential setbacks that we have considered:

+ Because most of the CPU overhead is performed by the ENHOBS, subscriber clients incur negligible additional CPU overhead.
+ Non-subscriber Bitcoin clients incur no additional costs—resources or monetary.
+ The Bitcoin fast-pay market could potentially open up to large-value point-of-sale transactions and attract new clients.
– It is critical to this system that a balance is found and maintained between the number of subscribers and ENHOBS for optimal performance and low cost.
– This system doesn't address the potential for malicious intent, and it potentially opens up another attack vector for those wanting to cheat the ENHOBS system.

5. Experimental Design

To evaluate ENHOBS, we performed simulation experiments in the Shadow simulation framework [8]. Building on the Bitcoin software client, `bitcoind` v0.12, we wrote a new plug-in to Shadow and made the modifications to implement our proposal. We simulated 3,000 Bitcoin client nodes, which is roughly 20% of the size of the Bitcoin network during peak capacity [9], for a total of seven simulated days. Although network churn was not simulated, latency between nodes and computing power were simulated via a configuration file that was previously written for the Shadow plug-in for Bitcoin [15] to match actual Bitcoin node data. However, we modified that plug-in to be compatible with `bitcoind` v0.12. The setup also contained a single modified node that behaved normally with the exception that it also introduced a new double-spend attack into the network every twelve minutes, i.e. five attacks per simulated hour. Average CPU utilization was recorded for the entire duration of the simulation.

We first studied the unmodified Bitcoin client, labeled as "Vanilla," which generated our control data to which all other data was compared. In the first experiment of our proposal, labeled as "All ENHOBS," we modified the client software by adding ENHOBS functionality to all 3,000 clients on top of their normal duties and conducted a seven-day experiment in order to accurately assess the performance hit that each client would take on average for comparative purposes. The ENHOBS functionality consists of deeper transaction inspection, alert generation, processing of any received alerts which includes dropping any double-spend transactions in the memory pool and storing the alert in memory until two new blocks have been appended to the block chain. Network traffic was monitored in order to confirm the existence of alert messages on the network, and time stamps were logged on each client once a double-spend alert had been processed. This also generated a control data set in regards to double-spend detection and response time to which the results of the next two experiments were compared.

In our next two experiments, we removed ENHOBS functionality from all 3,000 clients and introduced special ENHOBS client nodes in addition to the 3,000 vanilla clients. These *skinny* ENHOBS clients included all of the ENHOBS functionality introduced in "All ENHOBS" but were stripped of all other non-essential functionality. In other words, the skinny ENHOBS clients would enter transactions into the memory pool and update the block chain as new blocks were received, but it would not perform hashing functions to the transactions in memory, nor would it send out any network traffic with the exception of alert messages. The first of these experiments, labeled as "1% Skinny," involved saturating the network with 30 skinny ENHOBS clients which is equal to 1% of the number of vanilla clients. The second experiment, labeled as "2% Skinny," involved saturating the network with 60 skinny ENHOBS clients which is equal to 2% of the number of vanilla clients.

It is worth noting that by removing all non-essential functionality in the skinny ENHOBS, they have unique, identifiable traffic patterns – alerts being the only outgoing traffic – that can be detected by an adversary who could attempt a denial-of-service attack to prevent the ENHOBS from alerting a potential victim of a double-spend attack. For this reason, our final two experiments add *fat* ENHOBS clients with no Bitcoin functionality stripped so they may blend into the network by broadcasting transactions and blocks as well as alerts. These experiments add 60 fat ENHOBS clients (2% of the total number vanilla clients) and 120 fat ENHOBS clients (4% of the total number of vanilla clients) to the network. These experiments are labeled as "2% Fat" and "4% Fat" respectively.

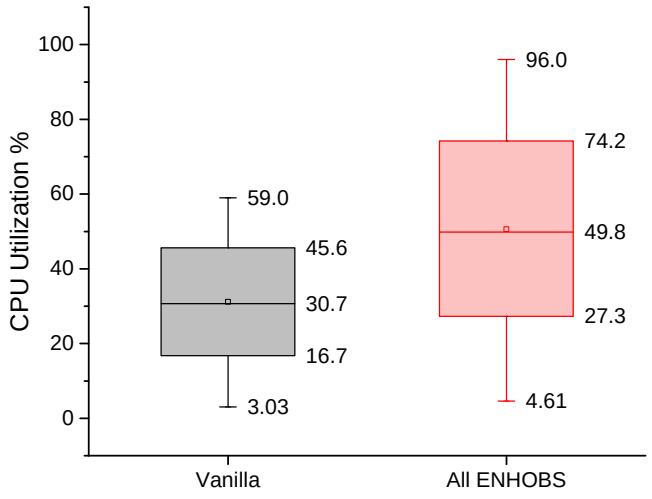

Figure 1. Client CPU utilization comparison

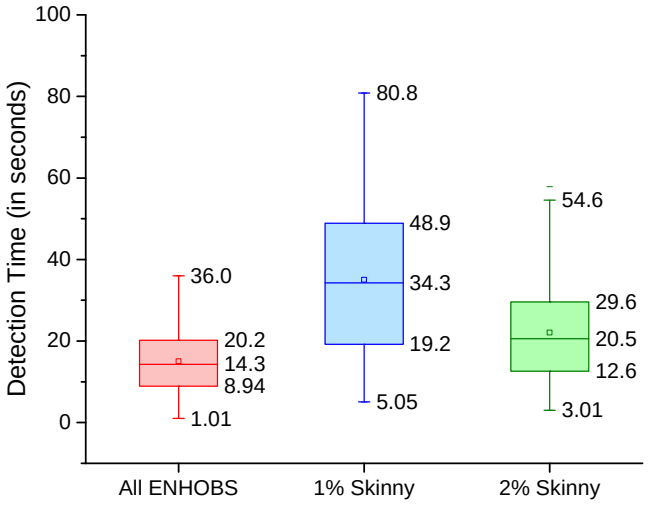

Figure 2. Time for clients to detect and process double-spends using skinny ENHOBS

6. Results

To assess the feasibility of introducing observer functionality in all clients, we evaluate the average CPU performance of all clients on the network before and after implementing the observer functions as well as the detection rate. Our implementation of observer functionality yields a detection rate of 100% on all clients either from direct detection or from receiving a double-spend alert message. In comparing the CPU utilization results between Experiments 0 and 1, depicted in Figure 1, our results show that adding the deep transaction inspection capabilities necessary for all Bitcoin clients to behave as observers increases the number of CPU cycles by an average of 63.1% causing the average CPU utilization to jump from 31.1% to 50.6%. Although the increase in CPU cycles was expected, these results have a negative impact on all clients - more specifically to any clients who don't conduct fast pay transactions.

In addition to CPU utilization, Experiment 1 yielded additional results depicted in Figure 2, which shows the distribution of the average time it takes for clients to detect a double-spend attack from the time the attack was introduced into the network either by direct detection or from receiving a double-spend alert from a peer. These results confirm our hypothesis that adding observer functionality can mitigate double-spending in a timely manner with the overall average response time being 15.03 seconds.

Considering the negative performance impact of implementing ENHOBS functionality on all clients, the results of Experiments 2 and 3, also depicted in Figure 2, show that introducing special clients with ENHOBS-only functionality into the network up to a certain saturation point is an efficient and effective solution. By introducing just 30 skinny ENHOBS into the network of 3,000 nodes, our results show that clients receive double-spend alerts in an average of 35.04 seconds and a median time of 34.32 seconds. If we double that number of skinny ENHOBS, we bring the average and median times for all clients to receive double-spend alerts down to 22.04 seconds and 20.48 seconds respectively.

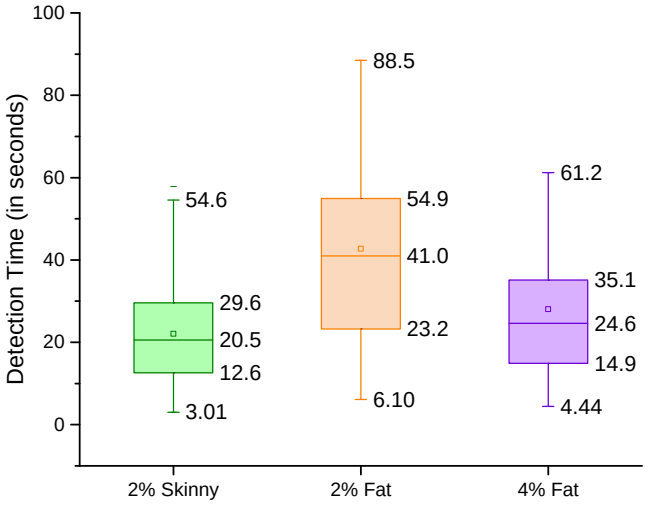

Figure 3. Time for clients to detect and process double-spends using fat ENHOBS

In order to evade detection by an adversary, ENHOBS must endure the additional performance requirements of the typical client, and the results of Experiments 4 and 5 – where fat ENHOBS are introduced into the network – are depicted in Figure 3. These results show it takes more than twice the number of fat ENHOBS than skinny ENHOBS to yield similar results. By introducing 60 fat ENHOBS into the network of 3,000 nodes, our results show that clients receive double-spend alerts in an average time of 42.76 seconds and a median time of 40.97 seconds. Doubling that number of fat ENHOBS brings the average and median times for all clients to receive double-spend alerts down to 27.87 seconds and 24.61 seconds respectively.

7. Discussion and Future Work

To implement our proposals successfully in the real Bitcoin network, we still need to address how to find a balance between the number of observer nodes, their associated performance (some may use more resources than others), and the number of alert subscribers so that it is beneficial and cost-effective for all parties. We can't ask Bitcoin to allow observers to mine BTCs upon detecting a double-spend the way everyone else does when hash farming without significantly altering the core functions of the source code. In addition to that, Bitcoin was designed to have a mining cap which will take place in a few years [9], meaning no more BTCs can be mined.

To preserve the core design of Bitcoin, we considered a solution similar to a mining community which involves creating a community of ENHOBS governed by an administrator that would require some type of Proof-of-Work (PoW) to ensure all ENHOBS are actually performing the work of looking for double-spend attacks. The administrator would use a subscription model where fast pay vendors (subscribers) pay the administrator a fair number of BTCs on a month-to-month basis which gets fairly distributed among the community given valid PoW. The amount would need to be high enough to justify the ENHOBS' cost of computing and low enough so that the cost of the subscription is not higher than the expected loss from double-spend attacks. This becomes easier to accomplish as more vendors subscribe to the community because as the number of subscribers increases to share the cost, the lower that subscription cost would be to each of them. Fast pay vendors could also add very small service fees to their transactions to fund the subscription.

As for the number of ENHOBS, it would be need to be subjected to a few controls. Adding more ENHOBS would increase the price of subscription and would offer marginally or negligibly better average response times as the number increases. The number of ENHOBS would also depend on network dynamics as nodes join or leave the system, so an appropriate number would need to be maintained.

Implementing the proof-of-work for ENHOBS could be done using a collective log verification system, in which logs are matched against the logs of a few other observers that were active during the same time frames. If only minor discrepancies are noticed, it can probably be accepted. If major discrepancies are found such as large blocks of transactions missing or missed double-spends, then the system could penalize or even blacklist the ENHOBS. We must also consider intentional sabotage similar to the block withholding attack [11], where a malicious node in the community could withhold any detection of a double-spend attack.

The subscription model, however, assumes that there is call for an ENHOBS market, which there isn't a cry for. As such, we believe that a small number of altruistic parties can ensure the health of the ENHOBS system by absorbing the costs while the vendors and other Bitcoin clients pay nothing. Some positive externalities could be captured by large Bitcoin players as well similar to Google as they invest in the overall health of the web. It works for Tor given that there is no profit in hosting relay nodes [16].

8. Related Works

Karame et al. proposed three of the countermeasures we considered in two papers [3], [4]. They backed up their proposals with sound theory and accurate mathematical equations. Their attack model was also very generalized in scope so that it didn't require very specific conditions to succeed. Their analyses of the inner workings of Bitcoin as well as user practices allowed them to model an attack that challenges some of the founding principles and claims of Bitcoin.

Bamert et al. presented the other two countermeasures considered in this paper to address the double-spending attack in fast pay transactions. They claim that these countermeasures could diminish the success of such attacks to less than 0.09% [5] by having the vendor connect to a large random sample of nodes in the Bitcoin network. The vendor then requests transaction data from each of them while denying incoming connections, thereby preventing an attacker from sending a fraudulent transaction to the vendor. This was only proposed in theory, and our analysis and experiments have expanded on this.

Andrychowicz et al. address the malleability problem present in Bitcoin transactions in [6]. They pose an experiment that demonstrates a high success rate for the attacker, and it is also the officially-given reason why MtGox suspended trading in February 2014.

Decker and Wattenhofer analyze Bitcoin's multi-hop broadcasting used to propagate transactions and blocks and update the ledgers [12]. Their analysis verifies the assumption that block chain forks are primarily caused by propagation delay in the network and are the main reason for ledger inconsistencies. They further go on to experiment and show that block chain forks can be avoided.

Meirs et al. propose an extension to Bitcoin called Zerocoin that will allow full anonymity in Bitcoin transactions through standard cryptographic assumptions without introducing trusted third parties [7]. Zcash is the real-world deployment of this idea, so it is interesting to consider whether and how it will be as effective in the marketplace as Bitcoin has been, as well as if there are unforeseen security risks in it as well.

The double-spend attack is possible largely due to lack of anonymity in practice. Many fast pay vendors publicly link their pseudonyms to their identities which give the attacker the ability to social engineer information relevant to the vendor's Bitcoin node. With this information, the attacker can attempt to make direct connections which are crucial to successful double-spend attacks. In two works, Biryukov et al. challenge the anonymity of Bitcoin users - even over the Tor network. They first found an effective means to de-anonymize users that links pseudonyms to the source IP addresses of transactions [17]. They then showed how using Bitcoin over Tor actually creates an attack vector

for man-in-the-middle attacks, granting the attacker full control of information flows between all users running Bitcoin over Tor [18]. Their proposed countermeasures, however, offer little detail.

The true anonymity of Bitcoin users has been called into question repeatedly due to the fact that all transactions are publicly known and can be analyzed. Möser states that users don't have anonymity but rather "pseudonymity" [19]. Möser also concluded that not all digital wallets provide the same level of anonymity, and many Bitcoin exchanges link personal information to Bitcoin addresses. Through transaction graphs, Möser measured the difficulty to find direct connections relating input and output in a transaction. However, Möser's work only analyzed three services out of dozens of possibilities.

Likewise, Ober, et al. found that by merging information regarding public Bitcoin addresses along with simultaneous usage generated concern that anonymity could be compromised in Bitcoin [13]. On the other hand, they found dynamic effects that increase anonymity.

9. Conclusion

Because Bitcoin's design only guarantees confirmation of a transaction after it has been hashed into the block chain [2], it is currently better suited for slow-pay transactions, as fast-pay vendors are vulnerable to double-spend attacks.

In this paper, we analyzed representative proposed countermeasures and have shown that an attacker still has options that allow it to circumvent these countermeasures. Based on this analysis, we proposed *enhanced observers* (ENHOBS), a hybrid approach that combines extends key ideas from prior work. We simulated ENHOBS and demonstrated that we can effectively counter the double-spend attack in fast pay transactions with reasonable overheads in terms of the number of ENHOBS in the network.

Vulnerabilities to attack can significantly hinder the growth of any system, and Bitcoin is no different. Bitcoin's vulnerability to double-spending has likely had a negative effect on its ability to acquire new vendor clients. However, the system-wide implementation of these countermeasures could potentially open the doors to a much larger network of vendors that handle large-value point-of-sale transactions such as supermarkets, gas stations, and even retail stores.

Acknowledgments

The authors would like to thank the reviewers for providing great feedback. We thank the University of Texas at Arlington and the Department of Education for supporting us with a Graduate Assistance in Areas of National Need (GAANN) fellowship. This work was also supported in part by NSF awards number CNS-1423163 and CNS-0954133 as well as Rochester Institute of Technology and the Signature Interdisciplinary Research Areas grant.

References

[1] "Blockchain info," https://blockchain.info/, accessed: 2017-01-01.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008, accessed: 2017-01-01.

[3] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending attacks on fast payments in bitcoin," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS'12)*.

[4] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Capkun, "Misbehavior in bitcoin: A study of double-spending and accountability," *ACM Transactions on Information and System Security (TISSEC)*, vol. 18, no. 1, June 2015.

[5] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welton, "Have a snack, pay with bitcoins," in *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing*.

[6] M. Andrychowicz, S. Dziembowski, D. Malinowski, and ukasz Mazurek, "On the malleability of bitcoin transactions," in *Proceedings of the 2015 Conference on Financial Cryptography and Data Security (FC 2015)*.

[7] I. Meirs, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (S&P'13)*.

[8] R. Jansen and N. Hopper, "Shadow: Running tor in a box for accurate and efficient experimentation," in *Proceedings of the 2012 Network and Distributed System Security Symposium (NDSS'12)*.

[9] "Bitcoin wiki," https://en.bitcoin.it/wiki/, accessed: 2017-01-01.

[10] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS'15)*.

[11] S. Bag, S. Ruj, and K. Sakurai, "Bitcoin block withholding attack : Analysis and mitigation," *IEEE Transactions on Information Forensics and Security*, vol. PP, no. 99, pp. 1–12, November 2016.

[12] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing*.

[13] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," *Future Internet*, vol. 5, no. 2, pp. 237–250, May 2013.

[14] H. Finney, "The finney attack," https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384.

[15] "Shadow plugin: Bitcoin," https://github.com/shadow/shadow-plugin-bitcoin, accessed: 2017-01-01.

[16] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Identifying proxy nodes in a tor anonymization circuit," in *Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems (SITIS '08)*.

[17] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymization of clients in bitcoin p2p network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14)*.

[18] A. Biryukov and I. Pustogarov, "Bitcoin over tor isn't a good idea," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy (S&P'15)*.

[19] M. Möser, R. Böhme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *Proceedings of the 2013 IEEE eCrime Researchers Summit*.