Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Exploiting Bitcoin's Topology for Double-spend Attacks

Bachelor Thesis

Matthias Lei

mlei@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Christian Decker
Prof. Dr. Roger Wattenhofer

August 13, 2015

Acknowledgements

I would like to thank my supervisor Christian Decker for his assistance and guidance during my bachelor thesis as well as Ingrid Giel, Andreas Enz and Daniel Yu for proofreading and reviewing my report.

Abstract

As a decentralized digital currency, one of the features Bitcoin draws its security from is that its clients broadcast transaction or block messages to a number of randomly selected peers to share their view of the current state of the transaction history. This feature makes it very hard for an attacker to manipulate the victim's view of the network for fraud.

In this thesis we show that the Bitcoin network leaks information about the topology. For this purpose we introduce BiPI, a program which reveals the peers of a Bitcoin Core client with high probability using timestamps. We measure the efficiency of BiPI before and after the release of the patch targeting this leak. To show how powerful this knowledge can be we then execute double-spend attacks with and without the identified peers and quantify the increase of the success probability.

Contents

Chapter 1

Introduction

What started 2008 with Satoshi Nakamoto's White Paper [1] has now grown to the most prominent digital currency. Bitcoin has an estimated market capitalisation of around 4 billion USD, as of July 2015, and more than 7.5 million digital wallets.

Bitcoin is used all over the globe. Online as well as regular shops accept Bitcoin as a payment method, large exchange platforms for Bitcoin such as Coinbase or Bitstamp exist, and Bitcoin ATMs have been deployed in many countries.[1] Some of these uses rely on fast payment, meaning the service offered must be delivered seconds after the transaction has been made. The merchant cannot wait for the transaction to be confirmed as this may take several minutes. This scenario is prone to the double-spend attack in which an attacker can use the same coins for two or more payments. However our measurements show that the success probability of a plain double-spend attack lies only around 12%. This kind of double-spend attack makes no assumption about the topology. Therefore the dissemination behaviour of transactions cannot be controlled.

Our hypothesis states that it is indeed possible to obtain insights about which peers a victim is connected to and, based on this advantage, that an adversary can boost his success probability of a double-spend to 60%. To prove this we developed the Bitcoin Peer Investigator (BiPI), a program which exploits an information leak in the gossiping mechanism for address dissemination.

A variety of clients implement the Bitcoin protocol and enable users to store and trade with Bitcoins. For the course of this thesis we concentrate on the most common one: Bitcoin Core. We use this client as an example of the protocol, but similar results can be achieved with other clients as we do not exploit specific properties of this client. Instead we only rely on the Bitcoin protocol.

[1] www.coindesk.com/bitcoin-atm-map

1.1 Transaction

Bitcoins are transferred from one Bitcoin *address* to another through *transactions*. A Bitcoin address represents a destination for payments and is a hash from the public portion of a public/private keypair. Each transaction contains at least one *input* and one *output* as well as the number of Bitcoins transferred to each output. An input is a reference to an output of a previous transaction, thus creating a directed acyclic graph. The transaction graph is public knowledge and everyone is able to track and verify the complete transaction history since the beginning of Bitcoin. An output references a Bitcoin address and contains the amount of Bitcoins transferred.

Bitcoin ensures that only the owner of the address can claim the output by using digital signatures. When a user wants to send Bitcoins to an address, she creates a transaction specifying the inputs and outputs and signs it with her private key before she broadcasts it to the network. Every receiving node checks whether the transaction is valid and correct and if so the transaction is further relayed to the node's peers. In particular every node checks whether the output claimed has not been used before. Bitcoin Core will only relay the first transaction, but any subsequent transaction claiming the same output will not be relayed. This way the transaction is spread through the whole network reaching clients and eventually the recipient.

In a fast payment scenario the transaction is considered to be confirmed for the merchant, which also runs a Bitcoin client, upon reception. Contrary to that in a non-fast payment scenario the merchant waits for the confirmation of the network.

1.2 Blockchain

The *blockchain* is Bitcoin's public ledger. It is Bitcoin's approach of eventually reaching consensus in its peer-to-peer network about, which transactions effectively happened and in which order. Every node in the network possesses its own blockchain, which denotes its own view of the transaction history. These views should not differ too much from node to node.

The blockchain consists, in the best case, of a single sequence of *blocks*, starting from the first genesis block all the way to the current block (Fig. 1.1). Every block carries a timestamp, a hash to a previous block, a collection of transactions and a nonce. Following the sequence from the first to the last block is equivalent to reviewing all the transactions ever made in chronological order. Therefore a transaction contained in a block must have its inputs point to transactions in previous blocks or in the same block. Creating one of these blocks is based on a proof-of-work (PoW) system and requires a significant amount of computational

power. By appending the right nonce to the binary representation of the payload mentioned above, the value of a given cryptographic hash function must lie below a target value in order to be accepted by the network. This requirement can only be met by brute-forcing. The target value is adjusted dynamically so that in average such a block is found roughly every ten minutes. Nevertheless studies show this duration has a standard deviation of 15 minutes [2]. Some nodes, called miners, in the network are solely dedicated to finding such blocks. They listen to newly announced transactions in the network, validate them and keep them in a pool. For every block found the miner is rewarded with a certain amount of Bitcoins plus the sum of all transaction fees from the transactions included in the block. As soon as a block is found the miner broadcasts it into the network and the receiving nodes can check for validity by computing the hash of the block and verifying all the transaction contained. If valid, the nodes append their block to their own blockchain. Note that modifying the block, and thus claiming the reward for the own sake, is equally as hard as mining a new block because every modification to a block results in a change of the hash value.

It can happen that during the propagation of a newly mined block another block is found by another miner, which has not yet heard of the new block, leading to a split of the network and ultimately to two different views of the transaction history [3]. This is called a fork. In order to resolve it both partitions enter a race about which one finds the next block and creates a longer chain, which is accepted by both partitions. The race can potentially go on for multiple blocks but the probability of it decreases exponentially for every additional block. However if a miner happens to create a longer chain originating from a non-current block, thus creating another history, the rest of the network has to rollback its history to the point where the fork happened. This requires theoretically a majority of the mining power of the Bitcoin network [1].

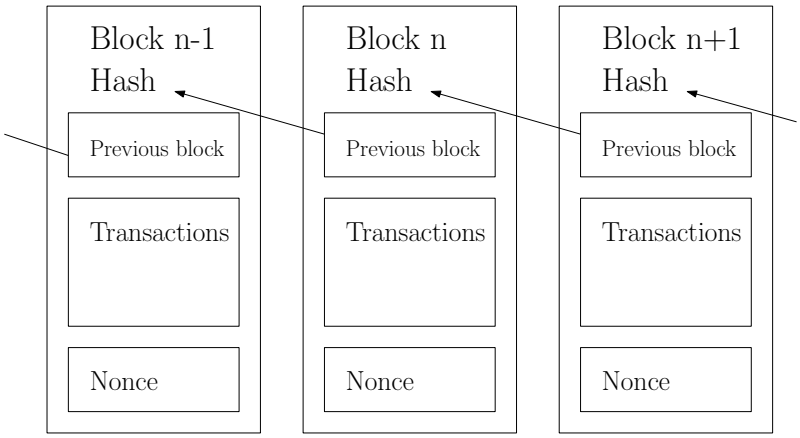

Figure 1.1: Example of a blockchain segment

1.3 Double-spend attack

The double-spend attack tries to spend the same Bitcoins more than once. Upon a successful attack the victim is left with an invalidated payment while having already delivered the service. Multiple variants of the double-spend attack exist. The one we consider in the thesis, the race attack, does only work for fast payment scenarios, e.g., ATMs, cafes or fast food chains.

A merchant using fast payment has to accept the payment once it receives the respective transaction and cannot afford to wait for the confirmation from the blockchain. Her transaction, however, can be rendered invalid afterwards, if another transaction claiming the same output is included in the blockchain and thus going to be accepted by the network. An attacker performing a double-spend injects two transactions into the network. A transaction seemingly paying for the service claimed and a transaction for the miners. The one for the miners can pay for another service or can transfer the Bitcoins back to another address owned by the attacker.

1.4 Related work

Since double-spend attacks still remain a problem in the Bitcoin protocol it has been topic of some research which study the difficulty of executing such an attack [2, 4, 5]. Also other types of related attacks has received considerable attention such as the eclipse attack which is much more powerful once set up but requires a tremendous effort to do so [6]. This attack goes much further than just knowing the peers of the victim. It tries to monopolize all connections of the victim. This enables the attacker to temporarily alter the victim's view of the network to her advantage.

While both of these methods target the victim directly, a successful 51% attack affects multiple clients at once. This attack was mentioned as a hypothetical security issue in Satoshi Nakamoto's paper [1]. As long as an attacker possesses the majority of CPU power in the network she could change transaction history. However, the research of Eyal and Sirer has shown that this statement does not hold. Even with less mining power than the majority it is worth mining selfish [7]. A much more general discussed aspect is the topology itself. Andrew Miller et al. present a method to find influential nodes and mining pools, which in return could be used to improve our results even further [8].

In conjunction with Bitcoin being a peer-to-peer network, the degree of anonymity is another widely discussed topic [9]. Ober et al. analyzes which dynamical effects increase or decrease anonymity [10]. A method proposed by Koshy et al. maps Bitcoin addresses to IP addresses [11]. Their method analyzes real-time transaction data collected over 5 months and tries to detect relay

patterns in the transaction graph.

Chapter 2

Revealing peers

In this chapter we introduce our program BiPI (Bitcoin Peer Investigator) to showcase that it is possible to identify the victim's peers. The goal of BiPI is to return a result set of peers which covers the victim's peers with high probability by observing the updates of timestamps in the victim's address pool.

Every node manages a pool of IP addresses which it can resort to if a connection to another Bitcoin peer is lost. In the case of Bitcoin Core, 8 of these connections are maintained. Those peers are used to broadcast and relay data messages, such as addresses, blocks or transactions. Each address carries a timestamp from the last time when the address was heard from. On startup, the client performs a DNS lookup in order to obtain a list of IP addresses believed to be active. The domains of the DNS servers are hardcoded into the Bitcoin Core client. It then connects itself to those peers and starts exchanging information about their view of the network. One part of this process involves updating the address pool using *addr* messages. Those messages are sent either as a response to a *getaddr* request or unsolicited. An *addr* message contains a random selection of addresses from the sender's pool, each carrying a timestamp of the most recent appearance in the network. Those addresses are updated under two conditions:

1. The address was sent as a part of an *addr* message and carries a more recent timestamp than the timestamp stored in the pool.
2. When a connected peer sends a *addr*, *version*, *inv*, *getdata* or a *ping* message the timestamp of its address is then updated in the pool of the receiving node.

Note that this behaviour only applies to Bitcoin Core version 0.10.0 and older. Condition 2 has been changed in version 0.10.1 which was released during the writing of this paper and we will address that in chapter 2.3.

The key idea is that the addresses which the victim is connected to are updated much more frequently because of condition 2. Therefore repeatedly

requesting addresses through *getaddr* messages first leads to a good coverage of the victim's address pool and second allows an estimate of the update frequency of her addresses.

For the course of this thesis we introduce two terms:

- Those peers which the victim is connected to are called **true peers**.
- Peers which the attacker believes to be potential true peers are called **positive peers**.

By default Bitcoin Core runs 8 of these outgoing connections, meaning the victim actively establishes the connection to those peers, in contrast to incoming connections which do not fall under the definition of true peers and are passively accepted by the victim. The set of positive peers should be larger than the set of true peers in order to mitigate the chance of having false negatives.

2.1 Implementation

BiPI is a Python program which operates in the Bitcoin network by using the data messages specified in the Bitcoin protocol. In particular it only relies on the messages belonging to the gossiping of addresses. BiPI only tries to identify the peers of a victim and will not perform double-spends. The victim's IP address has to be known in advance and be given as an input for BiPI. On startup it connects itself to the victim and attempts to take snapshots of the address pool at different points in time. The assumption is that each of those true peers' addresses has a more recent timestamp due to being updated more frequently. Together with other snapshots a score can be constructed about which addresses are the most likely ones to be the true peers. However there is no direct way to take a full snapshot from the victim's address pool. Instead different methods exist which approximate snapshots and thus produce an estimate of the update frequency. The whole procedure should not take up too much time as the peers of Bitcoin Core change from time to time and thus disturb the measurements.

BiPI approximates one snapshot by sending a burst of *getaddr* requests for having a meaningful coverage of the pool. We defined a snapshot to be 10 successive *getaddr* messages in a 2 second interval. Between every snapshot the program waits for 10 seconds until a total of 5 snapshots are taken. In the evaluation phase the 10 most recent addresses from every snapshot are collected into an intermediate set of (address, timestamp) tuples. These tuples are then aggregated into buckets, where each bucket is associated with an address. In a final step we sort the addresses by their buckets size in descending order and return the list. Hence the first address in the list is the likeliest one to be a true peer. Contrary, the last address is the likeliest to be a false positive. The

parameters chosen turned out to be producing the best results while keeping the running time at 3 minutes.

One alternative to the implemented method is to alter the sorting criterion. Instead of just sorting by the most recent timestamp one could also assign a high weight to multiple occurrences of the same address but with a different timestamp within a snapshot. Multiple occurrences of the same address are the result of overlapping *addr* packets due to the randomness of their content. This means that during the relatively short time of a snapshot the address has been updated which is a strong indicator for a true peer.

2.2 Evaluation

In order to test BiPI we set up a controlled environment. As a measure of success we use the *coverage* defined as the ratio between the numbers of peers correctly identified as true positive peers and the total number of true peers.

$$coverage := \frac{|true\ peers| \cap |positive\ peers|}{|true\ peers|}$$

Bitcoin Core represents 97% of all clients used in the network, so we choose this as the victim in our measurements.[1] We ran Bitcoin Core on a local machine with default settings, i.e., allowing incoming connections and randomly establishing 8 outgoing connections. Using `netstat` we gather the outgoing connections, i.e., the ground truth. BiPI was tested with three different versions of Bitcoin Core which were released during writing, namely 0.9.3, 0.10.0 and 0.10.1. All three versions brought changes to the peer-to-peer behaviour and these become apparent in the results. Fig. 2.1 represent the average of 120 test instances for every version.

The first plot illustrates the results under version 0.9.3 showing that the gossip of addresses indeed leaks additional information about the connected peers. Even looking only at the top 16 positive addresses yields a coverage of 93% and looking at the top 32 addresses the coverage reaches 97%.

Interestingly version 0.10.0 introduced a change to the gossiping behaviour which improved the success rate of the method developped for version 0.9.3. Clients running this version would not include obviously poor addresses in an *addr* message, e.g., addresses having an old timestamp or addresses which have had 10 successive failures in a week, etc. For comparison, an *addr* response under 0.9.3 usually contained 1000 addresses, whereas under version 0.10.0 it contains only around 300 addresses. The true peers are by definition never filtered out

[1] getaddr.bitnodes.io/api/v1/snapshots/1437935137

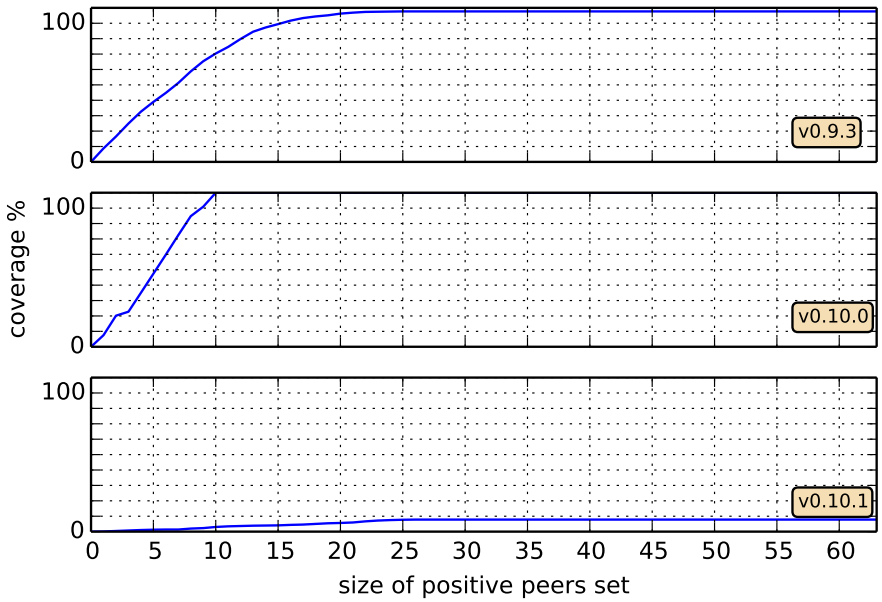

Figure 2.1: Average coverage for all three versions

and consequently the probability of including false positives is reduced. The second plot in Fig. 2.1 shows that it would be sufficient to only consider the top 10 addresses in order to achieve a 100% coverage.

This leak has been reported to the Bitcoin Core developers and a fix was released in version 0.10.1. The patch contained a change to the update condition 2 which rendered BiPI ineffective. As long as a the client maintains a connection to the peer, the peer's timestamp is not updated anymore. Only a connection establishment and a disconnect triggers an update. Hence the update frequency cannot be derived from the timestamps anymore. A timestamp of a connected node has now roughly the same update frequency as an unconnected one and sorting the addresses by timestamp yield which addresses have been included the most in *addr* packets because of condition 1. Note that the alternative methods proposed in chapter 2.1 would also not work as they rely on the update frequency. The third plot illustrates the effect: the coverage does not exceed 8% for the first 64 addresses.

Chapter 3

Double-spend with revealed peers

Now that we have found a method to reveal the peers we can start inspecting the security issues this leak causes. The double-spend attack serves as an example of an attack which benefits from additional knowledge about the topology of the network. In this chapter we quantify the benefit by examining the success chance of the attack without further insight about the topology compared to the case with insight. Referring to chapter 1.3 about how the double-spend attack works we define the two transactions:

1. $\boldsymbol{Tx_v}$ denotes the transaction which seemingly confirms the payment for the service claimed, but is then invalidated.
2. $\boldsymbol{Tx_a}$ denotes the transaction which the attacker broadcasts to the other peers and is included in the blockchain in the end.

For a double-spend to be successful two requirements have to be met (Fig. 3.2):

1. The victim should not hear from Tx_a until the service has been delivered and only receive Tx_v.
2. Tx_a has to be confirmed by the blockchain.

The first requirement implies that the victim has a local detection mechanism whether there is a conflicting transaction or not. Bitcoin Core shows conflicting transactions (Fig. 3.1), so the victim is able to abort before delivery if suspicion arises. But eventually she is going to detect the double-spend, namely when her blockchain is updated. The case, where the victim hears from Tx_a, after delivery, by a peer which relays the double-spend is very unlikely. This is due to the short transaction propagation time which lies within the acceptable range of waiting time for fast payments. The 50th percentile of a transaction to be propagated through the Bitcoin network lies currently around 1 second.[1] The

[1] bitcoinstats.com/network/propagation

second requirement ultimately invalidates Tx_v and sends the Bitcoins to the output of Tx_a.

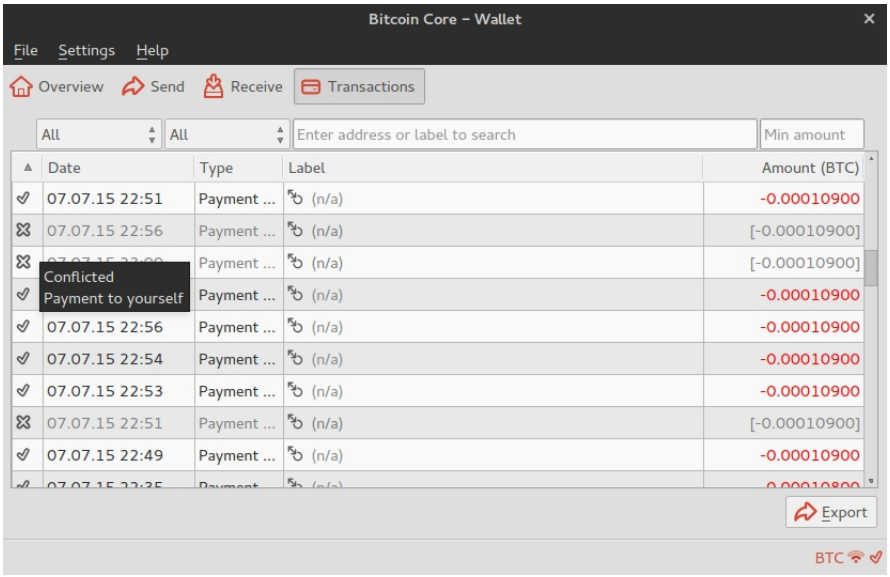

Figure 3.1: Bitcoin Core shows conflicting payments

In the traditional case, where the topology is unknown, the attacker must choose the recipients of these two transactions randomly. This makes it impossible to have some control over the success chance of any of the two requirements. The victim's peers could sit among the recipients of Tx_a as wells as the miners could sit among the recipients of Tx_v.

Once the true peers are known we can influence the success chance of both requirements. Under the assumption that the nodes do not relay transactions claiming an already used output, which is true for Bitcoin Core, we can create a temporarily isolated view for the victim from the rest of network. She believes that her payment is going to be confirmed while the attacker announces Tx_a to a large set of *other peers*. The larger this set is the more likely it is that the mining pools receive and process Tx_a and therefore meeting second requirement. It is important to point out that sending Tx_a and Tx_v at the same time is an issue because of the network latency. In some cases Tx_a would still reach the victim faster over two hops than Tx_v in only one hop. Therefore it is necessary to implement a delay between sending out Tx_v and Tx_a, with Tx_v being first, in the hope to sufficiently isolate the victim from the dissemination of Tx_a. But choosing the delay too large results in Tx_v being included in the blockchain.

Note that in order to avoid congestion Bitcoin nodes do not directly relay full *tx* messages to the peers upon reception. Instead the node first announces the availability of a new *tx* packet by broadcasting the hash of the packet. Only if a peer does not possess this hash the full *tx* packet is requested by issuing a

getdata request containing the hash.

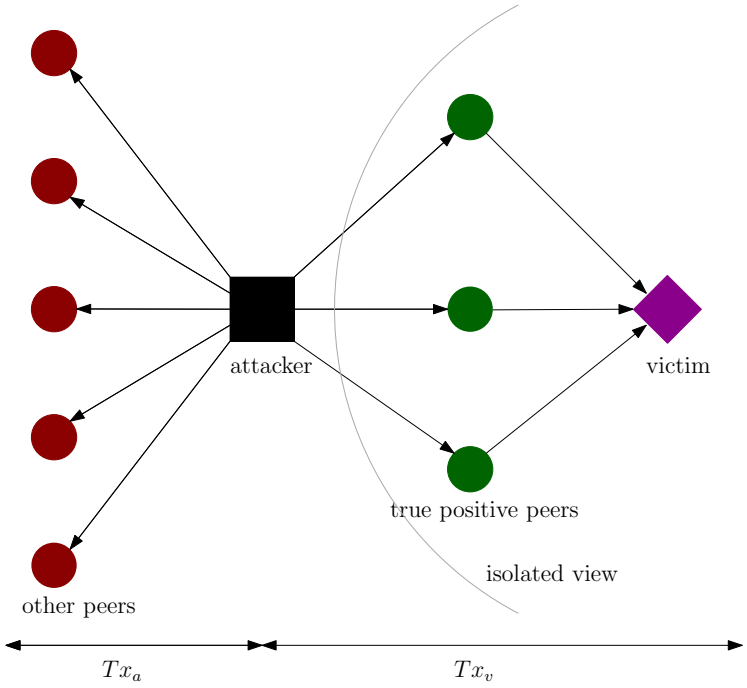

Figure 3.2: Attack scenario

3.1 Evaluation

For the attack scenario we implemented on the victim's side a Python dummy client, which mimics the network behaviour of Bitcoin Core, including establishing 8 outgoing connections and requesting the transactions announced in *inv* messages. The dummy client enables us to log all incoming messages for further evaluation which would not be possible with Bitcoin Core. On the attacker's side we also implemented a client with Python. The attacker is able to connect to other Bitcoin clients as well as to create, announce and send transactions. All Bitcoin addresses used in transactions throughout the thesis are owned by us. No third-party Bitcoin address was involved nor defrauded.

In this test environment we assumed the attacker ran BiPI from chapter 2 with a coverage of 100% having 16 positive peers, i.e., 8 false positives and 8 true positives. Furthermore, she chooses 128 other peers uniformly at random from *getaddr.bitnodes.io/api/v1/snapshots*. There is a trade-off in choosing the set size of positive peers. The size has to be sufficiently large in order to guarantee high coverage, but choosing the set too large increases the chance that Tx_v reaches some mining nodes before Tx_a and possibly resulting in Tx_v being confirmed.

We set the size to 16 which yields a 93% coverage. The set size of the other peers was chosen to be large relative to the positive peers set in order to increase the chance of Tx_a being confirmed.

The double-spend scenario is set up as follows (Fig. 3.6):

1. Attacker and victim connect to the their respective peers.
2. Attacker announces the availability of both transactions to the respective peers simultaneously in order to prime both groups by sending *inv* messages.
3. Attacker waits 5 seconds for incoming *getdata* request.
4. Attacker sends Tx_v to the positive peers.
5. After waiting a given delay the attacker sends Tx_a to the other peers.
6. Victim listens for messages for another 60 seconds before the test run is terminated.

After waiting for at least one hour we check which transaction has been confirmed by the blockchain by querying the local Bitcoin Core client and examine the log files to see whether the victim has heard of Tx_a. We ran this setup with several different delays ranging from 0.0 seconds to 0.5 seconds with 0.025 as step-size. In order to measure the improvement we also set up a base scenario as reference. In the base scenario we assume that the attacker has not run BiPI and broadcasts Tx_v to 16 randomly chosen peers. Every other parameter remains the same.

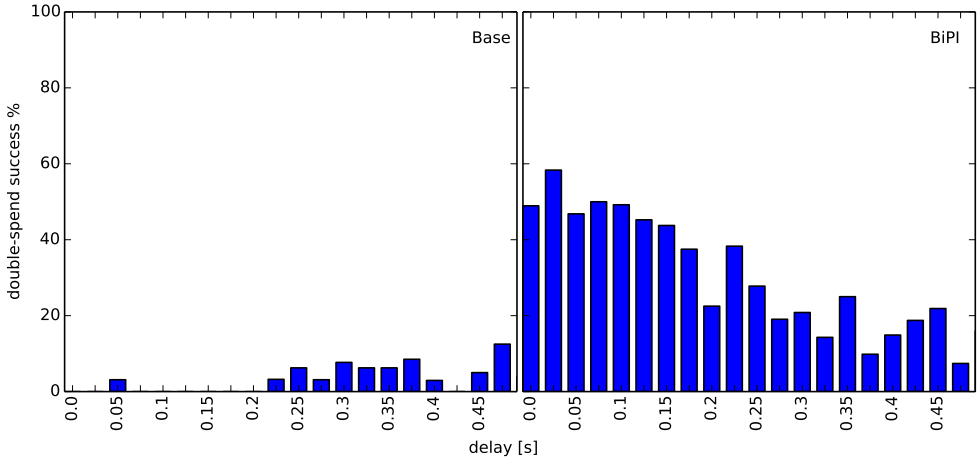

Figure 3.3: Success of double-spend attacks

3.1.1 Base scenario

As expected the base scenario has only a 12% success rate with 0.475 seconds delay (Fig. 3.3). The reason for failing lies in the early detection of Tx_a at the victim's side as both transactions are broadcast to a random set of peers (Fig. 3.5). While Tx_a has a much larger number of recipients than Tx_v it is clear that the victim has a very high probability of hearing from the wrong transaction even though Tx_v is given a headstart. On the contrary the probability which transaction is ultimately going to be confirmed does not depend on the different choice of the peers. Fig. 3.4 therefore shows almost the same distribution.

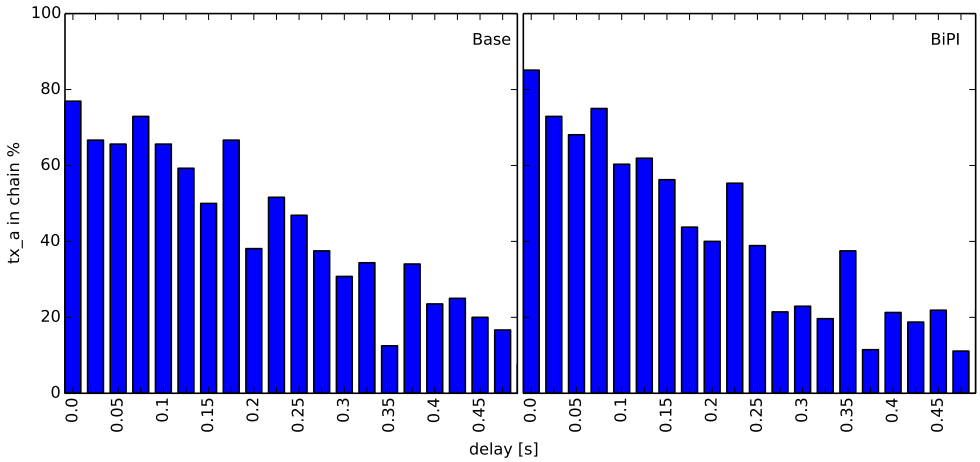

Figure 3.4: Percentage of runs where blockchain confirmed Tx_a

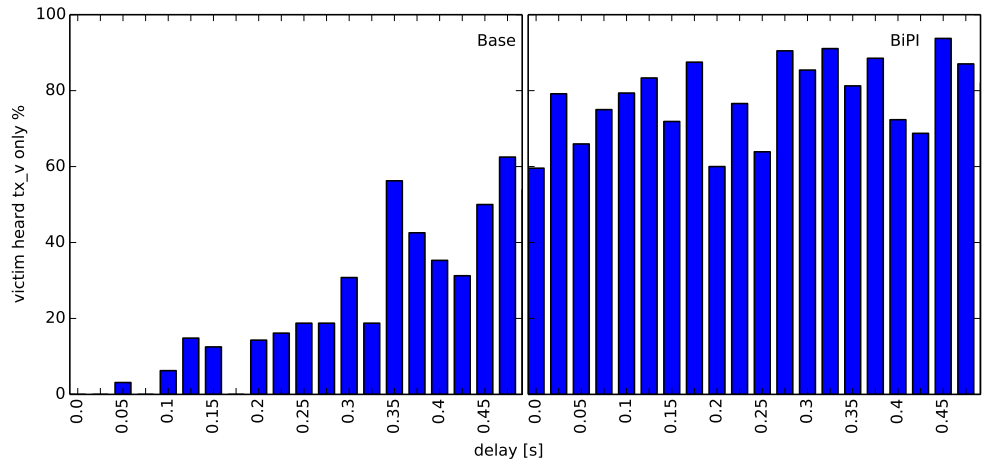

Figure 3.5: Percentage of runs where victim only heard Tx_v

3.1.2 BiPI scenario

As soon as the set of positive peers covers the true peers the chance of success increases to almost 60% with a delay of 0.025 seconds (Fig. 3.3). Surprisingly the delay has less influence on the detection chance than we expected (Fig. 3.5). A major reason is the diversity of clients for Bitcoin, some of which break the assumed isolation of the victim. Most of theses cases are attributed to a client named Bitcoin XT which is a patch on top of Bitcoin Core. It relays the first double-spend to its neighbours which acts as an alert signal for an attack. Further transactions conflicting with the same output, however, will not be relayed in order to avoid congestion in the network.

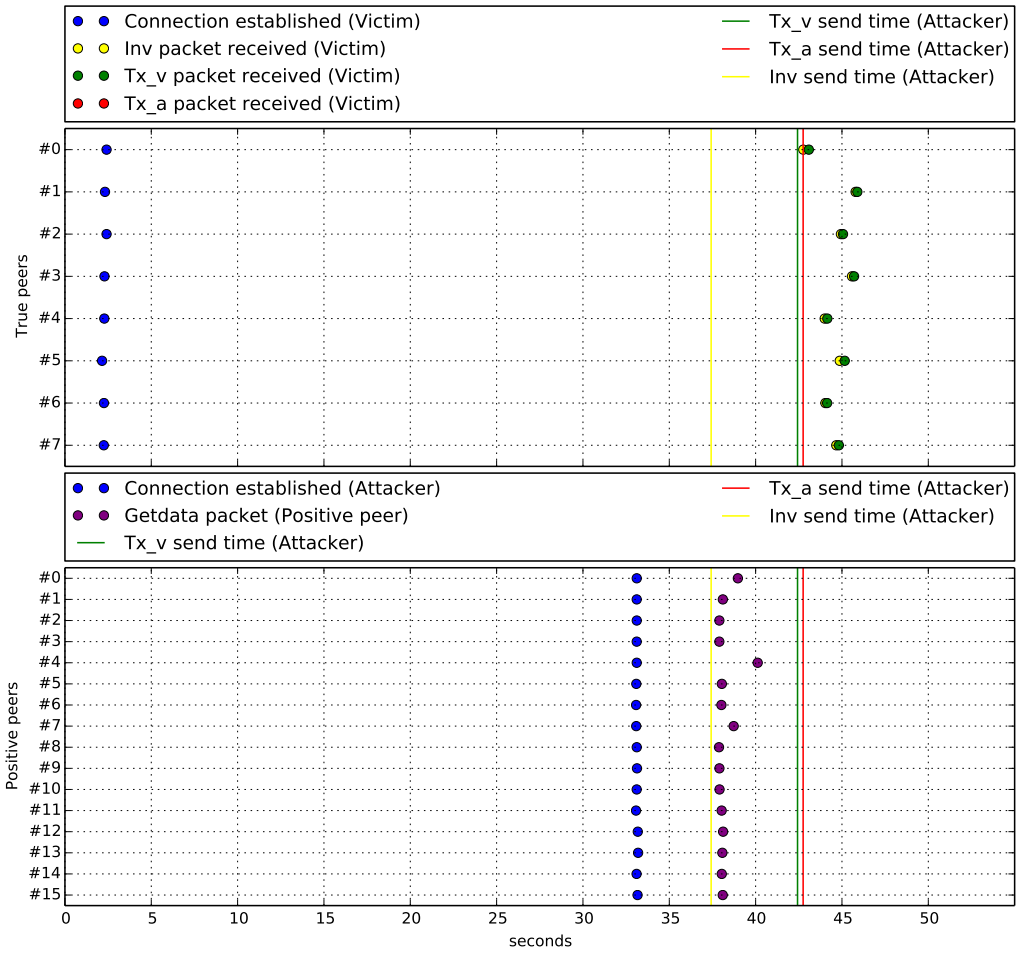

Figure 3.6: Example history of a BiPI scenario where victim does not receive Tx_a (0.325s delay)

Chapter 4

Conclusion

We have shown that it is indeed possible to gain significant advantage for double-spends if the topology is known beforehand. This knowledge directly translates into an increase from 12% to 60% success probability. Furthermore, we developed a program which is able to find the peers very reliably using timestamps in *addr* packages. However Bitcoin's recent update to the behaviour of timestamps under version 0.10.1 proved to be successful as it leaves no fingerprints of the currently connected peers on a client anymore. But if an attacker manages to identify the true peers on an alternative way, which can involve methods that operate beyond the Bitcoin protocol, then she is able to achieve the performance shown in chapter 3.

The only recommendation Bitcoin gives to the merchants is only offering fast payment for products not of high value. Nonetheless, because fast payment is very common this recommendation is a restriction to an otherwise promising technology, so the Bitcoin community developed detection and protection mechanisms against double-spends. But yet none of them can guarantee absolute security or has gained enough recognition.

Bibliography

[1] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf

[2] Karame, G., Androulaki, E., Capkun, S.: Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. IACR Cryptology ePrint Archive (2012)

[3] Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: Peer-to-Peer Computing (P2P), Conference on IEEE. (2013)

[4] Rosenfeld, M.: Analysis of hashrate-based double spending. arXiv preprint arXiv:1402.2009 (2014)

[5] Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., Welten, S.: Have a snack, pay with bitcoins. In: Peer-to-Peer Computing (P2P),Conference on IEEE. (2013)

[6] Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium (USENIX Security 15), Washington, D.C., USENIX Association (August 2015)

[7] Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography and Data Security. (2014)

[8] Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoin's public topology and influential nodes

[9] Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. (2013)

[10] Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future internet (2013)

[11] Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using p2p network traffic. Springer (2014)