

August 2, 2021

# 1 Introduction to Bitcoin

Bitcoin address = hash of the public key = receiving address

miners bundle transactions in blocks

blocks are advertised in order to claim a mining reward

blockchain = distributed public ledger

coins are represented only in form of transactions

UTXO unspent transaction output

keep global view of the blockchain consistent

pow + probabilistic distributed consensus protocol

the correct state of the blockchain is obtained by election

to verify transactions miners have to perform a computational task

pow puzzle in sha 256, difficulty exponential on the number of zeros

transaction verification is  $O(1)$

mine a block = hash the txs in the block + varying nonce = solve computational task

reward generating transaction + transaction fees

target value( for the mining difficulty adjustment) changed every 2016 blocks, keeps block verification time around 10 minutes

transactions are stored using a merkle tree

tempering a block is impossible

natural probabilistic blockchain forks and how they solve by themselves

mining pools

A creates a transaction with a BTC transfer to B, it advertises the transaction to the miners by broadcasting it on the network, miners include the transaction in a block and when they mine it the transaction will belong to the blockchain and will be verified after some time.

participants require no authentication (permissionless) to join the network.

bitcoin is a cryptocurrency

## 1.1 An introductory example to Bitcoin usage

A basic usage example, shown in Fig. 1, is presented here to get the reader started with Bitcoin.

Let Alice and Bob be two Bitcoin users with a third-party wallet software. Each wallet is associated with a pair ECDSA keys used to identify users. The private key is used to

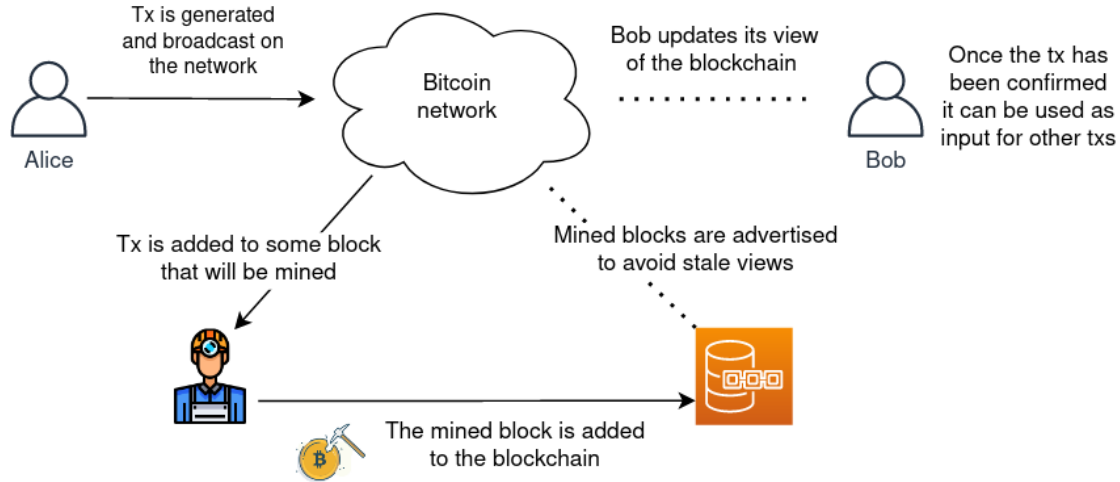


Fig. 1: Bitcoin introductory use case

sign issued transactions; instead, hashing the public key will produce the Bitcoin address on which Bob can receive Bitcoins.

Suppose that Alice wants to transfer some amount of Bitcoins to Bob. Alice will generate a transaction in which she uses some set of unspent Bitcoins. The generated transaction will then be broadcast to the bitcoin network.

A subset of the nodes, called miners, compete against one another in a parallel transaction confirmation process. Transactions are confirmed once they are included in a block added to the blockchain (*mined* block).

To create a block, miners have to solve an exponentially difficult computational problem and therefore have to employ their computational power for process. The miner that first creates the new block is rewarded with reward, that is included as a *reward generating transaction* in the block, and with the *transaction fees* paid by the users whose transactions have been confirmed.

Once the transaction is included in the blockchain it is confirmed thus the receiver will be able to spend again the transferred coins.

## 1.2 Proof-of-work (PoW)

Why PoW?

## 1.3 Transaction structure

Transactions implement transfers and, collected and ordered in the blockchain, represent the entire set of coins available to users and its history. As a case in point, one can determine the balance of a certain wallet just by scanning the transactions on the blockchain [?].

In Fig. 2 the reader can find the structure of a transaction, with its main fields.

In particular, the `lock_time` field refers to the time or the blockchain height after which the transaction can be included in a block.

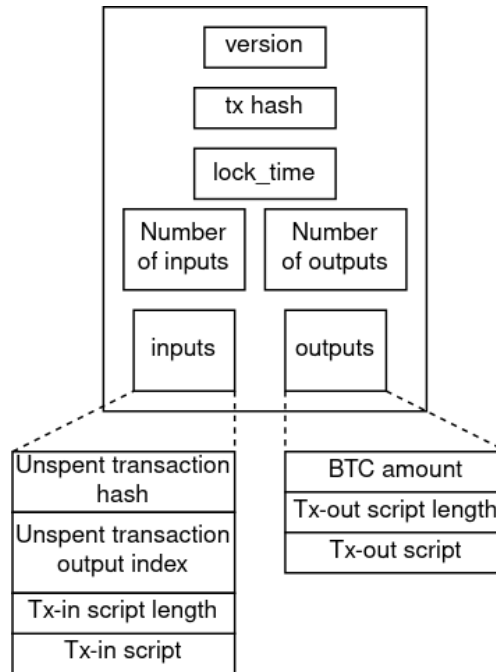


Fig. 2: Transaction structure

Transactions collect a set of *inputs* and *outputs*. Inputs are a collection of unspent coin sets, that are available to the user as previous transaction outputs or *unspent transaction outputs* (UTXO). Outputs, instead, specify the number of coins to be transferred.

Each input and output structure include *scripts*. These are a set of instructions, written in a Forth-like, stack based language that describe the means to access the transferred coins. The script of a standard transaction includes a public key that, when hashed, yields a Bitcoin address, and a private key signature. The most common scripts on the market are *Pay-to-PubKeyHash* (P2PKH) and *Pay-to-ScriptHash* (P2SH).

Broadcast transactions may be deemed invalid by honest miners and thus dropped. That might be the case for transactions that try to spend more than once the same set of coins or use invalid addresses. Nodes that broadcast invalid transactions may incur in bans if their behaviour is considered malicious.

## 1.4 Block structure and blockchain

ciao

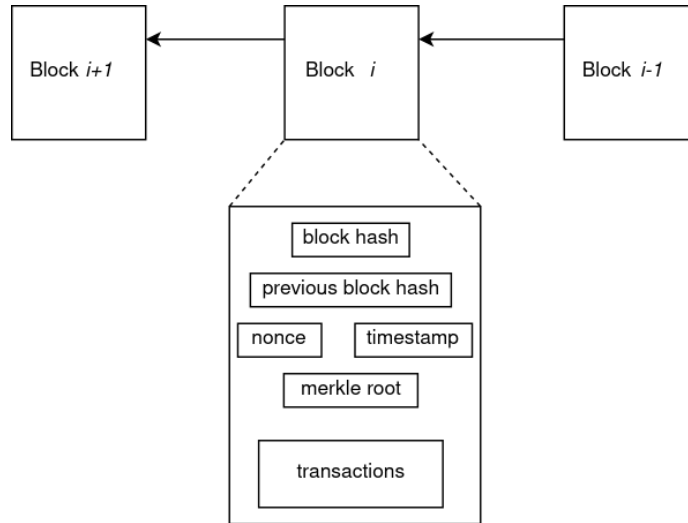


Fig. 3: Block structure in the blockchain

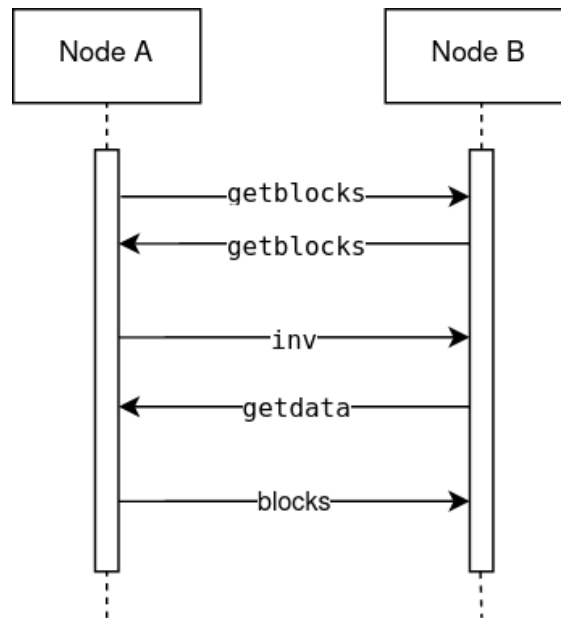


Fig. 4: blockchain synch

## 1.5 Miners behaviour

## 1.6 Consensus protocol

# 2 Bitcoin peer-to-peer protocol

## 2.1 Connection and peer discovery

In order to establish a connection two peers need to exchange version messages over an unencrypted TCP channel in an handshake fashion as shown in *Fig. 5*. The node that first

sends a **version** message is said to be establishing an *outgoing connection*. The receiving node sets up an *incoming connection* instead. Once the handshake is completed the connection is fully set up.

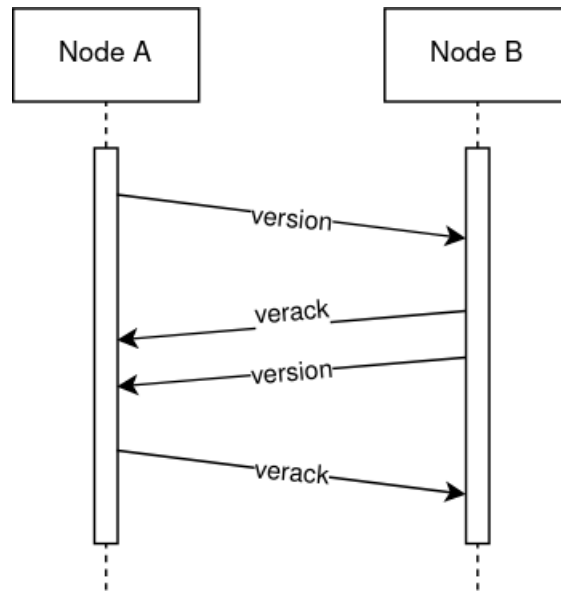


Fig. 5: Peers connection handshake

Each node is able to establish up to 8 outgoing connections and 117 incoming connections, for a total of 125 connections. In order to keep its connections active, each peer sends a message to each neighbour at least once every thirty minutes. If more than ninety minutes pass without receiving anything from a neighbour, the connection is dropped. Each node always tries to keep eight outgoing connections active, therefore, upon dropping an outgoing connection, a node tries to connect to another address in its peer cache.

Nodes on fresh bootstrap make use of hardcoded *DNS seeds* as their first mean to discover other peers. DNS seed servers are maintained by community members and provide clients with a list of addresses that can be either dynamically gathered through a periodic scan of the network or manually updated by server administrators. As a fallback option the user is able to specify through command line a list of addresses the client can connect to. Were both these two options to fail the client has a hardcoded list of peers it can directly connect to, though this is considered the last resort for bootstrapping peers.

Nodes at startup that were previously on the network shall first lookup peer names in their local address database, implemented as "peer.dat". The database contains the address of each peer the node has come to know during its lifetime in the network.

If the node has disconnected for a time too long, many of the addresses in the database may have become outdated or unreachable. A node that cannot connect to any address in the peer database or has spent up to eleven seconds trying to connect unsuccessfully to at least one of the peers in the database behaves as on fresh bootstrap and resolves to query a DNS first.

The use of a local address database, also called *peer cache*, provides reconnecting peers a fully-decentralized way to join the network and is the first line of defence against *fake*

*bootstrap attacks*; the topic along with other security concerns is discussed in Chapter ??.

Peer discovery after the first connection of a node is carried on through the exchange of **addr** messages containing the address and port number of other peers in the network. On connection set up the two nodes exchange **addr** messages, as shown in *Fig. 6*, providing each other with addresses from their local peer database. On top of that, every twenty-four hours each node advertises itself on the network with an **addr** message containing only its own address.

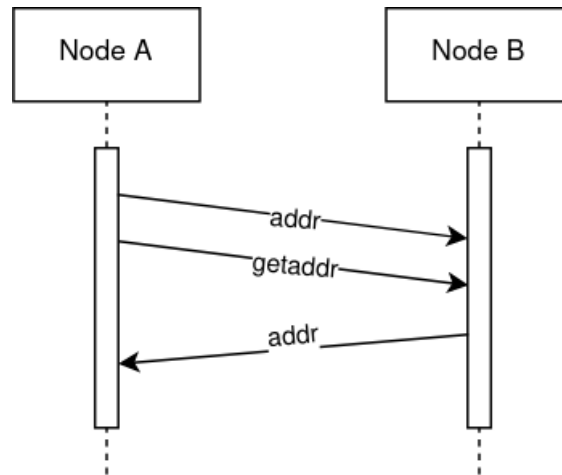


Fig. 6: Addresses exchange upon connection

**addr** messages can contain at most one thousand addresses; additionally those containing ten or fewer addresses are relayed. This behaviour contributes to the gossiping of self-advertisement messages even though messages are relayed only to a small subset of neighbours, namely to a couple of peers.

## 2.2 Peer database structure

The local peer database or cache serves the purpose of storing the addresses a node has come to know from **addr** messages and DNS seeds. The database is constituted of the **new** and the **tried** tables.

The **tried** table contains addresses of peers with whom the node has successfully established a connection in the past. It consists of 64 buckets that can store up to 64 unique addresses. Buckets are selected through a hash on the IP address.

The **new** table contains the addresses the node has not connected to and is therefore larger: it has 256 buckets of 64 addresses each.

Peer caches have been widely implemented in distributed systems so far as they are a fast and fully-decentralized way to rejoin the network after a disconnection. They allow peer-to-peer systems to overcome major issues at bootstrap time, mainly related to the presence of a single point of failure, i.e. bootstrap servers. Furthermore, they are the most scalable and efficient solution when compared to other decentralized mechanisms such as *address probing* [3].

## 3 Bitcoin security concerns

### 3.1 Common security threats on cryptocurrencies

Bitcoin’s decentralized, uncontrolled environment is open to a wide range of attacks, many of which can be applied to other cryptocurrencies as well. For this reason the following could be considered attack paradigms, with the same rationale, but different implementation amongst different cryptocurrency systems.

Attacks like *double-spending* aim to exploit system deficiencies: a malicious user tries to spend, with different transactions, the same set of coins two times. Although proof-of-work consensus makes double-spending infeasible, the attack can be still executed through other approaches that manipulate peers, as with network partitions, or were other conditions to be met [9].

Double-spending can be also carried out were an attacker to hold the majority of the computing power, this condition is known as *51%* or *majority* attack [14]. With the majority of the network’s hash rate an attacker would also be able to fork the blockchain to its own advantage and prevent transactions and blocks from being verified; this represents the worst security threat for any cryptocurrency.

Blockchain forks can be exploited to engineer block races as in *selfish mining* attacks, were the malicious user intentionally holds private his mined blocks [7], [11]. If the attacker successfully keeps his private chain longer than the public one it can release his mined blocks, thus claiming the rewards for himself and wasting other miners’ computing power.

These are just a few attacks that can be driven onto Bitcoin, as this chapter serves only introductory purposes. In the following sections there is more on network attacks, which are the focus of this thesis. The interested reader can find complete security overviews in the works of Conti et al., Saad et al. and Wang et al. [1], [13], [16].

### 3.2 Network security

Attacks that exploit vulnerabilities in the design and implementation of Bitcoin protocols and its peer-to-peer network fall under the category of network attacks.

Bitcoin is exposed to all the common threats that affect other peer-to-peer systems; these have been widely studied in the literature as in Touceda et al, 2012 [15]. The aim of this section is to understand how these attacks apply to Bitcoin and what are its defences.

The attacks reviewed in this chapter mainly focus on topology manipulation, with the exception of the *DDoS* attack.

Often the goal of the attacker is to alter the view of the network of a node, or group of nodes, by sending false network information with the purpose of monopolizing its connections.

Thus victims become isolated and a network partition is created. This condition, often referred to as the *eclipsing* of node, is inherently dangerous for the attacker can alter the node’s view of the blockchain, waste its computing power or simply cut off its communications.

More specifically, eclipsed victims are vulnerable to *N-confirmation double spending* and selfish mining attacks; furthermore 51% attacks are easier to launch [6]. For these reasons

network attacks on topology are fundamental enablers for mining and spending attacks [4].

In the following sections the reader can find both the attacks performed during the simulations reported in this thesis along with other network level attacks. These were added to better understand the mechanisms exploitable to carry out network attacks on cryptocurrencies.

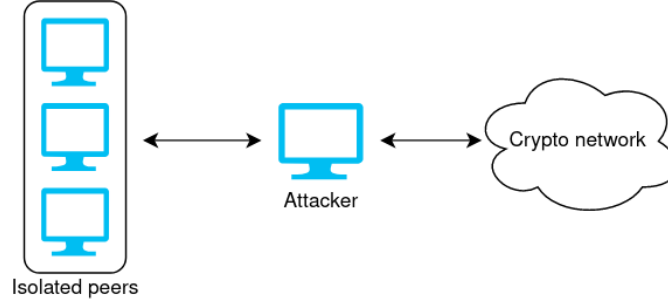


Fig. 7: Network partition

### 3.2.1 Sybil attack

In a Sybil attack a malicious user controls multiple identities on the network that can be either real machines or dummies, fake replicas of the attacker [5].

Such vector of attacks is applicable to any peer-to-peer system where users can create an arbitrary number of identities on the network for there is no node identification mechanism [10]. Thus, Bitcoin can suffer from such attacks.

Although the creation of fake identities cannot be exploited to carry out 51% attacks, since they do not hold any computing power, they can still be used to spread forged information or to withhold blocks.

In the latter scenario, Sybil nodes, upon receiving a new block, send `inv` messages to their neighbours, but then leave unanswered the following `getdata` messages, thus delaying the propagation of blocks, as shown in *Fig. 8*.

If the propagation is delayed long enough, the victim node could start mining the next block - in such scenario a block race can be engineered. Block races occur whenever the blockchain is forked and different subsets of peers work on different branches of the chain.

Since only the longest chain stored among all peers is considered valid, thus all the other branches are dropped, the malicious user can exploit such forks to waste other peers' computing power or to perform double-spending [17].

Let Tx 1 and Tx 2 be two different transaction released at the same time by the malicious user,

In the latter scenario, once the malicious user has successfully created a stale view of the blockchain in the victim, thus a fork, he releases two different transactions, Tx 1 and Tx 2, at the same time. Tx 1 is sent only to the victim node and Tx 2 is kept for its private chain, as shown in *Fig. 9*.

In the meanwhile Sybil nodes hamper the growth both of the public and victim's blockchain withholding blocks. As result the attacker can succeed at mining the next block first and



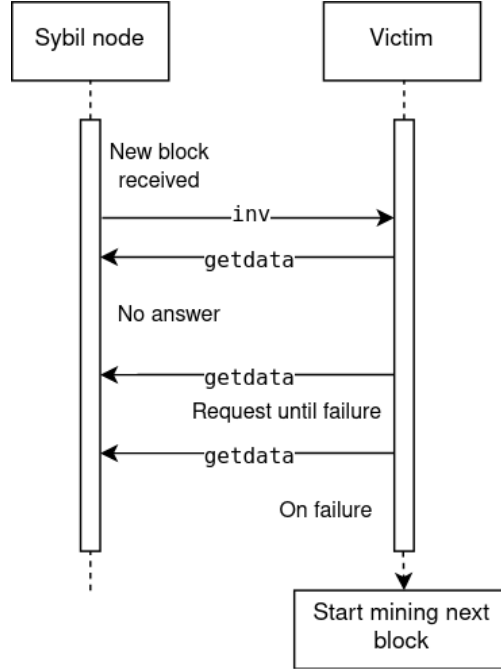


Fig. 8: Sybil nodes behaviour during the attack

then release the private blockchain therefore invalidating the victim's blockchain, as shown in Fig. 10.

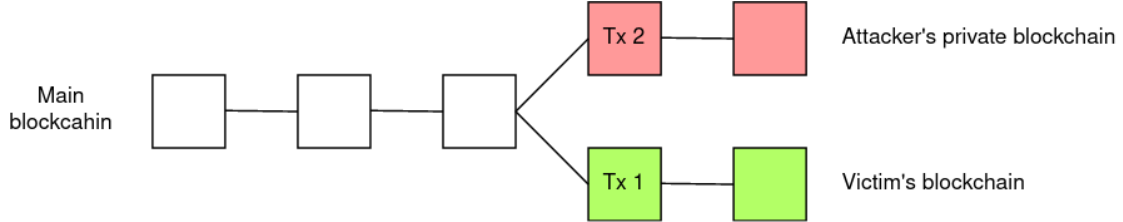


Fig. 9: Block race for double-spending

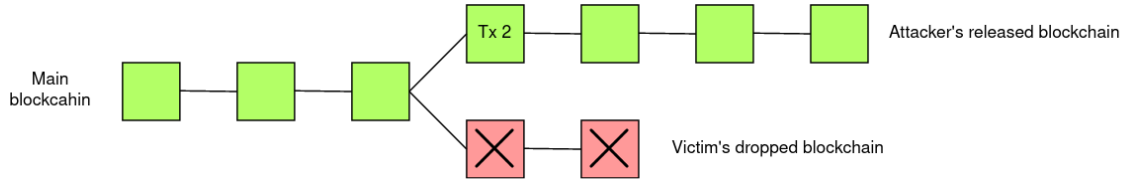


Fig. 10: The attacker releases the longest chain

Since Bitcoin requires no authentication or authorization process to join the network, there will always be the possibility for adversaries to create Sybils. Nevertheless counter-measures have been studied for different attacks based on Sybil attacks. For instance, setting a transaction block-confirmation timeout, have been suggested by Zhang et al. [18].

Be that as it may, the implications of Sybil attacks are still a topic open to study. The interested reader can find more in the work of Iqbal et al. [8].

### 3.2.2 Eclipse attack

The aim Eclipse attacks is to partition the network so that the attacker may be able to control all the data flow between the two partitions.

In order to do that the malicious node, or a set malicious nodes, floods the victim with a multitude of incoming connections and feed it bogus network information (`addr` exchanges), i.e. peer addresses controlled by the adversary, that will fill up their peer cache.

Upon restarting, with high probability the victim will form all of its eight outgoing connections with malicious IP addresses, thus finding itself isolated from the network.

Successful Eclipse attacks enable the launch of mining and spending attacks. Namely, eclipsing a subset of peers eliminates their mining power, hence making 51% attacks easier. Selfish mining attacks can be carried out as well.

The attacker that successfully splits the network can also engineer block races to deliberately waste resources of other miners or attempt at double-spend attacks, as in Fig. 9 and Fig. 10, described in Section 3.2.1.

Depending on the victim's policies, the malicious user can launch a *0-confirmation attack*, that is for the case in which the merchant decides to release goods before the transaction is confirmed. As in the case of Sybil based double-spend, the attacker releases two transactions and keeps the victim's view of the blockchain stale, so that it would be dropped later (Fig. 11).

Compared to the Sybil-based double-spend, Eclipse-based double-spend attacks of this kind have a greater chance of success, since they control all the victim's traffic and do not have to rely on delaying a block from spreading.

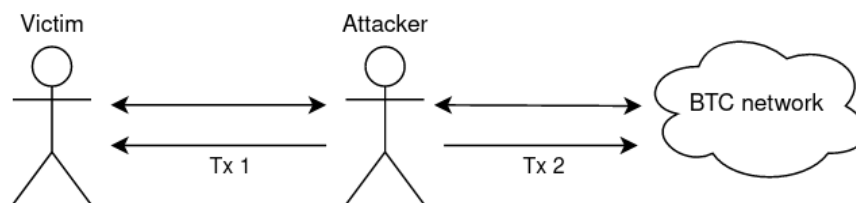


Fig. 11: 0-confirmation double-spend

Were the victim to adopt a  $N$ -blocks confirmation policy for transactions, the attacker can still launch a  *$N$ -confirmations double-spend* if it successfully eclipsed a subset of connected peers.

If that is the case, then the victims can be kept on an arbitrarily stale view of the blockchain and then be given one of the two transactions the attacker creates, while giving the other to the rest of the cryptocurrency network. After the eclipsed peers mine  $N - 1$  blocks, the transaction is confirmed and the goods are exchanged. Although the obsolete branch of the blockchain known by the victims will be dropped once the attacker releases the up-to-date (and longer) version disseminated in the network (Fig. 12 describes the process).

In 2015, Heilman et al. widely describe eclipse attacks and suggest many countermeasures to harden the network, a few of which were implemented in Bitcoin Core [6]. As part of the adoption of these countermeasures, the number of buckets in `new` and `tried` has been increased, the addresses have become deterministically hashed to a single slot inside a bucket and the heavy bias towards addresses with fresher timestamps when choosing new connections has been removed.

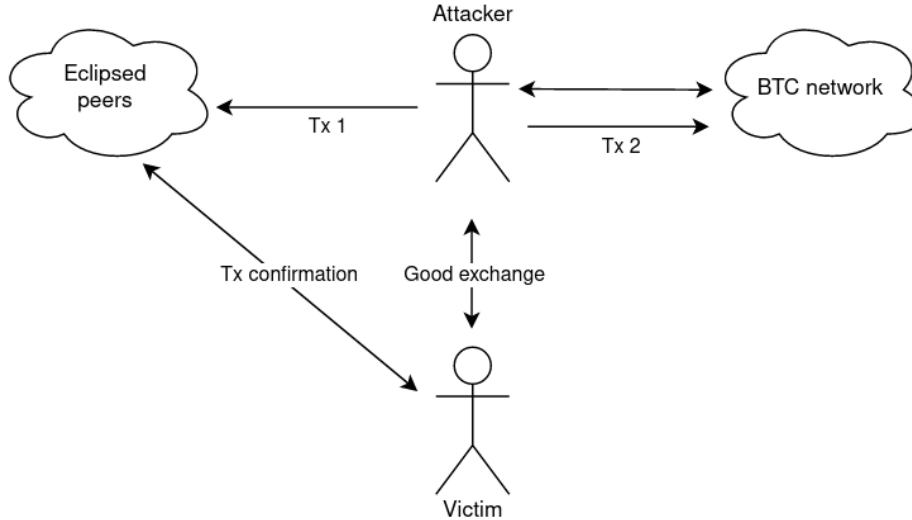


Fig. 12: N-confirmation double-spend

### 3.2.3 Fake bootstrapping attack

Every peer-to-peer system needs some mean to let nodes join the network: in Bitcoin bootstrapping nodes need to rely on the information provided by some other peer in the network.

In a fake bootstrapping attack the peers contacted while a node is still initiating its connections, and hence its knowledge of the network, flood the bootstrapping peer with malicious addresses.

The results in Chapter ?? show how feeding bogus information to a joining node can seriously influence its view of the network and lead to network partitions and peer eclipsing. This condition would also increase the efficacy of other network-level attacks, such as a DDoS, and the success rate of spending and mining attacks [6].

Prevention is done by ensuring a node can contact a trustworthy peer when joining the network. Peer caches and centralized bootstrap services serve this purpose, allowing joining nodes to make use of old neighbours while having central bootstrap servers as a fallback.

Other solutions have been implemented and tested over time, though their drawbacks usually exceed the benefits. As an example, randomly probing the address space, scouting for peers, decentralizes the bootstrap process and increases the variance of the network knowledge, but may not work if peers are behind NATs or firewalls and increases the latency for joining the network [3] [2]. In 2012, Touceda et al. discuss fake bootstrapping along with other security threats in their survey [15].

Bitcoin employs many countermeasures, and overcomes other bootstrap issues, with its peer cache implementation mixed with external mechanisms: each node contacts its stored addresses for successive connections and aims to establish multiple outgoing connections, thus it does not rely on a single bootstrap node. On top of that, DNS seeds provide a reliable external bootstrap service and the user is given the ability to manually insert addresses to connect to [12].

### 3.2.4 DDoS attack

## 4 Results

attack description methods adopted, metrics and software results obtained: show them, then discuss them

## References

- [1] Mauro Conti, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys Tutorials*, 20(4):3416–3452, 2018.
- [2] C. Cramer, K. Kutzner, and T. Fuhrmann. Bootstrapping locality-aware p2p networks. In *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004) (IEEE Cat. No.04EX955)*, volume 1, pages 357–361 vol.1, 2004.
- [3] Jochen Dinger and Oliver P. Waldhorst. Decentralized bootstrapping of p2p systems: A practical view. In Luigi Fratta, Henning Schulzrinne, Yutaka Takahashi, and Otto Spaniol, editors, *NETWORKING 2009*, pages 703–715, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [4] Maya Dotan, Yvonne-Anne Pignolet, Stefan Schmid, Saar Tochner, and Aviv Zohar. Survey on cryptocurrency networking: Context, state-of-the-art, challenges, 2020.
- [5] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
- [6] Sharon Goldberg Ethan Heilman. Alison Kendler, Aviv Zohar. Eclipse attacks on bitcoin’s peer-to-peer network. Cryptology ePrint Archive, Report 2015/263, 2015. <https://eprint.iacr.org/2015/263>.
- [7] Ittay Eyal and Emin Gun Sirer. Majority is not enough: Bitcoin mining is vulnerable, 2013.
- [8] Mubashar Iqbal and Raimundas Matulevičius. Exploring sybil and double-spending risks in blockchain systems. *IEEE Access*, 9:76153–76177, 2021.
- [9] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS ’12*, page 906–917, New York, NY, USA, 2012. Association for Computing Machinery.
- [10] Michal Kedziora, Patryk Kozłowski, and Piotr Jozwiak. *Security of Blockchain Distributed Ledger Consensus Mechanism in Context of the Sybil Attack*, pages 407–418. 09 2020.

- [11] Tin Leelavimolsilp, Long Tran-Thanh, and Sebastian Stein. On the preliminary investigation of selfish mining strategy with multiple selfish miners, 2018.
- [12] Mahmoud Mostafa. Bitcoin’s blockchain peer-to-peer network security attacks and countermeasures. *Indian Journal of Science and Technology*, 13:767–786, 02 2020.
- [13] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, DaeHun Nyang, and Aziz Mohaisen. Exploring the attack surface of blockchain: A systematic overview, 2019.
- [14] Sarwar Sayeed and Hector Marco-Gisbert. Assessing blockchain consensus and security mechanisms against the 51 *Applied Sciences*, 9(9), 2019.
- [15] Diego Suarez Touceda, Jose M. Sierra, Antonio Izquierdo, and Henning Schulzrinne. Survey of attacks and defenses on p2psip communications. *IEEE Communications Surveys Tutorials*, 14(3):750–783, 2012.
- [16] Hao Wang, Chunpeng Ge, and Zhe Liu. On the security of permissionless blockchain systems: Challenges and research perspective. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2021.
- [17] Shijie Zhang and Jong-Hyouk Lee. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Transactions on Industrial Informatics*, 15(10):5715–5722, 2019.
- [18] Shijie Zhang and Jong-Hyouk Lee. Mitigations on sybil-based double-spend attacks in bitcoin. *IEEE Consumer Electronics Magazine*, pages 1–1, 2020.