UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in

...

Elaborato finale

Titolo

Sottotitolo (alcune volte lungo - opzionale)

Supervisore

.....

Laureando

.....

Anno accademico .../...

Ringraziamenti

...thanks to...

Contents

Abstract

In the past decade cryptocurrencies have become increasingly popular and widespread, up to the point of being a good alternative to commonly used currencies.

The first and most popular cryptocurrency is Bitcoin, that in little more than ten years has gained millions of users.

The strength of cryptocurrencies lies in their strong cryptographic guarantees, that make transactions tamperproof, and their open, free-to-join networks. On top of that, the communities grow steadily, as users are incentivized to take part in transaction verification as they get a revenue out of it.

Furthermore, cryptocurrencies have demonstrated to be a game-changer in the field of distributed systems as they take a completely different approach on Byzantine fault tolerance.

As a result, the rise of cryptocurrencies has entailed a great number of studies both on their theoretical foundations and on their security.

A great deal of attention has been driven onto cryptocurrencies network security, as their fully-decentralized, open environments are vulnerable to a high number of attacks, ranging from common flooding denial-of-service and Sybil attacks to more sophisticated linkage deanonymisation attacks.

In this work we study the security of Bitcoin peer-to-peer network and address its vulnerabilities.

More specifically, we try to enhance the efficacy of a block-withholding Sybil attack by disseminating false network information during the formation of the network topology.

For the purpose, a discrete-time event-based blockchain simulator named LUNES-blockchain has been used.

The behaviour of the malicious simulated entities has been changed so that they would advertise other attacking nodes during peer discovery, thus increasing the probability of honest nodes connecting to malicious nodes.

Results show how the efficacy of the Sybil attack launched after the topology has formed is greatly increased. As a case in point, the messages created by honest nodes have an extremely low coverage, despite only a small number of attackers is on the network.

Nevertheless, we also show how the timing of the attack in fundamental. In a second attack scenario the Sybil nodes are on fresh bootstrap, thus honest nodes start first in building their own honest-only network.

The strategy of the adversary fails in this case, as honest nodes have enough time to build a small, sparse network of their own, before their caches are flooded with false network information.

The Sybil attack in this latter scenario is utterly not effective. Consequently, the robustness of the Bitcoin network demonstrated.

Nevertheless, single, isolated nodes are still vulnerable to such strategies, thus pointing out the need for reliable means to bootstrap nodes for every distributed peer-to-peer

system.

1 Distributed data storages

The reasons to choose to store data in a distributed database over a centralized one are many. The choice may come from the need for greater scalability, to enforce availability guarantees, since the system does not a single point-of-failure, or directly derive from the architecture of the system, since they the most natural choice for peer-to-peer systems.

The distributed approach to databases has been taken multiple times over time, with different architectures and assumptions on the system. A common and clear example of distributed database is the Domain Name System.

Also cryptocurrencies, which are the main topic of this work, manage a particular kind of distributed database, that allows data to be only read or appended, called ledger [28].

Be that as it may, from distributing data storages comes new challenges, that derive from the kind of failures the system may undergo. The architecture of fully-decentralized peer-to-peer systems makes them exposed to the biggest class of failures, hence reliable, secure implementations are harder to develop.

In the next chapter the reader can find a brief introduction to the problems concerning distributed databases with some key concepts, that are fundamental to understand Distributed Ledger Technologies and cryptocurrencies.

1.1 Addressing the issues of distributed ledgers

The desired property for distributed data storages is *data consistency*, which guarantees that the result of reads and writes will be predictable and the same for all peers. As an implication, local data has to be up-to-date.

Cryptocurrencies adopt *eventual data consistency*, a relaxed definition of data consistency, which ensures that if the ledger is not updated for a time long enough, all peers eventually agree on the state of the memory.

In order to grant data consistency, distributed architectures need a *consensus mechanism* so that peers would agree on the current memory state.

Nevertheless, achieving consensus, and consistency, is not trivial, since nodes communicate through an unreliable network and no assumption can be made on the activity of nodes.

Peers can undergo failures, be unreachable or, in the worst case scenario, act maliciously. In the latter case, nodes could even try to alter the content of database.

Different solutions to the problem depend on the set of assumptions on the system, and the reliability of its nodes and channels. Naturally, the fewer the assumptions on the system, the harder it is to find a solution for consensus, even with relaxed definitions.

The class of failures to which the system is exposed when no assumption on the network and on the behaviour of peers can be made are called *Byzantine failures*. The name derives from the famous Byzantine Generals Problem [21].

The systems considered in this work undergo Byzantine failures.

Ensuring data consistency is the main goal of any distributed database system. Different protocols and algorithms have been introduced in the past to achieve consensus in case of Byzantine failures, like Practical Byzantine Fault Tolerance or Byzantine Paxos.

Despite that, it is often hard to handle all the security threats incoming from malicious users, both on network and application level. Consequently, most distributed databases used to be run on controlled networks, with authenticated nodes only, and work with a weaker set of assumptions on the system.

Cryptocurrencies take a different approach on the subject and handle open, permissionless environments. Despite being exposed to Byzantine failures, they still manage to grant data consistency, and for this reason they have been widely studied in the past decade.

In the following chapter the reader can find an introduction to Bitcoin basic concepts that are mostly shared with all other cryptocurrencies.

2 Introduction to Bitcoin

Bitcoin is a cryptocurrency and an electronic payment system built on a fully-decentralized peer-to-peer network.

It uses a *permissionless* blockchain as ledger and relies on a probabilistic consensus algorithm based on *Proof-of-Work* (PoW) to achieve eventual data consistency.

The blockchain is a public tamperproof append-only list where all the transactions are stored. The resistance to tampering comes from the blockchain structure, as shown in Section 2.3. Each block contains the hash of the previous one, thus makeing tampering infeasible and ensuring data integrity.

A Proof-of-work is a computational task given to peers, which solution is necessary to validate data appended to the blockchain. The high amount of time required to solve PoW puzzles makes attacks on the consensus algorithm infeasible.

Under the assumption that no user holds more than 50% of the computing power on the network (see Chapter 4, PoW-based consensus algorithm achieves Byzantine fault-tolerance [23].

While the consensus algorithm makes peers agree on the order of blocks and transactions in the ledger, the proof-of-work ensures that no adversary that owns multiple identities is capable of validating and verifying its own forged blocks.

In order to validate its own blocks the adversary would need the computational power and the time required to solve the PoW puzzle. Therefore having multiple network identities is not enough to subvert the consensus algorithm.

Although, this implies that the result does not hold were the attacker to hold most of the computing power on the network. If that is the case, the adversary can simply keep building up its own ledger and eventually its blockchain will be the authentic, generally accepted one.

Bitcoin and its PoW-based consensus grant a secure, public and fully-decentralized storage for transactions, which was unthinkable of before the past decade. For this reason, Bitcoin has changed the way scholars approach Byzantine fault-tolerance and has been widely studied in the past years.

In the following sections the reader will find an introduction to the Bitcoin network, where its main components and their function are analysed.

2.1 An introductory example

A basic use case, shown in Fig. 2.1, is presented here to get the reader started with Bitcoin.

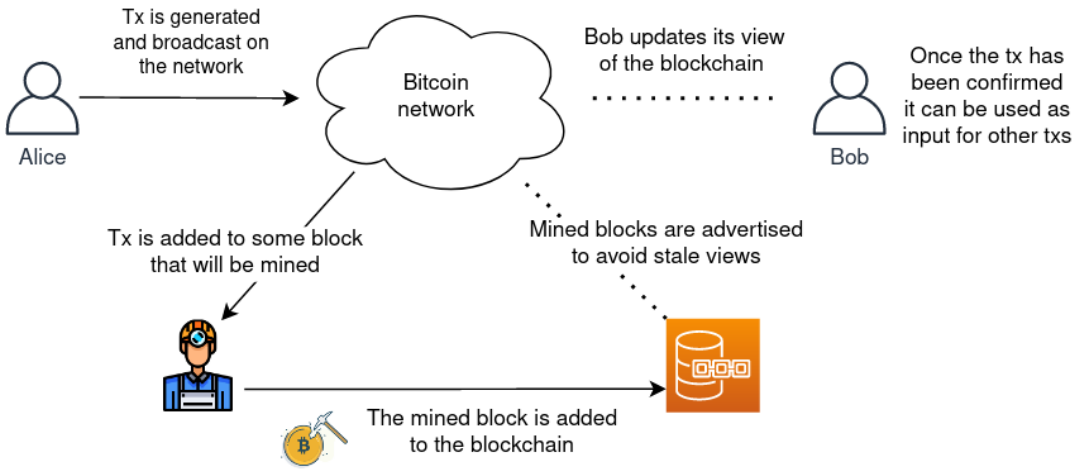

Fig. 2.1: Bitcoin introductory use case

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 sign issued transactions. instead, the public key is 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. The miner that first creates the new block earns Bitcoins, through a *reward generating transaction* included in the block and 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.

2.2 Transaction structure

Transactions implement transfers. Valid transactions are collected together in the blockchain, which represents the entire, ordered history of all sets of coins ever available to users. Hence, one can determine the balance of a certain wallet just by scanning the transactions on the blockchain [30].

In Fig. 2.2 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.

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*

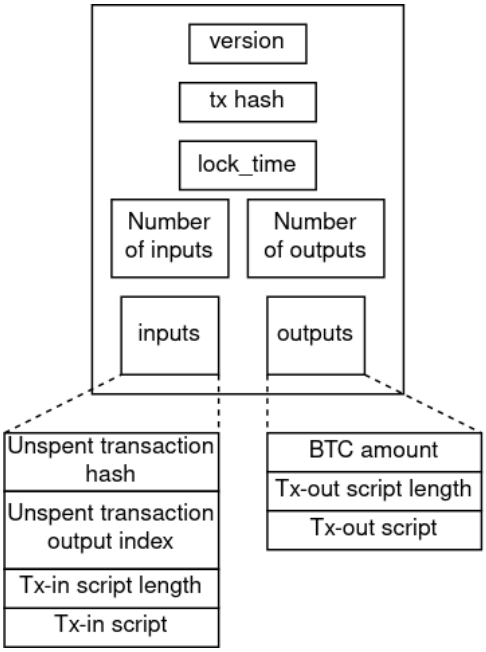

Fig. 2.2: Transaction structure

outputs (UTXO). Outputs, instead, specify the number of coins to be transferred along with a receiver address.

Each input and output structure includes *scripts*. These are sets 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).

Broadcasted transactions are verified before being added to a block and 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.

2.3 Block structure and blockchain

The blockchain is a public distributed ledger, structured as an append-only linked list. It stores all valid transactions form of blocks.

In a block a set of transactions is bundled together and stored as a Merkle Tree, which ensures the data is unaltered and undamaged from the moment it is put into the block.

On top of that, each block contains the hash of the previous block in the blockchain and its own hash, thus preventing entirely block tempering: altering a transaction would result in mismatching hashes in the Merkle Tree, as well as in the stored block hash and in the hash in the next block in the blockchain.

Blocks have to be mined in order to be added to the blockchain. This means that the right *nonce* needs to be found. The nonce is a varying value field that modifies the resulting hash of the block. A block is considered valid when a nonce such that the resulting hash is lower than a well-known *target value* is found. That is equivalent to finding an hash that begins with n zeroes.

Looking for the right nonce is exponential on n number of zeroes.

The target value, and therefore the *difficulty* of the computational task, is adjusted every 2016 blocks in order to keep the block generation and validation time around ten minutes.

Fig. 2.3 shows the main fields stored in a block.

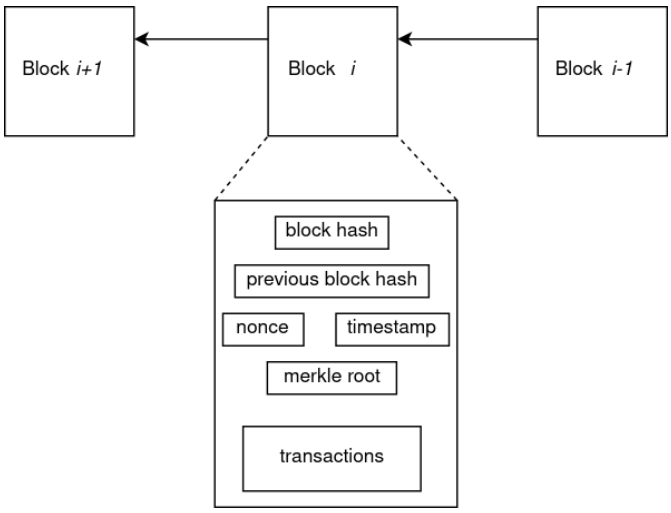

Fig. 2.3: Block structure and blockchain

2.4 Miners behaviour

The subset of peers that concurrently try to mine blocks by finding the right nonce are called miners, as described in Section 2.3. What motivates the miners to invest their computational power are mining rewards and transaction fees.

The mining reward is implemented as a transaction to the miner's wallet address that is included in the new block by the miner itself. Transaction fees are instead an amount of Bitcoins that the issuer of a transaction discretionally gives to the miners as an incentive for their work. Transactions with low transaction fees may incur in long processing times, since miners prioritize high rewards.

Transactions and blocks are gossiped through the network. Although each node has its own relay policy, it is in every peer's interest, as honest miners, to get to know most of the pending transactions available for mining, and to keep the view of the blockchain consistent at each node.

In particular, it is in the interest of miners to spread as fast as possible newly mined blocks in order to claim rewards and not waste time on stale views of the blockchain.

For the latter purpose, nodes need to keep their local blockchain view consistent with that of their neighbours. Nodes exchange `getblocks` messages to request `inv` messages which advertise local blocks. Missing blocks in a received `inv` message are retrieved with a `getdata` message [2]. The described message exchange can be found in Fig. 2.4.

Furthermore, `inv` messages are also gossiped when a node receives a new block. It is relevant to note that new blocks will be dropped by the receiving peers if any transaction is invalid or already spent. Nodes implicitly accept a block by starting to work on the next one, which contains the hash of last received valid block.

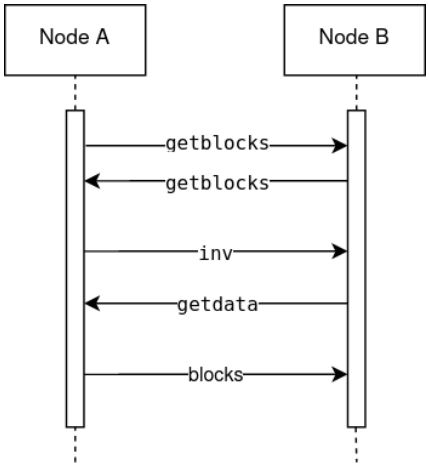

Fig. 2.4: blockchain synch

2.5 Proof-of-Work and Consensus protocol

In Bitcoin, as well as in other cryptocurrencies, the transaction verification process (mining) is distributed. The legitimacy of a transaction is verified by the majority of nodes before it is added to the public ledger, the blockchain.

These distributed environments are vulnerable to attacks in which an adversary connects multiple fake replicas of himself to the network. If the number of fake attacking nodes exceeds that of the honest nodes, the attacker can forge any kind of data and validate it, thus appending it to the blockchain.

To avoid such scenarios, in order to append data to the blockchain, a Proof-of-Work is required. PoWs are cryptographic math puzzles that require scanning for a value, the nonce, which correct value attests that the node has put computational effort and time to verify the transaction(s). Mining is also described in Section 2.1 and 2.3.

PoW discourages any kind of data forgery: in order to append arbitrary data to the blockchain, an adversary would require the majority of the computing power on the network. With more than the 51% of the *hashrate*, a user would be able to eventually win a block race against the remaining of the network, thus appending forged data to the ledger. This is although infeasible for networks large enough.

Furthermore, tampering the blockchain is impossible, thanks to the definition of Merkle tree and blockchain, as described in Section 2.3. Bitcoin is therefore resistant both to tempering and forgery.

Since mining is defined as a distributed process, and due to the lack of any central ledger, nodes may share inconsistent views of the blockchain. This is the case if relayed blocks are not delivered to some nodes or two or more nodes mine a new block approximately at the same time.

Forks on the blockchain that are not maliciously engineered are solved by themselves by following a simple rule: only the longest branch of the blockchain is considered valid.

If two nodes are working on different branches of the same height, due to the random nature of mining, eventually one will release a new block before another. Hence its blockchain will be the longest and the only one considered correct.

To ensure that all nodes agree on the order of the entries of the blockchain, some simple rules are defined. More specifically, they state that:

1. input and output values are rational

2. transactions only spend unspent outputs

3. all inputs being spent have valid signatures

4. no transactions spend inputs with a `time_lock` before the block in which they are confirmed

The PoW-based distributed consensus algorithm is Bitcoin's greatest strength, as it achieves Byzantine fault-tolerance with a set of adversaries as big as half of the network. It is defined as probabilistic since it guarantees eventual consistency. Furthermore, it is generally considered robust and extremely scalable, as no authorization is required to join the network and mine.

3 Bitcoin peer-to-peer network

Bitcoin nodes form an overlay network and communicate through a specific application-level protocol.

The topology of the Bitcoin network is unknown, but is generally considered to be structured as a random graph. Topology information is kept hidden as it can enable attacks on privacy [5]. The interested reader can find more in the work of Deshpande et al. [9].

The Bitcoin network is extremely diverse as different kind of nodes can join. The Bitcoin Reference Client developed by Nakatomo is provided with a *wallet*, which is an applicative module for payments, has a network interface, stores all the blockchain and is a miner at the same time. These can be defined as *full nodes*.

Nonetheless, not all nodes may want to store the entire blockchain or mine. Some miners often store only the headers of the blockchain in order to save space. Other users install only the wallet and do not take part in mining and verification at all.

An important exception are the unreachable nodes. These nodes work behind a NAT and often mine together in *mining pools*. A central server manages all the incoming traffic and shares the mining rewards among pool workers.

Nowadays most of the users that want to have a revenue from mining work in pools and therefore the number of the reachable nodes is not representative of the dimension of the network. In general the number of reachable nodes is always around ten thousand [4].

In this work only full nodes are considered as they are the ones that have an active role on the network.

The following chapters provide a complete overview of the Bitcoin protocol.

3.1 Connection and peer discovery

In order to establish a connection two peers need to exchange version messages over an unencrypted TCP channel in a handshake fashion as shown in Fig. 3.1. The node that first sends a `version` message establishes an *outgoing connection*. The receiving node sets up an *ingoing connection* instead. Once the handshake is completed the connection is fully set up.

Each node is capable to establish up to 8 outgoing connections and 117 ingoing connections, for a total of 125 connections. In order to keep its connections active, each peer

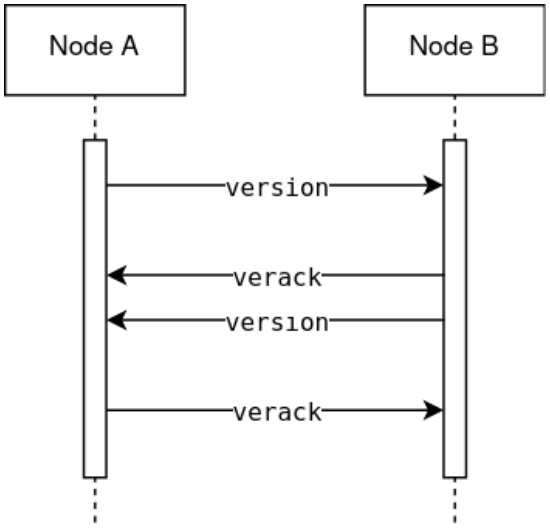

Fig. 3.1: Peers connection handshake

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 4.1.

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 [2] [1].

On connection set up the two nodes exchange `addr` messages, as shown in *Fig. 3.2*, 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.

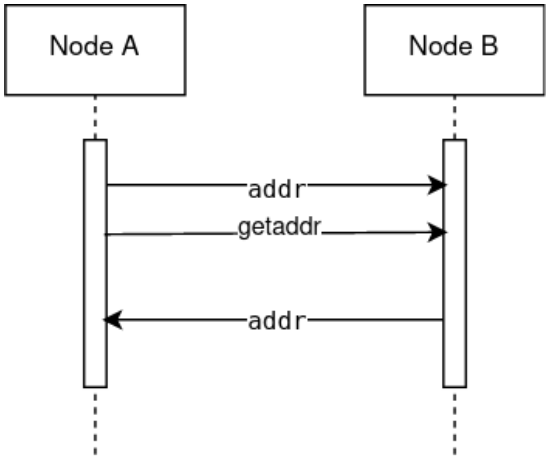

Fig. 3.2: Addresses exchange upon connection

`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.

3.2 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* [10].

3.3 Data dissemination

Data is `gossiped` through the network. A gossip or *epidemic* protocol is a communication paradigm for distributed systems where data is sent only to a subset of neighbours in order to reduce network traffic whilst still ensuring full coverage.

Nevertheless, no assumption can be made on the behaviour of nodes, therefore different nodes may implement different relay strategies.

Bitcoin Core simply broadcasts transactions and blocks to reduce their latency and keep local views consistent.

On top of that a simple strategy is adopted to try to enforce anonymity: it relays data with independent, exponential delays. Although, this policy has been proven extremely weak.

Various researches have proved that it is fairly easy for an adversary to deanonymize users [5], [20]. For this reason new dissemination protocols are now being adopted, first of which is *Dandelion*.

Dandelion has two distinct phases for dissemination: a *stem* phase and a *fluff* phase. During the stem phase the message is relayed multiple times to one neighbour only each time, so that the message creator is kept hidden. Once the stem phase ends, the data is normally broadcast in the network [6]

Dropping a message during the stem phase loses the data completely, thus the protocol is extremely weak to malicious behaviour from neighbours, like block withholding (see Sections 4.2).

For this reason, *Dandelion ++* was developed. In this version, honest nodes that forward data during the stem phase set a timer. If they do not see the forwarded message come back in fluff phase by the end of the timer, they broadcast the message themselves, forcing fluff phase [16].

With this simple change, Dandelion ++ greatly improves Dandelion robustness [27].

In this work, the tests are carried out using *fixed probability* gossip, where a message is sent to each neighbour with a fixed probability, and Dandelion ++.

4 Bitcoin security concerns

Bitcoin's decentralized, uncontrolled environment is open to a wide range of attacks, many of which are common threats amongst cryptocurrencies as well. For this reason, the attacks described here can be carried out on other systems as well. These attacks have the same rationale, but different implementations.

Decentralized permissionless blockchains with PoW-based consensus algorithms are by definition vulnerable to *51%* or *majority* attacks [23]. In these a user controls more than half of the computing power (often called *hash rate*) available on the entire network [26].

The attacker can thus subvert the consensus algorithm and validate harmful data, maliciously fork the blockchain or arbitrarily prevent blocks and transactions from verification.

In order to do this, the adversary has to exploit block races and Bitcoin's policy to consider the longest blockchain as the authentic one. The user that holds the majority of the blockchain only has to keep mining a private chain. Eventually its own chain will be longer than the public one, and therefore it can be released to invalidate the original blockchain and validate the forged one.

The way the correct state of the blockchain is elected, is exploited by other classes of attacks that engineer block races.

In block races two or more peers spend computing power mining different forks of the same height. By definition of Bitcoin's consensus, there will be a point at which only one branch will be selected as valid and all the others will be discarded.

Block races can enable double-spending, as shown in Section 4.2 and 4.3, or mining attacks such as *selfish mining*, where the malicious user holds private his mined blocks, thus forking the chain.

For the time the private chain of the attacker is longer that the public one, it can be released on the network, thus claiming the rewards for himself and wasting other miners'

computing power [15].

Most of the malicious users that aim for a revenue from their attacks opt for *double-spending* attacks. By definition, in double-spending attacks the same set of outputs is spent twice, hence the attacker gains Bitcoins or obtains resources from merchants for free.

Although proof-of-work consensus makes double-spending infeasible, since the transactions would not be validated, the attack can be still executed through other approaches. As a case in point, double-spending can be executed if the attacker successfully partitions the network (see Section 4.2, or were other conditions to be met [18]).

Since this chapter serves only introductory purposes, these are just a few attacks that can be driven onto Bitcoin. 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. [7], [25], [31].

4.1 Network security

Attacks that exploit vulnerabilities in the design and implementation of the Bitcoin protocol 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 [29]. The aim of this section is to understand how these attacks apply to Bitcoin and what are its defences.

In network attacks. often the goal of the adversary is to alter the view of the network of a node, or group of nodes, by sending false network information or by connecting a large number of adversary-controlled peers to the victim, with the purpose of monopolizing its connections.

In this way network partitions, or other states in which the attacker can effectively control the data received by the victim, are created.

Nodes that become isolated from the network become vulnerable to *N-confirmation double spending* or mining attacks, such as *selfish mining* and *stubborn mining* [24]. Furthermore 51% attacks are easier to launch, since a part of the network computing power is cut off.

For these reasons, network attacks are fundamental enablers for mining and spending attacks [11].

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.

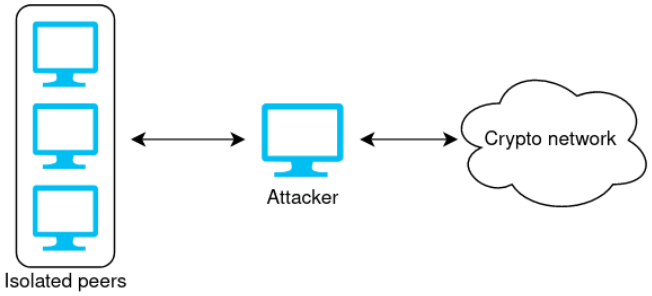

Fig. 4.1: Network partition

4.2 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 [12].

Such vector of attacks is applicable to any peer-to-peer system where users can create an arbitrary number of identities on the network, since there is no node identification mechanism [19]. 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. 4.2*.

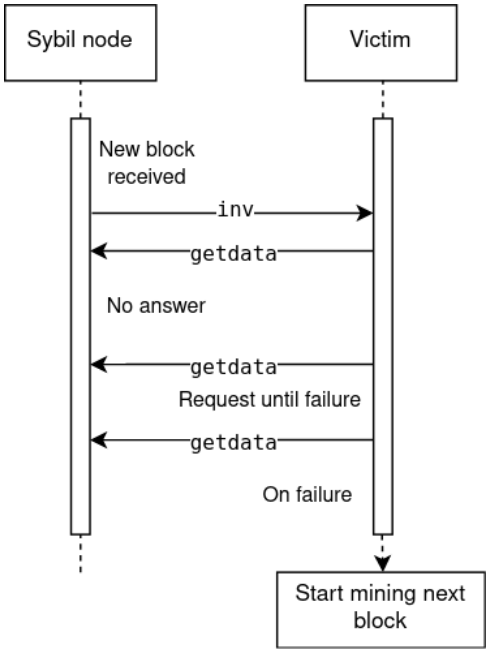

Fig. 4.2: Sybil nodes behaviour during the attack

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, all the other branches are dropped. The malicious user can exploit such forks to waste other peers' computing power or to perform double-spending [32].

To launch a double-spend, once the malicious user has successfully created a stale view of the blockchain in the victim, i.e. 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. 4.3.

In the meanwhile Sybil nodes hamper the growth both of the public and victim's blockchain by withholding blocks. As a result the attacker can mine the next block first and then release the private blockchain therefore invalidating the victim's blockchain, as shown in Fig. 4.4.

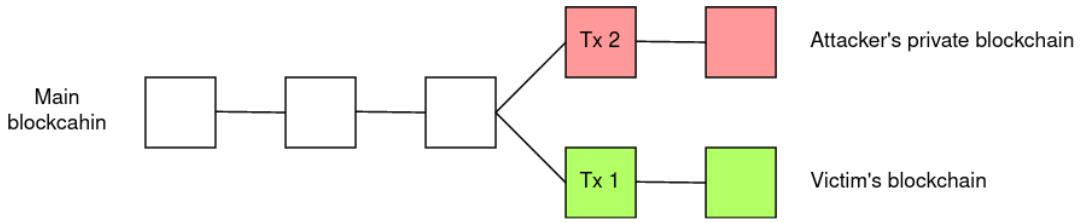

Fig. 4.3: Block race for double-spending

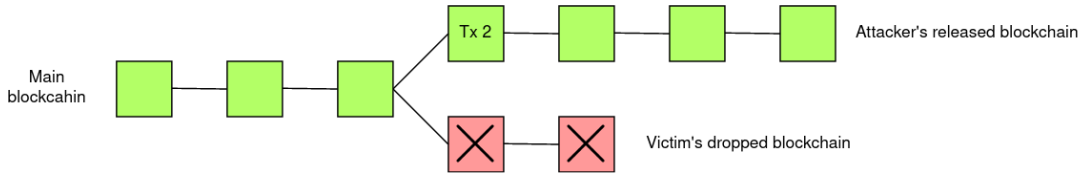

Fig. 4.4: The attacker releases the longest chain

Since Bitcoin requires no authentication or authorization to join the network, there will always be the possibility for adversaries to create Sybils. Nevertheless, countermeasures have been studied for different attacks based on Sybil attacks. For instance, setting a transaction block-confirmation timeout has been suggested by Zhang et al. [33].

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. [17].

4.3 Eclipse attack

The aim Eclipse attacks is to partition the network so that the attacker may be able to control all the data flowing 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, i.e. peer addresses controlled by the adversary, with `addr` exchanges, 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 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 the resources of other miners or attempt at double-spending attacks, as in Fig. 4.3 and Fig. 4.4, described in Section 4.2.

Depending on the victim's policies, the malicious user can launch a *0-confirmation double-spending 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. 4.5).

Compared to the Sybil-based double-spending, Eclipse-based double-spending 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 being disseminated. Although the attacker needs to go through the partition set up first, whereas creating Sybils is much easier.

Were the victim to adopt a N-blocks confirmation policy for transactions, the attacker

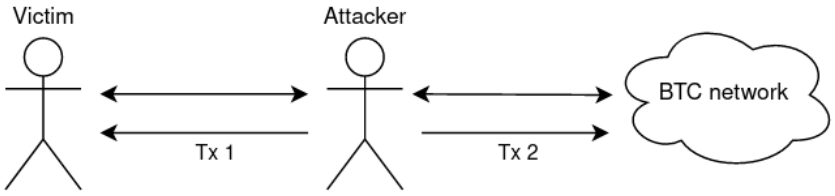

Fig. 4.5: 0-confirmation double-spend

can still launch a *N-confirmations double-spend* if it has 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. Thereafter, 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. 4.6 describes the process).

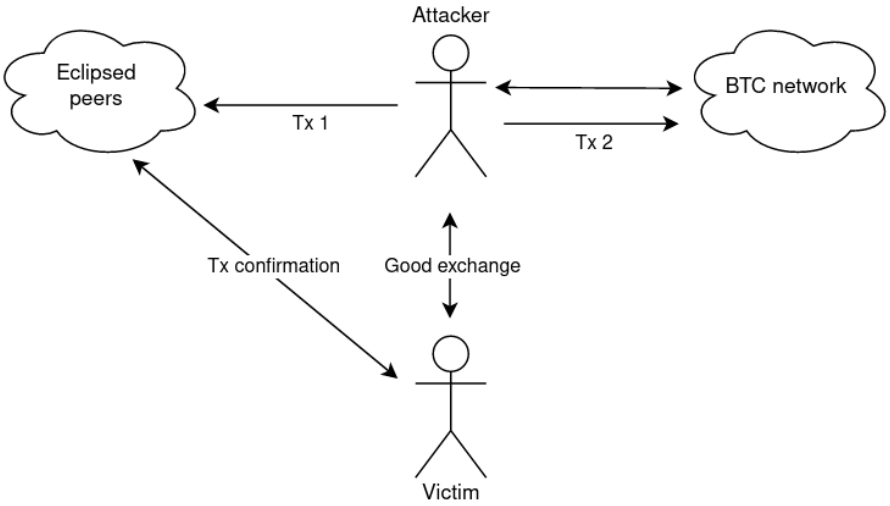

Fig. 4.6: N-confirmation double-spend

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 [14]. 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.

4.4 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 6 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 [14].

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 [10] [8].

In 2012, Touceda et al. discuss fake bootstrapping along with other security threats in their survey [29].

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 [22].

5 Enhancing Sybil attacks

This study focuses on the influence that the dissemination of false network information has during the construction of the topology of the overlay network and how this enhances other network-level attacks.

The performed attack is a Sybil attack, described in section 4.2. Sybil nodes are used to increase the connections with honest nodes the attacker has, thus to have greater control on the data flowing between honest nodes.

In this study, Sybil nodes drop all transactions to and from a chosen victim to perform a denial-of-service. The reader can infer that the same results are applicable for block-withholding attacks.

As previously stated, malicious nodes disseminate bogus network information: they try to advertise as much as possible the addresses of other attackers through `addr` messages. Consequently, honest nodes are more likely to connect to malicious nodes.

The setting of the attack is during the construction of the overlay topology: at the start the graph is only a tree and many nodes are on fresh bootstrap. Peers try to setup eight outgoing connections, emulating the behaviour of full nodes (Sec. 3.1). The way the graph forms is influenced by the network data advertised in this phase.

5.1 Attack details

The attack starts during the creation of the overlay topology, when nodes still have to setup most of their connections.

A group of randomly picked nodes is on fresh bootstrap. The rest of the nodes are

instead connected. The starting network graph is a tree, the choice comes from the need to have the sparsest graph possible.

In a series of rounds, nodes start to establish outgoing connections with addresses in their peer database. For honest nodes the upper bound on outgoing connections is eight, whereas there is no limitation for malicious nodes, that can therefore start connections with how many nodes they want. Sybil nodes prioritize connections with honest nodes, avoiding connections between each other.

Nodes on fresh bootstrap have to contact some DNS first. DNSs contain the data of a random subset of the nodes that were part of the network at the start. Their addresses will populate the bootstrapping node's peer database, that will then be able to connect to the network and start establishing more connections.

Network information is propagated through `addr` messages. These are exchanged upon establishing a connection. Bitcoin Core policy is to relay messages that contain at most ten addresses to a subset of neighbours [3]. For this reason, both attackers and honest nodes include in `addr` messages no more than ten addresses.

In the attack, honest nodes fill `addr` messages with a random subset of addresses in their peer database. Malicious nodes advertise only attacking nodes' addresses instead (Fig. 5.1).

Moreover, Sybils do not relay incoming `addr` messages, thus hampering the propagation of honest nodes addresses. Attackers work under the assumption that they share network knowledge, which means that once an attacker learns a new address, also all other attackers know it.

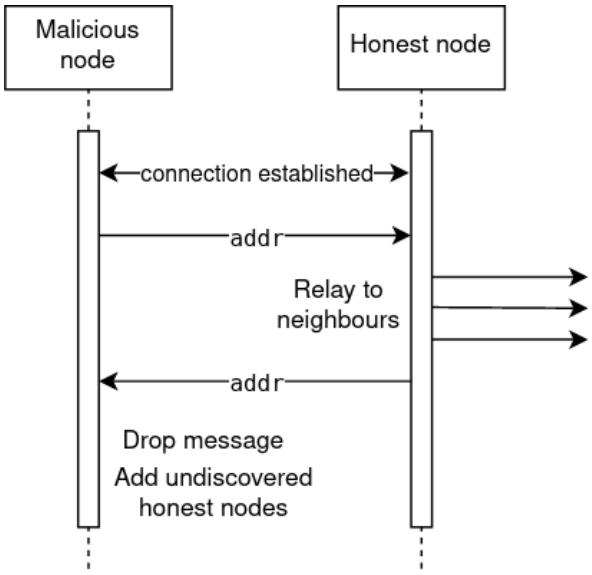

Fig. 5.1: Peer discovery during the attack

On top of that, each node self-advertises its address once with an `addr` message sent to its neighbours. Malicious nodes behave exactly as described before, dropping honest messages.

Once honest nodes cannot establish more connections, and the graph has become dense, the Sybil-based denial-of-service starts: Sybil nodes will drop all the transactions coming from the selected victim.

The Sybil attack has two different settings that lead to different results. In the first one, Sybil nodes are all in network from the start. In the second, all Sybil nodes are on

fresh bootstrap. More can be found in Section 6.

5.2 Metrics and performance evaluation

The tests performed aim to understand how the aforementioned changes in the behaviour of nodes affect the effectiveness of the Sybil-based DoS attacks when adopting different dissemination protocols.

The coverage of a message is defined as the percentage of nodes reached in the network while adopting a certain dissemination protocol.

To evaluate the robustness of fixed probability broadcast and Dandelion ++, the coverage of victims' messages is estimated.

Coverage values for different protocols are carried out through different simulations, while varying the number of attacking nodes and the underlying topology.

Other metrics that are used to evaluate the efficacy of the new behaviour of Sybils are the average degree of malicious nodes and the average percentage of malicious neighbours for honest nodes.

These can describe how effectively Sybil nodes have connected to honest nodes, thus influencing the effectiveness of the subsequent Sybils attack.

Furthermore, the average percentage of Sybils in the peer cache is used to assess how well the bogus network information was disseminated.

5.3 Software and implementation

The adopted simulator is LUNES-blockchain, a discrete-time event-based cryptocurrency-system simulator [13].

It consists of three main modules executed separately: topology creation, blockchain simulation and performance evaluation.

Topology creation is managed with the C library i-graph. Ideally, for each simulation a different set of graphs is generated.

Then the core simulation is run. Nodes mine blocks for the blockchain and exchange transactions with a previously specified dissemination protocol. The Sybil attack is carried out at this time.

At the end of the simulation, some scripts parse the data logged during the simulation and evaluate the robustness of the dissemination protocol.

The changes made to the behaviour of nodes apply only to the first part of the simulation: when nodes are still establishing outgoing connections. Only after this phase, the blockchain simulation is run and the Sybil attack is executed.

It is relevant to point out that the behaviour of nodes emulates as closely as possible that of Bitcoin full nodes.

The simulation proceeds as a series of epochs. In each epoch a different victim is randomly chosen among honest nodes, while the Sybil nodes are still the same from epoch to epoch.

All the tests are a collection of simulations. From a simulation to another the number of Sybil nodes is increased by some proportion of the total number of nodes in the network.

In the second attack scenario, when Sybils are all on fresh bootstrap, the number of total attackers in the network has been increased. Each run adds 100 Sybil nodes and the whole simulation is composed of 300 runs, for a total of 30000 attacking nodes.

On top of that, Sybil nodes join sequentially the network in the same epoch. This

information is relevant to understand the results in Section 6.1.

6 Results

The results in the work of Serena et al. show how both Dandelion ++ and fixed probability gossip achieve high coverage even with an extremely high percentage of attackers in the network[27].

These results are confirmed with the following data obtained from new simulations shown in Fig. 6.1 and 6.2.

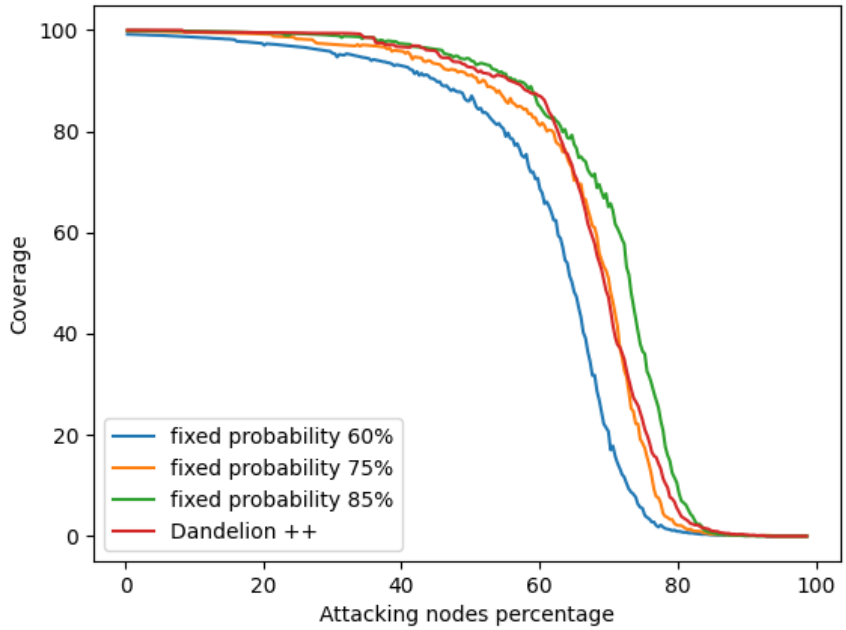

Fig. 6.1: overage results under standard Sybils behaviour with a network of 80000 edges

Changing the behaviour of Sybil nodes to disseminate bogus network information, as described in Section 5.1, yields the results shown in Fig. 6.3.

The outcome is similar for every protocol: coverage drops steeply for each of them, thus confirming the efficacy of the new behaviour.

Despite achieving the best coverage in previous results, even Dandelion ++ seems ineffective.

The following metrics are reported in order to study the topology of the new network and to assess how well the new behaviour of malicious nodes has worked.

The average degree of honest nodes reveals that each node is highly connected and has on average an higher degree than those in the results of Serena et al. (Fig. 6.4).

This further confirms the efficacy of the new attack, as previous results stated the importance of dense topologies as a mean to prevent Sybil attacks.

The average percentage of malicious neighbours is shown in Fig. 6.5. It is fundamental to evaluate the efficacy of the new Sybils behaviour, as the closer they get to eclipse honest nodes the higher the chance of carrying out a block-withholding attack would be.

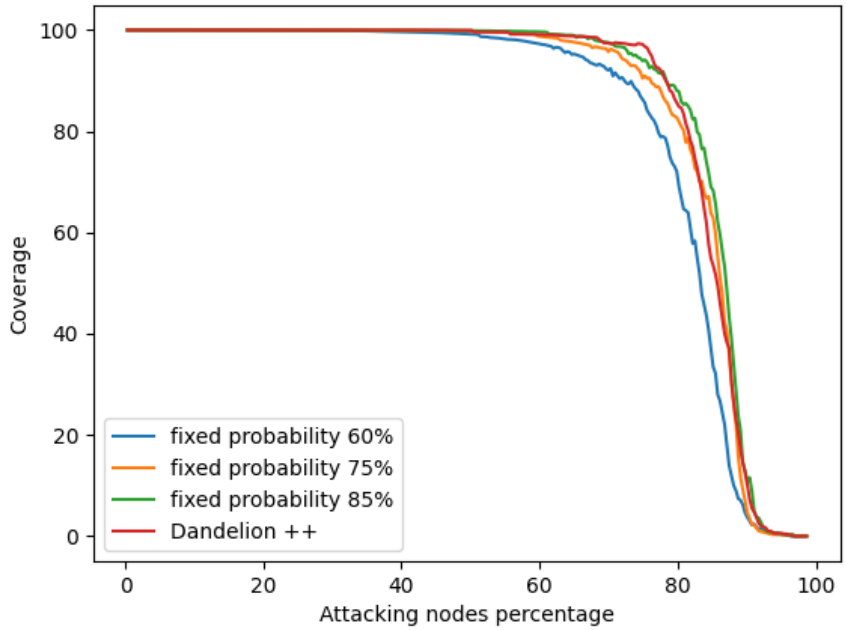

Fig. 6.2: Coverage results under standard Sybils behaviour with a network of 80000 edges

Furthermore, the reader can see how well false network information was spread in the graph shown in Fig. 6.6, as it depicts the percentage of malicious nodes in the peer cache of honest nodes.

The data can also be interpreted as the probability of a node connecting to a Sybil, since new connections are chosen uniformly at random from the cache.

The average degree of malicious nodes (Fig. 6.7) shows that Sybils are far more connected than honest nodes, especially when they are fewer. This is due to the lack of an upper bound on outgoing connections.

The descending trend of the degree is due to malicious nodes avoiding making connections between themselves. When the number of attacking nodes increases, that of honest nodes decreases proportionally, thus leaving fewer nodes to connect to.

With a samples of the degree distribution, Fig. 6.8, it is possible to understand how the topology of the underlying network was influenced.

Where there is a low percentage of Sybils in the network malicious nodes tend to have on average an extremely high degree.

As the percentage increases the tendency to create clusters emerges: some Sybils reach extremely high degrees, whilst others are left poorly connected.

Also DNSs have their role in determining the efficacy of the attack. The percentage of malicious addresses in their peer lists directly influences the chance of a node on fresh bootstrap connecting to a malicious node first. Fig. 6.9 shows a steady increase in the percentage of malicious addresses.

6.1 Alternative attack scenario

In the second attack scenario all malicious nodes are on fresh bootstrap and have to join the network while it is still forming. The number of attacking nodes in these simulations

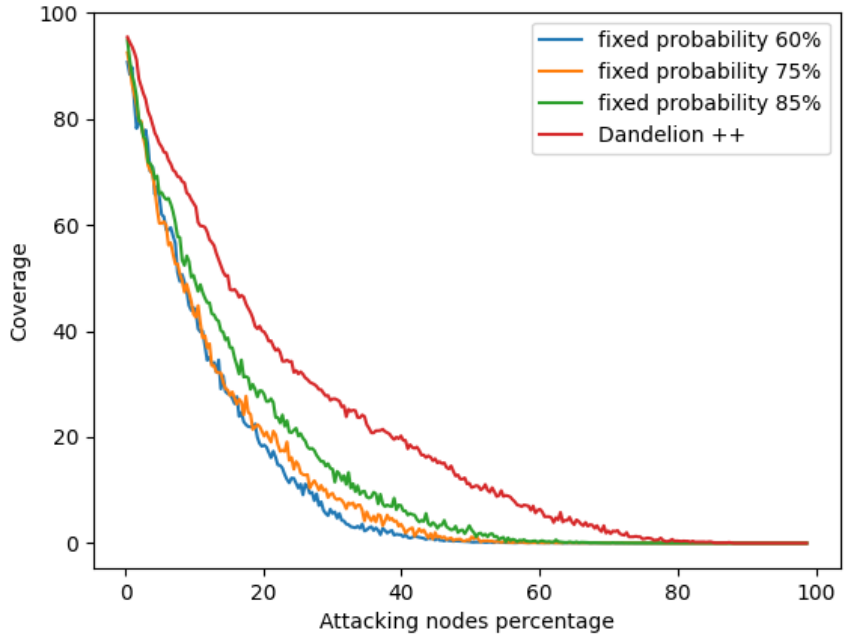

Fig. 6.3: Coverage results

has been increased up to 30000. A total of 300 simulations are run, 100 attacking nodes are added at each.

The attack does not achieve any results in this scenario, as shown by the coverage in Fig. 6.10. The network is thus resistant to such Sybil attack.

Also the the changes in the behaviour of Sybils have been ineffective, since rate of malicious neighbours is significantly lower (Fig. 6.11).

Despite that, Sybils still managed to disseminate malicious addresses well, although achieving results worse than those in the previous setting (Fig. 6.12).

Also the shape of the graph has changed. Sybils are less connected and face the same downward trend in the average degree, Fig. 6.13, as a result of the increasing number of attackers.

Moreover, by looking at a sample of the degree distribution, Fig. 6.14, the reader can see that only the Sybils that connect first (they connect sequentially, Sec. 5.3) are able to establish more connections, whereas the others are left in sparser areas of the graph.

In this scenario DNS peer lists did not have any attacker, since they cannot contain nodes on fresh bootstrap.

7 Conclusions

The way malicious nodes disseminate bogus network information makes the attack resemble fake bootstrapping and eclipse attack.

In particular, the adversary exploits `addr` messages exchanges and their relay policy to extensively disseminate false network information. Most `addr` messages are exchanged when peers setup connections, thus the attack efficacy is proportional to the number of

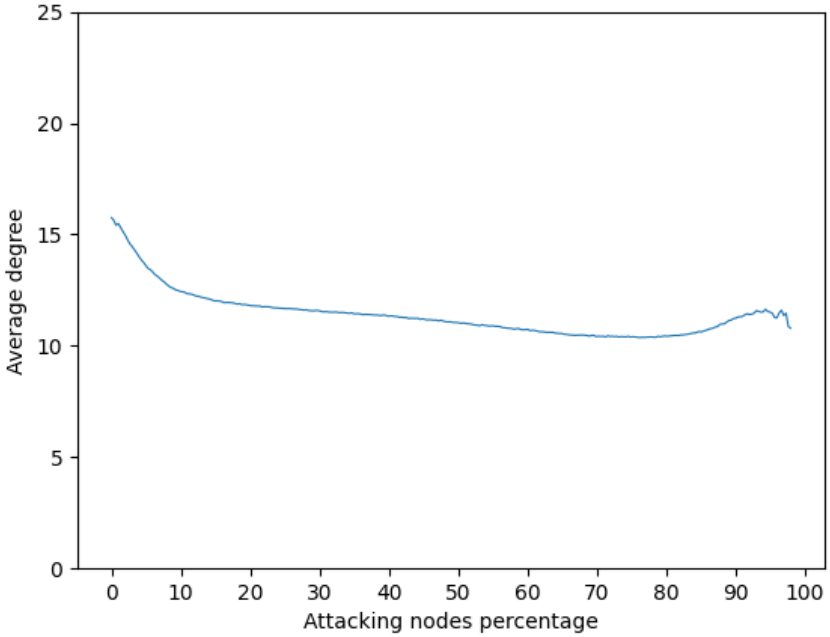

Fig. 6.4: Average degree of honest nodes

connections made over time.

On top of that, the adversary takes advantage Bitcoin's unique `addr` relay policy, that allows to further spread malicious addresses.

The efficacy of the strategy has been proven high, as with just 10% of the identities on the network the adversary can completely fill peer caches with malicious addresses (Fig. 6.6). Moreover, attackers are highly connected with honest nodes (Fig. 6.5.

The Sybil attack executed on the new overlay network is substantially more powerful, as it makes coverage drop steeply with less malicious nodes involved in the attack (Fig. 6.3).

It is necessary to point out that the new Sybil attack achieves high efficacy despite working with a network far more dense than the one in the results of Serena et al (Fig. 6.4). Their work assessed the importance of an highly connected graphs as a direct countermeasure to Sybil attacks of this kind.

Therefore, the results obtained imply an increased feasibility of the attack, since the adversary needs to control fewer identities to disrupt honest activities. With just the 20% of the nodes it is possible to make coverage drop below 40% (6.3).

Nonetheless, the setting of the attack is ideal and cannot be compared to that of a real network. The positive outcome is influenced by the increased capability to spread network information during the time the graph topology is still forming.

Under no circumstances the real Bitcoin network could undergo such process. For this reason the attack has been carried out on a second scenario where all the malicious nodes are on fresh bootstrap.

The results in Fig. 6.10 show how the attack was utterly ineffective even though the network was still being built at the time the Sybils started to join.

Furthermore, Sybils did not achieve their objective during network creation, since the average percentage of attacking neighbours (Fig. 6.11) is significantly lower.

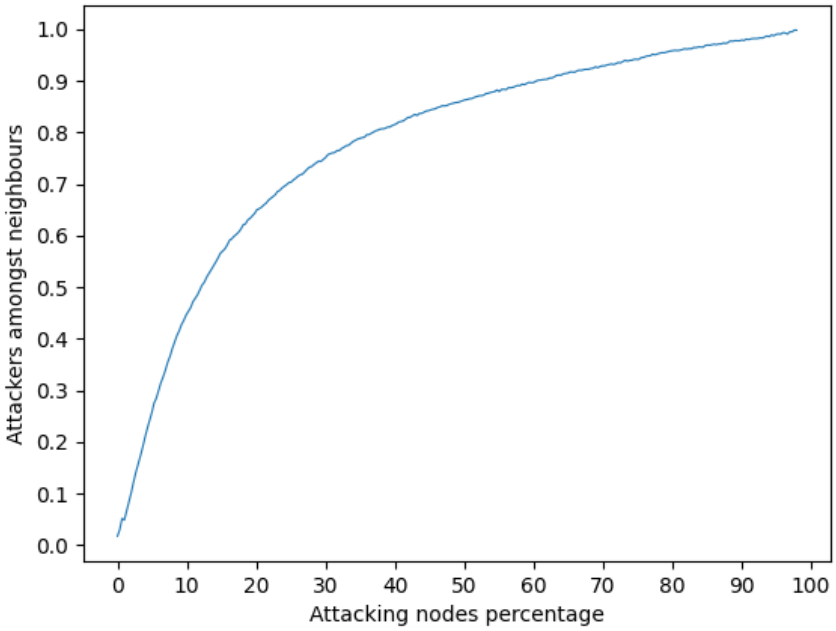

Fig. 6.5: Average percentage of Sybils among neighbours

Despite that, there is no gap that big in the rate of Sybils in the peer cache (Fig. 6.12) between the two attack scenarios, thus stating the attackers could still disseminate rather well their addresses.

This means that the positive outcome of the attack in the first scenario was influenced by two factors: the presence of Sybils in the DNS lists and the time at which malicious addresses start to be disseminated.

Although the first factor has been surely influential in later stages of the simulation, when the number of attackers increased, from Fig. 6.9 and 6.3 is possible to see how the steep decrease in coverage is not proportional to the increase in the percentage of attackers as DNS seeds.

On the other hand, the reader can now see how timing is essential for the good outcome of the attack, since starting the attack late allows honest nodes to form their own sparse network.

The data in Fig. 7.1, relative to the first attack scenario, shows the percentage of malicious addresses in peer caches before bootstrapping nodes start to join. In the second scenario, no Sybils can be in any peer cache at the start, since they have to join the network first.

It is evident that honest nodes have a good probability of connecting to malicious nodes as their first or second neighbour, with just a few Sybils in the network.

Whenever this happens their caches are filled with malicious addresses. From that point on the chance of connecting to another malicious address will be extremely high.

For this reason, the sooner a node connects to a malicious address the more likely it is that it will be eclipsed or have an high amount of Sybils amongst its neighbours.

This condition will further enhance the efficacy of the attack from then on: when a honest node connects to another, it is likely that it will send mostly malicious addresses to it, thus helping the adversary disseminating bogus information.

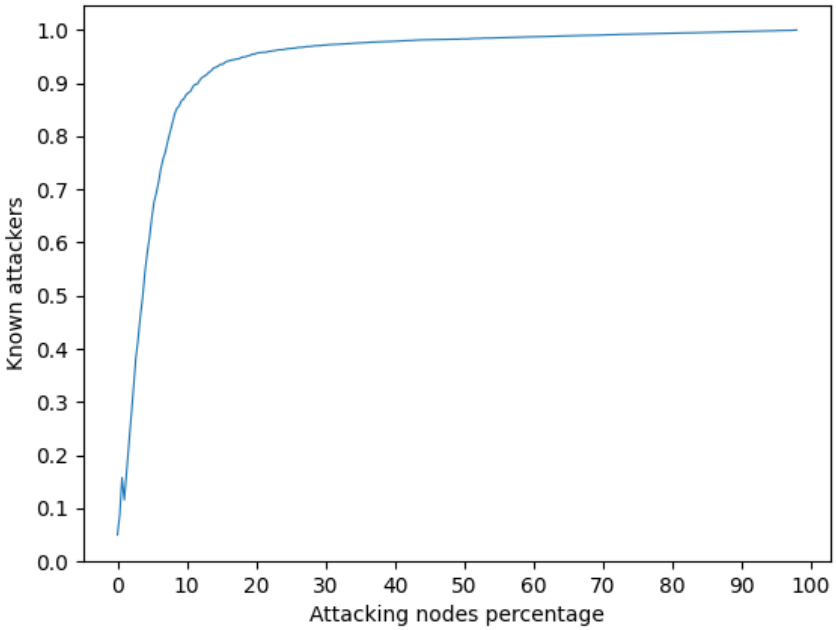

Fig. 6.6: Average percentage of Sybils in the peer cache

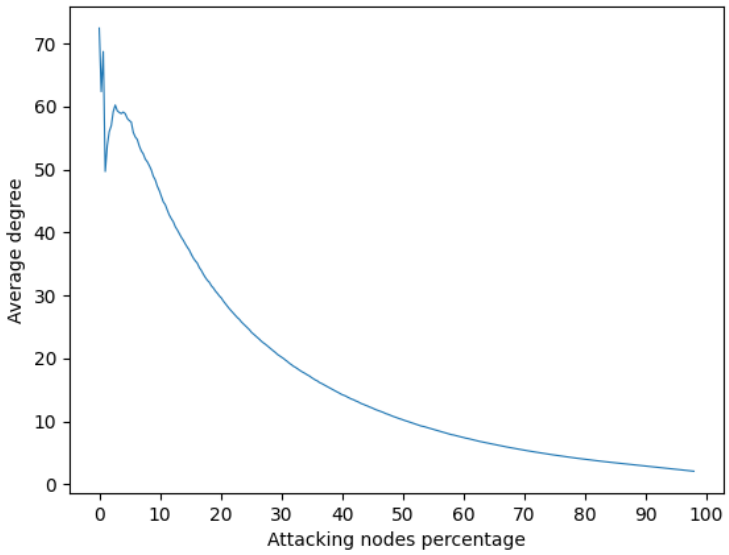

Fig. 6.7: Average degree of Sybil nodes

The sooner this chain of exchanges starts, the better for the attacker, as more and more nodes find only malicious addresses in their caches.

Although, the network develops differently in the second scenario. Most of the honest nodes have already at least one honest neighbour at the time malicious nodes start to join. Therefore it is likely that their second neighbour will be another honest node.

This creates a sparse honest-only subgraph that, despite being well connected with the Sybils subgraph, is entirely resistant to the attack.

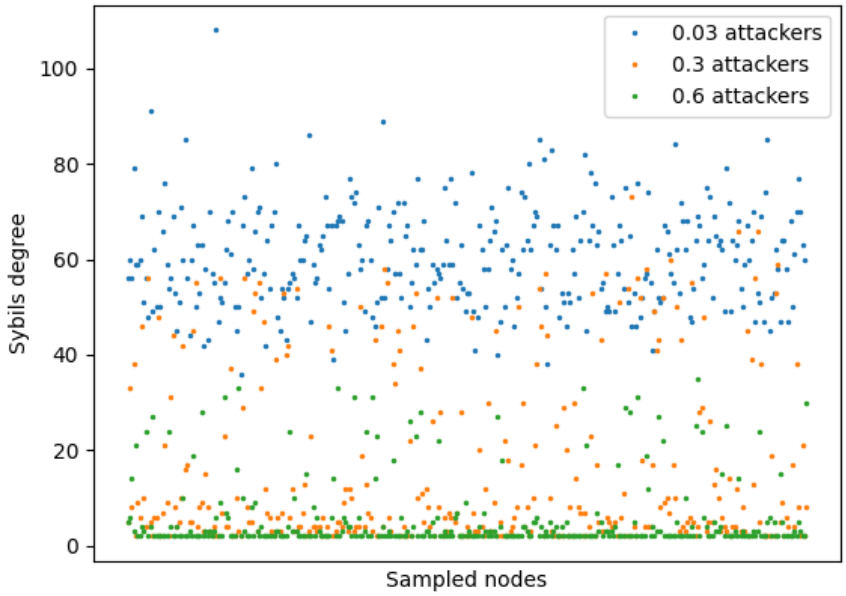

Fig. 6.8: Sample of degree distribution of Sybil nodes

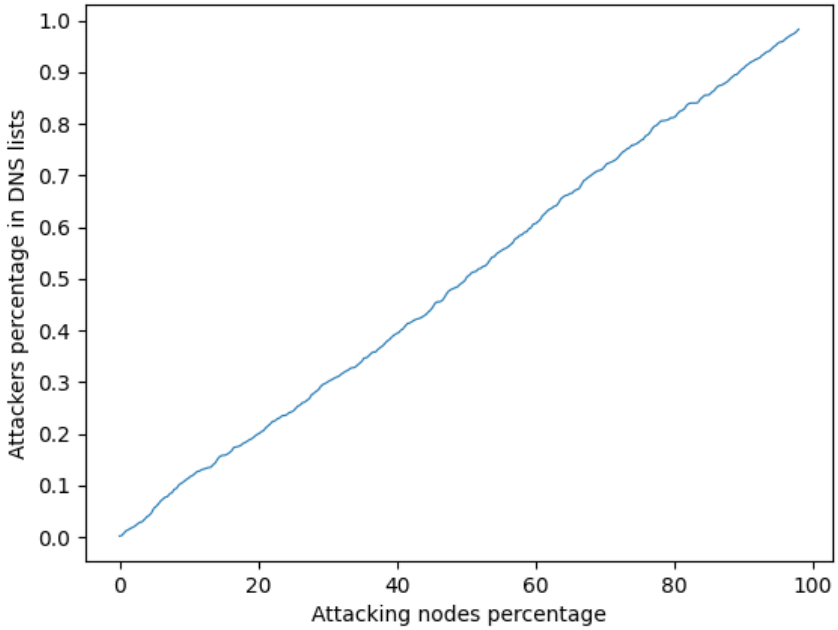

Fig. 6.9: Percentage of Sybils in the DNSs

Moreover, in this phase `addr` exchanges between honest nodes cannot contain Sybil addresses, as they are bootstrapping. This decreases the probability of connecting to a Sybil as second or third connection, therefore making the honest-only subgraph even more dense.

Lastly, malicious nodes cannot carry out fake bootstrapping, since their addresses are

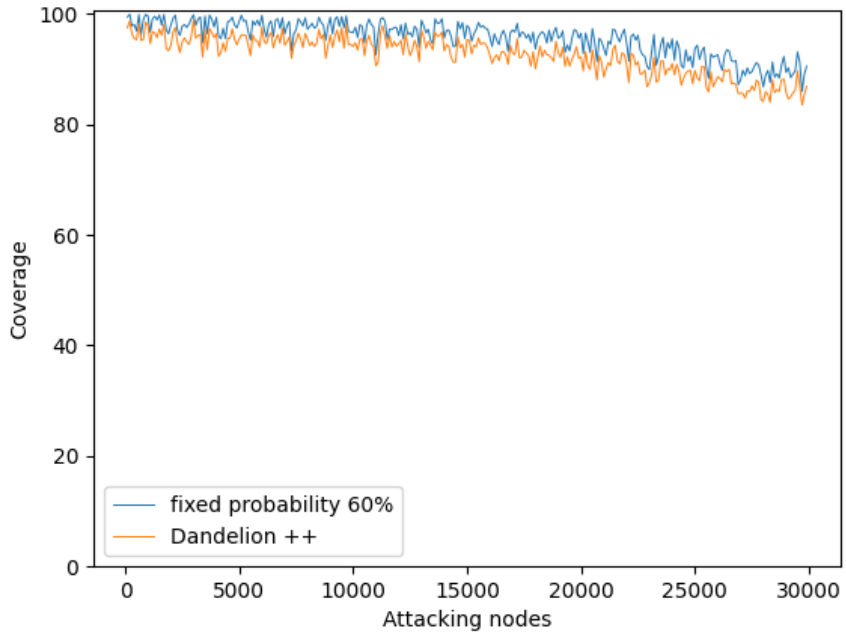

Fig. 6.10: Coverage with Sybils on fresh bootstrap (second scenario)

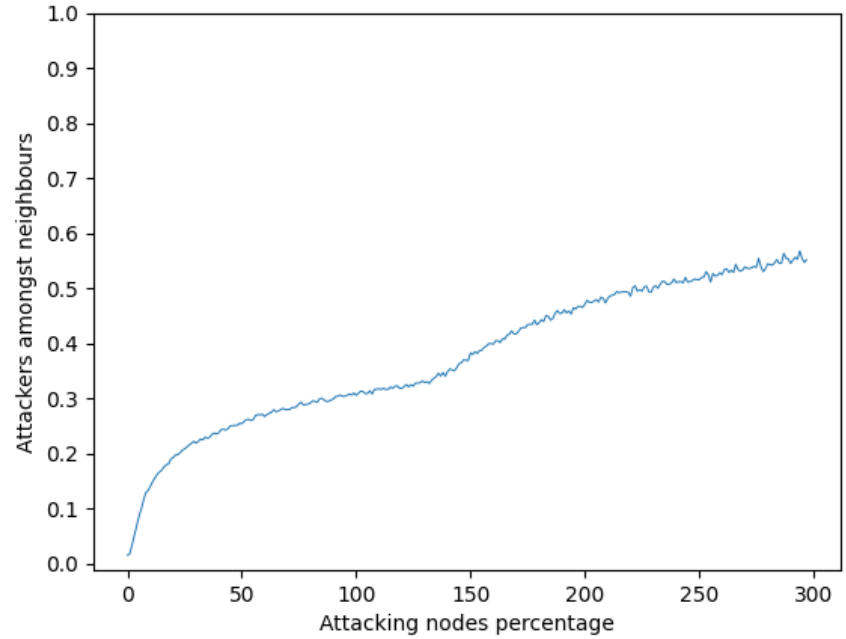

Fig. 6.11: Percentage of Sybil neighbours in the second attack scenario

not in DNS lists, and as a result bootstrapping honest nodes will surely contact other honest nodes first.

All of these reasons concurrently hinder the success of the attack. Despite succeeding at disseminating false network information, Fig. 6.12, the adversary can never make enough connections with honest nodes (Fig. 6.11). Thus, the data from the victim to the

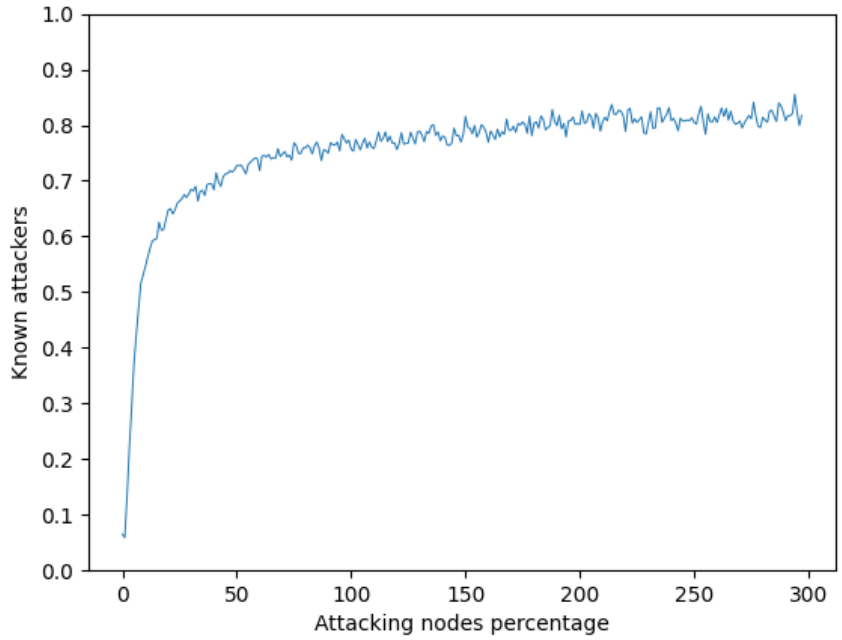

Fig. 6.12: Percentage of Sybils in the peer cache

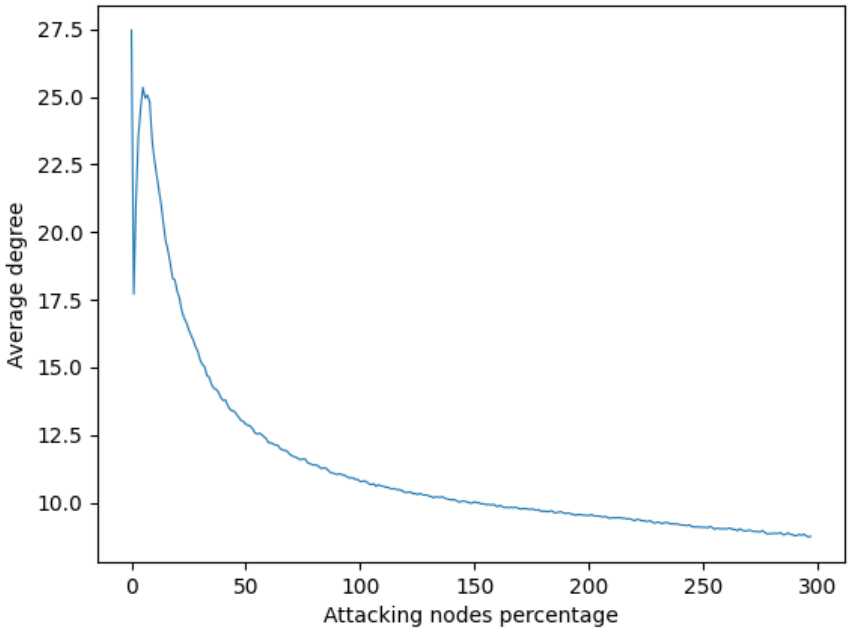

Fig. 6.13: Sybils average degree in the second attack scenario

rest of the network cannot be blocked.

The result holds even with and extremely large number of Sybils, that in the tests are as much as three times the total of honest nodes.

To conclude, the major factor for the success of the attack was the presence of Sybils in the network before the attack started. This has given a chance to the adversary to

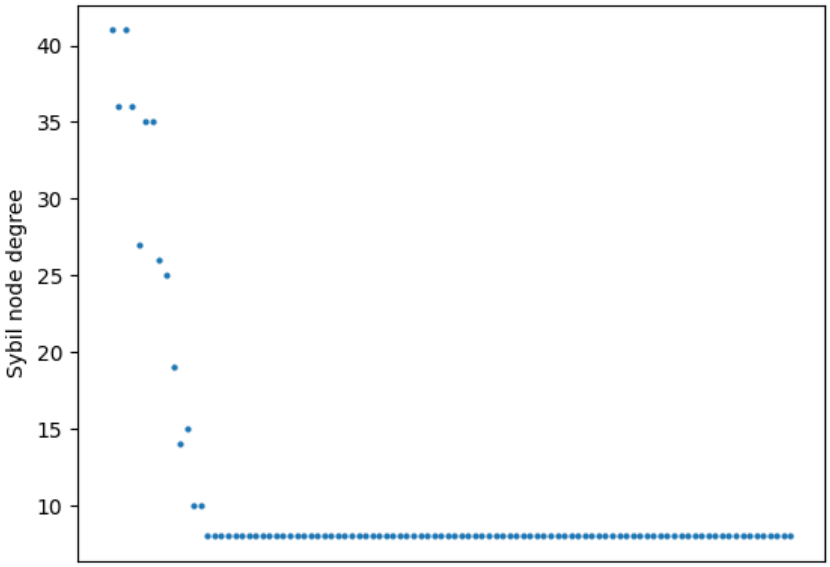

Fig. 6.14: Sample of Sybils degree distribution in the second attack scenario

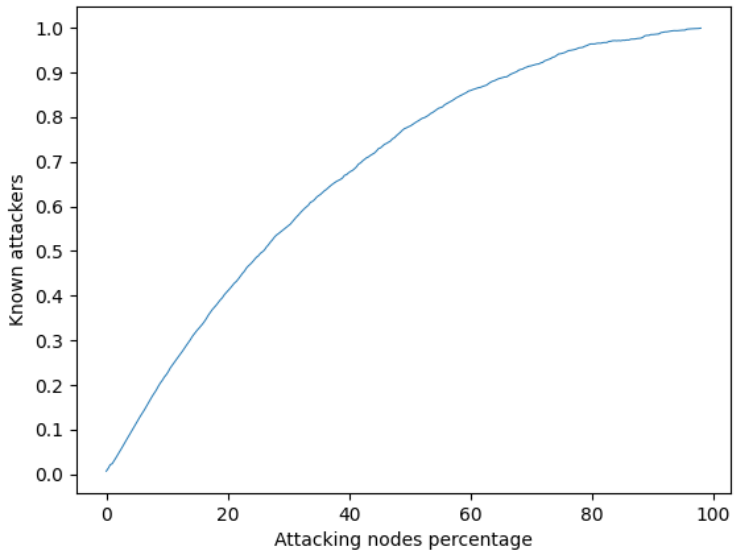

Fig. 7.1: Percentage of Sybils in the peer cache at the beginning of the graph construction in the first attack scenario

get inside DNSs seed lists and to start disseminating false information early on, thus completely changing the way the network forms.

Although it is likely that the attack would not be effective on the real Bitcoin network, this work has demonstrated how effective disseminating fake network information in the can be. This is particularly true if the adversary stays in the network long enough, thus disseminating fake information for longer and having a chance eclipse bootstrapping

victims.

The danger of fake bootstrapping for new, isolated nodes is real. An adversary that manages to keep these nodes isolated long enough may create multiple network partitions, or a single, larger partition that can be exploited for selfish mining or N-confirmations double-spending attacks.

New works can focus on an evaluation of the risks of fake bootstrapping for merchants and try to develop new secure means to join a peer-to-peer network.

Bibliography

[1] Bitcoin p2p network documentation. `https://developer.bitcoin.org/devguide/p2p_network.html`. Accessed: 2021-8-2.

[2] Bitcoin protocol documentation. `https://en.bitcoin.it/wiki/Protocol_documentation`. Accessed: 2021-8-2.

[3] Bitcoin source code. `https://github.com/bitcoin/bitcoin`. Accessed: 2021-8-2.

[4] Reachable bitcoin nodes. `https://bitnodes.io`. Accessed: 2021-8-21.

[5] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 15–29, New York, NY, USA, 2014. Association for Computing Machinery.

[6] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, and Pramod Viswanath. Dandelion: Redesigning the bitcoin network for anonymity. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1), June 2017.

[7] 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.

[8] 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.

[9] Varun Deshpande, H. Badis, and L. George. Btcmap: Mapping bitcoin peer-to-peer network topology. *2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, pages 1–6, 2018.

[10] 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.

[11] Maya Dotan, Yvonne-Anne Pignolet, Stefan Schmid, Saar Tochner, and Aviv Zohar. Survey on blockchain networking: Context, state-of-the-art, challenges. *ACM Comput. Surv.*, 54(5), May 2021.

[12] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[13] Gabriele D'Angelo and Stefano Ferretti. Highly intensive data dissemination in complex networks. *Journal of Parallel and Distributed Computing*, 99:28 – 50, 2017.

[14] 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`.

[15] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, June 2018.

[16] Giulia Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. *SIGMETRICS Perform. Eval. Rev.*, 46(1):5–7, June 2018.

[17] Mubashar Iqbal and Raimundas Matulevičius. Exploring sybil and double-spending risks in blockchain systems. *IEEE Access*, 9:76153–76177, 2021.

[18] 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.

[19] Michal Kedziora, Patryk Kozlowski, and Piotr Jozwiak. *Security of Blockchain Distributed Ledger Consensus Mechanism in Context of the Sybil Attack*, pages 407–418. 09 2020.

[20] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 469–485, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[21] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[22] Mahmoud Mostafa. Bitcoin's blockchain peer-to-peer network security attacks and countermeasures. *Indian Journal of Science and Technology*, 13:767–786, 02 2020.

[23] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008. Accessed: 2021-8-2.

[24] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 305–320, 2016.

[25] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, DaeHun Nyang, and David Mohaisen. Exploring the attack surface of blockchain: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(3):1977–2008, 2020.

[26] Sarwar Sayeed and Hector Marco-Gisbert. Assessing blockchain consensus and security mechanisms against the 51% attack. *Applied Sciences*, 9(9), 2019.

[27] Luca Serena, Gabriele D'Angelo, and Stefano Ferretti. Implication of dissemination strategies on the security of distributed ledgers. *CoRR*, abs/2007.15260, 2020.

[28] Ali Sunyaev. *Distributed Ledger Technology*, pages 265–299. Springer International Publishing, Cham, 2020.

[29] 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.

[30] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, 2016.

[31] 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.

[32] 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.

[33] Shijie Zhang and Jong-Hyouk Lee. Mitigations on sybil-based double-spend attacks in bitcoin. *IEEE Consumer Electronics Magazine*, 2020.