TorCoin

The power of distributed consensus on the Blockchain leveraged for Tor

1. Abstract

In this paper we introduce TorCoin, a distributed consensus protocol based on the Bitcoin block chain. This protocol will be used to establish new nodes on the network, and to determine node validity and bandwidth. TorCoin will run in collaboration with TorFlow, an existing code designed to determine bandwidth and monitor node behavior, using an RPC interface. To handle the computational costs of mining, we propose to partially outsource these costs to the Bitcoin network using the existing work sharing protocol. We discuss our implementation and provide an analysis of security concerns. Finally, we provide proof of concept, along with potential directions for future work.

2. Introduction

TorCoin is an altcoin used to achieve distributed consensus on the Tor network. It is designed to solve some of the scalability problems, specifically those related to the computational load on directory servers. In the current Tor protocol, the directory servers are trusted nodes which perform much of the administrative overhead, including regularly publishing a list of trusted nodes. As the size of the Tor network grows, it becomes increasingly more difficult for these nodes to make an accurate determination of which nodes are trusted on the time scale currently used. We propose using the Tor network to broadcast statements about their findings, and when something suspicious arises, the directory servers may intervene.

Using the block chain inevitably involves extra computational work, as validating a transaction has nontrivial cost. This problem will be solved via shared work with the Bitcoin network, and will be discussed further on. Onion Routers only need to download block headers to determine current network state but may opt in to the mining pool to increase security. An important distinction between the Tor network and the Bitcoin network is that the Tor network contains implicitly trusted nodes. These nodes, titled directory servers, are assumed to be trusted at all times. These trusted nodes allow us to remove some of the potential vulnerabilities of Bitcoin, which will be discussed further in Section 5.

3. Background

3.1. Tor Scaling Challenges

As the Tor network continues to grow, it is likely that the bandwidth, memory, or computational costs of keeping track of every node in the network will grow too large. TorCoin does not attempt to solve this problem. Instead, TorCoin solves problems related to specific resource bottlenecks: directory servers. Currently, directory servers keep track of all routers in the network and maintain this information is a constantly updated list. Additionally, these directory servers vote on the authenticity of each node in the network, and sign an agreed upon list, which is then routinely broadcast to the entire network. By using the collective resources of the network, the load on the directory servers can be drastically reduced.

In addition, distributing the entire Tor network topology using a blockchain means that an onion proxy, while it may not be able to examine the entire network at once, can be sure it can verify the authenticity of a node by query the blockchain on disk. This means fully distributed circuit creating designs, such as SALSA, may be considered again as viable options for future work. We discuss this in Section 6.3.

3.2. Leveraging the Blockchain

3.2.1. Distributed Consensus

The block chain is a ledger which maintains the full history of all transactions that have taken place on the network. This allows anyone to view the complete history of transactions at any time by requesting the block chain from the network and then simply reading through it. It does not, however, require that the entire block chain be in memory, which reduces network traffic and memory loads on nodes. The block chain is consistent across all nodes and therefore achieves consensus, even on a network of unknown size.

3.2.2. Proof of Work

Every transaction that takes place on the network must be authenticated. This is done through what is called a ‘proof of work’. The proof of work involves calculating the hash of the signed transaction information along with a nonce, a node chosen set of bits, such that the hash takes a specific form. Our knowledge of hash functions tells us that we cannot know the output for a given input a priori, and so the nonce must be guessed continuously until the hash meets the requirements. This makes creating a transaction non trivial to add to the block chain, but also makes it difficult to undo previous transactions once authenticated.

4. Design of TorCoin

4.1. Defining the Transaction

In our design of TorCoin, we modify the transaction format, as our consensus is unrelated to currency. Instead, we allow for three types of transactions: NODE_NEW, NODE_WARN, and NODE_AUDIT.

The NODE_NEW transaction is broadcast whenever a router wants to join the Tor network as a node. The node will post its IP address, public key, Tor version, and advertised bandwidth. This node, assuming that the IP address has not been used before, will then be added to the block chain as a valid node. A new node can replace an old node on the same IP address by broadcasting a NODE_NEW transaction with a new public key after a set amount of time has passed since the previous broadcast.

The NODE_WARN transaction is broadcast when a node suspects another node of providing false information or malicious behavior. A single warning transaction is not enough for any action to be taken, but given a sufficient amount of such transactions against a given node, the trusted directory servers will begin a node audit.

The NODE_AUDIT transaction is broadcast only by a directory server after completing a node audit. This audit is signed, and is only accepted by the network once verified that it was sent by a directory server. The audit can either change the observed bandwidth of a node, or deem a node malicious and ban it.

The following code snippet provides a pseudocode implementation of this transaction protocol.

```
TorTx {
        required public_key
        enum AdvertisementType {
                NEW_NODE
                NODE_AUDIT
                NODE_WARN
        }
        required AdvertisementType

        // params for NEW_NODE
        ip_address
        tor_public_key
        tor_version
        advertised_bandwidth

        // params for NODE_AUDIT
        signature
        observed_bandwidth
        node_validity

        // params for NODE_WARN
        node_identity
        reason_code
}

TorBlock {
        torcoin_version
        prev_block_hash
        difficulty
        timestamp

        repeated TorTx
        required merkle_root
        required nonce
}
```

```
SuperBlock {
        required TorBlock
        required BitcoinData
}


BitcoinData {
        // A Bitcoin format block header.
        required bytes header = 1;

        // The block's first transaction
        required bytes coinbase_tx = 2;

        // The merkle branch linking the
        // coinbase transaction to the
        // header.
        required bytes coinbase_merkle_branch
= 3;

        // The coinbase scriptSig
        // Contains bits of data in order.
        // Which one is our hash?
        required int coinbase_tx_index = 4;
}
```

4.2. Validating the Transaction

When a node broadcasts a transaction message to be added to a block, the authenticity of the sender and the validity of the contents must be verified. Before adding a received transaction message broadcast to the current working block, the node will perform any checks required, including public key validation.

4.4. Feasibility

In order to make this protocol viable, it must be the case that the additional computational costs are not excessive on Tor nodes, including those who opt-in to mining. Therefore, because mining is a computationally intensive activity, and because we do not plan to enforce participation by all nodes, the success of this protocol relies on the adoption of TorCoin in merged mining pools. Merged mining allows mining pools to opt-in to mining altcoins in

addition to Bitcoin. This would outsource much of the computational cost to the Bitcoin network, and increase the available computational power of the TorCoin network.

5. Security of TorCoin

Introducing the blockchain to Tor solves many scalability issues but, as with the introduction of any new codebase, brings with it all security vulnerabilities of the Bitcoin network. We will show that each of these vulnerabilities will not likely cause problems within the network. https://en.bitcoin.it/wiki/Weaknesses

5.1. 51% Attack

This attack takes advantage of the fact that with over half of the power of the network, an adversary can effectively bully the network into submission. For TorCoin and the Tor network in general, the definition of power is important to define. There are two attack vectors here: node count, and node computational power. Possession of over 50% of the nodes in the Tor network is something that should be either dealt with by the Tor protocol, and so is not discussed here further.

Possession of over 50% of the computational power in the Tor network is an entirely different issue. Tor is designed to provide anonymity in web browsing. The users and the node owners may be anyone, and the hardware is nonstandard. In the Bitcoin network, currency is at stake, and its inherent lucrative nature attracts users with powerful hardware. The TorCoin network needs to be able to handle the introduction of a computationally powerful adversary, something Tor does not necessarily worry about (although exceptionally high bandwidth capable nodes can cause some problems). Specifically, such an adversary should not be able to bully other nodes around or cause a DoS via the creation of excessive work for a node, in ways they were unable to do before.

Possession of 51% of the network's computational power allows for the attacker to force the network into ignoring any requests. By creating a new block chain fork which the attacker mines, this new fork will inevitably grow larger than the honest fork. This ultimately allows the attacker to ignore any set of transactions given enough time to for his malicious fork to catch up. In TorCoin, this can be abused to rescind or prevent any set of transactions from taking place. This is a serious problem, and emphasizes the need for merged mining adoption for this protocol to be secure.

5.2. Denial of Service

Both the Tor and Bitcoin networks are vulnerable to denial of service (DoS) attacks. While TorCoin does not mitigate these risks entirely, it does add resilience to the network as a whole.

5.2.1. Tor DoS Vectors

The Tor network relies on the directory servers being accessible at all times. These servers provide the required information about the Tor network topology to each node within the

network. The directory servers are responsible for both the generation of this information as well as the its distribution.

TorCoin still relies on trusted nodes to reach decisions about node validity and bandwidth but relies on the fully distributed blockchain for distribution. If an attacker were to shut down the trusted nodes the network topology would still be available on the blockchain. Updates to the network, however, still could not happen until a majority of trusted nodes were back online, as with the current directory server implementation.

5.2.2. Blockchain DoS Vectors

Because Bitcoin is distributed, only individual nodes are vulnerable to DoS attacks. These will not affect the health of the network as a whole but can be dangerous as the target node will fall behind the master blockchain and be left with an invalid view of the network. There are many safeguards built into the blockchain protocol to mitigate these attacks which we will not go into.

The TorCoin protocol deals with a very different transaction model than Bitcoin. Tor relies on valid information in the blockchain transactions to function and so the transactions become about much more than the blockchain itself. This new transaction model introduces three potential new DoS attack vectors, one with each transaction type. The `NODE_NEW` transaction is defined to be accepted only once per node during a given timeframe and thus any frequently repeated messages are rejected before they can make it into the blockchain. The `NODE_WARN` transaction is rate limited by unique node and thus also protected from making it into the blockchain. Lastly, the `NODE_AUDIT` transaction must be signed by a trusted TorCoin node in order to be accepted. Thus the TorCoin protocol introduces no new DoS vectors beyond those inherent in Bitcoin.

5.3. Timejacking

This attack relies on intentionally partitioning the network using fake timestamps, causing the target's nodes to only receive messages from the attacker's nodes. The target will believe that he has the true blockchain, but will instead have a fake. Once the attacker has convinced the target that he has been paid, the attacker allows the network to reconnect to the target nodes, and the target realizes he has not been paid at all. In Bitcoin, this is a problem because it can allow an attacker to double spend. With TorCoin, this would allow an attacker to convince a node that an attacker owned node had joined, or that an attacker owned node had put out a warning against it. This may allow an attacker to increase the chance that the target node routes traffic through the attacker's nodes, which would reduce the anonymity of the target. Because our current protocol assumes that all nodes are known, this issue could be resolved by asking random nodes for a time check at regular intervals.

5.4 Sybil Attack

The Sybil attack involves separating nodes from the network by limiting their ability to interact with other peers. The attacker can then use the isolated honest node to help him perform

other malicious activity. In the current TorCoin protocol, all nodes know all other nodes, and so this attack cannot occur.

6. Further Work

6.1. P2P Voting

With P2P voting, the burden of node audits is removed from the directory servers. In fact, with P2P voting, the need for directory servers is removed entirely. This will allow the network to be must more robust to attacks, as it cannot be brought down by DoS attacks to a small number of known targets. The use of P2P voting would further strengthen the 51% attack, as an attacker with 51% of the computational power would be able to shut down all honest nodes and take complete control of the network.

6.2. Hidden Services

Hidden service information could also be distributed among the network. This could be implemented using the block chain as the Tor Certificate Authority where hidden service introduction points and advertised via the blockchain.

6.3. Limited Network Knowledge

In a limited knowledge network, nodes cannot assume to know all other nodes. This would be implemented as an approach to reduce network bandwidth spend of learning about nodes which may never be used. This also allows for scaling into large networks at which point the nodes cannot store information about all nodes concurrently. However, this creates an attack vector for network partition attacks. Circuit management protocol would need to be changed so as not to reduce anonymity, but that change would be core to Tor and not TorCoin. Additionally, the Timejacking and Sybil attacks, both of which are non-issues in a complete knowledge network, would need to be addressed.

7. Conclusions

In this paper we introduced TorCoin, a block chain based consensus algorithm for use with the Tor network. We presented an implementation of the new transaction definitions, and discussed the security and anonymity concerns associated with it. The implementation of TorCoin shows promising results towards improving the scalability of the Tor network by reducing the load on the directory servers. Additionally, given sufficient participation from the Bitcoin community via merged mining, it should be possible to mitigate a 51% attack, the most notable attack vector which accompanies the block chain protocol.

8. Acknowledgements

Bitcoin, for possession of pathetic documentation, poor source code commenting, and effectively obfuscated C++.
Bitcoinj, for being readable and containing easy to navigate source code.

9. References

Alternate Chain - https://en.bitcoin.it/wiki/Alternative_chain
Namecoin - https://github.com/namecoin/namecoin
Ethereum - https://github.com/ethereum/wiki/wiki/White-Paper

10. Our Code

Wallet Viewer
torcoinj: https://github.com/torcoinj/torcoinj

TorCoin Client
torcoin: https://github.com/torcoinj/torcoin