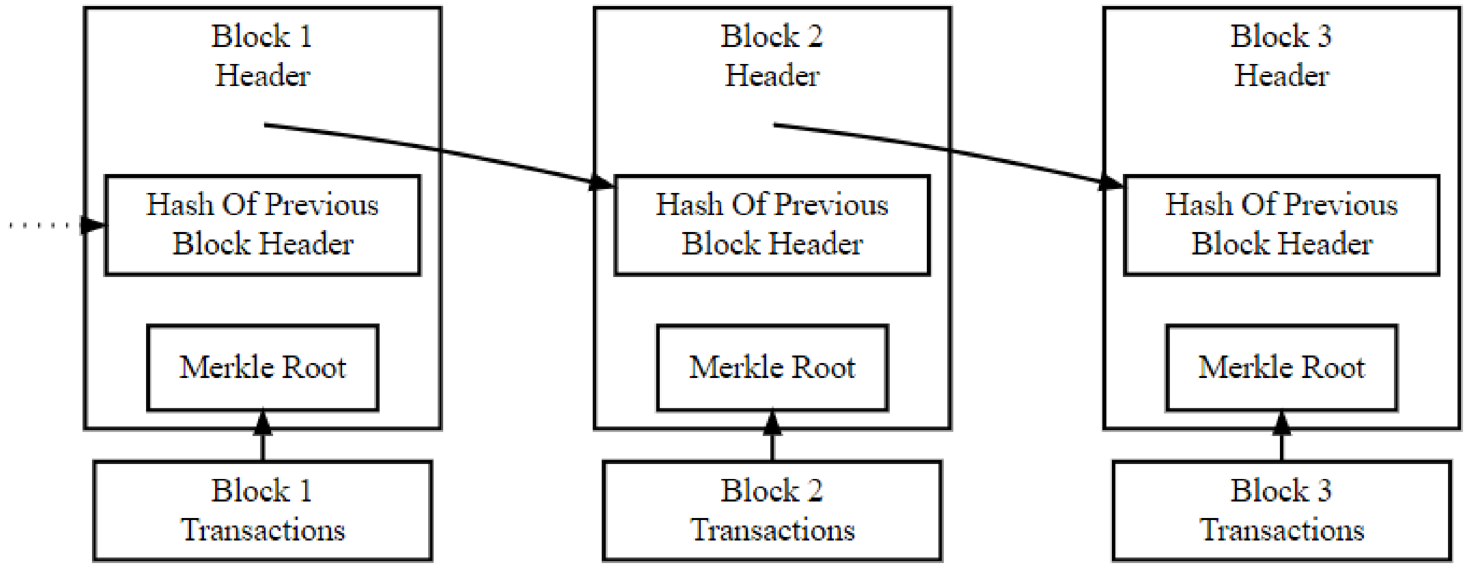
Securing IoT Configuration Deployments Using a Blockchain

ritcsec / January 16, 2019 / Uncategorized



Simplified Bitcoin Block Chain

By: Michael Rhodes

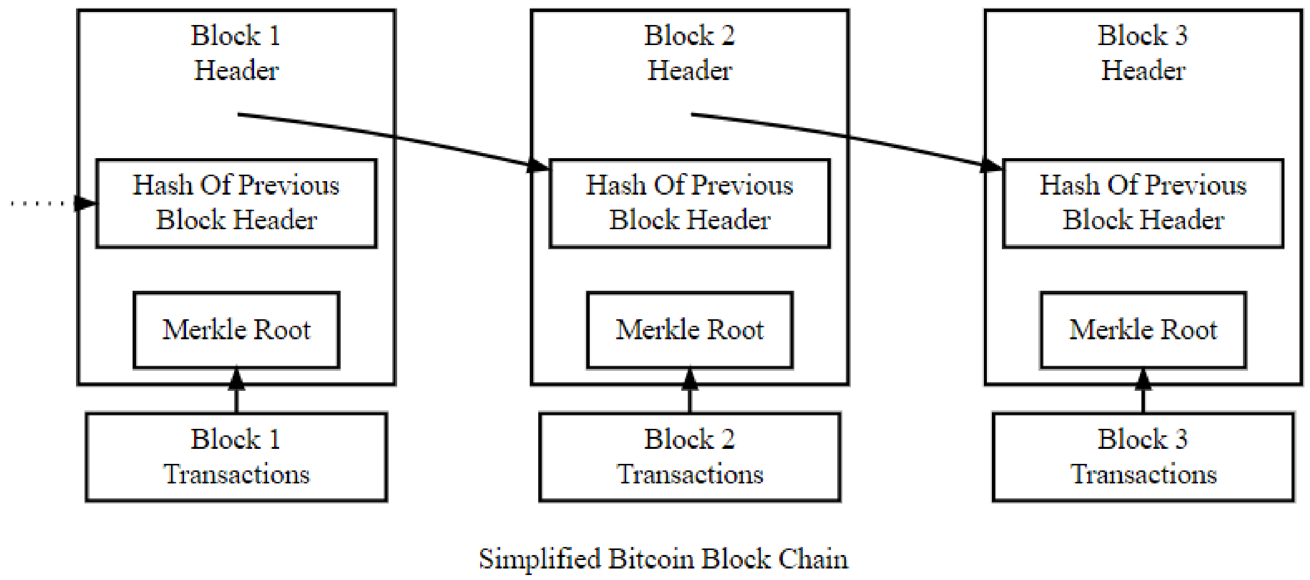
Since the dawn of Bitcoin, blockchain technology has been touted as the future of securing digital transactions. Due to its benefits over traditional methods of digital transactions, new applications for blockchain technology are being realized and implemented every day. These benefits include reduced fraud through increased transparency on transactions, no need for a central authority, reduced administrative costs through smart contracts, and a possible increase in speed and security of ownership of digital assets. My goal is to determine the efficacy of using blockchain technology for securing IoT configuration updates.

What is a blockchain?

Before we dive into this, it is important to understand the "magic" behind a blockchain. A blockchain is simply a distributed database of transactions. Each node participating in the blockchain has a copy of this database, which is called the ledger. The strength of a blockchain stems from the immutability of the ledger and the distributed trust model used to ensure the legitimacy of transactions.

The ledger and its components

The ledger is composed of blocks. Each block contains a list of valid transactions, timestamps, and info linking the current block to a previous block (chained blocks). The fundamental characteristic of the ledger relies on cryptographic hashes. All transactions in a block are hashed. In the case of Blockchain and Ethereum, these hashes are then aggregated into successive hashes to create a Merkle Tree (https://hackernoon.com/merkle-trees-181cb4bc30b4), where the root of the tree (the hash of all aggregated hashes) is placed in the block's header as the block's ID. Below is a picture of how blocks are chained together in the Bitcoin blockchain.



(source: https://bitcoin.org/img/dev/en-blockchain-overview.svg (https://bitcoin.org/img/dev/en-blockchain-overview.svg))

How are transactions agreed upon?

One of the biggest challenges in distributed computing is handling faulty or malicious nodes on the network. This is known as the Byzantine Generals Problem (BGP), and it consists of a group of generals, each commanding a portion of the Byzantine army. These generals are trying to formulate a plan for attacking a city in which they all surround via messengers. For the attack to be successful, all generals must agree on a common decision, as a halfhearted attack would be worse than either a coordinated attack or a coordinated retreat. The problem becomes complicated by the presence of a malicious general trying to convince the other generals to perform a halfhearted attack. This is known as a Byzantine failure, and a systems resistance to this is known as Byzantine Fault Tolerance (BFT). I found this (https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419) post very helpful with learning about BGP and how it relates to blockchains.

To overcome byzantine failures, a blockchain requires nodes to reach consensus before adding a new block to the blockchain. There are many different consensus algorithms; all of which aim to provide BFT. A great introduction to different consensus algorithm implementations can be found here (https://hackernoon.com/a-hitchhikers-guide-to-consensus-algorithms-d81aae3eb0e3).

Types of blockchains

There are three main methods in which a blockchain can be implemented:

- In an unpermissioned area – known as a public blockchain, every node can read and send transactions and can participate in the consensus process.
- In a consortium area – known as a partially permissioned blockchain, only defined nodes can participate in the consensus process. Reading and sending transactions can be restricted or public.
- In a permission area – only trusted nodes can write transactions into the blockchain. Therefore, consensus mechanisms are irrelevant. Private blockchains give up decentralization, and lack of necessity for a central authority, but still provide immutability.

Blockchains in IoT?

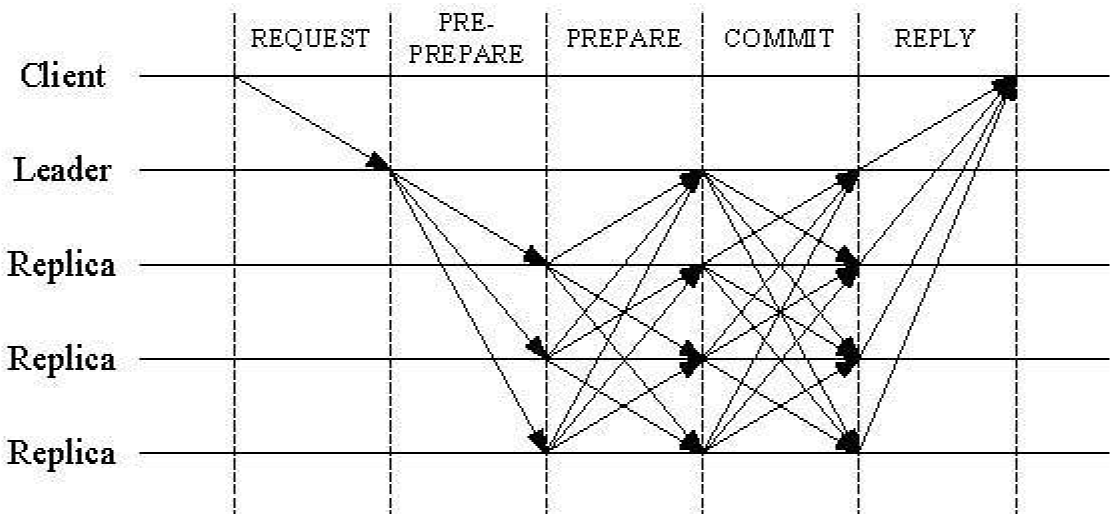
With the increasing relevance of IoT devices and their inherent security implications, it is important that administrators can quickly and securely push updates to their devices. This can become a major challenge when IoT deployments span thousands of devices. That is why I propose a blockchain be implemented to secure these transactions.

There have been several models proposed for how to implement a blockchain for IoT devices. One model involves each IoT device maintaining the ledger and participating in transactions. In this model each device would be provisioned with a private key to conduct transactions. With the limited resources available on most IoT devices, this implementation can be difficult. It could, however, work if the ledger was lightweight and the mining process was simplified. Another model involves the use of transaction nodes as intermediaries between the device and the blockchain. The IoT devices would communicate to the transaction nodes through an API, where the processing for the blockchain takes place.

Proof of Concept

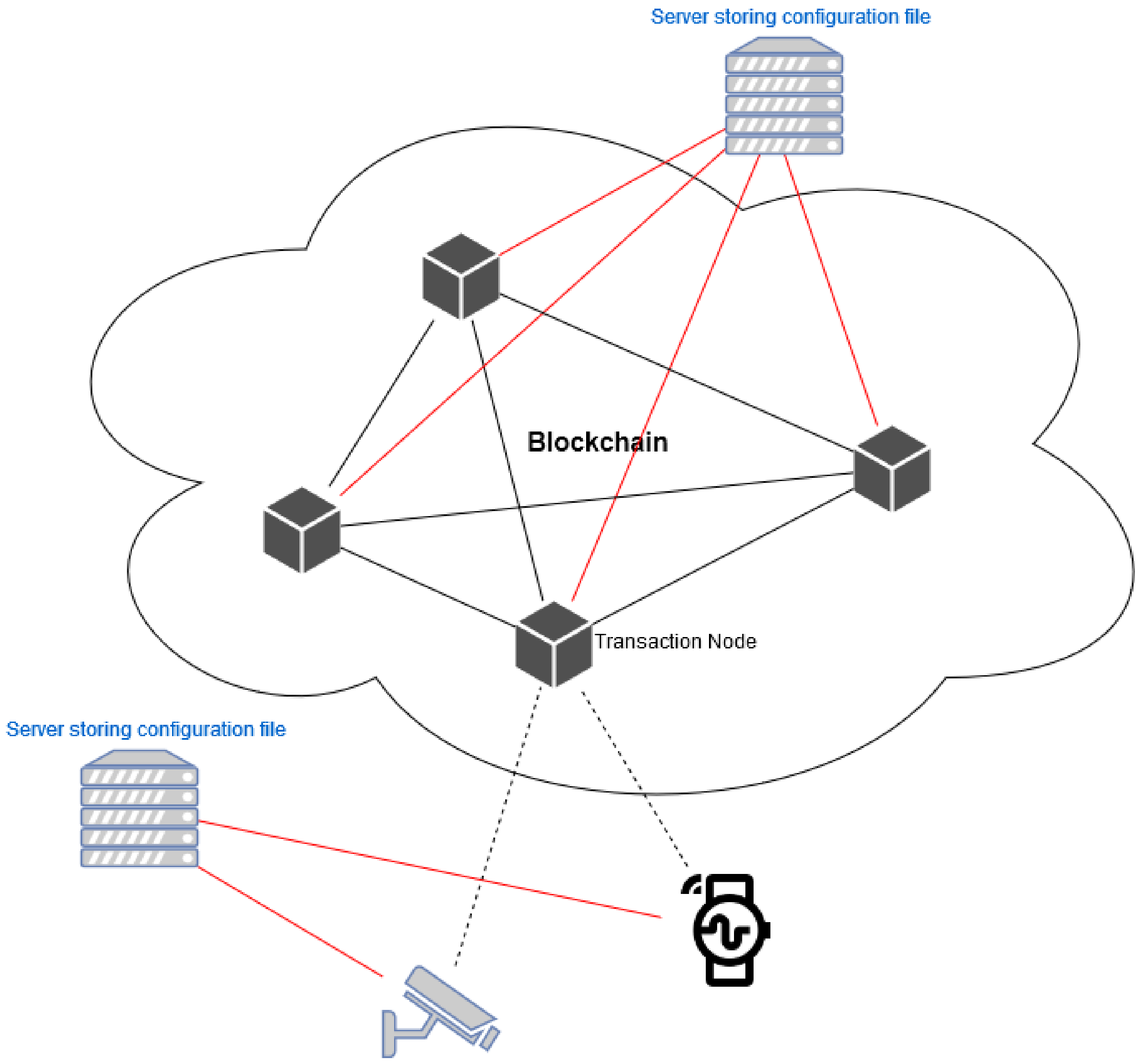
Source code: https://github.com/Michael-Rhodes/blockchain-IoT-PoC (https://github.com/Michael-Rhodes/blockchain-IoT-PoC)

To understand the effectiveness of using a blockchain for pushing configuration updates, I created a simple blockchain and a dummy IoT device using python. My implementation uses the practical byzantine fault tolerance (PBFT) consensus algorithm. This algorithm is implemented on several popular blockchains, including Hyperledger and Ripple. In short, this algorithm communicates between a group of 'generals'. Each general is responsible for sharing what it heard from the other generals, and once the message is agreed upon, each general validates the chain and shares its result to all other generals. Consensus is reached when greater than two-thirds of the generals agree on the chain. An example of the flow of communication can be seen below.



(source: https://itsblockchain.com/practical-byzantine-fault-tolerance-algorithm-pbft-consensus/ (https://itsblockchain.com/practical-byzantine-fault-tolerance-algorithm-pbft-consensus/))

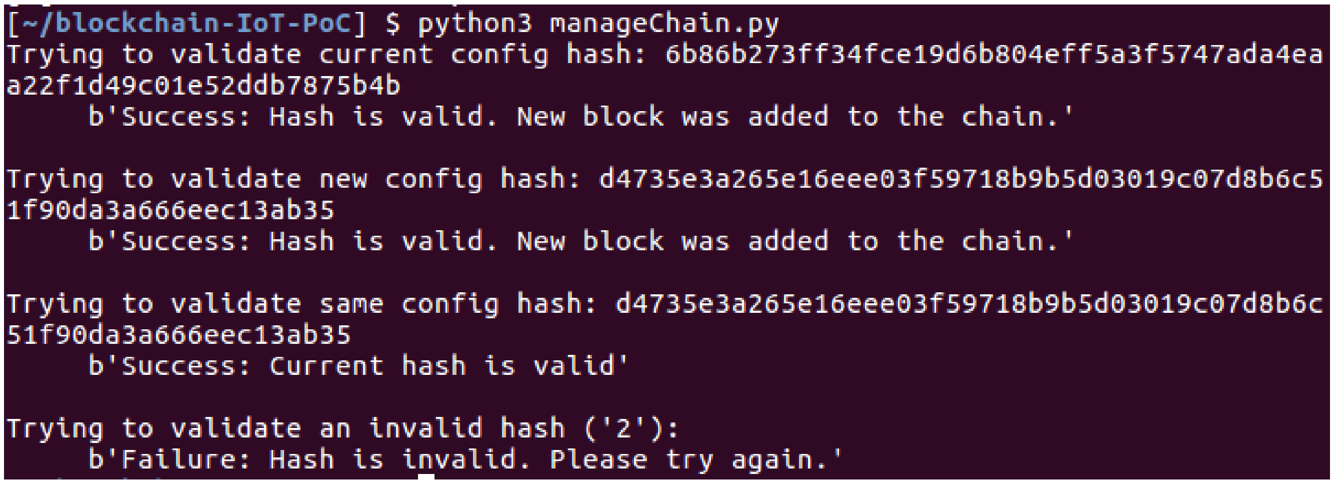
I decided to have all nodes on the network act as generals for consensus, since this is most likely to be implemented in a consortium or permissioned area. The blockchain is likely to be managed by a single organization and will want to control what nodes can read, write, and validate transactions.


Server storing configuration file
Blockchain
Transaction Node
Server storing configuration file

The test network consists of an IoT device, two configuration servers, and four nodes- the minimum required to reach consensus when a faulty/malicious node is present. The IoT devices in the network communicate with a separate configuration server than the nodes in the blockchain. This is to prevent a single point of failure in the system's trust model. For this implementation, I hardcoded the configuration servers into the scripts to simplify the code.

With this model, IoT devices can frequently download their configuration settings and create a hash of the data. They then communicate with a transaction node in the blockchain to validate the hash through an API. If the hash is in the last block in the blockchain, then the transaction node returns a success message. If the hash is not contained in the last block, the transaction node instantiates the PBFT algorithm by creating a new block and sharing it with all other nodes in the blockchain. If the consensus is an updated chain, then the transaction node returns a success message. Otherwise, a failure message is sent to the IoT device, prompting it to re-download the configuration file.

Below is the output of a few interactions with the blockchain. Two valid hashes and one invalid hash were supplied to the transaction node.



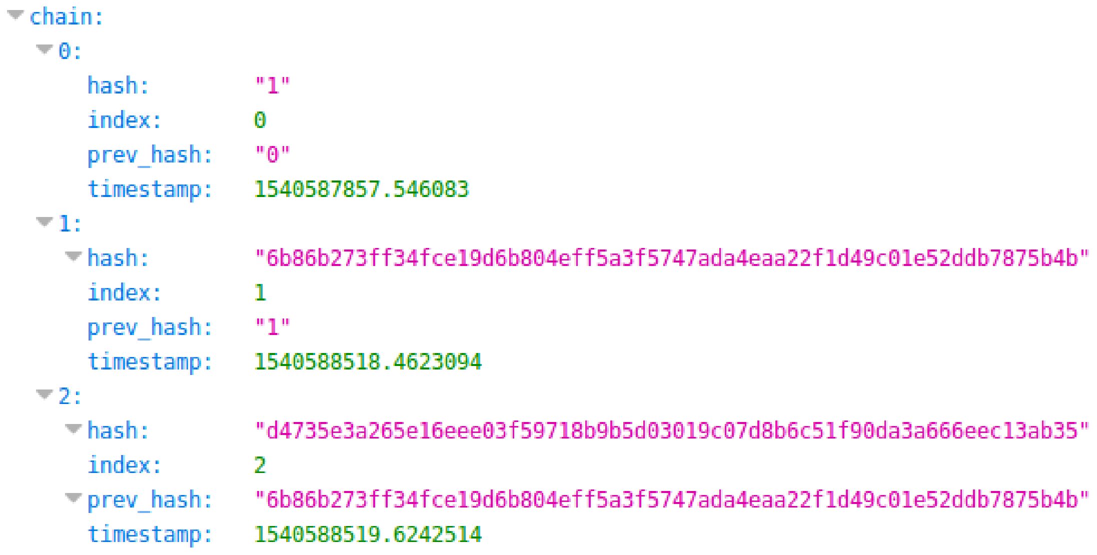
```
[~/blockchain-IoT-PoC] $ python3 manageChain.py
Trying to validate current config hash: 6b86b273ff34fce19d6b804eff5a3f5747ada4ea
a22f1d49c01e52ddb7875b4b
     b'Success: Hash is valid. New block was added to the chain.'

Trying to validate new config hash: d4735e3a265e16eee03f59718b9b5d03019c07d8b6c5
1f90da3a666eec13ab35
     b'Success: Hash is valid. New block was added to the chain.'

Trying to validate same config hash: d4735e3a265e16eee03f59718b9b5d03019c07d8b6c
51f90da3a666eec13ab35
     b'Success: Current hash is valid'

Trying to validate an invalid hash ('2'):
     b'Failure: Hash is invalid. Please try again.'
```

After the test, the ledger consists of three blocks: the genesis block and two containing valid hashes. The ledger is shown below.



```
chain:
  0:
    hash:        "1"
    index:       0
    prev_hash:   "0"
    timestamp:   1540587857.546083
  1:
    hash:        "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
    index:       1
    prev_hash:   "1"
    timestamp:   1540588518.4623094
  2:
    hash:        "d4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35"
    index:       2
    prev_hash:   "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
    timestamp:   1540588519.6242514
```

Conclusions

This model would be mainly beneficial for large deployments of IoT devices. While devices could be configured to pull configurations directly from a server and validate them using TLS, using a blockchain removes the root of trust from Certificate Authorities and distributes it across the blockchain. Since this model utilized transaction nodes to account for IoT devices with limited resources, it is important that only authenticated devices be allowed to communicate with the transaction nodes. I recommend a multi-factor approach for authentication, such as whitelisting and mutual authentication.

Not only does this model redistribute trust, it also allows administrators to accurately audit current and past configurations, due to the immutability of the ledger. Furthermore, this blockchain could be expanded to include the configuration files within the block, allowing administrators to easily retrieve old configurations and audit the differences.

This technology has several other practical uses for enforcing security in the Internet of Things. It could also be used for securing firmware updates, providing scalable device discovery, and to track communication between trusted nodes. Overall, this was a great learning experience. It really helped demystify the intricacies of blockchain technology.

Published by ritcsec

View all posts by ritcsec