Chainpoint

A scalable protocol for anchoring data in the blockchain and generating blockchain receipts

Authors: Wayne Vaughan, Jason Bukowski, Shawn Wilkinson

Contributors: Manu Sporny, Ryan Shea,
Christopher Allen, Paul Storcz, Jude Nelson

August 31, 2016
v2.1.0

Abstract

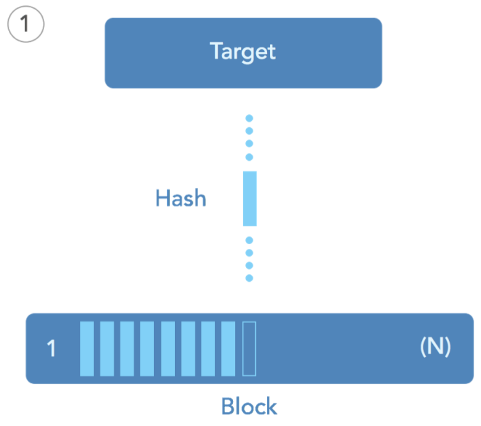
A standard for maximizing the scalability of anchoring data in the blockchain and generating blockchain receipts. Each receipt contains the information needed to verify the data without relying on a trusted third party. The original Chainpoint 1.0 specification has been updated based on a year of learning.

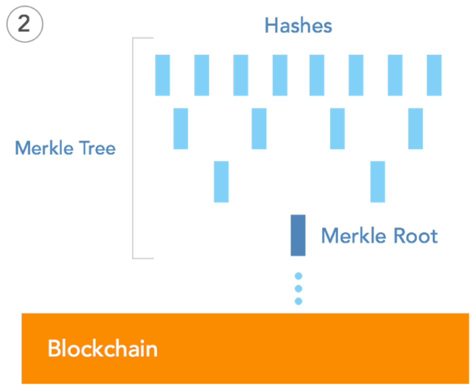
Introduction

The use of the Bitcoin blockchain [1] to timestamp and verify data in an immutable public ledger was pioneered by Manuel Aráoz with the creation of Proof of Existence [2]. This system, and others like it, notarize data in the blockchain by publishing a hash of the data in a Bitcoin transaction. By comparing the hash published in the blockchain with the hash of some data, it's possible to verify that the data existed at a specific time. At the time of writing, Bitcoin can handle approximately five transactions per second and each transaction costs approximately $0.10 USD [3]. These limitations make it impractical and cost prohibitive to anchor large volumes of data in the Bitcoin blockchain. What is needed is a scalable method to anchor data in the blockchain and a standard protocol that allows systems to read and verify the data.

Anchoring Data

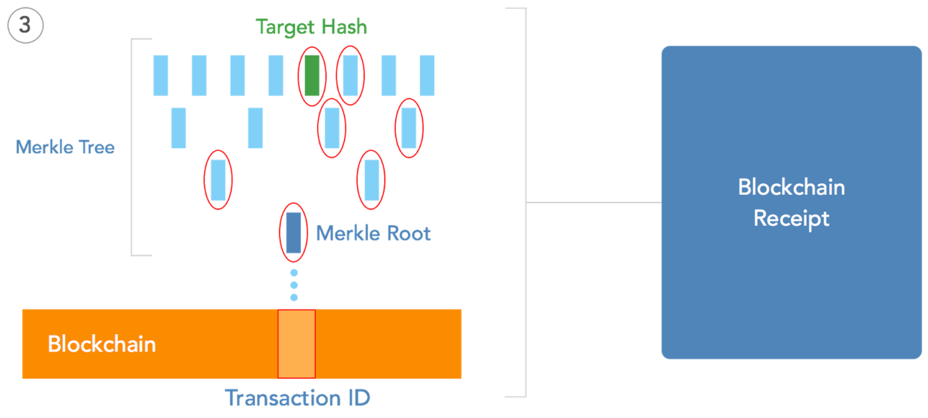
To anchor data in the blockchain, we start by using a standard hashing function such as SHA-256 to generate a unique hash of the target data. Multiple hashes are assembled into a block, which is simply a list of hashes. Periodically, these blocks are used to generate a Merkle Tree [4], and the Merkle Root is published in the blockchain via a transaction. By collating multiple hashes into a Merkle Tree and publishing the Merkle Root, we can anchor large volumes of data in the blockchain using a single transaction.





Creating Blockchain Receipts

In the real world, a receipt provides proof of a transaction. A ***blockchain receipt*** provides proof that some data existed at a specific time. It contains all the information needed to prove an individual hash was part of the Merkle Tree whose root was published in a transaction in the Blockchain. By tracing a path from the Merkle root to the target hash, we can generate a Merkle Proof that proves any one of the elements is in the Merkle tree, without having to know the entire tree. These elements can be used to create a blockchain receipt that contains, at minimum, the Target Hash, Merkle Proof, Merkle Root, and Transaction ID.



Blockchain Receipt Standard:

Chainpoint 2.0 receipts are JSON-LD compliant. A full description of all Chainpoint versions is available at http://chainpoint.org

@context	The JSON-LD Context of the document
type	The type of Chainpoint Receipt being described
targetHash	The hash value being anchored in hex string format
merkleRoot	The merkle root of the tree in hex string format
proof	An array of hash objects connecting targetHash to merkleRoot
anchors	An array of methods employed to anchor data to blockchain(s)

The anchors array contains one or more anchor objects.

type	The type of anchoring being performed
sourceId	The Id used to locate the anchored valued for the given source

JSON-LD example of a Chainpoint 2.0 receipt:

```
{
  "@context": "https://w3id.org/chainpoint/v2",
  "type": "ChainpointSHA256v2",
  "targetHash": "bdf8c9bdf076d6aff0292a1c9448691d2ae283f2ce41b045355e2c8cb8e85ef2",
  "merkleRoot": "51296468ea48ddbcc546abb85b935c73058fd8acdb0b953da6aa1ae966581a7a",
  "proof": [
    {
      "left": "bdf8c9bdf076d6aff0292a1c9448691d2ae283f2ce41b045355e2c8cb8e85ef2"
    },
    {
      "left": "cb0dbbedb5ec5363e39be9fc43f56f321e1572cfcf304d26fc67cb6ea2e49faf"
    },
    {
      "right": "cb0dbbedb5ec5363e39be9fc43f56f321e1572cfcf304d26fc67cb6ea2e49faf"
    }
  ],
  "anchors": [
    {
      "type": "BTCOpReturn",
      "sourceId": "f3be82fe1b5d8f18e009cb9a491781289d2e01678311fe2b2e4e84381aafadee"
    }
  ]
}
```

Receipt Types:
Chainpoint 2.0 supports the following Secure Hashing Algorithm types.

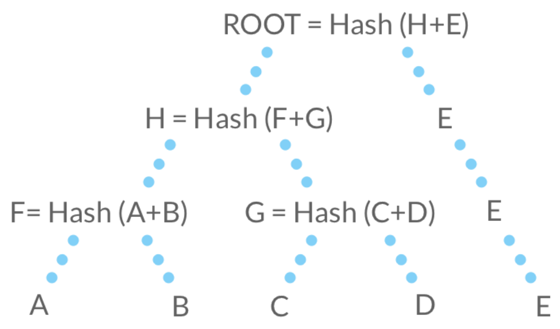
ChainpointSHA224v2	Chainpoint 2.0 receipt using SHA-224
ChainpointSHA256v2	Chainpoint 2.0 receipt using SHA-256
ChainpointSHA384v2	Chainpoint 2.0 receipt using SHA-384
ChainpointSHA512v2	Chainpoint 2.0 receipt using SHA-512
ChainpointSHA3-224v2	Chainpoint 2.0 receipt using SHA3-224
ChainpointSHA3-256v2	Chainpoint 2.0 receipt using SHA3-256
ChainpointSHA3-384v2	Chainpoint 2.0 receipt using SHA3-384
ChainpointSHA3-512v2	Chainpoint 2.0 receipt using SHA3-512

Anchor Types:
Chainpoint 2.0 supports the following anchor types. Additional anchor types are under development.

BTCOpReturn	Anchored to a Bitcoin transaction within an OP_RETURN output
ETHData	Anchored to an Ethereum transaction using the data field

Merkle Tree Construction

When constructing merkle trees from which proofs will be generated for Chainpoint receipts, lonely leaf (odd) end nodes on any given level should be promoted up to the next level, as depicted in the following diagram:



Chainpoint requires that hashes used in building merkle trees and proofs be handled internally in binary form only. When concatenating hashes and hashing those results, do not use the hex strings. The hex string representation of hashes is only used for displaying the hash values within the receipt.

Verifying Blockchain Receipts

1. Concatenate targetHash and the first hash in the proof array. The right or left designation specifies which side of the concatenation that the proof hash value should be on.
2. Hash the resulting value.
3. Concatenate the resulting hash with the next hash in the proof array, using the same left and right rules.
4. Hash that value and continue the process until you've gone through each item in the proof array.
5. The final hash value should equal the merkleRoot value
6. Ensure that the merkleRoot value is stored in the transaction specified in the anchors array. In the case of type 'BTCOpReturn', ensure that the BTC transaction with the id of sourceId has the merkleRoot stored in the OP_RETURN field

Storing Blockchain Receipts

Chainpoint receipts can be stored in a centralized database or a decentralized system such as Storj or IPFS.

Anchoring In Multiple Locations

The Chainpoint protocol can support multiple blockchains. The Merkle root for a blockchain receipt could be stored in Ethereum, Factom, or any other immutable and globally persistent data store.

Conclusion

We have outlined a scalable protocol for anchoring data in the blockchain and generating blockchain receipts. A description of the Chainpoint protocol is available at http://github.com/chainpoint.

References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, (2009). https://bitcoin.org/bitcoin.pdf.

[2] M. Araoz. What is Proof of existence?, (2014). http://www.proofofexistence.com/about.

[3] "Transaction Fees." Bitcoin Wiki, (2016). https://en.bitcoin.it/wiki/Transaction_fees

4] R.C. Merkle. Protocols for public key cryptosystems, (April 1980). IIn Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133.