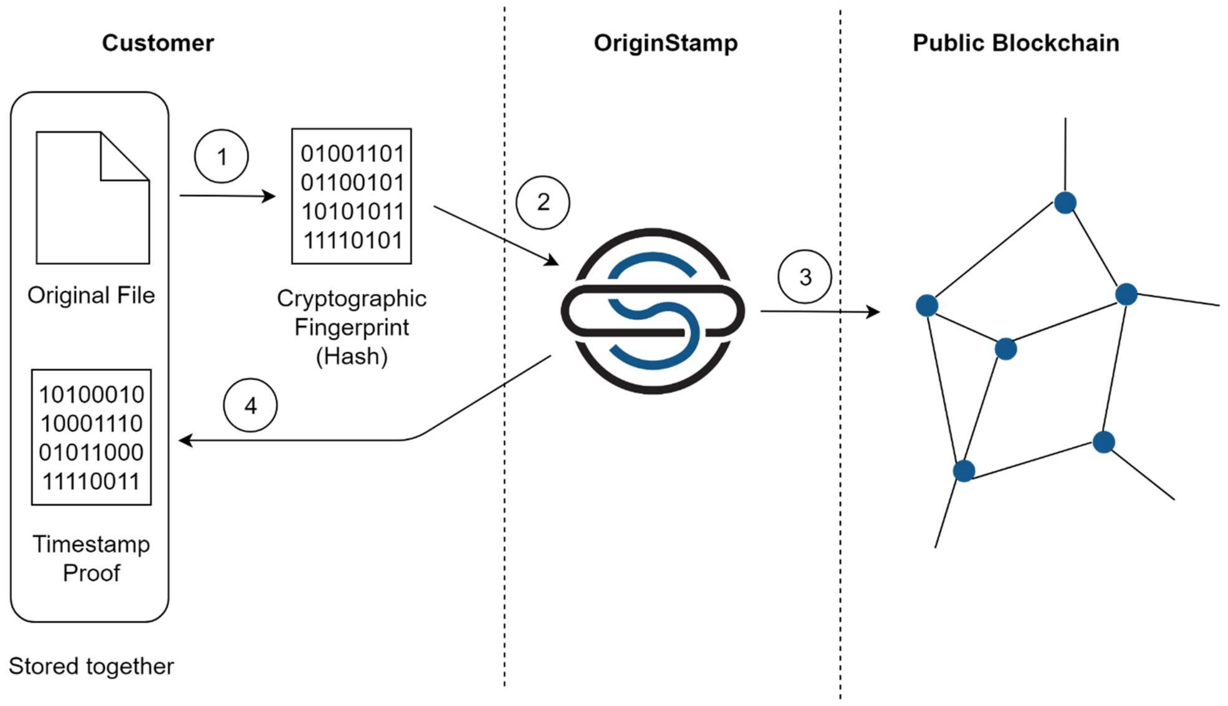
OriginStamp Whitepaper

May 22, 2019

OriginStamp AG, Rothausstrasse 1, 8280 Kreuzlingen, Switzerland

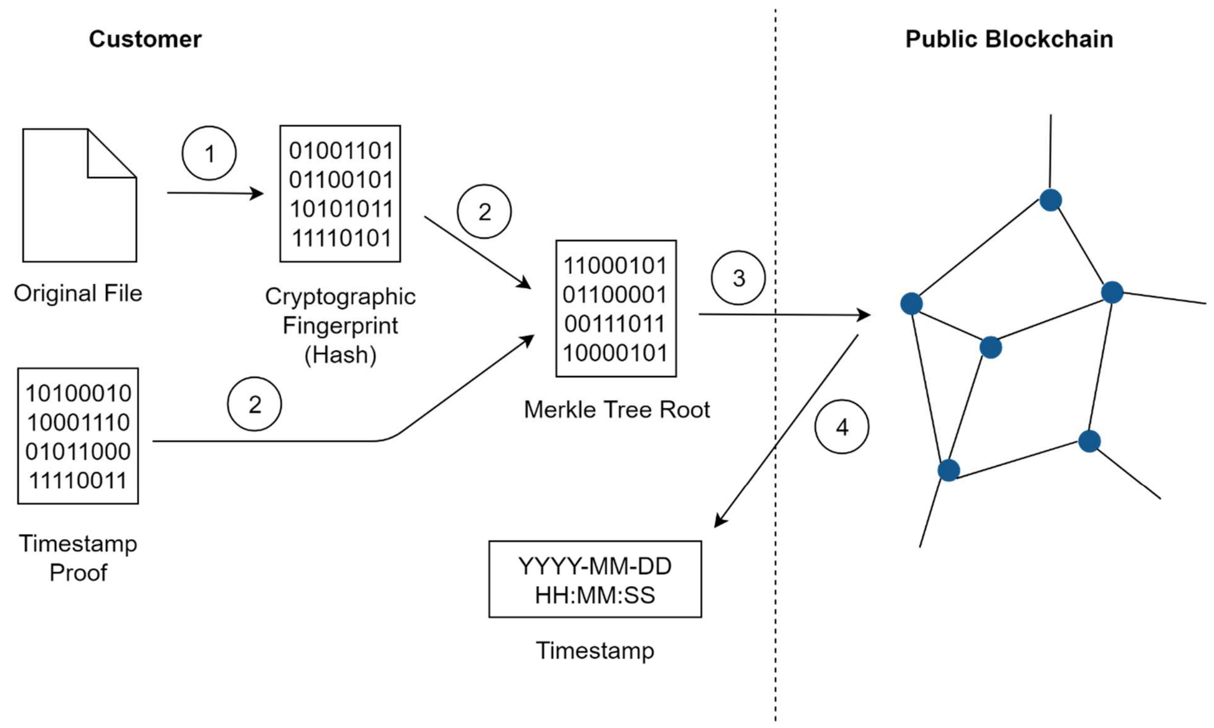
Contents

Creating a timestamp



1. The customer uses the publicly available SHA-256 hashing function (https://en.wikipedia.org/wiki/SHA-2) to create a hash of the original file. This can be achieved by using various online calculators or by calling libraries for computer programs.
2. The resulting hash is passed to OriginStamp via the OriginStamp API. This requires a valid API key.
3. During a fixed time interval, OriginStamp collects all incoming hashes from various customers. These hashes are then treated as leaves in a Merkle tree (https://en.wikipedia.org/wiki/Merkle_tree). The root of this tree is then written to a public blockchain. The exact way how this is accomplished, varies between the different blockchains (see below). The point in time where the root of the Merkle tree is written to the public blockchain is the tamper-proof timestamp of the file of which the customer submitted the hash.
4. Finally, OriginStamp returns a proof that the timestamp was created to the customer. It consists of a subset of the Merkle tree that is needed to verify the timestamp independent of OriginStamp. **It is important that the customer stores the original file and the timestamp proof together and ensures that both data files do not change.**

Verifying a timestamp



1. The customer uses the publicly available SHA-256 hashing function (https://en.wikipedia.org/wiki/SHA-2) to create a hash of the original file. This can be achieved by using various online calculators or by calling libraries for computer programs.
2. The customer takes the timestamp proof corresponding to the original file and verifies that the hash of the original file is contained in the timestamp proof. Then, the customer verifies the proof by verifying the contained subset of the Merkle tree.
3. The customer retrieves the date and time when the root of the Merkle tree was included into the public blockchain. The exact way how this is accomplished, varies between the different blockchains (see below).
4. The point in time where the root of the Merkle tree is written to the public blockchain is the tamper-proof timestamp of the file of which the customer wanted to verify the timestamp. If the described procedure fails, the timestamp cannot be verified.

A detailed step by step walkthrough can be found here:
https://github.com/OriginStampTimestamping/originstamp-verification

Submitting / Verifying a Merkle tree root

The process how the root of a Merkle tree is written to a public blockchain varies by the type of the blockchain.

Bitcoin

For the Bitcoin blockchain the Merkle tree root is interpreted as a private key. From this private key the corresponding public Bitcoin address is derived. The time of the first transaction to this public Bitcoin address is the tamper-proof timestamp. This works, since any change of the Merkle tree root would result in a different Bitcoin address.

A detailed guide can be found here: https://github.com/OriginStampTimestamping/originstamp-verification

A helpful tool to convert a private key into a public Bitcoin address is btcaddress-alpha: https://casascius.wordpress.com/2013/01/26/bitcoin-address-utility/