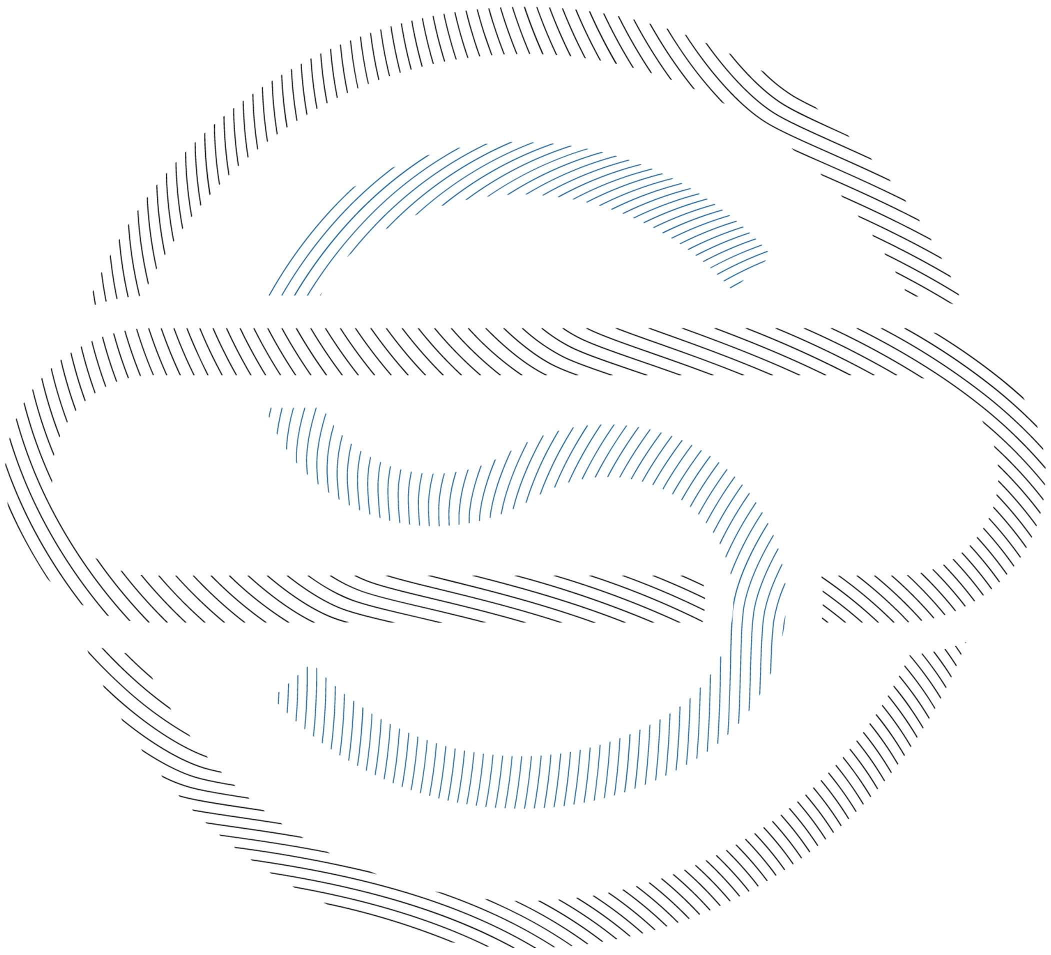
Timestamp Certificate

This timestamp was created with *Ethereum*



originstamp

Timestamp Sep-27-2024 18:00:35 UTC

Comment: 1018.eml

Hash:

3fb46291ff1819b41dc48f5d3de6f02eba2a899de9cb2714fe3ba0f363fadfb6

Transaction:

0xf4ab66ca32e9554a8e32e3b408e4dd4ee29a73f304e918b57c9bb5ba75bb5c77

Root Hash:

0fc0f1ac11cd9f11cdc9553b470c9141d7c90043e5022ac0376ccfd3be133459

Click here to verify your timestamp.

originstamp

Timestamp Certificate

Proof

The proof is necessary for the reproducibility of your document.

```
<?xml version="1.0" encoding="UTF-8"?>
<node type="key" value="0fc0f1ac11cd9f11cdc9553b470c9141d7c90043e5022ac0376ccfd3be133459">
 <left type="mesh" value="15b808d4f86da0c2fbd34ae8b769b4eb8e72a18daaebc4c2ef2a039e08ffc2aa">
  <left type="mesh" value="d8d37e1354b196334116f23c5201b1c9fa482e76f39d11ae65ebdc00d34ae2dc">
   <left type="mesh" value="6b3dfb47efe4cfcbcda56e74e6b1192e62d2d154ed4a147b183bf507c75bde7d"/>
   <right type="mesh" value="fa57d53dca31065312d3177447a9f4f96144506b42ddaa6dc17b78c8503f5b6b">
    <left type="mesh" value="729edab7ab86f50ab7e13007cd62215c3be46b1982e83f3e10809198050a8590">
     <left type="mesh" value="6a8debd8c008bbdeead09725d2e588c9cebcf78e4b3f13e64a9655d3000448d3">
      <left type="mesh" value="8a0974fa0b112075d73c8e30bfa0e351ed0cab56e1c5e0cdbc06f5c953f1be41">
       <left type="mesh" value="ab505cee132e6a00ff7e9e8ef026d4bf6d2323701c20d34d75638bd320d42205"/>
       <right type="mesh" value="a4cdfa6f81aa423d0592c056cee1fd0719c7b76e973061b1e5391e8a251640b8">
        <left type="mesh" value="dc68ae5f8822a4405e793653943ef05d60d238a011a542766ced690de0bf3ff2"/>
        <right type="mesh" value="5cc8ef53bcba443873a3cf5045ad902da3e18e2bb0d3bba6546e2862acd5b5a2">
         <left type="mesh" value="41eafa42eebd65c41b344b29b466c74d0767a74e7d4f8eeabd928eba267dea96"/>
         <right type="mesh" value="2bbf7e1e455386a9c0a17a6cad1d9379e17b6bef8be6b7762ce9fe277eb17679">
          <left type="mesh" value="972c72aa880e6bc4435ce5fcf01135c3ec14393775b287082115884e001d0e88"/>
          <right type="mesh" value="27474edc86a4aa558dd67949118d1b0ed0f4d7a9fcfb4abd29e3548e38c80067">
           <left type="mesh" value="89706e1a028e5c2dca0f5c3c999e49474594eac4d2d294bb35993fb93a91eaeb">
            <left type="mesh" value="f9b6e803e48bd01df4462b41028fa5418d2237d67a1d03e48c8ad69263c703d6">
             <left type="mesh" value="d521871dabb6dacd98a82b280e0823ed00d2fc56de3c873f1cbe95cbd2d5bd65"/>
             <right type="mesh" value="b7c7e73bf3d34c49c169ca601bded0d6cfe62e52a3479c1921930538bdca0555">
              <left type="mesh" value="f68f433ad024077d8dcf068220caae4b5ca3a51c34ff139fa0ca8bcc051aee34">
               <left type="mesh" value="f71a6eb95b5e1d96dc116922d24df36e4f2e68bd907cbf1cb2b62dc26fef3e32">
                <left type="mesh" value="76fd0b1d5336b2ea47246b694b01dcc66a9b1d2fa1256d4217c4312079f2944b"/>
                <right type="mesh" value="2d5befb1c0dde07f5f99fe06e9bd8fa8d1848487d3072f66457166d6a34ec21f">
                 <left type="mesh" value="3fb3fbe6e23f8e7c57261a4e83c61f776e2f1249806beec2c48f210e092859d6"/>
                 <right type="hash" value="3fb46291ff1819b41dc48f5d3de6f02eba2a899de9cb2714fe3ba0f363fadfb6"/>
                </right>
               </left>
               <right type="mesh" value="b31ecd14d4341120f01825d9d561db371dbe8de575b0fb851a5a747411a10437"/>
              </left>
              <right type="mesh" value="f77647b1983b9ebc877d2482f963d2fbf4c6d8e75bbcaabd7930757e3db8cb96"/>
             </right>
            </left>
            <right type="mesh" value="8fa6c1905a952eb29450601182c435fd37838c40d3a0c6e3a0b3362182f94b30"/>
           </left>
           <right type="mesh" value="38a27a71d9108e447ed3fbe56a8b6a9d15e11b9e1c0879dd866fb216467b6d8a"/>
          </right>
         </right>
        </right>
       </right>
      </left>
      <right type="mesh" value="0ae6417d04f030a7db497282187819d79ee9b7b62e86647a2adac75e74201538"/>
     </left>
     <right type="mesh" value="968b5bc39f1d4c38af98b0823928b23be5824f6f2514d35c4348395b1390c44b"/>
    </left>
    <right type="mesh" value="98a1900ccca4e154330c529c3a55f402078fcf2f93c09d0e699eb51bff4d4881"/>
   </right>
  </left>
  <right type="mesh" value="a4e75b13ce9a46580df7323d012d77b44e8b5b0f504c2c0b3f128ba91ae1377d"/>
 </left>
 <right type="mesh" value="119e900ef64db7436842c6a88027e6a8767c83a935887c35e6455c5060ccf650"/>
</node>
```

originstamp

Timestamp Certificate

Verification

OriginStamp is a timestamp service that uses various blockchains like the Bitcoin Blockchain to create tamper-proof timestamps for your data. Instead of backing up your data, OriginStamp embeds a cryptographic fingerprint in the blockchain. It is de facto impossible to deduce the content of your data from your fingerprint. Therefore, the data remains in your company and is not publicly accessible. All you need to do is send this fingerprint to OriginStamp via the interface. The integration of the RESTful API is very simple and convenient and allows all data to be easily tagged with a tamper-proof timestamp. This document shows the procedure and gives instructions for verifying a timestamp created with OriginStamp.

1. Determine the SHA-256 of your original file

There are numerous programs and libraries to calculate the SHA-256 of a file, such as MD5FILE. Simply drag and drop or select your file, to retrieve the SHA-256 of your file.

2. Validate your proof

First, it must be verified that the hash of the original is part of the evidence. In order to check this, the proof can be opened with a conventional editor and its content can be searched for the hash. If the hash cannot be found, either the file was manipulated or the wrong evidence was selected.

3. Determine the root hash

The Merkle tree is a tree structure, that allows to organize the seed more efficient than a plain-text seed file. The tree is built from the bottom to the top and follows a defined schema. The value of a node is determined by the aggregated hash of its children.

Left child =
a8eb9f308b08397df77443697de4959c156fd4c68c489995163285db
d3eedaef

Right child =
ab95adaee8eb02219d556082a7f4fb70d19b8000097848112eb85b1
d2fca8f67

Node = SHA-
256(a8eb9f308b08397df77443697de4959c156fd4c68c4899951632
85dbd3eedaefab95adaee8eb02219d556082a7f4fb70d19b8000097
848112eb85b1d2fca8f67) =
47e47c96302eeba62ed443dd0c89b3411bbddd2c1ff6bdfb1f833fa1
1e060b85

This step is performed for all levels of the tree until the hash of the root has been calculated. If the hash of the root is identical as proof, the calculation was successful and the root hash is verified. The top hash corresponds to the root hash we embedded in the blockchain through a transaction. For a more detailed explanation of the Merkle tree, we want to refer to Wikipedia.

4. Determine the Ethereum Transaction

Having determined the root hash in the previous step, we store this hash in a Smart Contract in the Ethereum blockchain.

5. Check the transactions

The respective log events must be searched for each transaction in this contract. These events are divided into topics. The root hash should be contained in a topic. Caution: Some services display the hashes with a *0x* prefix. As soon as the transaction has been found, the block time is the actual tamper-proof timestamp. To simplify the search, we added the transaction to the certificate.