OpenTimestamps

Bitcoin-based timestamping proof standard

Valerio Vaccaro

Satoshi Spritz Milano

March 18, 2025

Valerio Vaccaro 👀

- 💻 Bitcoin Developer and Hardware Expert
- 🔥 Contributor to Bitcoin open source projects
- ⚠️ DIY hardware enthusiast
- Bitcoin and Liquid Engineer at Blockstream

Social

- 🧑‍💼 **LinkedIn** linkedin.com/in/valeriovaccaro
- 🐙 **Github** github.com/valerio-vaccaro
- **Telegram** t.me/valeriovaccaro

License

This presentation is distributed under the Creative Commons CC BY-SA 4.0 license.

Images used in this presentation are property of their respective authors and are included for educational and illustrative purposes only.

May this presentation inspire you to become more self-sovereign!

Meme


IT WAS A VERY NICE TIMESTAMP
IT WAS THE PERFECT TIMESTAMP
makeameme.org

Outline

- What is OpenTimestamps
- Classical timestamping and certified date
- Create, upgrade and verify proofs
- Usage examples
- Flow and diagrams
- Calendars and resources

What is OpenTimestamps?

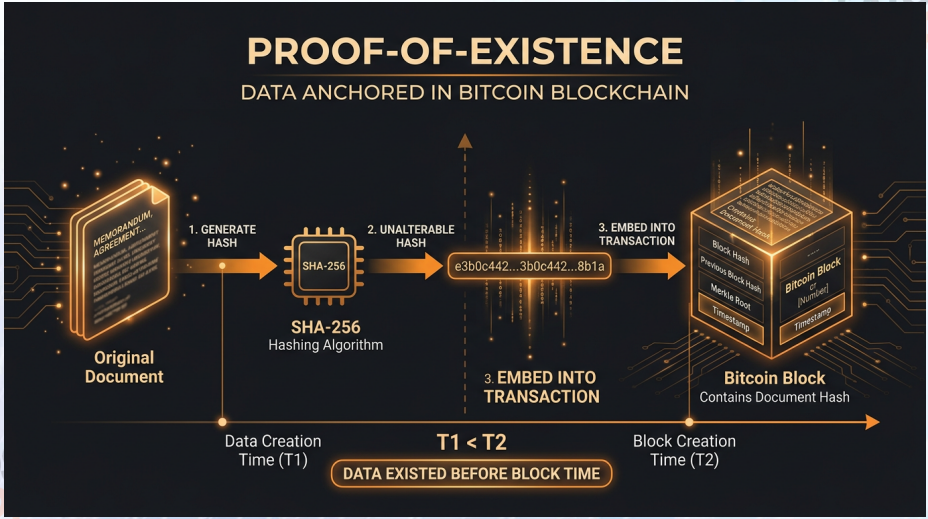
Blockchain timestamping standard

- **Proof-of-existence**: proves data existed before a certain time
- opentimestamps.org
- Trust-minimized: uses Bitcoin, no central authority
- Scalable: thousands of timestamps in one transaction

Benefits (Peter Todd, 2016)

- **Trust**: decentralized, auditable blockchain
- **Cost**: Merkle aggregation, negligible cost
- **Convenience**: timestamp in ~1 second (calendar server)

Proof-of-existence concept



What a timestamp can and cannot prove

Can prove

- ✅ **Record integrity**: data existed before an event
- ✅ **Software/PGP signing**: signature date verifiable
- ✅ **Evidence authenticity**: limits who could have altered

Cannot prove

- 🔴 **Ownership**: does not directly attribute to a person, though one can keep the receipt secret to prove knowledge of the file at a certain date
- 🔴 **Uniqueness**: does not solve doublespend
- 🔴 **Content**: proves existence, not meaning

Classical timestamping

Certified date examples

- ✉️ **Post office**: registered mail with return receipt
- 🚗 **Vehicle registry**: registration of vehicles
- 📜 **Notary**: notarial deeds with certified date
- 📅 **Certified date**: legal value (registry, stamp, certified email)

Context

- 🔑 All require a **trusted third party** (post office, registry, notary)
- ⚙️ Paper or bureaucratic processes, costs and delays

Difficulties: digital files and signatures

The problem

- ⚠ **Digital file**: no physical medium to mail or stamp
- ⚠ **Digital/PGP signature**: the signature proves authenticity, but **not when** it was applied
- 🔴 Without certified date: impossible to prove a document existed at a precise time

Why it matters

- 🔗 Contracts, patents, legal evidence, code commits
- 🛡 The date is often crucial for validity and priority

Certified date and key theft

Critical scenario

- PGP/software key **stolen** → revocation and new key issuance
- **Problem**: how to distinguish valid signatures (pre-theft) from forged ones (post-theft)?

Solution: timestamp on signature

- **Certified date on signature** → proves the signature existed **before** revocation
- Signature with pre-revocation timestamp = **valid**; without = suspect
- Essential to recognize legitimate signatures in case of key theft

Create a timestamp (stamp)

Command

```
$ pip3 install opentimestamps-client
$ ots stamp document.pdf
```

Output

- 🔑 Creates `document.pdf.ots` (proof file)
- 🔗 Sends hash to calendar servers (Alice, Bob, ...)
- ⚙ Receives incomplete proof in ~1 second

With Bitcoin confirmation wait

```
$ ots stamp --wait document.pdf
```

Upgrade a timestamp (upgrade)

Incomplete vs complete proof

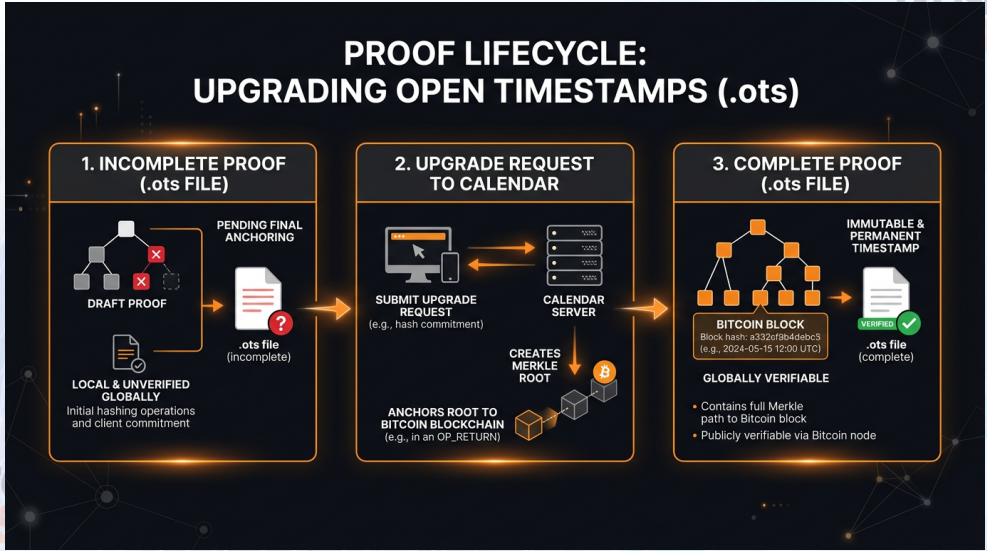
- ⚠️ **Incomplete**: depends on calendar server (PendingAttestation)
- ✅ **Complete**: Bitcoin proof in blockchain, local verification

Upgrade

```
$ ots upgrade document.pdf.ots
```

- 🔗 Downloads Bitcoin attestation from calendar
- 🔑 Saves complete proof to .ots file
- 🛡️ Verification possible without calendar

Proof lifecycle: Incomplete to Complete



Verify a timestamp (verify)

Command

```
$ ots verify document.pdf.ots
```

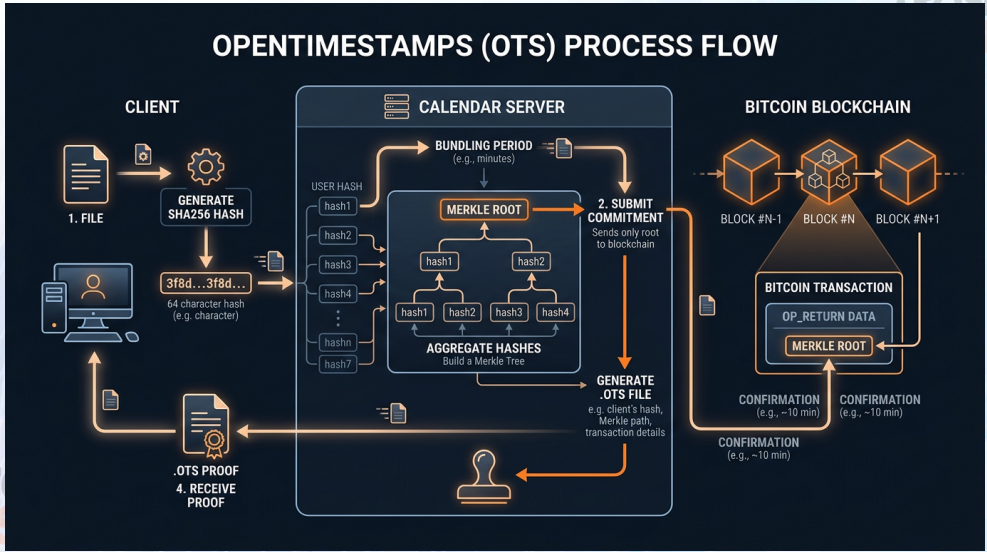
Process

- ⚙ Computes hash of original file (if provided)
- 🔑 Executes proof commitment operations
- 🔗 If incomplete: asks calendar for attestations
- ✅ Verifies against Bitcoin block header

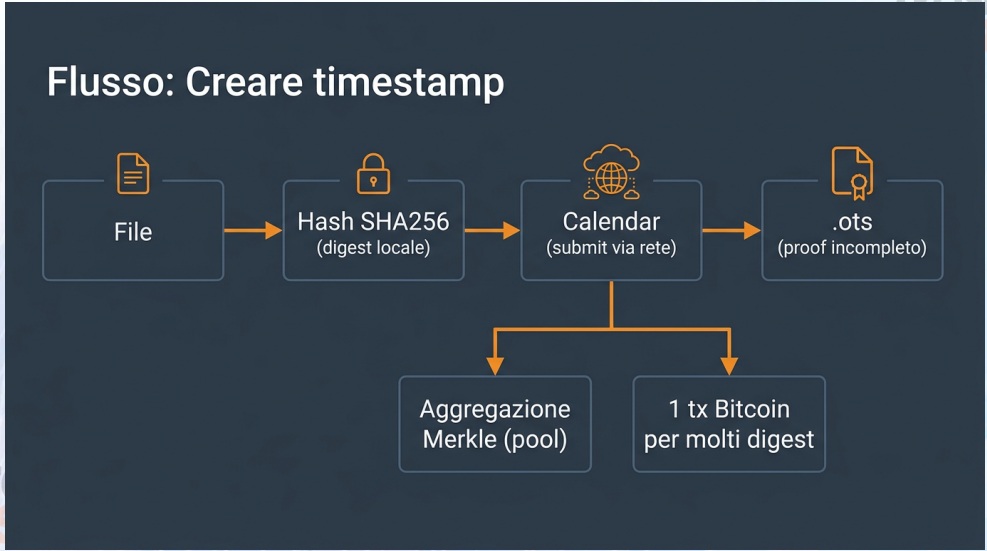
Output

```
Success! Bitcoin attests data existed as of Thu May 28 15:41:18 2015 UTC
```

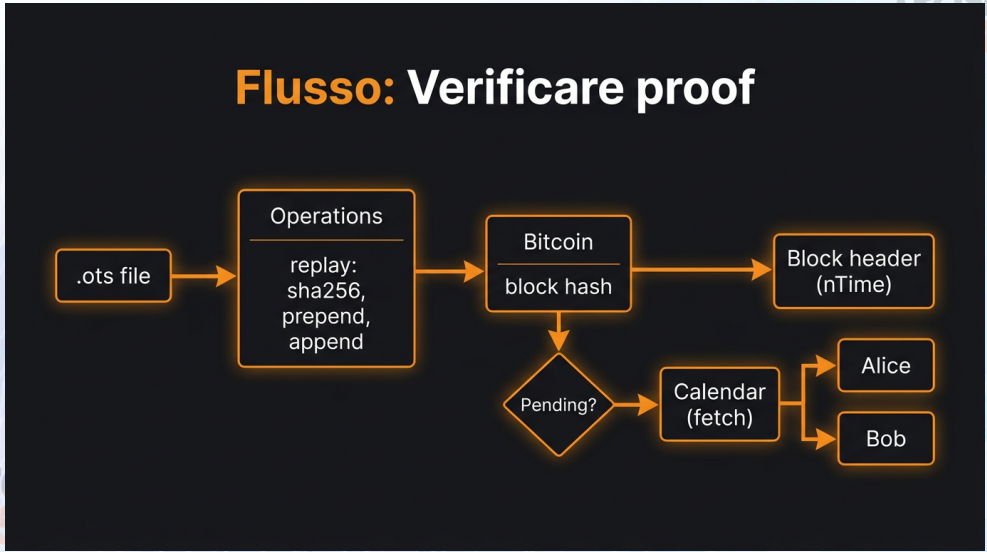
Diagram: OpenTimestamps flow



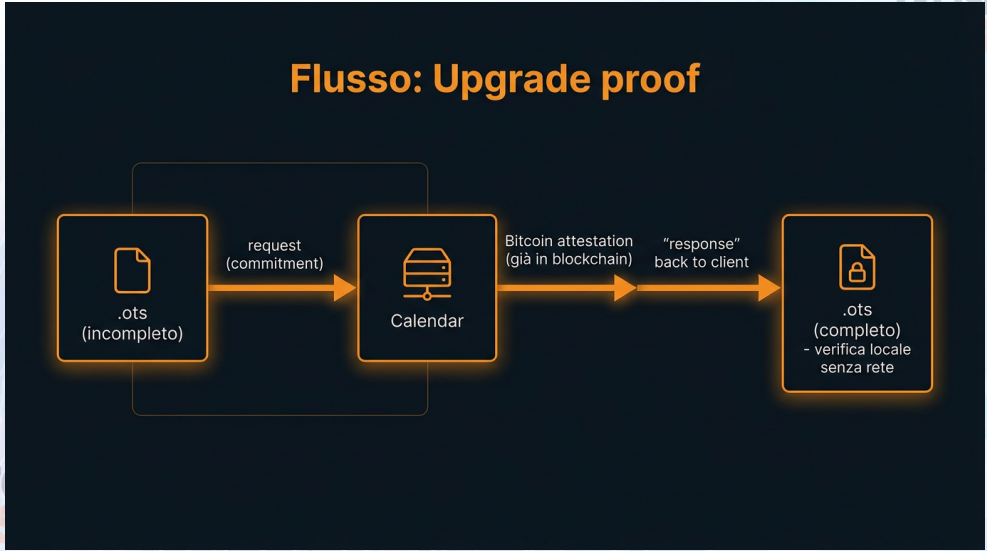
Flow: Create timestamp



Flow: Verify proof



Flow: Upgrade proof



Commitment operations

Operation tree

- **sha256**, **ripemd160**: hash
- **prepend**, **append**: fixed data
- **verify**: attestation (Bitcoin, Pending, . . .)

Example

```
message -> sha256 -> prepend X -> append Y -> sha256 -> verify BitcoinBlock(35
```

Public calendars

Free servers

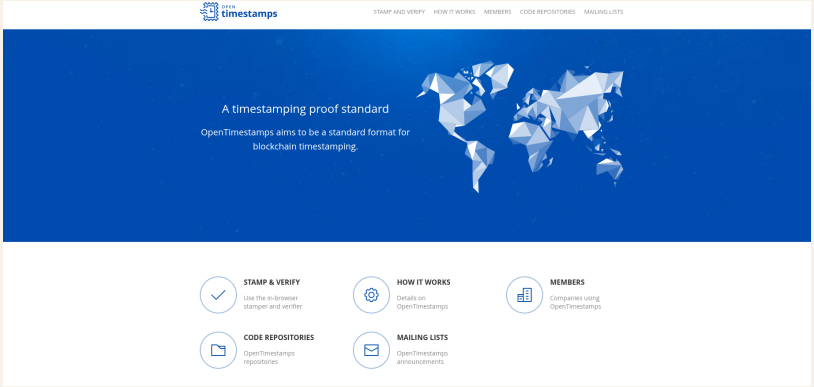
- **Alice**: alice.btc.calendar.opentimestamps.org
- **Bob**: bob.btc.calendar.opentimestamps.org
- **Finney**: finney.calendar.eternitywall.com

Redundancy

- By default: 2+ calendars per timestamp
- Calendars optional: `ots stamp --wait` (no calendar)

Usage examples

Web interface (opentimestamps.org)



Usage examples

Documents

```
$ ots stamp contract.pdf
$ ots verify contract.pdf.ots
```

Git commit

```
$ git commit -m "Release v1.0"
$ ots stamp .git/COMMIT_EDITMSG
```

PGP / software signing

- Timestamp on signature: key revocation date vs signature date

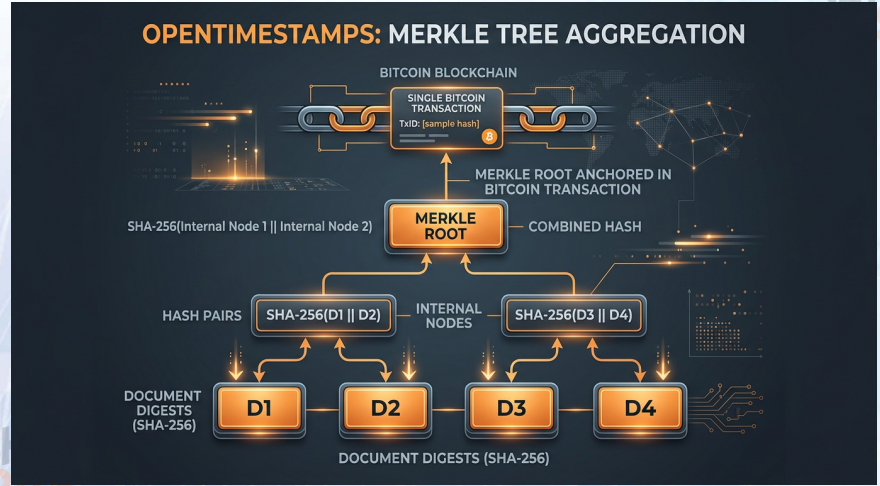
Examples: code

```
$ echo "Hello World!" > hello.txt
$ ots stamp hello.txt
$ ots info hello.txt.ots
$ ots verify hello.txt.ots
```

Info

```
$ ots info file.ots
File sha256 hash: 03ba204e50d126e4674c005e04d82e84c21366780af1f43bd54a37816b6a
Timestamp:
  ripemd160
  prepend ...
  verify BitcoinBlockHeaderAttestation(358391)
```

Merkle tree: aggregation



OpenTimestamps integration in GitHub

Why timestamp commits

- **IP protection**: Git commits are not immutable — dates can be modified
- **Temporal proof**: demonstrates work existed at a precise time (useful in legal contexts)
- **Permanent record**: blockchain timestamp = indelible proof

GitHub Action

- **open-timestamps-github-action** (yzernik/open-timestamps-github-action)
- Timestamps Git tags on the Bitcoin blockchain
- Available on GitHub Marketplace, integrates into workflows

GitHub integration: workflow

Usage example

```yaml
# .github/workflows/timestamp.yml
on:
  push:
    tags: ['*']
jobs:
  timestamp:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: yzernik/open-timestamps-github-action@v1
      - run: git push origin --tags
```

GitHub integration: workflow

Alternative: manual

```
$ ots stamp .git/COMMIT_EDITMSG
$ ots stamp .git/refs/heads/main
```

GitHub integration: resources

Documentation and links

- git-integration.md — official guide
- open-timestamps-github-action — automatic workflow
- Others: **tstamp** (Go), Python/JS/Rust clients

Use cases

- Releases and version tags
- Priority proofs for patents
- Audit trail for sensitive commits

References

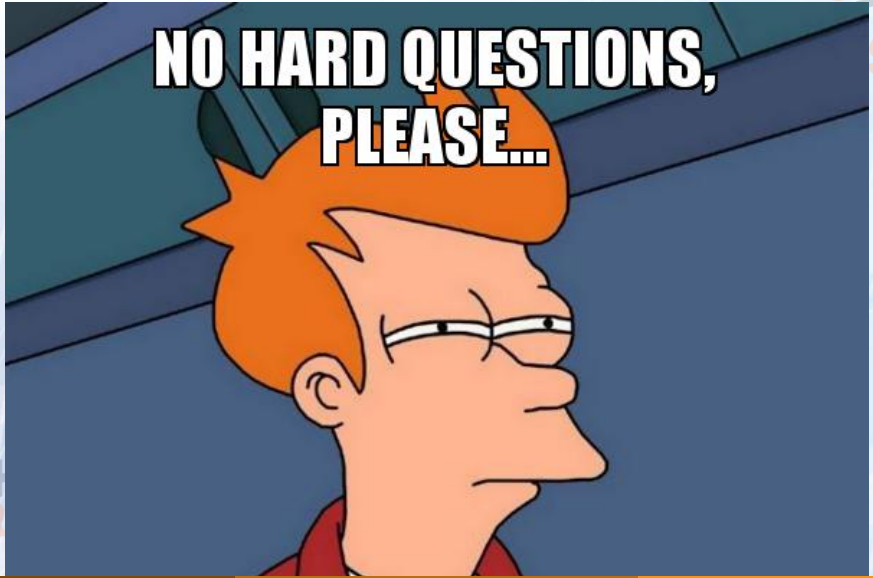
Blockstream / OpenTimestamps

- **opentimestamps.org**: opentimestamps.org
- **Client**: github.com/opentimestamps/opentimestamps-client
- **Announcement**: petertodd.org/2016/opentimestamps-announcement

Implementations

- python-opentimestamps
- javascript-opentimestamps
- rust-opentimestamps

? Questions



Satoshi Spritz

- 💻 Federation of local Bitcoiner groups
- 🎓 Free, privacy-oriented events
- 🤖 BITCOIN ONLY
- 🔧 Weekly Satoshi Spritz Connect online

Links

- satoshispritz.it
- t.me/SatoshiSpritzConnect

Officine Bitcoin

- 🤝 Italian Bitcoin community, fully free
- 🤖 BITCOIN ONLY
- 🎓 Education and project development

Links

- officinebitcoin.it

Bitcoinissimo

- Bitcoin podcast and statistics
- Episodes in Italian, English, Hungarian, Chinese, Russian, Spanish, French
- Real-time network stats, market data, block explorer

Links

- bitcoinissimo.it