

Independent Verification Bundle: step-by-step guide

Reconstruct the certified content, validate the SHA-256 evidence, verify the RFC 3161 timestamp and distinguish historical proof from current status.

Before you begin

An Independent Verification Bundle enables third parties to inspect the cryptographic evidence associated with a Certidox-certified document or email, independently of the usual verification display. This guide covers **historical certified content and timestamp evidence**. Current certification status is a separate live check.

The sample linked below comes from a **fictional mock hearing**. It includes a corresponding certified order form, encrypted payload, reading key, manifest, timestamp response and executable verification scripts. It is intended for training, not evidence of a real-world transaction.

Keep the reading key confidential. The bundle contains the material required to decrypt the certified reference. Share it only with people authorized to examine the underlying document or email.

Files used in the example

File	Role in verification
<code>document.enc</code>	AES-256-GCM encrypted certified reference.
<code>reading_key.txt</code>	32-byte reading key provided in hexadecimal form.
<code>manifest.json</code>	Certification ID, payload SHA-256, recorded event time and certifier information.
<code>response.tsr</code>	DigiStamp RFC 3161 timestamp response linked to the exact manifest bytes.
<code>decrypt.py</code> / <code>verify_timestamp.sh</code>	Convenience scripts to reconstruct and verify the evidence.
<code>readme.txt</code>	Bundle-specific technical instructions and limitations.

STEP 01

Preserve the original archive

Save the downloaded ZIP unchanged. Compute its SHA-256 and record the result together with its source and time of collection. Work on a separate copy to preserve reproducibility.

```
# macOS
shasum -a 256 verification-bundle.zip

# Linux
sha256sum verification-bundle.zip

# PowerShell
Get-FileHash .\verification-bundle.zip -Algorithm SHA256
```

This **ZIP hash** identifies the exact downloaded archive. It is not the certified document hash and should not be confused with the value in `manifest.json`.

STEP 02

Reconstruct the certified reference

Unpack your working copy into a controlled environment. Inspect the included scripts before running them. The example uses Python 3 and the `cryptography` package. The AES-256-GCM nonce is stored at the beginning of `document.enc` ; the reading key is taken from `reading_key.txt` .

```
python3 -m pip install cryptography
python3 decrypt.py
```

For a PDF certificate, the script writes `document.pdf` ; a structured email record may produce `document.json` . Confirm that the reconstructed content is readable and corresponds to the expected document or structured email content.

Email limitation: a structured email record is not a native `.eml` and does not independently establish that the email was delivered, opened or read. The archived record may include attachment metadata rather than the attachment binary.

STEP 03

Check SHA-256 content integrity

Read `payload_sha256` inside the exact `manifest.json` supplied in the bundle. Calculate SHA-256 over the reconstructed payload bytes, not over `document.enc` , the ZIP, or a screenshot.

```
# macOS
shasum -a 256 document.pdf

# Linux
sha256sum document.pdf

# Windows PowerShell
Get-FileHash .\document.pdf -Algorithm SHA256
```

The result must equal the manifest's `payload_sha256` exactly. A mismatch means you have **not** established the claimed content integrity. The sample's supplied verification script checks this relationship automatically, but independent recomputation remains advisable.

Important: for printed or converted copies, the image or scan file hash will normally differ. Compare their visible content with the reconstructed certified reference in Step 5.

STEP 04

Verify RFC 3161 timestamp evidence

The RFC 3161 timestamp is attached to the **exact bytes of** `manifest.json`, which contain the content hash and recorded certification information. Obtain the relevant DigiStamp trust certificates independently, validate their source and expected fingerprints, and confirm the timestamp response signature and message imprint using an RFC 3161-capable verifier.

```
openssl ts -verify \  
-in response.tsr \  
-data manifest.json \  
-CAfile trusted-tsa-certificates.pem
```

The `openssl ts -verify` command must succeed with an independently validated chain of trust. Do not treat a copied certificate or a green status display alone as proof.

For the example bundle, `bash verify_timestamp.sh` automates a check of the reconstructed payload and timestamp. Read the script before running it; it may download trust material. Independently confirm the DigiStamp certificate provenance.

Two different times

`certification_event_time_utc` is Certidox's recorded certification event time. The RFC 3161 `genTime` is the time asserted by the independent timestamp authority. They answer different questions and should not be presented as identical.

STEP 05

Compare the document presented

For a **byte-identical digital copy**, compute its SHA-256 and compare it with the certified reference hash. For a **printed original, photocopy, scan, screenshot or re-exported PDF**, compare the visible text, tables, checkboxes, annexes and all other material content against the reconstructed certified reference.

A QR code alone is not proof of the surrounding page content. Its function is to lead to the applicable certification record and reference so the comparison can be made.

STEP 06

Check current certification status

The bundle establishes **historical cryptographic evidence**. It cannot, by itself, prove that certification has not since been suspended, revoked or superseded. Use the free Certidox app to scan the associated QR code and consult the **live status before relying on the document**.

This distinction is particularly important for authorizations, orders, powers of attorney and other documents that may circulate in digital and printed originals and copies.

Record the outcome

Save a reproducible verification report containing: source of the ZIP, preserved archive hash, reconstructed payload hash, manifest values, RFC 3161 verification result and trust material used, evidence of comparison with the presented document, and the date of the separate live-status check.

Limits of the conclusion: cryptographic evidence protects the recorded statements and content against undetected modification. It does not, on its own, determine whether the underlying facts were true, whether the recorded certifier was legally authorized, or whether an email was delivered or read.