BERNSTEIN

TECHNICAL BRIEF

INNOVATION EVIDENCE PLATFORM

The Bitcoin certification protocol

How Bernstein turns your files into court-grade proof of existence, integrity and ownership - anchored to the Bitcoin blockchain, and verifiable by anyone without trusting Bernstein.

Every Bernstein certificate is a Bitcoin transaction whose address is cryptographically derived from your content and your keys. This brief explains the protocol end to end - the three-key model, the deterministic fingerprint, the registration transaction, the certificate trail, and how an independent party re-derives and checks the whole thing on-chain.

UPDATED	AUDIENCE	PROTOCOL
June 2026	Technical & procurement	3-of-3 multisig · P2WSH

1 · THE BASICS

1 What a Bitcoin certificate proves

A Bernstein certificate is cryptographic proof that a specific set of files existed, intact, under your control, at a specific moment - recorded in a Bitcoin transaction that no one can alter or backdate.

THREE QUESTIONS, ONE TRANSACTION

Each certificate answers **WHEN** (the block timestamp), **WHAT** (a one-way fingerprint of your exact content), and **WHO** (a key only you hold). The proof lives in the transaction's address, which is mathematically derived from all three - change any one and the address no longer matches the chain.

Bernstein applies client-side, zero-knowledge encryption first, then certifies a hash of your data. Your content never leaves your control. What is written to Bitcoin - and what a court or expert can later re-check - is a fingerprint, never the files themselves.

Why anchor to Bitcoin

Public

The blockchain is open to everyone. Anyone can look up the transaction and verify the proof - no account, no permission, no cooperation from Bernstein required.

Timestamped

Every transaction is sealed inside a block carrying an immutable timestamp, ordered by the network and confirmed by proof-of-work.

Immutable

Rewriting a confirmed block means redoing its proof-of-work - and every block after it. It is economically infeasible, which is what makes the record permanent.

What is - and is not - written to the chain

	On the blockchain	Never on the blockchain
Your data	A 256-bit cryptographic fingerprint	✗ The files or any clear-text
Your keys	Three *public* keys, inside the address	✗ Any private key
Identity	A control proof you can later sign	✗ Your name or personal data

The fingerprint is one-way: it cannot be reversed to recover your files, yet it changes completely if a single byte does. That is the whole basis of the proof.

2 · THE KEYS

2 Three keys, one address

Every certificate address is a **3-of-3 multisignature** built from three independent public keys. Combining them binds the certificate to your content *and* your ownership at the same time - no single party, Bernstein included, can produce or forge the address alone.

Owner key

From your HD wallet, generated on your device at signup. Proves the certificate is yours - the private half never leaves you and is never stored by Bernstein.

Data key

Derived deterministically from your project's fingerprint: the hash itself becomes a Bitcoin private key, and its public key goes into the address. This is what binds the certificate to your exact content.

Bernstein key

Issued by the platform per certificate from a hierarchical-deterministic path. Provides protocol compliance and ties the certificate into its trail.

WHY THE DATA KEY IS THE CLEVER PART

The fingerprint of your files is converted to WIF (Wallet Import Format) and treated as a Bitcoin private key; its public key becomes the **data key**. So the content is not just *referenced* by the certificate - it is *baked into the cryptography* of the address. Change one byte of one file and the fingerprint changes, the data key changes, the address changes, and it no longer matches the transaction on the chain. The match itself is the proof.

Combining the keys into the address

The three public keys are assembled into a 3-of-3 multisig script. Under protocol 2 (current) the script is hashed into a SegWit **P2WSH** address; under the legacy protocol 1 it produced a **P2SH** address. Either way, the address is a pure function of the three keys - reproducible by anyone holding the same inputs, and verifiable without Bernstein.

3 of 3	P2WSH	0
KEYS REQUIRED	SEGWIT MULTISIG ADDRESS	PRIVATE KEYS STORED

3 · THE FINGERPRINT

3 How the project fingerprint is computed

The whole proof rests on one deterministic number: the **project fingerprint**. It is computed the same way every time, by anyone, from the same files - no secret, no Bernstein-only step. That reproducibility is what lets a third party re-derive it independently.

1 · Hash each file

Every file is hashed individually with **SHA-256**. A cover hash is computed from the project title and description the same way.

2 · Sort & concatenate

The hashes are sorted alphabetically and concatenated into one string - so file order, naming or upload sequence can never change the result.

3 · Hash again

A final SHA-256 over the concatenation yields the 256-bit **project fingerprint** - the single value that becomes the data key.

DETERMINISTIC BY DESIGN

Two parties hashing the same files - in any order, on any machine - arrive at the exact same fingerprint. Add, remove or alter a single file and the fingerprint changes completely (the avalanche property of SHA-256). There is no way to find different files that produce the same fingerprint, which is precisely why it stands as proof of *what* was certified.

Because the fingerprint is the input to the data key, and the data key is part of the address, the address on the chain is ultimately a fingerprint of your content - wrapped in cryptography, anchored in a block, and impossible to forge after the fact.

4 · THE PROTOCOL

4 Registration, end to end

Certifying a project runs a fixed five-step pipeline - from raw files to a confirmed Bitcoin transaction. Each step is deterministic and independently reproducible.

1 · Fingerprint

Hash all files and the cover, sort, concatenate, hash again - the 256-bit project fingerprint (§3).

2 · HEX → WIF

Convert the fingerprint to Wallet Import Format, turning it into a valid Bitcoin private key.

3 · Derive data key

Compute the compressed public key from that private key - the data key.

4 · Build address

Combine the owner, data and Bernstein public keys into a 3-of-3 multisig P2WSH address.

5 · Anchor on-chain

Send a small, fixed amount of bitcoin to the address and broadcast the transaction. Once confirmed, the certificate exists.

Result

A confirmed Bitcoin transaction whose address encodes your content and your ownership, stamped with the block's time.

Anatomy of the registration transaction

WHAT THE TRANSACTION CARRIES

The transaction spends a Bernstein-funded input to pay the network fee, and writes three outputs: (**1**) the certificate output - the multisig address, carrying a small fixed 'persisting value' of bitcoin that flows forward into the next certificate in the trail; (**2**) an **OP_RETURN** marker embedding the protocol version and registry reference (e.g. `BERNSTEIN 2.1 REG <id>`), a human-findable tag on the chain; and (**3**) change. The fee is bounded by a hard ceiling, and only a hash is ever exposed - never your data.

Confirmation is at the block level, so the certified time has roughly ten-minute precision - more than sufficient for evidentiary purposes, and corroborated to the second by the qualified timestamps Bernstein anchors alongside Bitcoin (see the companion trusted-timestamping brief).

5 · THE TRAIL

5 An immutable trail of versions

Innovation is not a single moment - it evolves. When a project's contents change, Bernstein issues a **new** certificate and links it to the previous one on-chain, building a tamper-evident history of how the work developed over time.

Initial certificate

The first registration of a project. A funded input pays the fee; the multisig output opens the trail, carrying the persisting value forward.

Continuing certificate

Each later revision spends the *previous* certificate's output as an input, moving the same persisting value into a fresh address derived from the new content - cryptographically chaining old to new.

WHY THE CHAIN CANNOT BE FAKED OR REORDERED

Because every certificate after the first *spends* its predecessor, the sequence is fixed by Bitcoin itself: a continuing certificate cannot exist until the previous one is confirmed, and the links cannot be reordered, inserted or removed without rewriting Bitcoin's history. The result is an **immutable, ordered trail** - each link a dated, independently verifiable snapshot of the project at that revision. Signing each continuing transaction takes all three keys again, so no party can extend a trail alone.

The trail also closes the value loop cleanly: the persisting amount carried from one certificate to the next is held constant by construction, so nothing leaks and every transaction in the trail balances exactly.

6 · VERIFICATION

6 How anyone verifies - without trusting us

This is the point of the whole design: a certificate is checkable by **anyone**, with open tools, and no cooperation from Bernstein. Zero-trust - everything can be recomputed from the original files and the certificate's public data, then matched against the public chain.

1 · Compute the fingerprint

Hash each original file with SHA-256, sort, concatenate, hash again - reproduce the project fingerprint yourself.

2 · Hash → private key

Convert the fingerprint to WIF to recreate the data private key from the content alone.

3 · Derive the data key

Compute the compressed public key with any Bitcoin key tool - no Bernstein input.

4 · Rebuild the address

Combine the owner, data and Bernstein public keys (from the certificate PDF) into the 3-of-3 multisig address.

5 · Match on-chain

Look up the transaction ID on any blockchain explorer and compare your address with its output.

Match = proof

If the addresses match, the files existed in exactly that state at the transaction's block time. No match means something changed.

PROVE EXISTENCE, OWNERSHIP AND A PORTABLE SIGNATURE

The same tooling does three jobs: **prove existence** by matching the fingerprint registered on-chain; **prove ownership** by signing with your private key to demonstrate control of the certificate; and **generate a signature** a third party - a court, an investor, a counterparty - can independently validate. The official validator runs entirely in your browser; the manual path below needs only standard, off-the-shelf cryptographic tools.

	Where it comes from
Original project files	Your own copies - the exact uploads
Transaction ID	The certificate PDF
Owner & Bernstein public keys	The certificate PDF
SHA-256 + key tools + explorer	Open, standard, off-the-shelf

Everything required is either in your possession or printed on the certificate. Nothing depends on Bernstein staying online.

7 · STEP BY STEP

7 Verify it yourself, command by command

The conceptual path above, made concrete. With standard, open tools - a SHA-256 utility, Python, Ruby and any blockchain explorer - you can reproduce a certificate's address from your files and the certificate PDF, then match it on-chain. No Bernstein account, no API, no trust required.

Step 1 · Compute the project fingerprint

Hash every original file with SHA-256, then sort the hashes alphabetically, concatenate them into one string, and hash that once more. The result is the 256-bit project fingerprint (§3).

```
# hash every original file (macOS / Linux)
sha256sum -b cover.txt
sha256sum -b file1.pdf
sha256sum -b file2.png

# then sort the hashes alphabetically, concatenate
# them into one string, and SHA-256 it once more
# → that final hash is the project fingerprint
```

Step 2 · Convert the fingerprint to a Bitcoin private key (WIF)

Convert the HEX fingerprint to Wallet Import Format. This turns your content's fingerprint into a valid Bitcoin private key - the seed of the data key.

```
# hex2wif.py — fingerprint (HEX) → private key (WIF)
import binascii, base58, hashlib, sys

private_key  = sys.argv[1]
extended_key = "80" + private_key + "01"
first  = hashlib.sha256(binascii.unhexlify(extended_key)).hexdigest()
second = hashlib.sha256(binascii.unhexlify(first)).hexdigest()
wif    = base58.b58encode(binascii.unhexlify(extended_key + second[:8]))
print(wif.decode())
```

```
$ pip3 install base58
$ python3 hex2wif.py <your-fingerprint-hex>
```

Step 3 · Derive the data public key

Feed the WIF private key into any Bitcoin key-compression tool (e.g. **iancoleman.io/bitcoin-key-compression**) to obtain the compressed **data public key** - computed from your content alone, with no Bernstein input.

Step 4 · Rebuild the 3-of-3 multisig address

Combine the owner and Bernstein public keys (printed on the certificate PDF) with the data key you just derived, into the 3-of-3 multisig address.

```ruby
# multisig_address.rb — gem: bitcoin-ruby ~> 0.0.18
require "bitcoin"

owner_key     = "0365...2951"   # certificate PDF
data_key      = "02ad...9383"   # your steps 1–3
bernstein_key = "026a...ccf8"   # certificate PDF

p2sh, witness = Bitcoin::Script.to_p2sh_multisig_script(
  3, owner_key, data_key, bernstein_key)

# Protocol 2 — current (SegWit P2WSH)
program = Digest::SHA256.digest(witness)
puts Bitcoin.encode_segwit_address(0, program.bth)

# Protocol 1 — legacy (P2SH)
puts Bitcoin::Script.new(p2sh).get_p2sh_address
```

Step 5 · Match it on the blockchain

Open any explorer (e.g. **blockchair.com**), search the **Transaction ID** from the certificate, and find the multisig address in the transaction's outputs. Compare it to the address you computed.

MATCH = PROOF

If the address you derived equals the one in the transaction's output, the files existed in **exactly** that state at the block's timestamp - and the certificate is yours to control. No match means the files, keys or protocol version differ. The whole check runs without Bernstein; the official validator at **app.bernstein.io/validator** simply automates these same five steps in your browser.

Tools reference

	Where	What it does
Bernstein Validator	app.bernstein.io/validator	Official verifier - runs entirely in your browser, automating steps 1-5
File hash	fileformat.info/tool/hash.htm	SHA-256 a file without the command line (step 1)
Key compression	iancoleman.io/bitcoin-key-compression	Derive the public key from a WIF private key (step 3)
Blockchain explorer	blockchair.com	Look up the transaction and inspect its outputs (step 5)
Ruby runner	replit.com (Ruby)	Run the multisig script with no local setup (step 4)

Dependencies: Python `base58`; Ruby `bitcoin-ruby ~> 0.0.18`. Everything else is a standard SHA-256 utility and a web browser.

Troubleshooting

	Likely cause	Fix
Hash mismatch	A file was modified	Use the exact original upload, byte for byte
Address mismatch	Wrong protocol version	Derive both P2WSH (v2) and P2SH (v1)
Key derivation fails	Wrong hash format	Ensure the fingerprint is lowercase HEX
Transaction not found	Wrong network	Search a Bitcoin **mainnet** explorer

If all five steps line up and the addresses match, the proof holds - independently of Bernstein, indefinitely.

8 · WHY BITCOIN

8 What makes the proof hold

The protocol's strength comes from stacking independent cryptographic guarantees - any one of which is hard to break, and which together make a certificate practically impossible to forge or repudiate.

WHAT PROTECTS THE CERTIFICATE

- **Proof-of-work permanence.** Altering a confirmed certificate means redoing the block's proof-of-work and every block since - economically infeasible.
- **3-of-3 multisig.** The address needs all three keys; no single party - Bernstein included - can forge or alter it alone.
- **One-way fingerprint.** SHA-256 cannot be reversed to recover files, and no two inputs realistically share a fingerprint.
- **No private keys stored.** Bernstein never holds your owner key, so it cannot be stolen from us or used to impersonate you.

WHAT IT DELIBERATELY DOES NOT DO

- **It is not a signature of identity.** It proves control of a key, not a legal name - identity binding is a separate, optional layer.
- **Time is block-level.** Precision is ~10 minutes, so Bernstein anchors qualified timestamps alongside for to-the-second, regulator-grade time.
- **It certifies a hash, not meaning.** It proves *what* existed and *when*, not whether the content is original or lawful - that is for the holder to assert.

PRIVACY BY CONSTRUCTION

Your files are encrypted client-side with zero-knowledge encryption *before* anything is certified, and only a hash is ever anchored. The blockchain sees a fingerprint and three public keys - never your content, never your private keys, never your identity. You get a permanent public proof while your actual data stays entirely under your control.

9 · VERSIONS & COVERAGE

9 Protocol versions, and anchoring beyond Bitcoin

The protocol has evolved while keeping every historical certificate verifiable, and Bitcoin is reinforced by independent timestamping authorities so the proof holds under several trust frameworks at once.

	Protocol 1 · legacy	Protocol 2 · current
Address type	P2SH multisig	✓ P2WSH SegWit multisig
Fingerprint	Cover hashed twice, then combined	Sorted concatenation, single scheme
Fees	Legacy scriptSig - larger	✓ Witness-discounted - cheaper
Still verifiable	✓ Yes - by P2SH derivation	✓ Yes - by P2WSH derivation

Older certificates remain fully verifiable - a verifier simply derives the matching address type. No certificate is ever orphaned by an upgrade.

BITCOIN IS THE BACKBONE - NOT THE ONLY ANCHOR

Every Bernstein certificate is anchored to Bitcoin *and* corroborated by independent timestamping authorities - an EU qualified (eIDAS) timestamp, a China TSA, and a global RFC 3161 authority - so it carries borderless permanence and regulatory weight together. Bitcoin needs no authority and nothing to renew; the qualified timestamps add to-the-second time and a legal presumption inside their frameworks. See the companion *Trusted timestamping* brief for that layer.

P2WSH SegWit · multi-anchor LIVE

Protocol 2 in production on every new certificate, anchored to Bitcoin plus EU, China and global timestamping authorities.

Legacy P2SH certificates MAINTAINED

All protocol-1 certificates stay independently verifiable, indefinitely, by the same open derivation.

Enterprise & white-label ON REQUEST

Additional anchors or jurisdictions, and tailored verification packs, for enterprise and white-label deployments.

Want this brief tailored to your jurisdiction or folded into a procurement pack? Contact your Bernstein partner - www.bernstein.io.