BERNSTEIN

TECHNICAL BRIEF

INNOVATION EVIDENCE PLATFORM

The Bitcoin certification protocol

How Bernstein turns your files into verifiable evidence of existence, integrity and possession - anchored to the Bitcoin blockchain, and verifiable by anyone without trusting Bernstein.

Every Bernstein Bitcoin certificate corresponds to 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
October 2026	Technical & procurement	3-of-3 multisig · P2WSH

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016
October 2026

1 · THE BASICS

What a Bitcoin certificate proves

Bernstein uses the Bitcoin blockchain as a public notarization layer. It creates a cryptographic fingerprint of your data, constructs a Bitcoin address committing to that fingerprint and creates a transaction paying to that address. This is the registration transaction: once included in a Bitcoin block, it provides a public record against which the certified content can be checked.

In Bernstein, a **project** is a collection of files documenting a piece of work, with a title and description. A **hash function** turns data of any size into a fixed-length value, called a hash or cryptographic fingerprint. The same input always produces the same hash. Even a small change produces a completely different fingerprint. Hashing cannot reconstruct the original data and is distinct from encryption.

THREE QUESTIONS, ONE TRANSACTION

Each certificate answers **WHEN** (the block timestamp and position in the chain), **WHAT** (a one-way fingerprint of your exact content), and **WHO** (the owner key associated with the certificate). The transaction's address is derived from the content and public keys. Its place in a dated block supplies the chronology. A signature made with your owner key can separately demonstrate control of that key.

Bernstein encrypts your files on your device before they reach its servers and certifies a fingerprint of their content. Bernstein stores the encrypted, versioned files for later validation but cannot read them. What is committed to Bitcoin - and what a court or expert can later re-check - is the 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 catching up with the chain after it. Subsequent confirmations make this increasingly difficult.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

What is - and is not - written to the chain

	On the blockchain	Never on the blockchain
Your data	A cryptographic commitment to the content fingerprint	✗ The files or any clear-text
Your keys	A commitment to a script built from three public keys. The script can be revealed when spent	✗ Any private key
Identity	An association with the owner public key	✗ No name or verified identity is required by this protocol

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

Three keys, one address

A **private key** authorizes cryptographic signatures, and its corresponding **public key** lets others check them. Every certificate address is a **3-of-3 multisignature** built from three public keys. Combining them binds the certificate to your content and to the owner key. Anyone with the public inputs can reconstruct the address; spending its output requires signatures corresponding to all three keys.

Owner key

Generated on your device, per vault and project. Associates the certificate with a signing key you control. Bernstein stores the private half only in client-side-encrypted form.

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.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

Combining the keys into the address

The three public keys are assembled into a 3-of-3 multisig script (a Bitcoin spending condition requiring all three signatures), always in the same order: owner, data, Bernstein. The keys are never sorted, and sorted keys give a different address. Since 20 August 2018 the script is hashed into a SegWit **P2WSH** (Pay to Witness Script Hash) address. Earlier certificates use a **P2SH** (Pay to Script Hash) address. The address type is separate from the fingerprint protocol (§9). 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
SIGNATURES REQUIRED TO SPEND	SEGWIT MULTISIG ADDRESS SINCE AUGUST 2018	UNENCRYPTED OWNER PRIVATE KEYS STORED BY BERNSTEIN

3 · THE FINGERPRINT

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.

SHA-256 is the hash function used here. It produces a 256-bit fingerprint, normally displayed as 64 hexadecimal characters. The project cover is a text file named `Cover – <Version ID>.txt`. It contains the project title, description, project ID and version ID, and it is regenerated for every version. It does not contain a protocol version. The cover is hashed alongside the other files. Preserve the cover of each certified version exactly as exported.

1 · Hash each file

Every file is hashed individually with **SHA-256**, including the version's cover file (title, description, project ID and version ID).

2 · Sort & concatenate

The lowercase hex hashes are sorted alphabetically and concatenated as text 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). Finding different files that produce the same fingerprint is computationally infeasible, 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 independently reproducible for validation.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

4 · THE PROTOCOL

Registration, end to end

Certifying a project runs a fixed five-step pipeline - from raw files to a confirmed Bitcoin transaction. Fingerprint and address construction are deterministic and independently reproducible. Transaction confirmation depends on the Bitcoin network.

1 · Fingerprint

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

2 · HEX → WIF

Represent the fingerprint as a Bitcoin data-key value in Wallet Import Format (WIF), an encoding used by Bitcoin tools.

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, in that order, into a 3-of-3 multisig P2WSH address (P2SH before 20 August 2018).

5 · Anchor on-chain

Send a small amount of bitcoin (the persisting value) to the address and broadcast the transaction. Once confirmed, the certificate exists.

Result

A confirmed Bitcoin transaction whose address commits to your content and owner key, recorded in a dated block.

Anatomy of the registration transaction

WHAT THE TRANSACTION CARRIES

A Bernstein fuel input pays the network fee and, on a first certificate, also funds the persisting value. The transaction writes three outputs: (1) the certificate output - the multisig address, carrying a small 'persisting value' of bitcoin that flows forward into the next certificate in the trail, (2) an **OP_RETURN** marker of the form `BERNSTEIN <format version> REG <project ID>` (format 1.0 on older P2SH certificates, 2.x since P2WSH), a human-findable tag on the chain, and (3) change. Your original files are not published in the transaction.

Confirmation is at the block level. Bitcoin targets an average block interval of about ten minutes. This is not a guarantee of timestamp accuracy or confirmation time. Block timestamps are miner-supplied and constrained by Bitcoin's consensus rules.

5 · THE TRAIL

An immutable trail of versions

Innovation is not a single moment - it evolves. When a project's contents change, a new version remains a draft until you certify it. Bernstein then 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.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

Initial certificate

The first registration of a project. A fuel input pays the fee and funds the persisting value. 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 transaction references the previous certificate's output, 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.

When a trail starts, the persisting amount is set to about €1.62 in satoshis (2,600 sats as a fallback). Since May 2026 it is carried forward unchanged from one certificate to the next. In older trails it may vary from link to link. Network fees are paid by a separate fuel input, not from this amount.

6 · VERIFICATION

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 and the exported cover 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 data public key you derived with the owner and Bernstein public keys from the certificate PDF, in the order owner, data, Bernstein, to reconstruct 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 supplied files reproduce the commitment in that Bitcoin record. A mismatch can result from different files, keys or protocol rules.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

PROVE EXISTENCE AND DEMONSTRATE CERTIFICATE CONTROL

The same tooling does three jobs: **prove existence** by matching the fingerprint registered on-chain, **demonstrate certificate control** by signing with your owner private key, and **generate a signature** a third party - a court, an investor, a counterparty - can independently validate. Content matching does not require your owner private key. The official validator hashes your files in your browser and compares the result with the registered fingerprint. It does not rebuild the address or check the blockchain. Steps 2 to 5 do that, independently of Bernstein, with standard, off-the-shelf cryptographic tools.

	Where it comes from
Original project files and exported cover	Your own copies - the exact uploads and the `Cover – <Version ID>.txt` file of that version
Transaction ID	The certificate PDF
Owner & Bernstein public keys	The certificate PDF
SHA-256 + key tools + explorer	Open, standard, off-the-shelf

Preserve the exact files, cover and certificate data. With these inputs and access to the Bitcoin record, validation does not depend on Bernstein staying online.

7 · STEP BY STEP

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 and the version's cover file (`Cover – <Version ID>.txt`) with SHA-256. Sort the lowercase hex hashes alphabetically, concatenate them as text with no separators or line breaks, and hash that string once more. The result is the 256-bit project fingerprint (§3). The command below does all of this in a single pipeline. Building the string with `echo` (which adds a line break) or hashing the joined hex as binary gives a wrong fingerprint. For protocol-1 certificates, hash the cover hash once more before sorting (§9).

```
# macOS / Linux: hash each file and the cover, sort the hex hashes,
# join them as text and SHA-256 the result once more
shasum -a 256 "Cover – <Version ID>.txt" file1.pdf file2.png \
  | cut -d' ' -f1 | LC_ALL=C sort | tr -d '\n' | shasum -a 256

# → the 64-character hash printed is the project fingerprint
```

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

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

Use a trusted local Bitcoin key-compression tool to obtain the compressed **data public key** from the WIF data key - computed from your content alone, with no Bernstein input. The optional script below does this offline. Do not submit confidential files, content-derived keys or owner private keys to arbitrary online services.

```
# wif2pub.py — data key (WIF) → compressed public key (HEX), offline
import sys, base58
from ecdsa import SECP256k1, SigningKey

raw = base58.b58decode_check(sys.argv[1])
assert raw[0] == 0x80 and len(raw) == 34 and raw[33] == 0x01
print(SigningKey.from_string(raw[1:33],
curve=SECP256k1).get_verifying_key().to_string("compressed").hex())
```

```
$ pip3 install base58 ecdsa
$ python3 wif2pub.py <your-data-key-wif>
```

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. Pass the keys in this exact order: owner, data, Bernstein. Do not sort them, because sorted keys give a different address. Certificates registered since 20 August 2018 use P2WSH (addresses starting with bc1q). Earlier certificates use P2SH (addresses starting with 3).

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

```
# 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

# Keys must be passed in this exact order: owner, data, Bernstein (do NOT sort them).
p2sh, witness = Bitcoin::Script.to_p2sh_multisig_script(
  3, owner_key, data_key, bernstein_key)

# P2WSH (SegWit) — addresses starting with bc1q
program = Digest::SHA256.digest(witness)
puts Bitcoin.encode_segwit_address(0, program.bth)

# P2SH (legacy) — addresses starting with 3
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, your files reproduce the commitment registered in that Bitcoin record. No match means the files, keys, key order or protocol rules differ. This check does not require the owner private key or prove legal ownership of the work. The official validator at **app.bernstein.io/validator** hashes your files in your browser and compares them with the registered fingerprint. It does not rebuild the address or check the chain, so steps 2 to 5 remain the check that is independent of Bernstein.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

Tools reference

	Where	What it does
Bernstein Validator	app.bernstein.io/validator	Official verifier - hashes your files in your browser and compares them with the registered fingerprint. It does not rebuild the address or check the chain
File hash	`shasum -a 256` (local terminal)	SHA-256 each file and the `Cover - <Version ID>.txt` file, then the joined hashes (step 1)
Key compression	`wif2pub.py` or another trusted local Bitcoin key tool	Derive the compressed public key from the data-key WIF, offline (step 3)
Blockchain explorer	blockchair.com	Look up the transaction and inspect its outputs (step 5)
Ruby runner	Local Ruby installation	Run the multisig script locally (step 4)

Dependencies: Python `base58` and `ecdsa`, Ruby `bitcoin-ruby` (~> 0.0.18 installs 0.0.20 and needs a C compiler). Everything else is a standard SHA-256 utility and a web browser.

Troubleshooting

	Likely cause	Fix
Hash mismatch	A file was modified, or the cover is from another version	Use the exact original upload, byte for byte, and the cover file of the certified version
Fingerprint mismatch	Hashes joined with line breaks, or hashed as binary	Join the lowercase hex hashes as text with no separators (Step 1 command)
Address mismatch	Wrong fingerprint protocol, address type or key order	Apply the fingerprint protocol and the address type separately (§9), and pass the keys as owner, data, Bernstein without sorting
Key derivation fails	Invalid hash or key encoding	Check the fingerprint length, hexadecimal format and key encoding
Transaction not found	Wrong network	Search a Bitcoin **mainnet** explorer

A successful check reproduces the registration output from the certified inputs. Preserve those inputs so the check remains possible independently of Bernstein.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

8 · WHY BITCOIN

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 resistant to tampering and independently checkable.

WHAT PROTECTS THE CERTIFICATE

- **Proof-of-work permanence.** Altering a confirmed certificate means redoing the block's proof-of-work and catching up with the chain after it. Subsequent confirmations strengthen the record.
- **3-of-3 multisig.** Spending the certificate output requires signatures corresponding to all three keys. Reconstructing the address for validation requires only public inputs.
- **One-way fingerprint.** SHA-256 cannot be reversed to recover files, and no two inputs realistically share a fingerprint.
- **Owner key generated on your device.** It is created per vault and project, and Bernstein stores the private key only in client-side-encrypted form, never in clear.

WHAT IT DELIBERATELY DOES NOT DO

- **It is not a signature of legal identity.** Key control does not by itself establish authorship, ownership, originality or a valid transfer of rights. The record can support those claims alongside other evidence.
- **Time is block-level.** The block records a miner-supplied timestamp, subject to consensus rules. The ten-minute target interval is not a timestamp-precision guarantee.
- **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 before reaching Bernstein's servers, and only a cryptographic commitment is anchored. The registration does not publish your original content or require your name. Bernstein stores the encrypted certified versions for later validation but cannot read their content.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016

9 · PROTOCOL VERSIONS

Protocol versions

The protocol has evolved while keeping historical certificates verifiable. Two version axes apply, and they changed on different dates: the **fingerprint protocol** (protocol 1 before 23 November 2018, then protocol 2) and the **address type** (P2SH before 20 August 2018, then P2WSH). Check each one against the certificate and apply each rule separately.

	Legacy · older certificates	**Current** · since 2018
Fingerprint protocol	Protocol 1, before 23 November 2018: the cover hash is hashed once more, then all hashes are sorted, concatenated and hashed	✓ Protocol 2, from 23 November 2018: SHA-256 of the sorted lowercase hex hashes, concatenated as text
Address type	P2SH multisig, before 20 August 2018 (addresses starting with 3)	✓ P2WSH SegWit multisig, from 20 August 2018 (addresses starting with bc1q)
OP_RETURN marker	`BERNSTEIN 1.0 REG <project ID>`	`BERNSTEIN 2.x REG <project ID>`
Fees	P2SH: legacy scriptSig - larger	✓ P2WSH: witness-discounted - cheaper
Still verifiable	✓ Yes - with the fingerprint protocol and address type that applied to the certificate	✓ Yes - with protocol-2 rules and P2WSH derivation

Older certificates remain verifiable using their original rules. Preserve the exact files, the cover file of each certified version and the certificate data.

Questions about the protocol or independent validation? Contact the Bernstein team - www.bernstein.io/contact.

©Bernstein by IP LEVERAGED SAS - www.bernstein.io · Trusted IP protection, established 2016