

bfinger: a name that proves what it says

Ask what `alice@example.com` currently says about itself and get an answer that proves itself, from any host, checked by your own machine. bfinger is the first application in the bstack pattern series: a self-published identity and status record whose truth no longer depends on who served it, delivered once to every host by BSV Layered Multicast, and payable from the identity that just proved itself.

Author: Jeff Harris · Lightweb Inc. · jeff@lightweb.net

Pattern 001 · Version 01

Audience: application developers, founders and business leaders. No Bitcoin or networking background assumed; every term is explained on first use.

We = Lightweb Inc., designer of the bstack patterns and operator of the 1BSV network; **you** = a builder or decision maker evaluating them.

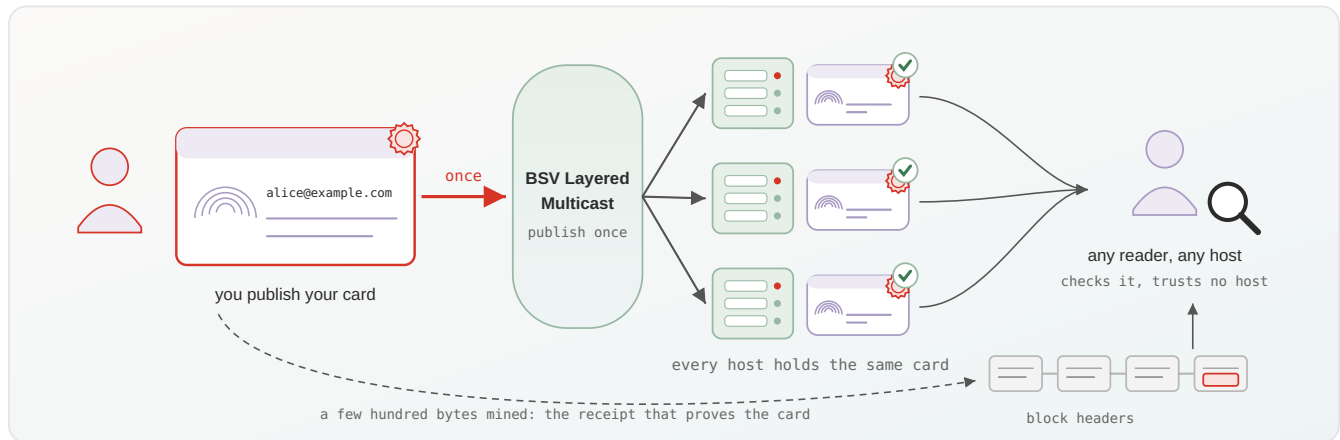


Figure 1: one self-published card, held identically by every host, verified by every reader.

The series, and where this fits

The bstack pattern series is a set of short briefs, one application each, meant to be read in sequence. Each one takes a building block that nearly every product needs (an identity, a message, a team, a secret, a record) and rebuilds it so that the data defends itself. Three ideas carry the whole series:

- **State proves itself.** Every change is a Bitcoin SV transaction, signed by its owner and anchored in a public ledger nobody can rewrite.
- **Publish once.** BSV Layered Multicast, an open networking standard, delivers each publication once to every host that subscribed to it, repairing any loss on the way.
- **Readers verify, so hosts compete.** Any reader checks what it is handed against data it holds itself, so a host is a replaceable service, never the keeper of the truth.

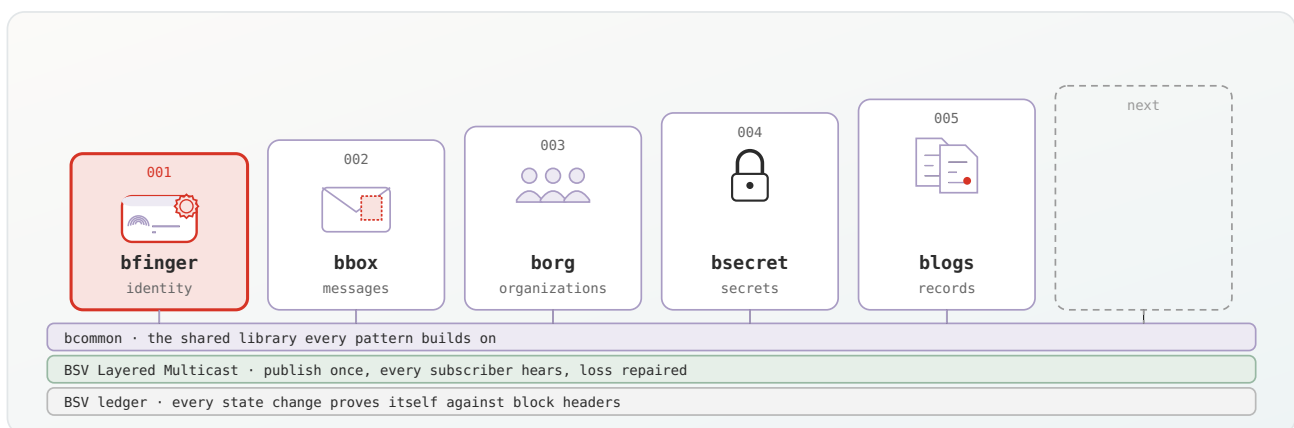


Figure 2: the series as one build. bfinger, in red, is where it starts.

Every pattern stands on its own, with its own repository and its own specification, and every pattern is built from the same foundation, so each one makes the next cheaper to build and easier to trust. bfinger comes first because the question it answers comes first in any system: who is this, and what do they say right now?

The problem

The internet's answer to "who is this?" is a server you are asked to trust. A profile page, a directory API, a social handle: each one returns whatever its operator's database holds, and HTTPS proves only that you reached the right server, not that it told the truth. Take the server away, or take it over, and the answer is gone or wrong, with nothing in your hands to tell the difference.

The old `finger` command had the same flaw. Ask `finger alice@example.com` and a program on example.com's machine told you who alice was; the answer was true because you trusted that machine. `bfinger` keeps the question and moves the truth.

What bfinger does

`bfinger alice@example.com` answers one question: what does that address currently claim about itself? The answer is a small record the owner publishes: a status line, a plan, any fields they choose. It arrives with everything needed to check it, and the first word of the reply says whether it held up.

```
$ bfinger -header-url woc:main alice@example.com
alice@example.com
VERIFIED signature, sequence 7 (update), proof at height 912430
key      02abab...abab (pinned 2026-04-11)
plan     shipping 2.0
status   available
```

Figure 3: a read. The values are illustrative; the first word is the verdict.

VERIFIED means the signature, the sequence, the key and the Bitcoin proof all checked against block headers from a source you chose, never from the host that answered. **VERIFIED-UNMINED** means every check passed but the latest update is not in a block yet. A refusal, or no record at all, and nothing the host said is printed as true. The same check is a command away for anyone: the 1BSV network's own identity, `1bsv@lightweb.net`, resolves on the BSV mainnet today, and its key fingerprint is printed on 1bsv.net for comparison.

How it works

Four roles, and the pairs you would expect to be coupled never need to touch. The **owner** signs and publishes. The **network** delivers. A **host** stores what arrived and answers questions. A **reader** asks any host, or several, and checks everything itself.

1. **The name points to a key.** The domain says which identity key (the public half of the owner's signing key) belongs to `alice`, and which hosts carry her record. Your copy of `bfinger` remembers that key on first contact, the way SSH remembers a server, and refuses a changed key unless the old key signed the change.
2. **Each update mines a receipt, not the data.** An update reaches the ledger as a few hundred bytes that commit to the record and spend the previous update, so the history is one unbroken chain that cannot fork.
3. **The record travels beside the receipt.** The record itself rides a transaction deliberately built so it can never be mined. That gives it a permanent name (its transaction id), proof that it belongs to the owner's history, and a stop button.
4. **Retraction is real.** One spend by the owner, called a sweep, makes every record it covers invalid at once, and every honest host stops serving them. The chain keeps a commitment to each update and when it was mined, so anyone who kept a copy can still prove it matched.
5. **Readers check against their own headers.** Block headers are the ledger's compact summary, 80 bytes per block. You choose where they come from; read from a public source, `bfinger` checks their proof of work itself, so the yardstick need not come from the host that answered.

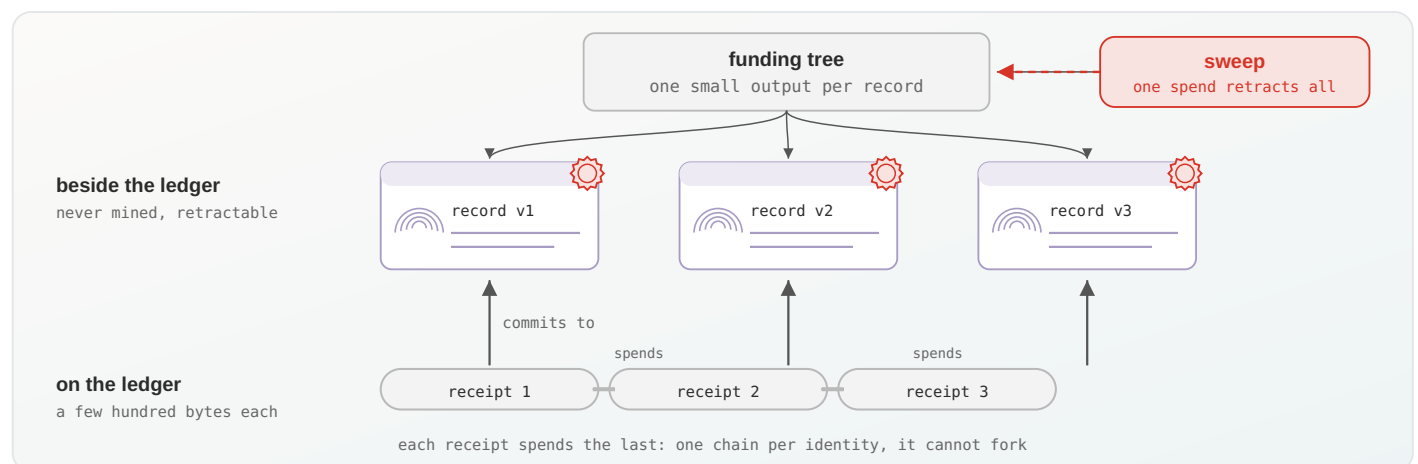


Figure 4: the committed record. A few hundred bytes are mined per update; the record rides beside it and can be withdrawn.

Two safeguards make the record hard to hijack. Producing the next update needs the signing key and a second secret kept in the owner's home, two factors that can live in two places. And a host that edits a record produces something that fails the check, so a dishonest host cannot forge or alter a record: at worst it withholds or serves something stale, and asking two hosts exposes any disagreement.

Words in this brief. *Identity key*: the public half of a signing key; it names you and is how others pay you. *Block headers*: the ledger's compact, self-checking summary. *Host*: a server that stores records and answers lookups, interchangeable with any other. *BRC*: a published Bitcoin standard (Bitcoin Request for Comments).

Why it is new

- **The truth moved from the server to the record.** Profiles, directories and status pages have always been as honest as their operator. Here a record is checked by arithmetic on the reader's own machine, so the operator's honesty stops being part of the product.
- **Content addressing, provability and retraction in one device.** A transaction built never to be mined names the record, proves where it belongs, and lets the owner withdraw it. It uses nothing but the transaction rules every Bitcoin verifier already implements: no new consensus rule, no proprietary protocol.
- **Hosts are replicas, not authorities.** Every host holds the same verifiable bytes. Adding a host is a subscription on the plane, or a catch-up from a peer without it; choosing one is an availability question, not a trust question.
- **Identity is a payment address.** `bfinger pay alice@example.com` derives a fresh payment destination from the identity key the lookup just verified (BRC-29). You pay the key that proved it controls the name, not an address pasted from an email. A BRC-100 wallet, the BSV standard for wallets that applications talk to, can hold the identity and the funds.
- **Open standards.** The record is a standard transaction package with its proofs attached (BEEF), so anything that already carries BEEF carries a bfinger record unchanged. bfinger's code is Apache-2.0 (one dependency carries the Open BSV License; its NOTICE explains).

Built to scale: BSV Layered Multicast

bfinger runs over plain point-to-point connections today: one host, self-hosted, is a complete deployment, and a good source of block headers is the only hard dependency. What changes with scale is distribution. Without a network, every extra host is another copy to upload, and hosts that keep each other in step move every record on the order of N-squared times.

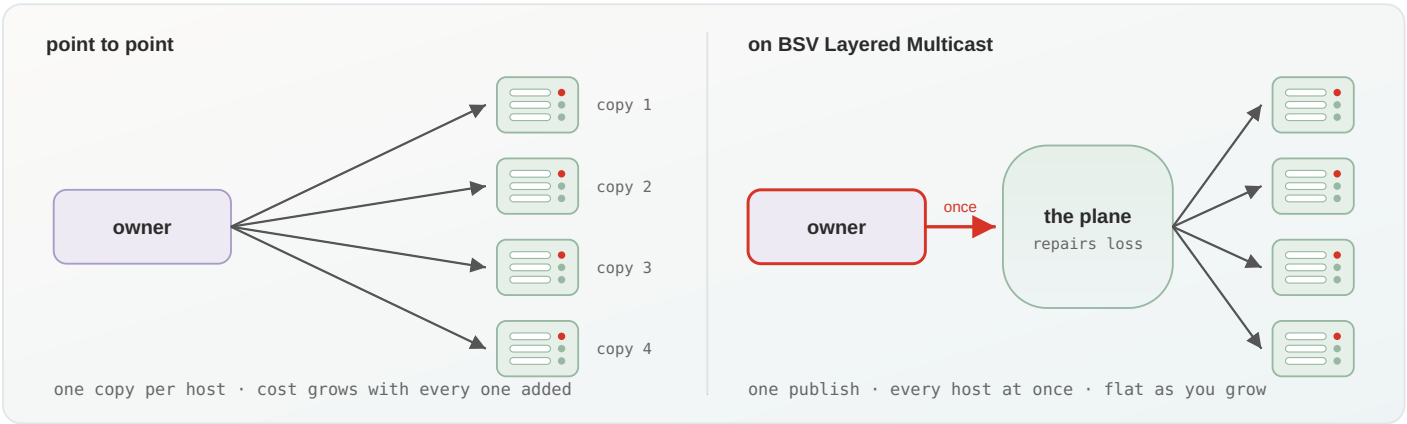


Figure 5: one publish reaches every subscribed host at once; the network repairs loss.

BSV Layered Multicast removes that cost structurally. The owner submits once; every host subscribed to the record's topic receives it at the same moment, complete and verbatim; loss is detected and repaired by the network rather than by the application. Adding the hundredth host adds nothing to the owner's side. The patterns are the architecture; multicast is how the architecture scales.

What it means for the business

Who	Pays	Gets
The owner	miner fees only: a fraction of a cent covers a first record and several updates	a record nobody can quietly alter, retractable at will
A reader	nothing for the base answer, by specification	an answer that proves itself, with no account and no credential
A host operator	its own hosting and the delivery it elects	a seam to price richer questions above the free answer
Anyone paying the owner	the payment, directly	a destination derived from a verified identity

- **Trust becomes a check, not a promise.** Disputes about what someone said, and when, reduce to arithmetic.
- **The free base answer is structural.** A lookup carries no account, so no host can charge for the base answer without knowing who asked. Paid services sit above that floor: availability, history, retention, convenience. A price attaches to a class of question, never to part of an answer.
- **No lock-in, by construction.** Records are a commodity every host holds; operators compete on service, and customers can leave without losing their data.

Honest limits

- **The record is public.** Write it like a bulletin board entry. Lookups carry no account, but the host learns which identity was asked about and from what network address, the subject's domain sees the resolution, and the cadence of updates is visible.
- **Retraction is not erasure.** It ends availability from honest hosts; it cannot un-copy what someone already fetched.
- **The domain still names the key** until handle certificates are adopted (borg, Pattern 003, verifies them).
- **Seeing a retraction needs a host or a spend index.** Block headers prove what was mined, not that a coin is still unspent.
- **The owner's home holds the second factor.** It is the one thing the network cannot reconstruct, so back it up with the care you give the key itself.

What comes next

bfinger ends where most products start to need more than a directory: you can now pay a verified identity, but the payment and the note that goes with it still need a way to reach that person. **bfinger pay** writes its payment notice to a local file; a message box is where such notices belong. Pattern 002, **bbox**, is that box: envelopes sealed to one recipient, kept by several independent hosts, priced by postage built into their structure, with the payment traveling inside the envelope. bfinger told you who someone is; bbox lets you reach them, and pay them, privately.

Run it on 1BSV

The 1BSV network is the commercial operation of BSV Layered Multicast, run by Lightweb Inc. For an application built on bfinger:

- **Your users ride free.** Publishing into the network is never billed, and neither are the network's repair retransmissions.
- **Hosts pay for what is delivered to them**, and for nothing else: overlay objects with their proofs on an application account, which also carries the live header feed to verify against. No subscriptions, no minimums, no commitments.
- **Early access.** The overlay lane is onboarding its first users. Onboarding is self-service with one approval step.

Go deeper

- bfinger, code and documentation (Apache-2.0): <https://github.com/lightwebinc/bfinger>
- *The bstack: publish once, prove everything, and let users ride free* (Lightweb Inc., Paper 009): <https://1bsv.net/papers/bstack-patterns.pdf>
- *The overlay object plane* (Lightweb Inc., Paper 006): <https://1bsv.net/papers/overlay-object-plane.pdf>
- The pattern series: <https://1bsv.net/patterns.html>

1BSV - BSV Layered Multicast - cash moves fast on 1BSV. · 1bsv.net

Lightweb Inc. · Jeff Harris · jeff@lightweb.net · github.com/lightwebinc

Unbounded Solutions for a Small World™ · © Lightweb Inc.