WataFast: A Multipath, Censor-Aware Tunnel Protocol with Hybrid Post-Quantum Key Agreement

Technical whitepaper, public edition 1.0 (November 2026) Author: evilork (protocol designer and lead engineer) | **Organization:** Watafast LLC (Wyoming, USA) | **Contact:** hello@wata.fast

> **About this edition.** This is the public edition of the WataFast technical whitepaper. It follows the internal protocol specification (v2, draft 0.7, October 2026) and the implementation README; "§N" points to a section of that specification so that reviewers who read it under agreement can cross-check every statement. The specification itself is not published. This edition omits the operating constants that are set per deployment and would help a filtering operator (filler sizes, attempt pacing, per-account entry counts, verdict thresholds); nothing else was changed in substance. Where the sources say something was not measured, this paper says "not measured". All figures in Section 5 are reproduced from specification section 13.

---

1. Abstract

WataFast is a tunnel protocol for one narrow problem: keeping an ordinary user connected to the open internet when national operator-level filtering (in Russia, DPI equipment commonly called TSPU) blocks or degrades circumvention protocols by signature, behaviour and address. It is implemented as one Rust codebase of nine crates (about 50,800 lines) for desktop, iPhone, iPad, Apple TV, Android and server nodes.

The design combines six ideas: (1) a hybrid post-quantum handshake, Noise IKpsk2 with an ML-KEM-768 secret in the pre-shared-key slot (§5); (2) one session over several carriers at once, which outlives any of them (§8); (3) a "curtain" defence against networks that freeze a connection after its first 16–20 KB (§8.5); (4) every device as a sensor, privately reporting which entries worked on which kind of network (§8.6); (5) keyed per-account distribution of entry addresses, so the account that leaked a blocked entry can be identified (§8.3); (6) a browser-identical TLS hello built with BoringSSL and tested by equality against a recording (§7.1.1).

The specification reports measurements against VLESS (Reality, Vision) on imitated censors, including several results where WataFast was behind VLESS in throughput (§13.5–§13.10). This paper reproduces both. A limit stated by the specification itself: at the time of the snapshot nothing had been measured from real Russian access networks (§7.1, §13). This is a documented, tested design measured under imitation, not a field-proven result.

2. Problem

2.1 Setting and threat model

The specification's first goal is to "keep paying users connected when VLESS/REALITY is filtered by protocol or behaviour" (§1). Its threat model (§2) assumes an adversary ("TSPU / operator") able to: block a node address; drop 27–33 % of packets on an address; count a subscriber's new connections; let the first 16–20 KB of a connection through and freeze it; block by the server name in the TLS hello; match a protocol signature or TLS fingerprint; notice many parallel connections to one name or the shape of TLS inside TLS; probe actively; replay a captured first message; harvest entry addresses through a bought subscription; tie entry address to exit address; record traffic for later decryption; and enforce address-based whitelists.

2.2 Why existing protocols fall short, per the specification

The specification argues this narrowly, and this paper does not extend it.

- **VLESS/REALITY** is described as "filtered by protocol or behaviour" in the target network (§1). It is the measured baseline (§13.5–§13.10) and stays in the application "as an independent fallback line, outside this protocol" (§7.4).
- **Behavioural rows.** A censor that counts new connections, freezes after 16–20 KB, or burns an address defeats any design where one session is one connection to one address. The response is structural: a session that survives loss of its connections (§8) and a disciplined, paced schedule of connection attempts (§8.4).
- **Imitated evidence.** Imitation results (§13.5, Section 5.5) favour WataFast on address freezing and port death, but are on a test node, not Russian networks.
- **Whitelists.** No protocol passes an address-based whitelist; this is out of scope (§1, §11).

The specification contains **no individual analysis of Hysteria2, Shadowsocks or XHTTP**. XHTTP appears only as part of the VLESS fallback line (§7.4). Any claim that those protocols fail in a specific way would be this paper's own; none is made.

Non-goals (§1): inventing ciphers; hiding that some TLS connection exists; passing whitelists by protocol means; compatibility with third-party clients. Only the organization's application and nodes speak the protocol (closed code, key-gated nodes, server-driven parameters).

3. Design

3.1 Identities and credentials (§4)

A device key (X25519, private half never leaves the device), a node key (X25519 per node) and an issuer key (Ed25519, held by the backend, rotatable by key id). A credential is a fixed 160-byte issuer-signed statement binding a device key to an account and limits (plan, rate limit, quota, validity). **Nodes are**

stateless about users: they need only issuer public keys and a small revocation set. A node checks version, key id, strict signature, time window, that the credential's device key equals the key authenticated in the handshake, and that neither device nor account is revoked.

3.2 Handshake (§5)

`Noise_IKpsk2_25519_ChaChaPoly_BLAKE2s`, with an **ML-KEM-768 shared secret in the PSK slot**: an attacker must break both X25519 and ML-KEM to recover traffic keys. The first message encrypts the device's static key and payload to the node, so a passive observer learns nothing about the device. The specification also states: a short timestamp window with a bounded replay cache (§5.4); anything that is not a valid first message gets the carrier's decoy behaviour and no protocol bytes (§5.2); a device with a wrong clock corrects itself from the HTTP `Date` header of the decoy answer; and parameters are tag-length-value lists whose unknown tags are skipped, so either side can learn a new one alone (§5.6).

Stated limits (§5.5): identity privacy of the first message is classical only; the ML-KEM implementation is not independently audited and is used only in hybrid; a leaked node key exposes identities in recorded first messages but not recorded traffic; the replay cache lives in process memory.

3.3 Record layer (§6, §6.1)

Records are sealed with ChaCha20-Poly1305 and hold frames for streams, window updates, datagrams, ping/pong, padding, close, and multipath control. Opening a stream costs no round trip: data follows the open frame while the node dials. Stream and control frames are **numbered** and kept by the sender until the peer confirms them on any carrier. UDP datagrams ride the same records without retransmission, with a short queue and a staleness drop so a call is not held back by a download; UDP towards the HTTPS port is dropped on purpose so browsers fall back to TCP.

3.4 Carriers (§7)

- `h2` **(§7.1).** The node serves a real static website over HTTPS with a publicly trusted certificate. The session is one long-lived HTTP/2 POST to a per-node path. If a request is not a valid first message, the answer is "byte for byte what the site says about any unknown path", so a prober meets an ordinary site.
- **Name and page-first (§7.1.2).** An entry may show no name, or another name, in the hello while the certificate is always verified against the true name. With page-first the device loads the front page and a few files before opening the session.
- `relay` **(§7.2).** A keyless forwarder: the address a device connects to and the address traffic leaves from can belong to different providers.
- `tcp` **(§7.3).** Tests only: random-looking bytes on an unusual port are themselves a signature.
- **UDP/QUIC (§7.4).** Not built.

3.5 One session over several carriers (§8)

A session runs over up to four carriers, each with its own handshake and keys, so "nothing an observer sees on one says it belongs with another" (§8.1). A leading carrier plus standing-by carriers that receive only as much as they have shown they deliver on time; stalled carriers are passed over and their frames resent

elsewhere; a dead carrier is forgotten with no loss to the session. A session with no carrier **waits** for several minutes and resumes where it stopped (§8.2), which is how sleep and network changes are survived. Small frames and datagrams are duplicated on a second carrier within a bandwidth cap. On 2026-10-13 **per-stream order ("lanes", §8.1.1)** was added so a phone with little memory is not limited by reordering room. Connection attempts are disciplined: few at a time, spaced out, last-working entry first (§8.4).

3.6 The curtain (§8.5)

The node sends a fixed filler of empty records right after its reply; the device counts them. A carrier on which all arrive is "proven"; one that stalls first is closed, reported and avoided. If every address is behind a curtain, **short carriers** (§8.5.1) with a small byte measure are used one after another. On the imitation where every entry freezes after 16 KiB, a small request took 0.5–1.3 s and 100 KB took 18 s, which the specification says suits "text and a patient download, not pages or video".

3.7 Keyed entry distribution (§8.3)

No device learns the entry pool. Each account is deterministically given a few entries, chosen by a keyed function over account and entry identifiers, preferring different groups (one provider or address). The key is shared by all nodes, so asking again or from another device reveals nothing new. The holder of the key can list which accounts knew a set of blocked entries.

3.8 Every device is a sensor (§8.6)

After a carrier proves itself, and at most once a minute, a device reports entry numbers, a TLS look, an outcome code (passed, unreachable, timeout, TLS failure, no reply, curtain, stalled), two durations and a coarse network label. The specification states it contains no device address, no destination and nothing about the person. A failure is reported only if another carrier worked beside it, so being offline is not blamed on an entry. The node counts an entry as not passing on a network when enough devices there report failures well in excess of successes, and tells devices on that network to try it last; a wrong verdict "costs a moment and never a connection". Reports can be disabled; the privacy policy must describe them before they reach users (§14).

3.9 Node policy (§10)

Rate limits, quotas, a destination ACL, and per-device byte accounting with no destinations. Three further mechanisms: a **side exit** (§10.1), prompted by the specification's report that on the fleet it replaces (2026-10-08) an exit address was placed in Russia by a large search company about two weeks after carrying Russian visitors, so certain services unavailable there stopped opening, which lets chosen host names leave via another address; optional **memory accounting** (§10.2, built 2026-10-14, off unless set; "nothing was measured on a node that serves people"); and **live revocation** (§10.3), a file re-read every 30 s that closes sessions of revoked devices, where a malformed file changes nothing. The node does not fetch the list: delivery to nodes is manual for now.

3.10 Whitelist mode (§11)

Measured twice (2026-08-03, 2026-09-20): filtering is by destination address, a packet to a non-listed address never leaves the operator network, and "no protocol passes it". The specification limits itself to client-side detection with an honest message instead of endless reconnecting, and relay readiness if a listed entry ever exists. A "soft regime" variant is "not measured by us yet". This is a statement of limits, not a bypass.

---

4. Implementation

4.1 Crates and size

Crate	Role
`wf-proto`	credentials, handshake, records, frames; no I/O, clock or globals
`wf-keys`	key and credential files and tool
`wf-session`	multi-carrier session driver, streams, flow control
`wf-carrier`	`h2` carrier and TLS looks
`wf-boring-ext`	three BoringSSL calls; the only crate with `unsafe`
`wf-server`	node daemon
`wf-relay`	keyless relay
`wf-client`	self-healing session behind a local SOCKS5 inbound
`wf-ffi`	client as a library: C interface (iPhone, iPad, Apple TV), JVM entry points (Android)

About 50,800 lines of Rust; 112 commits between 2026-10-12 and 2026-10-29 on the current branch. The project started earlier; no start date is claimed.

4.2 Safety rules

`unwrap`, `expect`, `panic!` and unchecked indexing are denied by workspace lints in production code (tests exempt). Checks are formatting, clippy with warnings denied, and tests, repeated for the browser-look build. `unsafe` is confined to one small crate. Dependencies are permissive-licence only (§3). No new ciphers: Noise, ChaCha20-Poly1305, BLAKE2s, X25519, Ed25519, ML-KEM-768 from existing libraries.

4.3 Platforms

- **Desktop:** an authenticated SOCKS5 inbound; in the application it is one more engine beside the existing one, with fresh credentials passed on every start through standard input (§12.1).

- **iPhone, iPad, Apple TV, Android:** these systems give a VPN service one process, so the client is linked in as a library (§12.2) with wake and network-change hooks and memory calls for the roughly 50 MiB iOS extension limit; the default profile holds at most 8 MiB. §12.2 says "built 2026-10-12, not yet inside an app"; for the state in November 2026 see Section 7.1.
- **Servers:** a static node binary, a keyless relay, a hardened service unit, and a journal that records no addresses.

4.4 The browser TLS look (§7.1.1)

A ClientHello travels in the clear, so a filter can drop a hello no real program sends at no cost to others. The `chrome` look reproduces a recorded Chrome 153 hello (cipher list, seventeen extensions in shuffled order between GREASE values, hybrid post-quantum key share, ALPN, ALPS and more). It uses BoringSSL, the library the browser uses, via the `boring` crate; one missing capability is supplied by a small patch to a copy of the library sources. The test is **equality with the recording, not likeness**: every stable field must equal it while randomised fields must differ between connections.

Not reproduced (§7.1.1): session resumption; the HTTP/2 settings and header order inside the encrypted channel; the traffic shape after the handshake. A phone browser's hello differs, so mobile apps "will need their own recording". The site's answers are recognisably those of the Rust libraries (§14).

---

5. Measured results

Data-centre vantage points do not see the target filtering; "results count only from real access networks" (§13), and **none in this section were taken from Russian access networks**. "Imitation" means packet rules on a test node.

5.1 Plain TCP, European test node, 2026-10-11 (§13.1). Loopback, one stream: 531 MB/s down, 492 MB/s up. From a device 244 ms away with about 1 % loss: 0.48 MB/s before tuning, 6.0 MB/s after BBR and self-widening windows; upload stayed at 0.1–0.3 MB/s, equal to a bare ssh upload on that path. From 13 data-centre vantage points (including Russia, Kazakhstan, Turkey) the session came up; 16–40 MB/s down and 6.5–27 MB/s up from Europe, Russia and Turkey, 8.9 / 3.4 from the USA, 3.5 / 2.4 from Kazakhstan.

5.2 `h2` carrier, same day (§13.2). Session up from all 13 points; speeds matched the plain carrier (Moscow 21.0 / 8.7 MB/s against 18.8 / 6.5). Setup is three round trips (0.22 s from Moscow). A form posted to the session path was answered exactly like one posted to a missing page. Four release-client hellos equalled the Chrome recording field by field.

5.3 Library, 2026-10-12 (§13.3). One Mac about 240 ms away; not a phone, not Russia. Process memory 4.7 MB idle, 6–10 MB during a 40 MB download. UDP: a 20 packets-per-second stream returned 120 of 120, median 240 ms (the path itself is about 240 ms). After a silent path cut, the dead session was given up 2.1 s after the wake call and the new one was up 1.1 s after the path returned.

5.4 Multipath, 2026-10-12 (§13.4). Desktop on a home line outside Russia and China. With both entries cut for 20 s during a 10 MB download, the session waited, recovered, and the file was whole (SHA-256 equal to a direct download). With the carrying entry cut for 20 s, the download continued on the other

carrier with a longest stand-still of 0.8 s. **Not measured:** a phone, a sleeping device, TUN, two providers, Linux and Windows clients.

5.5 Imitated censors next to VLESS, 2026-10-12 (§13.5). Six simultaneous requests (five small, one 285 KB); a page counts when all six arrive within 12 s. Each run was once or twice for a minute, "so the first digit holds and the second may not".

Imitation	WataFast	VLESS (Reality, Vision)
nothing	22 of 22, 1.1 s	14 of 14, 2.2 s
four new connections in a second freeze the address 2 min	31 of 31, rule never fired	0 of 5
first 16 KiB pass, rest held, on one entry of two	27 of 27, first after 2.3 s	(single entry)
same on every entry	0 pages whole; small 20 of 25, large 0 of 5	small 25 of 25, large 0 of 5
resets for 3 s, twice a minute	25 of 25, longest wait 7.6 s	16 of 18, 7 requests broken
main port dead 3 minutes	99 of 99, longest wait 4.1 s	6 of 20
2 % / 5 % / 10 % loss	19 of 19 (1.4 s) / 18 of 18 (1.4 s) / 14 of 14 (2.2 s)	14 of 14 (2.2 s) / 13 of 13 (2.5 s) / 10 of 10 (3.2 s)

Where every entry is behind the curtain, WataFast was behind VLESS on small requests. Megabytes downloaded in 15 s, two samples each:

Loss	One download: WataFast / VLESS	Six at once: WataFast / VLESS
none	148 / 146	138 / 234
2 %	80, 139 / 152, 137	106, 140 / 225, 229
5 %	47, 84 / 113, 104	72, 42 / 25, 17
10 %	8, 26 / 7, 7	8, 25 / 8, 11

WataFast was behind VLESS with several downloads on a clean line and at 2 % loss, and ahead at 5 % loss for six downloads; numbers under loss "move between runs by a factor of two and more, for both". Section 13.8 later found that these "six downloads" were really two, because the speed server held further ones back for both protocols. Load tests (512 simultaneous streams, 500 devices, a flood of strangers) had no failure.

5.6 Long line, 2026-10-12 (§13.6). Desktop 250 ms from the node, Mbit/s, mean and range. One download: before the fixes 67 (50–81), after 74 (50–96), VLESS 78 (60–100). Six at once: 55 to 69, VLESS

77. At 5 % loss, one download: 84 and 32 after, VLESS 8 and 5. On the node as it runs: one download 107 (92–125) against VLESS 56 (19–77); six at once 81 against VLESS 109. Conclusion in the specification: six downloads "are still a quarter behind".

5.7 Phone on a far path, 2026-10-13 (§13.7). A phone with the `middle` profile measured 72 Mbit/s down and 26 up where VLESS through another node showed 132 and 54. After the fix of the node's unconfirmed-data limit: WataFast 92 and 76 against VLESS 118 and 118; four uploads 30 and 34 against 69 and 10. "Still behind VLESS" for downloads, uploads at half.

5.8 Why a download stayed behind VLESS, 2026-10-13 (§13.8). A phone showed 98, 102 and 90 Mbit/s against 189 for VLESS. Four causes were found in turn: the OS receive-buffer cap (146 Mbit/s at 250 ms); the HTTP/2 window (about 190); the session's own windows (one download 145–191 against VLESS 114–143 after the fix); and the node's kernel congestion control overestimating a rate, causing about 5,600 retransmitted packets per carrier in 12 s against about 300 for VLESS. The fix, an even pace in the program, gave two downloads 232 and 246 against VLESS 244 and 223, and eight downloads 263 and 156 against 239 and 163. In a later round the rule gave 218 for both against VLESS 241 and 229. Unresolved: about one session in six starts on a slow path and crawls 3–8 s; once in about eighty rounds a download crawled for seven seconds for a reason "not found".

5.9 A game beside transfers, 2026-10-13 (§13.9). With a quiet carrier for datagrams, the median question-to-answer time beside four downloads went from 299–392 ms to 248–288 ms; no datagram was lost. A phone, a real game and Russia: not measured.

5.10 A fast line, 2026-10-13/14 (§13.10). The least favourable result. A tester on a home line of about 600 Mbit/s with an iPhone measured 264, 280, 393 Mbit/s down and 148, 189, 208 up with WataFast, against 506, 553, 538 down and 400, 454, 485 up with VLESS. The phone was not the limit (28 of 50 MB). A bench (which "knows nothing of a congestion control") showed an upload alone rising from 268 to 530 Mbit/s at 120 ms after the send-side change. Phone speed after the changes: not measured.

5.11 Summary. Under imitated address freezing, port death, resets, a curtain on one of two entries and packet loss, a multipath session completed pages where single-connection VLESS did not. WataFast's raw throughput on clean or fast lines has repeatedly been below VLESS; causes were traced and partly fixed and gaps remain. Behaviour on real Russian access networks is not shown in this edition.

---

6. Comparison

Only what the specification states; "not measured" means the sources contain no measurement or statement.

Property	WataFast	VLESS/REALITY	XHTTP	Hysteria2
Multipath (one session, several connections)	Yes, up to four carriers (§8)	No single-session multipath; XHTTP "xmux" opens several connections to the same server	No	No

Property	WataFast	VLESS/REALITY	XHTTP	Hysteria2
Imitated-censor tests	Yes, nine imitations (§13.5)	Yes, same imitations	Not measured	Not measured
TLS fingerprint	Recorded Chrome 153 hello, equality-tested; no mobile recording yet (§7.1.1)	Not measured	Not measured	Not measured
UDP	Yes over a TCP carrier; UDP towards the HTTPS port dropped; no UDP carrier yet (§6.1, §7.4)	Not measured	Not measured	Not measured
Revocation while node runs	File-based, 30 s reload; delivery manual (§10.3)	Not measured	Not measured	Not measured
Shutdown (whitelist) mode	No; detection and relay readiness only (§11)	Not measured	Not measured	Not measured
Hybrid post-quantum key agreement	X25519 + ML-KEM-768 (§5)	Not measured	Not measured	Not measured
Real Russian access-network data	None in this edition (§7.1, §13)	Not measured	Not measured	Not measured

The specification benchmarks only VLESS. The multipath row states publicly documented features of the other protocols; every other cell outside the WataFast column is "not measured" because the sources contain no measurement of it. A later revision will fill those cells from independent sources.

---

7. Deployment status and open questions

7.1 Status

- The specification is a proprietary draft (v2, draft 0.7, 2026-10-13); wire version 2 is incompatible with version 1.
- Built and tested in the repository: handshake, record layer, `h2`, relay and plain carriers, multipath, curtain and short carriers, entries and sensors, UDP, node policy, live revocation, memory accounting,

browser look, client libraries.

- The specification snapshot (mid-October 2026) describes desktop integration as macOS test builds (§12.1), the phone library as "not yet inside an app" (§12.2), and the protocol as "not for users yet" (§7.1).
- Update, November 2026 (stated by the organization): the client is now shipped inside the released applications for Windows, macOS, Linux, Android, iPhone, iPad and Apple TV, and production nodes in several countries serve paying users over the protocol alongside the existing VLESS fleet. Field measurements from Russian access networks are being collected by the device-as-sensor mechanism (§8.6); no aggregated figures are published in this edition.
- Not built (§3, §8.7): backend accounting batches, a pulled revocation feed, the signed manifest, a datagram carrier, entry rotation.
- No claim is made about adoption outside the organization's own products, or about users or revenue.

7.2 Open questions (§14)

Fixed pace against the found-rate rule; the phone window budget (16 MiB instead of 8 gave more speed at unknown memory cost); the rare slow-start crawl; a carrier that begins on a slow path (two cures unbuilt because they change early wire behaviour); one stream over several carriers under per-stream order (needs a wire change); upload over long lossy paths (needs a datagram carrier); the privacy policy for device reports; the node memory limit on a real Linux node; TLS look on phones and session resumption; whether the site should imitate a common web server; entry rotation.

---

8. Statement of contribution

- **Supported by the sources:** a complete protocol specification and tested implementation combining the six ideas above in one codebase; reproducible measurement procedures with negative findings recorded; an honest account of limits.
- **Not claimed in this edition:** that each idea is unprecedented (a prior-art survey is planned as a separate document); third-party adoption outside the organization's own products; external citation or recognition; aggregated performance figures from real Russian networks. Independent review of the design and of the measurement procedures is welcome at the address below.

Watafast LLC (Wyoming, USA) | Author: evilork | hello@wata.fast | Published at https://proxysvpn.com/watafast-whitepaper.html (PDF: https://proxysvpn.com/watafast-whitepaper.pdf) | Press: https://proxysvpn.com/en/press

© 2026 Watafast LLC · Wyoming, USA · This is the public edition of the WataFast technical whitepaper. The full specification is not published.