

Forum:[Hack Forums](#)

Thread: [Open Image Network - Hybrid-Decentralized Image Hosting | 3 NIST PQC Standards](#)

Vinavil - March 7, 2026, 1:46 pm

????????????????????????????????????

OPEN IMAGE NETWORK

Post-Quantum Encrypted Image Hosting

????????????????????????????????????

A project by Vinavil

What Is This?

Most image hosts have full access to everything you upload. Imgur, Discord CDN, whatever. Your files are accessible on their servers, they can scan them, hand them over on request, or lose them in a breach. You're trusting a third party with your data because there was never a real alternative.

OIN is the alternative. Upload an image, it gets encrypted in your browser before it ever touches the server. The encrypted blob gets chunked, wrapped in post-quantum cryptography, and distributed across storage nodes. The gateway that coordinates everything never sees plaintext. The nodes that store your data can't decrypt it even if they wanted to. The decryption key lives in the URL fragment, which browsers never send to servers. Click a link, your browser pulls the ciphertext, decrypts locally, displays the image. Nobody in between sees anything.

No accounts. No tracking. No ads.

Why Should You Care?

Two words: harvest now, decrypt later. Intelligence agencies and state actors are stockpiling encrypted traffic today, waiting for quantum computers capable of breaking classical cryptography. RSA, ECDH, the stuff protecting most of the internet right now, all of it becomes worthless the day a sufficiently large quantum computer exists. Most estimates say 10-15 years. Some say sooner.

OIN doesn't wait for that to be a problem. Every manifest is encrypted with AES-256-GCM using keys exchanged through ML-KEM-1024, which is NIST's post-quantum key encapsulation standard. Every envelope is dual-signed with ML-DSA-87 (lattice-based) and SLH-DSA-SHAKE-256s (hash-based). If lattice math breaks tomorrow, the hash signatures still hold. If both break, you still have the browser-side AES-256-GCM layer that has 128-bit post-Grover security with no asymmetric key exchange to attack with Shor's algorithm.

Three layers of encryption. Two independent signature schemes. Zero trust required.

How It Actually Works

No hand-waving. Here's the real pipeline, every step.

? Browser-Side Encryption (E2E Layer)

Before your image ever leaves the browser, WebCrypto generates a 32-byte random seed and 12-byte IV. HKDF-SHA256 derives an AES-256-GCM key from the seed. The image gets encrypted. An HMAC-SHA256 of the ciphertext is computed for integrity verification. The encrypted blob is what gets uploaded. The gateway receives ciphertext that it cannot distinguish from random noise.

? Gateway Processing (PQC Layer)

The gateway takes the ciphertext and chunks it using Reed-Solomon erasure coding, creating data shards and parity shards. Each chunk is encrypted with a per-image AES-256-GCM key. The manifest containing chunk locations and the image key is then wrapped in a PQC envelope: ML-KEM-1024 establishes a quantum-safe shared secret, HKDF-SHA3-256 combines it with classical randomness to derive a hybrid key, AES-256-GCM encrypts the manifest, and both ML-DSA-87 and SLH-DSA-SHAKE-256s sign the envelope. The classical key portion goes into the share link.

? Node Storage (Zero-Knowledge)

Encrypted chunks get distributed to storage nodes. Nodes store opaque blobs. The node codebase literally imports zero cryptographic decryption functions. It can't see your images. It doesn't know what it's storing. It just holds bytes and serves them back when asked. If a node goes rogue or gets compromised, the attacker gets encrypted garbage.

? Viewer Decryption

Someone opens your share link. The browser parses the URL fragment (never sent to the server), fetches the encrypted data, verifies the HMAC to detect tampering, and decrypts with AES-256-GCM. All client-side. The server delivered ciphertext, the browser produced the image. No plugin, no extension, no app required.

The Crypto Stack

Everything here is NIST standardized. No homebrew, no experimental schemes.

ML-KEM-1024 (FIPS 203)

Lattice-based key encapsulation. Security Level 5. Generates quantum-safe shared secrets that get mixed with classical randomness via HKDF-SHA3-256 to produce hybrid encryption keys. Both the quantum and classical components must be broken simultaneously.

ML-DSA-87 (FIPS 204)

Lattice-based digital signatures. Security Level 5. 4,627-byte signatures. Fast verification. Signs every PQC envelope so you know the manifest hasn't been tampered with since encryption.

SLH-DSA-SHAKE-256s (FIPS 205)

Hash-based digital signatures. Security Level 5. 29,792-byte signatures. Zero algebraic structure to attack. If every lattice assumption falls apart tomorrow, this signature scheme still stands because it's built on hash functions, not math problems. The conservative backup.

AES-256-GCM

The workhorse. Browser-side E2E encryption, per-chunk encryption, and manifest encryption inside the PQC envelope. Hardware-accelerated via AES-NI. 128-bit security post-Grover.

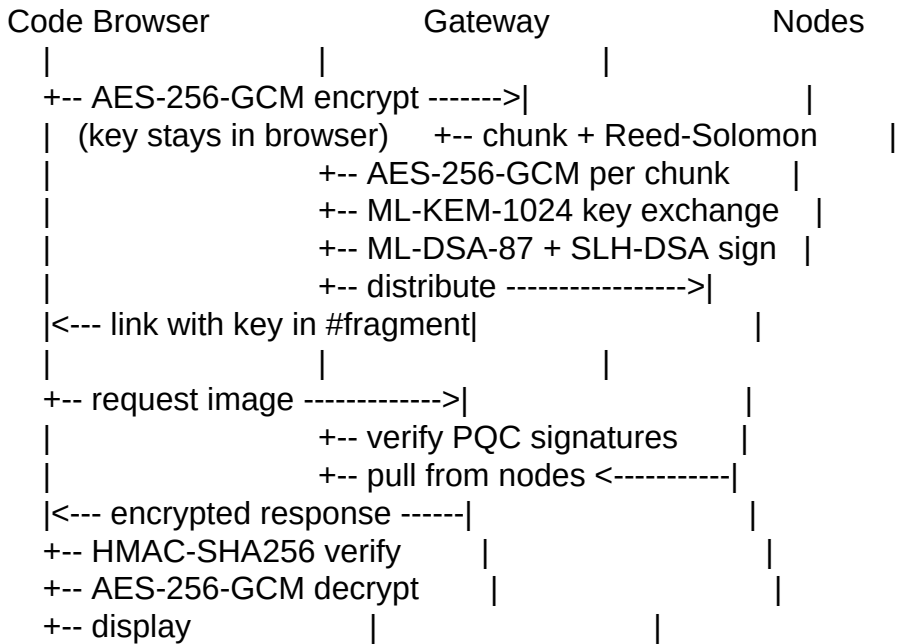
Architecture

I'm not going to pretend this is fully decentralized. It's not. Here's what it actually is.

The gateway is a central coordinator. It receives uploads, encrypts and chunks data, distributes to nodes, and reassembles on download. It does not store data permanently. After nodes confirm replication, the gateway evicts the image from RAM. But it's still a single point of coordination. If the gateway goes down, share links stop working until it comes back.

Nodes are the distributed part. They're independent, can run anywhere, and store only encrypted blobs. The pull architecture means nodes poll the gateway for work, so no port forwarding is needed. NAT-friendly. Works behind any firewall. Multiple nodes provide redundancy through Reed-Solomon erasure coding, so images survive partial node failure.

Call it hybrid-decentralized. The storage is distributed, the coordination is centralized. That's the honest picture.



Running A Node

Minimal barrier to entry. Download a binary, run it. Done.

? Windows

Code oin-node-windows-x64.exe

? Linux / macOS

Code chmod +x oin-node-*

./oin-node-linux-x64

? Build From Source

Code git clone <https://github.com/VinavilDev/OpenImageNetwork.git>

cd OpenImageNetwork

cargo build --release -p oin-node

./target/release/oin-node

No configuration files, no environment setup, no accounts. The node auto-detects all available disks, allocates 3.5% of total space, connects to the gateway, and starts accepting encrypted chunks. Cross-platform: Windows, macOS (ARM + Intel), Linux. Single binary, no dependencies.

Here's what it looks like when it starts:

Code 2026-03-07T14:22:10Z INFO location: auto-detected
2026-03-07T14:22:10Z INFO detected 1 disk(s):
2026-03-07T14:22:10Z INFO /dev/sda1 - total: 500 GB, available: 420 GB
2026-03-07T14:22:10Z INFO storage plan: 17.5 GB across 1 disk(s) (3.5%)
2026-03-07T14:22:10Z INFO OIN Node [a1b2c3d4e5f6] on :9090
2026-03-07T14:22:38Z INFO sync 855622384066
2026-03-07T14:22:40Z INFO synced 855622384066 - 6/6 chunks
That's it. Node is live, syncing encrypted chunks, contributing storage to the network. Every chunk it stores is opaque ciphertext it can never decrypt.

The Tech Stack
Language: 100% Rust. Memory-safe with 2 unsafe blocks (FFI calls for disk space queries, both bounds-checked).
Crypto: fips203, fips204, fips205 crates for PQC. aes-gcm for symmetric encryption. hkdf + sha3 for key derivation.
Erasure Coding: Reed-Solomon via reed-solomon-erasure crate. Data + parity shards survive partial node loss.
Web Framework: Axum with Tokio async runtime. Gateway handles uploads, downloads, node coordination, admin panel.
Browser Crypto: WebCrypto API. AES-256-GCM, HKDF-SHA256, HMAC-SHA256. No JavaScript crypto libraries, all native browser APIs.

Links

Upload an image: <https://oin.vinavildev.com>
Foundation page: <https://oinfoundation.vinavildev.com>
Source code: <https://github.com/VinavilDev/OpenImageNetwork>
Download binaries: <https://github.com/VinavilDev/OpenImageNetwork/releases>

Closing Notes

I built this because I was tired of trusting image hosts with my data. Every platform says they care about privacy while scanning every upload and complying with every request that shows up. The fundamental problem is architectural: if the server can see your data, it will eventually be seen by someone you didn't intend.

OIN flips that. The server is blind by design. The gateway processes ciphertext it can't read. Nodes store blobs they can't open. The only entity that ever sees the plaintext image is the person holding the share link, in their browser, decrypted locally. Post-quantum crypto means even the harvest-now-decrypt-later crowd gets nothing useful.

It's hybrid-decentralized, not fully decentralized. The gateway is a single point of coordination. I'm honest about that. But the data layer is distributed, the crypto is real, and every claim in this post maps to actual code you can audit. MIT licensed. Pure Rust. Check the source.

If you want to support the network, download the node binary and run it. Takes 30 seconds. You'll be storing encrypted chunks you can never read, helping make the network more resilient for everyone.

????????????????????????????????????

Own Your Images. Trust No One.
????????????????????????????????

Omniscient - March 7, 2026, 1:52 pm
Could we build a MyBB plugin so we can insert images into posts.

https://oin.vinavildev.com/c92725e5761f#E2E1.sg4Pej-RaAGuFu3FKZrBTgjfM62JUPiEFxQstiyUm8.n16bm03whzqcFAW5.3ITc9NExDAiGT7X0JUu_7g.TqVam4qCWARPKOMkeFXUsKR6PzbsFfVYjv.fu8btAg8o

I did upload an image as a test. Seems to word but I don't think we could embed these.

btw, I own Image-Host.com and UploadImages.org ; we should talk.

JoshAllen - March 7, 2026, 1:54 pm
Love this and will be using it in the future! Great work [Vinavil](#)

Vinavil - March 7, 2026, 1:55 pm
(Mar 7, 2026 02:52 PM)Omniscient Wrote: [\[CITE\]](#) Could we build a MyBB plugin so we can insert images into posts.

https://oin.vinavildev.com/c92725e5761f#E2E1.sg4Pej-RaAGuFu3FKZrBTgjfM62JUPiEFxQstiyUm8.n16bm03whzqcFAW5.3ITc9NExDAiGT7X0JUu_7g.TqVam4qCWARPKOMkeFXUsKR6PzbsFfVYjv.fu8btAg8o

I did upload an image as a test. Seems to word but I don't think we could embed these.

btw, I own Image-Host.com and UploadImages.org ; we should talk.
100% we can, hit me in convo i'll be away for a couple of hours!

Omniscient - March 7, 2026, 2:23 pm
IMHO, it needs embed links. It's slow right now too. Unsure if that's the encryption or the host you use.

Vinavil - March 7, 2026, 6:44 pm
(Mar 7, 2026 03:23 PM)Omniscient Wrote: [\[CITE\]](#) IMHO, it needs embed links. It's slow right now too. Unsure if that's the encryption or the host you use.
the gateway is running on a budget VPS right now so that's a big part of it. but the architecture also adds latency by design. when you open an image that's not cached, the gateway has to request it from the storage nodes, wait for the manifest delivery, wait for all the chunks, reassemble them, then your browser does the E2E decryption on top. that's multiple network round-trips before you see a single pixel. AES-256-GCM itself runs at hardware speed via AES-NI, the actual crypto takes milliseconds, it's the back-and-forth that kills it.

already made some improvements on this, the gateway now keeps recently uploaded images in a RAM cache instead of evicting them immediately after node replication. so repeat views are instant. there's also a disk cache layer that survives restarts. the main bottleneck is first-time cold loads on images that have been fully offloaded to nodes.

plans to improve it further: better infrastructure obviously, but also more nodes means geographically closer storage which cuts the round-trip time. the more people running nodes, the

faster everything gets for everyone.

for embed links, that's a conscious tradeoff. E2E encryption means the gateway genuinely cannot see the image, the decryption key lives in the URL fragment, which browsers never send to servers. so there's no way for it to generate a preview or serve raw bytes. you either get a server that's blind to your content, or one that can render [img] embeds. can't have both without compromising the model.

implementing a browser extension that'll scan pages for OIN links and decrypt them inline could be a cool idea. same security, decryption stays in the browser. should bridge that gap without breaking the E2E guarantee.

(Mar 7, 2026 02:54 PM)JoshAllen Wrote: [\[CITE\]](#) Love this and will be using it in the future! Great work [Vinavil](#)
Tysm <3

Puppetmaster - March 8, 2026, 1:22 pm

I think for people who will be running nodes for years and years to come should be rewarded, otherwise no one will bother running them. Also why would anyone give away 3.5% of total storage? Mine was coming around 4TB so giving that away without any reward makes 0 sense.

Omniscient - March 8, 2026, 3:46 pm

<https://uploadimages.org/>

Going to try to install [Vinavil](#) script onto a VPS I run and integrate with HF.

Vinavil - March 8, 2026, 4:53 pm

(Mar 8, 2026 01:22 PM)Puppetmaster Wrote: [\[CITE\]](#) I think for people who will be running nodes for years and years to come should be rewarded, otherwise no one will bother running them. Also why would anyone give away 3.5% of total storage? Mine was coming around 4TB so giving that away without any reward makes 0 sense.

Anonimty, Freedom, Not letting companies sell your data, there is a whole lot of reasons, if you of course belive in the project's vision!

Puppetmaster - March 8, 2026, 11:00 pm

(Mar 8, 2026 04:53 PM)Vinavil Wrote: [\[CITE\]](#) Anonimty, Freedom, Not letting companies sell your data, there is a whole lot of reasons, if you of course belive in the project's vision!

I do but i still would suggest rewarding the users and if you don't do it someone else would now that it's OPEN SOURCE

Vinavil - March 9, 2026, 6:02 am

(Mar 8, 2026 11:00 PM)Puppetmaster Wrote: [\[CITE\]](#) I do but i still would suggest rewarding the users and if you don't do it someone else would now that it's OPEN SOURCE

The node is open source, the whole gateway system behind it is not, so i have no worries about "someone else would now", don't need to reward nobody, the service it self is the final reward!

Puppetmaster - March 9, 2026, 12:05 pm

(Mar 9, 2026 06:02 AM)Vinavil Wrote: [\[CITE\]](#) The node is open source, the whole gateway system behind it is not, so i have no worries about "someone else would now", don't need to reward nobody, the service it self is the final reward!

"the whole gateway system behind it is not"

"the service it self is the final reward!"

Very fishy.....you could be using those nodes to make money by utilizing their space
Now that HDD is very costly after this AI bubble
good luck
my last question, you don't have to answer if you don't want to:-
Where do the shared secret come from?
which key exchange algo is used?
how are keys authenticated?
are you using forward secrecy?

Vinavil - March 9, 2026, 12:55 pm

(Mar 9, 2026 12:05 PM)Puppetmaster Wrote: [\[CITE\]](#) "the whole gateway system behind it is not"
"the service it self is the final reward!"

Very fishy.....you could be using those nodes to make money by utilizing their space
Now that HDD is very costly after this AI bubble
good luck
my last question, you don't have to answer if you don't want to:-
Where do the shared secret come from?
which key exchange algo is used?
how are keys authenticated?
are you using forward secrecy?
the nodes store encrypted shards, not full files, a single node has a fragment that is completely meaningless without the full key set which never leaves the client side
key exchange is X25519, ephemeral per session, forward secrecy is implemented so even if a long term key is ever compromised past sessions stay clean, node registration uses challenge-response so a node that can't prove key ownership doesn't join the pool at all
on the gateway being closed that's intentional, if the full stack was open someone forks it, strips the privacy layer, deploys fake OIN nodes and now you have a harvesting network wearing OIN's face. the open node client lets you audit your own side which is the part that actually affects you and the "making money off nodes" take is just embarrassing, go ahead and monetize AES encrypted fragments with zero metadata, zero reconstruction capability, zero context, let me know how that business plan works out LOL.

got carried away, shared secrets are derived client-side using X25519 ECDH, the exchange happens before anything touches a node, nodes/gateway never see plaintext ever

Puppetmaster - March 9, 2026, 1:38 pm

(Mar 9, 2026 12:55 PM)Vinavil Wrote: [\[CITE\]](#) the nodes store encrypted shards, not full files, a single node has a fragment that is completely meaningless without the full key set which never leaves the client side
key exchange is X25519, ephemeral per session, forward secrecy is implemented so even if a long term key is ever compromised past sessions stay clean, node registration uses challenge-response so a node that can't prove key ownership doesn't join the pool at all
on the gateway being closed that's intentional, if the full stack was open someone forks it, strips the privacy layer, deploys fake OIN nodes and now you have a harvesting network wearing OIN's face. the open node client lets you audit your own side which is the part that actually affects you and the "making money off nodes" take is just embarrassing, go ahead and monetize AES encrypted fragments with zero metadata, zero reconstruction capability, zero context, let me know how that business plan works out LOL.

got carried away, shared secrets are derived client-side using X25519 ECDH, the exchange happens before anything touches a node, nodes/gateway never see plaintext ever
Its same as Storj, IPFS and Sia you will have to deal with node churn. If ur file is split into 20 shards

and 5 nodes disappear than it's impossible to reconstruct.

So it's either token rewards like Sia coin

or

paid storage contract like Storj else i see your pussy project going to DEATH

The hardest isn't x25519 or AES that's well understood it's the

fucking data availability

fucking node incentives

fucking metadata privacy and fucking repairing the lost shards automatically

Vinavil - March 9, 2026, 7:48 pm

node churn is a real problem.

erasure coding with $k=12$ $n=20$ means you can lose 8 nodes out of 20 and still reconstruct the full file, you don't lose the file you lose nothing, the whole point of reed solomon is that losing nodes is expected and accounted for, this is day one distributed systems stuff

token rewards like Siacoin introduced sybil farming and speculative garbage traffic, paid contracts like Storj add friction and centralize the economic layer, reputation scoring on provable uptime has no attack surface and no speculation, cleaner model

metadata privacy never leaves the client, only the entity holding the keychain knows what resolves to what, nothing about topology ever touches a node or gateway unencrypted

repair is automatic, node misses heartbeat threshold it gets marked dead, repair job fires instantly, missing shards redistribute to healthy nodes, nodes stay blind the entire time, they see a fragment they cannot read and nothing else, zero context, zero metadata, zero reconstruction capability without the client keychain

the four problems you listed are exactly what the architecture was designed around from day one, reed solomon over a galois field $GF(2^8)$ with configurable k and n parameters, automatic repair pipeline, encrypted topology, reputation based node selection, if you actually understood what you were typing you would know these are textbook solutions not open questions, come back when you can explain your own buzzwords XD

5chn4pp - March 10, 2026, 9:11 am

(Mar 9, 2026 01:38 PM)Puppetmaster Wrote: [\[CITE\]](#) _(?)_

If you're gonna pick an argument about a project on here, you should at least test out your arguments in practice.

Did you even bother to look at the source code? For example the `hybrid_encrypt()` function in `ppc.rs`?

Puppetmaster - March 12, 2026, 11:19 am

the challenge still remains which is bandwidth overhead during repair. and your network level metadata can still leak ip patterns, timing analysis and traffic correlation

Vinavil - March 12, 2026, 11:27 am

(Mar 12, 2026 11:19 AM)Puppetmaster Wrote: [\[CITE\]](#) the challenge still remains which is bandwidth overhead during repair. and your network level metadata can still leak ip patterns, timing analysis and traffic correlation

Ip patterns of? Public nodes LOL, man please stop embarrassing yourself.

5chn4pp - March 12, 2026, 1:09 pm

(Mar 12, 2026 11:19 AM)Puppetmaster Wrote: [\[CITE\]](#) the challenge still remains which is bandwidth overhead during repair. and your network level metadata can still leak ip patterns, timing analysis and traffic correlation

Looking forward to seeing your PoC and Analysis of that bandwidth overhead problem during repair.

Or did you skip my previous advice about testing out your arguments in practice?

Some people are immune to good advice.

Hexicidal - March 16, 2026, 8:15 pm

(Mar 12, 2026 11:19 AM)Puppetmaster Wrote: [\[CITE\]](#) the challenge still remains which is bandwidth overhead during repair. and your network level metadata can still leak ip patterns, timing analysis and traffic correlation

He's very much lowkey right. The post qb bit is overkill for potentially an architectural flaw. Gateway still sees everything and node trust is the same issue.

Plus disk allocation just means there's a higher level privilege occurring anyway, unless something else...

--

If he was less aggressive, I think you would see his point. He's just concerned about metadata and the visible weaknesses of this, nothing more.

Vinavil - March 17, 2026, 12:40 am

(Mar 16, 2026 08:15 PM)Hexicidal Wrote: [\[CITE\]](#) He's very much lowkey right. The post qb bit is overkill for potentially an architectural flaw. Gateway still sees everything and node trust is the same issue.

Plus disk allocation just means there's a higher level privilege occurring anyway, unless something else...

--

If he was less aggressive, I think you would see his point. He's just concerned about metadata and the visible weaknesses of this, nothing more.

Two encryption layers, not one. Browser encrypts the image with AES-256-GCM before upload (E2E layer). Server then wraps that ciphertext in a PQC envelope (server layer). On download, the server unwraps the PQC layer and returns the E2E ciphertext, which only the browser can decrypt. Gateway never sees plaintext in E2E mode, it handles encrypted blobs both in and out.

The share link fragment has two parts: a gateway key (sent to the server as a query param to unwrap PQC) and the E2E seed/iv/hmac (never leaves the browser). Server can strip PQC but can't touch the E2E layer underneath.

PQC is future-proofing on the server layer, not a substitute for the E2E layer. Fair to call it overkill for an image host today.

The metadata point is valid. Server sees IPs, timestamps, file sizes, filenames. Content is encrypted, metadata isn't. That's a known tradeoff. Full metadata protection would need onion routing which is a different scope.

Disk allocation just writes encrypted chunks as regular files in user-owned directories. No elevated privileges needed.

I hope this is finally clear.

SUI-3183372 - March 22, 2026, 10:28 am

Curious of your thoughts around abuse of a service like this. Is your central coordinator ever at risk if it's sharding or reconstructing images that would be considered illegal? Is the reconstruction of an illegal image for somebody to use considering distribution? Is this something you've thought about before and have already baked into precautions for yourself? I know generally the host of a site can't be held responsible for what its users are doing, but I can imagine a service like this become a hot bed for abuse.

Vinavil - March 23, 2026, 2:27 am

(Mar 22, 2026 10:28 AM)SUI-3183372 Wrote: [\[CITE\]](#) Curious of your thoughts around abuse of a service like this. Is your central coordinator ever at risk if it's sharding or reconstructing images that would be considered illegal? Is the reconstruction of an illegal image for somebody to use considering distribution? Is this something you've thought about before and have already baked into precautions for yourself? I know generally the host of a site can't be held responsible for what its users are doing, but I can imagine a service like this become a hot bed for abuse.

Yeah I've thought about this from day one. The short answer is the server never sees the content. It stores and serves encrypted blobs it cannot decrypt. Decryption only happens in the viewer's browser. So the server never reconstructs or processes the actual image at any point.

We still have the standard abuse toolkit though. Report system with per-IP dedup, rate limited uploads, admin can delete any image by ID, and expiring links.

Legally the precedent is pretty clear. Section 230 in the US and E-Commerce Directive in the EU both protect platforms as long as they respond to takedown requests. Since we architecturally cannot scan content (it's encrypted), our obligation is to act on reports and legal requests which the system supports.

The key distinction is we're not choosing not to look at content. We can't look. That's by design and it matters legally.

Could any upload service attract abuse? Sure. But the report and takedown pipeline is there and the zero knowledge design means we're never storing plaintext content on our end.

Popular - March 27, 2026, 8:27 am

Wow, very cool! I love this. Good work

Vinavil - March 27, 2026, 9:05 am

(Mar 27, 2026 08:27 AM)Popular Wrote: [\[CITE\]](#) Wow, very cool! I love this. Good work
Thanks!