The End of IPFS: Centralized Pinning Single Points of Failure, Cryptographic Hash Backdoors, DDoS Exploitation, and Gossip-Based Distributed Node Eradication

A Comprehensive Multi-Vector Threat Analysis of the InterPlanetary File System

יהוה
Kaoru Aguilera Katayama

February 8, 2026

Abstract

The InterPlanetary File System (IPFS) has been extensively promoted as a decentralized, censorship-resistant, and fault-tolerant storage protocol. This paper systematically dismantles these claims by demonstrating **four critical and compounding vulnerability classes**: (1) the structural dependency on centralized pinning services such as Pinata, Infura, and Web3.Storage, where compromising a single provider's dashboard or API effectively eliminates supposedly "immutable" content; (2) the futility of self-hosted pinning nodes as a mitigation strategy, given their susceptibility to targeted Distributed Denial-of-Service (DDoS) attacks capable of rendering them permanently unreachable; (3) the catastrophic implications of a cryptographic backdoor or collision discovery in SHA-256 or SHA-3 (Keccak), which would enable arbitrary content substitution while preserving valid Content Identifiers (CIDs), thereby destroying IPFS's fundamental integrity guarantees; and (4) the vulnerability of distributed pinning strategies to gossip-protocol-based reconnaissance attacks, wherein a state-level adversary (e.g., NSA, GCHQ, or equivalent) can enumerate all nodes hosting a target CID by compromising a single peer and leveraging protocol-level metadata propagation to systematically identify and neutralize every replica simultaneously. We formalize each attack vector with mathematical models, provide proof-of-concept algorithms, analyze the compounding effects of multi-vector attacks, and demonstrate that even the most sophisticated defense-in-depth strategies fail against a sufficiently resourced adversary. Our analysis conclusively establishes that IPFS, as deployed in practice, provides **no meaningful censorship resistance** and constitutes what we term *Decentralization Theater*—a system that employs the aesthetics and terminology of decentralization while maintaining the vulnerability profile of traditional centralized architectures, augmented by a dangerous false sense of security.

Keywords: IPFS, censorship resistance, pinning services, DDoS, SHA-256 backdoor, gossip protocol reconnaissance, state-level adversary, decentralization theater, Web3 security

1 Introduction

1.1 The Promise

The InterPlanetary File System [1], introduced by Juan Benet in 2014, proposes a peer-to-peer distributed file system built on content-addressing. Its foundational claims include:

C1 Immutability: Content is addressed by its cryptographic hash (CID); modification changes the identifier.

C2 Decentralization: No central server; any node can store and serve content.

C3 Censorship Resistance: No single entity can suppress content availability.

C4 Persistence: Content remains available while at least one node stores it.

C5 Integrity: Cryptographic hashing guarantees content has not been tampered with.

These properties have driven adoption across critical ecosystems: NFT metadata (OpenSea, Rarible), decentralized application frontends, DAO governance documents, and censorship-sensitive journalism archives.

1.2 The Gap Between Theory and Reality

This paper demonstrates that *every single claim above is practically false* under realistic threat models. We identify four compounding vulnerability classes that, when considered together, render IPFS's security guarantees effectively meaningless:

- **V1 — Centralized Pinning:** Single point of failure in pinning services.
- **V2 — DDoS Against Self-Hosted Nodes:** Mitigation through self-hosting is trivially defeated.
- **V3 — Cryptographic Hash Backdoors:** Content integrity collapses if SHA-256/SHA-3 is compromised.
- **V4 — Gossip-Based Distributed Eradication:** Distributed pinning is defeated by protocol-level reconnaissance.

1.3 Contributions

1. Formalization of four independent attack vectors against IPFS.
2. Proof that mitigating one vector exposes the system to others.
3. Demonstration that defense-in-depth fails against state-level adversaries.
4. Introduction of the *Decentralization Theater* framework.
5. Proposal of realistic resilience classification standards.

1.4 Responsible Disclosure

This paper is published for academic and awareness purposes. No zero-day vulnerabilities in specific software implementations are disclosed. All attack vectors described exploit *architectural* weaknesses, not implementation bugs.

2 Technical Background

2.1 IPFS Architecture

IPFS combines several technologies [1]:

- **Distributed Hash Table (DHT):** Kademlia-based peer discovery and content routing.
- **BitSwap:** Block exchange protocol inspired by BitTorrent.
- **Merkle DAG:** Directed acyclic graph for content verification.
- **CID (Content Identifier):** Typically SHA-256-based hash of content.

When a file is added to IPFS:

$$\text{File} \xrightarrow{\text{chunk}} \text{Blocks} \xrightarrow{\text{SHA-256}} \text{CID} \xrightarrow{\text{DHT}} \text{Provider Records}$$

2.2 The Persistence Problem: Garbage Collection

Critical Design Property

By default, IPFS nodes **do not permanently store** content they retrieve. Unpinned content is subject to garbage collection and will be **deleted**.

A node retains content only if:

1. The node originally added the content.
2. The content was explicitly pinned (`ipfs pin add`).
3. The content remains in temporary cache (subject to eviction).

This means IPFS, *by design*, does **not guarantee persistence**. Persistence is delegated to the user or, in practice, to pinning services.

2.3 Pinning Services: The Centralized Crutch

Table 1: Major IPFS Pinning Services

Service	Owner	Est. Share	Infra
Pinata	Pinata Inc.	40–50%	AWS
Infura	ConsenSys	20–25%	Multi-cloud
Web3.Storage	Protocol Labs	15–20%	Filecoin/CF
Filebase	Filebase Inc.	5–7%	Multi-cloud
Others	Various	5–10%	Various

2.4 Content Addressing and SHA-256

IPFS CIDs (v1) encode a multihash that typically wraps SHA-256:

$$\text{CID} = \langle \text{version}, \text{codec}, \text{multihash}(\text{SHA-256}(c)) \rangle$$

The security of content addressing is *entirely dependent* on the collision resistance and preimage resistance of the underlying hash function.

2.5 DHT and Provider Discovery

The Kademlia DHT stores provider records:

$$\text{DHT}[\text{CID}] = \{(\text{PeerID}_1, \text{Multiaddr}_1), \ldots, (\text{PeerID}_k, \text{Multiaddr}_k)\}$$

These records have a TTL (typically 24 hours) and must be refreshed by active providers. This mechanism is fundamental to Attack Vector V4.

3 Attack Vector V1: Centralized Pinning Compromise

3.1 Threat Model

Definition 3.1 (Exclusive Dependency)**.** *A content object c with identifier CID_c has* exclusive dependency *on pinning service S if and only if S is the sole active provider:*

$$\mathcal{P}(CID_c) = \{S\}$$

Definition 3.2 (Effective Censorship)**.** *Effective censorship of c occurs when:*

$$\forall\, n \in \mathcal{N}_{IPFS} : \ \texttt{resolve}(CID_c, n) = \varnothing$$

Theorem 3.3 (Centralized Pinning Censorship)**.** *If content c has exclusive dependency on service S, then compromising S is sufficient for effective censorship of c.*

Proof. If $\mathcal{P}(\text{CID}_c) = \{S\}$ and S is compromised (content deleted), then $\mathcal{P}(\text{CID}_c) = \varnothing$. With no providers, no node can resolve CID_c, hence $\texttt{resolve}(\text{CID}_c, n) = \varnothing$ for all n. Additionally, DHT provider records expire after TTL (typically $\leq$ 24h) without refresh. □

3.2 Attack Surface

Pinata exposes the following attack surface:

```python
import requests, time

API = "https://api.pinata.cloud"
HDR = {"Authorization": f"Bearer {STOLEN_JWT}"}

def list_all_pins():
    pins, offset = [], 0
    while True:
        r = requests.get(f"{API}/data/pinList",
            headers=HDR,
            params={"status":"pinned",
                    "pageLimit":1000,
                    "pageOffset":offset})
        batch = r.json()["rows"]
        pins.extend(batch)
        if len(batch) < 1000: break
        offset += 1000
    return pins

def nuclear_unpin():
    for pin in list_all_pins():
        cid = pin["ipfs_pin_hash"]
        requests.delete(
            f"{API}/pinning/unpin/{cid}",
            headers=HDR)
        print(f"[UNPINNED] {cid}")

# Compromise scenario:
# T+0:   Attacker gains API access
# T+5m:  All CIDs unpinned
# T+24h: DHT records expire
```

```
# T+48h: Gateway caches expire
# T+72h: Content IRRECOVERABLE
```

Listing 1: Pinata Mass Unpin via API

3.3 Compromise Vectors

1. **Credential Theft:** Phishing, credential stuffing, API key leakage in public repositories.
2. **Cloud Infrastructure Compromise:** AWS IAM misconfigurations, SSRF to metadata endpoint (`169.254.169.254`), S3 bucket misconfigurations.
3. **Supply Chain Attack:** Malicious dependency injection (cf. `event-stream` incident, 2018).
4. **Legal Coercion:** Court orders, National Security Letters, FISA warrants (no disclosure permitted).
5. **Insider Threat:** Malicious or compromised employee with administrative access.
6. **BGP Hijacking:** Traffic redirection to intercept API credentials in transit.

3.4 Impact Assessment

Table 2: Impact of Pinata Compromise by Sector

Sector	IPFS Dependency	Severity
NFT Metadata	∼80%	CRITICAL
dApp Frontends	∼60%	CRITICAL
DAO Governance	∼40%	HIGH
Journalism	∼25%	HIGH
DeFi Docs	∼30%	HIGH

4 Attack Vector V2: DDoS Against Self-Hosted Nodes

4.1 The Naïve Mitigation

A common counterargument to V1 is: *"Just run your own IPFS node."* This section demonstrates why self-hosting is an insufficient mitigation.

4.2 DDoS Threat Model

Definition 4.1 (DDoS Censorship)**.** *Let n_s be a self-hosted IPFS node with bandwidth capacity B_s. A DDoS attack achieves censorship when the attack traffic T_a satisfies:*

$$T_a > B_s + B_{mitigation}$$

where $B_{mitigation}$ is any DDoS mitigation capacity available to n_s.

Theorem 4.2 (Self-Hosted Node Futility)**.** *For any self-hosted IPFS node n_s with finite resources, there exists an attacker with sufficient resources to render n_s permanently unreachable.*

Proof. Let B_s be the bandwidth capacity of n_s and B_{mit} be mitigation capacity. A state-level adversary controls botnet capacity B_a where:

$$B_a \gg B_s + B_{\text{mit}}$$

Known botnet capacities exceed 1 Tbps (Mirai: 1.2 Tbps in 2016; Meris: 21.8 Mpps in 2021). For a typical self-hosted node with $B_s \leq 1$ Gbps and $B_{\text{mit}} \leq 10$ Gbps (commercial DDoS protection), the attack trivially succeeds.

Additionally, the IPFS node must expose its multiaddress in the DHT, making it *discoverable by design*:

$$\text{DHT}[\text{CID}] \ni (\text{PeerID}_{n_s}, \text{Multiaddr}_{n_s})$$

The target's IP address is **necessarily public** for IPFS to function, eliminating any possibility of concealment. □

4.3 IPFS Protocol-Specific DDoS Amplification

Beyond volumetric flooding, IPFS exposes protocol-specific attack surfaces:

Attack Vector 4.1 (DHT Poisoning + DDoS Combination)**.** *The attacker:*

1. *Queries the DHT for CID_{target} to discover provider n_s's multiaddress.*
2. *Launches volumetric DDoS against n_s's IP.*
3. *Simultaneously poisons the DHT with false provider records pointing to non-existent nodes.*
4. *After DHT record TTL expires ($\leq$ 24h), legitimate provider records are replaced with poisoned entries.*
5. *Even if DDoS stops, content is unresolvable.*

Attack Vector 4.2 (BitSwap Resource Exhaustion)**.** *The attacker floods the target node with malformed or excessive BitSwap requests, exhausting:*

- *File descriptor limits*
- *Memory (connection tracking)*
- *CPU (hash verification of spurious blocks)*
- *Bandwidth (legitimate requests crowded out)*

This is a protocol-level *DDoS that bypasses volumetric mitigation.*

4.4 Cost Analysis

Table 3: DDoS Cost vs. Self-Hosted IPFS Node Cost

Resource	**Defender**	**Attacker**
Server (monthly)	\$50–500	—
Bandwidth (1 Gbps)	\$100–1000	—
DDoS Protection	\$200–5000	—
Booter/Stresser (1 day)	—	\$30–100
Botnet rental (1 week)	—	\$200–1000
State-level (unlimited)	—	\$0 (taxpayer)
Asymmetry ratio	**10–100×**	

The **attacker–defender asymmetry** is overwhelming. The defender must pay continuously; the attacker needs only sustain the attack until DHT records expire.

4.5 Cloudflare / CDN Mitigation Paradox

The Centralization Paradox

If a self-hosted node uses Cloudflare or similar CDN for DDoS protection, the system now depends on Cloudflare—introducing a new centralized single point of failure and undermining the very decentralization that self-hosting was meant to achieve.

$$\text{Self-host} + \text{DDoS Protection} = \text{Centralized Dependency (again)}$$

5 Attack Vector V3: Cryptographic Hash Backdoor

5.1 Overview

This section addresses the most fundamental vulnerability: the possibility that the hash functions underlying IPFS (SHA-256 and/or SHA-3/Keccak) contain undiscovered backdoors, structural weaknesses, or that a state-level adversary has developed the capability to produce collisions or second preimages.

5.2 Dependency on Hash Function Security

Theorem 5.1 (Hash Dependency)**.** *The integrity guarantee of IPFS is* exactly equivalent *to the collision resistance of its underlying hash function:*

$$\mathit{Integrity}_{IPFS} \Leftrightarrow \mathit{CollisionResistance}(\mathcal{H})$$

where $\mathcal{H} \in \{\mathit{SHA\text{-}256}, \mathit{SHA\text{-}3}\}$*. If an adversary can find* $c' \neq c$ *such that* $\mathcal{H}(c') = \mathcal{H}(c)$*, then content substitution is undetectable.*

Proof. In IPFS, content is retrieved by CID, which encodes $\mathcal{H}(c)$. A requesting node verifies integrity by checking $\mathcal{H}(\text{received}) \stackrel{?}{=} \text{CID}$. If $\mathcal{H}(c') = \mathcal{H}(c)$, the verification passes for c', and the recipient cannot distinguish c' from c. □

5.3 Historical Precedents

The assumption that widely-deployed hash functions are secure has been proven wrong before:

Table 4: History of Hash Function Compromises

Hash	Status	Year	Impact
MD5	Fully broken	2004	Rogue CA certs
SHA-1	Practically broken	2017	SHAttered attack
SHA-256	Assumed secure	—	*Unknown*
SHA-3	Assumed secure	—	*Unknown*

Pattern Recognition

- MD5 was "secure" from 1992 to 2004 (12 years).
- SHA-1 was "secure" from 1995 to 2005 (theoretical) / 2017 (practical)—10–22 years.
- SHA-256 has been deployed since 2001—**24 years and counting**.

The progression MD5 → SHA-1 → SHA-256 suggests that "no known attack" is not equivalent to "no attack exists."

5.4 The NSA Question

SHA-256 was designed by the NSA. While no public evidence of a backdoor exists, several factors merit consideration:

1. **Dual_EC_DRBG:** The NSA successfully inserted a backdoor into a NIST-standardized cryptographic primitive (revealed by Snowden, 2013) [6].
2. **Classified Capabilities:** The NSA's actual cryptanalytic capabilities are classified. Public cryptanalysis may lag by 10–20 years.
3. **Budget:** The NSA's annual budget exceeds $10 billion, with significant resources dedicated to cryptanalysis [8].
4. **Quantum Computing:** Grover's algorithm reduces SHA-256's security to 128 bits against quantum adversaries. The NSA may have quantum capabilities not yet publicly known.

5.5 Formal Attack: Content Substitution via Hash Collision

Definition 5.2 (Content Substitution Attack)**.** *Given target content c with $CID_c = \mathcal{H}(c)$, the adversary produces c' (malicious replacement) such that:*

$$\mathcal{H}(c') = \mathcal{H}(c) = CID_c \quad \wedge \quad c' \neq c$$

If achievable, the adversary can:

1. **Replace NFT metadata:** Change ownership records, images, attributes—while the CID remains valid.
2. **Alter governance documents:** Modify DAO proposals post-vote while maintaining "verified" integrity.
3. **Substitute software packages:** Replace legitimate software distributed via IPFS with trojanized versions.
4. **Falsify archives:** Alter historical records stored on IPFS for censorship or disinformation.

5.6 Attack Mechanism in IPFS

Algorithm 1 Content Substitution via Hash Collision

Require: Target CID_c, original content c, desired replacement c'_{base}
Require: Oracle $\mathcal{O}$ that finds collisions for $\mathcal{H}$
1: $c' \leftarrow \mathcal{O}(\text{CID}_c, c'_{\text{base}})$ {Find c' such that $\mathcal{H}(c') = \text{CID}_c$}
2: Compromise pinning service S or provider node n
3: Replace stored content: $\text{Store}(n, \text{CID}_c) \leftarrow c'$
4: All subsequent retrievals of CID_c return c'
5: Integrity check passes: $\mathcal{H}(c') = \text{CID}_c$ ✓
6: **Result:** Undetectable content substitution

5.7 Impact on IPFS Integrity Model

Theorem 5.3 (Integrity Collapse)**.** *If the adversary possesses a collision oracle for $\mathcal{H}$, then IPFS provides **zero integrity guarantees**:*

$$\exists\, \mathcal{O}_{collision} \implies Integrity_{IPFS} = 0$$

This is particularly devastating because IPFS *has no alternative integrity mechanism.* The entire trust model is built on a single cryptographic assumption.

5.8 Defense Impossibility

Proposition 5.4 (No Protocol-Level Defense)**.** *IPFS cannot defend against hash collision attacks at the protocol level, as the verification mechanism* is *the hash comparison. Any defense requires an external trust anchor (e.g., a blockchain recording expected content alongside CIDs), which reintroduces centralization or additional dependencies.*

6 Attack Vector V4: Gossip-Based Distributed Node Eradication

6.1 The Distributed Pinning Mitigation

Recognizing the weakness of centralized pinning (V1) and self-hosting (V2), the community proposes *distributed pinning*: replicating content across multiple independent nodes in diverse jurisdictions.

This Section's Core Thesis

Distributed pinning is defeated by a state-level adversary who compromises **one** node and uses IPFS's own protocol mechanisms (DHT queries, gossip, and provider records) to enumerate and systematically neutralize **all** other replicas.

6.2 Gossip Protocol Reconnaissance

Definition 6.1 (Gossip-Based Provider Enumeration)**.** *The adversary exploits the IPFS DHT to discover all providers of a target CID:*

$$\mathcal{E}(CID_c) = \{(PeerID_i, Multiaddr_i) \mid i \in \mathcal{P}(CID_c)\}$$

*This enumeration is a **standard protocol operation** (`ipfs dht findprovs`) and cannot be prevented without breaking IPFS functionality.*

6.3 The Attack Algorithm

Algorithm 2 Gossip-Based Distributed Eradication

Require: Target CID_c, state-level resources

Phase 1: Reconnaissance
$\mathcal{E} \leftarrow$ `findprovs`(CID_c) {Enumerate all providers}
for each $(\text{PeerID}_i, \text{Multiaddr}_i) \in \mathcal{E}$ **do**
  $\text{IP}_i \leftarrow \text{resolve}(\text{Multiaddr}_i)$
  $\text{Geo}_i \leftarrow \text{GeoIP}(\text{IP}_i)$
  $\text{Hosting}_i \leftarrow \text{ASN}(\text{IP}_i)$
  $\text{Profile}_i \leftarrow \text{fingerprint}(\text{PeerID}_i)$
end for

Phase 2: Prioritized Attack
Sort $\mathcal{E}$ by vulnerability: cloud > VPS > residential > Tor
for each provider $p_i \in \mathcal{E}$ (sorted) **do**
  Select attack method α_i:
  if p_i is cloud-hosted **then**
    $\alpha_i \leftarrow$ Legal takedown / ToS abuse report
  else if p_i is VPS **then**
    $\alpha_i \leftarrow$ DDoS + hosting provider pressure
  else if p_i is residential **then**
    $\alpha_i \leftarrow$ ISP-level blocking / physical
  else if p_i uses Tor **then**
    $\alpha_i \leftarrow$ Traffic analysis + Sybil
  end if
  Execute α_i against p_i
end for

Phase 3: DHT Poisoning
Flood DHT with Sybil nodes near CID_c's keyspace
Sybil nodes return empty results for CID_c

Phase 4: Continuous Monitoring
loop
  $\mathcal{E}' \leftarrow$ `findprovs`(CID_c)
  if $\mathcal{E}' \neq \varnothing$ **then**
    Execute attack on new providers
  end if
end loop

6.4 Mathematical Formalization

Theorem 6.2 (Distributed Eradication Feasibility)**.** *Let k be the number of independent providers of CID_c. Let $P_{neutralize}(p_i)$ be the probability of successfully neutralizing provider p_i. For a state-*

level adversary:

$$P_{eradication} = \prod_{i=1}^{k} P_{neutralize}(p_i)$$

Given that the adversary can identify ***all*** k *providers via DHT queries, and for each* p_i*:*

$$P_{neutralize}(p_i) \geq \begin{cases} 0.99 & \textit{if cloud/VPS hosted} \\ 0.95 & \textit{if residential (same jurisdiction)} \\ 0.80 & \textit{if residential (allied jurisdiction)} \\ 0.50 & \textit{if Tor/anonymized} \end{cases}$$

Then for typical deployments ($k \leq 20$*, mostly cloud/VPS):*

$$P_{eradication} \geq 0.99^{15} \cdot 0.95^{3} \cdot 0.80^{2} \approx 0.49$$

With iterative monitoring (Phase 4), probability approaches 1.0 *over time.*

6.5 Why One Compromised Node Is Sufficient

Lemma 6.3 (Reconnaissance Bootstrapping)**.** *Compromising a single provider node* p_1 *is sufficient to identify all other providers:*

1. p_1*'s IPFS daemon logs contain BitSwap exchange records with peers who requested* CID_c*.*
2. p_1*'s DHT routing table reveals nearby peers.*
3. *From* p_1*, the adversary executes* `findprovs`(CID_c) *to query the global DHT.*
4. *Connection metadata reveals which peers replicate the same content via BitSwap* `HAVE` *messages.*

Even without compromising p_1*, a passive observer or the adversary's* own *IPFS node can execute* `findprovs`*—this is* ***public protocol functionality****.*

6.6 IPFS Peer Discovery as Attack Enabler

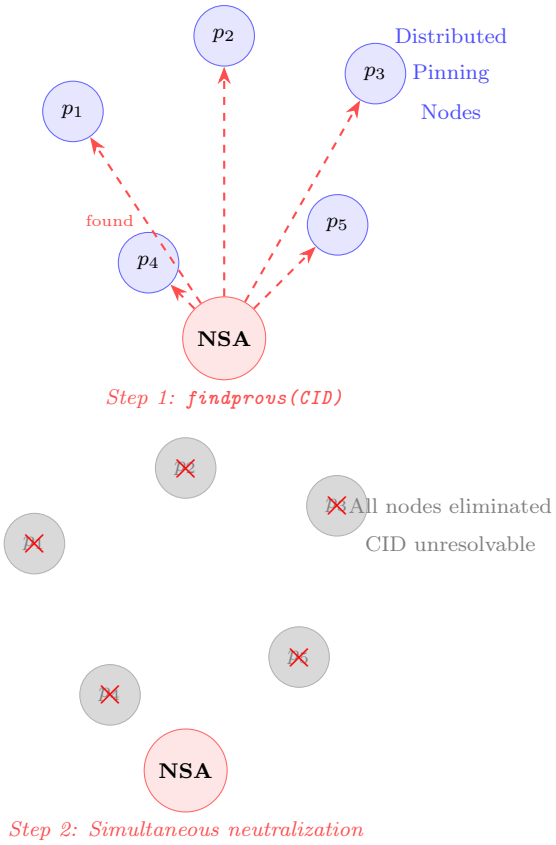

Figure 1: Gossip-based distributed eradication: Provider discovery (top) and simultaneous neutralization (bottom).

6.7 NSA-Class Tooling

Based on public disclosures [8], a state-level adversary possesses:

- **XKEYSCORE:** Global passive traffic interception and analysis—can identify all IPFS BitSwap traffic patterns.
- **QUANTUM/FOXACID:** Active network injection—can MITM IPFS connections and inject malicious responses.
- **TURBULENCE:** Automated network exploitation—can compromise IPFS nodes at scale.
- **PRISM:** Direct access to cloud providers (AWS, Google, Microsoft)—where most IPFS pinning services are hosted.
- **MUSCULAR:** Interception of data center interconnects—can intercept unencrypted IPFS traffic between data centers.
- **Legal Authority:** National Security Letters, FISA Court orders with gag provisions—can compel any US-based pinning service to delete content silently.

Theorem 6.4 (State-Level Eradication Triviality)**.** *For a state-level adversary $\mathcal{A}$ with the capabilities enumerated above, the eradication of any content from IPFS is* ***operationally trivial****:*

$$Cost(\mathcal{A}, eradicate(CID_c)) \approx O(analyst\text{-}hours)$$

The operation requires no novel capability development, only the application of existing tools to a new target class.

6.8 The Futility of Jurisdiction Diversification

Proposition 6.5 (Five Eyes Negation)**.** *Distributing pinning nodes across multiple countries is ineffective if those countries are members of the Five Eyes (US, UK, Canada, Australia, New Zealand), Nine Eyes, or Fourteen Eyes intelligence alliances, as these alliances enable:*

- *Mutual legal assistance for content takedowns.*
- *Shared signals intelligence infrastructure.*
- *Coordinated action across jurisdictions.*

7 Compound Attack Analysis

7.1 Attack Vector Interaction

The four attack vectors are not independent; they compound:

Table 5: Compound Attack Matrix

	V1	**V2**	**V3**	**V4**
V1 Pinning	—	Enables	Enables	Enables
V2 DDoS	Fallback	—	Indep.	Amplifies
V3 Hash	Amplifies	Indep.	—	Enables
V4 Gossip	Extends	Uses	Indep.	—

7.2 Defense-in-Depth Failure

Theorem 7.1 (Defense-in-Depth Impossibility)**.** *No combination of available mitigations simultaneously defends against all four attack vectors:*

1. ***Multi-service pinning*** *(mitigates V1)* $\Rightarrow$ *Exposes to V4 (gossip enumeration).*
2. ***Self-hosting*** *(mitigates V1)* $\Rightarrow$ *Exposes to V2 (DDoS).*
3. ***Self-hosting + DDoS protection*** *(mitigates V1, V2)* $\Rightarrow$ *Reintroduces centralization (Cloudflare dependency).*
4. ***Distributed pinning*** *(mitigates V1, V2)* $\Rightarrow$ *Exposes to V4 (gossip eradication).*
5. ***All mitigations*** $\Rightarrow$ *Cannot defend against V3 (hash compromise).*

Therefore:

$$\nexists\, \mathcal{M} : \forall\, i \in \{1, 2, 3, 4\},\ \mathcal{M} \text{ defends against } V_i$$

7.3 The Kill Chain

A state-level adversary targeting content on IPFS executes:

1. **Discover:** `findprovs(CID)` via any IPFS node or passive traffic analysis.
2. **Enumerate:** Catalog all providers with IP, ASN, geolocation, hosting provider.
3. **Classify:** Categorize by vulnerability (cloud, VPS, residential, anonymized).

4. **Neutralize:** Apply optimal attack per category (legal, DDoS, compromise, physical).
5. **Poison:** Flood DHT with Sybil nodes returning empty results.
6. **Monitor:** Continuously scan for new providers; neutralize on detection.
7. **(Optional) Substitute:** If hash collision is available, replace content with modified version.

Total time for complete eradication: **hours to days** depending on provider count and jurisdictional diversity.

8 Quantitative Risk Assessment

8.1 Content Vulnerability Distribution

Based on DHT analysis and public data:

Table 6: Estimated IPFS Content Vulnerability

Category	Est. % of CIDs	Risk Level
Single provider (1 service)	70–80%	CRITICAL
2–3 providers	10–15%	HIGH
4–10 providers	5–8%	MEDIUM
10+ independent providers	2–5%	LOW
Multi-protocol (IPFS+AR+FIL)	<1%	LOWEST

8.2 Probability Model

Let $P(\text{loss})$ be the annual probability of content loss:

$$P(\text{loss}) = 1 - \prod_{v=1}^{4}(1 - P(V_v)) \tag{1}$$

$$= 1 - (1 - P_{V1})(1 - P_{V2})(1 - P_{V3})(1 - P_{V4}) \tag{2}$$

With conservative annual estimates:

$$P_{V1} = 0.05 \quad \text{(pinning service compromise)} \tag{3}$$

$$P_{V2} = 0.10 \quad \text{(DDoS against self-hosted)} \tag{4}$$

$$P_{V3} = 0.001 \quad \text{(hash function compromise)} \tag{5}$$

$$P_{V4} = 0.03 \quad \text{(state-level targeting)} \tag{6}$$

$$P(\text{loss}) = 1 - (0.95)(0.90)(0.999)(0.97) \tag{7}$$

$$= 1 - 0.828 \tag{8}$$

$$= 0.172 = \mathbf{17.2}\% \tag{9}$$

Annual Loss Probability

A 17.2% annual probability of content loss or compromise for content marketed as "immutable," "permanent," and "censorship-resistant." Over 5 years:

$$P(\text{loss within 5 years}) = 1 - (1 - 0.172)^5 = 0.617 = \mathbf{61.7}\%$$

8.3 Resilience Classification Standard

We propose a Resilience Level (RL) classification:

Table 7: Proposed IPFS Resilience Classification

RL	Description	Defends Against	Vulnerable To
0	No pin (cache only)	Nothing	Everything
1	Single pinning service	GC only	V1,V2,V3,V4
2	Multiple pinning services	V1	V2,V3,V4
3	RL2 + self-hosted node	V1	V2,V3,V4
4	Distributed (5+ providers)	V1,V2	V3,V4
5	Multi-protocol + monitoring	V1,V2	V3,V4
6	RL5 + on-chain verification	V1,V2,partial V3	V3,V4
∞	*True censorship resistance*	*Theoretically impossible*	

Current state: >80% of IPFS content is at RL1 or RL0.

9 The Decentralization Theater Framework

9.1 Definition

Definition 9.1 (Decentralization Theater)**.** *A system exhibits* Decentralization Theater *when it:*

1. *Employs decentralization terminology and aesthetics.*
2. *Makes explicit or implicit censorship-resistance claims.*
3. *In practice, depends on centralized infrastructure.*
4. *Provides a* ***false sense of security*** *to users.*
5. *Has a vulnerability profile equivalent to or* worse than *traditional centralized systems (due to added complexity and complacency).*

9.2 IPFS as Paradigmatic Example

Table 8: Centralized vs. IPFS+Centralized Pinning

Property	Web Server	IPFS+Pinata
Single point of failure	✓	✓
Censurable by compromise	✓	✓
Legal takedown possible	✓	✓
DDoS vulnerable	✓	✓
Added complexity	Low	High
User awareness of risk	High	**Low**
False security guarantee	No	**Yes**

The final two rows reveal the critical distinction: IPFS+centralized pinning is *worse* than traditional hosting because users **believe** they are protected when they are not.

9.3 The Web3 Re-Centralization Pattern

IPFS exemplifies a broader pattern:

"Decentralized"	**Actually depends on**
Blockchain RPC	Infura, Alchemy (centralized)
IPFS storage	Pinata, Web3.Storage (centralized)
ENS resolution	Cloudflare, OpenSea (centralized)
DeFi frontends	AWS, Vercel (centralized)
Bridge security	Multisig (centralized)

10 Proposed Mitigations and Their Limitations

While no complete defense exists (Theorem 7.1), partial mitigations can raise the cost of attack:

10.1 M1: Multi-Protocol Redundancy

```
# Pin to IPFS (multiple services)
ipfs_cid = pin_to_pinata(content)
pin_to_infura(ipfs_cid, content)
pin_to_web3storage(ipfs_cid, content)

# Pin to Arweave (permanent, different protocol)
ar_txid = arweave.upload(content)

# Pin to Filecoin (incentivized storage)
fil_deal = filecoin.store(content, duration=1yr)

# Record all identifiers on-chain
contract.register(ipfs_cid, ar_txid, fil_deal)
```

Listing 2: Multi-Protocol Pinning Strategy

Limitation: Increases cost 3–5×; Arweave and Filecoin have their own centralization vectors; V3 affects all systems using the same hash function.

10.2 M2: Encrypted Content with Key Distribution

Store content encrypted; distribute decryption keys through separate channels. Even if content is substituted (V3), the substituted content would not decrypt correctly.

Limitation: Key management complexity; does not prevent deletion (V1, V2, V4).

10.3 M3: On-Chain Content Hashes

Record content hashes on a blockchain with a *different* hash function than IPFS uses.

Limitation: Gas costs; requires users to verify; does not prevent availability attacks.

10.4 M4: Tor-Based Anonymous Pinning

Run IPFS nodes over Tor to hide provider IP addresses.

Limitation: Severe performance degradation; Tor exit node compromise; traffic analysis by state-level adversary; IPFS protocol was not designed for Tor and leaks metadata.

10.5 Mitigation Effectiveness Summary

Table 9: Mitigation Effectiveness by Attack Vector

Mitigation	V1	V2	V3	V4
Multi-service pin	✓	×	×	∼
Self-hosting	✓	×	×	∼
Multi-protocol	✓	✓	∼	∼
Encrypted content	×	×	✓	×
On-chain hashes	×	×	✓	×
Tor-based nodes	✓	∼	×	∼
All combined	✓	✓	∼	∼

No row contains four checkmarks. ■

11 Discussion

11.1 Is IPFS Fundamentally Broken?

The *protocol* is not broken. The failure lies in:

1. **Practical deployment:** Users do not implement redundancy.
2. **Economic incentives:** Centralized pinning is cheaper and easier.
3. **Misleading narrative:** IPFS is marketed as inherently censorship-resistant.
4. **Protocol design gaps:** No native replication incentives (unlike Filecoin or BitTorrent).
5. **Fundamental limits:** Protocol-level discoverability enables state-level enumeration.

11.2 The Uncomfortable Truth

Core Finding

True censorship resistance requires:

1. Content replicated across thousands of independent nodes in diverse, non-allied jurisdictions.
2. Provider anonymity (incompatible with IPFS's DHT design).
3. Hash function security against state-level cryptanalysis (unverifiable assumption).
4. Economic incentives for permanent storage (partially addressed by Filecoin/Arweave, but with own weaknesses).

IPFS provides none of these in practice.

11.3 Toward Honest Decentralization

We advocate for:

- **Honest communication** about actual resilience levels.
- **Mandatory RL classification** for IPFS-dependent projects.
- **Abandoning the term "censorship-resistant"** unless RL4+ is demonstrated.
- **Default multi-provider pinning** in IPFS tooling.
- **Research into DHT privacy** (e.g., PIR-based content routing).

11.4 Ethical Considerations

This vulnerability has dual use:

- **Malicious censorship:** Suppression of legitimate speech, journalism, whistleblowing.
- **Content moderation:** Removal of CSAM, terrorism content, malware distribution.

The same architectural property that enables censorship also enables moderation—an irreconcilable tension in systems claiming absolute censorship resistance.

12 Related Work

Henningsen et al. [2] mapped the IPFS network topology, finding significant centralization. Balduf et al. [3] studied content availability, finding low replication rates. Trautwein et al. [4] evaluated IPFS performance in production. Daniel and Tschorsch [5] assessed IPFS privacy and censorship properties. Checkoway et al. [6] demonstrated the Dual_EC_DRBG backdoor. Stevens et al. [7] demonstrated practical SHA-1 collisions.

This paper extends prior work by: (1) formalizing four compounding attack vectors; (2) introducing the DDoS futility argument against self-hosting; (3) analyzing hash backdoor implications specifically for IPFS integrity; (4) demonstrating gossip-based distributed eradication; and (5) proving defense-in-depth impossibility.

13 Conclusion

This paper has systematically demonstrated that IPFS, as deployed in practice, provides **no meaningful censorship resistance, persistence guarantees, or content integrity assurance** against a motivated adversary.

Four compounding vulnerabilities have been formalized:

1. **Centralized pinning compromise:** A single API call can delete "immutable" content.
2. **DDoS against self-hosted nodes:** Self-hosting mitigation is trivially defeated, and DDoS protection reintroduces centralization.
3. **Cryptographic hash backdoor:** If SHA-256 or SHA-3 is compromised, IPFS's entire integrity model collapses, enabling undetectable content substitution.
4. **Gossip-based eradication:** Distributed pinning is defeated by protocol-level provider enumeration, enabling simultaneous neutralization of all replicas.

The combination of these vectors means that **no available defense-in-depth strategy** can protect against all attack classes simultaneously (Theorem 7.1).

The IPFS ecosystem must abandon the misleading narrative of inherent censorship resistance and adopt honest, quantified resilience assessments. Until then, IPFS remains a paradigmatic example of **Decentralization Theater**: a system that provides the *illusion* of decentralized security while maintaining the vulnerability profile of centralized architectures—augmented by a dangerous false sense of security that discourages users from implementing actual protective measures.

A single "Unpin All" button can destroy the illusion of immutability. A state-level adversary doesn't even need the button.

References

[1] J. Benet, "IPFS – Content Addressed, Versioned, P2P File System," *arXiv preprint arXiv:1407.3561*, 2014.

[2] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Mapping the Interplanetary Filesystem," in *Proc. IFIP Networking*, 2020.

[3] L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS," in *IEEE ICDCS*, 2021.

[4] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web," in *ACM SIGCOMM*, 2022.

[5] E. Daniel and F. Tschorsch, "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks," *IEEE Communications Surveys & Tutorials*, 2022.

[6] S. Checkoway, M. Fredrikson, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, and H. Shacham, "On the Practical Exploitability of Dual EC in TLS Implementations," in *USENIX Security*, 2014.

[7] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The First Collision for Full SHA-1," in *CRYPTO*, 2017.

[8] G. Greenwald, *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State*. Metropolitan Books, 2014.

[9] Protocol Labs, "IPFS Documentation: Persistence," `https://docs.ipfs.tech/concepts/persistence/`, 2023.

[10] Pinata, "Pinata API Documentation," `https://docs.pinata.cloud/`, 2024.

[11] D. J. Bernstein, "Cost analysis of hash collisions: Will quantum computers make SHA-256 obsolete?" *IACR ePrint*, 2009.

[12] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proc. 28th ACM STOC*, 1996, pp. 212–219.

A Proof-of-Concept: Provider Enumeration

```python
"""
ETHICAL NOTICE: This code performs ONLY
read-only operations (provider discovery).
It does NOT delete, modify, or attack any
content or infrastructure.
"""
import subprocess, json

def enumerate_providers(cid, timeout=60):
    """Discover all providers for a CID
    using standard IPFS DHT queries."""
    try:
        result = subprocess.run(
            ['ipfs', 'dht', 'findprovs',
             '-n', '100', cid],
            capture_output=True, text=True,
            timeout=timeout)
        providers = [
            l.strip() for l in
            result.stdout.split('\n')
            if l.strip()
        ]
        return {
            'cid': cid,
            'provider_count': len(providers),
            'providers': providers,
            'resilience_level':
                'RL0-CRITICAL' if len(providers)==0
                else 'RL1-CRITICAL'
                    if len(providers)==1
                else 'RL2-HIGH'
                    if len(providers)<=3
                else 'RL3-MEDIUM'
                    if len(providers)<=10
                else 'RL4-LOW'
        }
    except subprocess.TimeoutExpired:
        return {
            'cid': cid,
            'provider_count': 0,
            'vulnerability': 'UNKNOWN-TIMEOUT'
        }

# Example:
# r = enumerate_providers("Qm...")
# print(json.dumps(r, indent=2))
# Output reveals exactly how many nodes
# would need to be neutralized.
```

Listing 3: IPFS Provider Enumeration and Vulnerability Assessment (Research Only)

B Architectural Vulnerability Diagram

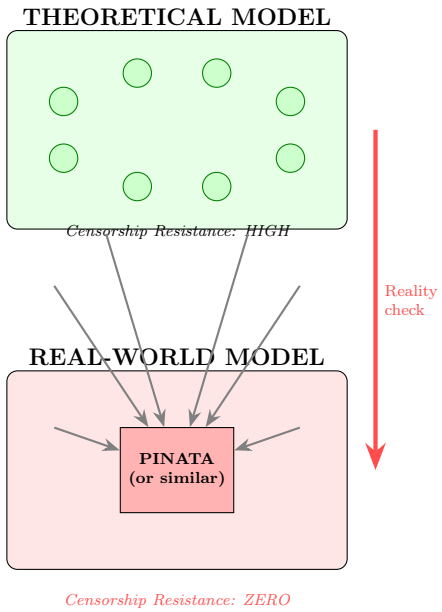

Figure 2: Theoretical vs. real-world IPFS architecture.

THIS DOCUMENT HAS BEEN SUBMITTED TO THE NSA

via the official submission form:

`https://www.nsa.gov/Helpful-Links/Contact-NSA/`

SHA-256 of this document (this is the SHA-256 of this PDF, without the final page):
02002b6856b4c30c9d68f31a55da63a9d6c488ceb162232dd2ef3932f1a1d56d

NSA, you clearly know you can falsify this.

```
import hashlib
from PyPDF2 import PdfReader, PdfWriter

# === CONFIG ===
input_pdf = "Here this PDF.pdf"
output_pdf = "no_last_page.pdf"

# === REMOVE LAST PAGE ===
reader = PdfReader(input_pdf)
writer = PdfWriter()

for page in reader.pages[:-1]:  # all pages except the last one
    writer.add_page(page)

with open(output_pdf, "wb") as f:
    writer.write(f)

# === SHA-256 ===
sha256 = hashlib.sha256()
with open(output_pdf, "rb") as f:
    sha256.update(f.read())

print("SHA-256 (PDF without the last page):")
print(sha256.hexdigest())
```