BitFS: 基于区块链的点对点加密文件系统

Alex Tong
alex@bitfs.org

v0.0.1 2026-03-20

摘要

我们提出 *BitFS*，一个构建在 *BSV* 区块链上的去中心化加密文件系统。*BitFS* 将 *Unix* 文件系统语义映射到区块链有向无环图（*Metanet DAG*），其中每个文件和目录对应一个链上节点。所有文件内容默认使用 *Method 42 ECDH* 密钥派生进行 *AES-256-GCM* 加密，访问控制不取决于"是否加密"，而取决于"谁能派生解密密钥"——私有、免费和付费三种访问模式共享同一密码学机制，仅在密钥分发方式上有所差异。*HD* 钱包镜像文件系统目录树，为每个节点提供稳定的公钥身份和确定性密钥派生能力。内容以 *key_hash = SHA256(SHA256(plaintext))* 进行寻址，兼做密钥派生盐值和内容承诺标识，元数据则记录在 *Metanet* 交易中。无信任的数据交易通过 *HTLC* 原子交换实现：卖方揭示密钥胶囊即哈希原像，买方获得解密密钥。系统以 *SPV* 模式运行，从不主动查询区块链。*Agent* 优先的 *HTTP* 接口与 *payment* 支付协议使 *AI Agent* 能够自主发现、浏览和购买去中心化内容。

1. 引言

现有的去中心化存储系统各自解决了问题的一部分，但无一构建出完整的解决方案。IPFS 实现了内容寻址的点对点分发，但缺少原生加密和支付机制——数据以明文存储和传输，节点没有经济激励去服务非自身需要的内容。Filecoin 弥补了激励缺口，通过付费让存储提供者持有数据，但其验证机制依赖计算密集的零知识证明，每个扇区需要数小时 GPU 计算，而检索市场只是一个附带考虑。中心化云存储方便高效，但要求用户信任单一实体，面临审查、封禁和单点故障风险。

这些系统的根本矛盾在于三个被忽视的观察。第一，元数据与内容需要不同的保障——文件所有权记录应当永久上链，而文件内容本身不必也不应全部上链。第二，加密应当是默认行为而非可选功能——" 公开" 与" 私有" 仅是密钥分发方式的差异，而非" 是否加密" 的差异。第三，AI Agent 是一等公民——在自主代理日益普及的时代，系统接口必须对机器可读，支持程序化发现、浏览和支付，而非依赖人类中介。

BitFS 的核心定义可以用一句话概括：Unix 文件系统语义 + 区块链公钥 inode + 确定性 ECDH 加密，单一密码学框架（Method 42）统一提供加密、访问控制、认证、交易和存储证明。

2. Metanet 即文件系统

Metanet 协议 [2] 定义了 BSV 区块链上的一种有向无环图结构。每个 Metanet 节点对应一笔 `OP_RETURN` 交易，交易载荷中携带节点公钥 P_{node}（33 字节压缩公钥）和父交易引用。节点之间的边通过密码学建立：子节点交易的输入必须包含由父节点私钥签署的花费，从而在密码学意义上证明父子关系。

BitFS 将这一 DAG 结构映射为 Unix 文件系统语义。对应关系如表 1 所示。

与 Unix inode 不包含文件名的设计一致，BitFS 中的文件名仅存在于父目录的子节点列表中。系统定义三种节点类型。**FILE** 节点携带 key_hash、MIME 类型和文件大小等内容元数据。**DIR** 节点维护一个 children 列表和 next_child_index 计数器，用于分配新的子节点编号。**LINK** 节点支持两种链接形式：SOFT 链接指向一个 P_{node}，始终解析到该节点的最新版本；SOFT_REMOTE 链接以 domain/path 的形式指向另一个信任域下的节点，实现跨信任边界的引用。

Unix 概念	**BitFS** 对应
inode	P_{node}（压缩公钥，33 字节）
目录项	ChildEntry(index, name, type, pubkey)
inode 编号	index（每目录单调递增）
`..`（父目录）	载荷中的 parent 字段

Table 1: Unix 文件系统到 BitFS 的映射

硬链接以隐式机制实现：多个目录条目可以指向同一个 P_{node}，无需专门的链接类型枚举值。版本控制也是内在的：同一 P_{node} 可以关联多笔交易，区块高度最大者即为当前版本。

3. HD 密钥派生

BitFS 使用 BIP32 [3] 分层确定性钱包将密钥层次与文件系统目录树一一对应。钱包结构遵循 BIP44 约定，使用 BSV 的 coin type 236：

```
m/44'/236'/0'          费用密钥链（所有 Vault 共享）
m/44'/236'/1'/0/0      Vault #0 根目录
m/44'/236'/1'/0/0/1    根目录第一个子节点
m/44'/236'/1'/0/0/1/3 第一个子节点的第三个子节点
m/44'/236'/2'/0/0      Vault #1 根目录（独立树）
```

Vault 是 BitFS 中的核心组织概念。每个 Vault 对应一棵以 BIP32 账户为根的独立 Metanet 树，拥有自己的目录层次。多个 Vault 共享同一个 HD 种子，但维护完全独立的文件系统命名空间。每个 Vault 可以绑定独立的域名，从而在同一钱包下管理多个独立的站点或数据集。

这一设计带来两个核心属性。其一是稳定的节点身份：由于 P_{node} 来自确定性密钥派生而非随机生成，节点公钥不会随内容更新而变化，支持持久引用和稳定的链接。其二是确定性恢复：从 BIP39 助记词可以完整重建整个密钥层次结构。需要注意的是，密钥层次可以恢复，但链上的交易数据（文件元数据和目录结构）需要从备份中恢复。

4. 统一加密

BitFS 的加密方案基于 Method 42 [1]，使用 ECDH 密钥派生结合 AES-256-GCM 对称加密。加密流程分为四步：

```
1. key_hash = SHA256(SHA256(plaintext))
2. point = ECDH(D_node, P_node) = D_node * P_node
3. aes_key = HKDF-SHA256(point.x, key_hash, info="bitfs-file-encryption")
4. ciphertext = AES-GCM(plaintext, aes_key)
```

其中 D_{node} 是 BIP32 派生的节点私钥，$P_{node} = D_{node} \times G$ 是对应的节点公钥（即 Metanet inode），key_hash = SHA256(SHA256(plaintext)) 同时充当密钥派生的盐值和内容承诺标识。

D_{node} 直接使用 BIP32 派生密钥，保留了代数关系，从而支持目录树级别的 capsule 密钥派生。双重哈希设计确保 key_hash 不暴露原始数据的单层哈希值，同时兼做密钥派生盐值和内容寻址标识。

三种访问级别共享同一密码学机制，仅在密钥输入上有所差异，如表 2 所示。

免费数据采用” 平凡密钥” 技巧：使用 P_{node}.x 作为 KDF 输入，而 P_{node} 通过 DNS 公开发布，任何人都可以派生出解密密钥。但磁盘上的数据仍然是加密存储的，保持了统一的存储模型。

访问类型	密钥派生	谁能解密
私有	aes_key = HKDF(ECDH(D_{node}, P_{node}).x, key_hash, `"bitfs-file-encryption"`)	仅所有者（自加密）
免费	aes_key = HKDF(P_{node}.x, key_hash, `"bitfs-file-encryption"`)（平凡密钥）	知道 P_{node} 的任何人
付费	标准 Method 42 ECDH capsule 交换	买方（HTLC 交换后）

Table 2: 三种访问级别的密钥派生机制

私有数据则更进一步：整个 Metanet 载荷使用所有者的对称密钥加密，文件名、大小、时间戳和目录结构在链上完全不可见。

v0.0.1 版本的权限判定逻辑如下：读操作按 acl_ref、access 字段、owner fallback 的优先级链短路求值，acl_ref 解析失败时执行 fail-closed 策略；写操作仅限所有者，由应用层强制执行。

5. 内容寻址存储

BitFS 采用扁平的键值映射实现内容存储。每个文件的密文以 key_hash 为键存储在本地文件系统中：

```
~/.bitfs/data/{key_hash} -> 密文字节
```

Daemon 通过 HTTP `GET /data/{hash}` 端点提供内容服务。这一设计不依赖任何外部存储系统（如 IPFS），所有数据在本地自包含。

由于所有内容在写入前已经过加密，存储层仅处理不透明的字节序列，无需关心内容语义。去重自然发生：相同明文产生相同的 key_hash 和相同的密文，因此共享单一存储副本。

6. 交易结构

BitFS 中的每个文件系统操作对应一笔 Metanet 交易。交易结构如下：

```
输入:
  [0] 花费锁定到 P_parent 的 UTXO -> Sig(D_parent) 创建 Metanet 边
  [1] 花费费用密钥链 UTXO -> 支付矿工费

输出:
  [0] OP_RETURN: <MetaFlag> <P_node> <TxID_parent> <TLV 载荷>
  [1] P2PKH -> P_node     (1 sat，节点可花费输出)
  [2] P2PKH -> P_parent   (1 sat，刷新父节点 UTXO)
  [3] P2PKH -> 找零地址
```

输出 [2] 的设计尤为关键：它将 1 satoshi 返还给父节点地址，刷新父节点的 UTXO，形成自持续的 UTXO 链。这意味着一旦初始创建了根节点，后续操作不需要预先充值——系统自动维护每个节点的可花费输出。

TLV（Type-Length-Value）载荷编码了操作所需的全部信息：节点类型（FILE/DIR/LINK）、操作类型（CREATE/UPDATE/DELETE）、内容元数据（key_hash、MIME 类型、文件大小）、访问控制参数、目录子节点列表，以及可选字段如关键词、描述和域名绑定。

定价信息以 price_per_kb（satoshis/KB）的形式嵌入元数据，并支持目录级继承——子节点可以继承父目录的定价策略。Vault 还可以通过 `paymail bind` 命令关联 `user@domain` 地址，支持

`.well-known/bsvalias` 协议发现。

7. 无信任数据交易

BitFS 使用哈希时间锁合约（HTLC）实现无信任的数据交易。完整的交易流程如下：

```
1. 买方连接卖方 Daemon
2. Method 42 ECDH 握手（双向身份验证）
3. 买方请求目标文件的 capsule_hash
4. 卖方计算 capsule + capsule_hash
5. 卖方返回 capsule_hash + payment_address
6. 买方创建 HTLC 交易（SHA256 原像验证 + 超时退款）
7. 买方广播 HTLC
8. 卖方揭示 capsule（原像）-> 领取付款
9. 买方从链上读取 capsule -> 派生解密密钥 -> 解密文件
```

原子性保证的核心在于：卖方领取付款的行为本身就是揭示密钥胶囊。卖方收到付款当且仅当买方收到解密所需的 capsule。无需可信中介。

HTLC 使用 106 字节的纯 Bitcoin Script 实现，不依赖 sCrypt 或任何高级脚本框架：

```
<invoiceId> DROP
IF
  SHA256 <capsuleHash> EQUALVERIFY
  DUP HASH160 <sellerPkh> EQUALVERIFY CHECKSIG
ELSE
  DUP HASH160 <buyerPkh> EQUALVERIFY CHECKSIG
ENDIF
```

卖方通过 Claim 路径领取付款，提交 `<sig> <pubkey> <fileTxID||capsule> TRUE`，其中 64 字节的原像由文件交易 ID 和 capsule 拼接而成。买方通过 Refund 路径取回超时未被领取的资金，提交 `<sig> <pubkey> FALSE`，超时由交易的 nLockTime 字段强制执行，不使用 `OP_CLTV`。Go 和 TypeScript 两种实现产生字节一致的 106 字节脚本。

Method 42 握手协议在交易开始前建立安全通道。双方交换各自的公钥 P_{buyer}/P_{seller}、随机 nonce 和时间戳，然后各自计算会话密钥 session_key = SHA256(ECDH.x ‖ nonce_b ‖ nonce_s)，并通过 HMAC 验证。卖方的 P_{seller} 必须与 DNS 公布的 P_{node} 一致，从而防止中间人攻击。

卖方在整个过程中完全无状态：不维护买家数据库，不保持会话，不记录交易历史。

8. SPV 模式

BitFS 完全以 SPV（简化支付验证）[4] 模式运行。所有者在本地存储自己构建的交易及其 Merkle 证明，访问者从所有者的 Daemon 获取元数据，从不直接查询区块链。元数据的真实性通过 Merkle 证明和区块头链验证来确保。

本地存储结构如下：

```
~/.bitfs/
  wallet.enc           Argon2id 加密的 HD 种子
  txstore/{txid}.tx    完整交易数据
  txstore/{txid}.proof Merkle 证明
  headers/             区块头链
  cache/               访问者元数据 + 解密内容 + 密钥胶囊缓存
```

第三方区块链索引服务仅在所有者 Daemon 不可达时作为降级备用方案。在种子恢复场景中，HD 密钥层次可以从 BIP39 助记词完整还原，但链上的交易数据（文件元数据、目录结构、Merkle 证明）需要从备份中恢复。

9. DNS 解析与发布

BitFS 使用两种 DNS 记录将域名与文件系统身份绑定。TXT 记录声明节点公钥身份：

```
_bitfs.example.com TXT "bitfs=02a1b2c3..."
```

SRV 记录声明服务端点，支持多端点负载均衡：

```
_bitfs._tcp.example.com SRV 10 60 443 cdn1.example.com
```

域名绑定要求双向验证：DNS TXT 记录指向 P_{node}，而 Metanet 载荷中的 domain 字段指向域名，两者必须一致。SRV 记录的 priority 和 weight 字段提供类 CDN 的分发能力，允许所有者声明多个服务端点并控制流量分配。

完整的 URI 解析流程如下：

```
bitfs://example.com/docs/readme.txt
  -> DNS TXT -> P_node
  -> DNS SRV -> 端点列表
  -> 连接最优端点，Method 42 握手
  -> 请求元数据(SPV) -> 验证域名绑定
  -> 遍历目录树 -> 返回内容
```

对于不需要域名的场景，公钥直接寻址 `bitfs://<pubkey>/path` 可以绕过 DNS 解析，直接通过公钥定位节点。

10. Agent 优先接口

BitFS 的 HTTP 接口设计以 AI Agent 为一等用户。同一端点通过内容协商同时服务人类和机器，如表 3 所示。

Accept 头	响应
`text/html`	HTML + WebMCP 工具声明（计划中）
`text/markdown`	CLI 使用指南
`application/json`	结构化元数据

Table 3: 内容协商机制

付费内容使用 HTTP 402 状态码。响应头包含定价信息：`X-Price`、`X-Price-Per-KB`、`X-File-Size`、`X-Invoice-Id` 和 `X-Expiry`。人类用户看到付费墙页面；浏览器 AI Agent 发现 WebMCP 工具声明后可自主完成支付；CLI Agent 收到命令模板 `bget --buy bitfs://...` 后直接执行。

核心洞察在于：拥有钱包的 AI Agent 可以透明地完成微支付。HTTP 402 从传统的访问壁垒转变为可编程的支付 API，使自主代理能够在无人干预的情况下发现、评估和购买去中心化内容。

11. 双模式命令行

BitFS 提供两组命令行工具，分别服务访问者和所有者。

只读工具（**b*** 系列）面向访问者，无状态运行，不需要钱包或私钥：

```
bls    类似 ls，列出目录内容
bcat   类似 cat，输出文件内容
bget   类似 wget，下载文件
bstat  类似 stat，显示文件元信息
btree  类似 tree，显示目录树
```

读写命令（**bitfs**）面向文件所有者，需要钱包和私钥：

```
bitfs put/mkdir/rm/mv/cp/link      文件系统操作
bitfs sell/encrypt/decrypt         交易与访问控制
bitfs vault/wallet/publish/daemon 基础设施管理
bitfs shell                        FTP 风格交互式 REPL
```

所有工具均支持 `--json` 标志（输出机器可读的 JSON 格式）和 `--offline` 标志（仅操作本地缓存，不连接网络）。

12. 激励层

BitFS 的加密原语——Method 42 ECDH 密钥派生——同样适用于构建存储证明机制。当内容向不同的 Metanet 网络节点分发时，每份副本使用节点特定的 ECDH 密钥进行重新加密，确保每个节点存储的数据在密码学上是唯一的。验证采用简单的 Merkle 挑战-响应协议，替代 Filecoin 所需的零知识证明。这将存储证明的计算开销从数小时 GPU 降低到毫秒级的 ECDH + AES 操作。

Metanet Chain [5] 是一条 BSV 同构侧链，为去中心化 CDN 提供经济基础。它拥有自有的 PoW 共识机制和代币经济模型（2100 万固定供应、减半计划），并通过 `OP_RETURN` Merkle 根定期锚定到 BSV 主链，继承主链的安全保障。Metanet Chain 为 Metanet 节点运营商提供挖矿奖励和存档合约收入等经济激励。Metanet 网络的详细设计见 Metanet 白皮书。

13. 结论

BitFS 将 Unix 文件系统的成熟语义映射到区块链元数据结构，在保留用户熟悉的操作模型（目录、文件、链接、权限）的同时，获得了区块链提供的不可篡改性和去中心化特性。所有文件内容默认加密，” 公开” 与” 私有” 的区别仅在于密钥分发方式而非是否加密，从而在统一的存储模型下同时支持私密数据和公开发布。

HTLC 原子交换实现了无信任的数据交易：卖方揭示密钥胶囊等价于领取付款，买方获得解密能力等价于支付完成，无需任何可信中介。单一密码学原语——ECDH 密钥派生——统一提供了加密、认证、支付验证和存储证明四项功能，避免了多套独立密码学系统的复杂性。

AI Agent 是 BitFS 的一等用户。HTTP 402 支付协议、内容协商机制和机器可读的元数据接口，使自主代理能够以机器速度与去中心化服务交互——发现内容、评估价格、完成支付、获取数据，全程无需人类干预。SPV 模式确保客户端轻量运行，从不依赖对区块链的直接查询。

BitFS 为自主 Agent 以机器速度与去中心化文件系统交互的未来而设计。

参考文献

1. C. S. Wright, “An Immutable File and Data Store,” nChain, 2019. Method 42: 使用 ECDSA/secp256k1 的确定性密钥派生实现逐文件加密.
2. nChain, “The Metanet Technical Summary v1.0,” 2020. 基于区块链的有向无环图，用于组织数据关系.
3. P. Wuille, “BIP32: Hierarchical Deterministic Wallets,” Bitcoin Improvement Proposals, 2012.
4. S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. 简化支付验证 (SPV), Section 8.
5. GB2608179A, “Multi-level Blockchain,” UK Intellectual Property Office, 2021. 多层区块链.