The ₿Word

How Institutions Can Embrace Bitcoin

Presentation Template

Supporting The Developer Ecosystem

Bitcoin:
Open Source Money

John Newbery
Executive Director, Brink

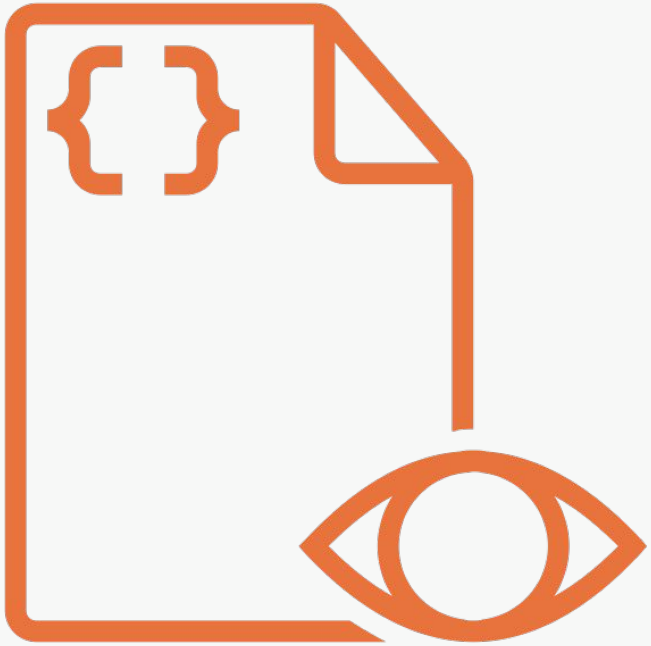
brink

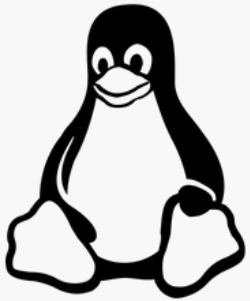
Bitcoin: Open Source Money

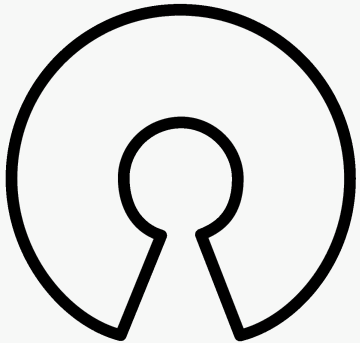
The Bitcoin network is developed and maintained by a decentralized group of contributors

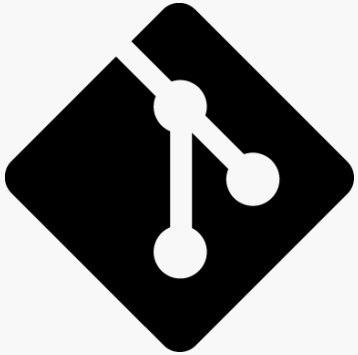
Free & Open Source Software

- **Free like speech**
- Unlike proprietary software:
  - **Run for any purpose**
  - **Download and view source code**
  - **Make modifications**
  - **Redistribute**
- Developed and maintained by public







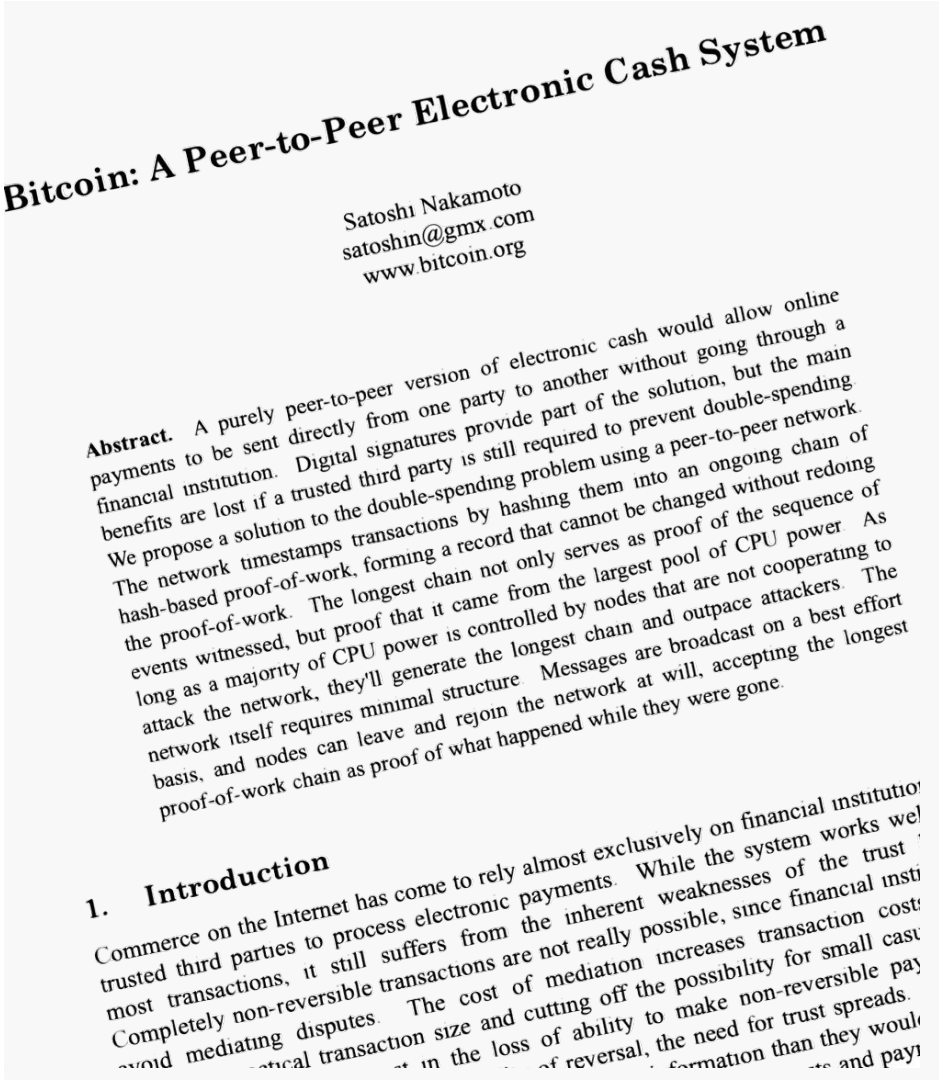


A (short) history of FOSS

- **1983-85**: GNU project
- **1991**: Linux operating system
- **1998**: Netscape
- **2005**: Git

Enter Satoshi

- **2007**: Satoshi starts working on Bitcoin
- **October 2008**: First announcement
- **January 2009**: Software released
- **December 2009**: First new contributor
- **2009-2010**: Various releases
- **December 2010**: Satoshi's last post



Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutio
trusted third parties to process electronic payments. While the system works wel
most transactions, it still suffers from the inherent weaknesses of the trust
Completely non-reversible transactions are not really possible, since financial insti
avoid mediating disputes. The cost of mediation increases transaction cost
ctical transaction size and cutting off the possibility for small casu
t, in the loss of ability to make non-reversible pay
of reversal, the need for trust spreads.
formation than they would
ts and payr

The Project Grows

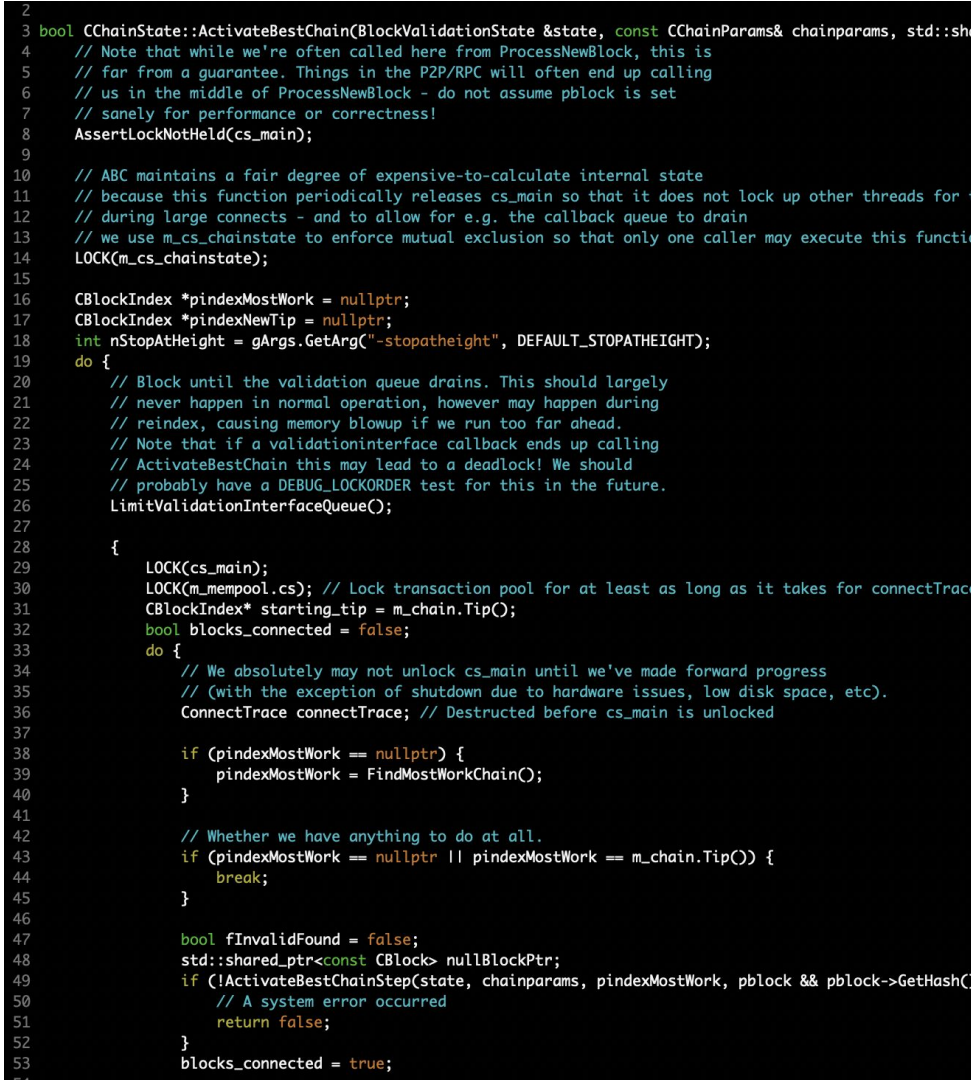
- **December 2010**: Gavin Andresen becomes maintainer
- **December 2010**: Development moves to Github
- **August 2011**: BIP process formalized
- **December 2013**: Bitcoin Core
- **2021**: Bitcoin Core has 6 maintainers and 800 code contributors

Roles in open source projects

- Research
- Maintenance
- Documentation
- Translations
- Development

What do the developers do?

- Review & test
- Find and fix bugs
- Improve performance
- Improve security
- Add features



```
bool CChainState::ActivateBestChain(BlockValidationState &state, const CChainParams& chainparams, std::sh
    // Note that while we're often called here from ProcessNewBlock, this is
    // far from a guarantee. Things in the P2P/RPC will often end up calling
    // us in the middle of ProcessNewBlock - do not assume pblock is set
    // sanely for performance or correctness!
    AssertLockNotHeld(cs_main);

    // ABC maintains a fair degree of expensive-to-calculate internal state
    // because this function periodically releases cs_main so that it does not lock up other threads for
    // during large connects - and to allow for e.g. the callback queue to drain
    // we use m_cs_chainstate to enforce mutual exclusion so that only one caller may execute this functi
    LOCK(m_cs_chainstate);

    CBlockIndex *pindexMostWork = nullptr;
    CBlockIndex *pindexNewTip = nullptr;
    int nStopAtHeight = gArgs.GetArg("-stopatheight", DEFAULT_STOPATHEIGHT);
    do {
        // Block until the validation queue drains. This should largely
        // never happen in normal operation, however may happen during
        // reindex, causing memory blowup if we run too far ahead.
        // Note that if a validationinterface callback ends up calling
        // ActivateBestChain this may lead to a deadlock! We should
        // probably have a DEBUG_LOCKORDER test for this in the future.
        LimitValidationInterfaceQueue();

        {
            LOCK(cs_main);
            LOCK(m_mempool.cs); // Lock transaction pool for at least as long as it takes for connectTrac
            CBlockIndex* starting_tip = m_chain.Tip();
            bool blocks_connected = false;
            do {
                // We absolutely may not unlock cs_main until we've made forward progress
                // (with the exception of shutdown due to hardware issues, low disk space, etc).
                ConnectTrace connectTrace; // Destructed before cs_main is unlocked

                if (pindexMostWork == nullptr) {
                    pindexMostWork = FindMostWorkChain();
                }

                // Whether we have anything to do at all.
                if (pindexMostWork == nullptr || pindexMostWork == m_chain.Tip()) {
                    break;
                }

                bool fInvalidFound = false;
                std::shared_ptr<const CBlock> nullBlockPtr;
                if (!ActivateBestChainStep(state, chainparams, pindexMostWork, pblock && pblock->GetHash()
                    // A system error occurred
                    return false;
                }
                blocks_connected = true;
```

Features

- Schnorr/Taproot:
  - **Scalability**
  - **Fungibility and Privacy**
  - **Functionality**
- Peer-to-peer improvements:
  - **Increased connectivity / eclipse protection**
  - **Network privacy**
- And much more ...

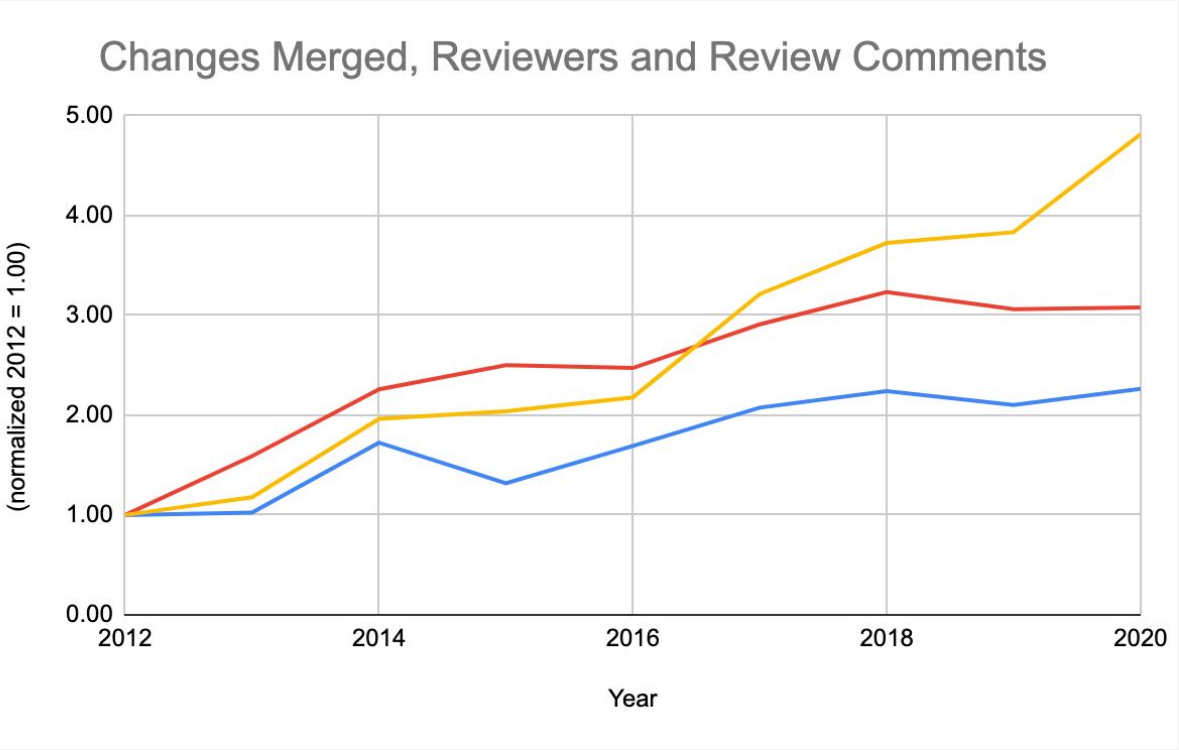
Bitcoin has bugs!

The Bitcoin Developer Ecosystem

libsecp256k1

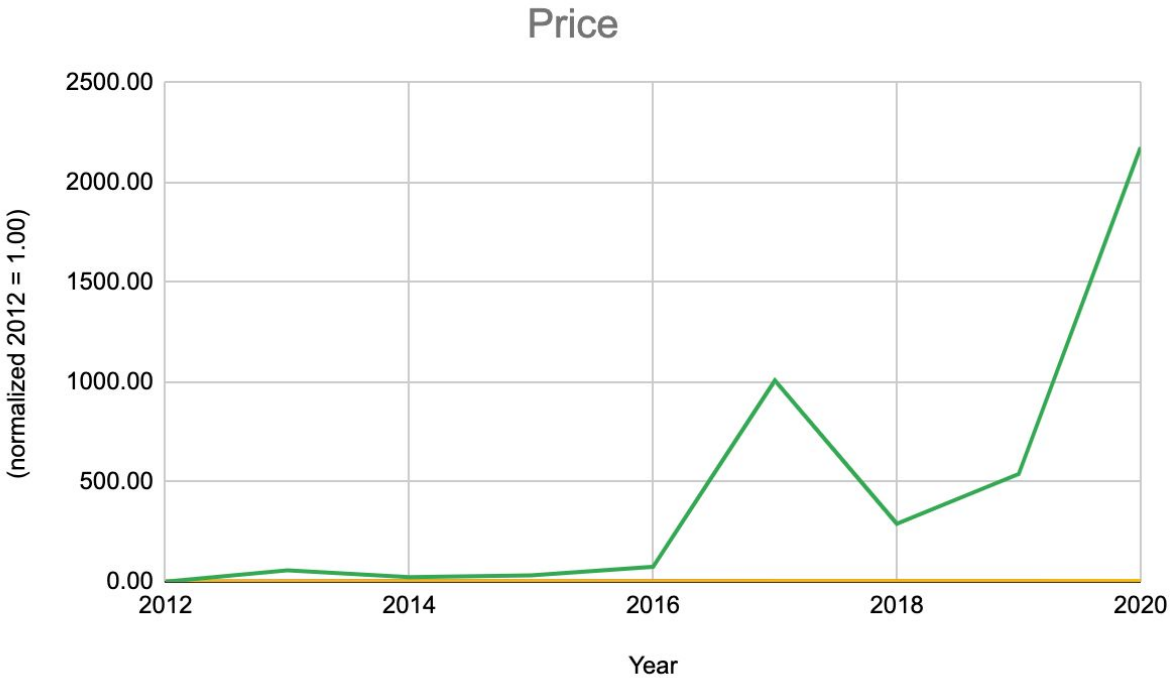
Bitcoin Core

- Changes Merged
- Reviewers
- Review Comments

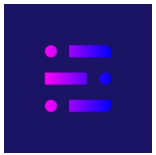


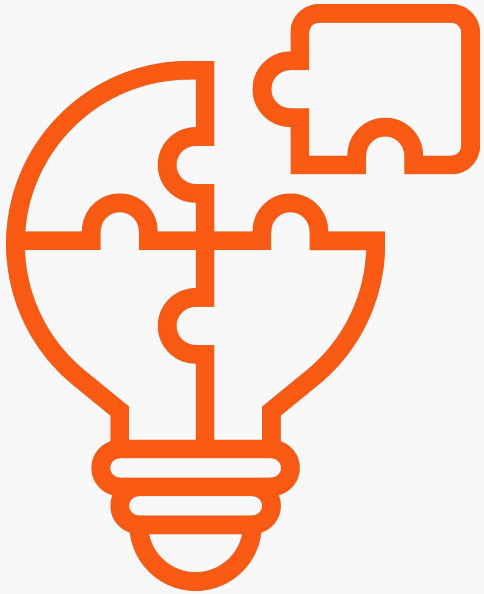
Bitcoin Core

- Changes Merged
- Reviewers
- Review Comments
- Price



Supporting Organizations





Challenges

- Identifying and attracting talent
- Education and mentoring
- Retention and financial support
- Legal challenges

How to Help

Thank you!