Bitcoin Whitepaper

Structure, content and history of Satoshi's original paper

Valerio Vaccaro

Satoshi Spritz Connect

March 17, 2026

Valerio Vaccaro 👀

- 💻 Bitcoin Developer and Hardware Expert
- 🔥 Contributor to Bitcoin open source projects
- ⚠️ DIY hardware enthusiast
- Bitcoin and Liquid Engineer at Blockstream

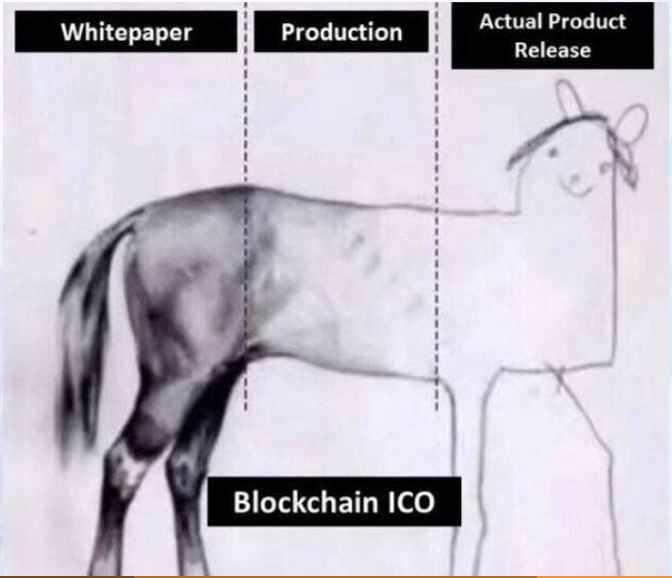
Social

- 🧑‍💼 **LinkedIn** linkedin.com/in/valeriovaccaro
- 🐙 **Github** github.com/valerio-vaccaro
- **Telegram** t.me/valeriovaccaro

License

May this presentation inspire you to become more self-sovereign!

Meme


Whitepaper
Production
Actual Product Release
Blockchain ICO

Outline

- History: Satoshi and Bitcoin's birth
- Whitepaper structure (12 sections)
- Core concepts: transactions, proof-of-work, network
- Incentives, privacy, simplified verification
- References and resources

History: The Whitepaper (2008)

October 31, 2008

- **"Bitcoin: A Peer-to-Peer Electronic Cash System"** published on the cryptography mailing list
- Author: **Satoshi Nakamoto** (pseudonym, identity unknown)
- Solved the **double-spending problem** that plagued previous digital currencies
- Combined: cryptographic hashing, proof-of-work (Hashcash), peer-to-peer networking

Context

- Financial crisis 2008: trust in banks eroded
- Built on prior work: b-money (Wei Dai), Hashcash (Adam Back), timestamping (Haber & Stornetta)

History: Genesis Block (2009)

January 3, 2009

- Satoshi mined **Block #0** (genesis block)
- 50 BTC reward to address `1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa` (unspendable by design)
- Coinbase message: *"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"*

January 9, 2009

- Bitcoin v0.1 released on SourceForge
- bitcoin.org registered

Satoshi and Disappearance

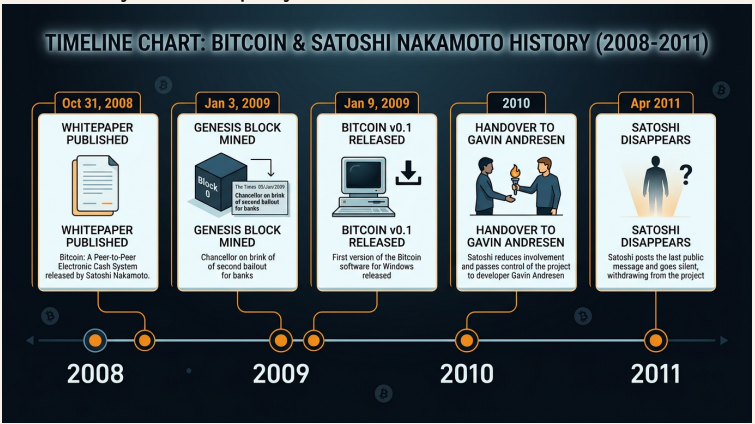
Timeline

- **Oct 2008 – Apr 2011**: Active development and communication
- **Until mid-2010**: Controlled all source code modifications
- **2010**: Transferred control to Gavin Andresen and other developers
- **2011**: Last message: *"I've moved on to other things"* — disappeared

Satoshi and Disappearance

Legacy

- ~1.1M BTC estimated holdings (never moved from early blocks)
- No central authority, no company, no founder tokens



Whitepaper Structure (12 Sections)

1. **Introduction** — Trust-based model problems
2. **Transactions** — Chain of digital signatures
3. **Timestamp Server** — Hash chain for ordering
4. **Proof-of-Work** — CPU voting, consensus
5. **Network** — Steps to run the network
6. **Incentive** — Block reward, transaction fees
7. **Reclaiming Disk Space** — Merkle trees
8. **Simplified Payment Verification** — SPV, light clients
9. **Combining and Splitting Value** — Multiple inputs/outputs
10. **Privacy** — Pseudonymous public keys
11. **Calculations** — Attacker success probability
12. **Conclusion** — Summary of the system

1. Introduction

Problem: Trust-based payments

- Financial institutions as trusted third parties
- **Non-reversible transactions** impossible (mediation costs)
- **Double-spending** requires a trusted party to prevent
- **Fraud** accepted as unavoidable

Solution

- **Cryptographic proof** instead of trust
- Peer-to-peer **distributed timestamp server**
- Secure if honest nodes control majority of CPU power

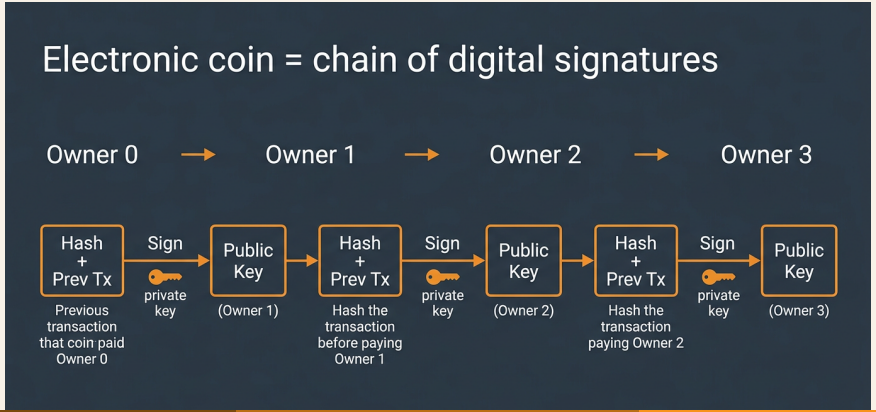
2. Transactions

Electronic coin = chain of digital signatures

- Each owner transfers coin by signing: *hash(previous_tx) + next_owner_public_key*
- Payee verifies signatures to verify chain of ownership

Double-spending problem

- ⚠️ Payee can't verify that previous owners didn't double-spend
- 🔑 **Mint model** (trusted authority): fate of system depends on one entity
- ✅ **Bitcoin**: Transactions publicly announced; **order** agreed by consensus



3. Timestamp Server

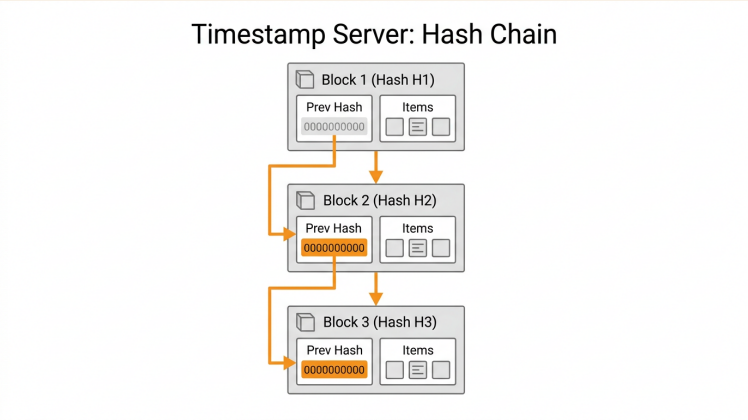
Hash chain

- Hash of block of items $\rightarrow$ timestamp
- Each timestamp **includes previous hash** $\rightarrow$ chain
- Each new timestamp reinforces the ones before it

3. Timestamp Server

From paper

- *"The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash."*



4. Proof-of-Work

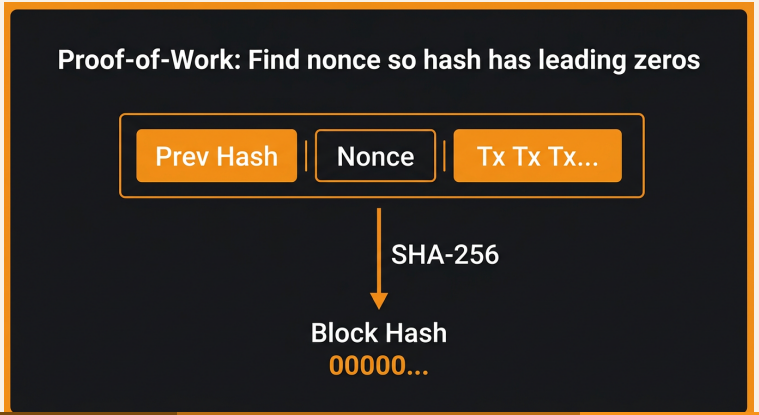
Hashcash-style PoW

- ⚙ Find nonce such that hash(block) has **zero bits** at start
- 🔑 Work exponential in zero bits; verification = single hash
- 🛡 Block cannot be changed without redoing the work

4. Proof-of-Work

Consensus mechanism

- 📈 **One-CPU-one-vote** (not one-IP-one-vote)
- ✅ **Longest chain** = majority decision
- ⚙ **Difficulty adjustment**: moving average, target blocks per hour



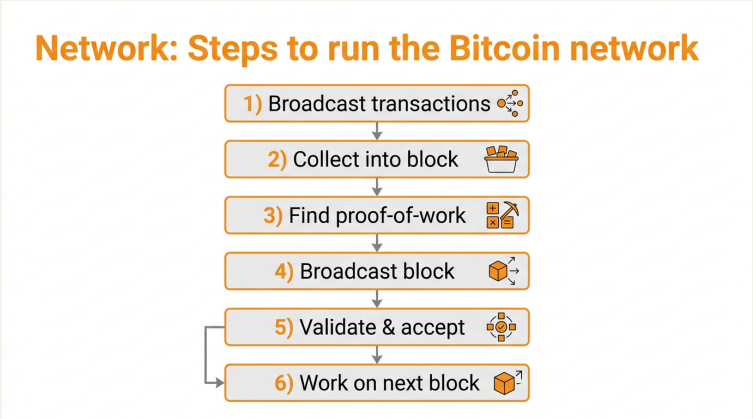
5. Network

Steps

1. New transactions broadcast to all nodes
2. Each node collects transactions into a block
3. Each node works on proof-of-work for its block
4. When found $\rightarrow$ broadcast block
5. Nodes accept if transactions valid and not spent
6. Express acceptance by working on next block (hash of accepted block as prev hash)

Tie-breaking

- 🔗 Two blocks simultaneously? Work on first received, keep other branch
- ✅ Next PoW found → one branch longer → switch to it



6. Incentive

Block reward

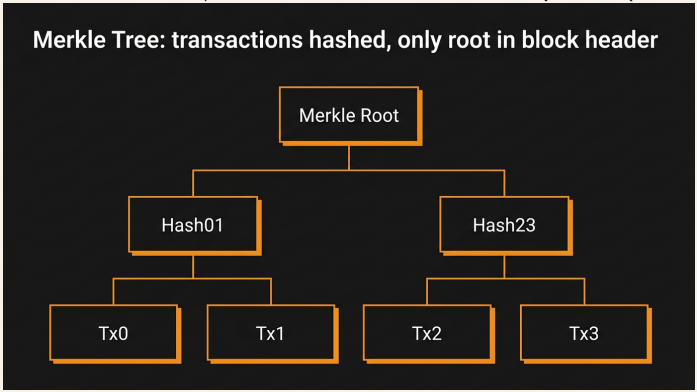
- 🪙 First tx in block = **new coin** to block creator
- ⚙️ Analogous to gold miners expending resources
- 📈 Constant new coins → then transition to **transaction fees only** (inflation-free)

Honesty incentive

- 🛡️ Attacker with more CPU: defraud vs generate new coins
- ✅ More profitable to play by the rules

Merkle tree

- Transactions hashed in **Merkle tree**; only root in block header
- Spent transactions can be pruned (stub branches)
- ~80 bytes/block header $\times$ 6/hr $\times$ 24 $\times$ 365 $\approx$ **4.2 MB/year** (2008 estimate)



8. Simplified Payment Verification

Light clients

- 🔑 Keep only **block headers** of longest chain
- 🔗 Get **Merkle branch** linking tx to block
- ✅ Verify network accepted it; blocks after = further confirmation

Caveat

- ⚠️ Reliable if honest nodes control network; vulnerable if attacker overpowers

9. Combining and Splitting Value

Multiple inputs/outputs

- ⚙ Transactions have **multiple inputs and outputs**
- 🔑 Single input from larger tx, or multiple inputs combining smaller amounts
- ✅ At most two outputs: payment + change

No full history needed

- 🔗 Fan-out (tx depends on many) — no need to extract complete standalone copy

10. Privacy

Pseudonymity

- All transactions **public** — but public keys can be **anonymous**
- Like stock exchange “tape”: time and size public, parties unknown

Best practice

- **New key pair per transaction** to avoid linking
- Multi-input txs reveal inputs owned by same owner

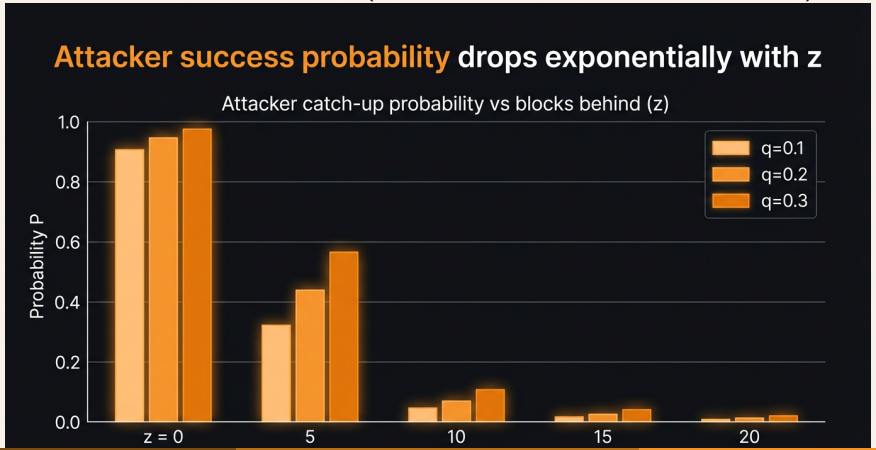
11. Calculations

Attacker catch-up probability

- **Gambler's Ruin**: probability attacker catches up from z blocks behind
- Drops **exponentially** as z increases
- $P < 0.1\%$: q=0.10 $\rightarrow$ z=5; q=0.30 $\rightarrow$ z=24; q=0.40 $\rightarrow$ z=89

Confirmation wait

- ⚙ Recipient waits for tx + z blocks
- 🛡 Sender can't prepare chain ahead (receiver gives pubkey shortly before)



12. Conclusion

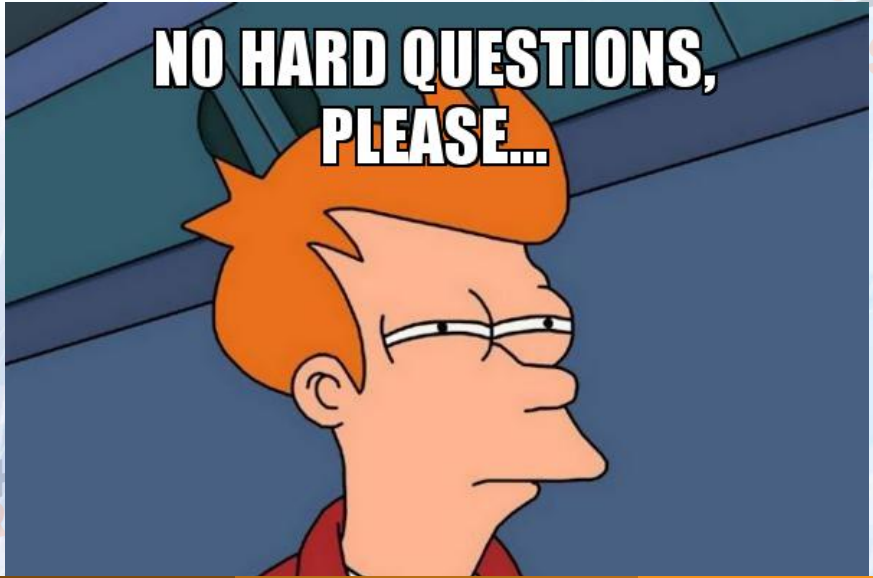
Summary

- Electronic transactions **without trust**
- Coins = digital signatures; PoW prevents double-spending
- Peer-to-peer; nodes vote with CPU; minimal structure
- Leave/rejoin at will; longest chain = proof of history

Design principles

- ✅ Unstructured simplicity
- ✅ No identification required
- ✅ Best-effort message delivery

? Questions



References

Whitepaper

- bitcoin.org/bitcoin.pdf — Original PDF
- bitcoin.org/en/bitcoin-paper — Web version, translations

Prior work cited

- Wei Dai: b-money (1998)
- Adam Back: Hashcash (2002)
- Haber & Stornetta: Timestamping (1991, 1993, 1997)
- Merkle: Merkle trees (1980)

Satoshi Spritz

- 💻 Federation of local Bitcoiner groups
- 🎓 Free, privacy-oriented events
- 🤖 BITCOIN ONLY
- 🔧 Weekly Satoshi Spritz Connect online

Links

- satoshispritz.it
- t.me/SatoshiSpritzConnect

Officine Bitcoin

- 🤝 Italian Bitcoin community, fully free
- 🤖 BITCOIN ONLY
- 🎓 Education and project development

Links

- officinebitcoin.it

Bitcoinissimo

- Bitcoin podcast and statistics
- Episodes in Italian, English, Hungarian, Chinese, Russian, Spanish, French
- Real-time network stats, market data, block explorer

Links

- bitcoinissimo.it