

# Day 3: Cryptographic Foundations Building Trust Without Institutions

Digital Finance Introduction

Prof. Dr. Joerg Osterrieder

Digital Finance

2025

## Morning Sessions

1. **3.1 Cryptographic Building Blocks**  
Hashing, Keys, Digital Signatures
2. **3.2 Blockchain Mechanics**  
Consensus, Blocks, Trilemma

## Afternoon Sessions

3. **3.3 Wallets & Transactions**  
User Experience Gap
4. **3.4 Bitcoin vs. Ethereum**  
Two Design Philosophies



---

**Day Arc: From smallest building blocks to highest-level design philosophy**

# Agenda

- 1 3.1 Cryptographic Building Blocks
- 2 3.2 Blockchain Mechanics
- 3 3.3 Wallets, Transactions & UX Gap
- 4 3.4 Bitcoin vs. Ethereum

## Remember from Day 1:

- Financial systems require **trust** between parties
- Traditional solution: Trusted intermediaries (banks, clearinghouses)
- Digital alternative: Mathematical/cryptographic guarantees

## Today we explore **HOW** cryptography provides trust:

- Hash functions → Data integrity verification
- Digital signatures → Identity and authorization
- Blockchain → Tamper-evident, distributed ledgers

*Note: This material is more technical than Days 1-2. Take your time with the concepts.*

# Key Technical Terms

## Definitions you'll encounter today:

**Turing-complete** A system that can compute anything computable (like a general-purpose computer). Ethereum is Turing-complete; Bitcoin is not.

**Byzantine fault tolerant** A system that works correctly even when some participants are malicious or fail. Named after the “Byzantine Generals Problem.”

**Nonce** “Number used once” – a value that changes to produce different hash outputs.

**Gas** Ethereum's unit for measuring computational work. More complex operations cost more gas.

*Don't worry if these feel abstract now—they'll make more sense as we work through examples.*

## 3.1 Cryptographic Building Blocks

# The Trust Problem in Digital Systems

## Traditional Trust Model

- Banks verify your identity
- Courts enforce contracts
- Governments back currency
- Intermediaries everywhere

Problem: Single points of failure

## Cryptographic Trust Model

- Mathematics verifies identity
- Code enforces agreements
- Network backs value
- Trust is distributed

Solution: Trust through verification

**Key Insight:** Cryptography lets us replace “trust me” with “verify this”

# Three Cryptographic Primitives

## Hash Functions

Digital fingerprints

Integrity

Data hasn't changed

## Public-Key Crypto

Identity with-  
out authority

Identity

You are who you claim

## Digital Signatures

Unforgeable

Non-repudiation

You can't deny it

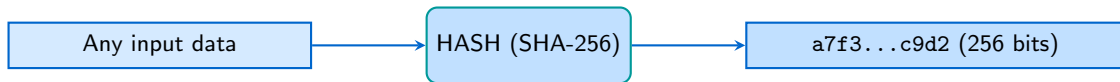
**Focus:** What these tools *guarantee*, not how the math works

---

**These three primitives are the atoms of decentralized trust**

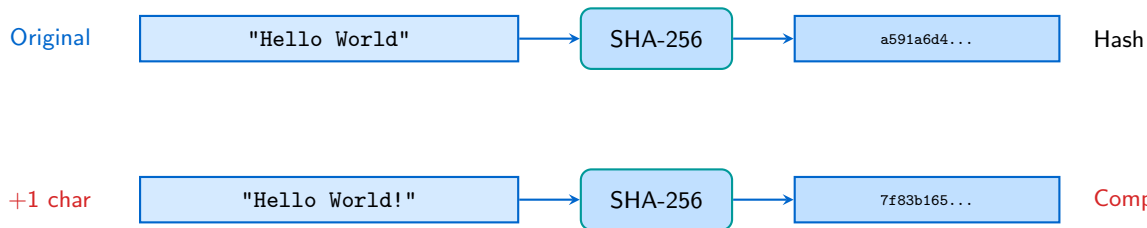


# Hash Functions: Digital Fingerprints



- Deterministic:** Same input → same output, always
- One-way:** Cannot reverse to find input
- Collision-resistant:** Practically impossible to find two inputs with same hash
- Avalanche effect:** Tiny change → completely different output

# Hash Function Visualization: The Avalanche Effect



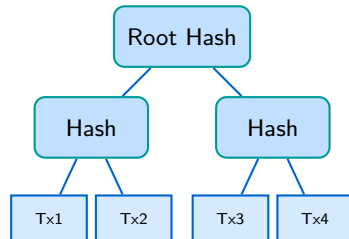
## Why this matters for blockchain:

- Change one transaction → entire block hash changes
- This change cascades through all subsequent blocks
- Tampering becomes immediately detectable

# What Hash Functions Guarantee

## Use Cases in Blockchain

- **Data integrity:** Verify nothing changed
- **Block linking:** Each block contains hash of previous
- **Transaction IDs:** Unique identifier for every transaction
- **Mining puzzles:** Finding hashes with specific properties
- **Merkle trees:** Efficiently verify large data sets



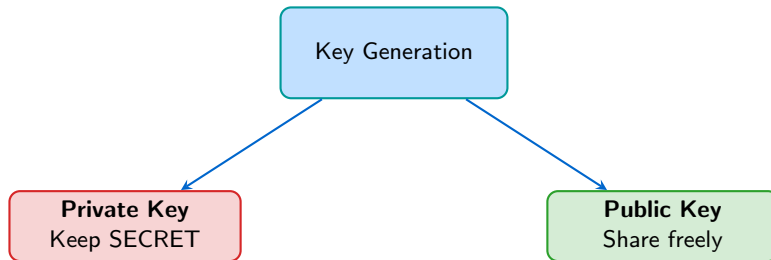
**Merkle Tree**

Verify any transaction with  $O(\log n)$  hashes

## The Guarantee

If two hashes match, the data is identical (with overwhelming probability).

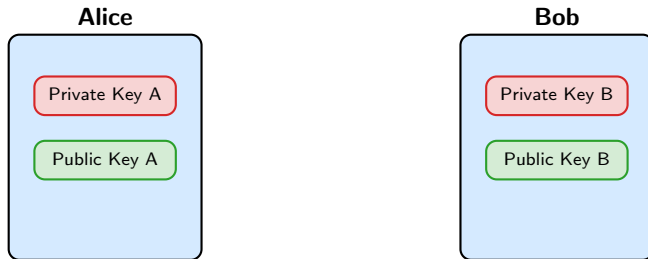
# Public-Key Cryptography: Identity Without Authority



**Mathematical relationship:** Public key is *derived* from private key

Easy: Private  $\rightarrow$  Public      Im-  
possible: Public  $\rightarrow$  Private

# Public-Key Cryptography: How It Works



**To send encrypted message to Bob:**

1. Alice encrypts with Bob's **public key**
2. Only Bob can decrypt with his **private key**

**Key insight:** Only the person with the private key can decrypt

# From Public Key to Blockchain Address



## What You Control

- Private key = your identity
- Whoever has it controls the funds
- Lose it = lose everything
- Share it = share everything

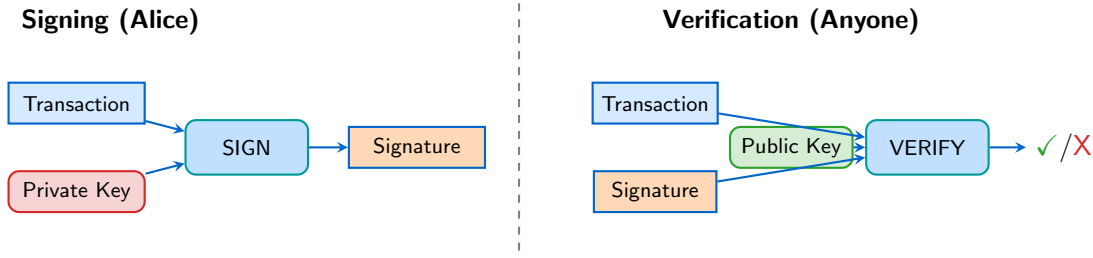
## What You Share

- Address = your “account number”
- Safe to share publicly
- Used to receive payments
- Cannot derive private key from it

---

**“Not your keys, not your coins” – a fundamental principle of crypto**

# Digital Signatures: Unforgeable Proof



## What Digital Signatures Guarantee:

- **Authentication:** Only private key holder could create this signature
- **Integrity:** The message hasn't been altered
- **Non-repudiation:** Signer cannot deny having signed

# Digital Signatures in Blockchain

Every blockchain transaction includes a digital signature proving authorization:

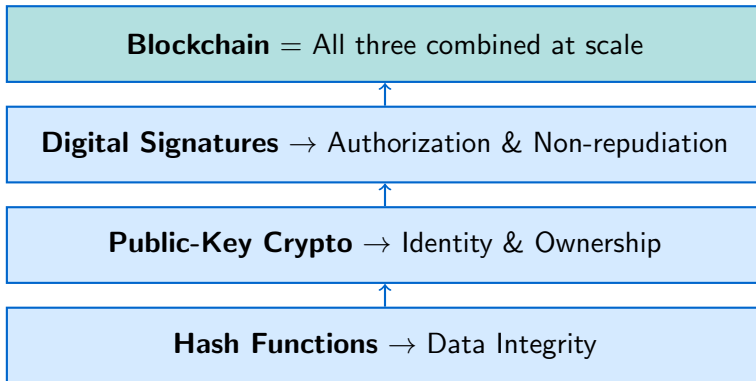
Field	Value
From	0x7a2b...9f3c
To	0x9c4d...e8f2
Value	1.5 ETH
Nonce	42
Gas Limit	21,000
Gas Price	50 gwei
<b>Signature</b>	$v, r, s$ ( <i>proves authorization</i> )

## How it works in practice:

1. Alice creates transaction: "Send 1.5 ETH to Bob"
2. Alice signs with her private key
3. Network verifies signature using Alice's public key



# Summary: The Cryptographic Trust Stack



## Key Takeaway

Cryptography transforms trust from “believe the institution” to “verify the math.” No bank, government, or third party needed – just mathematics.

## 3.2 Blockchain Mechanics

# What Is a Blockchain?

*“A blockchain is a distributed ledger that is append-only, cryptographically linked, and maintained by consensus.”*

## Distributed

- No central server
- Thousands of copies
- No single point of failure

## Append-Only

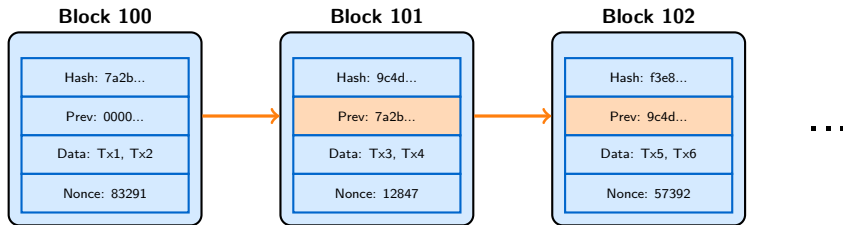
- Can only add data
- Cannot modify history
- Immutable record

## Consensus-Based

- Network agrees on state
- No trusted authority
- Rules enforced by code

**Simple analogy:** A shared Google Doc that everyone can read,  
only append to, and no one can delete from

# Blockchain Structure: Blocks Linked by Hashes



**The chain property:** Each block contains the hash of the previous block

**Why this matters:** Change Block 100 → its hash changes → Block 101's "Prev" becomes invalid → cascading invalidity

---

**This is why blockchain history is considered immutable**

# Why Tampering Is Detectable

Original Chain	Tampered Chain
Block 100: Hash = 7a2b... ↓ (valid link)	Block 100: Hash = x9f2... ↓ (broken link!)
Block 101: Prev = 7a2b... ↓ (valid link)	Block 101: Prev = 7a2b... X ↓ (broken link!)
Block 102: Prev = 9c4d...	Block 102: Prev = 9c4d... X
✓ Valid	X Invalid

## To tamper with history, attacker must:

1. Change the target block's data
2. Recalculate that block's hash
3. Recalculate *every subsequent block's hash*
4. Do this faster than the honest network adds new blocks

# The Consensus Problem

## The Challenge

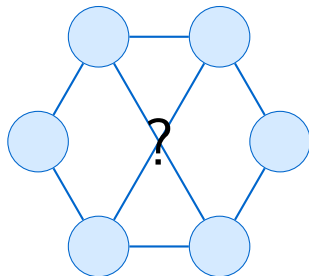
In a decentralized network:

- No central authority
- Nodes don't trust each other
- Messages can be delayed or lost
- Some nodes may be malicious

## The Question

How do thousands of strangers agree on:

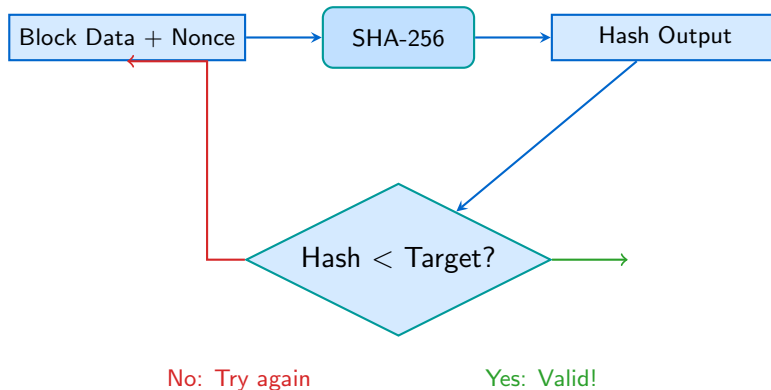
- Which transactions are valid?
- What order do they go in?
- What is the “true” history?



Who decides the truth?

## Consensus Mechanism

# Proof of Work (PoW): Bitcoin's Answer



## The “work” in Proof of Work:

- Find a nonce that makes the block hash start with many zeros
- Requires billions of guesses (brute force)
- Easy to verify, just hash once and check

# Proof of Work: Security Through Energy

## Why It Works

- Attack requires controlling 51% of computing power
- This costs billions in hardware + electricity
- Rational: mining honestly is more profitable
- Economic security: attack cost  $>$  potential gain

## Advantages

- Battle-tested (Bitcoin since 2009)
- Highly secure
- Truly decentralized

PoW trades energy for security – the “work” is the cost of attack

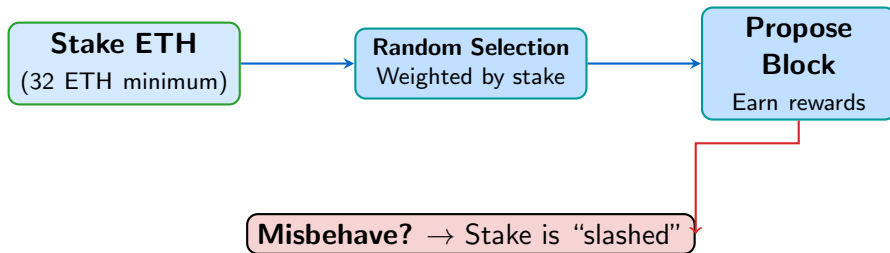
## Criticisms

- **Enormous energy consumption**
- Bitcoin uses more electricity than some countries
- Environmental concerns
- Hardware arms race (specialized ASICs)

Bitcoin Network:  
 $\approx 150$  TWh/year  
(more than Argentina)



# Proof of Stake (PoS): Ethereum's Answer



## The “stake” in Proof of Stake:

- Lock up cryptocurrency as collateral
- More stake = higher chance to be selected as validator
- Honest behavior = earn rewards
- Malicious behavior = lose your stake (“slashing”)

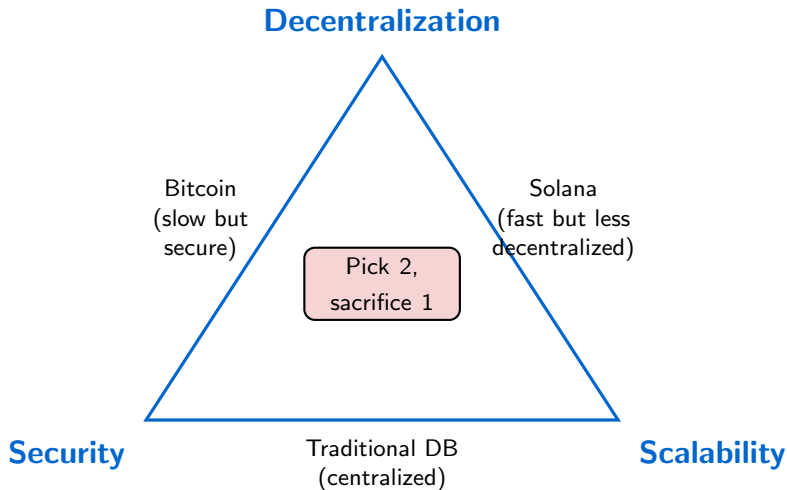
# PoW vs. PoS: Head-to-Head Comparison

Aspect	Proof of Work	Proof of Stake
Security basis	Computational power	Economic stake
Energy usage	Very high	Low (~99.9% less)
Hardware required	Specialized (ASICs)	Standard computers
Entry barrier	High (equipment cost)	High (32 ETH)
Attack cost	Hardware + electricity	Acquire 51% of stake
Decentralization	Mining pool concentration	Wealth concentration
Finality	Probabilistic	Faster, more definite
Examples	Bitcoin, Dogecoin	Ethereum, Cardano, Solana

**Key insight:** Neither is “better” – they make different tradeoffs

Ethereum transitioned from PoW to PoS in Sept 2022 (“The Merge”)

# The Blockchain Trilemma



# The Trilemma Explained

## Decentralization

- Many independent validators
- No single point of control
- Censorship resistant
- Geographic distribution

Tradeoff: More nodes = slower

## Security

- Resistant to attacks
- Immutable history
- Byzantine fault tolerant
- Economic finality

Tradeoff: More verification = slower

## Scalability

- High throughput (TPS)
- Low latency
- Low fees
- Handle demand spikes

Tradeoff: Speed often requires centralization

System	TPS	Finality	Validators
Bitcoin	7	60 min	~15,000 nodes
Ethereum	15-30	12-15 min	~500,000 validators
Solana	65,000	0.4 sec	~1,900 validators

# Scaling Solutions: Breaking the Trilemma?

## Layer 1 Solutions

*Improve the base blockchain*

- Larger blocks (more data per block)
- Faster block times
- More efficient consensus
- Sharding (parallel processing)

### Examples:

- Ethereum 2.0 (sharding planned)
- Solana (parallel processing)
- Near Protocol (sharding)

## Layer 2 Solutions

*Build on top of base layer*

- Process transactions off-chain
- Settle on main chain periodically
- Inherit security from L1
- Much higher throughput

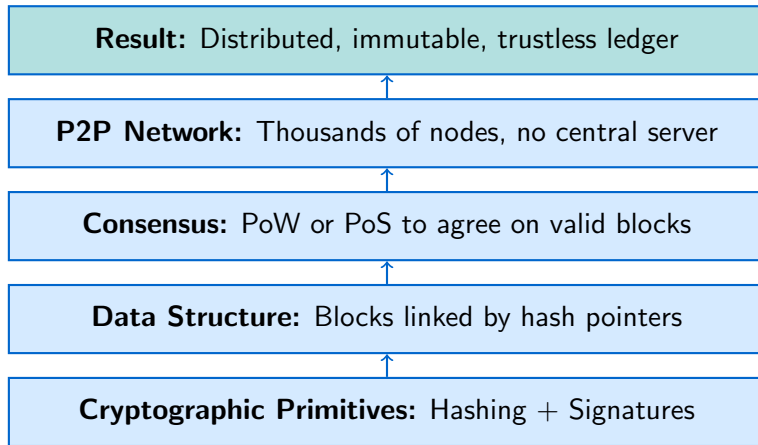
### Examples:

- Lightning Network (Bitcoin)
- Optimism, Arbitrum (Ethereum)
- Polygon (Ethereum)

## Key Insight

Layer 2 solutions let blockchains scale WITHOUT sacrificing decentralization or security at the base layer.

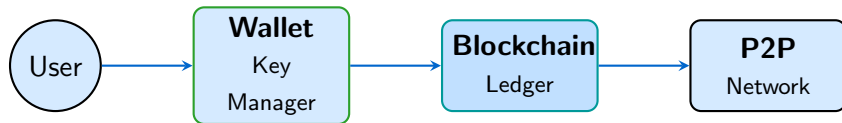
## Summary: How Blockchains Work



**The fundamental tradeoff:** Blockchains sacrifice efficiency for trust minimization. When you need trustless coordination among strangers, this tradeoff is worth it.

## 3.3 Wallets, Transactions & UX Gap

# From Theory to Practice: How Users Interact



**Key insight:** Users never interact with the blockchain directly.

The **wallet** is the interface – it manages keys, signs transactions, and broadcasts them to the network.

## Common Misconception

Wallets don't "store" cryptocurrency. The blockchain stores balances.  
Wallets store *keys* that prove you can spend those balances.



# What Is a Wallet, Really?

## A wallet is a key management tool

It does three things:

1. **Stores private keys** (securely, hopefully)
2. **Signs transactions** using those keys
3. **Broadcasts transactions** to the network

## It does NOT:

- Store cryptocurrency
- Connect you to a bank
- Know your identity
- Require permission to create

### Wallet Contents

Private Key 1

0x7a2b...secret

Address 1: 0x9c4d...

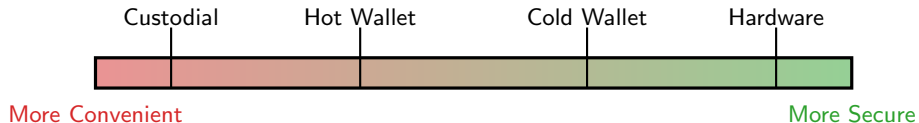
Address 2: 0xf3e8...

Balance: 1.5 ETH

(queried from blockchain)

# Types of Wallets: A Spectrum of Tradeoffs

## Security vs. Convenience Tradeoff



Type	Key Control	Example	Risk
Custodial	Exchange holds keys	Coinbase, Binance	Exchange hack
Hot Wallet	You hold, online	MetaMask	Malware, phishing
Cold Wallet	You hold, offline	Paper wallet	Physical loss
Hardware	You hold, secure chip	Ledger, Trezor	Device failure

# Custodial vs. Non-Custodial: Who Holds the Keys?

## Custodial (Exchange) Wallet

You → Exchange → Blockchain

- Easy to use
- Password recovery possible
- Fiat on/off ramps
- You don't control keys
- Exchange can freeze funds
- Exchange can be hacked

## Non-Custodial (Self-Custody)

You → Your Wallet → Blockchain

- Full control
- No counterparty risk
- Censorship resistant
- You are responsible
- Lose keys = lose funds
- No customer support

**“Not your keys, not your coins”**

FTX collapse (2022): \$8B in customer funds lost

# Anatomy of a Blockchain Transaction

## Ethereum Transaction Structure

Field	Value	Purpose
From	0x7a2b...9f3c	Sender's address
To	0x9c4d...e8f2	Recipient's address
Value	1.5 ETH	Amount to transfer
Nonce	42	Tx count (prevents replay)
Gas Limit	21,000	Max computation units
Gas Price	50 gwei	Price per gas unit
<b>Signature</b>	v, r, s	Proves authorization

The signature covers ALL fields above it – proving the sender authorized this exact transaction.

# Understanding Gas: The Cost of Computation

## What is Gas?

- Unit measuring computational work
- Each operation has a gas cost
- Simple transfer: 21,000 gas
- Smart contract call: varies (100K+)

## Gas Economics

$$\text{Fee} = \text{Gas Used} \times \text{Gas Price}$$

### Example:

- Gas used: 21,000
- Gas price: 50 gwei
- $\text{Fee} = 21,000 \times 50 \text{ gwei} = 0.00105 \text{ ETH}$

**Gas prices fluctuate wildly – from cents to hundreds of dollars per transaction**

## Why Gas Exists

1. Prevents infinite loops
2. Compensates validators
3. Prioritizes transactions
4. Allocates scarce block space

### Network Congestion

High demand →  
Higher gas prices →  
Higher fees

(Supply and demand)

## What crypto asks users to understand:

### Traditional Banking

- Username and password
- “Forgot password?” link
- Customer support
- FDIC insurance
- Fraud protection
- Clear error messages

Forgiving of mistakes

### Cryptocurrency

- 24-word seed phrase
- Lose phrase = lose everything
- No customer support
- No insurance (usually)
- Irreversible transactions
- Cryptic error: “insufficient gas”

Unforgiving of mistakes

# Common User Errors (And Their Consequences)

Error	Consequence	Recovery?
Lost seed phrase	Permanent loss of funds	No
Sent to wrong address	Funds gone forever	No
Wrong network (ETH to BSC)	Funds stuck/lost	Sometimes
Insufficient gas	Transaction fails	Yes, retry
Approved malicious contract	Wallet drained	No
Phishing attack	Keys stolen	No
Didn't verify contract	Funds stolen	No

## Scale of the problem:

- \$3.8 billion lost to crypto hacks in 2022 (Chainalysis)
- Estimated 20% of Bitcoin permanently inaccessible (lost keys)
- Phishing remains the #1 attack vector

# Improving Crypto UX: Current Efforts

## Technical Solutions

- **Account abstraction:** Smart contract wallets with recovery
- **Social recovery:** Trusted friends can help recover
- **Multi-sig:** Require multiple keys to transact
- **Hardware wallets:** Secure key storage
- **ENS names:** Human-readable addresses

## UX Improvements

- **Transaction simulation:** Show what will happen before signing
- **Clear warnings:** “This will drain your wallet”
- **Gas estimation:** Show fees in dollars
- **Address books:** Saved, verified addresses
- **Better error messages:** Human-readable explanations

## The Goal

Make self-custody as easy as using a bank app, without sacrificing security or decentralization.



## Decision tree for choosing a wallet:

### 1. Who holds the keys?

- Exchange holds them → **Custodial** (easy, but risky)
- You hold them → Continue to step 2

### 2. Is it connected to the internet?

- Yes → **Hot wallet** (MetaMask) – convenient for daily use
- No → **Cold/Hardware wallet** (Ledger) – secure for savings

**Key Takeaway:** Wallet choice = tradeoff between convenience and security.  
Most users need *both*: hot wallet for daily use, cold storage for savings.

## 3.4 Bitcoin vs. Ethereum

## Bitcoin

*"Digital Gold"*

- Store of value
- Fixed supply (21M)
- Minimal, robust
- Conservative changes
- "Don't break what works"

**Goal:** Sound money that  
no one can inflate or censor

## Ethereum

*"World Computer"*

- Platform for applications
- Programmable money
- Feature-rich, flexible
- Active development
- "Move fast, iterate"

**Goal:** Decentralized platform  
for any application

Year	Event
2009	Bitcoin launches
2012	Colored coins on Bitcoin (limited tokens)
2013	Ethereum whitepaper (Vitalik Buterin)
2015	Ethereum launches
2022	Ethereum transitions to Proof of Stake

## Vitalik Buterin's insight (2013):

Bitcoin's scripting language was too limited. He proposed a blockchain with a **Turing-complete** programming language – one that could run any program, not just simple transactions.

## The Key Difference

Bitcoin: “Is this a valid payment?” (simple yes/no)

Ethereum: “Run this arbitrary code and tell me the result” (general computation)

## Technical Comparison: Head to Head

Feature	Bitcoin	Ethereum
Launch	2009	2015
Consensus	Proof of Work	Proof of Stake (since 2022)
Block time	~10 minutes	~12 seconds
Supply cap	21 million BTC	No hard cap
Issuance	Halving every 4 years	Dynamic (can be deflationary)
Scripting	Limited (Bitcoin Script)	Turing-complete (EVM)
Primary use	Value transfer, store of value	Smart contracts, DeFi, NFTs
TPS	~7	~15-30

**Key insight:** These aren't competing for the same use case.

Bitcoin optimizes for *security and immutability*.

Ethereum optimizes for *programmability and flexibility*.

## Core Properties

- **Fixed supply:** 21 million, ever
- **Predictable issuance:** Halving every 210,000 blocks
- **Decentralized:** No one controls it
- **Censorship resistant:** Anyone can transact
- **Immutable:** Rules don't change

## The Narrative

- “Digital gold” – scarce, durable
- Hedge against inflation
- Separation of money and state
- Base layer for financial system

---

**Bitcoin maximalists:** “We already have one neutral, global money. Why do we need more?”

## Bitcoin Supply Curve

$(- (x - 2009)/4)); [da$   
Asymptotically approaches 21M

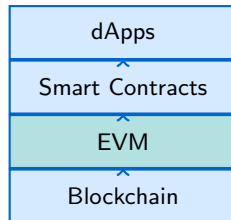
## Core Properties

- **Smart contracts:** Self-executing code
- **EVM:** Ethereum Virtual Machine
- **Programmable:** Any logic possible
- **Composable:** Contracts call contracts
- **Tokens:** Create new assets easily

## The Narrative

- Platform for decentralized apps
- “DeFi” – finance without banks
- NFTs, DAOs, and more
- Base layer for Web3

## Ethereum Stack



Apps built on contracts on EVM

---

Ethereum enables “programmable money” – money with built-in rules

# What Are Smart Contracts?

## Traditional Contract

1. Legal document
2. Human interpretation
3. Court enforcement
4. (Maybe) execution

Requires: lawyers, courts, trust

## Smart Contract

1. Code on blockchain
2. Automatic execution
3. No intermediaries
4. Guaranteed outcome

Requires: only the code

### Definition

A smart contract is code stored on a blockchain that automatically executes when predetermined conditions are met.

**Key property:** Once deployed, the code cannot be changed. “Code is law.”



# Smart Contract Example: Escrow

## Traditional Escrow

1. Buyer sends money to escrow agent
2. Seller ships goods
3. Buyer confirms receipt
4. Escrow agent releases funds to seller

## Problems:

- Trust the escrow agent
- Agent takes a fee
- Disputes are slow
- Counterparty risk

## Smart Contract Escrow

1. Buyer sends ETH to contract
2. Contract holds funds
3. Buyer calls `confirmReceipt()`
4. Contract automatically sends to seller

## Advantages:

- Trust the code (auditable)
- Minimal fees (just gas)
- Instant execution
- No counterparty risk

**Key insight:** Smart contracts replace trusted intermediaries with verified code.

# What “Programmable Money” Enables

## DeFi

- Lending/borrowing
- Decentralized exchanges
- Yield farming
- Derivatives

## NFTs/Tokens

- Digital ownership
- Tokenized assets
- Loyalty programs
- Gaming items

## DAOs

- Decentralized governance
- Treasury management
- Collective ownership
- Voting systems

All built on Ethereum's programmable foundation

## The Power of Composability

Smart contracts can call other smart contracts. This means DeFi protocols can be combined like Lego blocks – creating entirely new financial products.

# The Tradeoff: Flexibility vs. Security

## Bitcoin's Approach

- Limited scripting = fewer bugs
- Simple = easier to secure
- Ossification as a feature
- “If it ain’t broke...”

## Security record:

- Never been hacked
- No major protocol bugs
- Predictable behavior
- 15+ years of uptime

## Ethereum's Approach

- Full programmability = more power
- Complex = more attack surface
- Continuous improvement
- “Move fast, fix things”

## Security record:

- Many contract hacks
- The DAO hack (2016): \$60M
- Ongoing exploits in DeFi
- “Code is law” cuts both ways

## The Fundamental Tradeoff

More programmability = more capability = more things that can go wrong

Chain	Programmability	Market Cap	Niche
Bitcoin	Low	#1	Store of value
Ethereum	High	#2	Smart contract platform
Solana	High	#5	High-speed DeFi
Cardano	High	#9	Academic approach

## Different chains, different niches:

- Bitcoin: Store of value, “digital gold”, settlement layer
- Ethereum: Smart contract platform, DeFi hub
- Other L1s: Compete on speed, cost, specific use cases

**Key insight:** It's not “which chain wins” – it's “which chain for which use case”

# Discussion Questions

1. **Store of value vs. programmability:** Can Ethereum also be “sound money”? Can Bitcoin add smart contracts?
2. **The DAO hack:** Ethereum’s response (hard fork to reverse the hack) violated “code is law.” Was this the right call?
3. **Energy debate:** Bitcoin’s PoW energy use is criticized, but it’s also what makes it secure. Ethereum moved to PoS – did it sacrifice anything?
4. **Maximalism:** Many believe only one blockchain will “win.” Others see room for many. What’s your view?
5. **Regulation:** How might different design philosophies affect regulatory treatment? Is ETH a security?

# Summary: Two Philosophies

## Bitcoin

- **Philosophy:** Minimalism
- **Goal:** Sound money
- **Priority:** Security, decentralization
- **Approach:** Conservative
- **Scripting:** Limited by design
- **Consensus:** Proof of Work
- **Supply:** Fixed at 21M

## Ethereum

- **Philosophy:** Generality
- **Goal:** World computer
- **Priority:** Programmability
- **Approach:** Progressive
- **Scripting:** Turing-complete
- **Consensus:** Proof of Stake
- **Supply:** Dynamic

**Not competitors – different tools for different jobs**

Understanding both is essential for Digital Finance

## Day 3: Key Takeaways

1. **Cryptographic primitives** (hashing, public-key crypto, signatures) are the building blocks of trustless systems
2. **Blockchains** combine these primitives with consensus mechanisms to create distributed, immutable ledgers
3. **The trilemma** (decentralization, security, scalability) forces every blockchain to make tradeoffs
4. **Wallets** are key managers, not coin containers – “not your keys, not your coins”
5. **UX remains a major barrier** to mainstream crypto adoption
6. **Bitcoin and Ethereum** represent fundamentally different visions: sound money vs. programmable platform

---

**Tomorrow: Smart contracts, DeFi protocols, and tokenization (Days 4-5 bridge)**

# Hands-On Labs for Day 3

Notebook	Topic	You Will...
NB05	Cryptographic Primitives	Hash data, generate keys, sign & verify
NB06	Mini-Blockchain	Build blocks, link them, simulate tampering
NB07	Transactions	Construct, examine, and analyze tx on testnet

## Lab Goals:

- Make abstract cryptography tangible through hands-on coding
- See blockchain data structures in action
- Experience the UX challenges firsthand

**Remember:** Focus on WHAT these tools guarantee,  
not HOW the underlying math works



# Programmable Finance

*Smart Contracts, DeFi, and Tokenization*

- How smart contracts enable DeFi protocols
- Lending, borrowing, and trading without intermediaries
- Tokenization: bridging traditional and crypto finance
- Real-world asset tokenization
- Risks and opportunities in DeFi

Day 3: Foundations



Day 4: Applications



Day 5: Integration