

Lesson 15: Public Key Cryptography & Digital Signatures

Module 2: Blockchain Fundamentals

Digital Finance

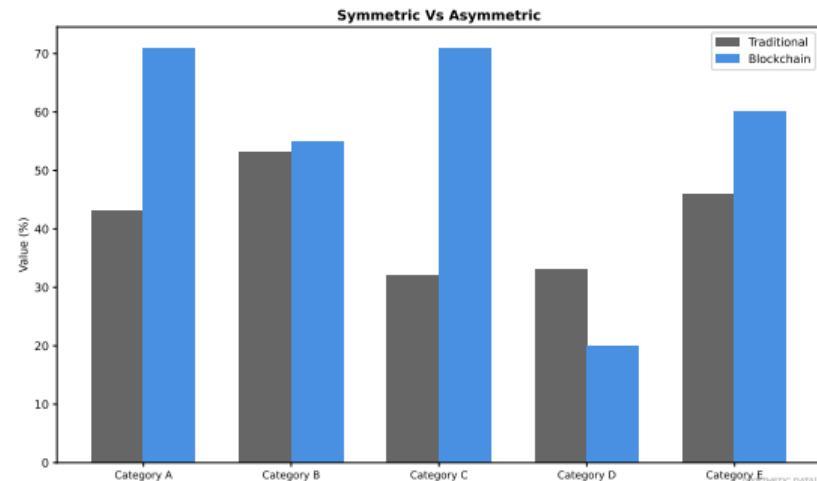
The Problem: Secure Communication Over Insecure Channels

Challenge:

- How do two parties communicate securely without meeting?
- How do you verify someone's identity online?
- How do you prove authorship of a message?

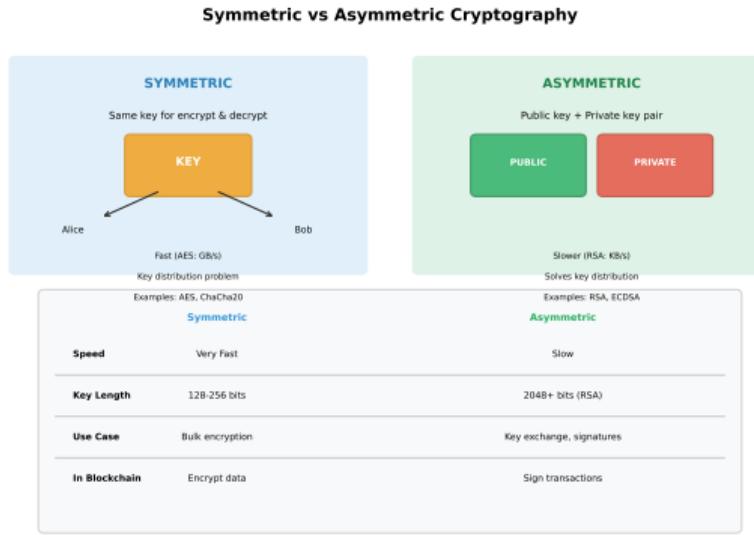
Traditional Solution:

- Symmetric cryptography (shared secret)
- Problem: Key distribution
- Requires secure channel to share key



Key concepts from this slide inform practical applications in finance.

Symmetric vs Asymmetric Cryptography



- **Symmetric:** Same key for encryption and decryption (AES, DES)
 - **Asymmetric:** Key pair – public key encrypts, private key decrypts
 - **Blockchain Use:** Asymmetric for identity, symmetric for bulk data

Comparative analysis helps identify the right tool for specific requirements.

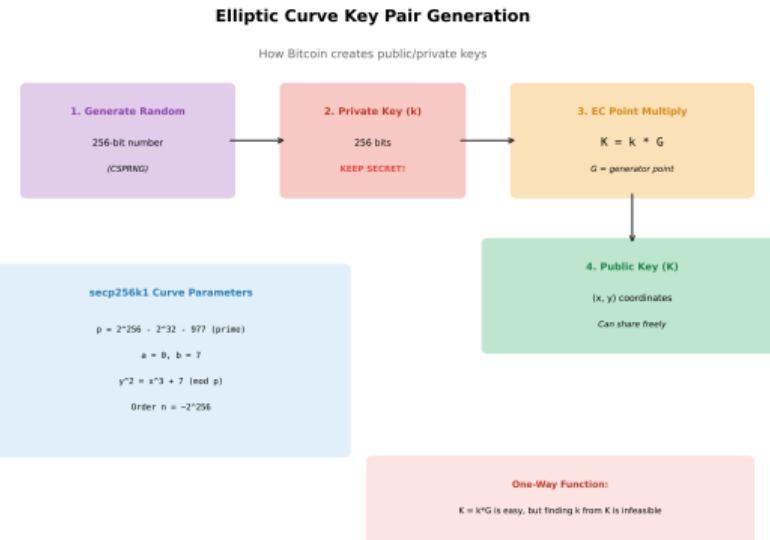
Public Key Cryptography: Revolutionary Idea

Key Pair Structure:

- **Public Key:** Shared openly
- **Private Key:** Kept secret
- Mathematical relationship
- One-way function (easy to compute, hard to reverse)

Properties:

- Encrypt with public → decrypt with private
- Sign with private → verify with public
- Cannot derive private from public



Understanding history helps predict future developments in the technology.

Mathematical Foundation: Trapdoor Functions

One-Way Function:

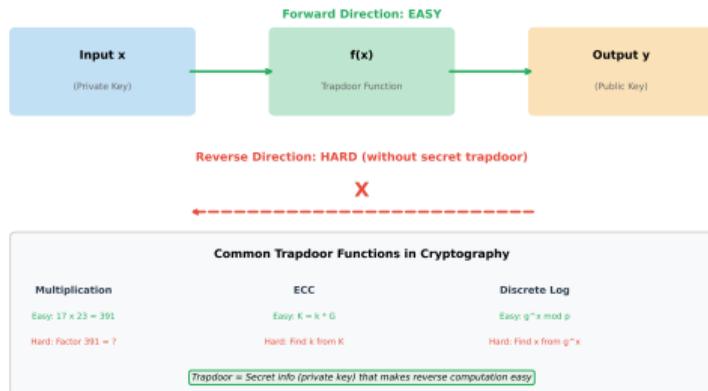
$$y = f(x) \quad (\text{easy})$$

$$x = f^{-1}(y) \quad (\text{hard})$$

Examples:

- Factoring large primes (RSA)
- Discrete logarithm (Diffie-Hellman)
- Elliptic curve discrete log (ECDSA)

Trapdoor Functions: The Foundation of Public Key Crypto



Source: Diffie & Hellman, "New Directions in Cryptography" (1976)

Trapdoor: Secret information (private key) makes inverse easy

Key concepts from this slide inform practical applications in finance.

RSA Cryptography (Classic Approach)

Key Generation:

- ① Choose two large primes: p, q
- ② Compute $n = p \times q$ (modulus)
- ③ Compute $\phi(n) = (p - 1)(q - 1)$
- ④ Choose public exponent e (commonly 65537)
- ⑤ Compute private exponent $d \equiv e^{-1} \pmod{\phi(n)}$

Encryption/Decryption:

$$\text{Ciphertext: } c = m^e \pmod{n} \quad | \quad \text{Plaintext: } m = c^d \pmod{n}$$

Example: $p = 61, q = 53, n = 3233, e = 17, d = 2753$

- Message $m = 123$: $c = 123^{17} \pmod{3233} = 855$
- Decrypt: $m = 855^{2753} \pmod{3233} = 123$

Cryptographic primitives provide the security foundation for blockchain systems.

Elliptic Curve Cryptography (ECC)

Why ECC?

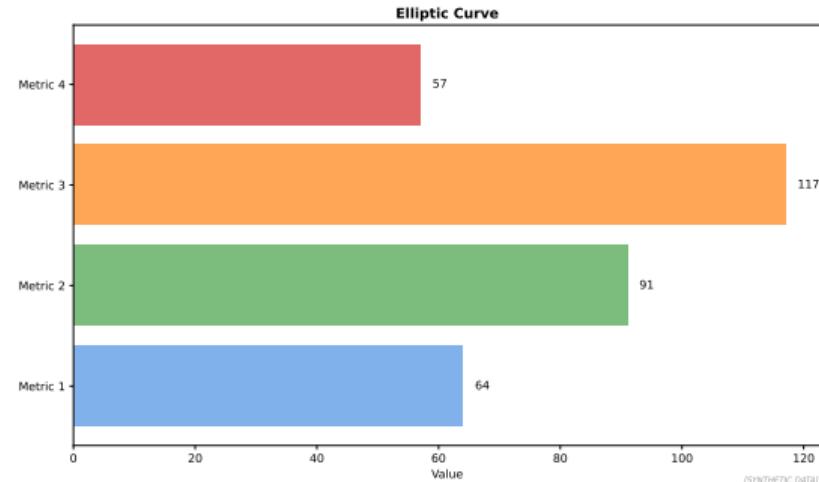
- Smaller key sizes (256-bit ECC \approx 3072-bit RSA)
- Faster computations
- Lower bandwidth
- Standard in Bitcoin/Ethereum

Curve Equation:

$$y^2 = x^3 + ax + b$$

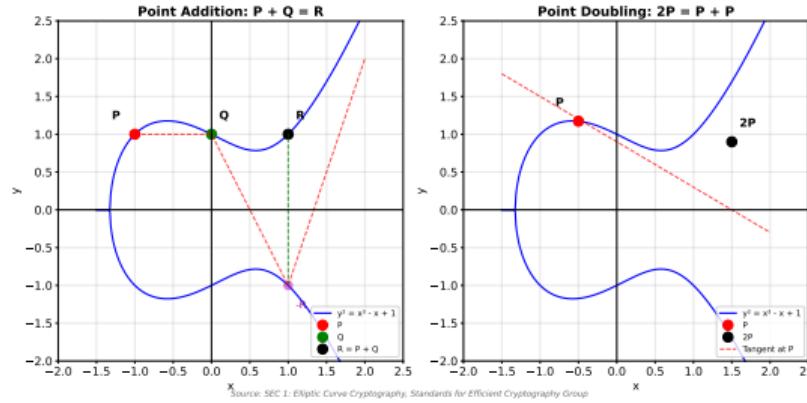
Bitcoin uses: secp256k1

$$y^2 = x^3 + 7$$



Cryptographic primitives provide the security foundation for blockchain systems.

ECC Point Addition: The Core Operation



Operations:

- **Point Addition:** $P + Q = R$ (draw line through P and Q , reflect third intersection)
- **Point Doubling:** $P + P = 2P$ (tangent line at P)
- **Scalar Multiplication:** $nP = P + P + \dots + P$ (n times)

Key concepts from this slide inform practical applications in finance.

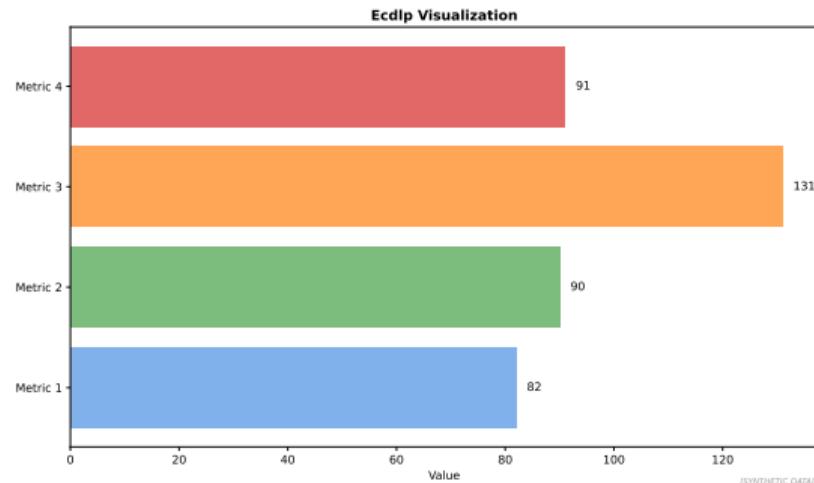
ECC Security: Discrete Logarithm Problem

Easy Problem:

- Given point G and scalar k
- Compute $P = k \cdot G$
- Fast using double-and-add

Hard Problem (ECDLP):

- Given points G and P
- Find scalar k such that $P = k \cdot G$
- No efficient algorithm known
- This is the **private key**



Security: Best attack takes $\mathcal{O}(\sqrt{n})$ operations for n -bit key

Security analysis identifies vulnerabilities and helps design robust systems.

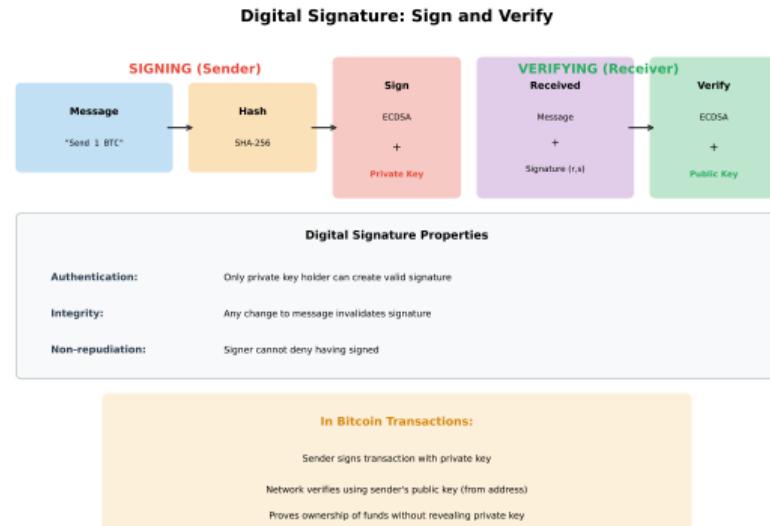
Digital Signatures: Proving Authorship

Purpose:

- Prove message was created by you
- Ensure message wasn't altered
- Non-repudiation (can't deny)

Process:

- ① Hash the message
- ② Encrypt hash with private key
- ③ Attach signature to message
- ④ Verify: Decrypt with public key, compare hash



Key concepts from this slide inform practical applications in finance.

Signature Generation:

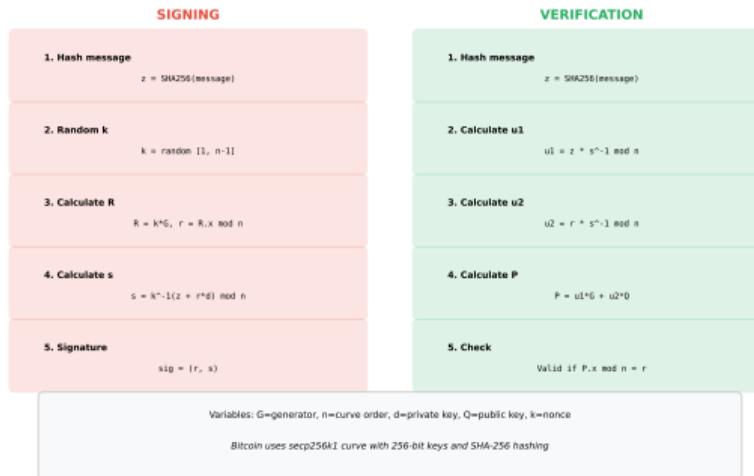
- ① Message m , private key d , public key $Q = d \cdot G$
- ② Hash message: $z = \text{hash}(m)$
- ③ Choose random k , compute $R = k \cdot G = (x_R, y_R)$
- ④ Compute $r = x_R \bmod n$
- ⑤ Compute $s = k^{-1}(z + rd) \bmod n$
- ⑥ Signature: (r, s)

Signature Verification:

- ① Compute $w = s^{-1} \bmod n$
- ② Compute $u_1 = zw \bmod n$, $u_2 = rw \bmod n$
- ③ Compute $P = u_1 \cdot G + u_2 \cdot Q$
- ④ Valid if $x_P \equiv r \pmod{n}$

Key concepts from this slide inform practical applications in finance.

ECDSA: Elliptic Curve Digital Signature Algorithm



Key Properties:

- Signature size: 64 bytes (256-bit r + 256-bit s)
- Cannot forge without private key
- Each signature requires unique random k (reuse breaks security!)

Key concepts from this slide inform practical applications in finance.

Cryptocurrency Wallets: Key Management

Wallet Components:

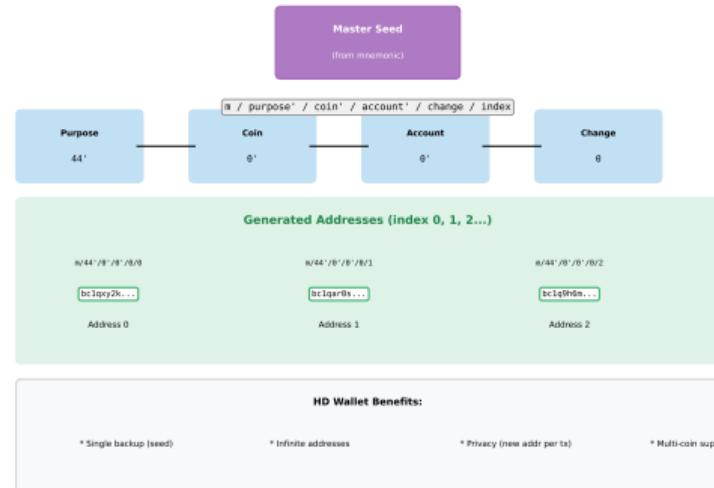
- Private key (spend authority)
- Public key (derived from private)
- Address (hash of public key)

Key Derivation:

Private Key $\xrightarrow{\text{ECC}}$ Public Key $\xrightarrow{\text{Hash}}$ Address

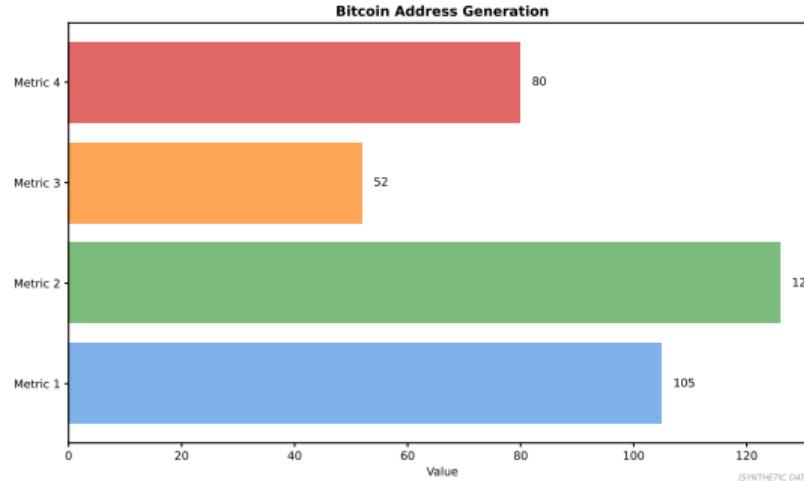
One-way: Cannot derive private from address

BIP-32/44: Hierarchical Deterministic Wallet



Cryptographic primitives provide the security foundation for blockchain systems.

Bitcoin Address Generation



Steps:

- ① Generate 256-bit private key (random number)
- ② Compute public key: $\text{PubKey} = \text{PrivKey} \times G$ (secp256k1)
- ③ Hash public key: SHA256, then RIPEMD160
- ④ Add version byte, compute checksum
- ⑤ Base58 encode → Address (e.g., 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa)

Bitcoin remains the largest cryptocurrency by market cap and network security.

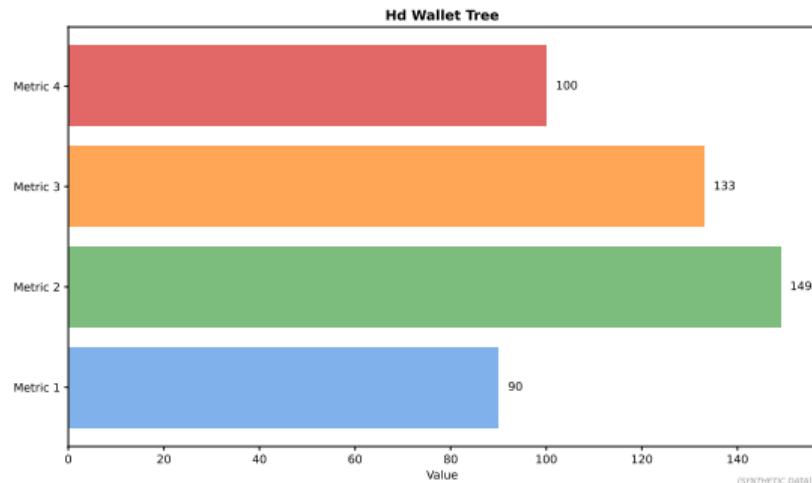
Hierarchical Deterministic (HD) Wallets

Problem:

- Managing multiple random keys
- Backup complexity
- Privacy (address reuse)

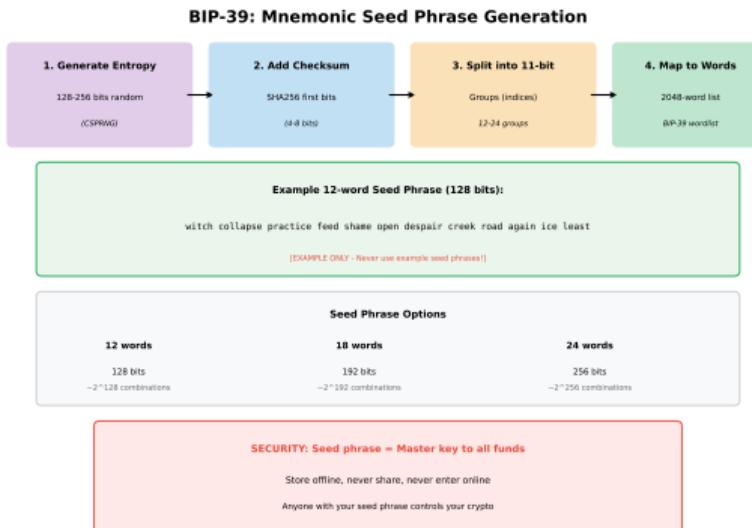
Solution (BIP-32/44):

- Master seed (12/24 words)
- Derive infinite keys deterministically
- Hierarchical tree structure
- One backup for all keys



Key concepts from this slide inform practical applications in finance.

Mnemonic Seed Phrases (BIP-39)



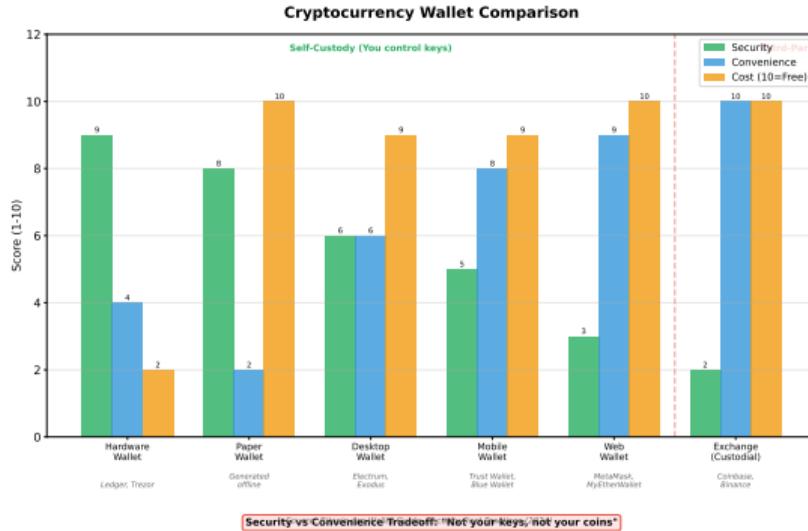
12-Word Example:

witch collapse practice feed shame open despair creek road again ice least

Properties: 128-bit entropy = 12 words, 256-bit = 24 words; 2048-word BIP-39 dictionary; checksum ensures validity

BIP-39 mnemonic phrases enable human-readable backup of cryptographic keys.

Wallet Types: Hot vs Cold



Hot Wallets:

- Connected to internet
- Software/mobile wallets
- Convenient but vulnerable

Cold Wallets:

- Offline storage
- Hardware wallets, paper wallets
- Secure but less convenient

Comparative analysis helps identify the right tool for specific requirements.

Security Best Practices

Never Share:

- Private keys
- Seed phrases
- Wallet files without encryption

Recommendations:

- Use hardware wallets for large amounts (Ledger, Trezor)
- Backup seed phrase offline (metal, paper in safe)
- Multi-signature for institutional custody
- Never store keys on cloud services
- Verify addresses carefully (malware can swap addresses)

Warning: Lost private key = Lost funds permanently

Security analysis identifies vulnerabilities and helps design robust systems.

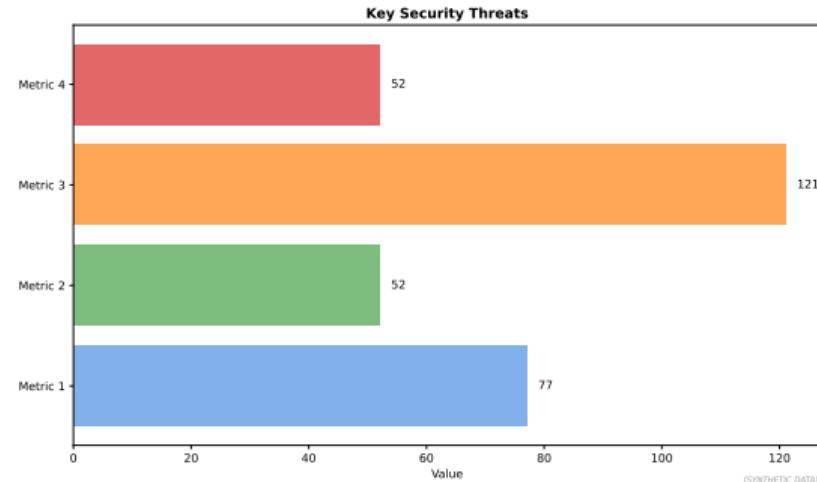
Real-World Implications

Benefits:

- Self-custody (be your own bank)
- No intermediaries
- Censorship resistance
- Programmable ownership

Challenges:

- User error (lost keys)
- No password reset
- Irreversible transactions
- Phishing attacks



Key concepts from this slide inform practical applications in finance.

- **Public Key Cryptography:** Two keys (public/private), trapdoor functions
- **ECC:** Efficient, smaller keys, basis for Bitcoin/Ethereum signatures
- **ECDSA:** Digital signatures prove transaction authorship
- **Wallets:** Manage private keys, addresses derived from public keys
- **HD Wallets:** One seed generates infinite keys (BIP-32/39/44)
- **Security:** Private key = ownership, loss is permanent

Next Lesson: Proof of Work – how cryptography secures the blockchain