E-cash and Cryptocurrency

Outline

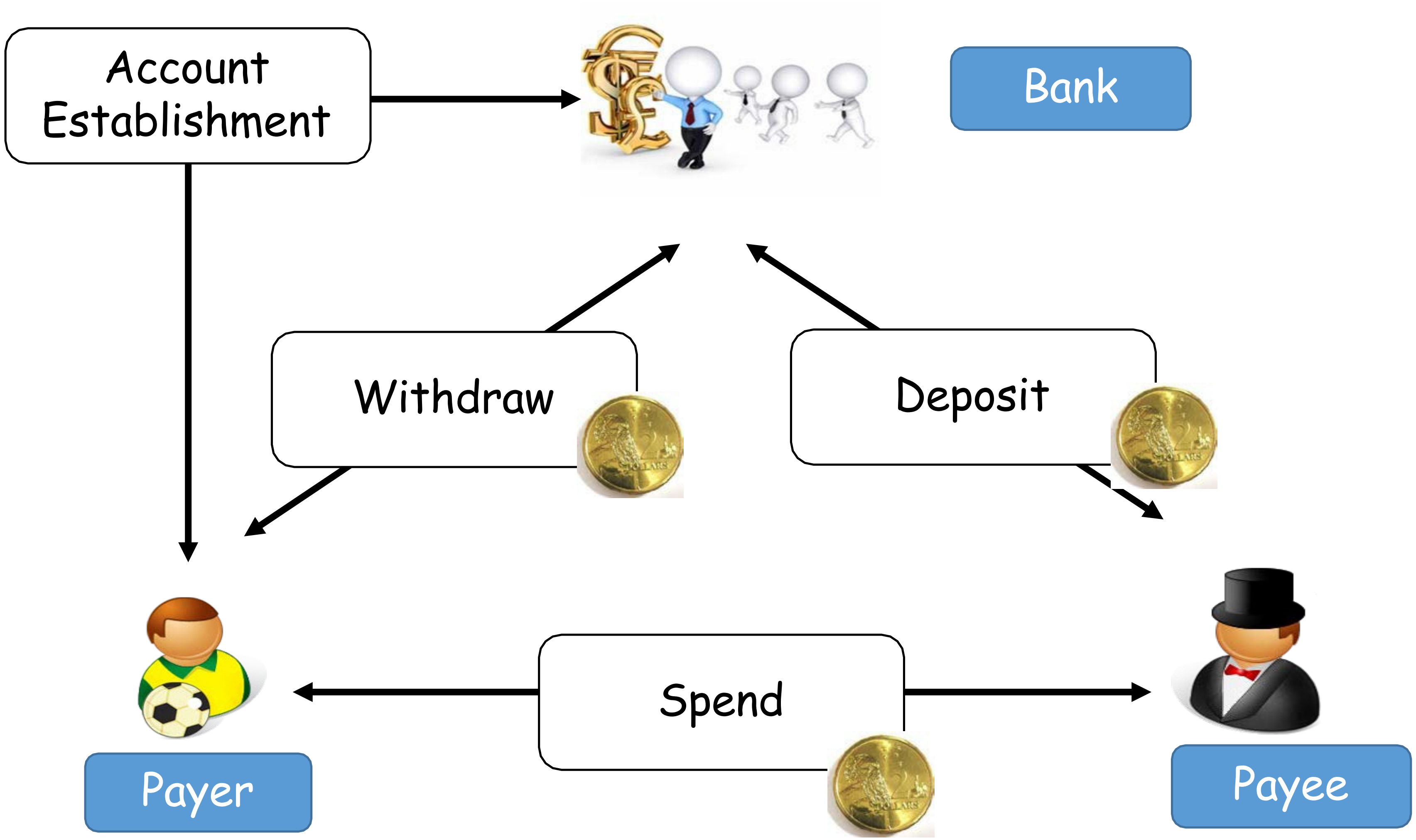
- Old style e-cash systems
  - Blind signature based

- Cryptocurrency
  - Blockchain
  - Bitcoin
  - Privacy enhancement

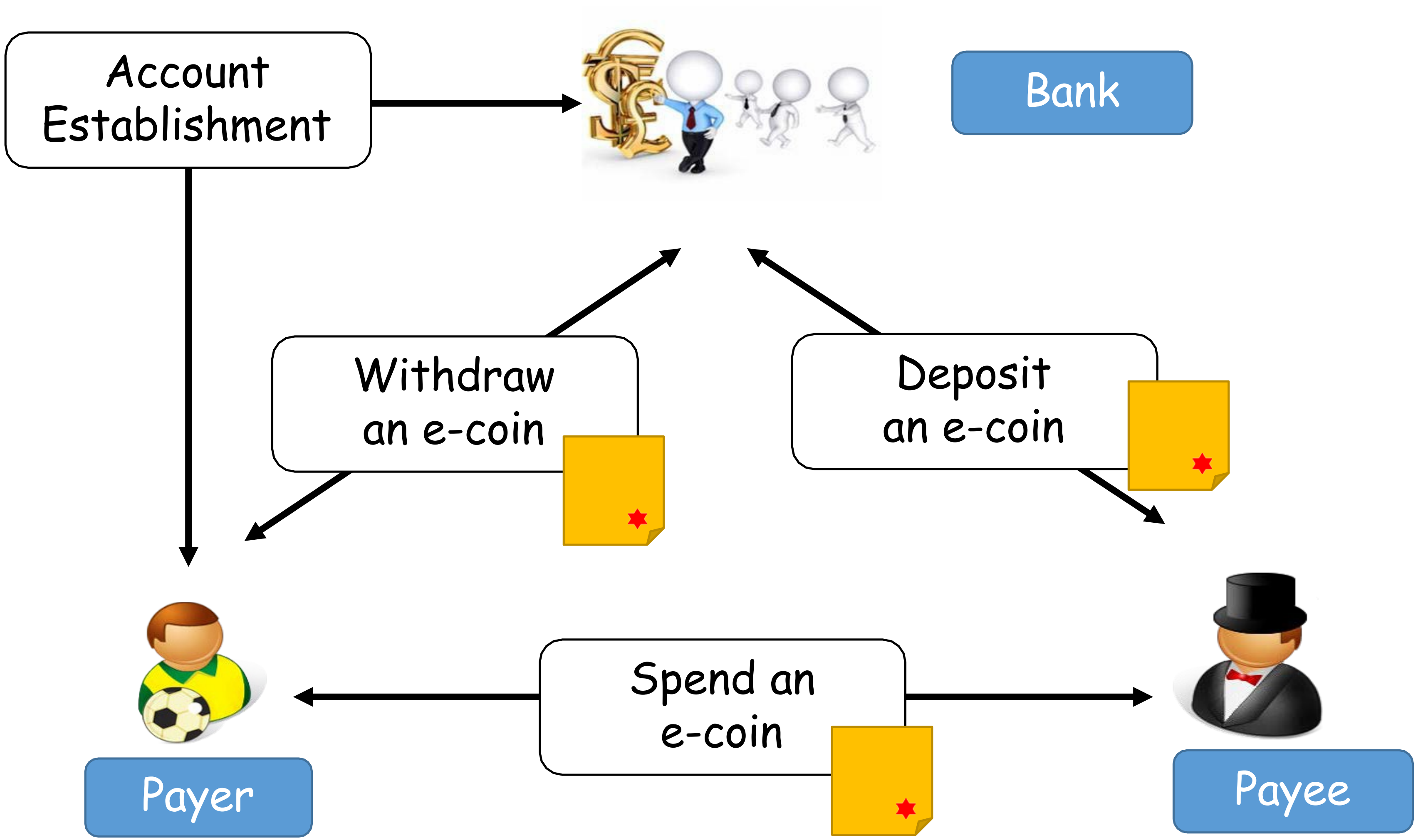
Electronic Cash

- Digital money in the form of a bit string
  - Problem: it can be easily duplicated
- Other problems:
  - What format?
  - How is it generated?
  - How to spend it?
  - Double spending prevention?
  - Anonymity?

Traditional cash system



Old style e-cash system



How to generate e-cash

- Should be generated by the trusted authorities (i.e., banks)
- Bank issues a digital string containing:
  - value, serial number, bank ID

- How to prevent forgery?
  - The note must be digitally signed by the bank

- Problems:
  - The user can make copies of the digital string (with the signature) and double spend the money
  - How to ensure user privacy

How to prevent double spending

- If the bank stays online
  - the payee checks with the bank when receiving an e-coin
  - If not spent before, the payee accepts the e-coin and deposits to the bank
  - If the e-coin has been spent, reject the payment
- If the bank is offline
  - More difficult

How to achieve payer anonymity?

- The bank knows the payer who withdrew the e-coin
- The bank can link the e-coin deposited by the payee
  - The bank knows how the payer spent the money
- How to prevent this?
  - Can use blind signature

How to use blind signature

- Payer generates an e-coin M = (value, serial number, bank ID)
- Payer sends value and *blinded* M (i.e., B in the previous slide) to the bank
- Bank deducts value from payer's account and issues a blind signature on B
- Payer unblind the signature to get the banks' normal signature on M

Problem:

If we allow coins with different values, how can the bank ensure the value given by the user is the same as the value in M?

Online e-cash

1. Payer prepares a coin M = ($1, serial number, bank ID)
   - The serial number should be unique (can use a large (e.g., 256-bit) random number)
2. Payer blinds the coin and obtains a blind signature on it from the bank
3. Payer un-blinds the signature to get bank's normal signature on M
4. Payer spends the coin(s) at a payee's shop
5. Payee verifies Banks signature
   - If invalid, reject the transaction
   - Otherwise, proceed
6. Payee sends the coin to bank for double-spending checking
7. Bank checks if the coin has been used by checking the serial number
8. Bank informs the payee the checking result
9. If checking is successful, payee completes transaction and deposit the coin to his own account

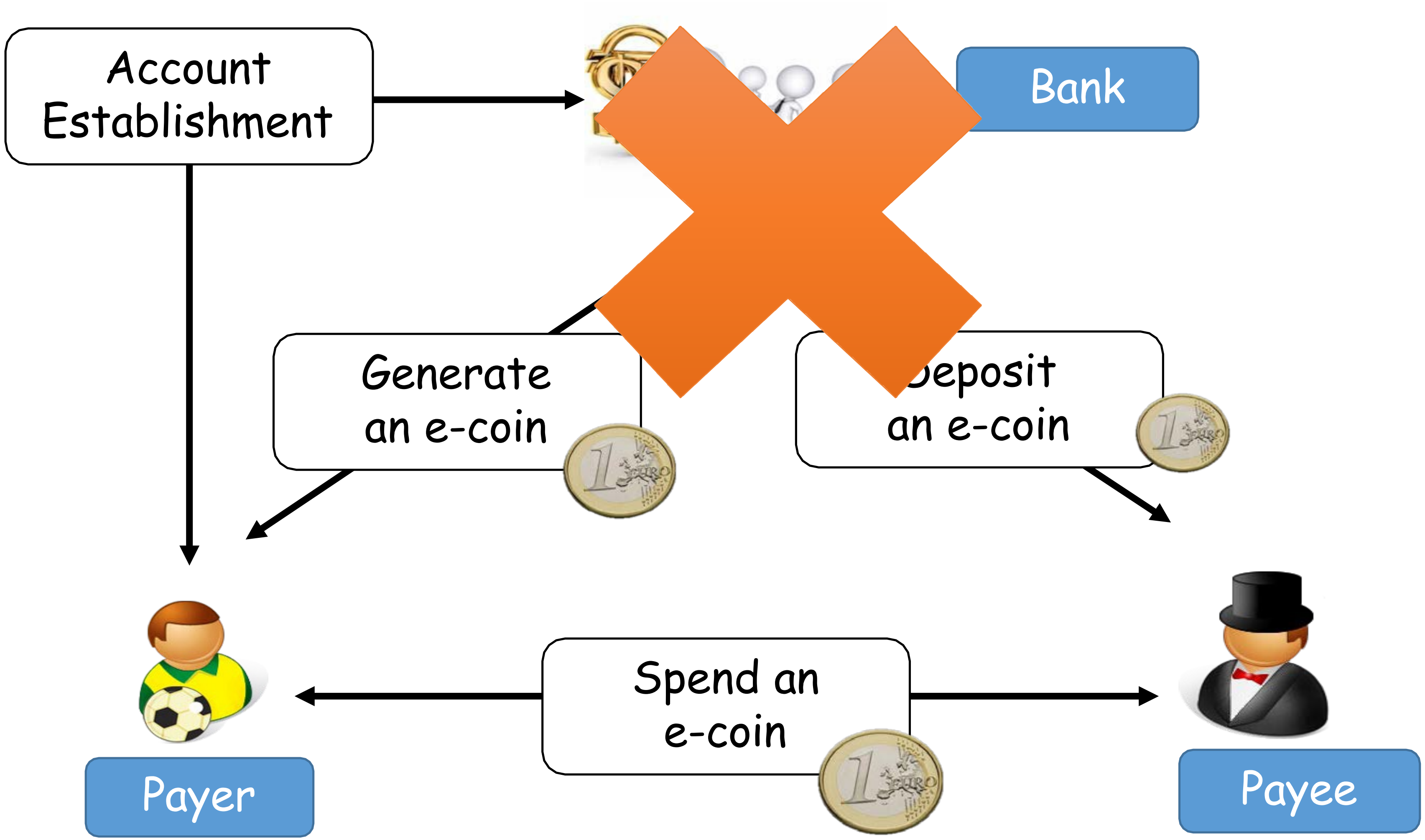
Summary

- Coin forgery prevention
  - Digital signature
- Double spending
  - Online checking by the bank
- Anonymity
  - Blind signature

Problems

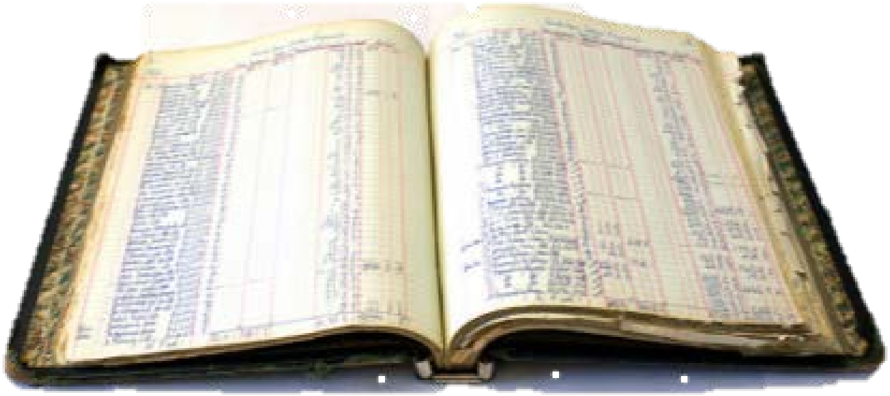
- How to prove that Alice paid Bob?
  - What if Bob takes the e-coin but does not complete the transaction
  - Need a mechanism to prove ownership
- What if the coin is lost?
  - E.g. the usb or harddisk is corrupted
- How to enable offline payment?

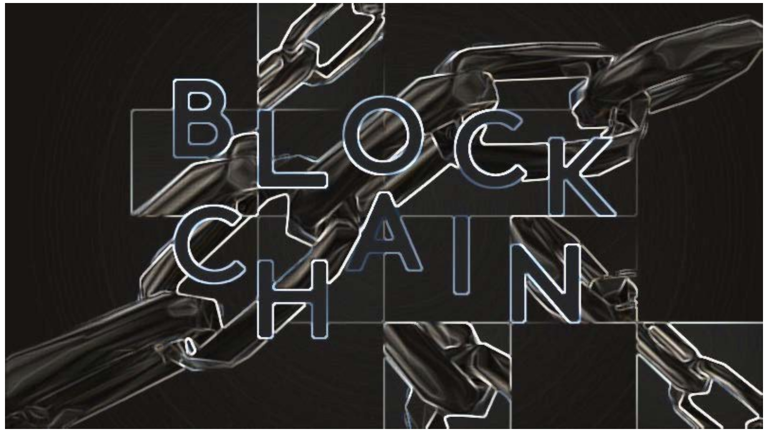
Decentralized Digital Currency



Blockchain

- A chain of blocks
- Public ledger/database
- Records all transactions across P2P network
- Shared among all nodes



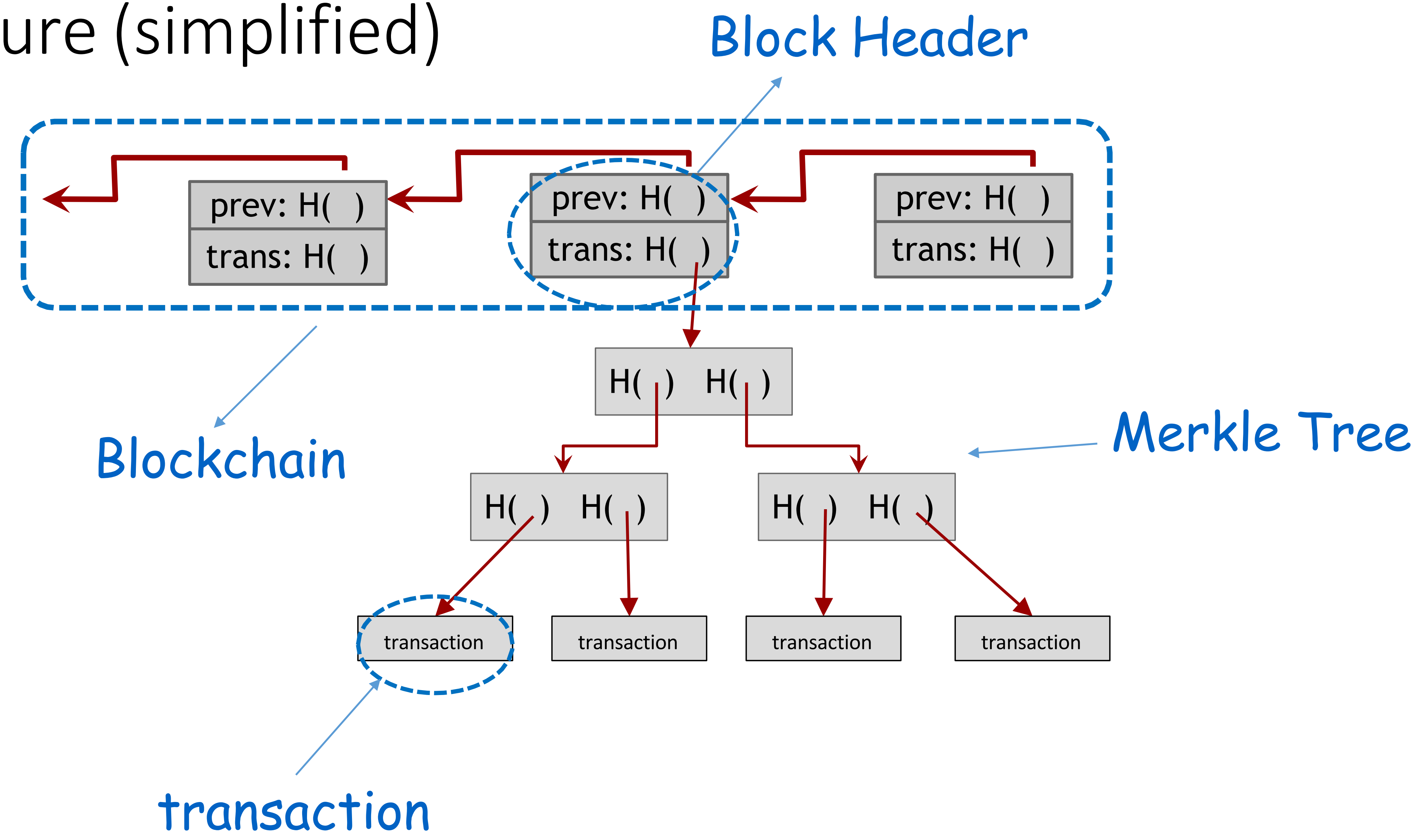


Components

Transaction → Block → Blockchain

Transaction	Block	Blockchain
Payer, Payee, amount, etc…	A list of transactions	A chain of blocks

Structure (simplified)



Crypto tool #1: Hash Function

- A cryptographic hash function (e.g., SHA-256)
- Chaining the blocks
- Building the Merkle Tree
- Proof of work

Crypto tool #1: Hash Function

- The Merkle Tree
  - The Merkle root is included in the block header
  - Can guarantee the immunity of all the transactions in the block
- Question: to verify a transaction in a block
  - What are data you need to perform the verification?
  - How many hashes do you need to perform?

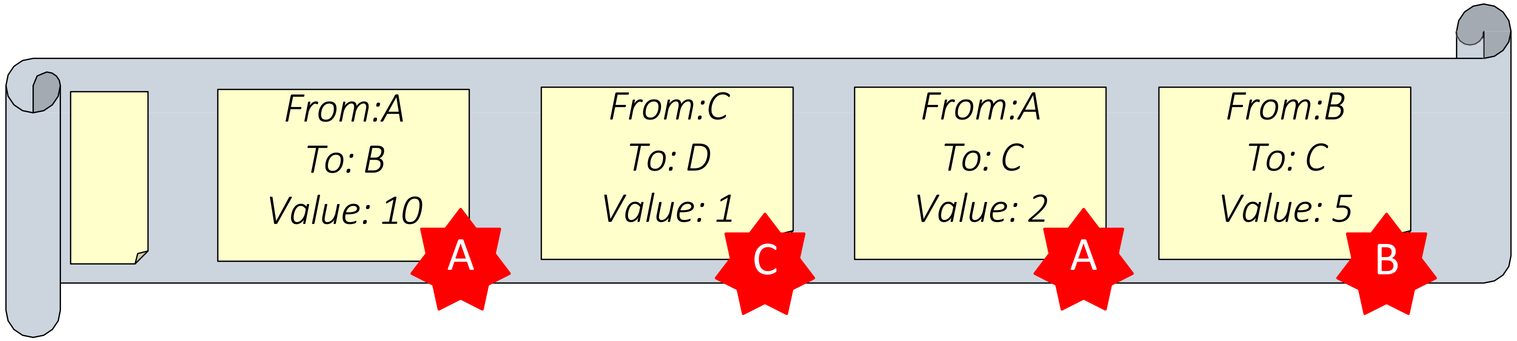
Crypto tool #1: Hash Function

- To generate/mine a new block
  - Hash the previous block header ➔ H_{i-1}
  - Collect the transactions and generate the Merkle root ➔ H_{trans}
  - Search for a "suitable" Nonce (or Salt) value
    - The hash of the above (and other) data in the header has N zeros at the beginning
    - How many Nonces do you need to try (on average) to mine a new block?
    - N can be adjusted to change the mining difficulty level

- Incentive: block creation reward + transaction fee
- A consensus protocol allows all the miners to agree on the next block (e.g., if two miners find the next new block at the same time)

Crypto tool #2: Digital Signature

- Used by a payer to authorize (i.e., digitally sign) a transaction
  - ECDSA is the most commonly used scheme
- Payer's private signing key: spending key
- Payee's public key: payment receiving address



- Privacy: addresses not directly connected to real-world identity

Privacy Enhancement

- Bitcoin privacy: Public key does not need to be “Certified”, i.e., Pseudonymity
  - Transaction are still linkable
  - Transaction analysis could reveal user identity

- Identity privacy enhancing techniques
  - One-time address

One-time address

- Hierarchical structure
- User keeps a master key pair
- Many one-time key pairs can be generated based on the master key pair
- Each transaction involves only a one-time key pair
  - Transactions are unlinkable
  - Examples: Deterministic Wallet (Bitcoin), Stealth Address (Monero)

Hiding the amount

- In the previous construction, we only considered identity privacy
- Desirable to also hide transaction amount
- Tool: a commitment scheme

Commitment Scheme

- A commitment scheme consists of three algorithms (Gen, Commit, Reveal)
  - Gen
    - outputs the system parameters: param
  - Commit
    - input: message m
    - output: a committed string C + a value D

    ➔ C is the commitment, (m, D) is the opening of the commitment
  - Verify
    - on input C, (m, D), output valid/invalid

Commitment Scheme

- Security requirements
  - binding
    - if Commit(m) ➔ C, D
    - it is impossible to find (m’, D’) such that Verify(C, D’, m’) = valid but m’ ≠ m
    - Meaning C can only have one valid openning
  - Hiding
    - Given C only, it is impossible to learn anything about m

- How are these requirements related to the cryptocurrency scenario?

Pedersen Commitment

- Let p and q be large prime numbers
  - q | p-1
- Let g and h be two generators (or primitive elements) of a group G with order q
  - G is a subgroup of Z*p
- The parameter is (p, q, g, h)

Pedersen Commitment

- Commit
  - To commit a message m in {0, …, q-1}
  - Pick a random number D in {0, …, q-1}
  - Compute $C = g^m h^D \bmod p$
  - The commitment string is C
  - Keep the opening (m, D) private

Pedersen Commitment

- Verify
  - From C and (m, D)
  - Everyone can check if
    - $C = g^m h^D \bmod p$

Example

- Let p = 23, q = 11, g = 3, h = 2

- Let’s say we would like to commit a message m = 7
- Generate a random number D = 8
- Compute $C = 3^7 2^8 \bmod 23$
- C = 6

Security Requirements

- Hiding
  - given C, no one can tell what has been committed
- Binding
  - given $C = g^m h^D \bmod p$, no one can output (m', D') such that $C = g^{m'} h^{D'} \bmod p$ and $m \neq m'$

Hiding

- Let p = 23, q = 11, g = 3, h = 2

- Given C = 6, can you tell what has been committed?

Impossible

Hiding

- For any value of C and m, there exists a D such that $C = g^m h^D \bmod p$
- E.g. when C = 6…

C	m	D
6	1	1
6	2	4
6	3	7
6	4	10
6	5	2
6	…	…

Binding

- What if I can find two values such that:
  - $C = g^m h^D \bmod p$
  - $C = g^{m'} h^{D'} \bmod p$
- I can compute the discrete logarithm of h to base g mod p
- By contradiction, it is infeasible to break binding for large p and q

Hiding the amount

- The amount v is stored as $C = g^v \cdot h^r$
- (v, r) is known to the account owner
- When performing a Confidential Transaction
  - Create a one-time account (OTA) for the payee (using the deterministic wallet)
  - Create a commitment for the OTA $C' = g^{v'} \cdot h^{r'}$
  - Authorise the transaction by signing the commitment
- Question: how to prove v = v'?

- Caution: Need a trusted setup - Dicrete log between g and h must be unknown to anyone

Privacy-focused Cryptocurrencies

- Based on more advanced cryptographic techniques
- Monero (Ring Confidential Transactions)
  - With additional crypto techniques
  - Stealth Address
  - Ring signature
- Z-cash
  - Based on (advanced) zero-knowledge proofs