Blockchains

Thierry Sans

A centralized ledger (Trusted Third Party)

From	To	Amount
	Alice	100
	Bob	50
Alice	Bob	20

➡ The bank controls the access to the ledger and ensures its correctness

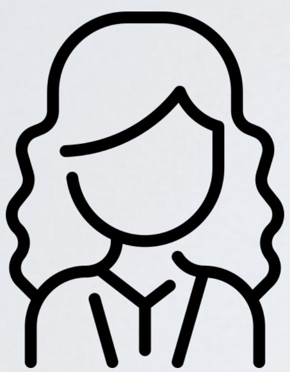
Pros/cons of using a centralized ledger

- ✓ Easy to authenticate the users
- ✓ Easy to ensure that data entries are valid
- ◉ But what if the bank goes down? (reliability issue)
- ◉ And what if the bank (or a malicious employee) cooks the books? (security issue)

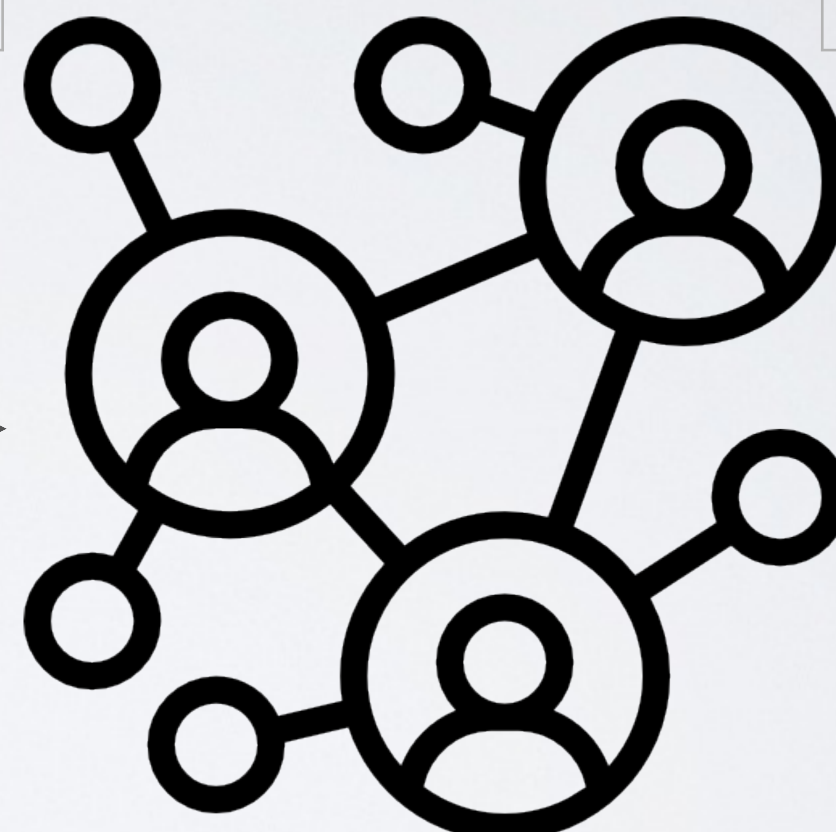
A decentralized ledger (over a P2P network)

From	To	Amount
	Alice	100
	Bob	50
Alice	Bob	20

From	To	Amount
	Alice	100
	Bob	50
Alice	Bob	20



`T(Alice, Bob, $20)`



From	To	Amount
	Alice	100
	Bob	50
Alice	Bob	20

➡ All nodes have a copy of the ledger and ensure its correctness locally

Pros/cons of using a decentralized ledger

- ✓ Some nodes can go down but not the network entirely (better reliability)
- ✓ Some nodes can be malicious, but the rest of the network will have the legitimate copy of the ledger (better security)
- ◉ Harder to authenticate users
- ◉ Hard to ensure that all nodes have the same ledger (consistency)

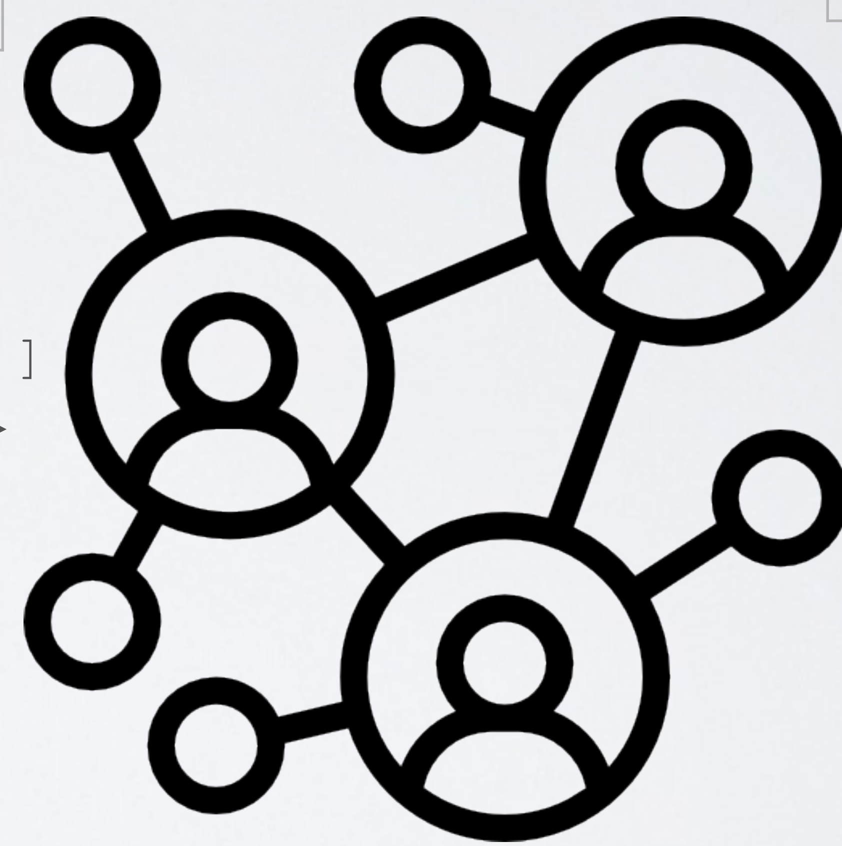
Solving Authentication

Using public-key cryptography and digital signature

From	To	Amount
	pk_A	100
	pk_B	50
pk_A	pk_B	20

(pk_A, sk_A)

pk_A, sign[sk_A, T(Alice, Bob, $20)]



From	To	Amount
	pk_A	100
	pk_B	50
pk_A	pk_B	20

From	To	Amount
	pk_A	100
	pk_B	50
pk_A	pk_B	20

- ➡ The public key is the identity
- ➡ The signature is the authentication mechanism

Solving Consistency

What a P2P network looks like

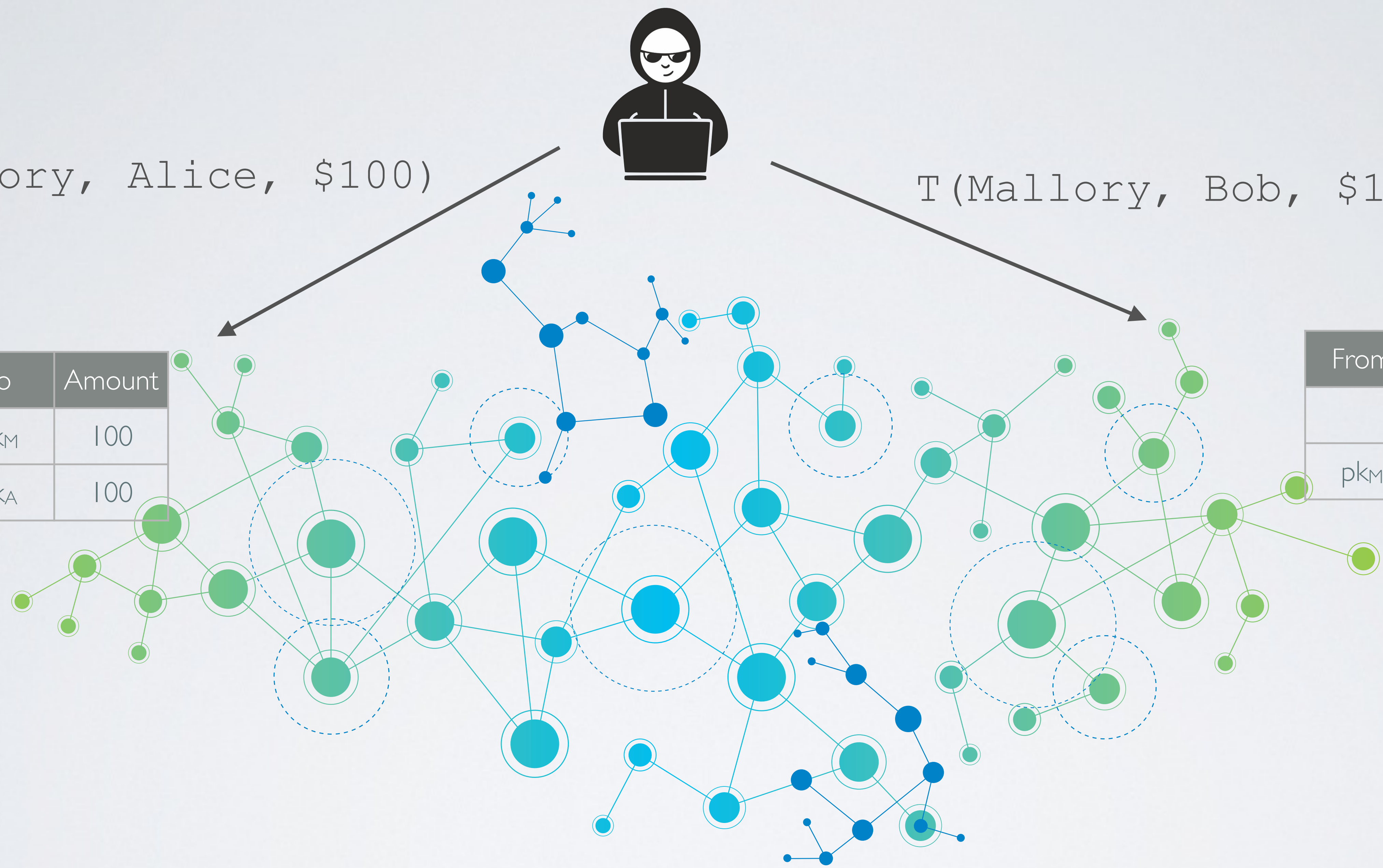
Data Propagation in P2P network

Flooding routing algorithm
When receiving a transaction, forward it to all connected peers

➡ A transaction might take time to be broadcasted on the network

- An attacker can use that to do a double spending attack by broadcasting two conflicting transactions to distant nodes in the network

Double spending attack example



From	To	Amount
	pk_M	100
pk_M	pk_A	100

From	To	Amount
	pk_M	100
pk_M	pk_B	100

From	To	Amount
	pk_M	100
??	??	??

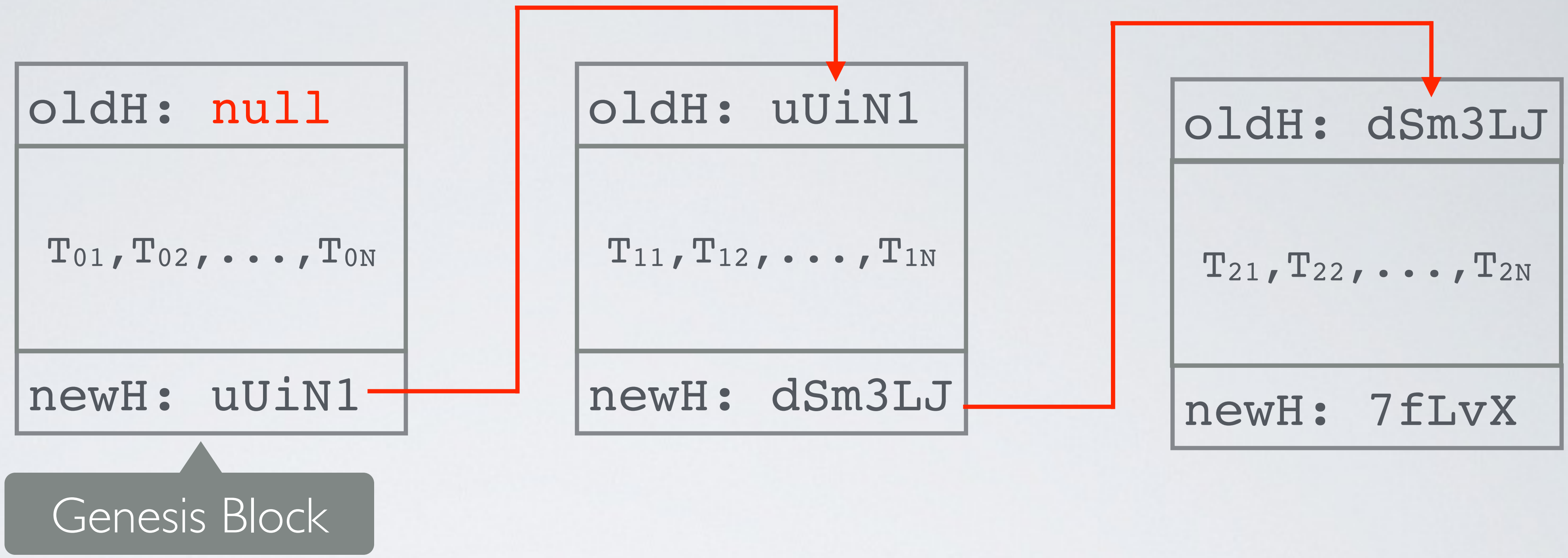
The Blockchain Solution

The idea is to have the all nodes in the network "*agreeing*" from time to time about a snapshot of the valid transactions so far

- All transactions are verified and accepted into a mempool of unconfirmed transactions
- Every t seconds, "*the network selects one node*" to create a block of confirmed transactions
- The block is chained to the previous one
- That block is broadcasted to the network and each node check whether this block is valid

✓ The time interval between two blocks should be long enough so that "most" of the network has had time to receive the block

Example



A block is valid if

- The old hash corresponds to the previous block hash
- The block hash is $\mathtt{H(oldH + T_0 + T_1 + \ldots + T_n)}$
- All transactions are valid (no double spending)

One big problem to solve ...

How does the network "*agree*" on which node should create and broadcast the block?

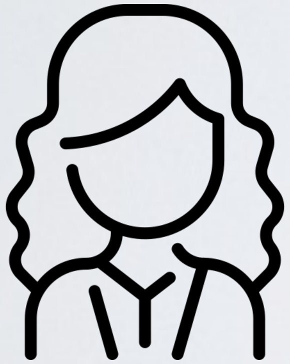
➡ Consensus (coming soon)

- Proof of Work (Bitcoin)
- Proof of Stake (Ethereum)

Two Types of Blockchains

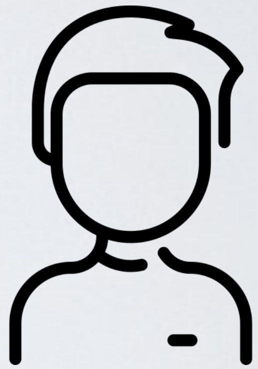
Account-based blockchains

(pk_A, sk_A)



Tx	From	To	Amount
1		pk_A	100
2		pk_B	20
3	pk_A	pk_B	60
4	pk_B	pk_A	70

(pk_B, sk_B)

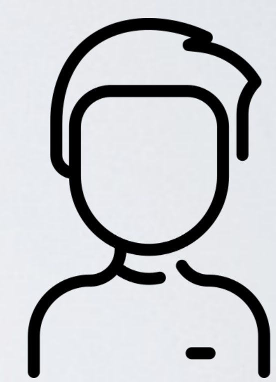


Coin-based blockchains
(a.k.a UTXO Unspent Transaction Output)

(pk_1, sk_1)
(pk_4, sk_4)
(pk_5, sk_5)

Tx	Inputs	Outputs
1		$pk_1(100)$
2		$pk_2(20)$
3	$pk_1(100)$	$pk_3(60)$ $pk_4(40)$
4	$pk_2(20)$ $pk_3(60)$	$pk_5(70)$ $pk_6(10)$

(pk_2, sk_2)
(pk_3, sk_3)
(pk_6, sk_6)



pros and cons

UTXO-based (e.g Bitcoin)

- ✓ Some relative privacy (no links between keys)
- ◉ Hard to manage all of these keys
- ➡ Intermediate solution: HD wallets (coming later)

Account-based (e.g Ethereum)

- ✓ Easy way to manage keys
- ◉ Hard to have privacy (transactions are all linked)
- ➡ Candidate solution: ZK-proofs (coming later)