Reminder: proj #1 is posted on the course web site. Due Oct. 3
(1) **SHA256**: a collision resistant hash function that outputs 32-byte hash values

**Applications:**

- a binding commitment to one value: \( \text{commit}(m) \rightarrow H(m) \)
  or to a list of values: \( \text{commit}(m_1, \ldots, m_n) \rightarrow \text{Merkle}(m_1, \ldots, m_n) \)

- Proof of work with difficulty \( D \):
  \[ \text{given } x \text{ find } y \text{ s.t. } H(x, y) < 2^{256}/D \text{ takes time } O(D) \]
Digital Signatures

Physical signatures: bind transaction to author

Bob agrees to pay Alice 1$

Problem in the digital world:

anyone can copy Bob’s signature from one doc to another
Digital signatures

Solution: make signature depend on document

Signer

Verifier

Bob agrees to pay Alice 1$

signature

signing algorithm

secret signing key (sk)

public verification key (pk)

'verify' or 'reject'
Digital signatures: syntax

**Def:** a signature scheme is a triple of algorithms:

- **Gen():** outputs a key pair \( (pk, sk) \)
- **Sign\((sk, msg)\):** outputs sig. \( \sigma \)
- **Verify\((pk, msg, \sigma)\):** outputs ‘accept’ or ‘reject’

**Secure signatures:** (informal)

Adversary who sees signatures on many messages of his choice, cannot forge a signature on a new message.
Families of signature schemes

1. **RSA signatures (old ... not used in blockchains):**
   - long sigs and public keys (≥256 bytes), fast to verify

2. **Discrete-log signatures:** Schnorr and ECDSA (Bitcoin, Ethereum)
   - short sigs (48 or 64 bytes) and public key (32 bytes)

3. **BLS signatures:** 48 bytes, aggregatable, easy threshold (Ethereum 2.0, Chia, Dfinity)

4. **Post-quantum signatures:** long (≥600 bytes)

(details in CS255)
Signatures are used everywhere:
• ensure Tx authorization,
• governance votes,
• consensus protocol votes.

$sk_1$  

$sk_2$
In summary ...

Digital signatures: \((\text{Gen}, \text{Sign}, \text{Verify})\)

- \(\text{Gen}() \rightarrow (pk, sk)\),
- \(\text{Sign}(sk, m) \rightarrow \sigma\),
- \(\text{Verify}(pk, m, \sigma) \rightarrow \text{accept/reject}\)

signing key \(\rightarrow\) verification key
Bitcoin mechanics
This lecture: Bitcoin mechanics

Total market value:

- Jan. 2009: Bitcoin network launched
- Sep. 2022: $386B
This lecture: Bitcoin mechanics

user facing tools (cloud servers)

applications (DAPPs, smart contracts)

Execution engine (blockchain computer)

Sequencer: orders transactions

Data Availability / Consensus Layer

today

next week
First: overview of the Bitcoin consensus layer

end users

\[ \text{sk}_A \]
\[ \text{sk}_B \]
\[ \text{sk}_C \]

signed Tx

Bitcoin P2P network

typically, miners are connected to eight other peers (anyone can join)
First: overview of the Bitcoin consensus layer

Miners broadcast received Tx to the P2P network.

Every miner:
Validates received Tx and stores them in its mempool (unconfirmed Tx).

Note: miners see all Tx before they are posted on chain.
Every ≈10 minutes:

- Each miner creates a candidate block from Tx in its mempool
- A “random” miner is selected (how: next week), and broadcasts its block to P2P network
- All miners validate new block
Selected miner is paid 6.25 BTC in **coinbase Tx** (first Tx in the block)

- only way new BTC is created
- block reward halves every four years
  \[ \Rightarrow \text{max 21M BTC (currently 19.1M BTC)} \]

note: miner chooses order of Tx in block
Next week:

**Safety / Persistence:**

- to remove a block, need to convince 51% of mining power *

**Liveness:**

- to block a Tx from being posted, need to convince 51% of mining power **

(some sub 50% censorship attacks, such as feather forks)
Bitcoin blockchain: a sequence of block headers, 80 bytes each

- **Genesis block**

- **BH₁**
  - version (4 bytes)
  - prev (32 bytes)
  - time (4 bytes)
  - bits (4 bytes)
  - nonce (4 bytes)
  - Tx root (32 bytes)

- **BH₂**
  - prev
  - Tx root

- **BH₃**
  - prev
  - Tx root...

- **80 bytes**

- **Coinbase Tx**
Bitcoin blockchain: a sequence of block headers, 80 bytes each

**time**: time miner assembled the block. Self reported.
   (block rejected if too far in past or future)

**bits**: proof of work difficulty

**nonce**: proof of work solution

**Merkle tree**: payer can give a short proof that Tx is in the block

new block every $\approx$10 minutes.
<table>
<thead>
<tr>
<th>Height</th>
<th>Mined</th>
<th>Miner</th>
<th>Size</th>
<th>#Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>648494</td>
<td>17 minutes</td>
<td>Unknown</td>
<td>1,308,663 bytes</td>
<td>1855</td>
</tr>
<tr>
<td>648493</td>
<td>20 minutes</td>
<td>SlushPool</td>
<td>1,317,436 bytes</td>
<td>2826</td>
</tr>
<tr>
<td>648492</td>
<td>59 minutes</td>
<td>Unknown</td>
<td>1,186,609 bytes</td>
<td>1128</td>
</tr>
<tr>
<td>648491</td>
<td>1 hour</td>
<td>Unknown</td>
<td>1,310,554 bytes</td>
<td>2774</td>
</tr>
<tr>
<td>648490</td>
<td>1 hour</td>
<td>Unknown</td>
<td>1,145,491 bytes</td>
<td>2075</td>
</tr>
<tr>
<td>648489</td>
<td>1 hour</td>
<td>Poolin</td>
<td>1,359,224 bytes</td>
<td>2622</td>
</tr>
<tr>
<td><strong>Timestamp</strong></td>
<td>2020-09-15 17:25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>648493</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Miner</strong></td>
<td>SlushPool</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Number of Transactions</strong></td>
<td>2,826</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Difficulty</strong></td>
<td>(D) 17,345,997,805,929.09 (adjusts every two weeks)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Merkle root</strong></td>
<td>350cbb917c918774c93e945b960a2b3ac1c8d448c2e67839223bbcf595baff89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transaction Volume</strong></td>
<td>11256.14250596 BTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Block Reward</strong></td>
<td>6.25000000 BTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fee Reward</strong></td>
<td>0.89047154 BTC (Tx fees given to miner in coinbase Tx)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
View the blockchain as a sequence of Tx  (append-only)
Tx structure (non-coinbase)

**input:**
- input[0]
- input[1]
- input[2]
- witnesses (part of input)
- locktime

**output:**
- output[0]
- output[1]

**TxID = H(Tx)** (excluding witnesses)

- 32 byte hash
- 4 byte index
- program
- ignore
- 8 bytes
- program

#BTC = value/10^8

- earliest block # that can include Tx

- output: value
- ScriptPK

- input: TxID
- out-index
- ScriptSig
- seq
Example

Tx1: (funding Tx)

UTXO: unspent Tx output

Tx2: (spending Tx)

identifies a UTXO
Example

Tx1: (funding Tx)

UTXO: unspent Tx output

Tx2: (spending Tx)

identifies a UTXO

null locktime
Validating Tx2

Miners check (for each input):

1. The program $\text{ScriptSig} | \text{ScriptPK}$ returns true

2. $\text{TxID} | \text{index}$ is in the current UTXO set

3. sum input values $\geq$ sum output values

After Tx2 is posted, miners remove UTXO$_2$ from UTXO set
An example (block 648493)

**COINBASE (Newly Generated Coins)**

0.00000000 BTC

**inputs**

- 3PuJbxJS1pKxf8EdVR18yBkD1fPAbgUtyw 0.72333974 BTC

** outputs**

- 0.05000000 BTC
- 0.00192000 BTC

**sum of fees in block added to coinbase Tx**

6.25 + Tx fees = 7.14047154 BTC
Bitcoin average Tx fees in USD (last 60 days, sep. 2022)

Bitcoin average Tx fees in USD (all time)
All value in Bitcoin is held in UTXOs

Unspent Transaction Outputs

The total number of valid unspent transaction outputs. This excludes invalid UTXOs with opcode OP_RETURN

Sep. 2022: miners need to store ≈85M UTXOs in memory
Focusing on Tx2: **TxInp[0]**

- **Value**: 0.05000000 BTC
- **Pkscript**:
  - OP_DUP
  - OP_HASH160
  - 45b21c8a0cb687d563342b6c729d31dab58e3a4e
  - OP_EQUALVERIFY
  - OP_CHECKSIG
- **Sigscription**:
  - 304402205846cace0d73de82dfbdeba4d65b9856d7c1b1730eb401cf4906b2401a69b
dc90220589d36d36be64e774c8796b96c01f29768191abeb7f56ba20ff0351280860c01
  - 03557c228b080703d52d72ead1bd93fc72f45c4ddfb4c2b720a0c458e2d069c8dd9e

**from UTXO** *(Bitcoin script)*

**from TxInp[0]**
A stack machine. Not Turing Complete: no loops.

Quick survey of op codes:
1. **OP_TRUE** (OP_1), **OP_2**, ..., **OP_16**: push value onto stack
   - 81
   - 82
   - 96

2. **OP_DUP**: push top of stack onto stack
   - 118
3. control:

99  OP_IF <statements> OP_ELSE <statements> OP_ENDIF

105  OP_VERIFY: abort fail if top = false

106  OP_RETURN: abort and fail

what is this for? ScriptPK = [OP_RETURN, <data>]

136  OP_EQVERIFY: pop, pop, abort fail if not equal
Bitcoin Script

4. arithmetic:
   \textbf{OP\_ADD, OP\_SUB, OP\_AND, ...}: pop two items, add, push

5. crypto:
   \textbf{OP\_SHA256}: pop, hash, push

   \textbf{OP\_CHECKSIG}: pop pk, pop sig, verify sig. on Tx, push 0 or 1

6. Time: \textbf{OP\_CheckLockTimeVerify} (CLTV):
   fail if value at the top of stack > Tx locktime value.
   usage: UTXO can specify min-time when it can be spent
Example: a common script

<table>
<thead>
<tr>
<th>&lt;sig&gt;</th>
<th>&lt;pk&gt;</th>
<th>DUP</th>
<th>HASH256</th>
<th>&lt;pkhash&gt;</th>
<th>EQVERIFY</th>
<th>CHECKSIG</th>
</tr>
</thead>
</table>

**stack:** empty

1

⇒ successful termination
Alice wants to pay Bob 5 BTC:

- step 1: Bob generates sig key pair \((pk_B, sk_B) \leftarrow \text{Gen}()\)
- step 2: Bob computes his Bitcoin address as \(addr_B \leftarrow H(pk_B)\)
- step 3: Bob sends \(addr_B\) to Alice
- step 4: Alice posts Tx:

ScriptPK_B:
DUP HASH256 <addr_B> EQVERIFY CHECKSIG
Transaction types: (1) P2PKH

“input” contains ScriptSig that authorizes spending Alice’s UTXO

- example: ScriptSig contains Alice’s signature on Tx

  $\implies$ miners cannot change $\text{ScriptPK}_B$ (will invalidate Alice’s signature)

---

**DUP HASH256 <addr_B> EQVERIFY CHECKSIG**

**Point to Alice’s UTXO**

- **input:** 7 BTC
- **UTXO** for Bob: 5
- **ScriptPK** for Bob: $\text{ScriptPK}_B$
- **UTXO** for Alice (change): 2
- **ScriptPK** for Alice: $\text{ScriptPK}_A$
- 0
Later, when Bob wants to spend his UTXO:

create a $T_{x_{spend}}$

$T_{x_{spend}}$: 

| TxID | 0 | ScriptSig$_B$ |

points to UTXO$_B$

<sig> <pk$_B$> (authorizes spending UTXO$_B$)

<sig> = Sign(sk$_B$, Tx) where Tx = ($T_{x_{spend}}$ excluding all ScriptSigs) (SIGHASH_ALL)

Miners validate that ScriptSig$_B$ | ScriptPK$_B$ returns true
P2PKH: comments

- Alice specifies recipient’s pk in $\text{UTXO}_B$

- Recipient’s pk is not revealed until UTXO is spent
  (some security against attacks on pk)

- Miner cannot change $<\text{Addr}_B>$ and steal funds:
  invalidates Alice’s signature that created $\text{UTXO}_B$
Segregated Witness

ECDSA malleability:
Given \((m, \text{sig})\) anyone can create \((m, \text{sig}')\) with \(\text{sig} \neq \text{sig}'\)
\[\Rightarrow\] miner can change sig in Tx and change TxID = SHA256(Tx)
\[\Rightarrow\] Tx issuer cannot tell what TxID is, until Tx is posted
\[\Rightarrow\] leads to problems and attacks

**Segregated witness:** signature is moved to witness field in Tx
\[\text{TxID} = \text{Hash(Tx without witnesses)}\]
Transaction types: (2) **P2SH:** pay to script hash

(pre SegWit in 2017)

Let’s payer specify a redeem script (instead of just pkhash)

Usage:

- payee publishes hash(redeem script) ← Bitcoint addr.
- payer sends funds to that address

\[
\text{ScriptPK in UTXO: } \text{HASH160 } <\text{H(redeem script)}> \text{ EQUAL}
\]

\[
\text{ScriptSig to spend: } \langle \text{sig}_1 \rangle \langle \text{sig}_2 \rangle \ldots \langle \text{sig}_n \rangle \langle \text{redeem script} \rangle
\]

payer can specify complex conditions for when UTXO can be spent
Miner verifies:

(1) `<ScriptSig> ScriptPK = true` ← payee gave correct script

(2) `ScriptSig = true` ← script is satisfied
Example P2SH: multisig

**Goal**: spending a UTXO requires $t$-out-of-$n$ signatures

Redeem script for 2-out-of-3: (set by payer)

\[
\langle 2 \rangle \ <PK_1> \ <PK_2> \ <PK_3> \ <3 \rangle \ \text{CHECKMULTISIG}
\]

hash gives P2SH address

ScriptSig to spend: (by payee)

\[
\langle 0 \rangle \ <\text{sig1}> \ <\text{sig3}> \ <\text{redeem script}>
\]
Next lecture: interesting scripts, wallets, and how to manage crypto assets