Note: HW#1 is posted on the course web site. Due Sep. 28.
Recap

(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
(2) **Digital signatures:** (Gen, Sign, Verify)

Gen() $\rightarrow$ (pk, sk),

Sign(sk, m) $\rightarrow$ $\sigma$, Verify(pk, m, $\sigma$) $\rightarrow$ accept/reject

signing key

verification key
This lecture: Bitcoin mechanics

Oct. 2008: paper by Satoshi Nakamoto
Jan. 2009: Bitcoin network launched

Total market value:

Sep. 2020: $200B
This lecture: Bitcoin mechanics

Layer 3: user facing tools (cloud servers)
Layer 2: applications (DAPPs, smart contracts)
Layer 1.5: compute layer (blockchain computer)
Layer 1: consensus layer

next week
First: overview of the Bitcoin consensus layer

end users

\(\text{sk}_A\) \n\(\text{sk}_B\) \n\(\text{sk}_C\)

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 Tx before they are posted on chain
First: overview of the Bitcoin consensus layer

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
First: overview of the Bitcoin consensus layer

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
  \[\Rightarrow \text{max 21M BTC (currently 18.5M BTC)}\]

note: miner chooses order of Tx in block
Properties (very informal)

Next week:

**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 **
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

...
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 ≈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>
</tbody>
</table>
|:block 648493:

**Timestamp** | 2020-09-15 17:25  
**Height** | 648493  
**Miner** | SlushPool  
**Number of Transactions** | 2,826  
**Difficulty** | 17,345,997,805,929.09 (adjusts every two weeks)  
**Merkle root** | 350cbb917c918774c93e945b960a2b3ac1c8d448c2e67839223bbcf595baff89  
**Transaction Volume** | 11256.14250596 BTC  
**Block Reward** | 6.25000000 BTC  
**Fee Reward** | 0.89047154 BTC
View the blockchain as a sequence of Tx (append-only)
**Tx structure (non-coinbase)**

- **Inputs**
  - input[0]
  - input[1]
  - input[2]

- **Outputs**
  - output[0]
  - output[1]

- **Witnesses**
- locktime

- **Segwit (4 bytes)**

- **TxID**

- **ScriptSig**

- **ScriptPK**

- **Script**

- **Value**
  - value = #BTC/10^8
  - [10^-8, ..., 2^{37}]
Example

**Tx1:**
(funding Tx)

- **input:**
  - value: 2
  - ScriptPK

- **UTXO₁**

**Tx2:**
(spending Tx)

- **TxID:** 1
- **ScriptSig**

Identifies a UTXO

**null locktime**

**Tx1:**
(funding Tx)

- **TxIn[0]**
  - value: 2
  - ScriptPK

- **UTXO₁**

**Tx2:**
(spending Tx)

- **TxID:** 1
- **ScriptSig**

Identifies a UTXO

**null locktime**

**UTXO:** unspent Tx output
**Example**

**Tx1:** (funding Tx)

- **TxIn[0]**: input
- **TxOut[0]**: ScriptPK
- **UTXO₁**: unspent Tx output

**Tx2:** (spending Tx)

- **TxID**: 1
- **TxIn[0]**: identifies a UTXO
- **TxOut[0]**: output
- **TxOut[1]**: output
- **UTXO₃**, **UTXO₄**: unspent Tx output

**TxID** identifies a UTXO

null locktime
Validating Tx2

Miners check (for each input):

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

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

3. sum input values \( \geq \) sum output values

After Tx2 is posted, miners remove UTXO from UTXO set
An example (block 648493) [2826 Tx]

**COINBASE (Newly Generated Coins)**

- **input**: Tx0
  - 0.00000000 BTC

- **output**: 1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE
  - 7.14047154 BTC
  - 0.00000000 BTC
  - 0.00000000 BTC

---

**3PuJbxJS1pKxf8EdVR18yBkD1fPAbgUtyw**

- **input**: 0.72333974 BTC

- **output**: 1E5Ao1VUnA5HffvXf2Xmud6avUgwkFnj
  - 0.00917379 BTC
  - 0.61504199 BTC
  - 0.09290152 BTC
  - 0.00616444 BTC

---

**17MWze4Z1uPjnvqvj7SAnGtxcoVq11H8A**

- **input**: 0.05000000 BTC

- **output**: 3G3C2RFQ8gsf77EQpdR4ZReChWFKEHhxVU
  - 0.04808000 BTC
  - 0.04808000 BTC

---

sum of fees in block added to coinbase Tx
Tx fees

Bitcoin average Tx fees in USD

![Bitcoin average Tx fees in USD graph](image1)

Ethereum average Tx fees in USD

![Ethereum average Tx fees in USD graph](image2)
Focusing on Tx2:  TxInp[0]

<table>
<thead>
<tr>
<th>Value</th>
<th>0.05000000 BTC</th>
</tr>
</thead>
</table>
| Pkscript       | OP_DUP
|                | OP_HASH160
|                | 45b21c8a0cb687d563342b6c729d31dab58e3a4e
|                | OP_EQUALVERIFY
|                | OP_CHECKSIG                   |
| Sigscript      | 304402205846cace0d73de82dfbdeba4d65b9856d7c1b1730eb401cf4906b2401a69b
dc90220589d36d36be64e774c8796b96c011f29768191abeb7f56ba20ff0351280860c0103557c228b080703d52d72ead1bd93fc72f45c4ddb4c2b7a20c458e2d069c8dd9e |
All value in Bitcoin is held in UTXOs

Unspent Transaction Outputs

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

Sep. 2020: miners need to store ≈70M UTXOs in memory
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
4. arithmetic:
   - **OP_ADD, OP_SUB, OP_AND, ...**: pop two items, add, push

5. crypto:
   - **OP_SHA256**: pop, hash, push
   - **OP_CHECKSIG**: pop sig, pop pk, verify sig. on Tx, push 0 or 1

6. Time: **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

\[
\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \rightarrow \text{\texttt{DUP}} \rightarrow \text{\texttt{HASH256}} \rightarrow \text{\texttt{pkhash}} \rightarrow \text{\texttt{EQVERIFY}} \rightarrow \text{\texttt{CHECKSIG}}
\]

\textbf{stack}: empty

\begin{align*}
&\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \\
&\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \\
&\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \rightarrow \text{\texttt{\textless\textless\texttt{hash}}} \\
&\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \rightarrow \text{\texttt{\textless\textless\texttt{hash}}} \rightarrow \text{\texttt{pkhash}} \\
&\text{\texttt{\textless\textless\texttt{sig}}} \rightarrow \text{\texttt{\textless\textless\texttt{pk}}} \\
&1
\end{align*}

\Rightarrow \text{successful termination}

\text{\texttt{verify}}(\text{\texttt{pk}}, \text{\texttt{Tx}}, \text{\texttt{sig}})
Transaction types: (1) P2PKH

Alice want 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 creates Tx:

```
input  5  ScriptPK_B  2  ScriptPK_A  0
7 BTC  UTXO_B for Bob  UTXO_A for Alice (change)
```

ScriptPK_B:
```
DUP HASH256 <Addr_B> EQVERIFY CHECKSIG
```
Later, when Bob wants to spend his UTXO:

Create a $T_{x_{\text{spend}}}$

$T_{x_{\text{spend}}}$: 

- **TxID**: 0
- **ScriptSig$_B$**: 

$\text{ScriptSig}_B$:

\[ <\text{sig}> <\text{pk}_B> \]

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

Miners validate that $\text{ScriptSig}_B \mid \text{ScriptPK}_B$ returns true.
P2PKH: comments

• Alice specifies recipient’s pk in UTXO\textsubscript{B}

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

• Miner cannot change <Addr\textsubscript{B}> and steal funds:
  invalidates the signature that created the UTXO\textsubscript{B}
Segregated Witness

ECDSA malleability:
• given (m, sig) anyone can create (m, sig’) with sig ≠ sig’
⇒ miner can change sig in Tx and change TxID = SHA256(Tx)
⇒ Tx issuer cannot tell what TxID is, until Tx is posted
⇒ leads to problems and attacks

Segregated witness: signature is moved to witness field in Tx
TxID = Hash(Tx without witnesses)
Transaction types: (2) P2SH: pay to script hash

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

ScriptPK in UTXO: HASH160 <H(redeem script)> EQUAL

ScriptSig to spend: <sig_1> <sig_2> ... <sig_n> <redeem script>

payer can specify complex conditions for when UTXO can be spent
P2SH

Miner verifies:

1. $\langle\text{ScriptSig}\rangle \text{ ScriptPK} = \text{true} \quad \leftarrow \text{payee gave correct script}$

2. $\text{ScriptSig} = \text{true} \quad \leftarrow \text{script is satisfied}$
Example P2SH: multisig

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

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

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

hash gives P2SH address

ScriptSig to spend: (by payee)

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