MIT Digital Currency Initiative and the University of Brasilia presents

Cryptocurrency Design and Engineering

Lecture 11: Computability and Smart Contracts
Taught by: Neha Narula
October 14th, 2025
MAS.S62

Bitcoin programmability recap

- Recall: script defines the spending predicates for UTXOs
  - scriptPubKey and scriptSig
- Stack-based scripting language; transaction-local. No global state.
- Examples: Pay to script hash, multisigs

Today

- Upgrade to Bitcoin: Taproot
- Covenants
- Client-side validation
- Getting real-world data into blockchains: stablecoins, oracles, and bridges

Example: P2SH 1-of-2 multisig

scriptPubKey	scriptSig
OP_HASH160 748284390f9e263a4b 766a75d0633c50426e b875 OP_EQUAL	OP_0 <sig> <OP_1 <PK1> <PK2> OP_2 OP_CHECKMULTISIG>

Reveals script and alternate spending pubkey

What if I had multiple script spending paths?

Idea: Taproot. Bitcoin upgrade in 2021

Taproot: motivation

- Regular key spends and script spends look quite different onchain
- Script spends always require revealing the script
- Perhaps many contracts fit into the following model:
  - Known set of participants who are online
  - Only need to use script in times of disagreement; otherwise all sign how funds should move

Taproot

- Taproot spending: a key and a script. Two ways:
  - Key path
  - Script path
- Soft fork to Bitcoin in 2021 with three parts:
  - Schnorr signatures (BIP 340)
  - Merklized Abstract Syntax Trees (BIP 341)
  - Tapscript (BIP 342)

Simplified Taproot

Script: m, public key P=xG
Send to key Q:
Q = P + H(P || m)*G

Options for spending:

1. Key path: Sign with private key q = x + H(P || m)
2. Script path: a) Reveal (P || m), script m, and script arguments b) Run the script

Aggregated signatures

- Recall: Schnorr signatures; musig2
- Multiple parties interactively produce one public key and one signature
  - Looks like a regular one-key signature on-chain
  - Cannot distinguish between this case and the one party case

MAST: Merklized Abstract Syntax Tree

- Might want to have a big “OR” of functionality in your script
- Commit to tree of potential scripts that could be used to spend
  - Specify Merkle root of script tree in output
- Reveal specific script at spend time
  - Also show Merkle path to leaf script to prove it’s in the root
- Keeps other, unused scripts hidden!

Putting it all together

Merkle root of scripts: mr, aggregate public key P=xG

Send to key Q:

$Q = P + H(P \| mr)*G$

Options for spending:

1. Key path: Can sign with aggregate signature for aggregate public key P
2. Script path: Reveal (mr, P), path to leaf in mr, leaf script, script arguments, and run the leaf script

Taproot pros/cons

Pros

- Indistinguishability between P2PKH and P2SH in the cooperative case
- Less on-chain storage
- Privacy of unused script options
- Upgradeability in Tapscript

Cons

- Overhead over raw P2SH
- Complexity

Future Bitcoin script upgrades: covenants

- Currently, script specifies conditions for information needed to spend an output
- Currently no way to further restrict *how that spend looks* (what the spending transaction looks like)
  - No real transaction introspection
- Covenants add this functionality
- Why? Way of adding richer programmability to Bitcoin!

Examples

- Spending transaction can only spend to certain addresses
- Spending transaction must preserve some structure (amounts, further spending conditions)
- Use cases:
  - Vaults
  - Congestion control
  - Lightning network optimizations (future class)
  - Generally, gets us to way richer scripting functionality

Current proposals for covenants

- OP_CHECKTEMPLATEVERIFY
- OP_TXHASH,OP_CHECKSIGFROMSTACK
- OP_CAT

Covenant concerns

- “Recursive” covenants: can restrict how a UTXO can be spent in perpetuity
  - Can already simulate this with multisig…
- Pandora’s box in terms of adding scripting to Bitcoin
  - Storage
  - Processing
  - More DeFi / fewer money use cases?
- This debate is happening right now!

Covenant-like functionality without a soft fork

- New opcodes require a soft fork
- There's a lot of debate about whether they're even a good idea
- Efforts to provide additional functionality without a soft fork:
  - Colliderscript
  - Functional encryption

Programmability concepts

Want to **agree** on the **execution** of a program

- Could execute the program on the blockchain (smart contracts)

Programmability concepts

Want to **agree** on the **execution** of a program

– Could execute the program on the blockchain (smart contracts)

- Scalability: everyone executes everything
- Privacy: everyone can see inputs, results, execution
- Limits: blockchain might not support required functionality

Programmability concepts

Want to **agree** on the **execution** of a program

- Could execute the program on the blockchain (smart contracts)

Programmability concepts

Want to **agree** on the **execution** of a program

- Could execute the program on the blockchain (smart contracts)
- Could use the blockchain simply for commitments to data (client-side validation)

Client-side validation

- Blockchain commits to some state transitions and data
- Clients are responsible for
  - (1) obtaining/storing the preimage to that data, if needed
  - (2) validating the commitments are to correct state transitions
- The blockchain will *not* check execution for you!

Bitcoin examples

- Colored coins (2013)
- Omnilayer (used to be how USDT was issued - 2014)
- Today: Ordinals, Runes, Taproot Assets (2023 onwards)

Example: Omnilayer

- Goal: Issue arbitrary tokens on top of Bitcoin
  - Mint, transfer, redeem
- Tether was initially issued as a client-side validated protocol on Bitcoin called Omnilayer (2014-2023)
- Idea: store metadata on token minting, transfers, and redeeming in `OP_RETURN` data
- Issuer promises to follow what the Omnilayer protocol on the blockchain says when processing redemptions

Omnilayer in detail

Txid: 0548df Idx: 1 scriptSig	Value scriptPubKey
Txid: 1c2ef Idx: 2 scriptSig	0 OP_RETURN 6f6d6e690000000 000000001f000000 002faf0800

- Not input/output based; just instructions to the Omnilayer protocol state machine
- Piggybacks off of Bitcoin addresses
- No validation (beyond the Bitcoin transaction) from the Bitcoin network!
- `OP_RETURN` outputs are no longer spendable; not stored in the UTXO set

Client-side validation pros/cons

Pros

- Scalability
- Privacy
- Not limited by script

Cons

- Users have to store data
- Worse threat model than SPV
- “Pollutes” the blockchain with arbitrary data (80 byte `OP_RETURN` limit)

Integrating the real world with the blockchain world

- Real-world assets
  - Stablecoins
- Oracles
- Bridges

Backed stablecoins

- Stablecoin **issuer**
  - I give them $1 in the real world, they give me one digital **token** representing that $1 on the blockchain
- Most widely used stablecoins today: USDC by Circle ($74B) and USDT by Tether ($182B)
- Primarily use US Treasuries (or treasury repos) for backing

visaonchainanalytics.com/supply, accessed Oct 14 2025

Stablecoins on Ethereum: ERC-20

- An API for fungible tokens
- Smart contract mapping address->balance
- Interface (simplified):
  - `transfer (address to, uint256 amount)  bool`
  - `totalSupply()  uint256`
  - `balanceOf(address owner)  uint265`
- Additional functionality:
  - `mint(address to, uint256 amount)`
  - `burn(address from, uint256 amount)`

ERC-20

- Can use this interface for any fungible token you might want to issue
- Issuer/backing model works for other assets besides dollars: Euros, gold, oil, etc

Stablecoins on Bitcoin?

- Currently only via client-side validated protocols:
  - Omnilayer (no longer really in use)
  - RGB
  - Runes (BRC-20)
  - Taproot Assets
- Future: rollups (next lecture)

Backed stablecoin concerns

- Relying on issuer to maintain backing
  - No way to cryptographically prove backing
  - Historically, audits have been inadequate
- Issuer can freeze accounts or refuse minting or redemptions
- Issuer can rewrite balances (though this is auditable on-chain)

Decentralized “stablecoins”

Question: Can you decentralize the issuer?

- Then who holds the backing?
- Do you need real dollar-based backing?
  - MakerDAO and DAI (over-collateralized)
  - Terra/Luna (blew up in May 2022)
  - Ethena (recently depegged on Binance)

Integrating the real world with the blockchain world

- Real-world assets
  - Stablecoins
- Oracles
- Bridges

Oracles

- Essentially a pre-specified key that inputs signed data into the blockchain
- Uses:
  - Pricing
  - Information on real-world events
    - Prediction markets: election winners
    - Insurance: weather, etc
- Decentralized oracles: many keys; maybe voting

Bridges

- We live in a multi-blockchain world… How do you transfer assets from one blockchain to another?
  - Lock or burn on one, create on another
- Atomic cross-chain swaps not really used in practice
- Bridges
  - Multisig (similar to trusted issuer model)
  - Proof systems – smart contract validates execution on other blockchain

Summary

- Bitcoin programmability: script -> Taproot -> ???
- On-chain vs. client-side validation
- Where blockchain programmability intersects with the real world: stablecoins, oracles, bridges

Next week: scalability

Approaches:

- Vertical scaling
- Payment channels and Lightning network
- Sharding
- Rollups
  - Optimistic
  - ZK
- Guest lecture: BitVM and rollups