Exploring BitVM

Antoine Riard, Bitcoinology, February 2024

About: ariard

- bitcoin protocol hacker ~2018 (base-layer and lightning)
- managing partner @ thelab31.xyz (R&D / security consulting “boutique”)
- areas of research interest: protocol security and bitcoin scalability
  - cross-layer mempool issues (e.g mempoolfullrbf)
  - lightning: time-dilation, dust-inflation and pinning attacks
  - coinpool and payment pools research
- privacy note: no photo thanks

What the heck is BitVM ?

- a “new” computing paradigm to express Turing-complete Bitcoin contracts
- fundamental innovation”: logic gate commitment in today’s Bitcoin Script
- promises: chess games, validity proofs verifications, alt-chain bridges
- *disclaimer*: my understanding of the paper material only
    - it might be imperfect, paper not very complete

Bitcoin Script: a 101

- math ops: OP_ADD ; OP_SUB, OP_EQUAL
- crypto ops: OP_HASH160 ; OP_SHA256 ; OP_CHECKSIG
- control flow ops: OP_IF ; OP_ELSE ; OP_ENDIF
- *Forth-like* programming model (with stack), here since original client release
    - script interpreter run at each transaction verification by all ful-nodes
        - modulo performance optimizations (see “-assumevalid”)

Archeological note: ZKCP

- ZKCP = Zero-Knowledge Contingent Payment
- “Can you provide Y such as H(X)=Y ?” where H() computer program
- put Y as the haslock of an on-chain or off-chain swap contracts
- known since ~2011, ZK computation practical only since 2015
  - a nullc old idea

BitVM abstraction: bit value commitment

- create a commitment to bit value with dual OP_HASH160 and control flow

- script is the following:

```
OP_IF
    OP_HASH160 <0xdeadbeef>
    OP_EQUALVERIFY <1>
OP_ELSE
    OP_HASH160 <0xbeefdead>
    OP_EQUALVERIFY <0>
OP_ENDIF
```

BitVM abstraction: logic gate commitment

- create NAND operation from bit value commitment

- script is the following:

  OP_BITCOMMITMENT “left door” OP_TOALTSTACK
  OP_BITCOMMITMENT “right door” OP_TOALTSTACK
  OP_FROMALTSTACK OP_BOOLAND OP_NOT
  OP_EQUALVERIFY

BitVM abstraction: gate circuit

- decompose your program (e.g WASM stack machine) in a sequence of gates
- commit those N number of gates in a Taproot tree in a P2TR
- counterparties can execute the gates on-chain if disagree off-chain execution
- optimistic case: counterparties exchange data off-chain on committed program

A BitVM example: what is A XOR B

- Paul and Vicky wishes to agree on the XOR result of A and B
  - A and B two 1 byte random word
- Paul and Vicky pre-compute all XOR operations logical gates in circuit tree
- Paul and Vicky exchanges signatures for non-optimistic C&R transactions
- Paul and Vicky backs the XOR execution with a 1 BTC deposit
  - Paul or Vicky broadcasts the funding tx to start off-chain execution

A BitVM example: nth's bit "fraud"

- Vicky: "A XOR B does not equal A XOR !B but A XOR B !"
- Vicky runs off-chain the execution of A XOR B until finding *gate op N*
- Vicky: "Pauls shows me on-chain gate op N and its data input !"
  - "and you're better doing it fast after 2 weeks or I take your money"

A BitVM example: slash with equivocation

- Paul showed contrary bit on-chain / off-chain
- Vicky can equivocate on-chain by unlocking equivocation ability
    - she knows both *x0* and *x1* preimages commitment for gate X
- Vicky can broadcast a punishment tx and finish the C&R phase

BitVM challenges #1: circuit scale ?

- taproot tree size limit you can encode in PT2R (see bip341)
- witness growth scale with the circuit complexity
- witness merkle branches to be paid in case of C&R
- not all complex contracts on-chain might be for everyone economic user
  - valuable and interesting contracts might be limited to an economic minority…

BitVM challenges #2: fee fault-tolerance

- chainspace beefy witness for challenge-and-response (C&R)
- no guarantee of stable network mempool feerates during whole C&R phase
- counterparty have to provision lot of fee values
- malicious counterparty might trigger C&R at worst fee network times

BitVM challenges #3: “challenge” DoS

- ideally efficient sampling techniques to verify circuit execution in minimal steps
- ZK proofs techniques: bulletproofs / starks / folding schemes
- malicious counterparty might engage in *correct-yet-lengthy* verification steps
- moon maths more uncertain cryptographic breaks of high-value contracts
    - good to design cryptographic *honeypot* !

BitVM challenges #4: pre-signed sequence txn

- all the permutations of the circuit gates might have to be pre-committed

- factorial sequence of pre-signed transactions to generate

- computational barrier above a certain number of gates

- data / code separation covenants and cross-input fetch *might* solve this
    - ~ *reusable science rocket style* circa 2004

BitVM future: “minimum-valuable-contract” ?

- the BitVM design paradigm *works* on the whiteboard
- big uncertainty on the MVC and fee sats / computational costs on average user
- full bitvm toolchain hard challenge (e.g *harder* than lightning ?)
- trust-minimized “join five” or “dice bet” sounds first viable apps

Thanks to Bitconology!