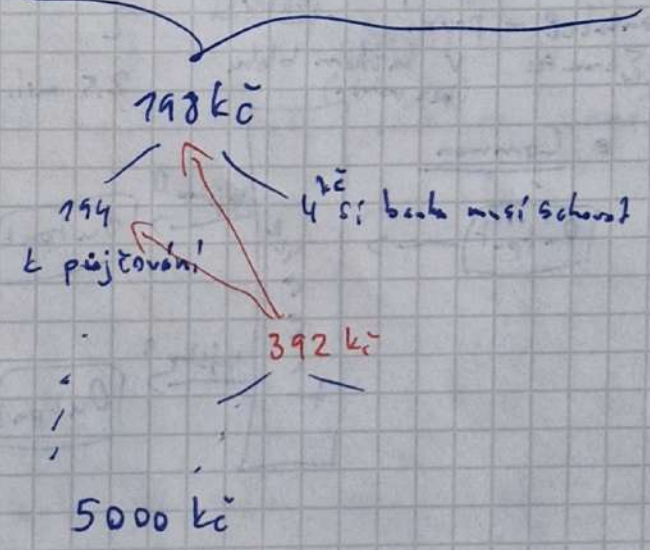
Vznik peněz

- centrální banka - pár dělků to zařizuje
- komerční banky - z půjček
  - musí mít povinné finanční rezervu (v ČR 2%)
    - z toho, co půjčuje

vložíme 100kč do banky: 2kč si schová
98kč může půjčit

⇒ my máme (vloženo) 100kč, někdo si půjčil (těch) 98kč



Bitcoin

→ distribuovaná databáze transakcí

↑

— organizované v blocích po ~1000 transakcích

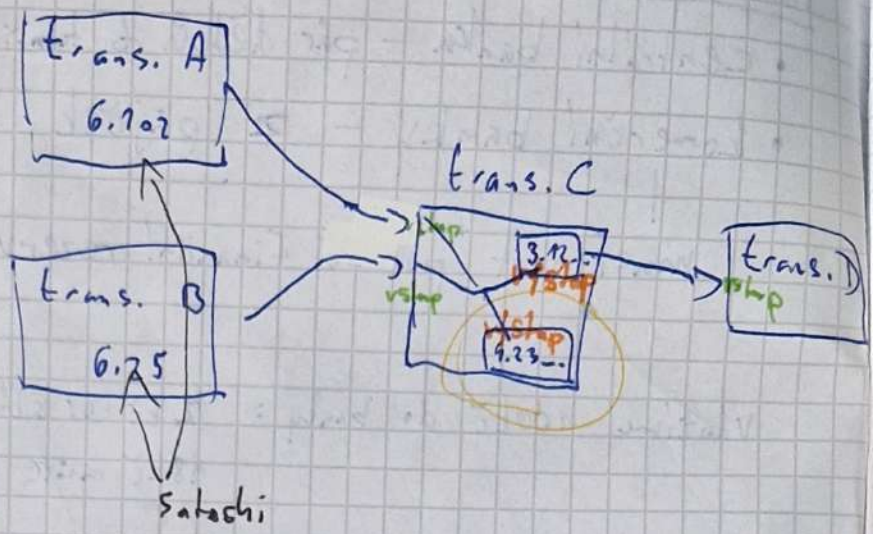
- proof-of-work - potvrzení každé změny v té db
  - → hash s určitými vlastnostmi
  - → CPU → GPU → (FPGA) → ASIC (application-specific integrated circuit)
    - ↑ HW-zadrátovaný alg. té hash. funkce
- v peněžence (klíčence) máme privátní klíče, které nás oprávňují k těm BTC

Transakce

UTXO = Unspent Transaction Output

- nedělitelné

vždy celý výstup je vstupem další transakce



1 BTC = 100 000 000 sat

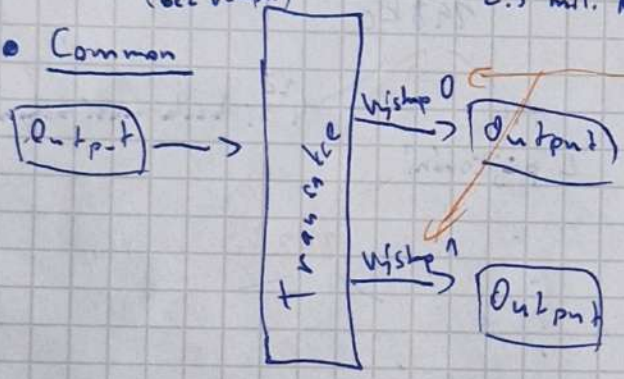
2

2.5 mil. Kč

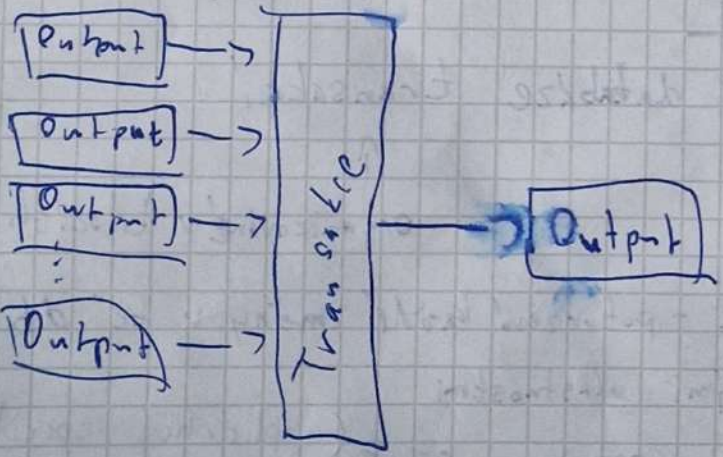
Coinbase transakce – první – těžaři
- block fee + Σ trans. fee v každém bloku (bez vstupu)

typy:

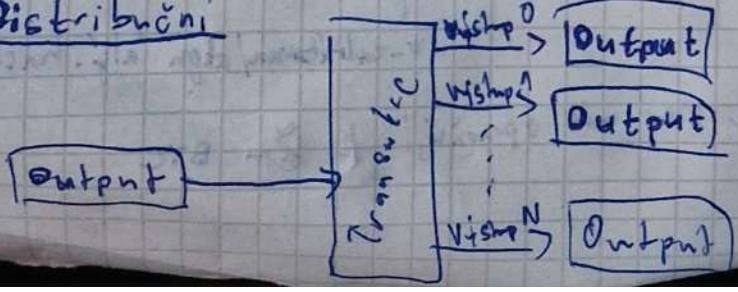
- Common



- Agregační



- Distribuční



Jak vypadá transakce

- vstupy:
  - id transakce + čísla výstupu, odkud se ten vstup bere
  - podpis, že na to máte právo (máte přístup k tomu priv. klíči)
- výstupy:
  - hodnota
  - scriptPubKey (v jazyce Script) (hash) → podmínka na to, aby to mohlo být použito jako vstup nějaké další transakce

v mempoolu jsou validated transakce, ale ještě ne confirmed

↑ Syntax, rozmezí hodnot, ...

blockchain – tak dlouho se mění nonce, aby po dvojnásobném zahashování hodnota odpovídala hodnotě difficulty target (byla menší než)

Bloky

id = Sha256(Sha256(header))

identifikátor předchozího bloku

1. blok – genesis block (2009) – jen coinbase transakce

1. blok s transakcí (#181) – Satoshi → Hal Finney

výška bloku = pořadí toho bloku v blockchainu

v každém bloku: difficulty target + nonce ← jednou za 2016 bloků se spočítá, jak rychle byly vyřešeny, a nastaví se diff. targ. tak, aby to trvalo kolem 10 min

Těžba (XXe-mily – nic nestojí ⇒ spam)

mění se nonce, dokud hash < diff. target

→ když je publikovaný blok, tak to všichni zkontrolují a když je to ok, tak si ho zařadí do svého blockchainu

za každých 210 000 bloků (~4 roky) se udělá halving = zpolovení se odměna za vytěžený blok

Vlastnictví bitcoinu

= schopnost s ním nakládat – mít privátní klíč (256b), kterým můžeme podepisovat transakce s ním

priv. klíč ~> (jednoduché, el. křivka: priv. klíč * G) veř. klíč —hash→ adresa !

(zpětně) nejde efektivně (diskr. log.)

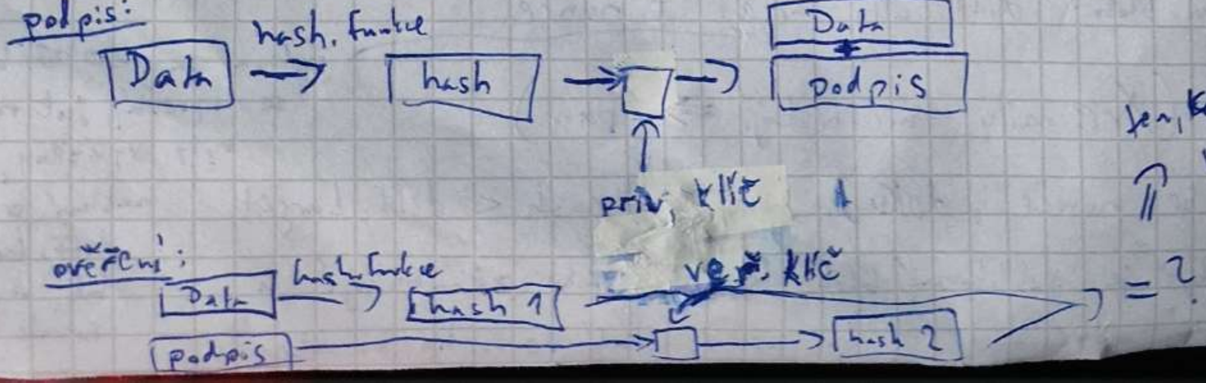
ty klíče jsou v peněžence (klíčence)
- sw
- HW !! lepší zabezpečení

master key pro "walletku"

recovery seed
- 12/24/20 slov
- Shamirův alg. – rozdělí to na víc lístečků a k obnovení stačí třeba 3 z 5 (viz později)

Digit. podpisy

- Elipt. křivky – tečna v lib. bodě protne křivku právě v jednom bodě, inverze je složitá
- Schnorrovy podpisy (v BTC systému od 2021) – lineární



⇒ ten, kdo to podepsal, vlastní přísl. priv. klíč

Hash. funkce

- SHA 256
- RIPEMD 160 — na adresy — 256 ~> 160b

(veř. klíč → SHA256 → RIPEMD160 → encoding → adresa)

serializace (encoding)

- Base 58 check ——> adresa
- Bech32 ~> samooprav. kódy

jazyk Script

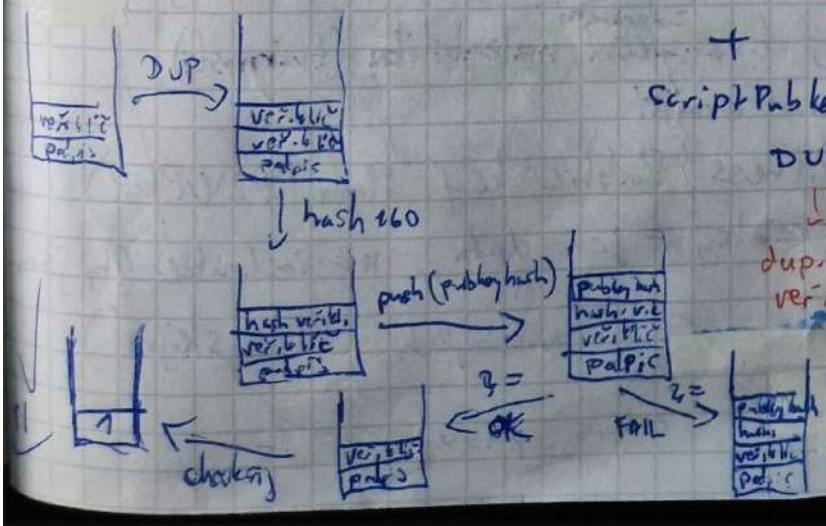
- zásobníkový (2 zásobníky)
- ne cyklus! - úmyslně není Turing complete
- speciální crypto instrukce - hashování, kontrola podpisů
- validní když top ≠ 0 nebo je zás. prázdný

Podpisová schémata

- pay to public key hash (P2PKH)
  - vznik: vygeneruje se `DUP HASH160 .:. EQUALVERIFY CHECKSIG` do položky scriptPubKey (pubkey hash)
  - vstup: v položce scriptSig je veřejný klíč a podpis
    - podpis: veř. klíč podepsaný priv. klíčem

při kontrole validity:
odem. script: podpis + veř. klíč → zás.
+
scriptPubKey
DUP HASH160 ~ EQUALVERIFY CHECKSIG

- DUP: dup. veř. klíče
- HASH160: hash veř. klíče
- EQUALVERIFY: kontrola
- CHECKSIG: kontrola, zda podpis je podepsán tímto veř. klíčem



• Pay to Multisig (P2MS)

2 veř.klíč A veř.klíč B veř.klíč C 3 CHECKMULTISIG

↓ kolik platných ověřených podpisů musí být

\#

ale NE — velké složité skripty ⇒ "drahé"

timelock — absolutní čas
— rel. - váže se na # vytěžených bloků

měřeno v blocích

př.: • 2/3
• 1/3 + právník + timeout — např. výstup je možné použít až za 30 dní
• právník + velký timeout

TIMELOCK — smart contract

• Pay to Script Hash (P2SH)

— místo skriptu se tam dá jeho hash

1) ověří se ten hash → • HASH160 hash EQUAL ← plátce (scriptPubkey)
2) pushne se ten skript na ty podpisy → • podpisy skript ← příjemce (scriptSig)

⇒ jednodušší, menší transakce + privátnost :)

↑ legacy schémata

• Segwit (Segregated Witness)

— rozšíření o možnost umístit podepisovací data jinam

5=5+1-1 ... "jiný hash" ⇒ FUJ

⇒ škálovatelnost, zabránění transakcím malleability (tvárnosti), menší transakce ⇒ levnější

↳ • Pay To Witness Public key Hash (P2WPKH)

ScriptPubkey - jen data - ne instrukce (ty jsou vždy stejné)

↑ publicKeyHash

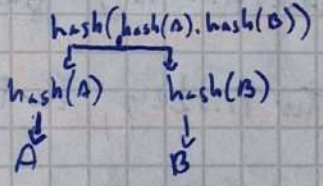
místo scriptSig je nově použito witness - podpis, pubkey

• Pay to Witness Script Hash (P2WSH)
- taky jen data
ALE ještě locking script byte code

tedy: • scriptPubKey: script hash
• witness: podpisy | locking script byte code

Schnorrovy podpisy
- lineární (X ECDSA)
→ můžeme sčítat veř. klíče a podpisy ✓✓
→ menší multisig transakce + větší abstrakce (privátnost)
→ levnější

Merkle Tree
- binární hash. strom
- data v listech → hashe → zřetězování + hash...



→ jde ověřit, že něco je součástí toho hashe ✓✓

Merkelized AST z toho scriptu → EUT

MAST

Merkelized Alternative Script tree → ✓

- ifové alternativy v listech
→ hashe zřetězených alternativ
- většinou jedna happy path - nejpravděpod., nejblíže vrcholu
=> nevyvážený

Taproot

- rozšíření BTC → rozšíření Scriptu
- MAST + TapScript + Schnorrovy podpisy
- nové adresové schéma: Pay-To-Taproot (P2TR)
  - key path – stejně jako P2KH, ale se Schnorr. podpisem (více klíčů)
  - \+ (volitelné) alt. skripty v MASTu
  - tweak = hash(Pubkey + MASTroot)

⇓ tweaked public key (scriptPubkey) = Pubkey ⊗ tweak (viz dále)

utracení: to_pub(tweak(priv. klíč)) = tweak(pub key)

Schnorr_sig(Tw Privkey, Tw Pubkey) do witness položky

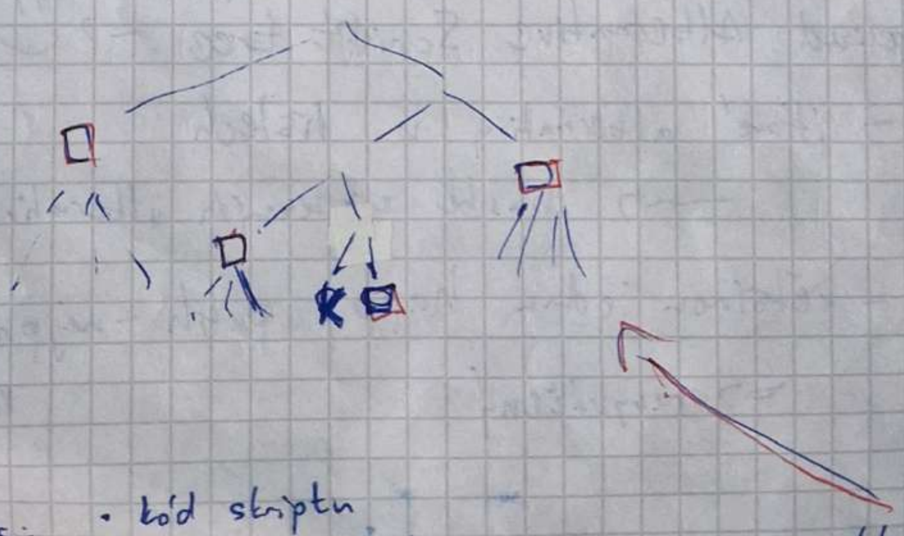
scriptPubkey: Pubkey ⊗ tweak (tweaked public key)

tweak = hash(Pubkey + MAST root)

witness:
- happy path: Schnorr sig pomocí priv. klíče
- script path: podpisy, skript, control block
  - skript – z listu Merkle tree – ten, který se použil
  - control block ⇒ s tou Merkle path

Merkle path (proof)

= Dk. že nějaký list je v Merkle tree bez zveřejňování ostatních



witness v utrácení:
- kód skriptu
- vstupy skriptu
- control block → tam je ta merkle path

Bitcoinová síť

- Full node client – obsahuje celý blockchain (~ ↑ TB)
- SPV wallet – spíše mobilní zařízení, dělají verifikaci
  - mají jen headery transakcí, které se jich nebytují
  - celé mají jen ty své
- mining node
  - 2010 první pool – všichni těží, když někdo vyhraje, tak se rozdělí
    (Slush/Braiins)

Stratum protocol

Jak dostat k sobě jen ty transakce, které nás zajímají

→ Bloomovy filtry – adresy dám do Bloom filtru → pošlete mi ↑ dostanu příslušné transakce
  - ale analýzou blockchainu se dá odhalit identita !!
  +
  možné DoS ⇒ nody to vypínají nebo dovolují jen některým

→ Compact Block Filters – na straně serveru
  - ∀ server si nějakým filtrem prožene hashé ∀ transakcí

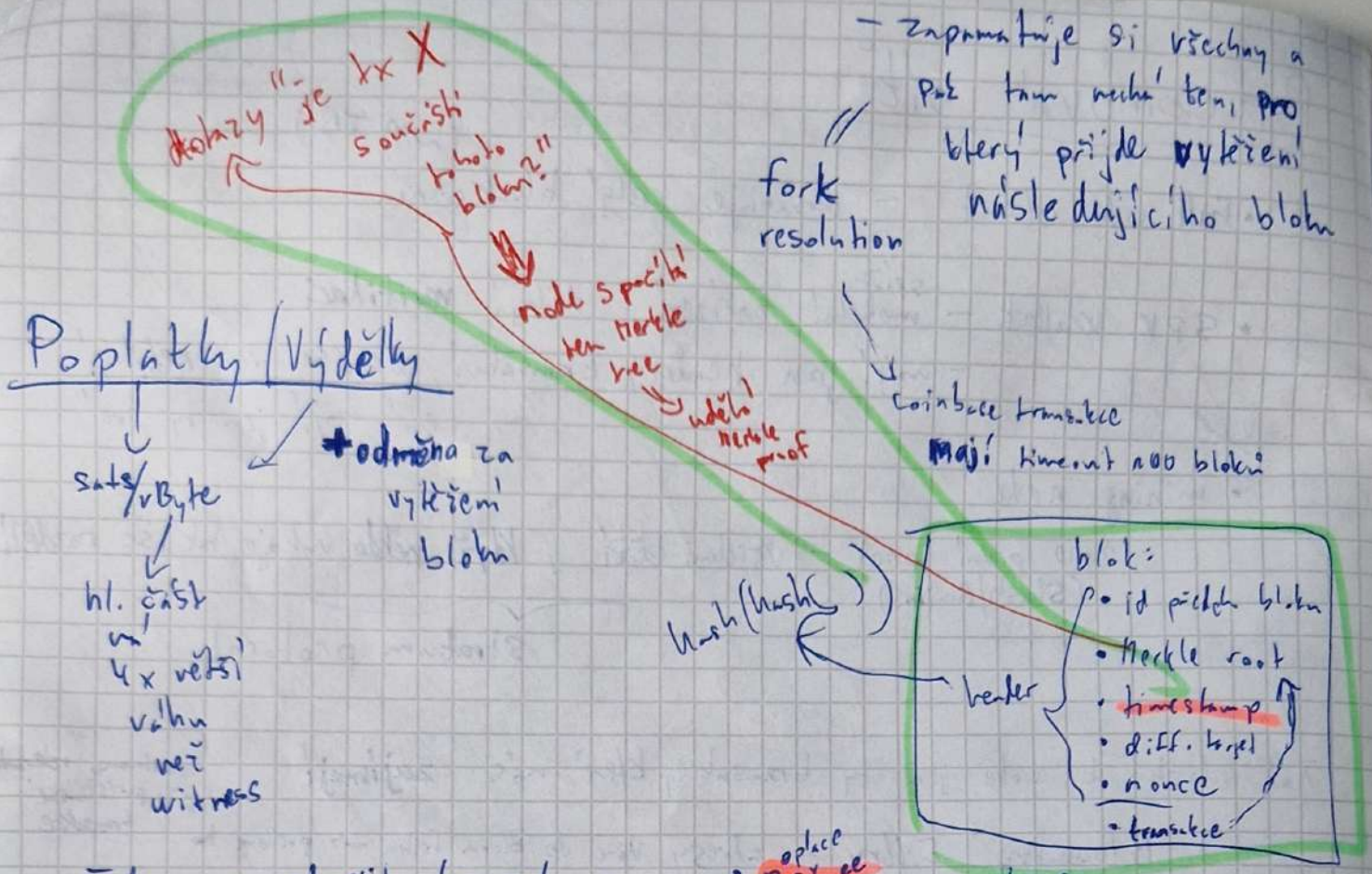
"Bloom filtry" (GCS)

Full nody si pro každý blok spočítají mapování adresy → bloky
⇒ to si od nich klienti mohou stáhnout, vybrat si z bloku ty pro ně zajímavé bloky a ty si (celé) vyžádat

klient má: • headery ∀ bloků
• headery filtrů

Konsenzus

1) klient chce vyrobit transakci – vyrobí a pošle okolním uzlům
2) Full node když obdrží transakci verifikuje a dá ji do mempoolu
3) těžaři – z mempoolu si vyberou transakce, které se jim líbí, a dají si to do candidate blocku a těží!
4) Full nody si přidají vytěžený blok do blockchainu – ale když víc těžařů stihlo najednou vytěžit nějaký blok, tak se to musí řešit:



– zapomatuje si všechny a pak tam nechá ten, pro který přijde vytěžení následujícího bloku

fork resolution

Coinbase transakce mají timeout 100 bloků

Jak popostrčit transakci:

- RBF (replace-by-fee)
  - dokud není trans. vytěžená, tak ji lze nahradit novou s větším poplatkem
- CPFP - Child Pay For Parent
  - výstup transakce použijeme v jiné s vysokým poplatkem

Útoky – žádný útok na prok. zatím nebyl úspěšný

- 51%-útok
  - kdyby někdo vlastnil většinu těžebního výkonu
    → může sabotovat fungování protokolu ⇒ těžit si bloky do zásoby ⇒ double spend (zahodí se starý chain, kde se to stratilo, a nahradí se novým, kde to je ještě nestrácené)
  - ale hroooodně energeticky náročné
    - =security budget problém
    - ale v budoucnu bude klesat ta odměna těžařům ⇒ méně těžařů ⇒ útok míň náročný?
  - blokování adres
    - takže prakticky snad nejtěžší nejde

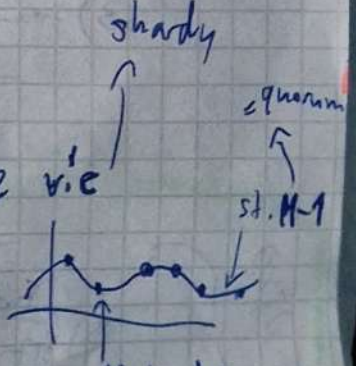
- útoky na burzy, směnárny, ...
  - časté a úspěšné !! (např. 2025 - Bybit - ale vedoucí to zaplatil :))
  - → nikdy tam dlouho nenechávat větší prostředky !

Peněženky (klíčenky)

- hierarchicky determ. odvozování klíčů
  - ↳ vygeneruje se 128b seed → master key
  - priv. klíče
  - veř. klíče
  - adresy
- vytváření, posílání transakcí, přijímání vytěžených bloků
- např. • Trezor, Ledger, Bitkey, ...
  - SW: aplikace na mobil, rozšíření prohlížeče, ...
  - HW
  - SW: open-source :)
  - Papírové - zastralé (dříve)
  - na burzách (z nich jsem vždy dokončil první čtyři písmena)

úrovně:
- účel - typ adresy
- typ měny (BTC, ...)
- account (podle zdroje těch BTC)
- receiving/change

- seed backup ! - 12/24 angl. slov
  - → passphrase
  - → vícedílná záloha rozšifrováním papírku se seedem - NE !
  - → Shamir backup - místo 1 seedu jich je víc
    - poly n-tého stupně je určen n+1 body
    - 1024 slov (→ 10 bitů) - 1 shard má 20 slov



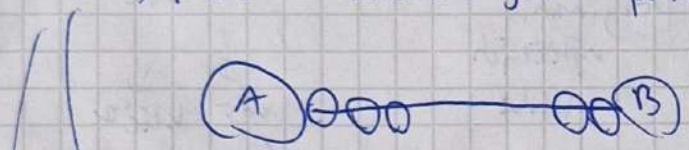
  - → Super-Shamir - skupiny Shamirů
  - → Uber Super-Shamir - skupiny Super-Shamirů
  - → SSKR (Sharded Secret key Reconstruction)

Škálovatelnost BTC

- asi jen 7 transakcí/s !!
- nebudeme do blockchainu dávat všechno hned, ale budeme si to psát někam stranou, a až pak to celé tam dáme najednou

Lightning Network

- kanály, kde transakce jsou privátní – do blockchainu se dá až výsledek



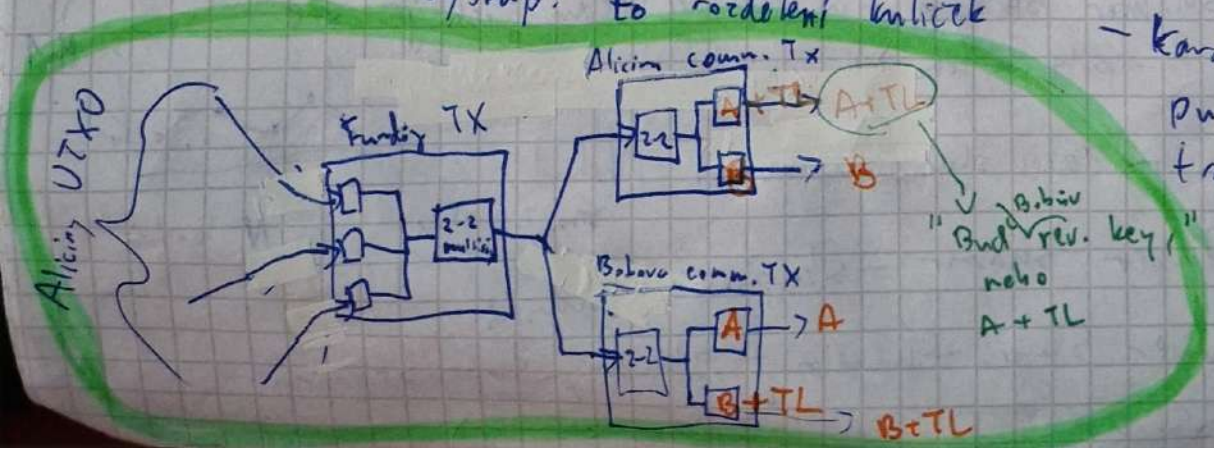
první dám kap. (#kuliček)

2 transakce: • funding – reprezentuje kap. kanálu – jedna strana tam uzamkne nějaké BTC v téhle transakci, kterou podepíší obě strany kanálu

1) handshake A ⟷ B
2) Funding transakce – 2-of-2-multisig adresa
   – ještě se nezveřejňuje – až když druhá strana podepíše comm. tx

• commitment – reprezentuje rozdělení prostředků mezi těmi stranami

3) commitment transakce
   – vstupem je výstup funding transakce
   – výstup: to rozdělení kuliček
   – kanál se ukončí publikováním této transakce do blockchainu



Routování v LN

- příjemce vygeneruje preimage => hash(preimage) -> LN faktura
- onion routing
  - Alice připraví (více alternativních) cestu:

A -> B -> C -> D

next-hopy "orci ti poslat"

$P_B(\to C, P_C(\to D, P_D(msg)))$

(BTW po rozbalení své části pak dá uzel do nové zprávy na konec padding -> nejde poznat podle délky, že už je to blízko konce :))

HTLC
- pošlou se ty BTC, ale:
  - je timeout - pak se kdyžtak vrátí
  - druhá strana si je musí odemknout pomocí t.ho preimage
- příjemce zpět pošle preimage
  -> odemkne se a odhalí se preimage pro toho předchozího
  -> -> -> zase a zase a zase

Watchtowers

- hlídají za mě, jestli druhá strana nepublikovala tu starou comm. tx, a kdyžtak udělají tu punishment tx
- např. Eye of Satoshi

Otevření a zavření kanálu

- channel forever
- mutual agreement – kooperativní
  - jako commitment trans.
- forced closing
  - publikace poslední comm. transakce
    → ale "velká" fee
- při podvádění protistrany
  - publishne starou comm. transakci
  → veškerá kap. kanálu jde na stranu toho poškozeného

Platby

- basic – faktura vytvořená přijímajícím
  - ale lze zaplatit jen jednou
  - má krátkou expiraci ≈15 min !!
  → ne donaty, dýška, ... !!
- LNURL
  - nadstavba nad zákl. protokolem
  - např. LNURLp – scan → https → za pomocí ext. webserveru se statická informace přivede na dyn. (QR kód)
    - LNURLw
    - LNURL PayToAddress
  - BOLT 12
  - přímo zabudované v LN a nepotřebuje webserver !!

Challenge

- vlastní LN uzel
  - channel management, rozdělení liquidity ⇒ náročné
  - fees
  - problémy s velkými platbami – byt, dům, ...
- custodiální walletky – mobilní aplikace
  - jednoduché, spolehlivé (ten provozovatel má roztažené tlusté kanály po celém světě)
  - ale musíme tomu věřit

hledá se něco mezi tím – jednoduchost ale zároveň důvěra

⇒ • reaktivní bezp. model
– sice kontrolujeme, co se děje na blockchainu, a když se tam dělo něco v rozporu s protokolem, tak zasáhne

• stateful model
- předem podepsané řetězky transakcí

Škálovatelnost - L2

Ark protokol

- klient - server (~~BTC~~, LN)
  - Ark Service Provider (ASP)
    - koordinuje akce klientů
- non-custodial, trust-minimized
  - "prostě můžeme kdykoliv odejít" ☺
  - "potřeba důvěry v ten server je nízká" ☺
- VTXO
  - → strom předem podepsaných transakcí
  - vezme ty svoje a publikuje je
- Kolo (round)
  - periodicky - cca každou hodinu (6 bloků)
  - vždy se vytvoří transakční strom
    - kořen on-chain
    - zbytek off-chain quad-tree
    - listy = VTXO
  - expirace za 30 dní
    - → je potřeba, aby ta peněženka byla aspoň 1× za 30 dní online
    - jinak si to Ark server vezme "pro sebe"
      - → do custodial wallet
      - a kdykoliv nám to vrátí (není technicky vynuceno, ale dělají to kvůli reputaci)
- Refresh
  - vidíme, že stromeček už brzo umře, ale my tam to naše VTXO chceme nechat
  - → forfeit transakce
    - = vzdáváme se práva disponovat s naším VTXO
    - transakce se spendí (condition jen s providerem)
      - ale vstupem bude ta stará VTXO A connector z nového stromu
  - chceme aby uživatelé dělali refreshe zodpovědně
  - pobídky
  - je kvůli nim potřeba extra likvidita - to VTXO je paralelně ve 2 stromech
- + ještě obrana - počkání na refresh
- - kdyby se někdo spolčil s ASP, tak by mohl udělat double-spend
  - → je tam dlouhý (~1 den) timelock, aby se to stihlo zjistit a do světa to rozhlásit
    - snadná detekce a dokonzistelnost ☺

Spark protokol

- ale větší potřeba důvěry!!
- transakce ověřují Spark Entity - operátoři, nelze mít jistotu, že klíče nezahodí !!

různé s různými / napůl rozdělený klíč, transakce = změna té naší půlky - výstup

ale asi ne :)

Liquid Network

- sidechain, zamkneme si nějakou likviditu a za to dostaneme nějaký ekvivalent
  - → L-BTC ↔ BTC (L-BTC kryté bitcoinem)
  - - běžná ~1 blok za minutu
  - místo Proof-of-work je Liquid Federation - potvrzují to dědci

Taproot Assets protokol

- edge nody - směňují různé měny za BTC
- hrozně složitá impl. - ještě nedokonáno

Ecash

- Cashu protokol → mincovna
- projekt Fedimint

v určených hodnotách - typicky mocniny 2

- uživatelé drží nějaké bankovky - to je podepsané ve walletách

- prevence double-spendu:
  - mincovna si pamatuje, které bankovky podepsala
  - když nám někdo něco pošle, tak si řekneme mincovně, aby ta bankovka zneplatnila a vydala nám novou ve stejné hodnotě → zapamatuje si ji jako utracenou

- blind signatures – BDHKE – aby mincovna mohla podepsat něco, co nevidí

MINCOVNA → priv. klíč k
+ veř. klíč $K = k \cdot G$

Alice → tajné číslo x
→ $Y := \text{hash\_to\_curve}(x)$

chce: $S = k \cdot Y$

+ další tajné číslo – blinding factor r
→ blind message $M' := Y + rG$

podepíše:
$S' := kM' =$
$= kY + krG$

unblinding:
$S = S' - rK =$
$= kY + krG - krG = kY$
→ podpis x :)

kontrola:
$S == k \cdot \text{hash\_to\_curve}(x)$

- multipath payment – experimentální implementace
  - máme mince z různých mincoven a chceme jimi něco zaplatit
  - → ty mincovny si to zařídí a půjde to :)

₿ a reálný svět

kde koupit

- burza (exchange office) – na principu nabídky a poptávky
  - ✗ – ale musíme se tam "svlíknout do naha" – doklady, ... (KYC) !! :((k.v.t. za kolik) + pozor na podmínky výběru (poplatky)
- směnárna (exchange) – služba – dopředu řekne ten kurz, likviditu si shání třeba na burzách
- bitcoinmat
  - vysoké poplatky
  - vyšší částky podléhají KYC
- Vexl – aplikace jako Tinder pro bitcoinery
  - reputační model na základě přátel a jejich přátel
  - kluby = komunitní skupiny

ekosystém

- atomické swapy – např. Boltz
- decentralizované půjčky – např. Firefish
  - 3-3 multisig viz slidy: payment oracle = banka + price oracle – kurz ₿ + EPM = ten, kdo tam dal ₿
  - princip: držet ₿ je dobré, ale je potřeba i něco platit
  - dáme do zástavy ten ₿, dostaneme peníze a ta cena ₿ naroste, takže se to zaplatí a pošlou nám to zpátky, jinak si to nechají
- služby za ₿ – Bitrefill → poukázky (třeba do Kauflandu), Travala (cestování)
- bridge mezi krypto a vnějším fin. světě – Bleap karta (bez poplatků :)) + i možnost spoření
- decentralizované pred. trhy – sázky na základě nabídky a poptávky

$ ETFunds (exchange traded) - zaměřené, automaticky spravované fondy

- nevlastníme hy $!
- in kind redemptions ✗ dříve: akcie → (vložené) prodáme → (daně) → koupíme $

↳ vyplacení hodnoty přímo v tom aktivu - není zdanitelné !!

→ boomeři, investoři, DIP, daňová optimalizace (např. na Slovensku, kde jsou zisky z kryptoměn hodně daněny, ale z ETF ne)
(jdeš tam to tlo koupit obecně ETF)

CBDC (central bank digital currency)

- opačné vlastnosti oproti $
- programovatelné - např. kdo neutratí aspoň 2000 $ -- za měsíc, tak musí platit daň
  + regule - např. množství benzínu, vína, --

Ethereum

- decentral. platforma, na které se dá počítat
- smart kontrakty
- měna Ether, nejm. jednotka Wei
  - vznikají a pálí se

Accounts

- Externally owned (EOA) - pub x priv. key → transakce
- Contract account - obsahuje smart kontrakt

→ adresa, nonce, balance, code hash, storage hash

Merkle Patricia Tree (Trie)

- Merkle + trie

→ key-value mapping s extension nody (kombinují společné prefixy do jednoho uzlu)

key i 32b → po čtyřech bitech

serializace pomocí RLP nebo SSZ

code hash → kód

u contract accounts nějaká data

Merkle Patricia tree

state trie - Merkle-Patricia trie

hash adresy → [nonce, balance, code hash, storage hash]

v databázi

storage hash → kořen Merkle-patricia tree, ve kterém je tento storage

RLP serializace (cc. prefix)

→ byty, stringy, listy

↓ 0x80 + délka a ten string

↘ 0xC0 + délka

SSZ serializace

- víme schéma ≈ struct
- variable-sized typy jsou pointery na konec, kde ta data začínají, offsety, kde to začíná

Transakce

= změna stavu světa - kdo má kolik Eth.

gas = unit of computation cost

- iniciovány jen EOA
- podepíše se priv. klíčem a všem broadcastne

from, to, podpis, nonce, hodnota, input data, gaslimit, ...

Typy:
- transfer
- contract deployment
- contract execution

Bloky transakcí

- nový každých 12s, proof-of-stake
- transakce strictly ordered

time slot, proposer (validátor), parent root (hash předch. bloku), state root (state trie), body

Ethereum Virtual Machine (EVM)

- stack-based jazyk
- transient memory ≈ heap
- persistent storage ≈ disk → v Merkle-Patricia tree
- high-level jazyk (Sidity/Vyper/..) → překlad

Gas

- base fee – spálí se
- priority fee – dýško pro validátory

Nody

consensus client + execution client

↓ Konsenzus, gossip, forky

↓ Spouštění transakcí

- full node – nedávná historie (a starší umí zrekonstruovat)
- archive node – celá historie – pro analýzy a debugging
- light node – jen headery

Proof-of-Stake protokol (PoS)

- validátor tam dá (32 Eth) stake
- ∀ 12 s: (= slot) → block proposer
  - vybere se jeden validátor (náhodně)
    → ten dá dohromady nový blok
    → committee to zkontroluje
  - hlasuje se o tom, jestli je ten blok správně
- ∀ 32 slotů: (= epocha → checkpoint) Finalizují se ty bloky
  - po 2 epoších už je ten blok daný a nejde s ním nic dělat (~~15 min~~)

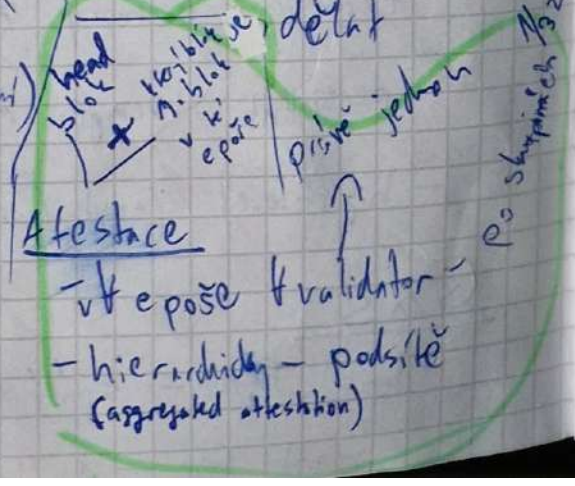
FUJ ⇒ slashing ⇒ Forced exit period (36 dní)
- víc bloků v jednom slotu
- hlasování vícekrát
- nekonzist. hlasování

Gasper

- alg. na fork choice a finalitu
  - ↓ podle největší váhy (dané #hlasů)
  - ↓ 2/3 hlasů

Atestace

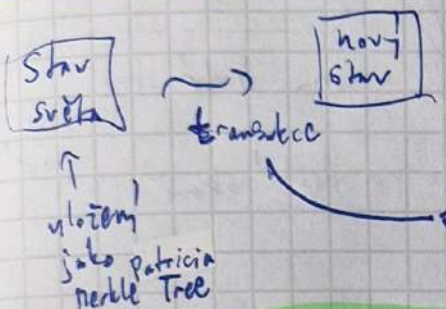
- v epoše 1 validátor → 1 atestace
- hierarchicky – podsítě (aggregated attestation)



Útoky

→ přeskládání bloků, blokování, double finality
- útočníci riskují ztrátu $ stakí
- social attacks

Smart contracts



- immutable kód, je transparentní, výsledek je deterministický

⇒ tokeny
- peníze
- rights
- collectibles (NFT, certifikáty, ..)
- věci do her, ..

Oracles

⇒ propojení s vnějším světem – reagují na události v kontraktu

publish-subscribe

request-response

- např. ChainLink – decentralizovaná oracle network

DApps

- decentralizované aplikace
= smart contract + UI

Web 3.0

- backend + databáze distribuované – třeba v Ethereu

Škálovatelnost – Layer 2 ✗ Sharding (onchain)

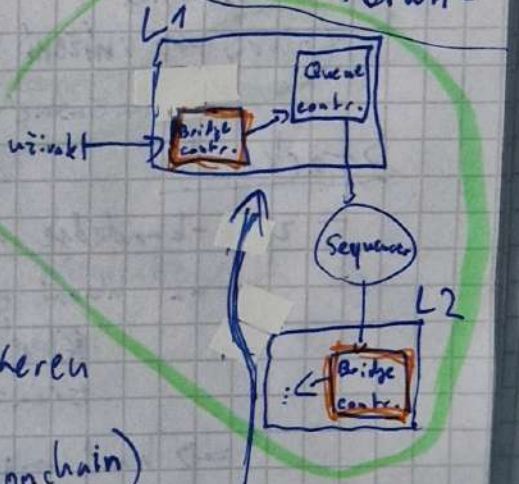
⇒ paralelní blockchainy
- už se nedělá

Rollups

- transakce offchain a pak se to celé pošle na blockchain

optimistic – transakce se posílají operátorovi a ten je pak submitne do Eth
- je challenge period (1–2 týdny) na kontrolu – fraud proving

zero knowledge – na hlavní chain se pak pošle jenom summary + proof, transakce se provádí na offchain VM
- single-round interactive – celé se to pošle a poromluví se výsledky
- multi-round interactive – postupně se to řeší



State channels

- jako LN
- když chce někdo vystoupit, tak po dobu nějaké period můžeme do blockchainu nahrát novější transakci

Sidechains (není L2)

- virtuální blockchainy s vlastními pravidly
- pomocí bridge připojeny k hlavnímu Eth. blockchainu

two-way

Bridge – třeba mezi BTC a Eth. (X L2)

- Lock-and-mint
- Burn-and-mint – na jedné straně se to spálí a na druhé straně vytvoří nové
- Atomic swaps – výměna na jedné straně za něco na druhé straně
  - např. liquidity networks

L2 Networks

Arbitrum One

- optimistické rollupy
- závěrečné transakce = settlementy

Optimism

- open-source stack na Optimistické rollupy

Base (Coinbase)

- optim. rollupy
- bez tokenů
- bloby → snížení fees
  = dočasná data (X calldata – navíc, drahé)

InK (Kraken)

- bez tokenů

Unichain (Uniswap)

- optimalizované pro central. DeFi nance

Starknet

- zero-knowledge rollupy
  - na hlavní chain se dává jen potvrzení pomocí hash
    kterého jde ověřit, že dané off-chain transakce proběhly
  - => hned můžeme využívat tokeny, které z toho máme (X optimistické – musí se určitou dobu čekat, jestli někdo nevznese námitku)

Linea

- taky zero-knowledge rollupy

Scroll

- zero-knowledge rollupy

zkSync Era

- taky zero-knowledge rollupy

Mega ETH

- high-performance → Web 3.0 :)
  → důraz na paralelismus, async. konsenzus

Problém data availability

→ jak nemuset na ten hlavní blockchain posílat všechno

- data avail. sampling
- data availability committees

Posílají se tam jen checkpointy

dědí bezpečnost Ethereum

L1 – jiné blockchainy

Solana

- proof-of-stake + proof-of-history ⇒ není potřeba složitě ověřovat (×× Ethereum) ⇓ rychlejší

  ↓ transakce mají (daný) čas

  $T(i+1) = sha256(T(i))$

  400ms sloty – výběr leadera, validátoři hlasují pro blok

  ↓ hlas pak nejde změnit po určitou dobu — stále se zvětšuje

- oblíbená platforma na meme coiny (nízké fee)

Monero

- důraz na bezpečnost, anonymitu
- RandomX algoritmus → CPU (×× jiné – GPU)

  PoW+

  ↓ decoye → transakci posílá jeden z množiny {sender, decoy, ...}

  ↓ posílaná částka se "šifruje"

  r... blinding factor

- ne smart contracty

Tron

- delegovaný PoS ⇒ high perf :), ale menší decentralizace :(
  - volíme si své reprezentanty a na ty to delegujeme

ChainLink

⇒ decentralizované oracle networky

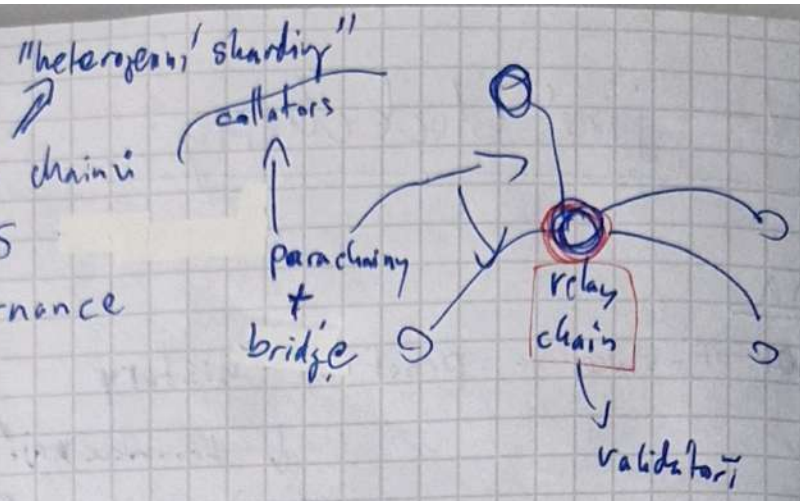
Cardano

- hodně staví na researchích
- PoS pomocí Ouroboros protokolu

  ↳ finalizace na základě nejdelšího chainu

- on-chain governance
- měna Ada

Polkadot



- několik paralelních chainů
- nominated PoS
- on-chain governance

Cosmos

- BFT + PoS
- dobré na cross-chain komunikaci

Kaspa

- PoW + BlockDAG

Gnosis

- sidechain Etherea
- základ pro spoustu stablecoinů

Memecoiny

- 2013 - Dogecoin

Stablecoiny

!!

- Fiat-backed - peníze
- commodity-backed - zlato, ropa, ...
- crypto-backed
- algorithmic-backed - cenu držíme na základě algoritmu, který se stará o vyrovnání
  - FUJ

DeFi

- V produkty nad blockchainem
- ENS (obdoba DNS)

Půjčky

- peer-to-peer - přímo od někoho
- pool-based - pool lidí, co dají k dispozici tokeny, a my si z toho můžeme půjčovat

- flash loans = v jedné transakci jde zařídit, že se to buď stane všechno, nebo nic

Pojištění

- např. v Keni

Crowdfunding

- ICO

DAO

- kolektivně vlastněné organizace
- např. charity, granty, ...
  - a hlasování, co zafinancovat

Kryptografie

G – generátor pubkey + privkey

S – m + privkey → podpis

V – m + pubkey + podpis → ověření

DSA

$p = qa + 1$ — prvočísla

⇒ diskr. log. těžký

Schnorrova grupa Z_p^* generovaná generátorem $g \in Z_p^*$, $g^q = 1$

→ G: • vybere $s \in Z_q^*$ → privkey

• pubkey $v := g^s \bmod p$

S: $h := hash(m)$, $r \in Z_q^*$ nonce

$y1 := (g^r \bmod p) \bmod q$

$y2 := (h + s \cdot y1)\, r^{-1} \bmod q$

podpis: $y := (y1, y2)$ → když něco z toho vyšlo 0, tak opakujeme s novým r

V: 1) ověříme, že $1 \in y1, y2 < q$

2) $y3 := (y2)^{-1}$

3) $((g^h v^{y1})^{y3} \bmod p) \bmod q \stackrel{?}{=} y1$

↖ jenom public věci ☺

ECDSA

– el. křivky

Schnorrovy podpisy