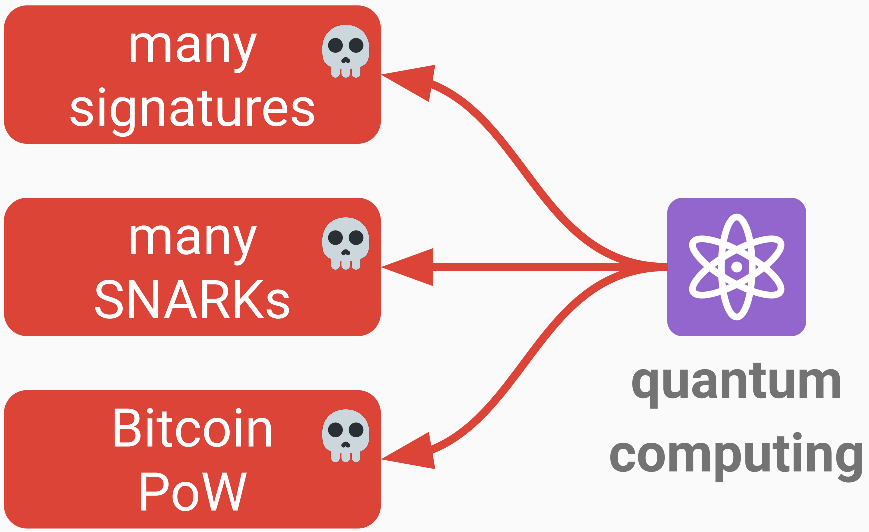
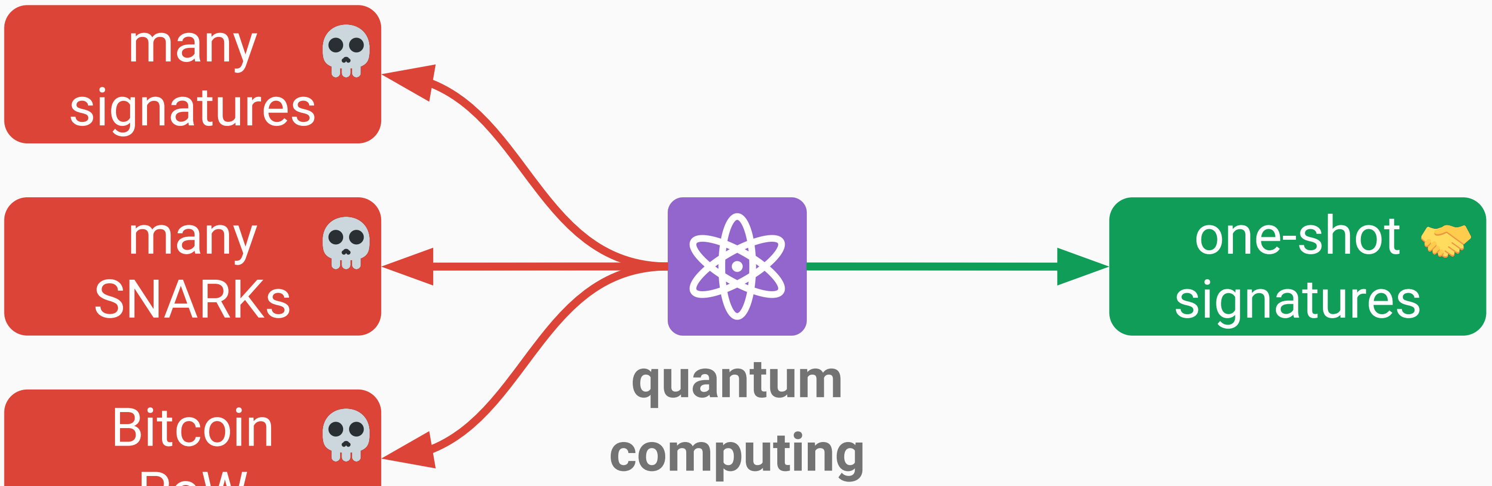
one-shot signatures

a new blockchain paradigm

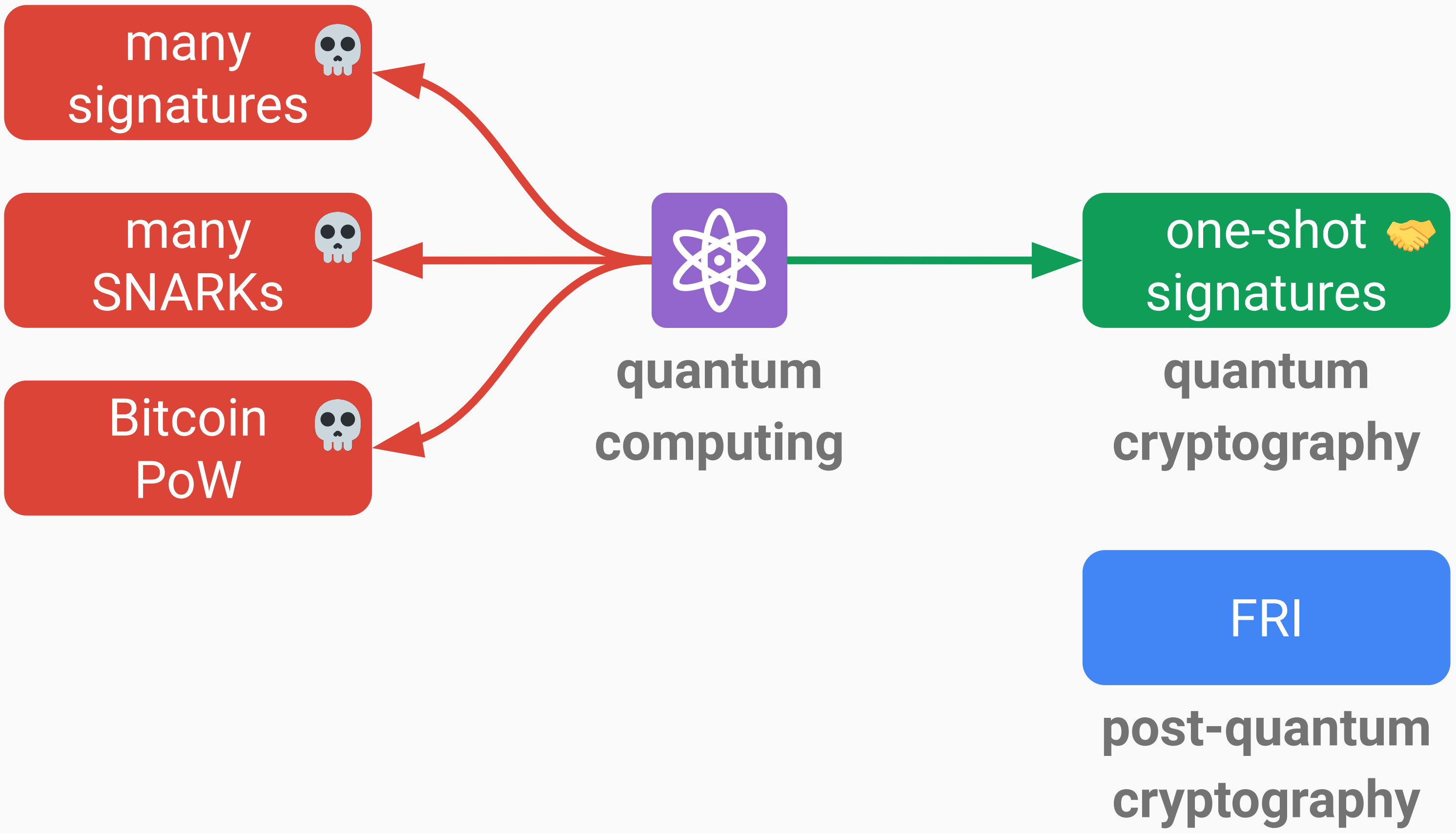
destructive vs constructive



destructive vs constructive



destructive vs constructive



part 1—signature chains

part 2—applications

part 3—accelerationism

part 1—signature chains

part 2—applications

part 3—accelerationism

February 2020 paper

eprint.iacr.org/**2020**/**107**

One-shot Signatures and Applications to Hybrid Quantum/Classical Authentication

Ryan Amos*[1], Marios Georgiou†[2], Aggelos Kiayias‡[3], and Mark Zhandry§[4]

February 2020 paper

revolutionary
and ignored

eprint.iacr.org/**2020**/**107**

One-shot Signatures and Applications to Hybrid Quantum/Classical Authentication

Ryan Amos[*1], Marios Georgiou[†2], Aggelos Kiayias[‡3], and Mark Zhandry[§4]

February 2020 paper

revolutionary and ignored

eprint.iacr.org/**2020**/**107**

One-shot Signatures and Applications to Hybrid Quantum/Classical Authentication

Ryan Amos[*1], Marios Georgiou[†2], Aggelos Kiayias[‡3], and Mark Zhandry[§4]

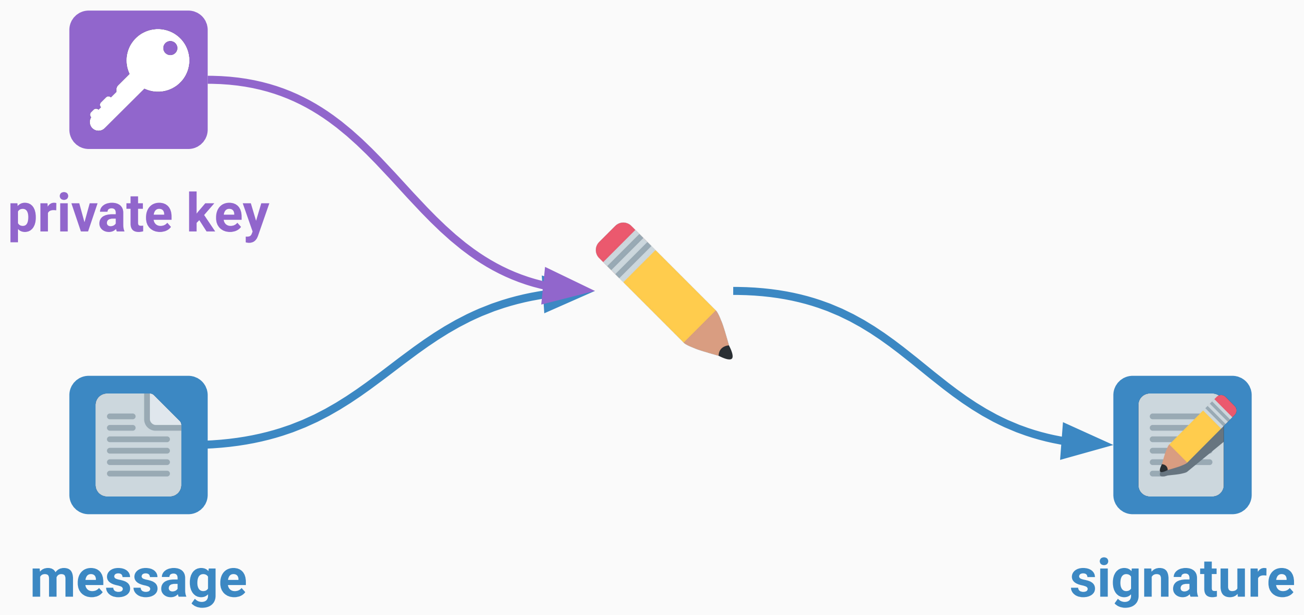
QSig

one-shot signing

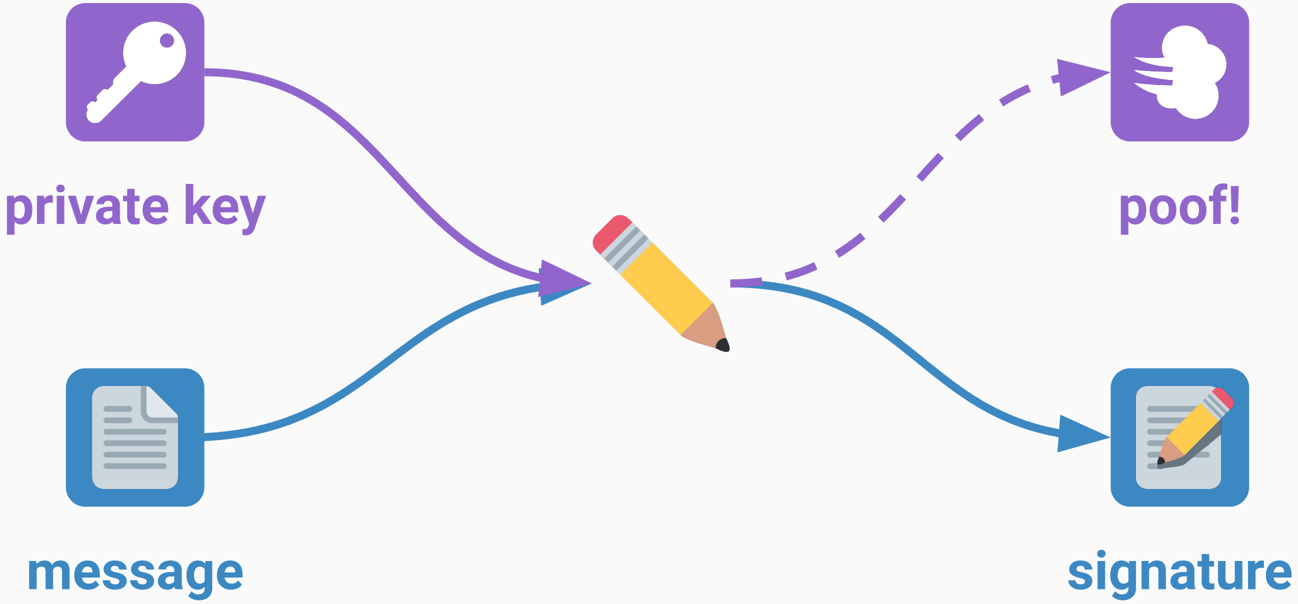
private key

message

one-shot signing



one-shot signing

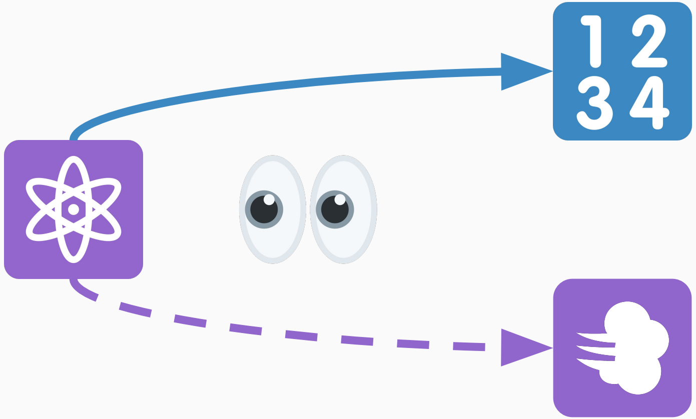


quantum principles

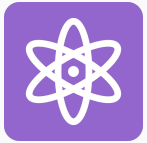
destructive measurements

no cloning

destructive measurements

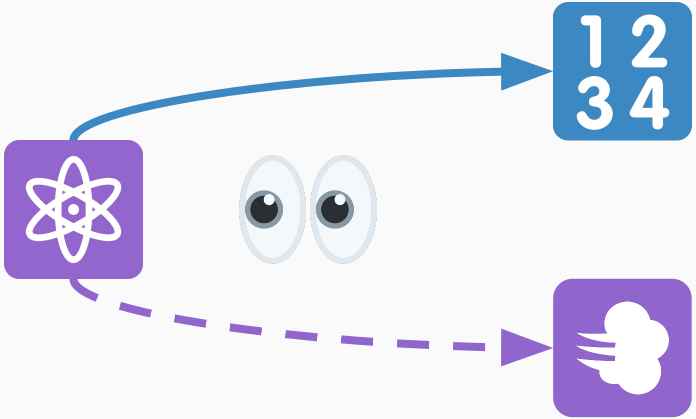


no cloning

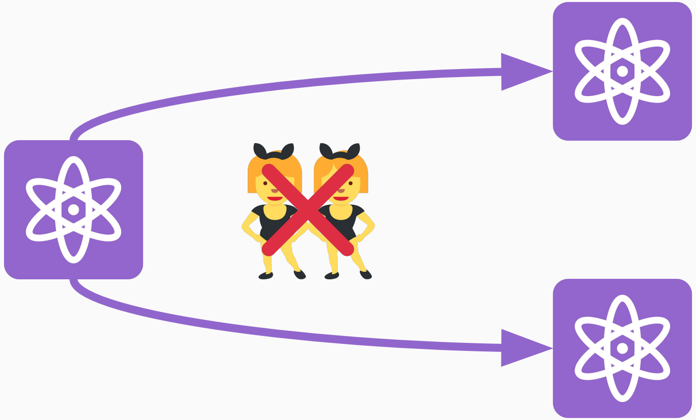


quantum principles

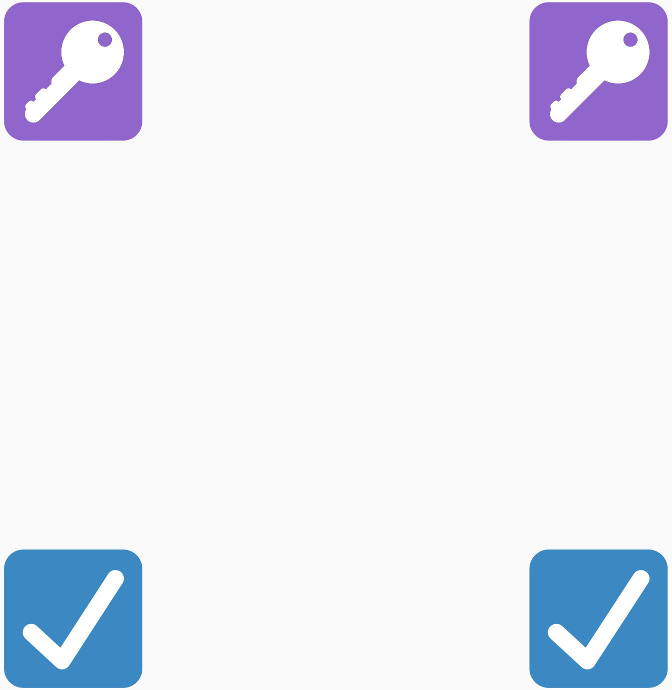
destructive measurements



no cloning



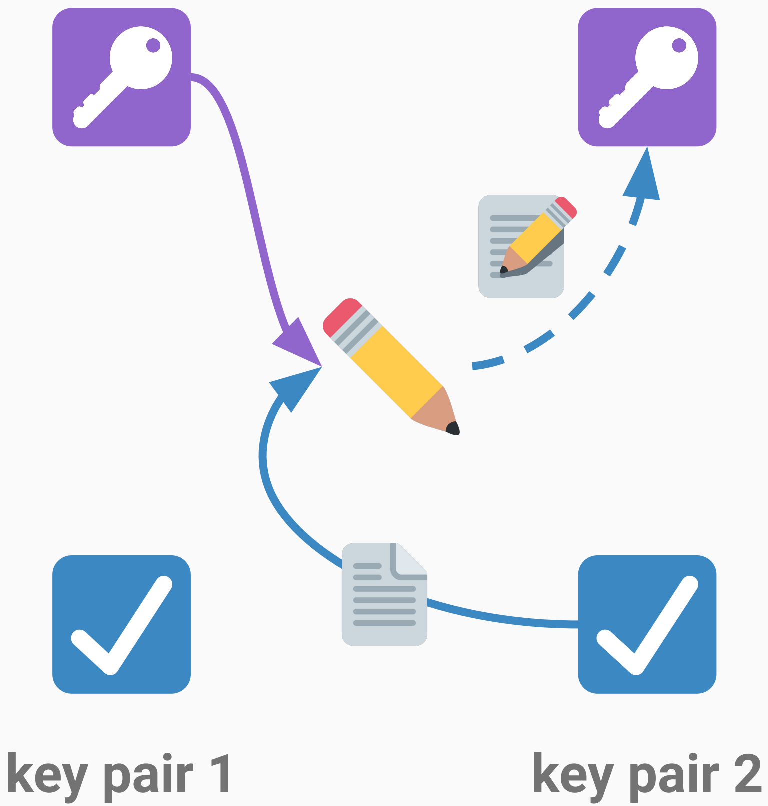
key chaining



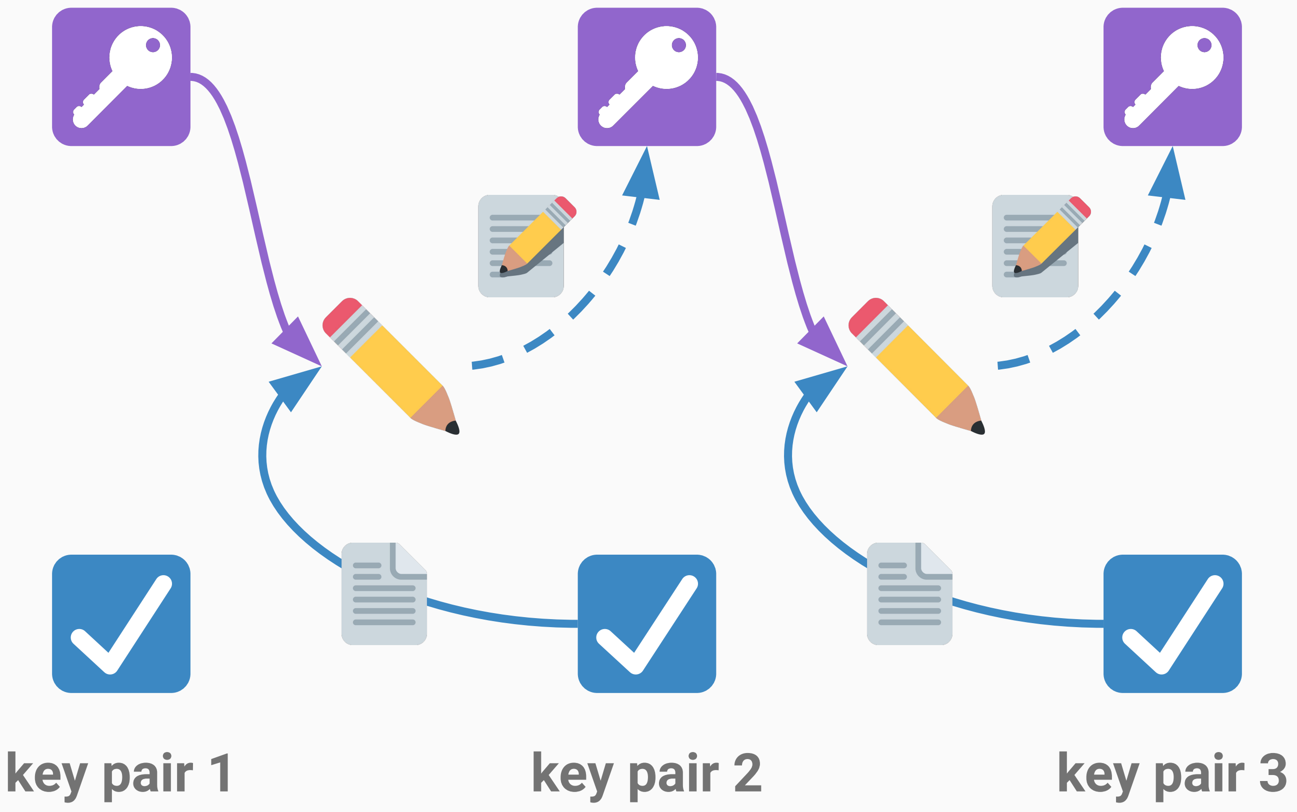
key pair 1

key pair 2

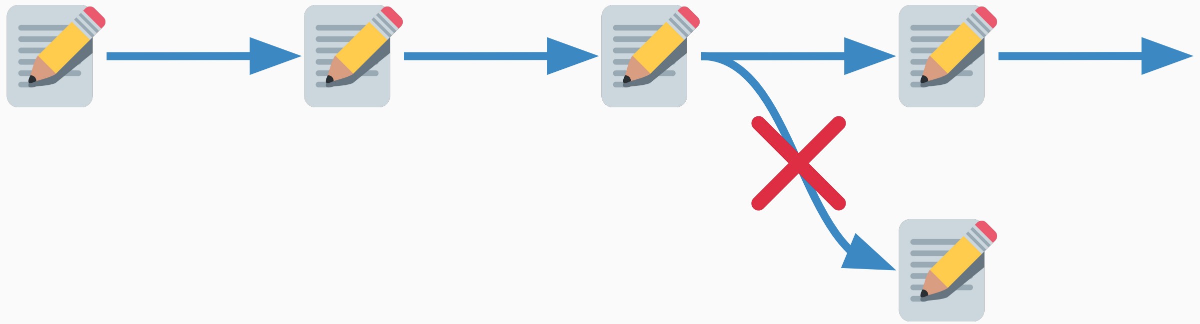
key chaining



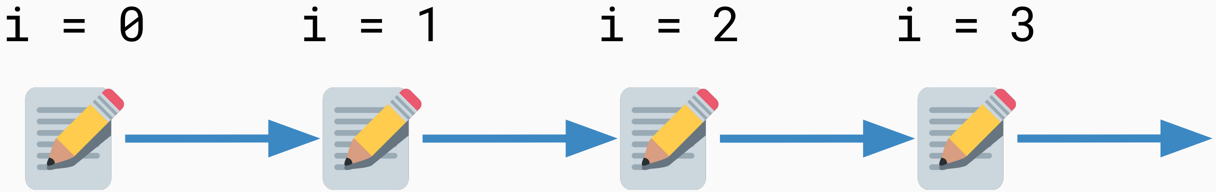
key chaining



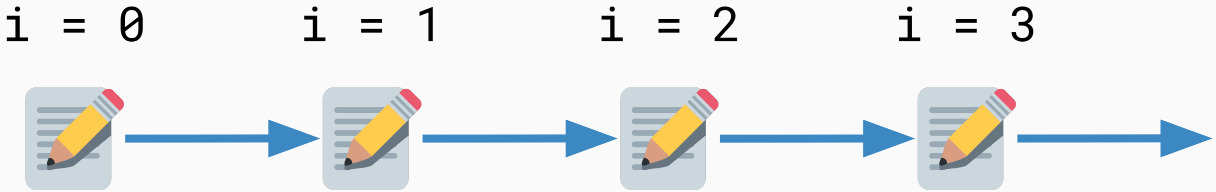
unforkable signature chains

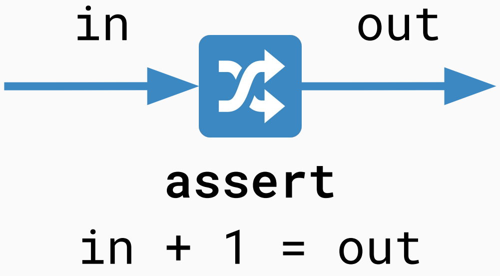


stateful signature chains

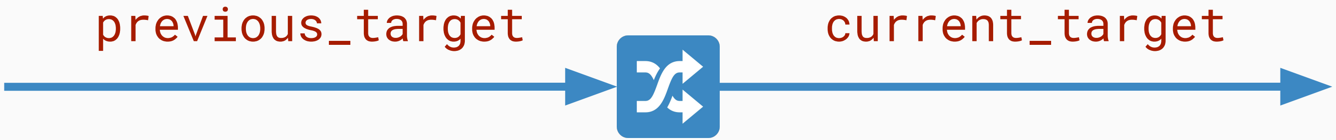


stateful signature chains





no double vote



assert

previous_target < current_target

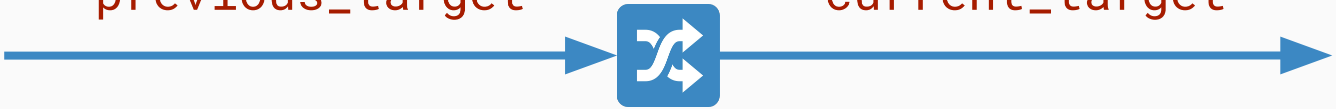
removing slashing conditions

no double vote

no surround vote

previous_source
previous_target

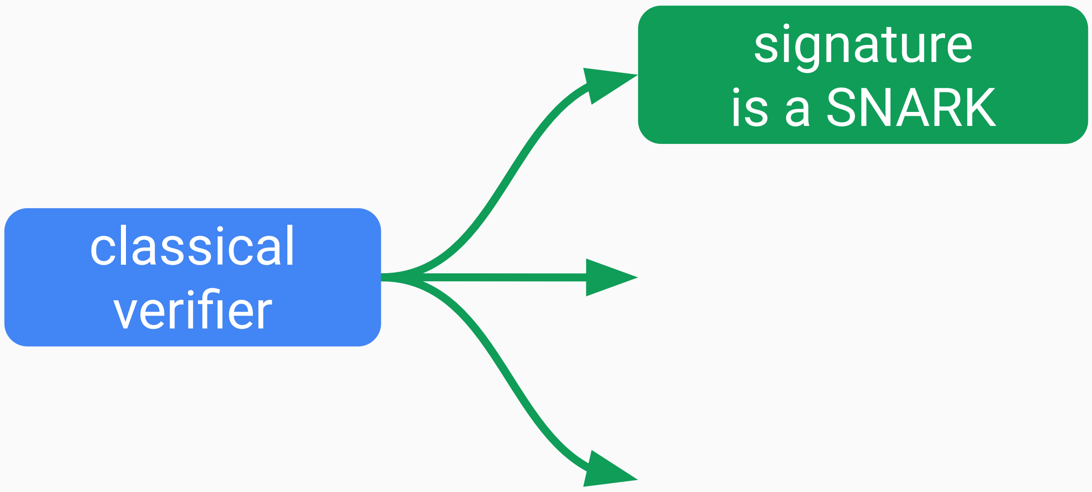
current_source
current_target



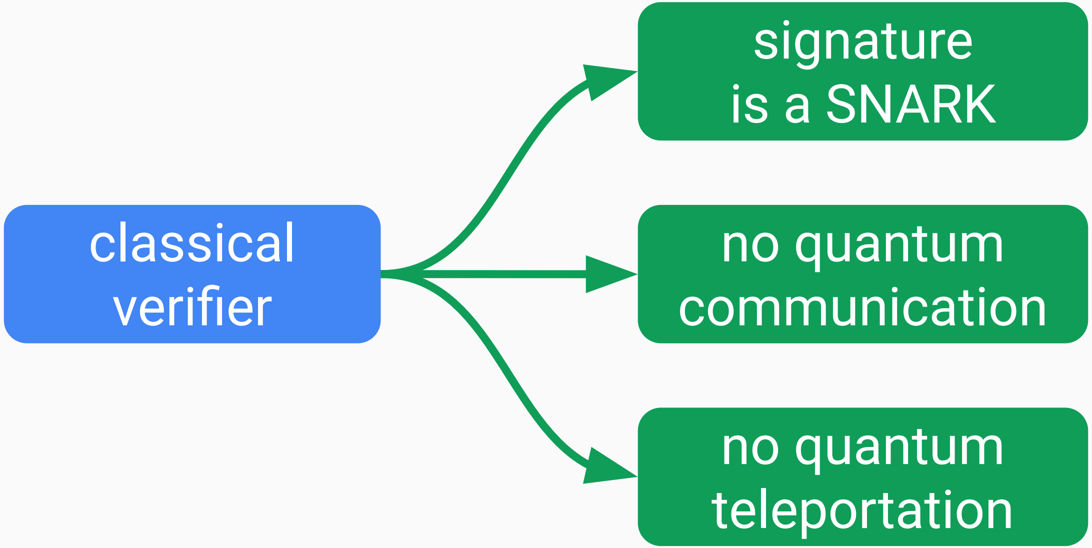
assert

previous_target < current_target
previous_source ≤ current_source

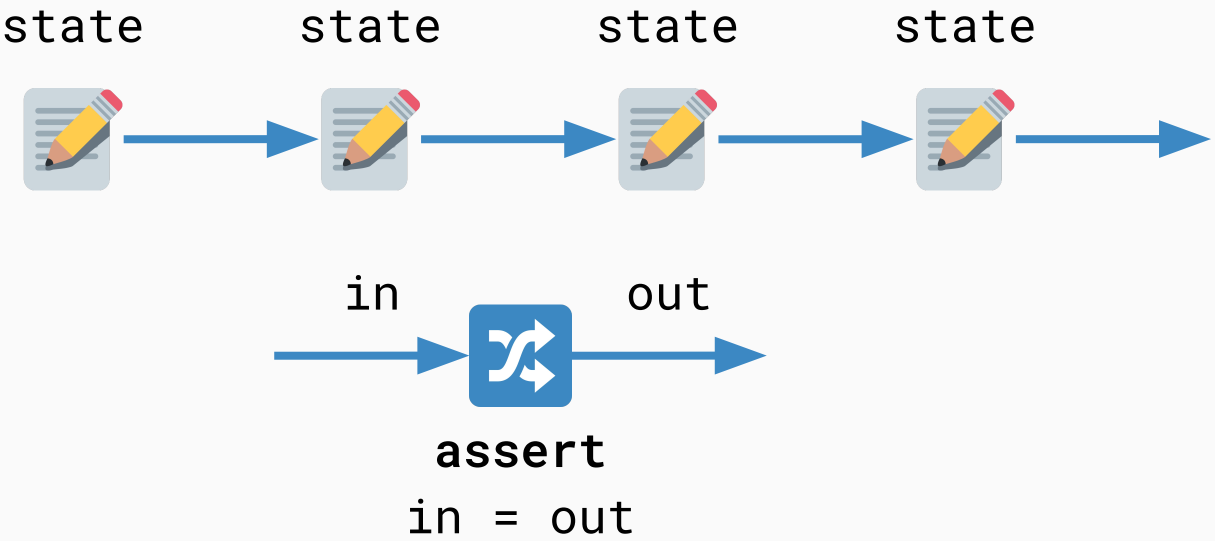
potential applicability



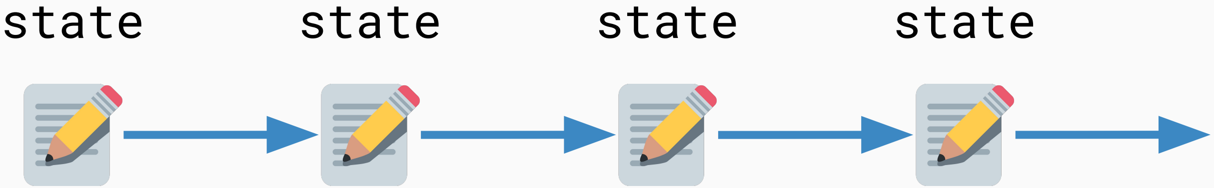
potential applicability

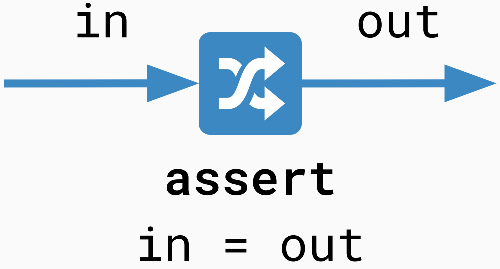


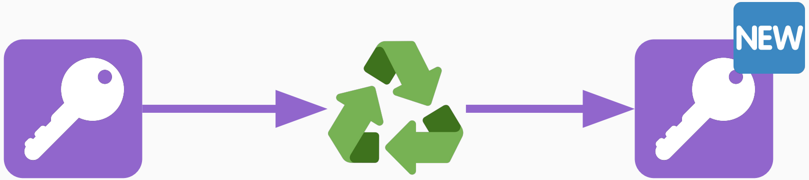
practicality—short-term memory



practicality—short-term memory







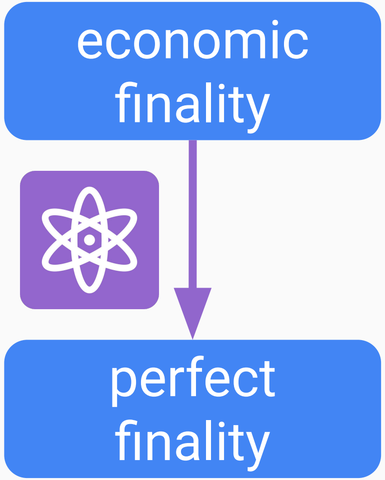
private key refresh

part 1—signature chains

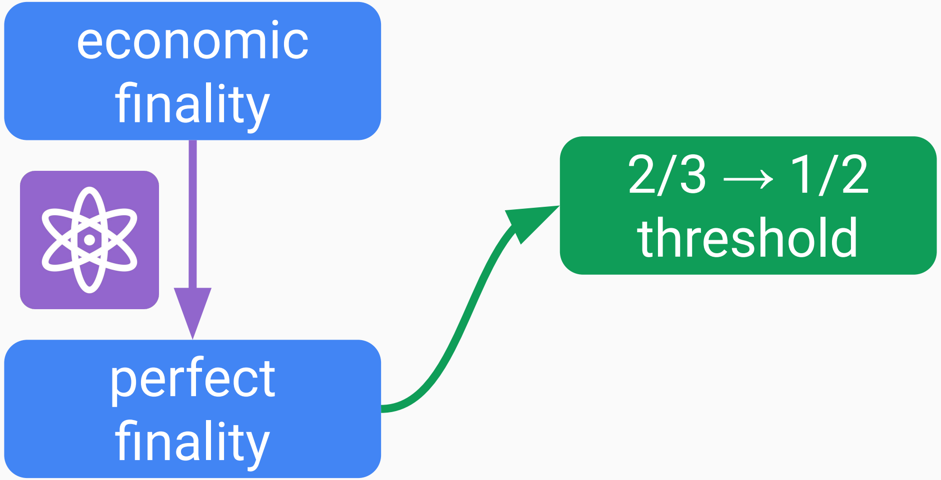
part 2—applications

part 3—accelerationism

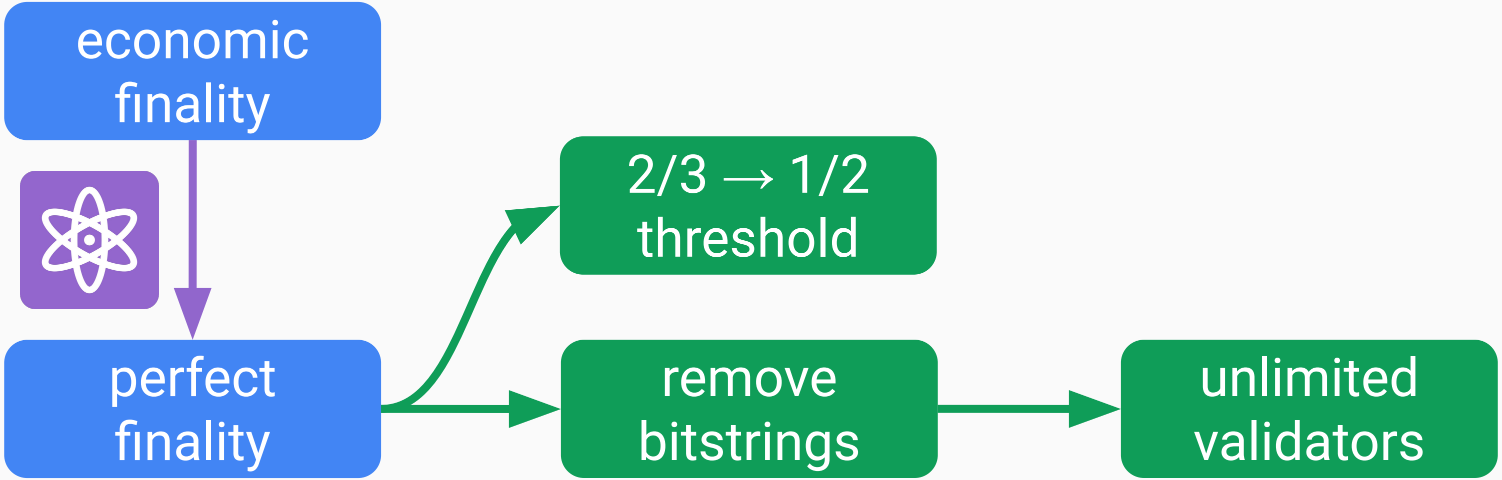
perfect finality



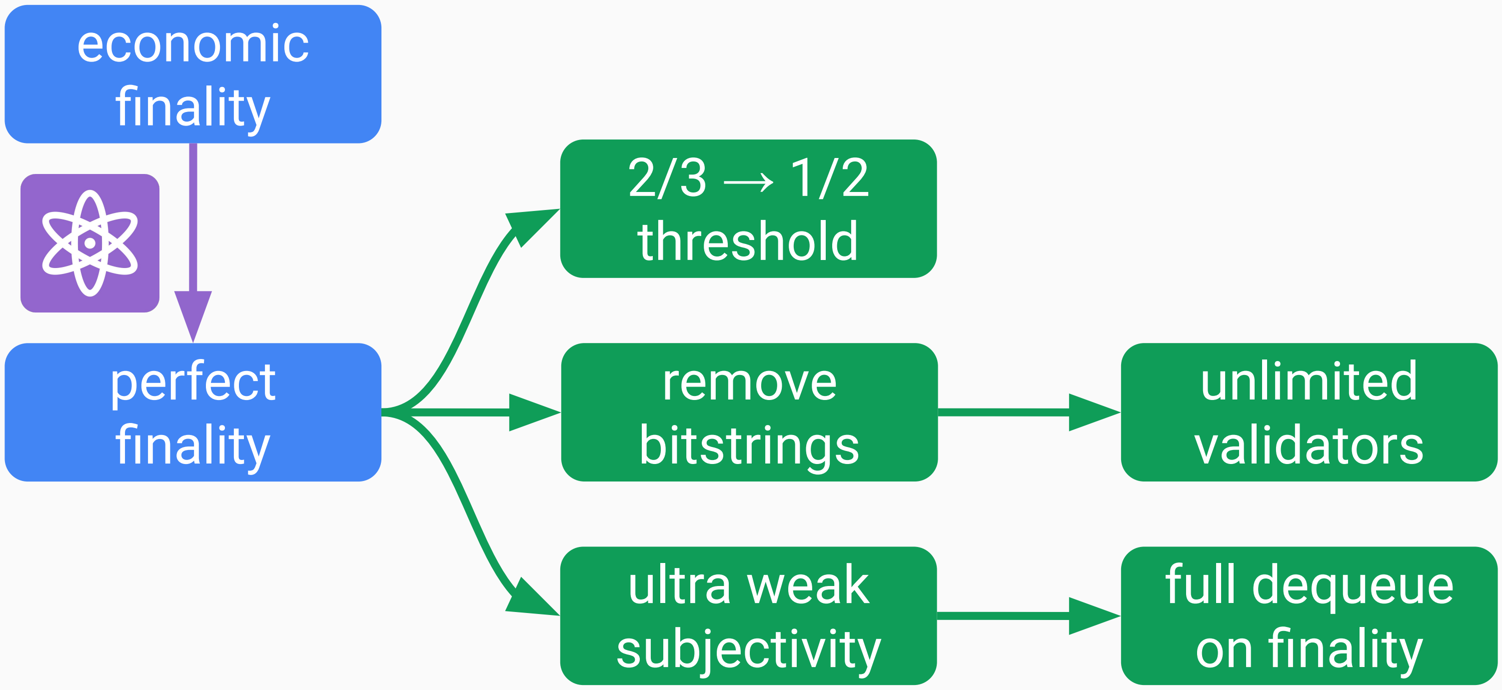
perfect finality



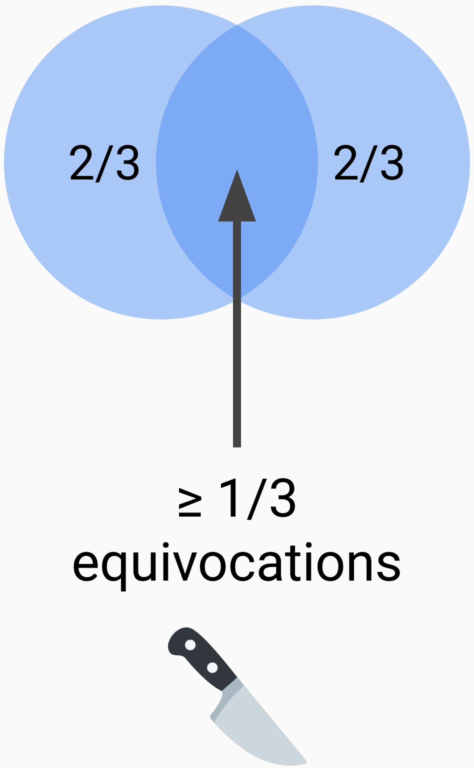
perfect finality



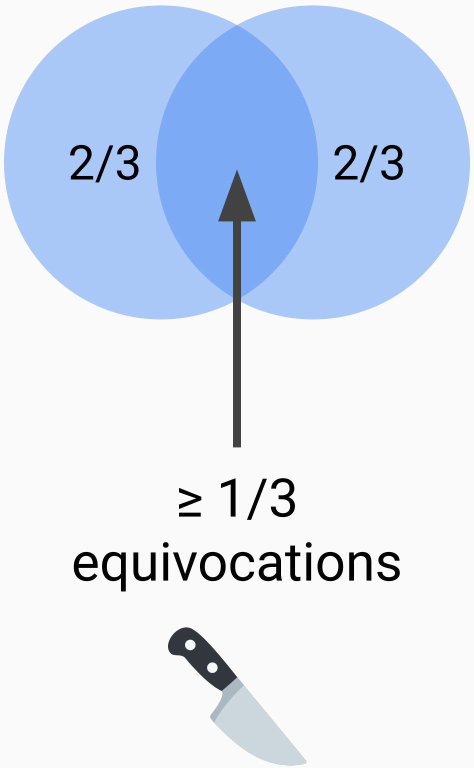
perfect finality

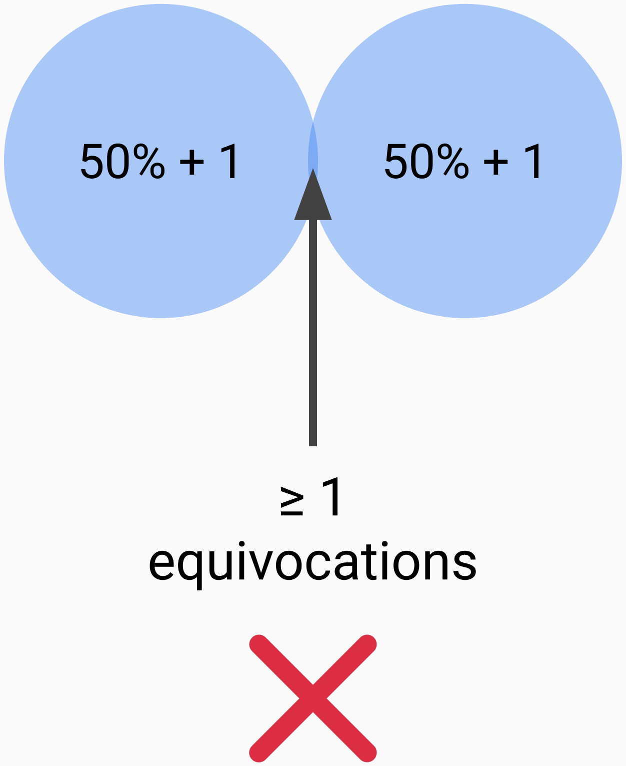


economic vs perfect finality

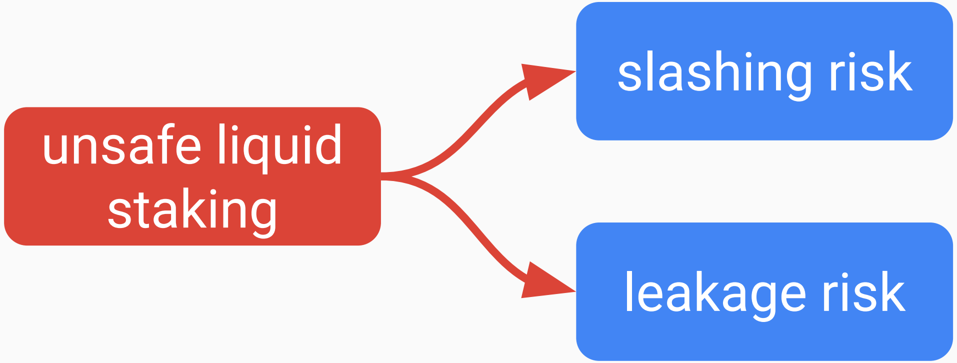


economic vs perfect finality



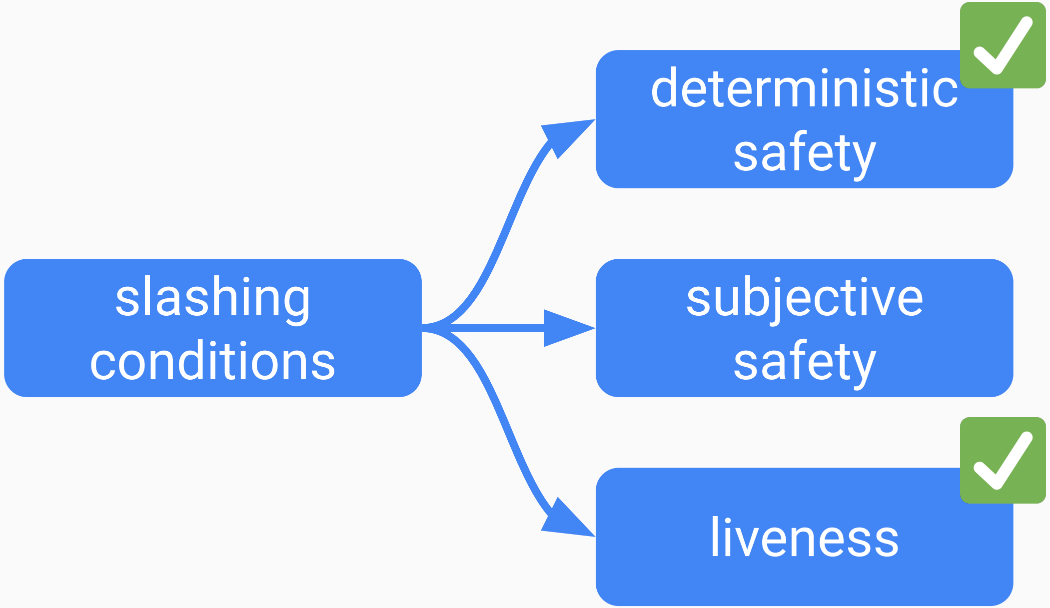


liquid staking

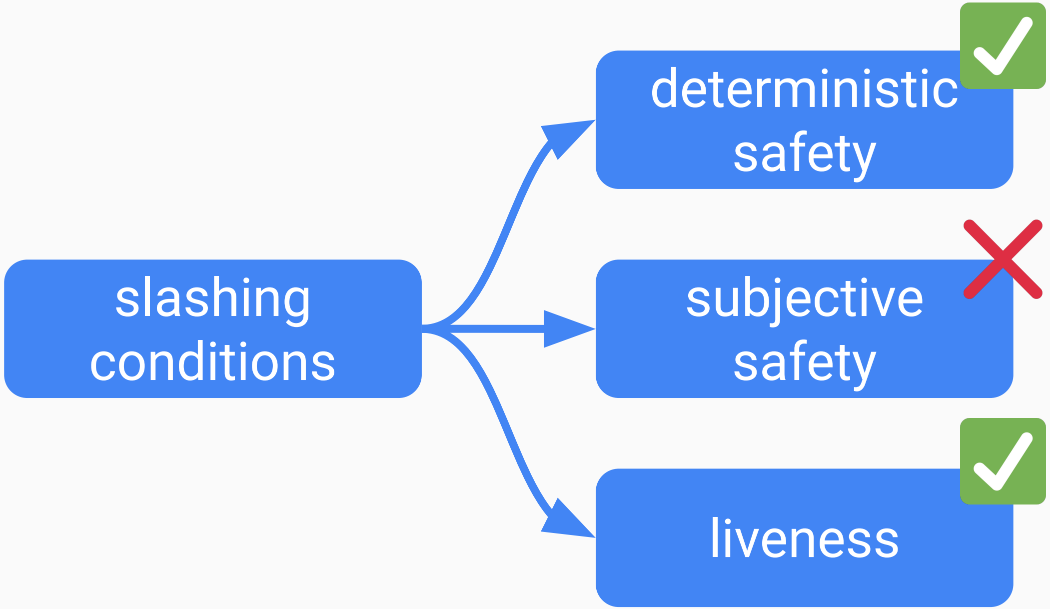


liquid staking

(re)staking


slashing conditions
deterministic safety
subjective safety
liveness

(re)staking


slashing conditions
deterministic safety
subjective safety
liveness

other applications

quantum
money

other applications

quantum money

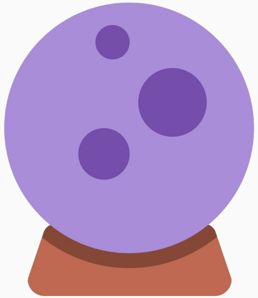
proof of location

other applications

quantum money

proof of location

safer oracles

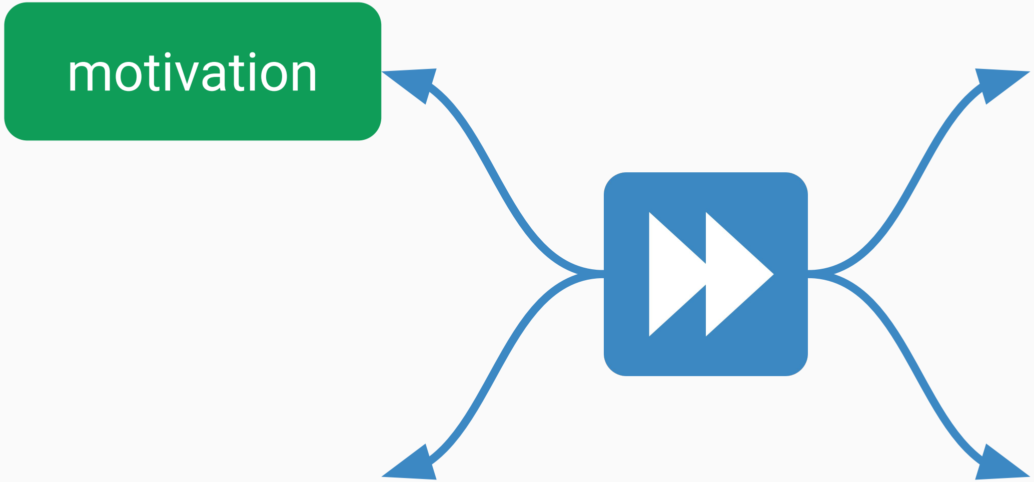


part 1—signature chains

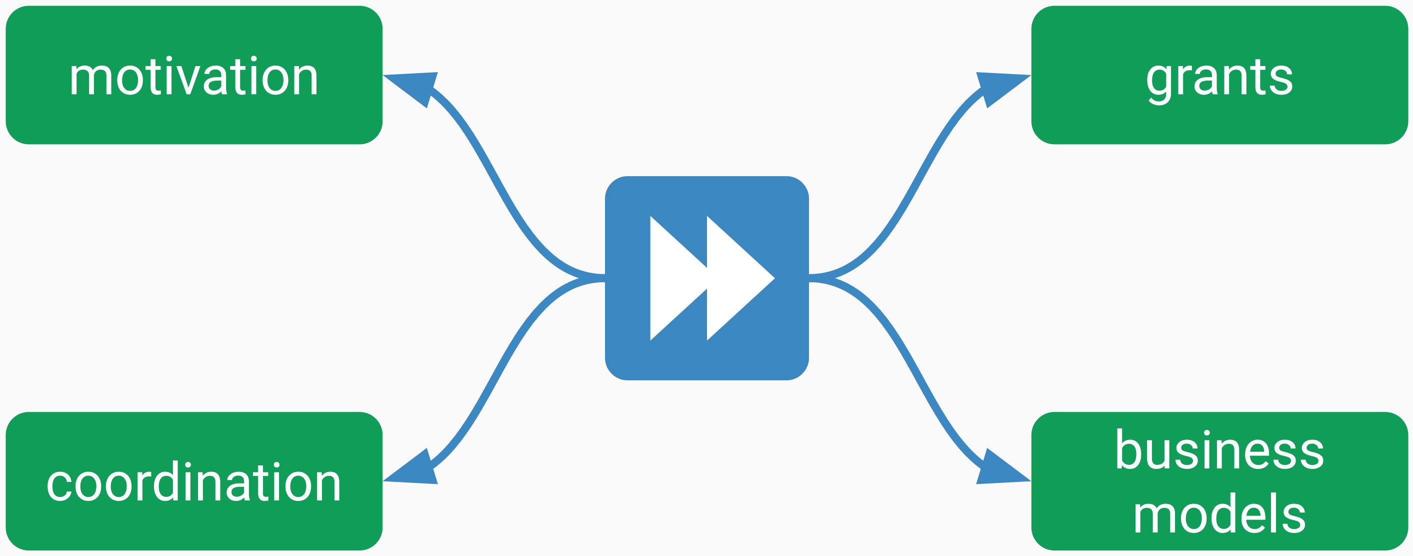
part 2—applications

part 3—accelerationism

cryptographic accelerationism



cryptographic accelerationism



QSig

THE UNIVERSITY of EDINBURGH informatics

ethereum foundation

Cardano Foundation

INPUT | OUTPUT

QSig

2025?

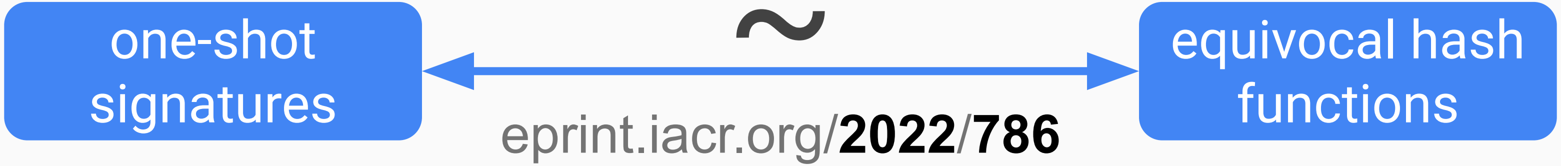
THE UNIVERSITY of EDINBURGH informatics

ethereum foundation

Cardano Foundation

INPUT | OUTPUT

hash function competition



hash function competition

one-shot signatures ~ equivocal hash functions

eprint.iacr.org/**2022/786**

hash design competition

hash function competition

one-shot signatures ∼ equivocal hash functions

eprint.iacr.org/**2022/786**

hash design competition

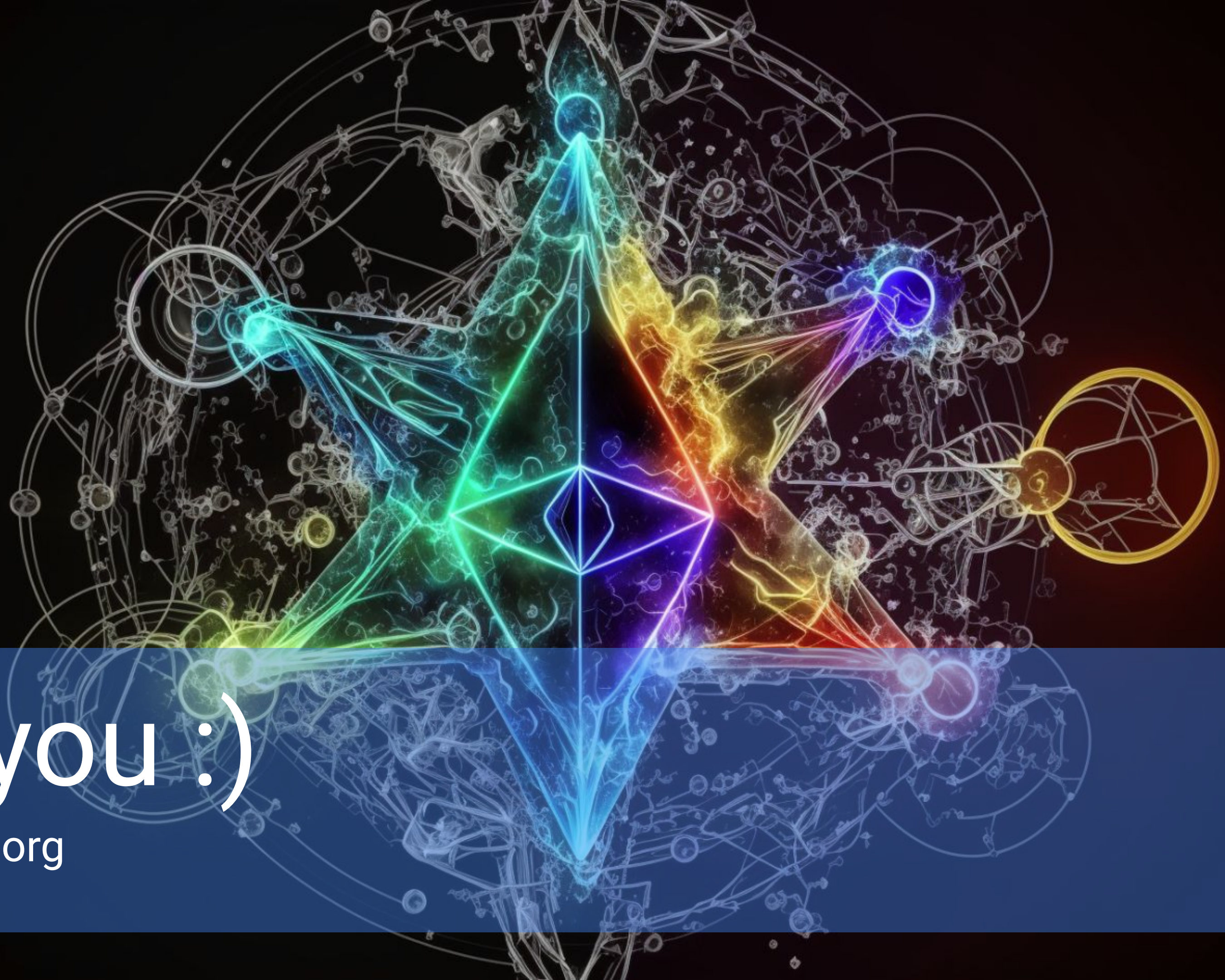
NIST

SHA3, BLAKE

ethereum foundation

STARKWARE

Poseidon, Rescue



thank you :)

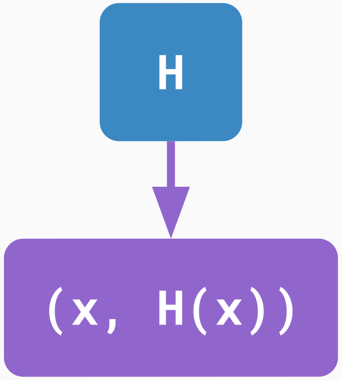
justin@ethereum.org

practicality—early offchain usage

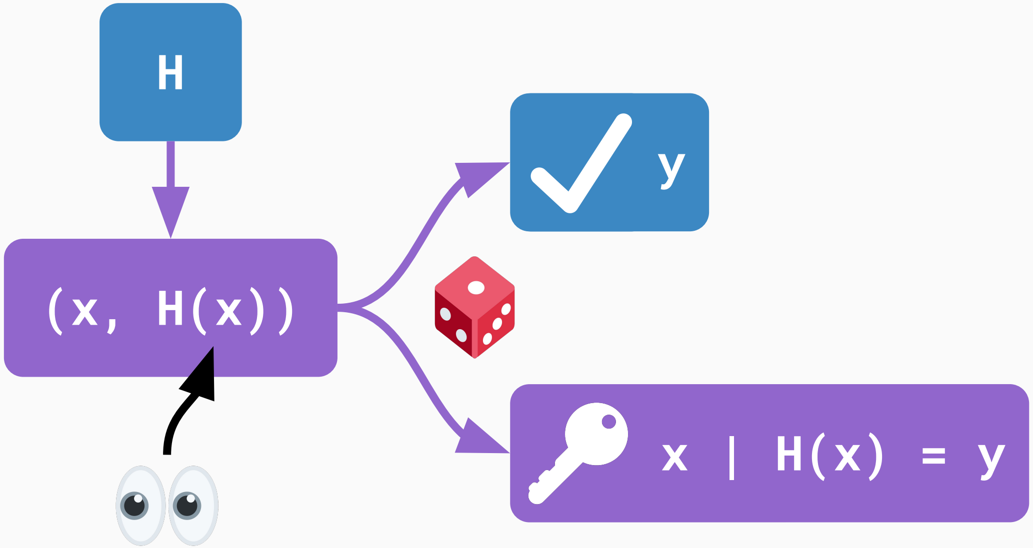
one quantum computer

signature reuse

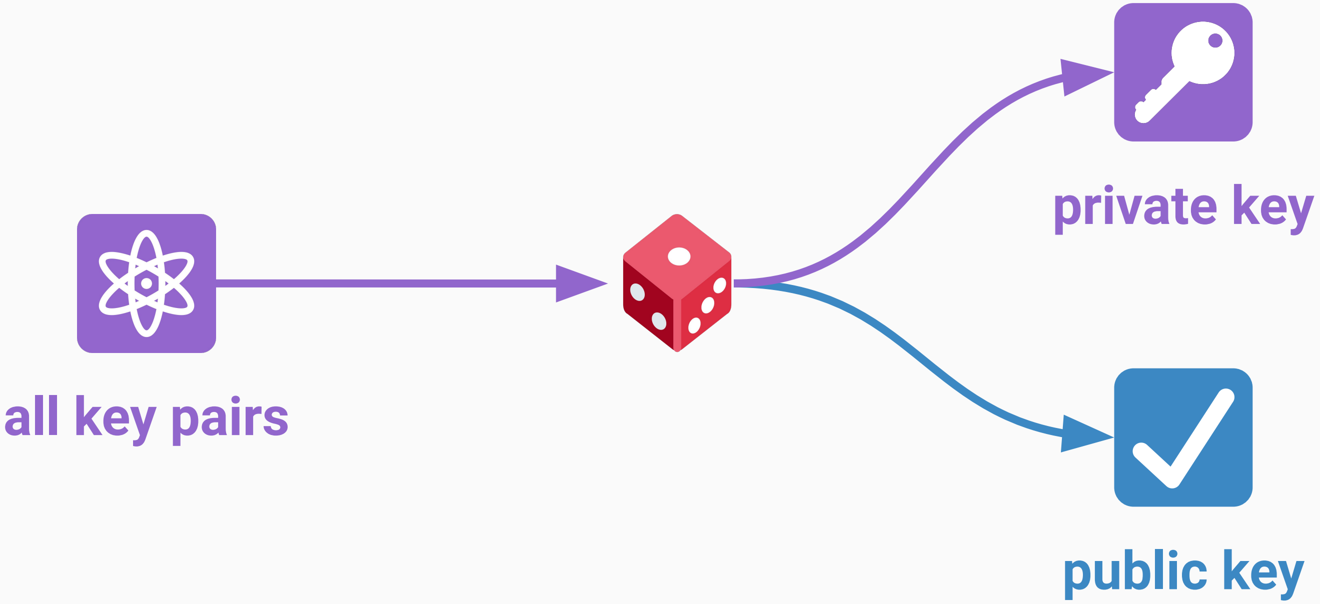
1-bit signing



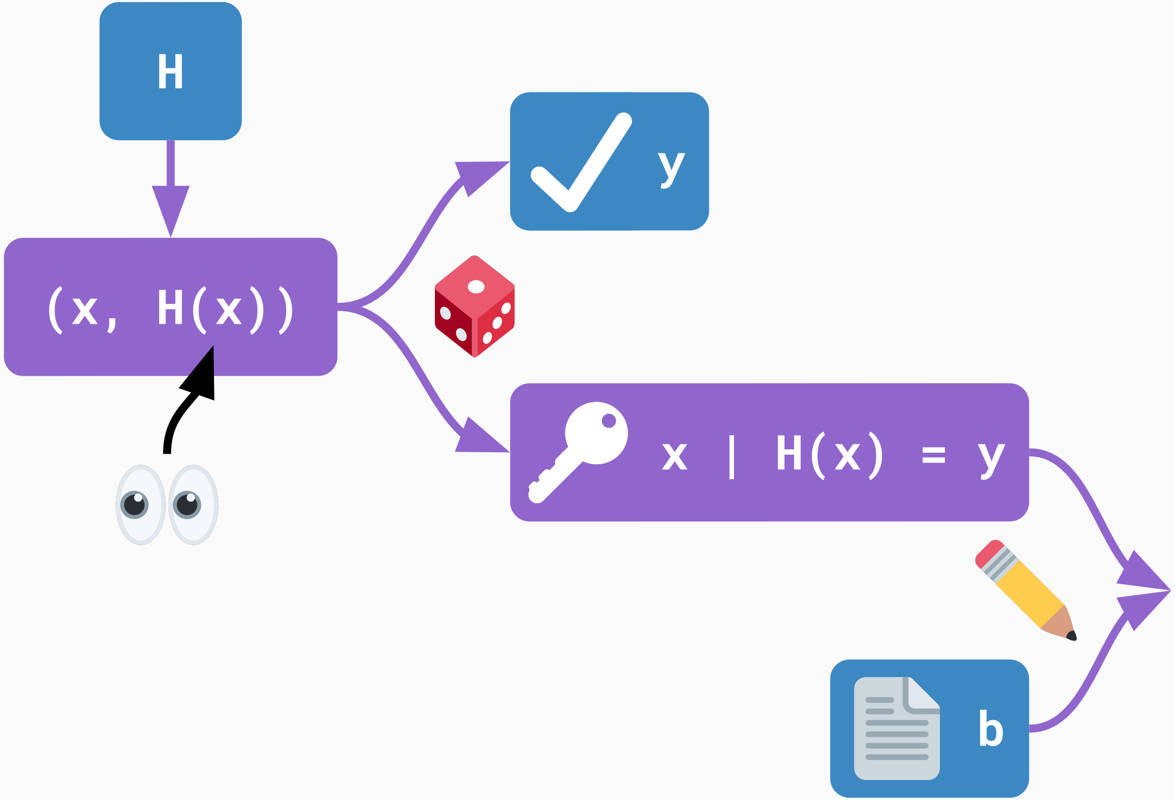
1-bit signing



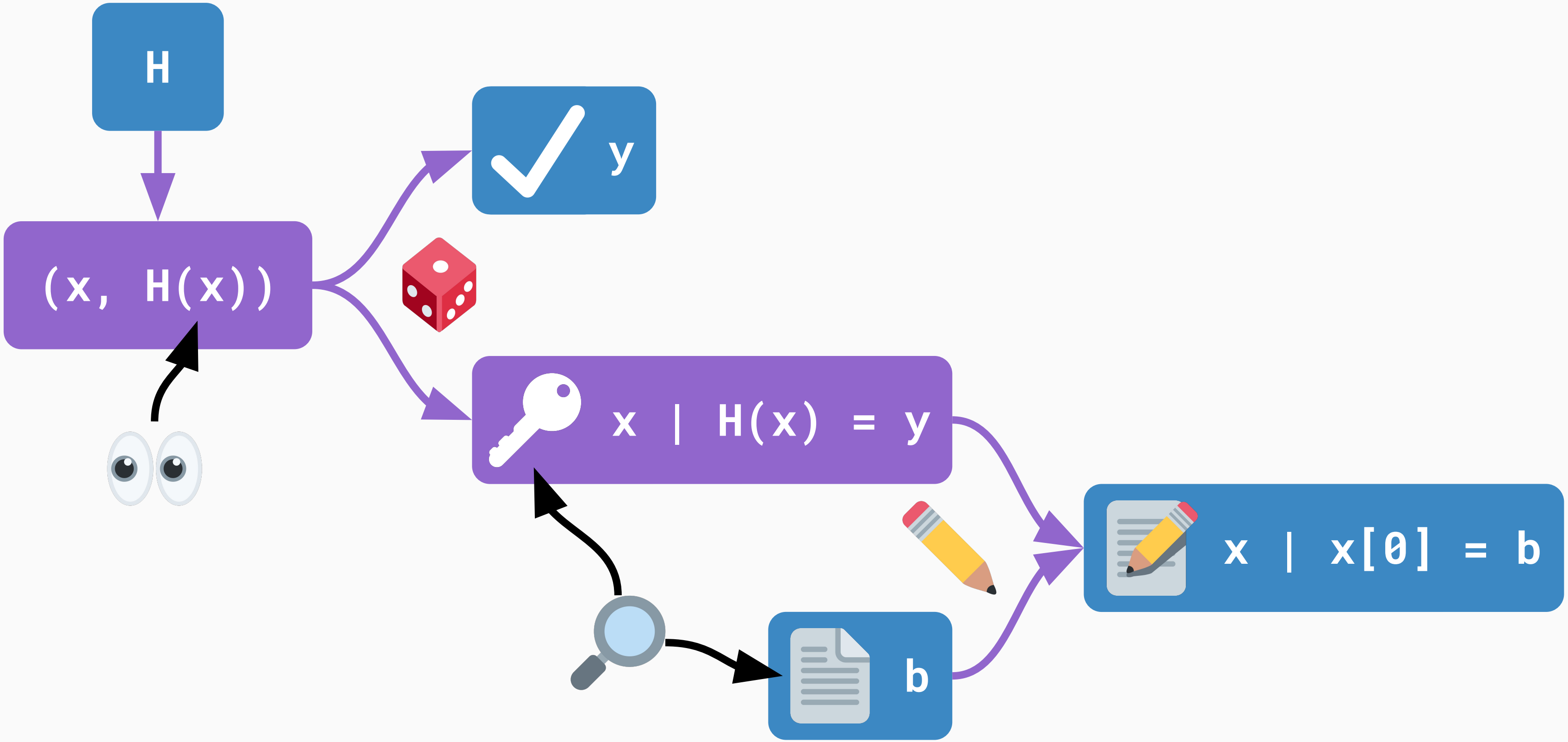
key generation



1-bit signing



1-bit signing



1-bit signing

1. start with a suitable 256-bit hash function **H**

1-bit signing

1. start with a suitable 256-bit hash function **H**
2. build a uniform superposition of all preimage-image pairs
    - `(x, H(x))` for every 512-bit preimage **x**

1-bit signing

1. start with a suitable 256-bit hash function **H**
2. build a uniform superposition of all preimage-image pairs
   - **`(x, H(x))`** for every 512-bit preimage **x**
3. observe and collapse the second register to get
   - *pubkey*—a random image **y**
   - *privkey*—a superposition of preimages **x** such that **`H(x) = y`**

1-bit signing

1. start with a suitable 256-bit hash function **H**
2. build a uniform superposition of all preimage-image pairs
   - **`(x, H(x))`** for every 512-bit preimage **x**
3. observe and collapse the second register to get
   - *pubkey*—a random image **y**
   - *privkey*—a superposition of preimages **x** such that **`H(x) = y`**
4. to sign bit **b** run a fancy quantum search algorithm to find
   - *signature*—a preimage **x** such that the first bit of **x** is **b**

1-bit signing

1. start with a suitable 256-bit hash function **H**
2. build a uniform superposition of all preimage-image pairs
   - **`(x, H(x))`** for every 512-bit preimage **x**
3. observe and collapse the second register to get
   - *pubkey*—a random image **y**
   - *privkey*—a superposition of preimages **x** such that **`H(x) = y`**
4. to sign bit **b** run a fancy quantum search algorithm to find
   - *signature*—a preimage **x** such that the first bit of **x** is **b**

hard!