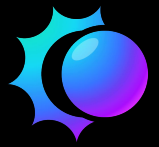
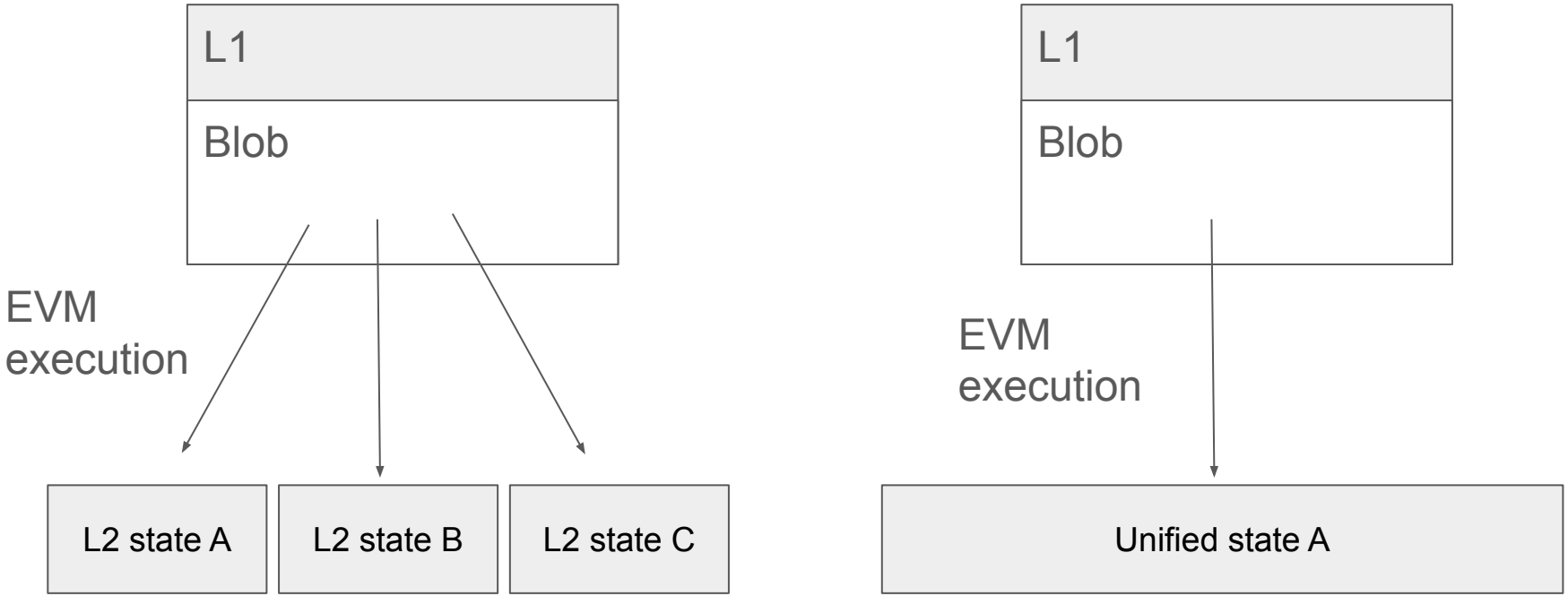
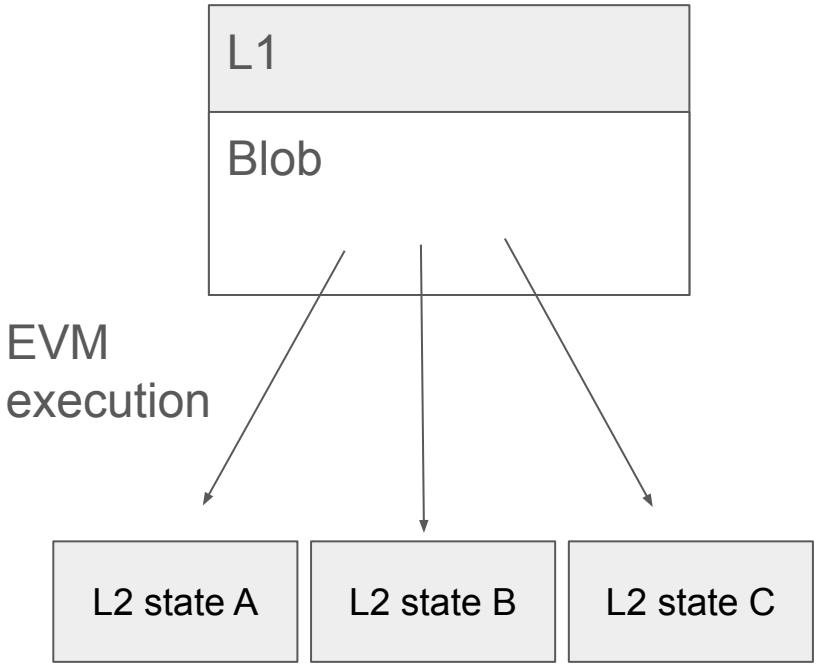
About Statelessness

Leona Hioki from Intmax 

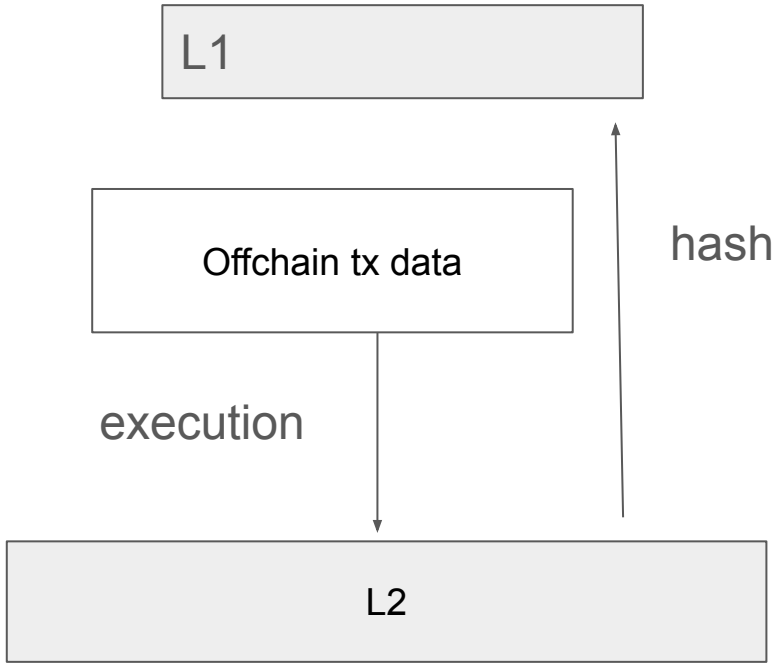
What's the difference?



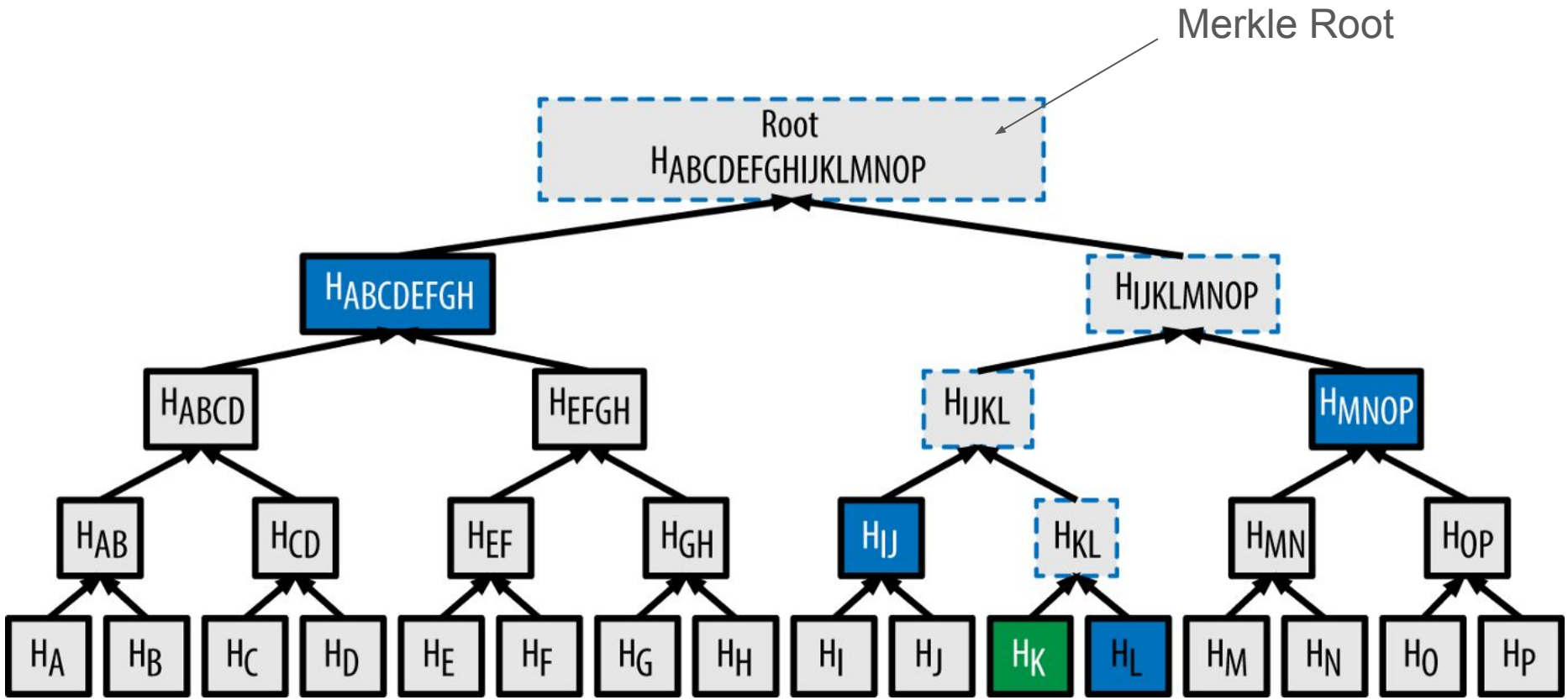
Stateful Rollups



Stateless L2

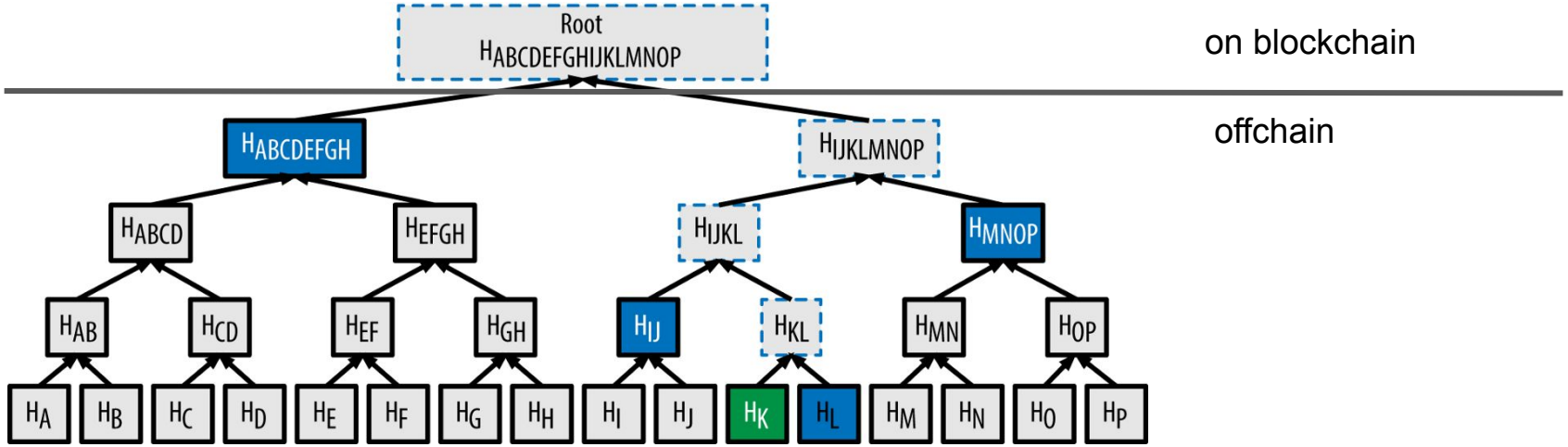


Merkle Tree:

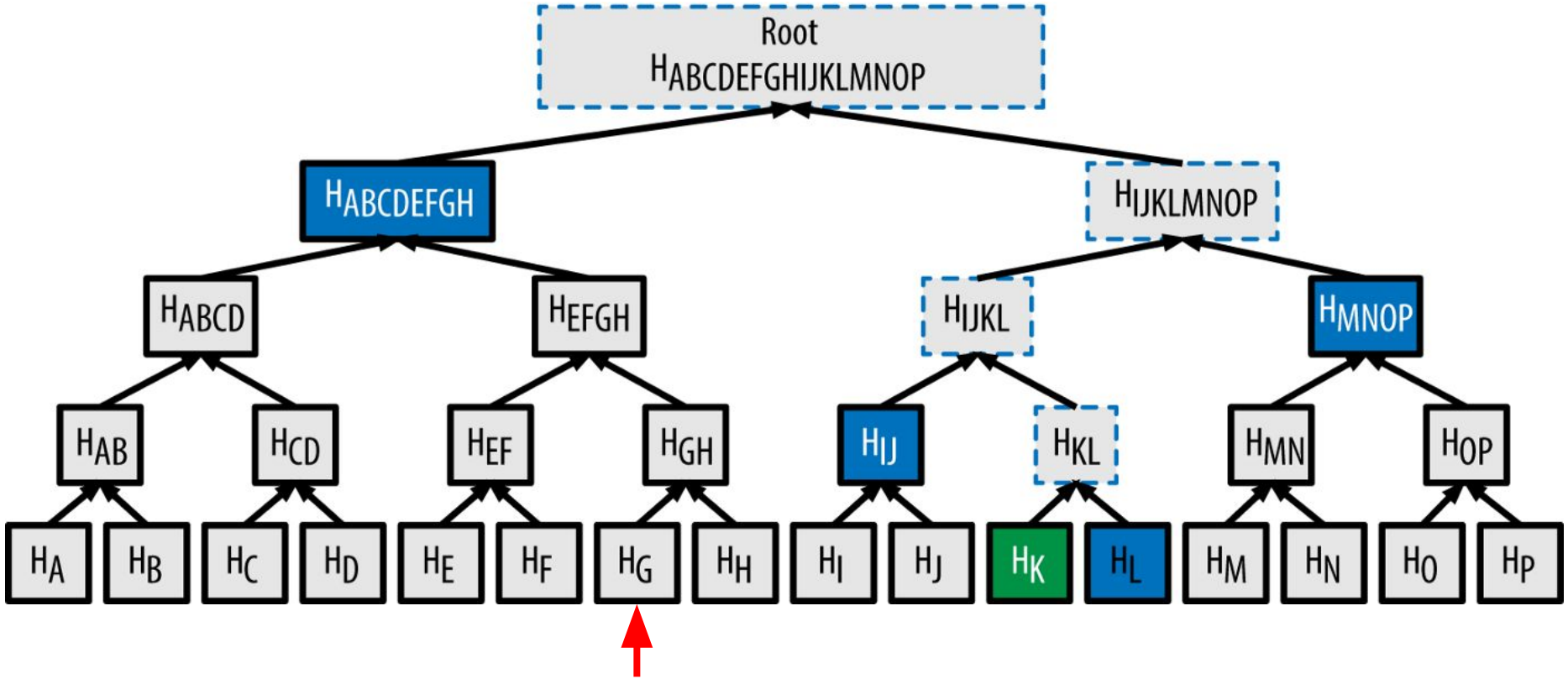


The set of blue values is called a Merkle Proof.

How to make a stateless system
= Putting only Merkle roots onchain = Small block

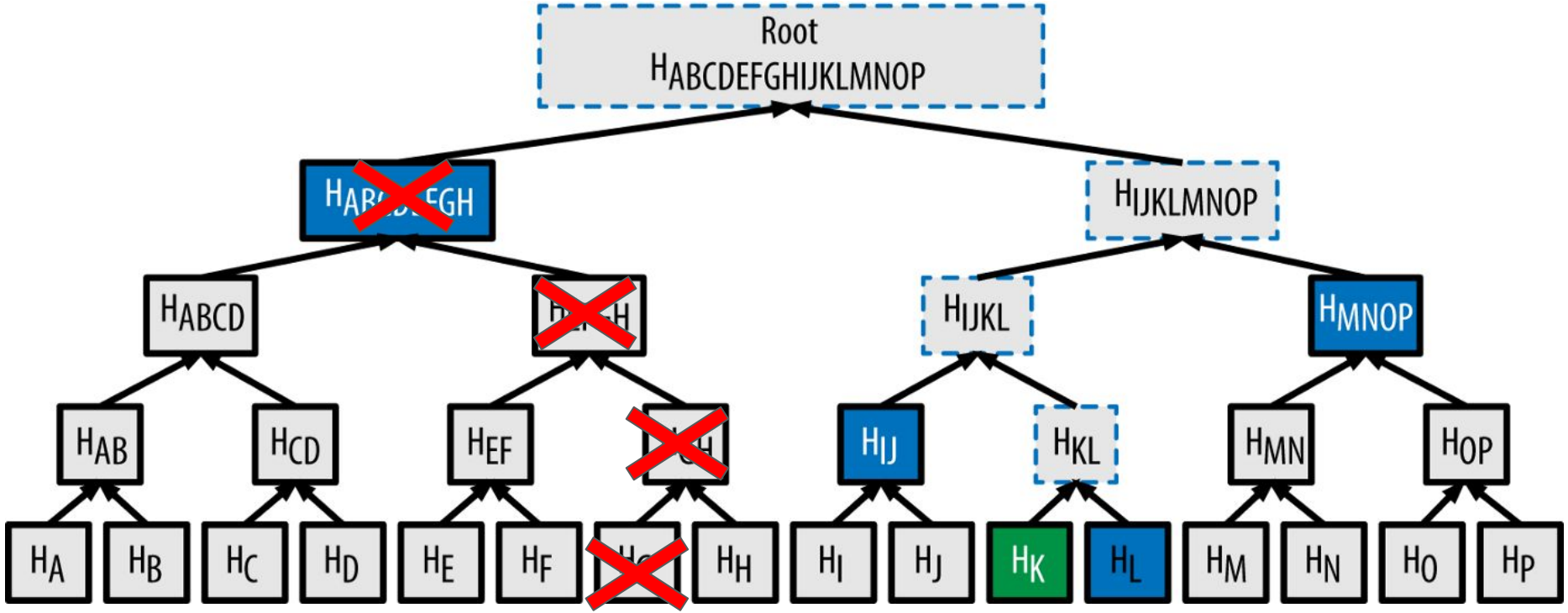


Question1: what if everybody on this Earth lost H_G ?

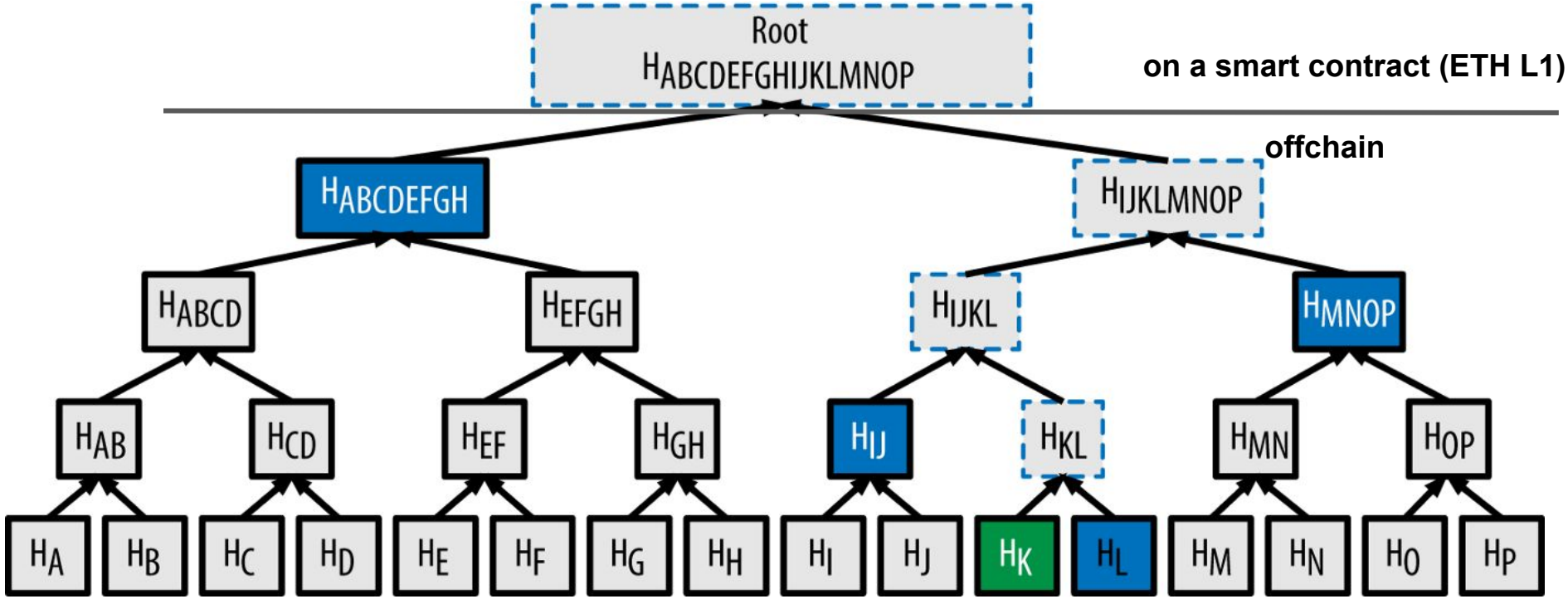


Answer:

Nobody can make any inclusion proof. (=DA attack)

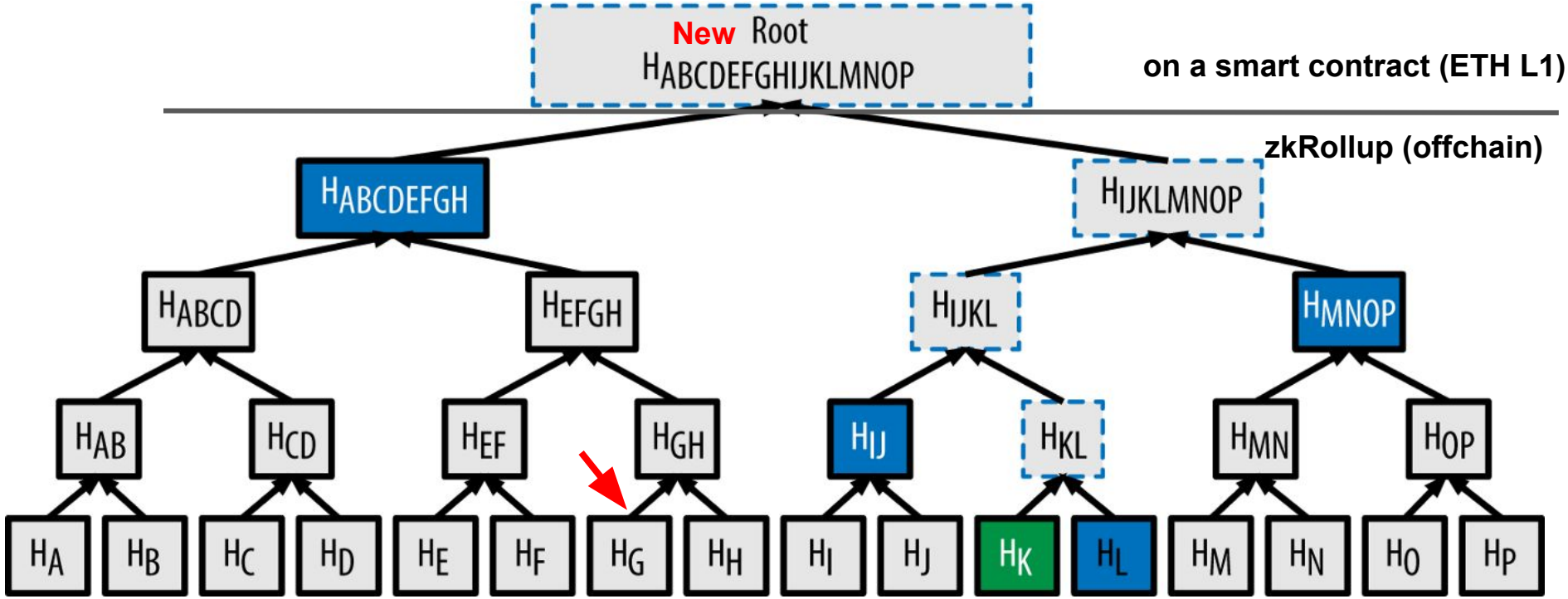


Question: How about distributing proofs H
to its asset holder directly?



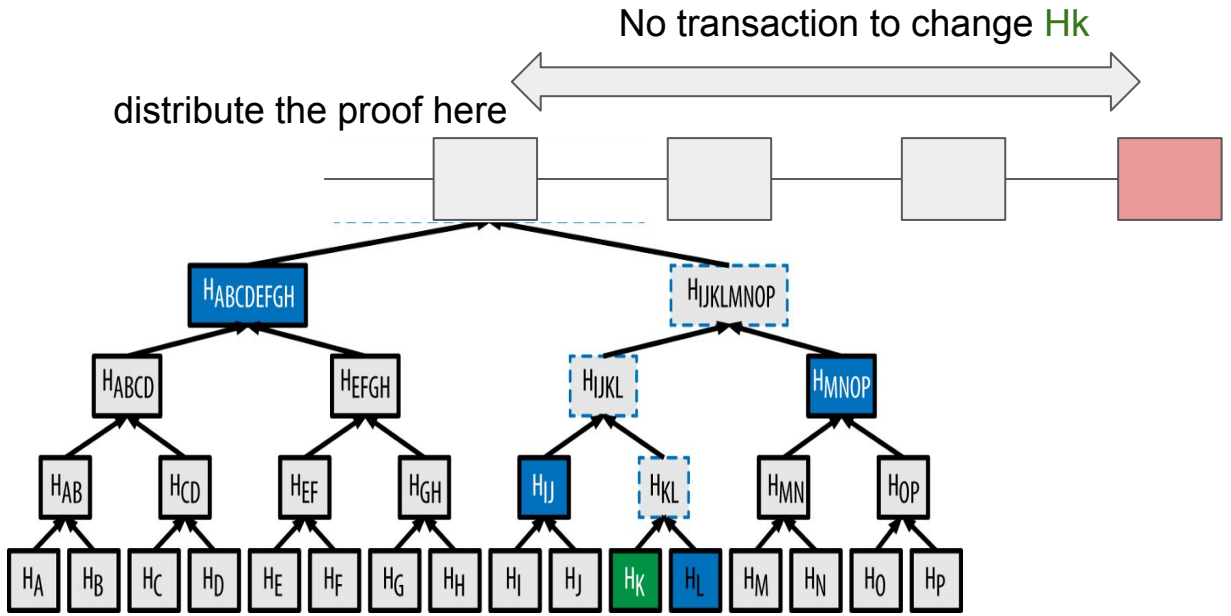
Distributing proof(blue) to the user side directly not to lose them.

Answer: The root will change if any of data(Ha~Hp) changes.
So the proof will be no longer valid.



This is what is proposed in “Limit of revocable blockchain (Miranda)” as a limit of statelessness.

But, what if you can prove that you did not change the data by proving you did not send any transaction? That makes statelessness.



	Stateless	Stateful
Scalability	Good	Bad
Privacy	Good	Bad
User Requirement	Bad	Good
Functionality	Bad	Good
Unstoppability	Good	Bad
Developer Experience	Bad	Good

“Impossibility of Stateless Blockchain”

~Limits on revocable proof systems, with applications to stateless blockchains~

・What they stated is correct.

“the system must either have a linear-sized global state **or** require a near-linear rate of local proof updates”

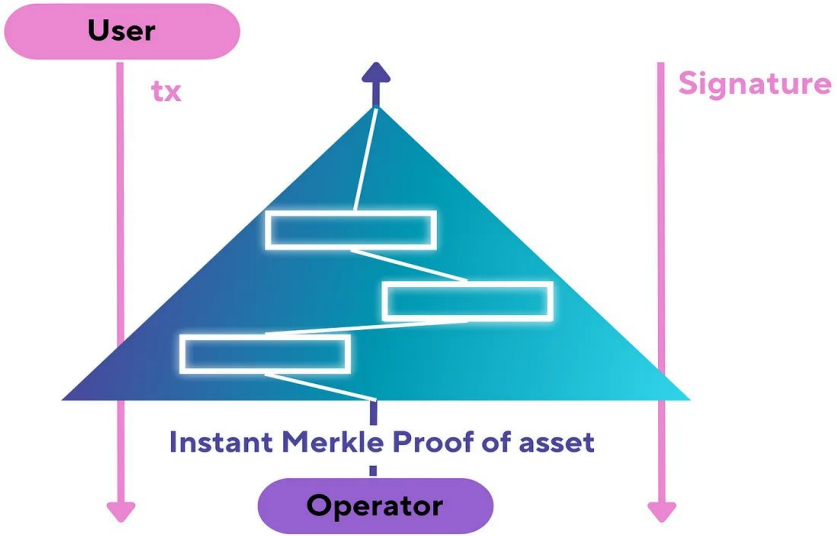
Q: How do we avoid that problem?

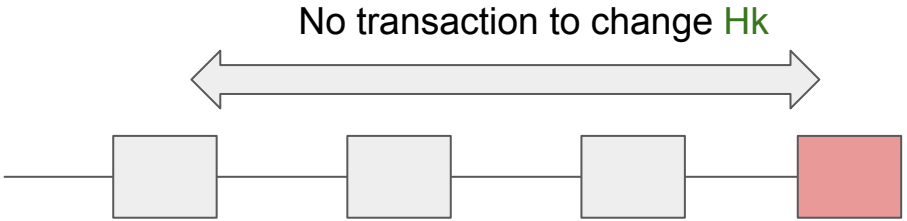
A: Making local proof updates less critical.

Proof updates without online requirement or with less online requirement are the key.

Proof updates without online requirement

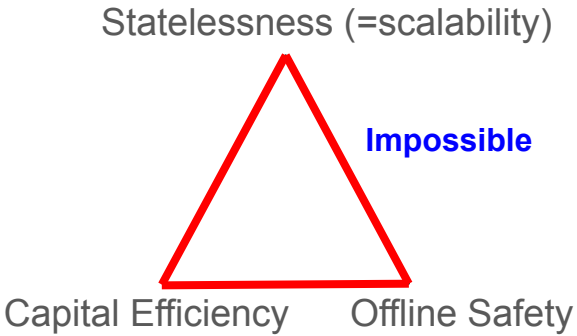
One example is signing back + proof of no transaction (by SMT)

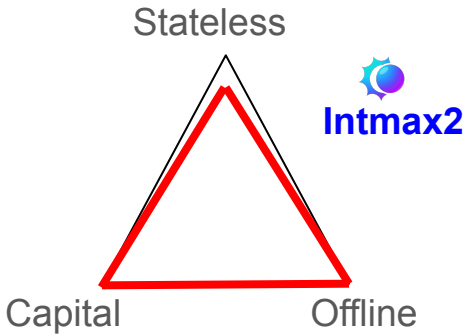


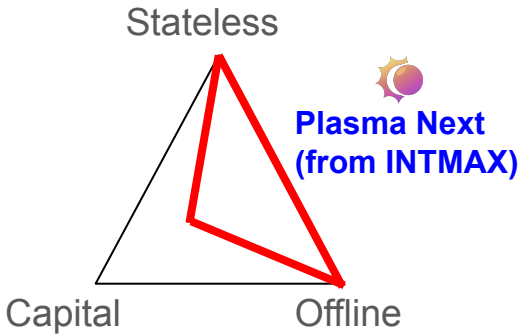


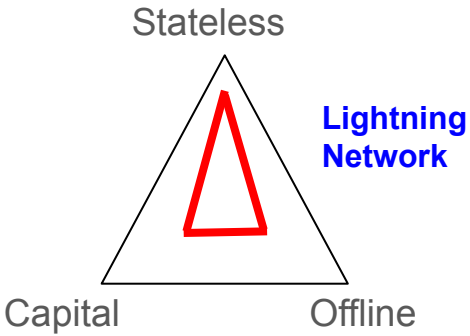
it consumes 5 bytes for each user.
But each user can put 10K transfers to one tx without additional bytes.

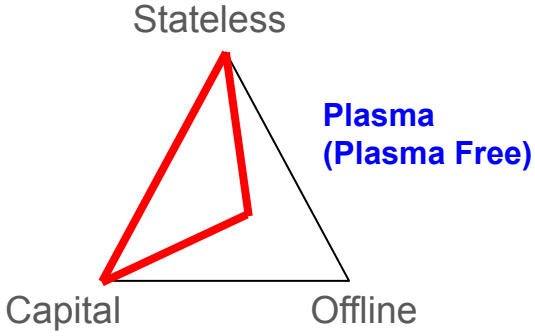
Stateless Trilemma

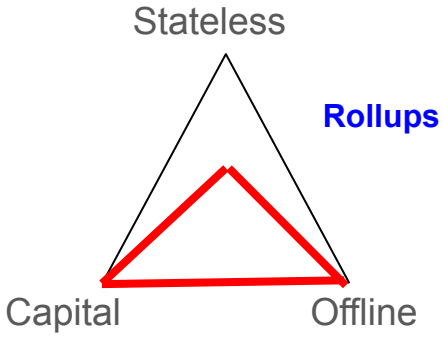




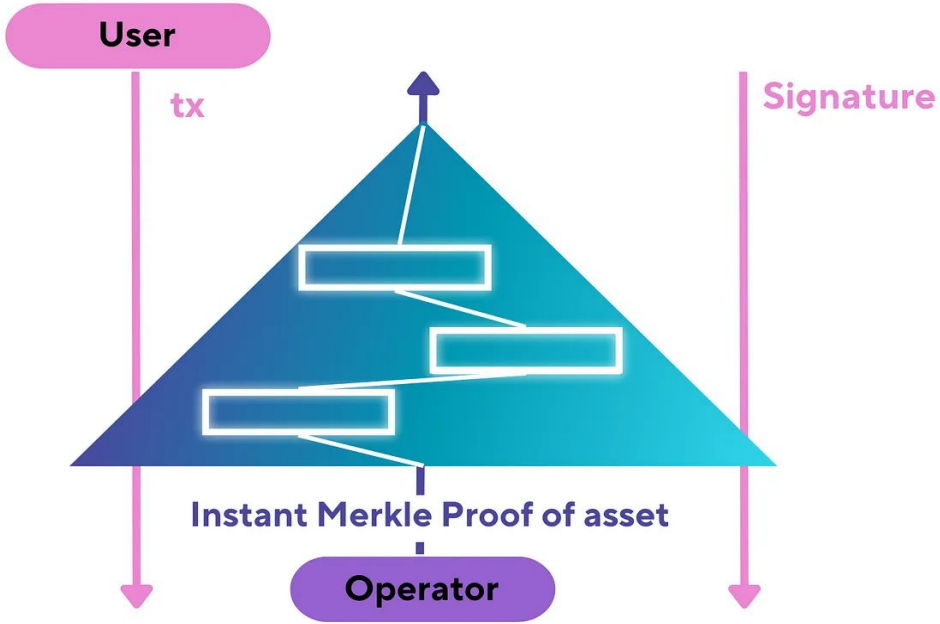






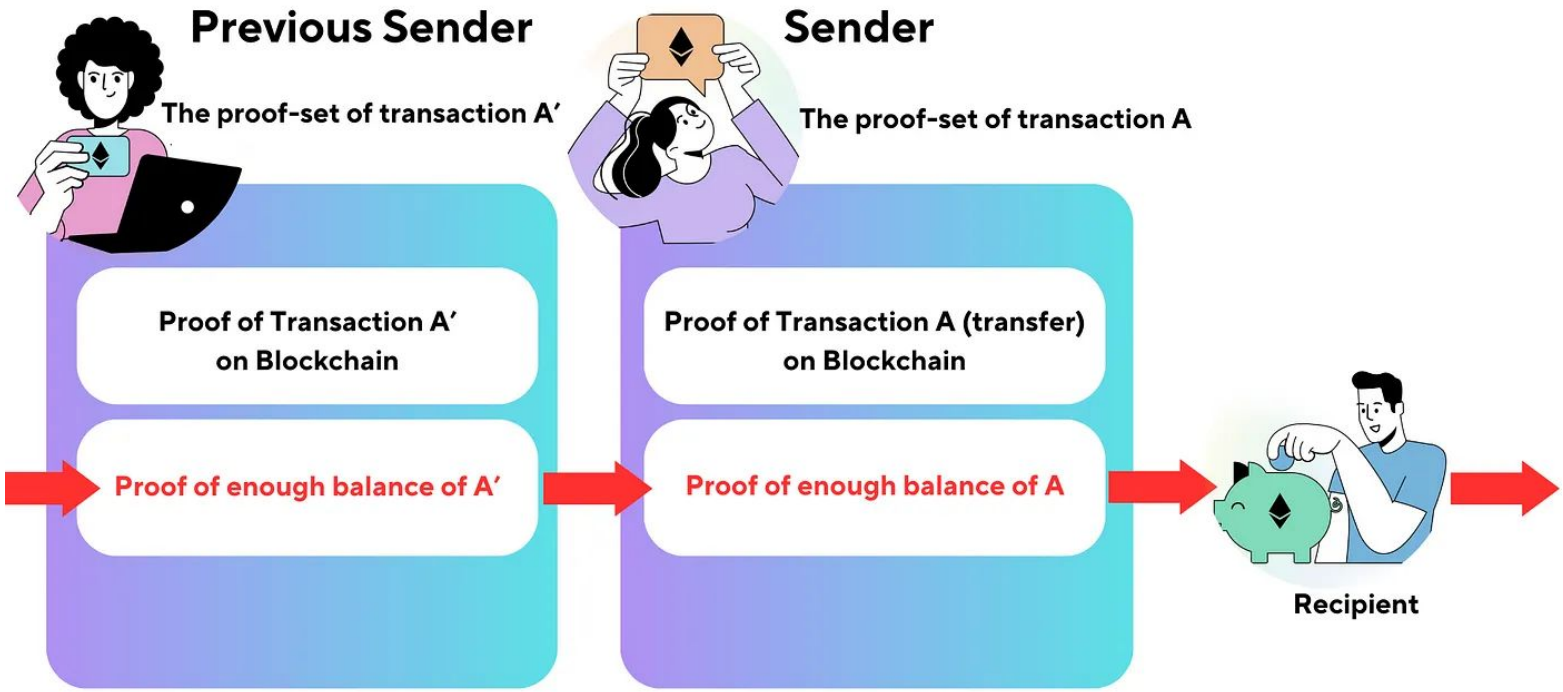


Stateless systems in detail.
Intmax2 = Hyper parallelized Mina on each client-side UTXO

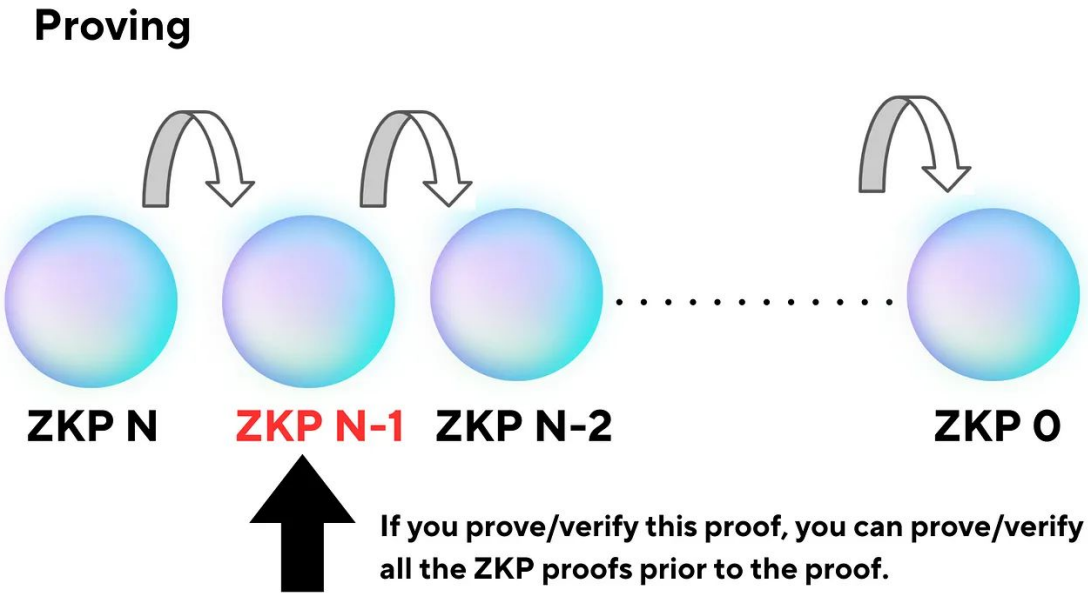


1. Safe proof (UTXO) distribution to avoid DA costs.
   Let’s say the Merkle proof itself is UTXO.

2. A sender sends ZKP & Merkle proof of a UTXO to recipients

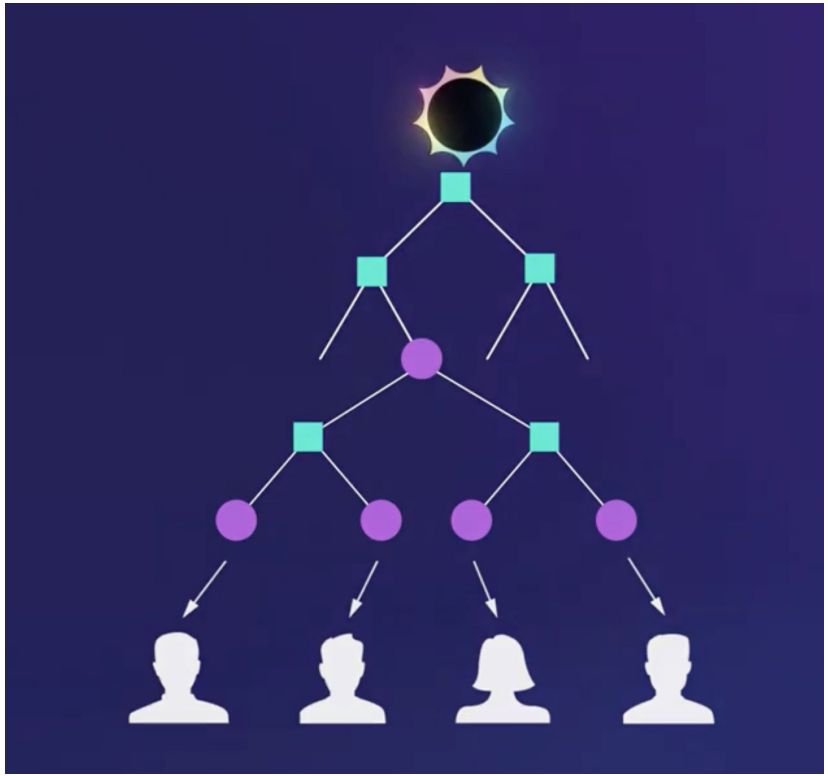


3. Mina style ZKP verification for each client side UTXO.



Not only the data preservation but also zkp computations are on client side.

4. Each tx consumes 5 bytes onchain-cost, but it can include an unlimited number of transfers.



If we set the sender of the aggregated transaction as a proxy.

Many senders can share the 5 bytes cost. It makes the complete statelessness.