Ouroboros:

A Provably Secure Proof-of-Stake Blockchain Protocol

Aggelos Kiayias
University of Edinburgh

Alexander Russell
University of Connecticut

Bernardo David
Tokyo Inst. of Technology and IOHK

Roman Oliynykov
IOHK

Presenter: Sabina Fischlin

What is Proof-of-Stake (PoS)?

What is Proof-of-Work (PoW)?

What is Proof-of-Work (PoW)?

The problem: how to reach consensus when anyone can continuously append blocks to the chain?

What is Proof-of-Work (PoW)?

The problem: how to reach consensus when anyone can continuously append blocks to the chain?

PoW solution: make parties solve a computational puzzle to add a block

Problems with PoW

Bitcoin energy consumption

Description	Value
Bitcoin's current estimated annual electricity consumption* (TWh)	56.71
Annualized global mining revenues	$7,043,523,805
Annualized estimated global mining costs	$2,835,321,741
Country closest to Bitcoin in terms of electricity consumption	Greece
Estimated electricity used over the previous day (KWh)	155,360,095
Implied Watts per GH/s	0.234
Total Network Hashrate in PH/s (1,000,000 GH/s)	27,620
Electricity consumed per transaction (KWh)	834.00
Number of U.S. households that could be powered by Bitcoin	5,250,596
Number of U.S. households powered for 1 day by the electricity consumed for a single transaction	28.17
Bitcoin's electricity consumption as a percentage of the world's electricity consumption	0.25%
Annual carbon footprint (kt of CO2)	27,786
Carbon footprint per transaction (kg of CO2)	408.42

Source: https://digiconomist.net/bitcoin-energy-consumption

Problems with PoW

Bitcoin energy consumption

Description	Value
Bitcoin's current estimated annual electricity consumption* (TWh)	56.71
Annualized global mining revenues	$7,043,523,805
Annualized estimated global mining costs	$2,835,321,741
Country closest to Bitcoin in terms of electricity consumption	Greece
Estimated electricity used over the previous day (KWh)	155,360,095
Implied Watts per GH/s	0.234
Total Network Hashrate in PH/s (1,000,000 GH/s)	27,620
Electricity consumed per transaction (KWh)	834.00
Number of U.S. households that could be powered by Bitcoin	5,250,596
Number of U.S. households powered for 1 day by the electricity consumed for a single transaction	28.17
Bitcoin's electricity consumption as a percentage of the world's electricity consumption	0.25%
Annual carbon footprint (kt of CO2)	27,786
Carbon footprint per transaction (kg of CO2)	408.42

Source: https://digiconomist.net/bitcoin-energy-consumption

Problems with PoW

PoW means that Bitcoin is slow

Average time to confirm a transaction: **1 hour**

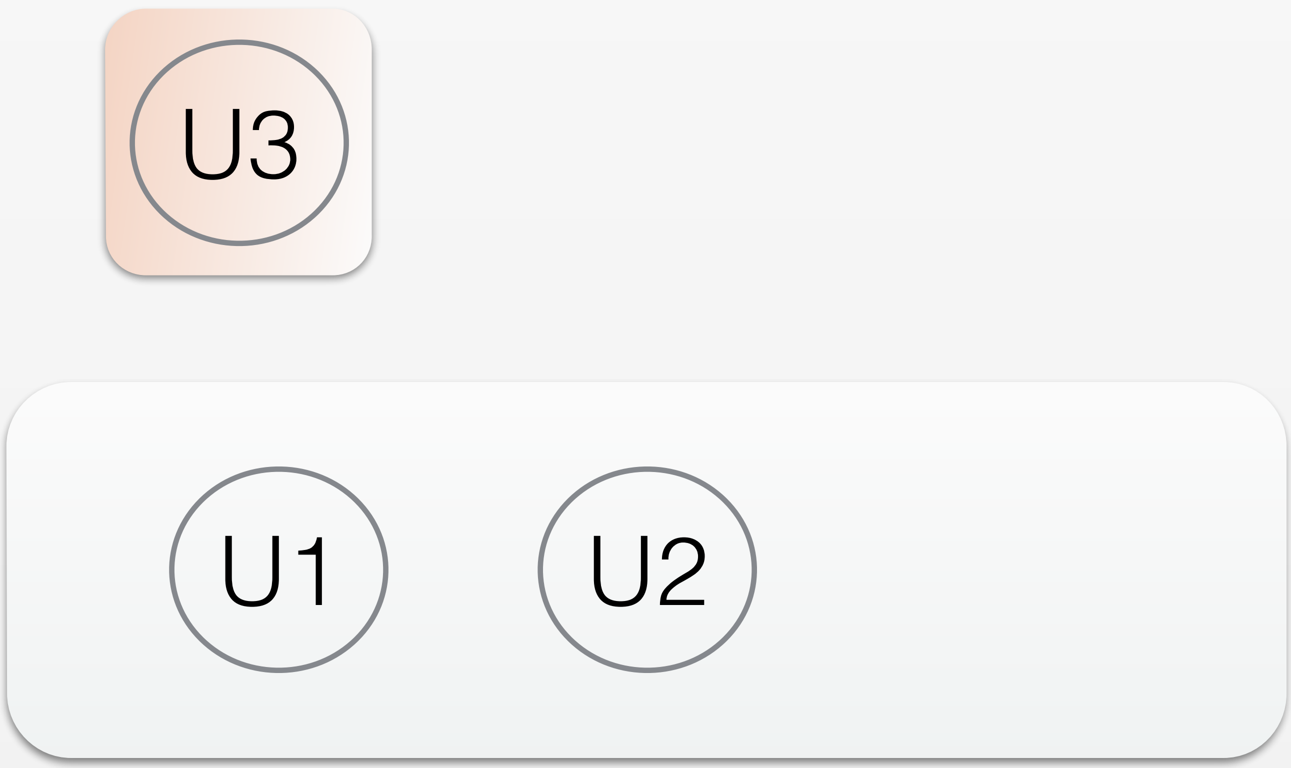
Doing it differently

PoW solves consensus by making it expensive for the parties to add a block

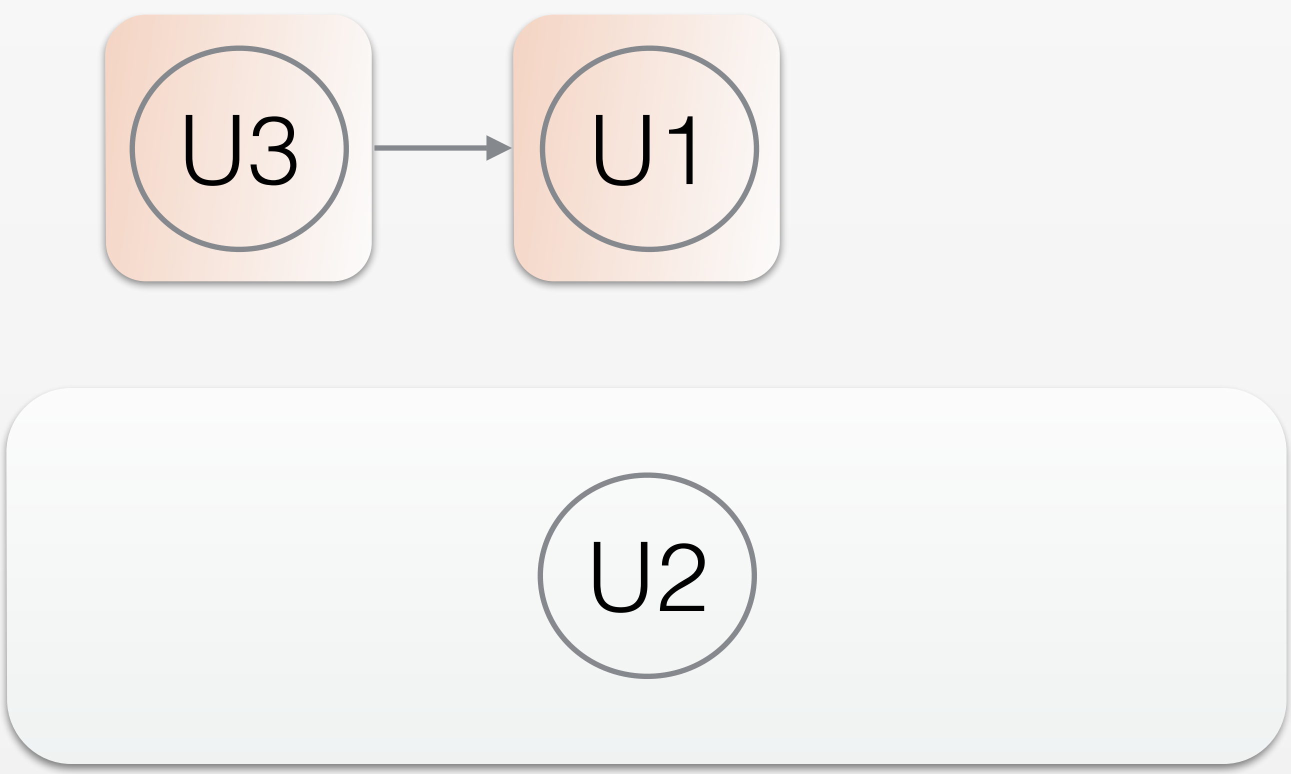
... could we substitute it for something else which requires effort from the parties?

Who can extend the chain?

Who can extend the chain?



Who can extend the chain?



Who can extend the chain?

Who can extend the chain?

In PoW: party which extends the chain chosen at random proportionally to hash rate

Who can extend the chain?

In PoW: party which extends the chain chosen at random proportionally to hash rate

In PoS: choose the party at random proportionally to the amount of stakes it possesses

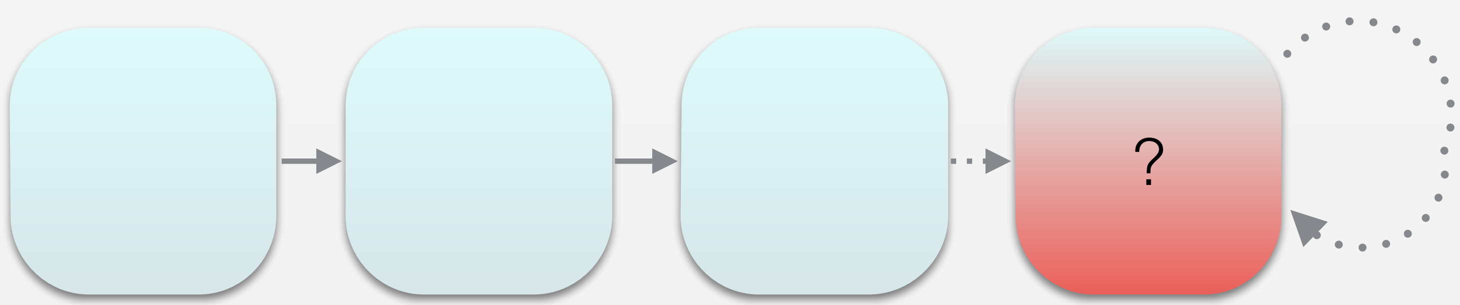
The idea behind Proof-of-Stake

* current stakeholder distribution taken directly from the ledger

* a randomised selection process will determine the stakeholder(s) which may append the next block(s) (leaders)

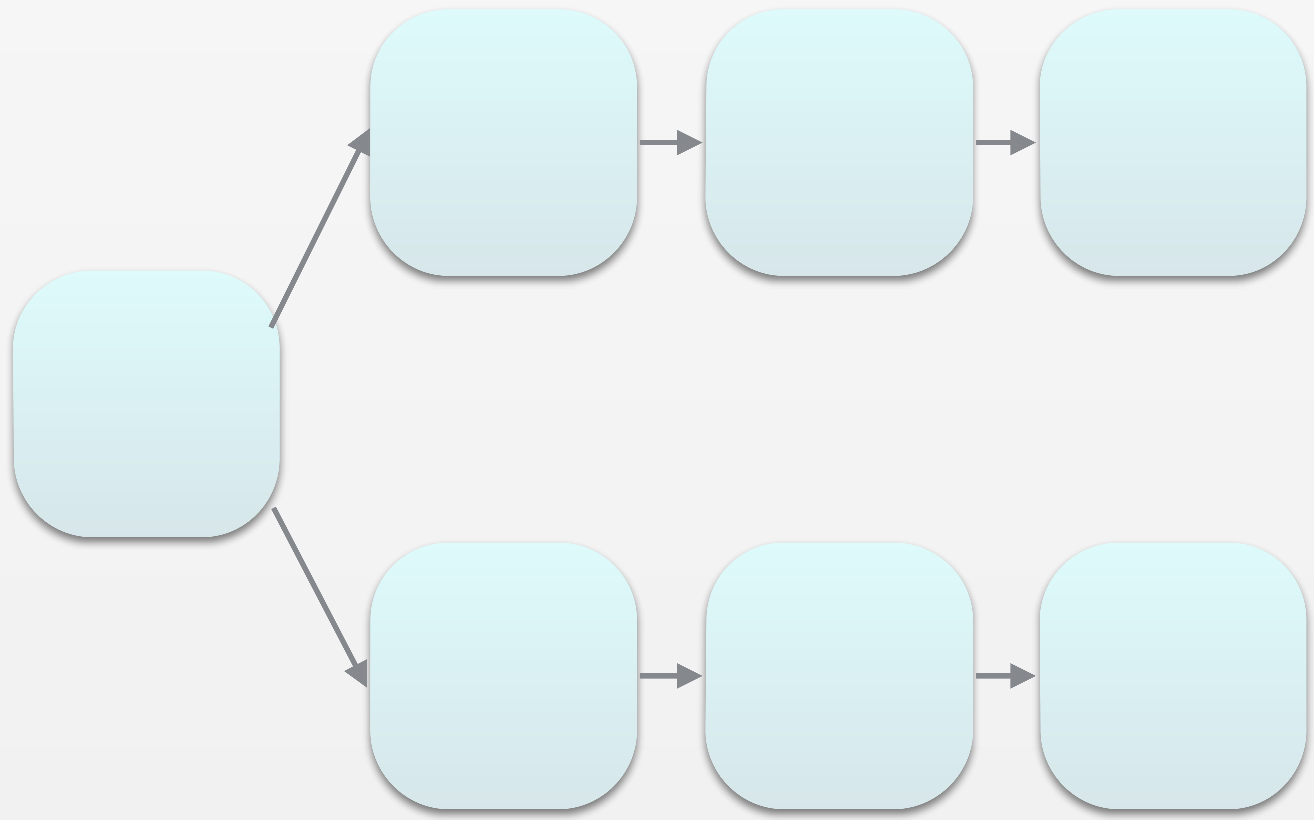
PoS design challenge 1

grinding attacks: an adversary may try to bias the randomised leader election



PoS design challenge 2

nothing-at-stake attacks: no effort to add a block, may add blocks on multiple histories

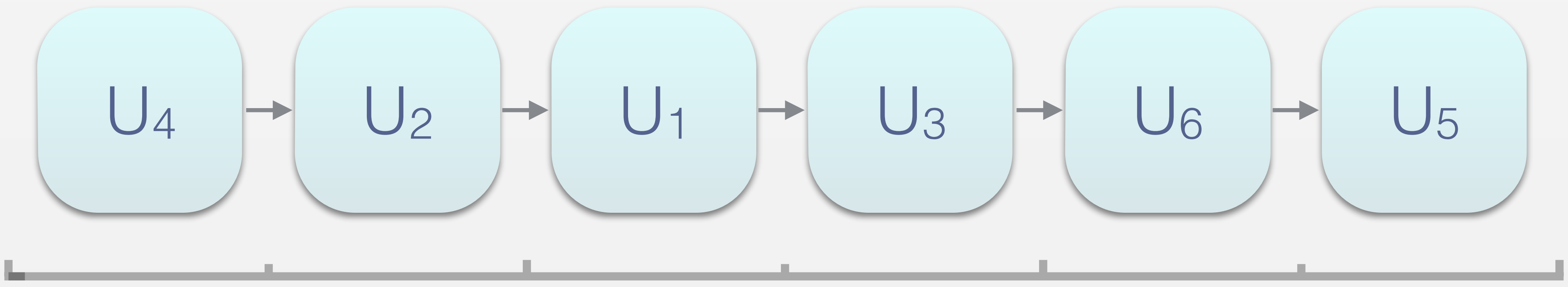


Moving on to the protocol presented in the paper ...

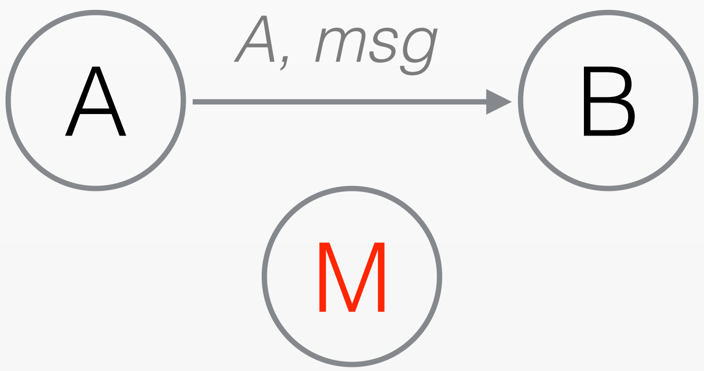
Setting

Synchronous setting:
- division into time slots
- one leader elected per time slot -> each slot one block can be added to the chain

(requires some kind of clock synchronisation)



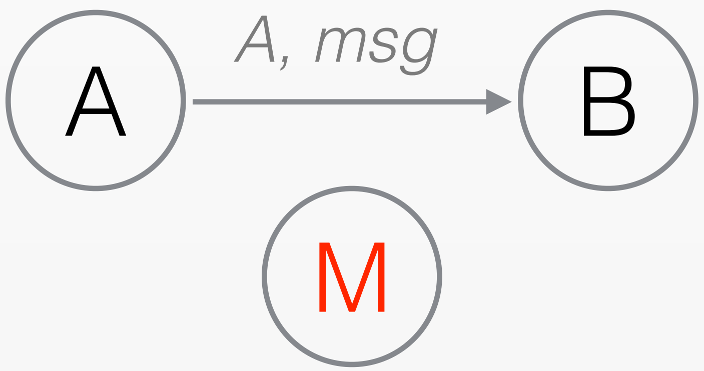
Setting



Adversary

Assume a single adversary who:

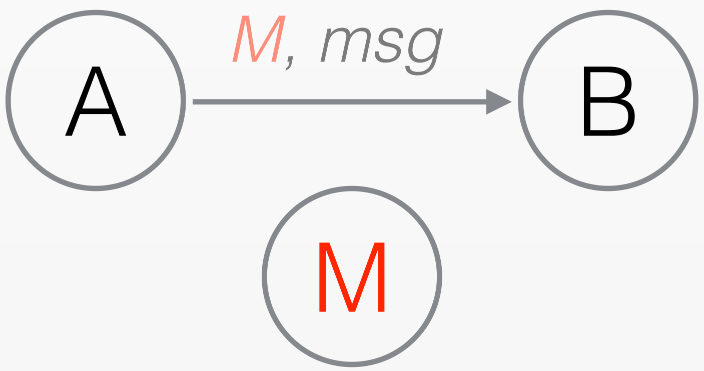
Setting



Adversary

Assume a single adversary who:
 * can change the sender of a message (spoof)

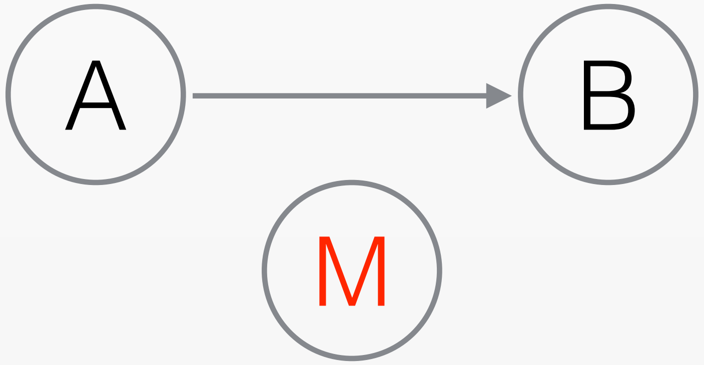
Setting



Adversary

Assume a single adversary who:
 * can change the sender of a message (spoof)

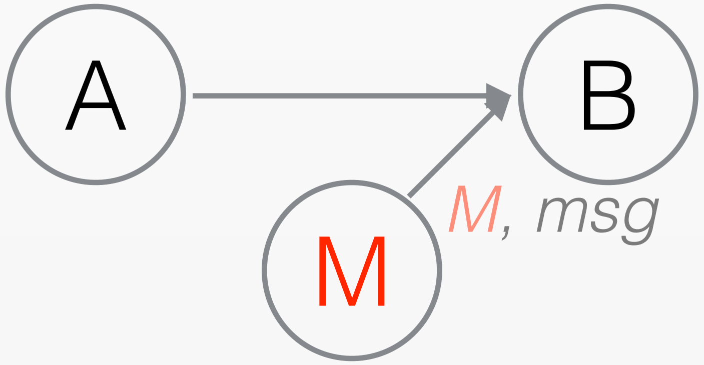
Setting



Adversary

Assume a single adversary who:
* can change the sender of a message (spoof)
* can create new messages (inject)

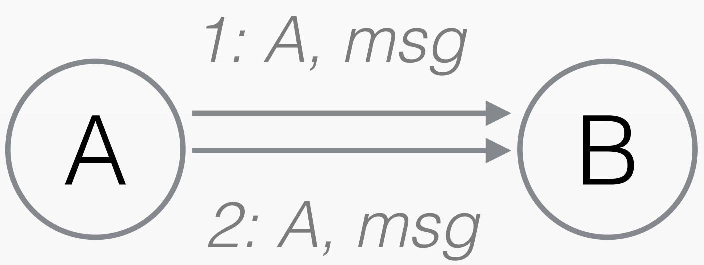
Setting



Adversary

Assume a single adversary who:
  * can change the sender of a message (spoof)
  * can create new messages (inject)

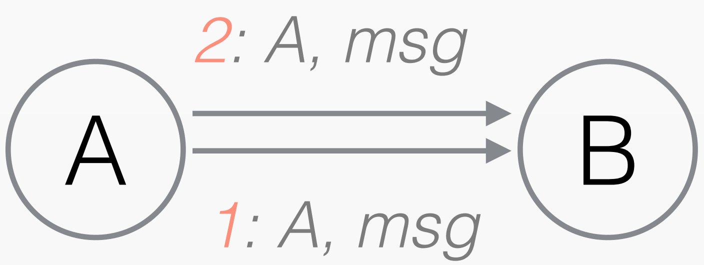
Setting



Adversary

Assume a single adversary who:
* can change the sender of a message (spoof)
* can create new messages (inject)
* can reorder messages

Setting



Adversary

Assume a single adversary who:

* can change the sender of a message (spoof)
* can create new messages (inject)
* can reorder messages

Setting

Adversary

Assume a single adversary who:
 * can change the sender of a message (spoof)
 * can create new messages (inject)
 * can reorder messages

BUT
 * <u>cannot</u> withhold messages of honest parties

Designing the protocol

design a protocol which works under certain assumptions

prove security properties of that protocol

relax assumptions and use the protocol to make an inductive claim

presented in four stages

Stage 1 - STATIC

Assumptions:

1. stake distribution is fixed at the beginning
2. adversary is static (i.e. a fixed number of adversarial nodes)

Stage 1

Leader election

Genesis block

U_1, s_1

...

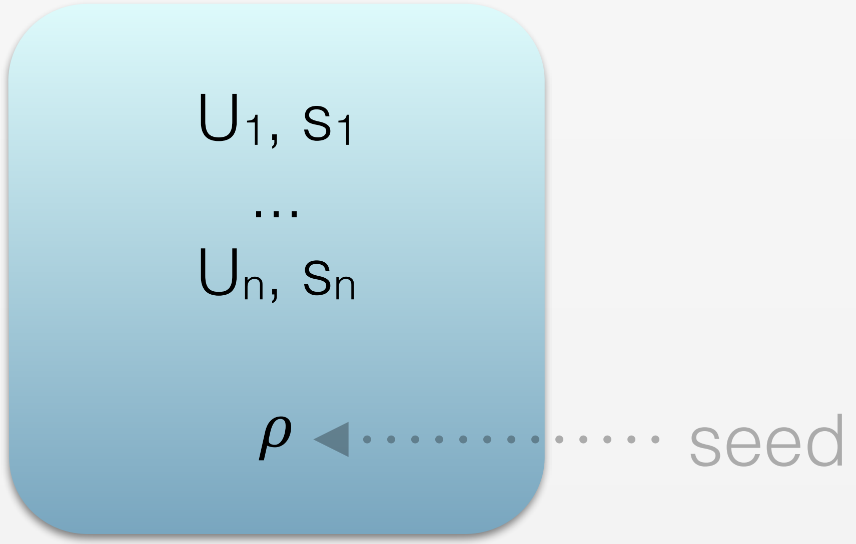
U_n, s_n

ρ

Stage 1

Leader election

Genesis block



Stage 1

Leader election

U_1, s_1

…

U_5, s_5

ρ

Stage 1

Leader election

U_1, s_1
...
U_5, s_5

ρ

Slot leaders computed by each party using a deterministic function and the seed.

Each party is elected with probability:
the party's stake / total stake

Stage 1

Leader election

U_1, s_1
…
U_5, s_5

ρ

Consider the following distribution:

$s_1 = 1/8$
$s_2 = 1/8$
$s_3 = 1/8$
$s_4 = 1/8$
$s_5 = 1/2$

Stage 1

Leader election

U_1, s_1
...
U_5, s_5

ρ

Consider the following distribution:

$s_1 = 1/8$	000
$s_2 = 1/8$	001
$s_3 = 1/8$	010
$s_4 = 1/8$	011
$s_5 = 1/2$	100, 101, 110, 111

Stage 1

Leader election

U_1, s_1
...
U_5, s_5

ρ

Consider the following distribution:

$s_1 = 1/8$	000
$s_2 = 1/8$	001
$s_3 = 1/8$	010
$s_4 = 1/8$	011
$s_5 = 1/2$	100, 101, 110, 111

ρ = 011 001 000 010 111 ...

Stage 1

Leader election

U_1, s_1
…
U_5, s_5

ρ

Consider the following distribution:

$s_1 = 1/8$	000
$s_2 = 1/8$	001
$s_3 = 1/8$	010
$s_4 = 1/8$	011
$s_5 = 1/2$	100, 101, 110, 111

ρ = 011 001 000 010 111

(in reality a bit more complicated)

Stage 1

Leader election

U_1, s_1
…
U_5, s_5

ρ

U_4 U_2 U_1 U_3 U_5

Stage 1

Chain extension

U_1, s_1

…

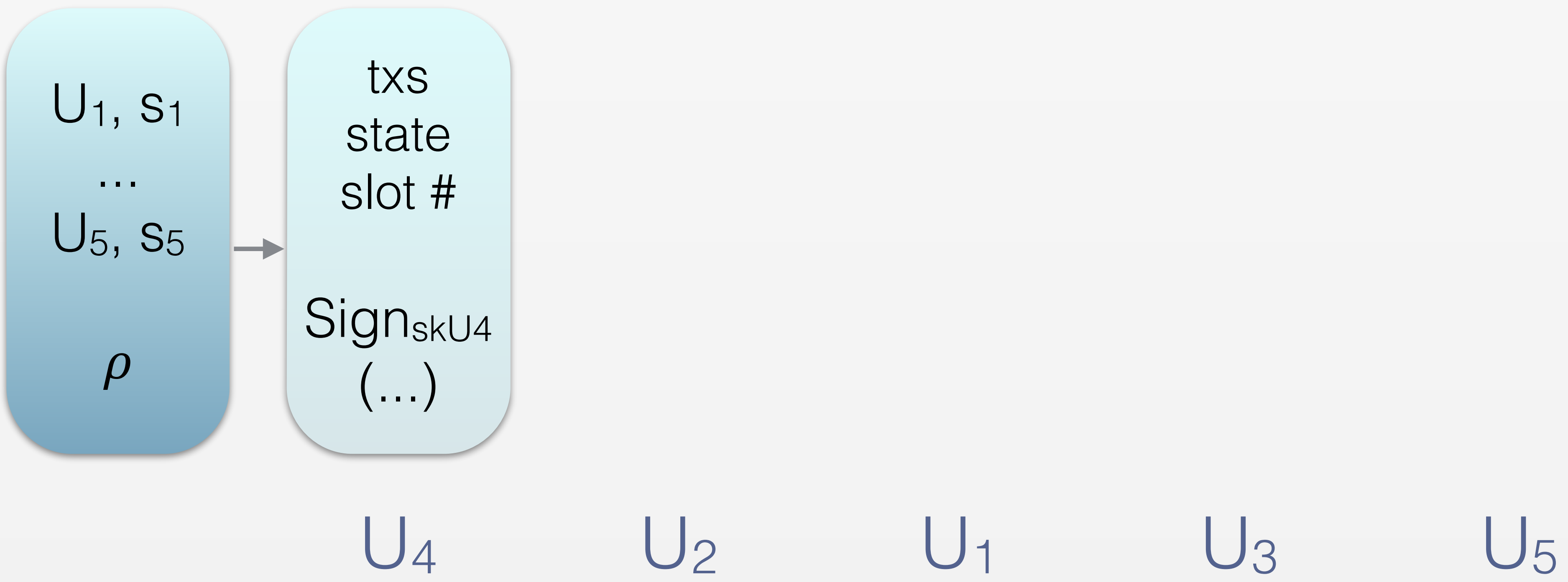
U_5, s_5

ρ

U_4 U_2 U_1 U_3 U_5

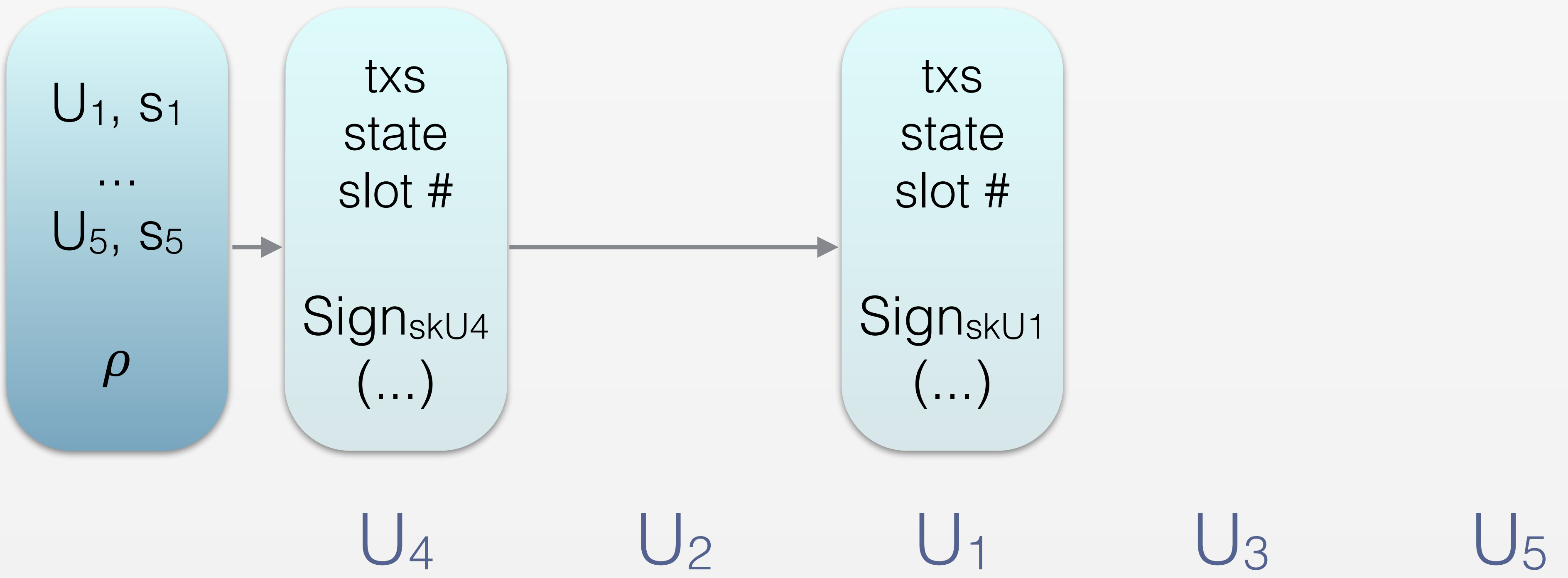
Stage 1

Chain extension



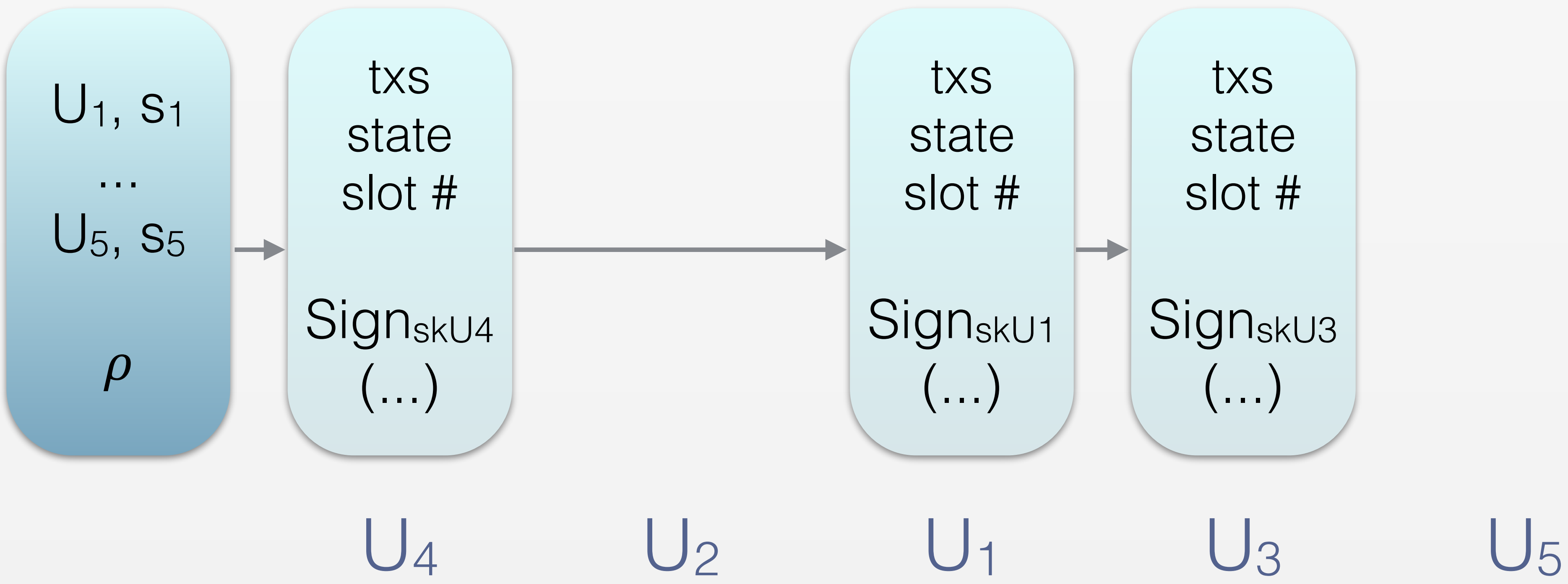
Stage 1

Chain extension



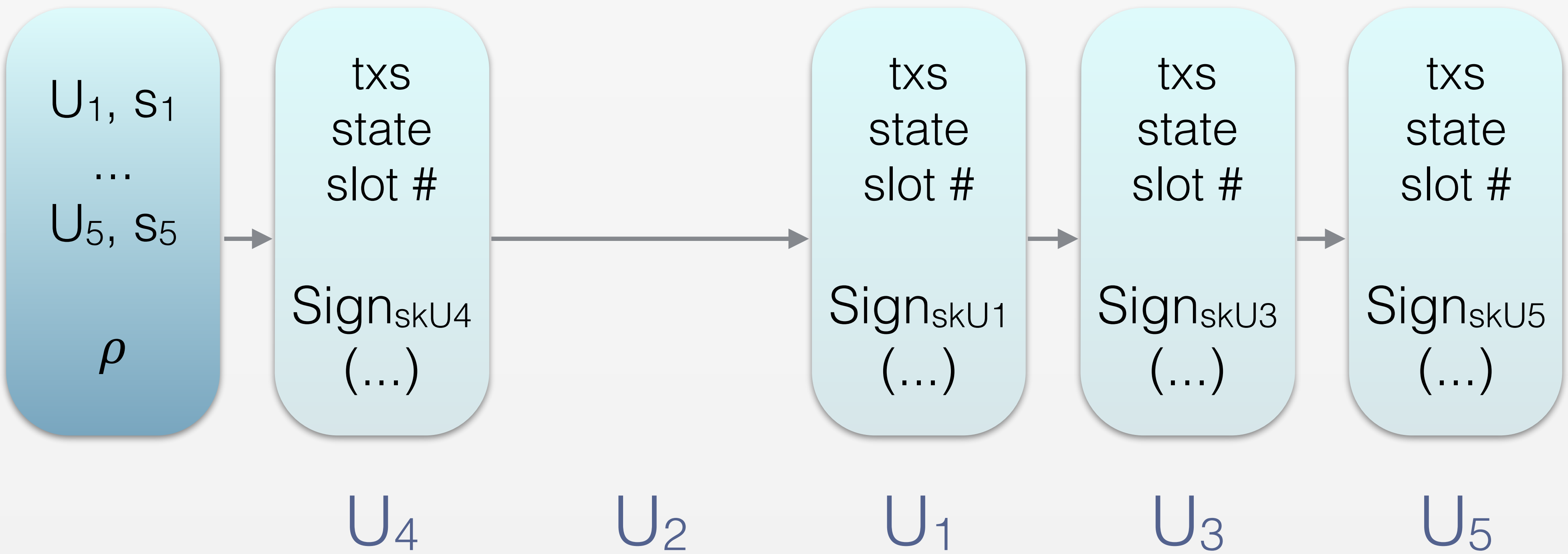
Stage 1

Chain extension



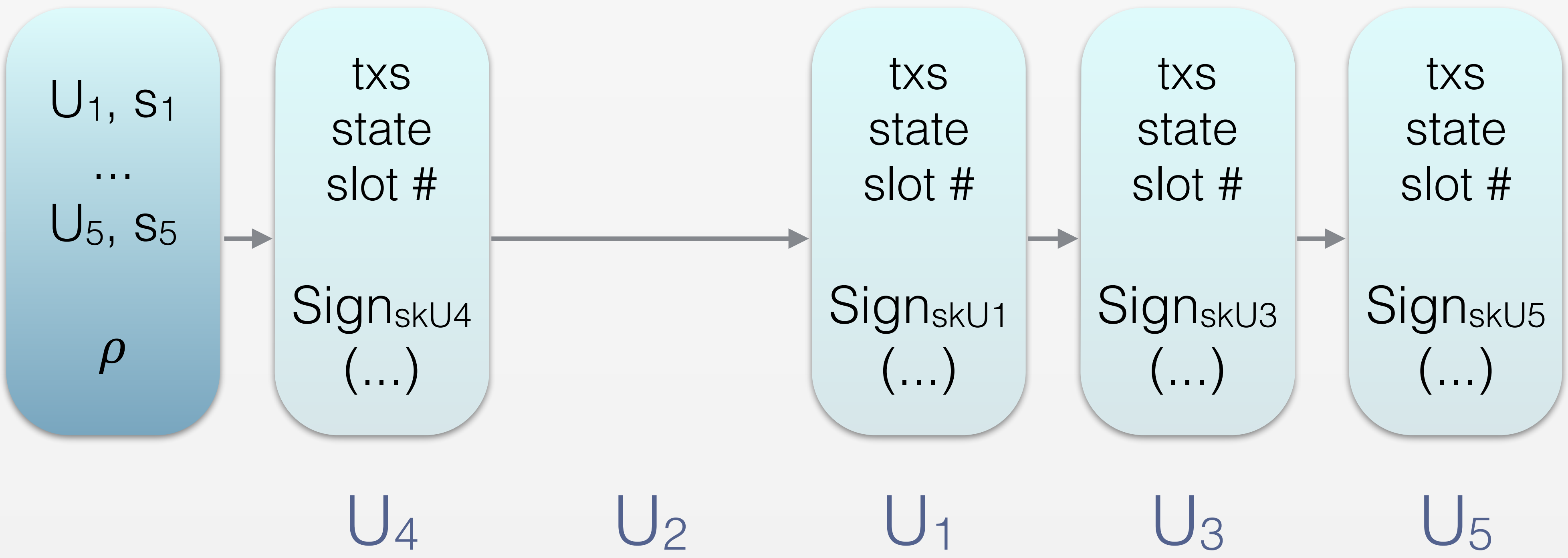
Stage 1

Chain extension



Stage 1

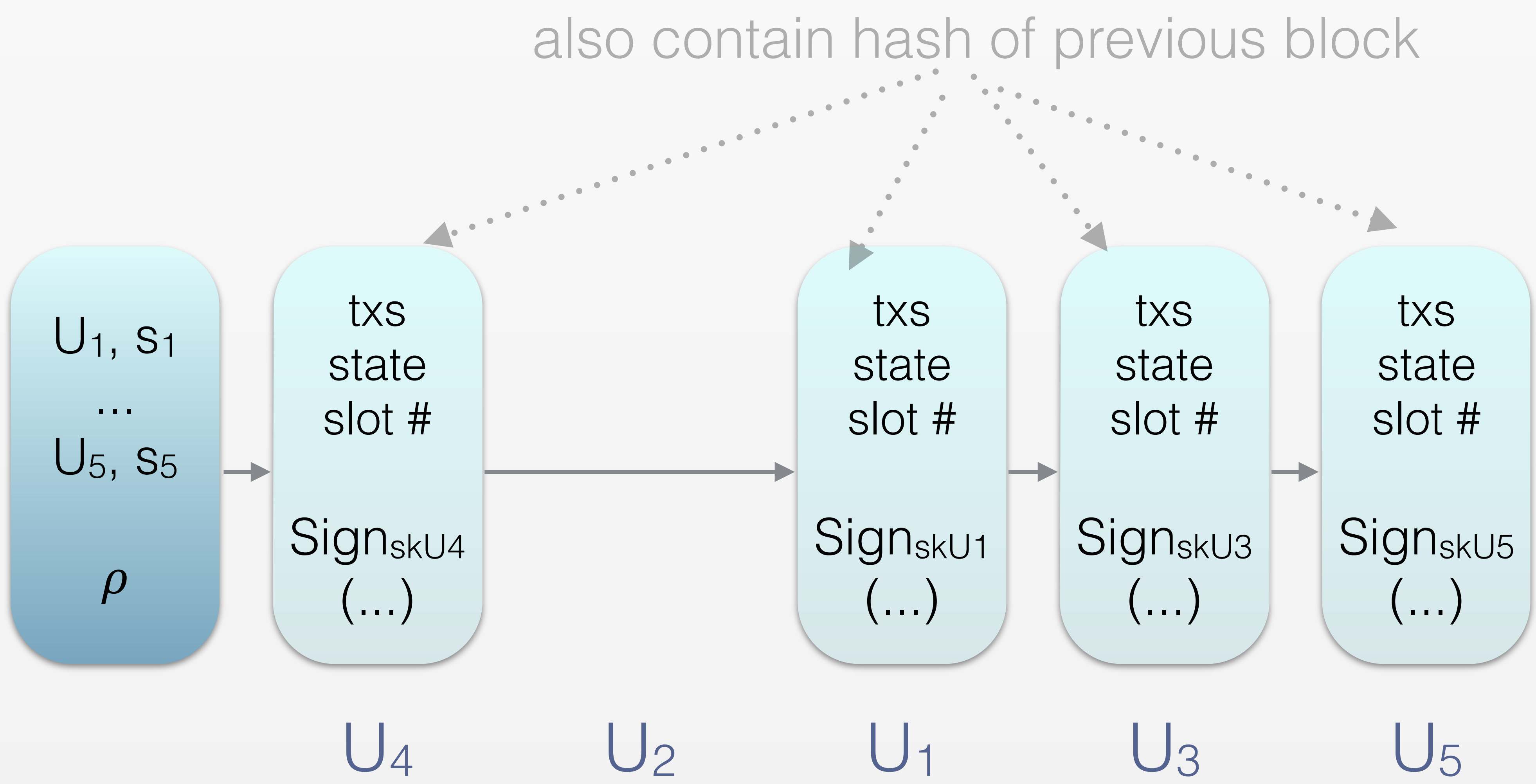
Chain extension



U_i are identified by their verification key (public key) vk_{U_i}
Each block's content is signed with the leader's signing key (secret key) sk_{U_i}

Stage 1

Chain extension



U_i are identified by their verification key (public key) vk_{U_i}
Each block's content is signed with the leader's signing key (secret key) sk_{U_i}

Stage 1

A Stronger Adversary

Note that the adversary is much stronger in the PoS setting than in the PoW setting:

* knows entire leader sequence in advance

* can generate multiple blocks per slot without any cost

* may choose to withhold information

Stage 1

Security Analysis

Stage 1

Security Analysis

But wait...

... what do we actually want to prove?

Stage 1

Robust Transaction Ledger

Stage 1

Robust Transaction Ledger

Persistence: if one party has confirmed a transaction as stable, all the other parties will (eventually) confirm it in the same position on the ledger.

Stage 1

Robust Transaction Ledger

Persistence: if one party has confirmed a transaction as stable, all the other parties will (eventually) confirm it in the same position on the ledger

Liveness: If all honest nodes attempt to include a transaction, then eventually all nodes responding honestly will report the transaction as stable

Notion of robust transaction ledger formally defined in: Garay, J., Kiayias, A. and Leonardos N. *The Bitcoin Backbone Protocol: Analysis and Applications*, 2014, https://eprint.iacr.org/2014/765

Robust Transaction Ledger

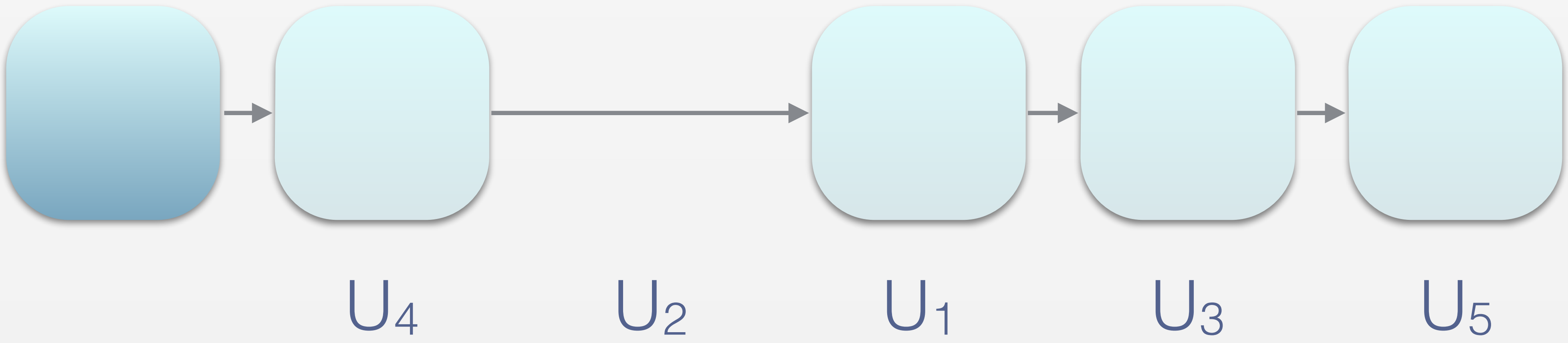
Common Prefix: given two parties and their chains, then removing k blocks from one chain will result it being a prefix of the other

Chain Quality: ratio between adversary's and honest blocks is bounded

Chain Growth: the chain will continue to grow by a certain rate

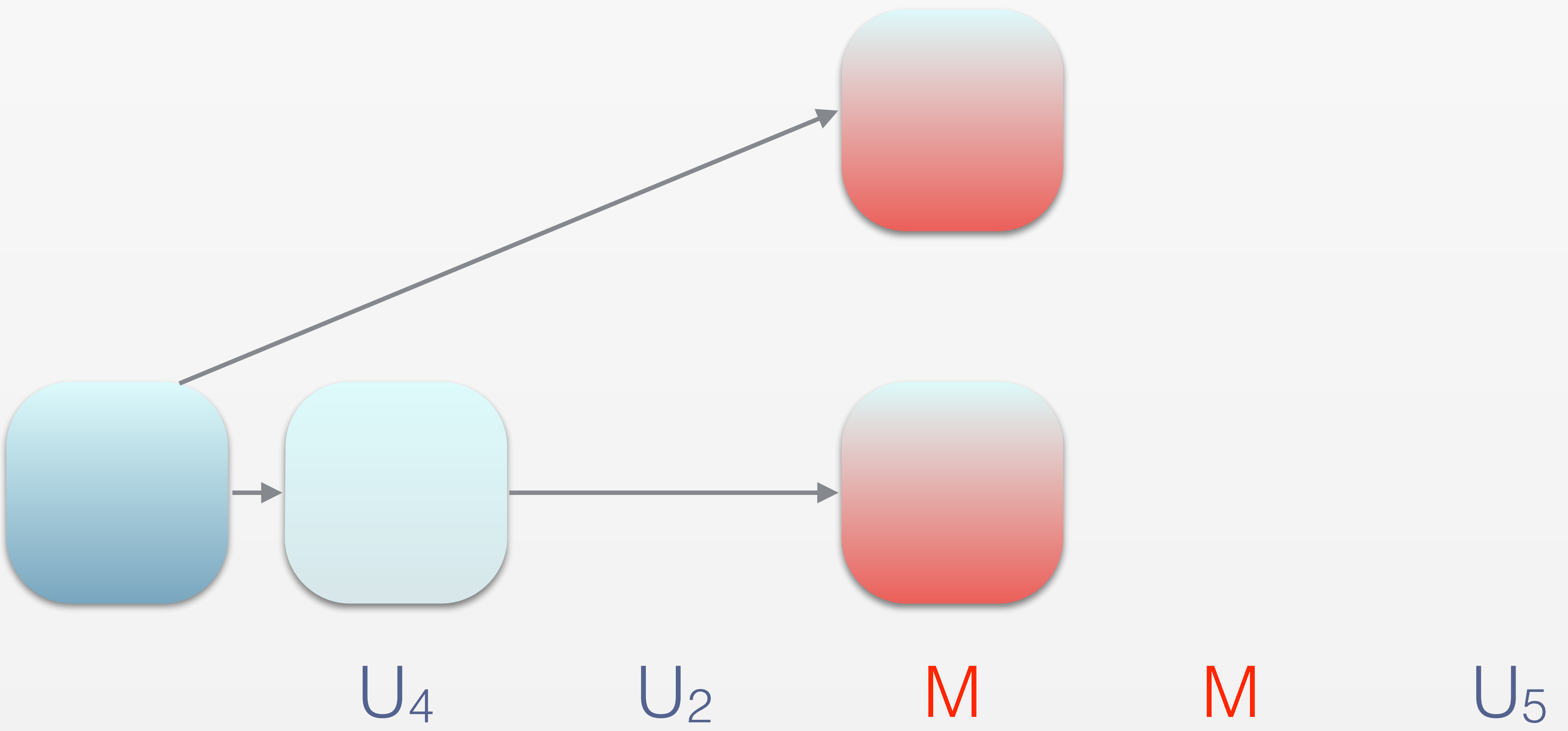
Stage 1

What if an adversary extends the chain



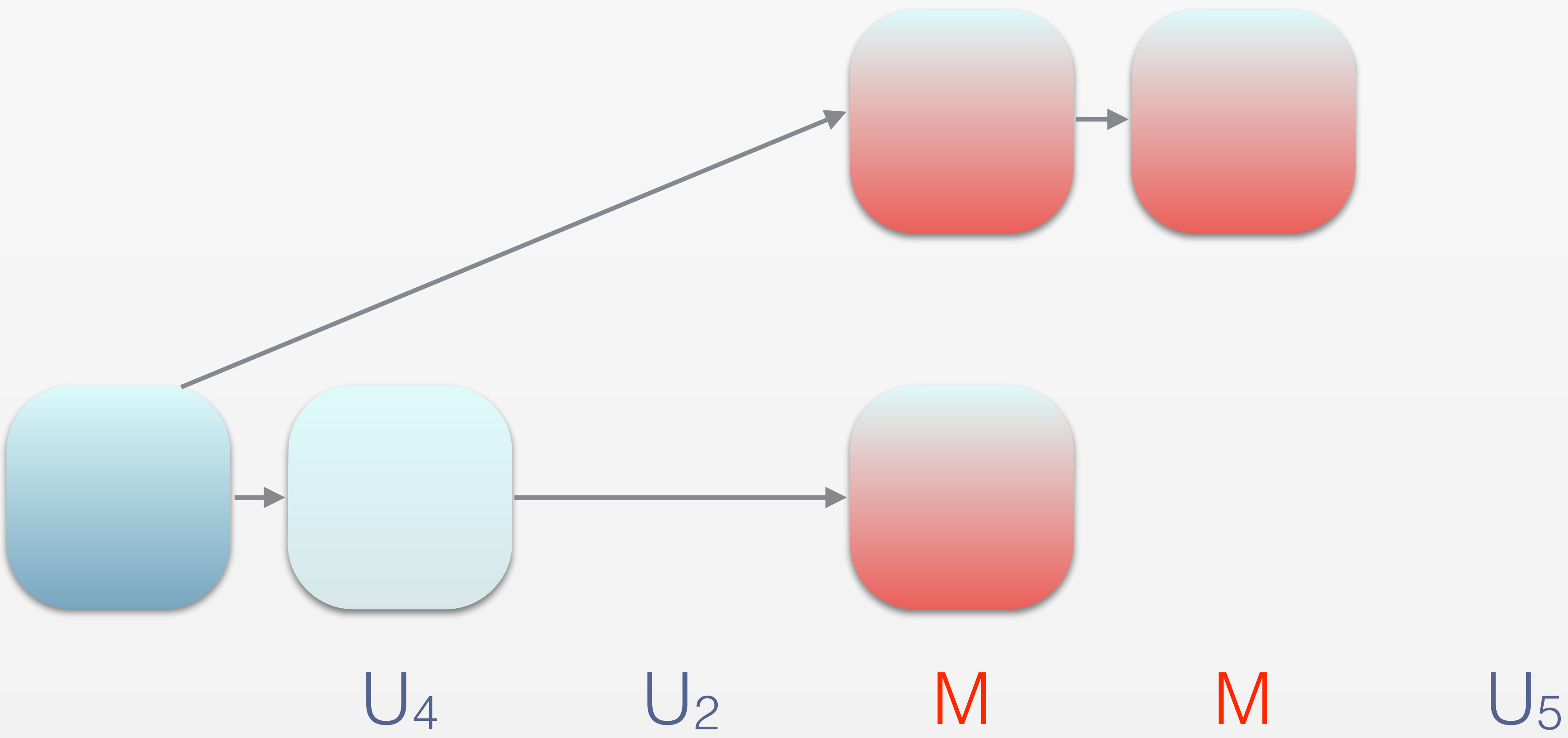
Stage 1

What if an adversary extends the chain



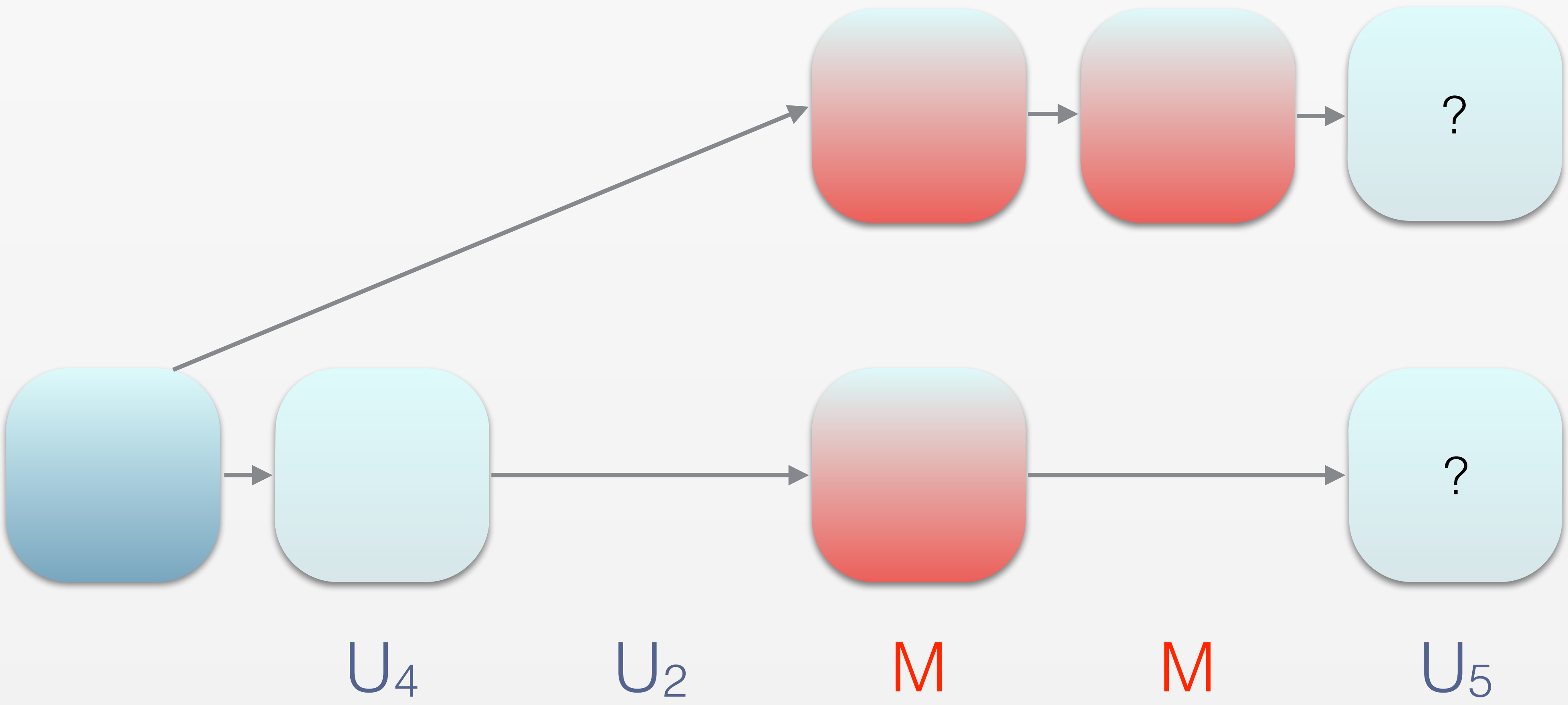
Stage 1

What if an adversary extends the chain



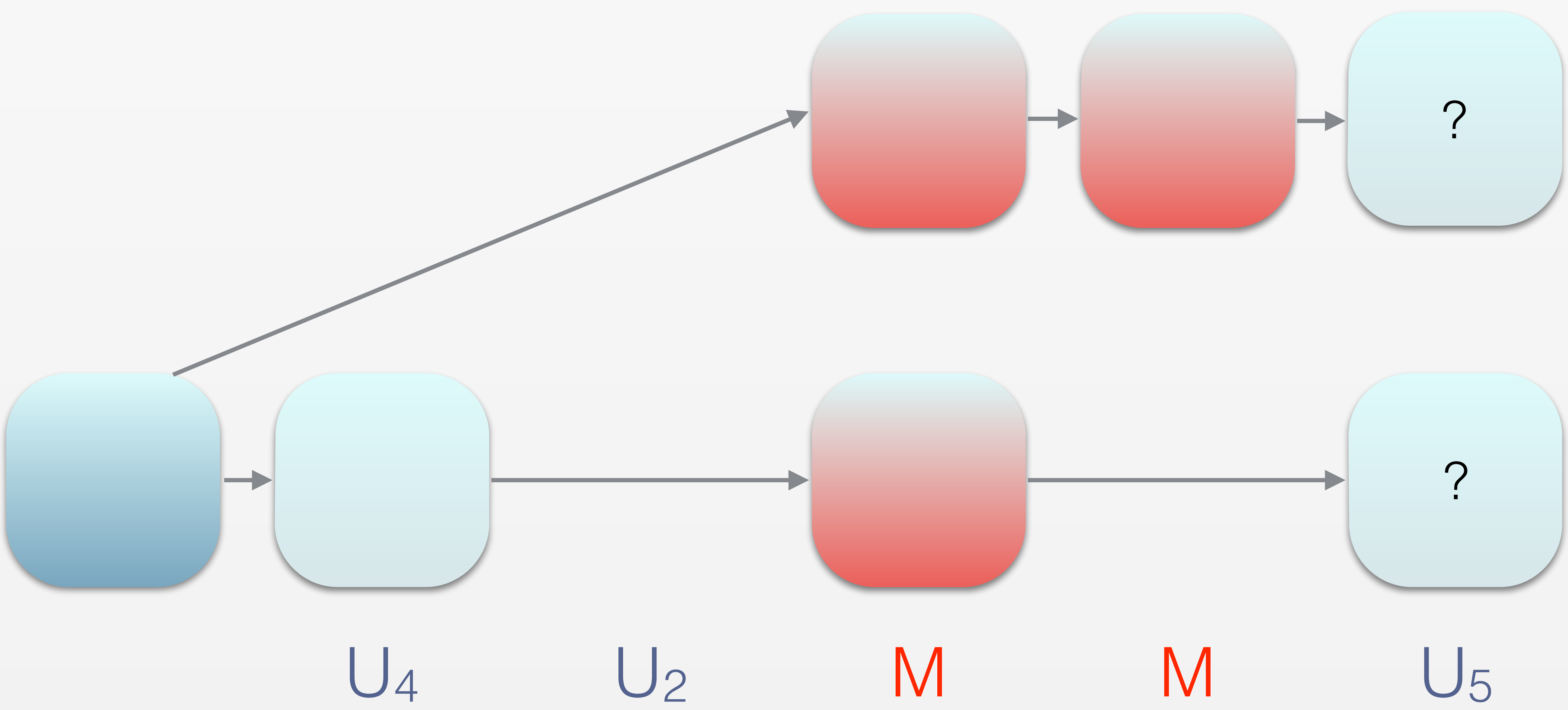
Stage 1

What if an adversary extends the chain



Stage 1

What if an adversary extends the chain



How can we analyse the likelihood of this event?

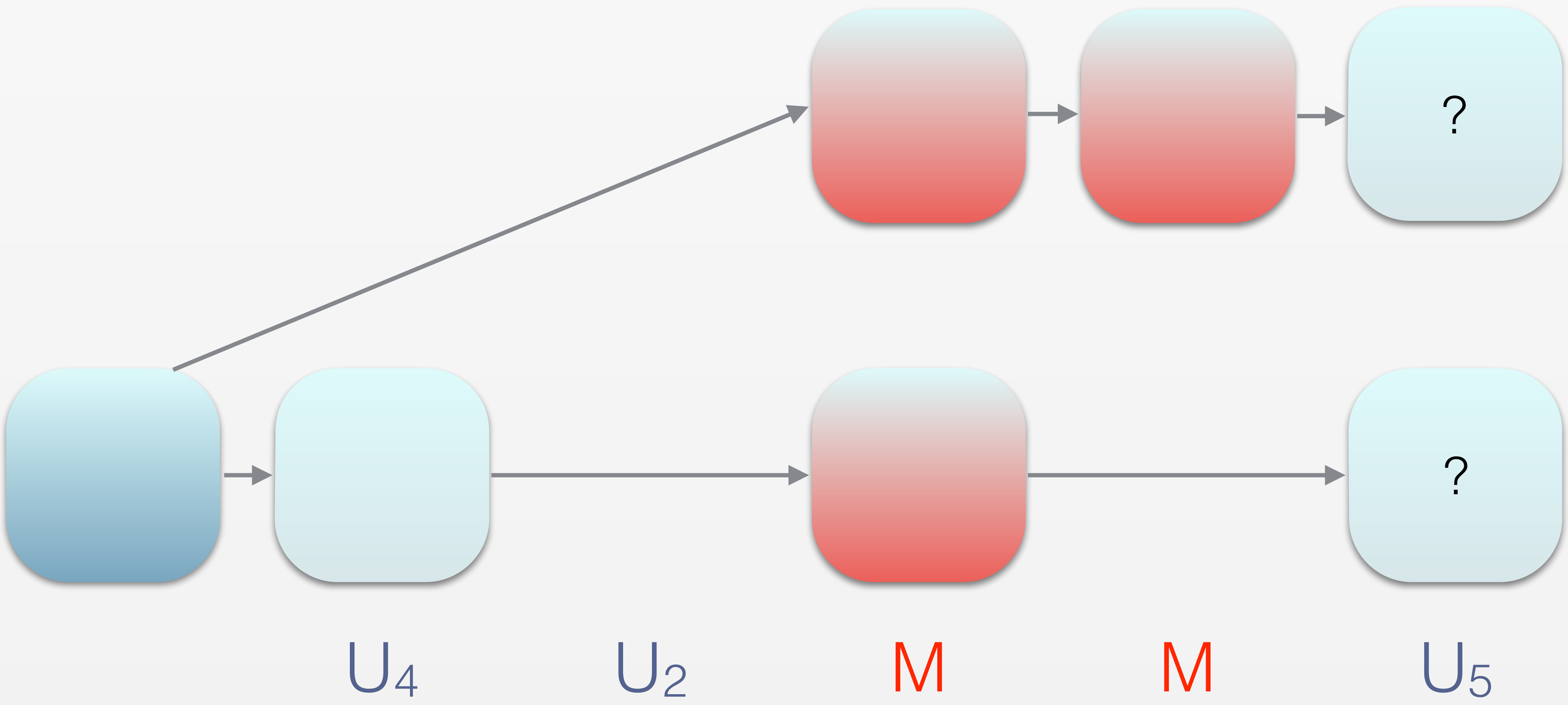
Stage 1

Forkable Strings

Reducing it to a combinatorial problem using the notion of forkable strings

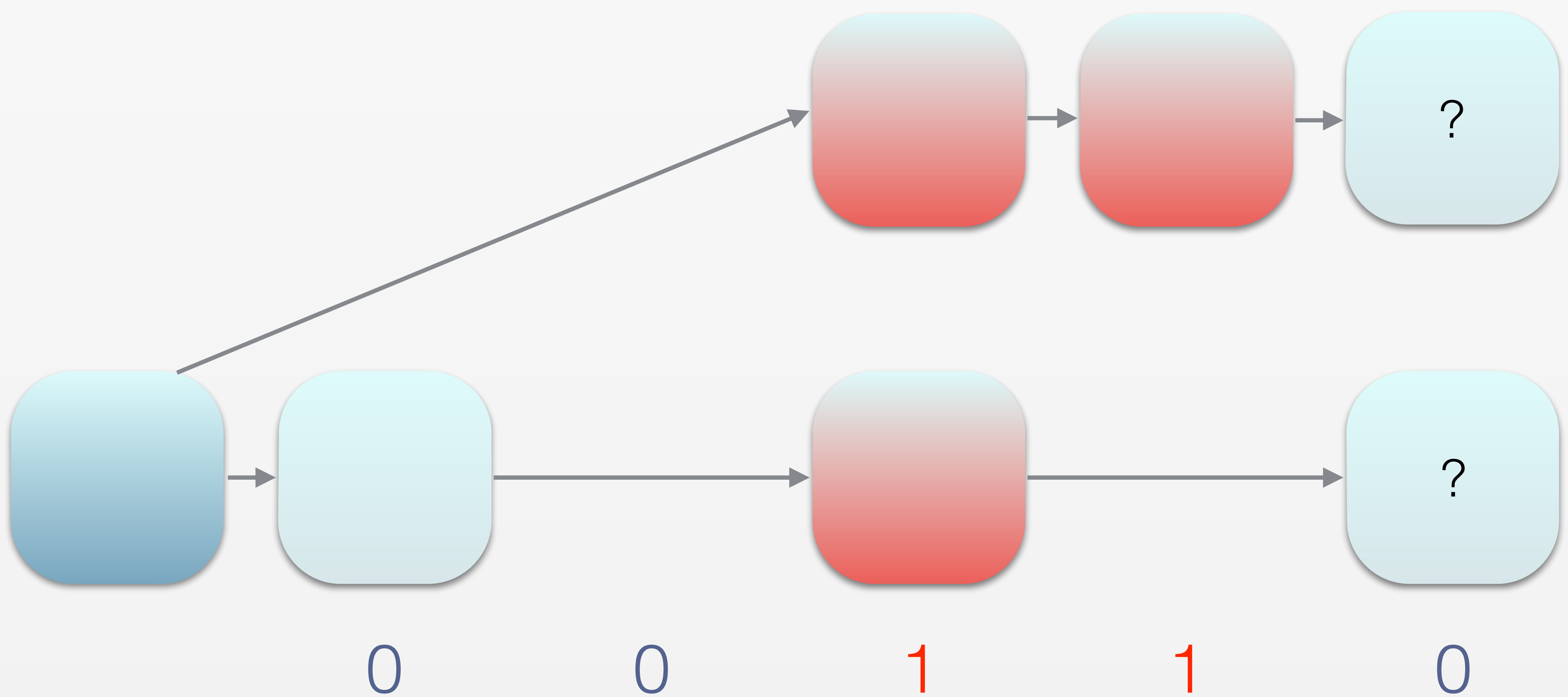
Stage 1

Forkable Strings



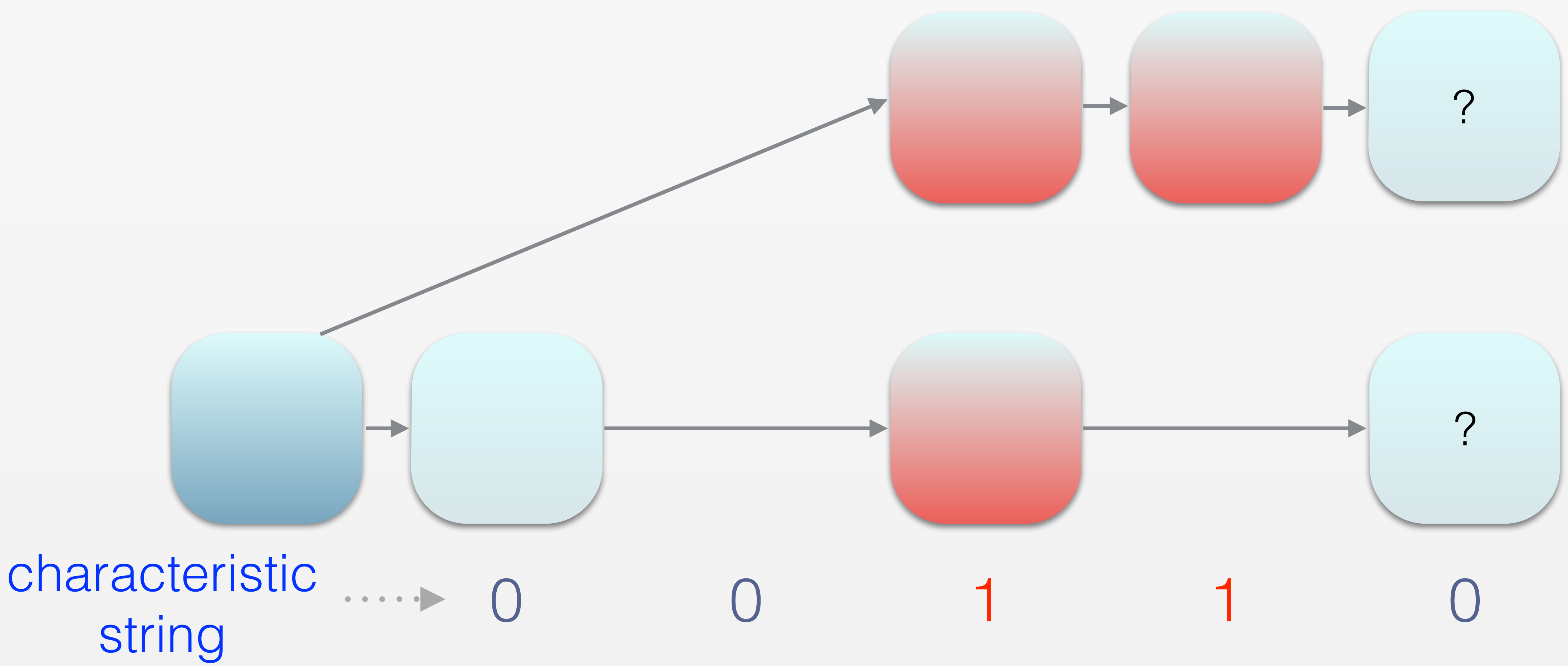
Stage 1

Forkable Strings



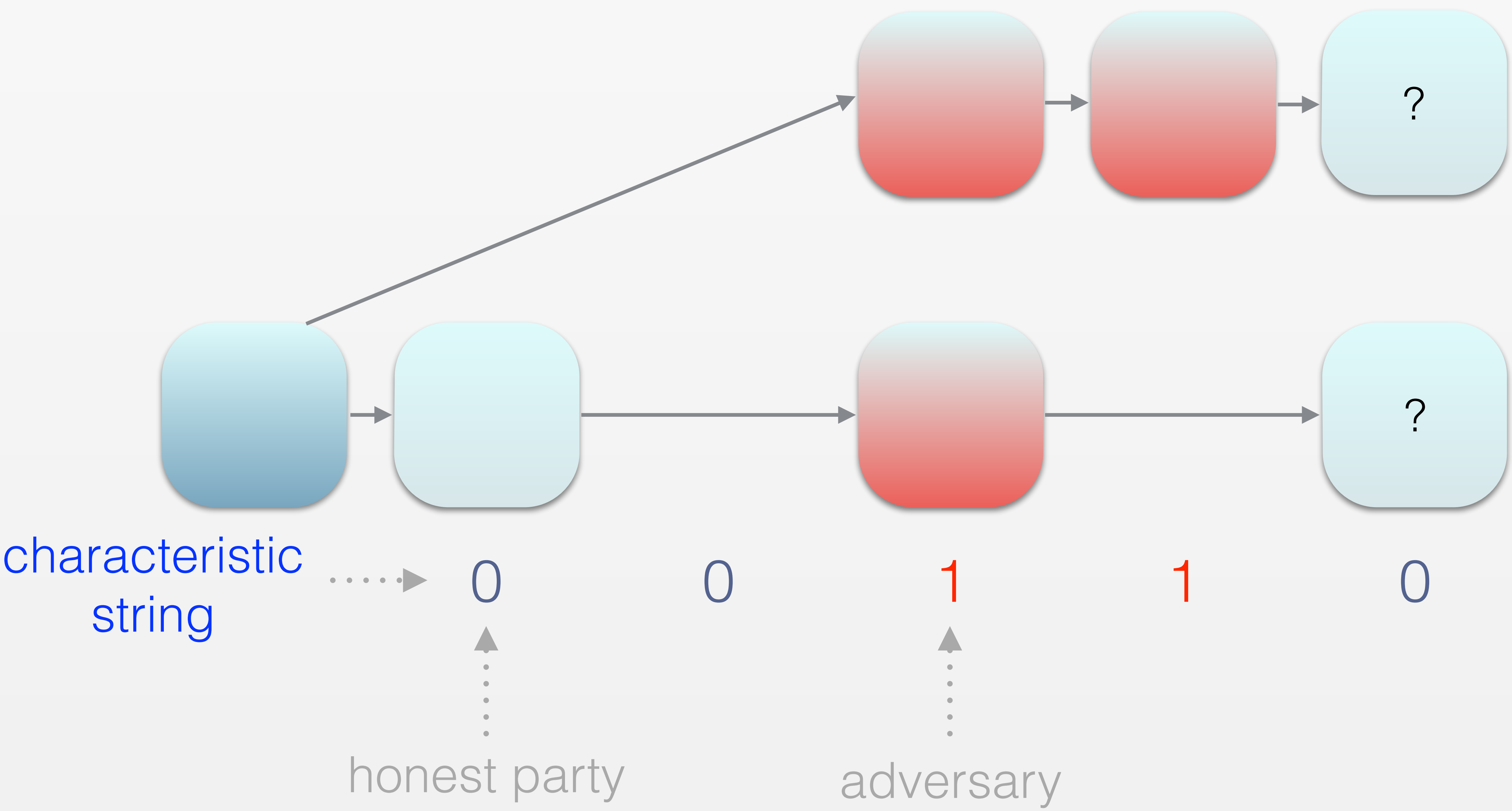
Stage 1

Forkable Strings



Stage 1

Forkable Strings



Stage 1

Forkable Strings

We call a characteristic string forkable iff there exists a possible fork where at least two branches have the same maximum length

Stage 1

Forkable Strings

We call a characteristic string forkable iff there exists a possible fork where at least two branches have the same maximum length

* a string is never forkable if there are < 1/3 1's
  (= adversarial nodes)
* a string is always forkable if there are ≥ 1/2 1's

Stage 1

Forkable Strings

We call a characteristic string forkable iff there exists a possible fork where at least two branches have the same maximum length

* a string is never forkable if there are < 1/3 1's (= adversarial nodes)
* a string is always forkable if there are ≥ 1/2 1's

AND

* the density of forkable strings decreases exponentially in its length

Stage 1

Forkable Strings

Through this combinatorial notion of forkable strings, able to prove (with overwhelming probability):

* common prefix
* chain growth
* chain quality

In other words, the properties which make a robust transaction ledger!

Taking it further ...

Stage 2 - DYNAMIC

Until now:
  * finite chain! Need to be able to add more blocks...

Stage 2 - DYNAMIC

Until now:

* finite chain! Need to be able to add more blocks...
* static stakeholder distribution! Stakes change over time...

Stage 2 - DYNAMIC

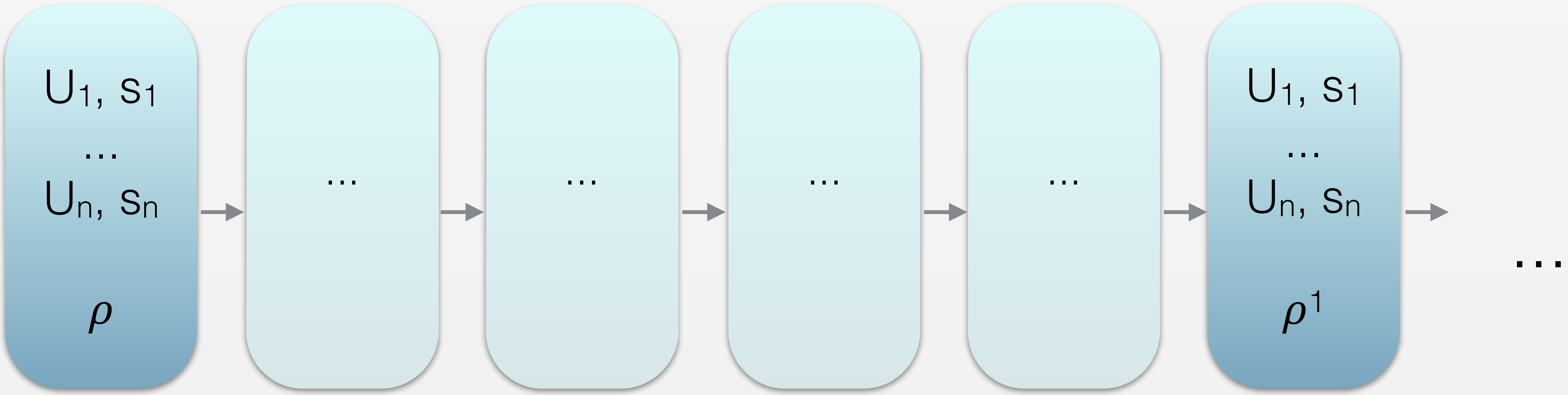
Until now:
* finite chain! Need to be able to add more blocks...
* static stakeholder distribution! Stakes change over time...

Idea:
call the finite number of blocks an epoch
elect leaders for an epoch at a time

→ need a new seed for each new stakeholder election (provided by a trusted beacon)

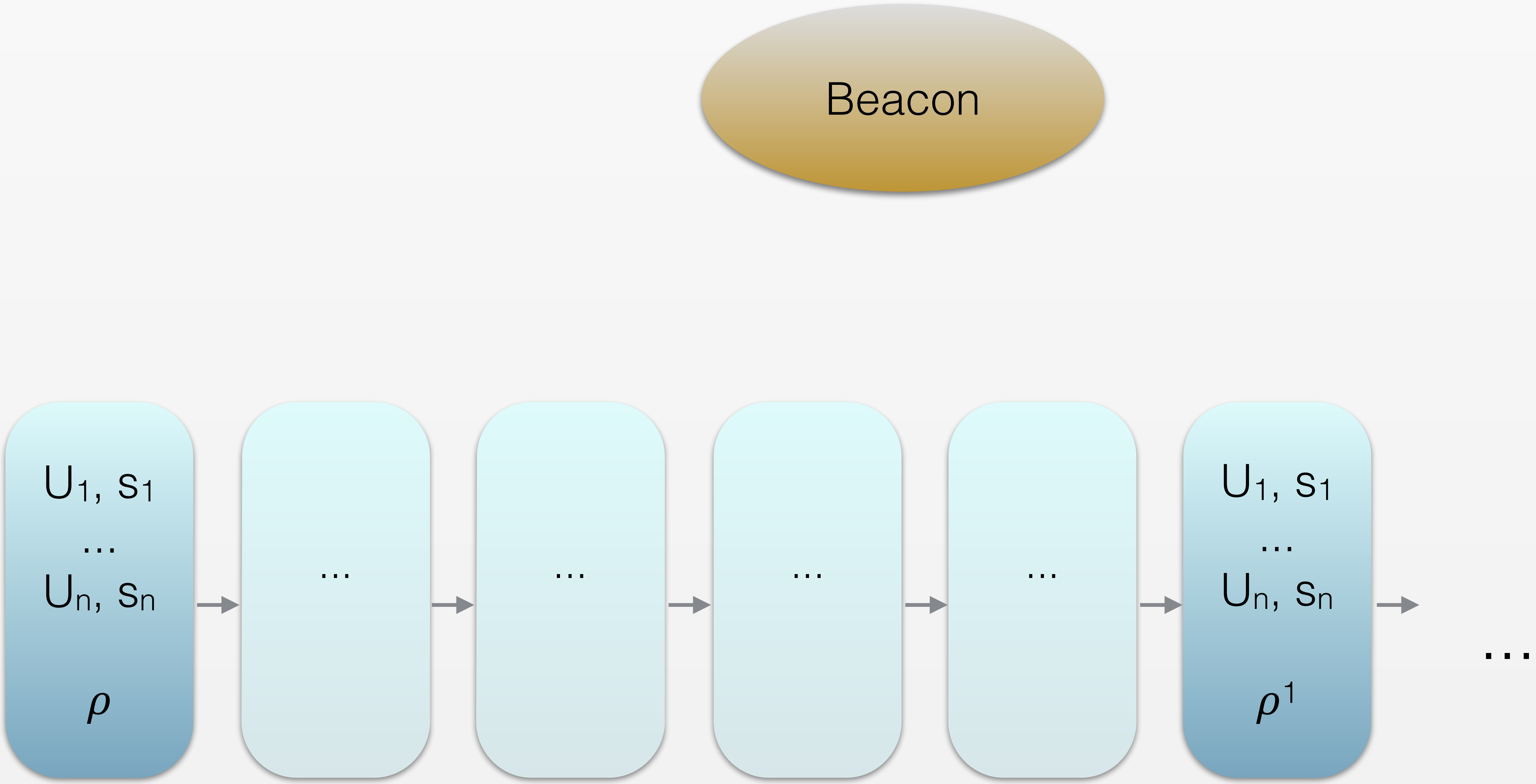
Stage 2

Trusted beacon



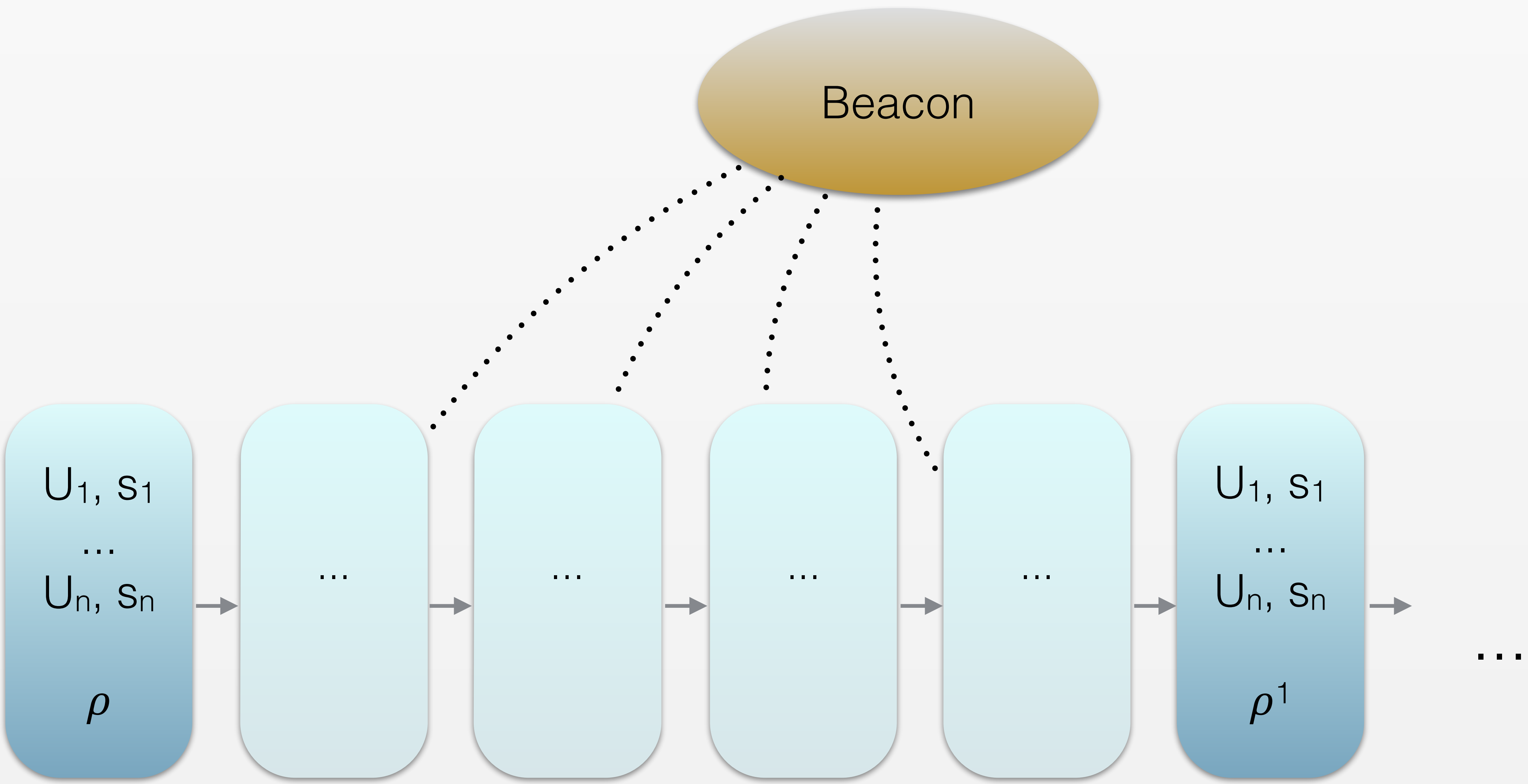
Stage 2

Trusted beacon



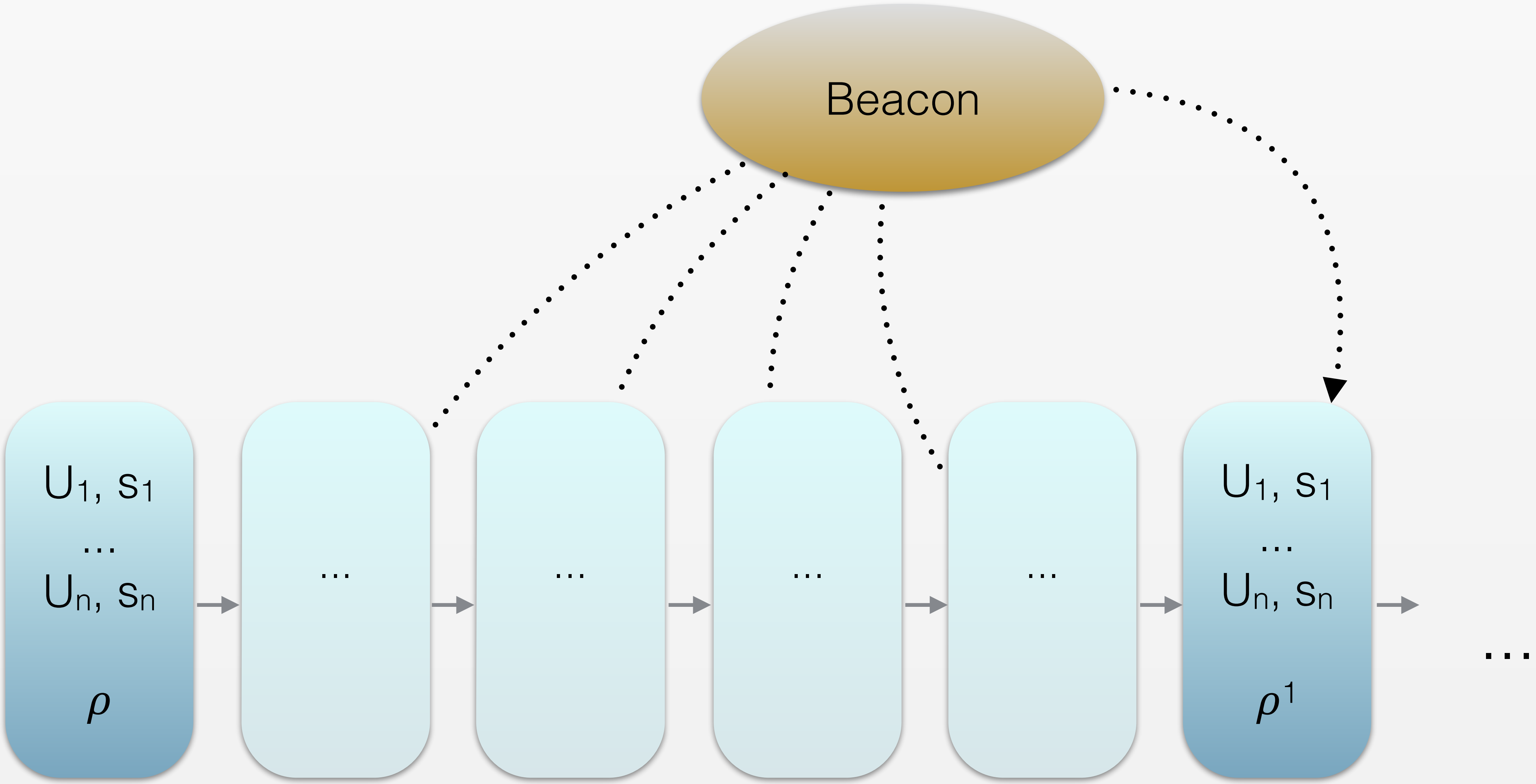
Stage 2

Trusted beacon



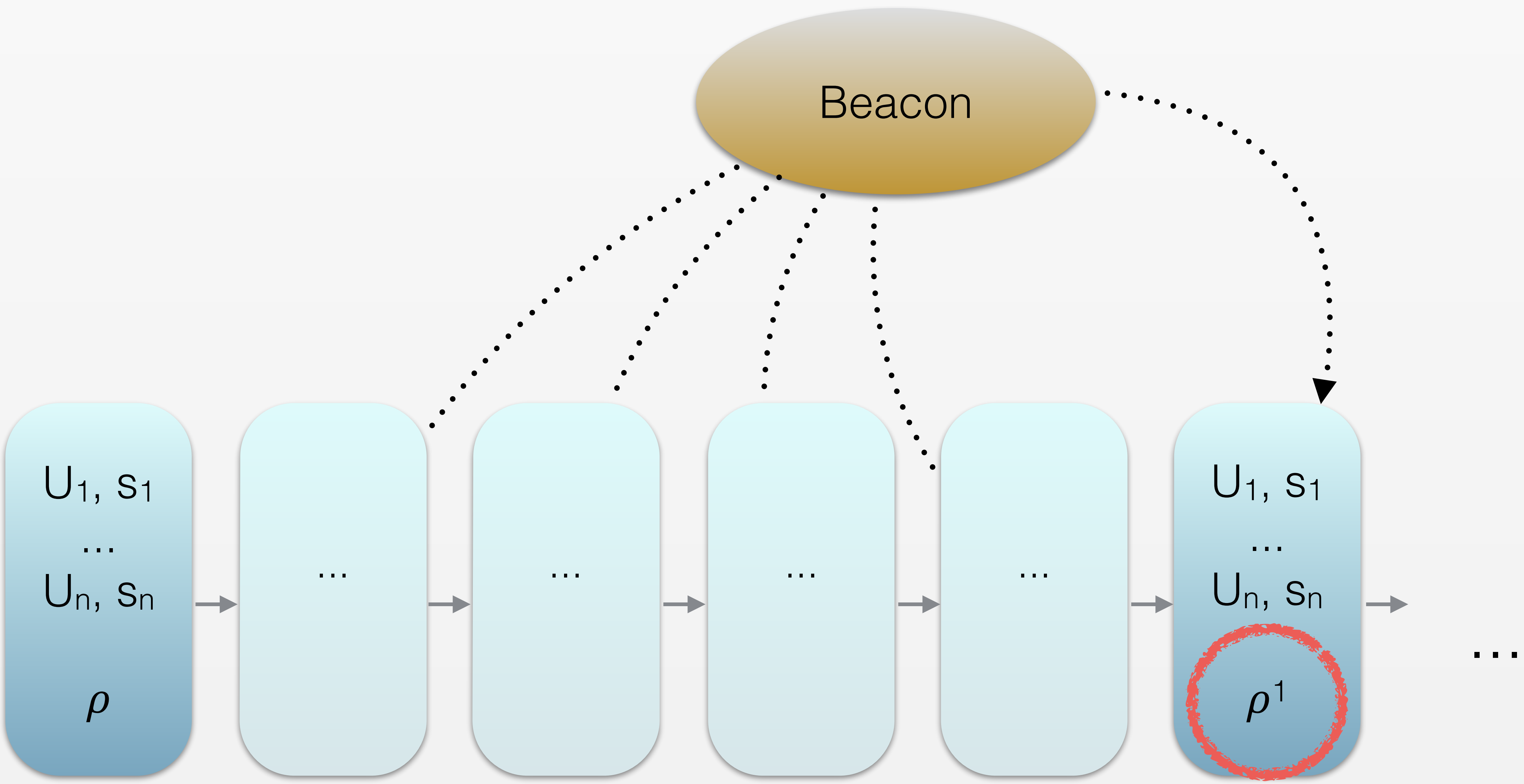
Stage 2

Trusted beacon



Stage 2

Trusted beacon



Stage 3 - replacing the beacon

Have to replace the trusted beacon resource we previously assumed

Will use known cryptographic tools to simulate it

Stage 3

Simulating the beacon

Elected leaders of previous epoch form a committee which executes a multi-party coin tossing protocol to determine the seed

Stage 3

Simulating the beacon

Elected leaders of previous epoch form a committee which executes a multi-party coin tossing protocol to determine the seed

But such protocols may be easily aborted by an adversary

... need to ensure an output!

Stage 3

Simulating the beacon

Solved by using Publicly Verifiable Secret Sharing

"A secret sharing scheme allows to share a secret among several participants such that only certain groups of them can recover it. Verifiable secret sharing has been proposed to achieve security against cheating participants."

Source: Stadler, M., *Publicly Verifiable Secret Sharing*, 1996
https://www.ubilab.org/publications/print_versions/pdf/sta96.pdf

Stage 3

Simulating the beacon

Combining the two:

⇒ a multi-party computation protocol with guaranteed output delivery!

If majority of leaders are honest, this provides the parties with clean randomness

Attacks

A selection of analysed attacks

Attacks

Grinding attacks?

Attacks

Grinding attacks

Not possible

Attacks

Grinding attacks

Not possible
Prevented by coin tossing protocol which is guaranteed to be uniformly random

Attacks

Nothing-at-stake attacks?

Attacks

Nothing-at-stake attacks

Not possible
Forkable strings would enable the nothing-at-stake at attacks (probability negligible)

Attacks

51% attacks?

Attacks

51% attacks

Possible!
Persistence and liveness can be violated

Summary

PoS is an alternative to PoW to improve time and energy consumption

Challenges of PoS:
* leader election process requires randomness taken from the ledger
  ->vulnerable to attacks
* no cost to add blocks

Solved in Ouroboros using:
* a multi-party computation protocol which guarantees outputs
* probability negligible (shown by reducing the problem to forkable strings)

Some attacks against protocols using PoW still possible against protocols using PoS (most notably the **51% attack**)