Proof of Space

Murat Osmanoglu

Proof of Space
(SpaceMint)

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal
  - a few disk accesses with minimal computation

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal
  - a few disk accesses with minimal computation
- even if the reward is much smaller than the cost of buying disk space for mining, space still be dedicated towards mining

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal
  - a few disk accesses with minimal computation
- even if the reward is much smaller than the cost of buying disk space for mining, space still be dedicated towards mining
  - unused disk space is available on many personal computers

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal
  - a few disk accesses with minimal computation
- even if the reward is much smaller than the cost of buying disk space for mining, space still be dedicated towards mining
  - unused disk space is available on many personal computers
  - most mining is currently done by specialized ASICs, and they have no use other than Bitcoin mining. No such devices required for PoSpace

Proof of Space
(SpaceMint)

- to create a block, miners invest disk space instead of computing power
  - dedicating more disk space means higher chance to create a block
- once the dedicated space is initialized, the cost of mining is marginal
  - a few disk accesses with minimal computation
- even if the reward is much smaller than the cost of buying disk space for mining, space still be dedicated towards mining
  - unused disk space is available on many personal computers
  - most mining is currently done by specialized ASICs, and they have no use other than Bitcoin mining. No such devices required for PoSpace
- a simple idea can be applied to Bitcoin to avoid 'mining pools' : instead of applying the hash function to a nonce directly, it will be applied to the signature of the nonce (similar idea can be adapted to SpaceMint)

Proof of Space
(SpaceMint)

Prover

Verifier

Proof of Space
(SpaceMint)

Prover

At the initialization,
prover stores some data
F of size N

Verifier

At the initialization,
verifier keeps only small
piece of information S
about F

Proof of Space
(SpaceMint)

Prover

At the initialization,
prover stores some data
F of size N

Verifier

At the initialization,
verifier keeps only small
piece of information S
about F

At any time later,
verifier initialize a proof
execution phase

Proof of Space
(SpaceMint)

Prover

At the initialization,
prover stores some data
F of size N

Verifier

At the initialization,
verifier keeps only small
piece of information S
about F

At any time later,
verifier initialize a proof
execution phase

At the end,
verifier outputs reject
or accept

Proof of Space
(SpaceMint)

Prover		Verifier
At the initialization, prover stores some data F of size N		At the initialization, verifier keeps only small piece of information S about F
	⟸	
	⟹	At any time later, verifier initialize a proof
	⟸	execution phase
	⟹	
		At the end, verifier outputs reject or accept

- we demand that verifier is highly efficient in both phases, and prover is highly efficient in the execution (generating the proof) and in the access to F

Proof of Space
(SpaceMint)

Prover

At the initialization,
prover stores some data
F of size N

Verifier

At the initialization,
verifier keeps only small
piece of information S
about F

At any time later,
verifier initialize a proof
execution phase

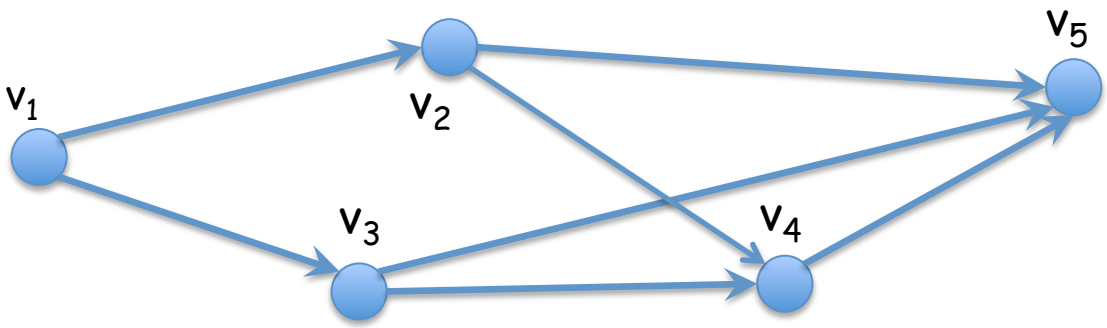
At the end,
verifier outputs reject
or accept

- we demand that verifier is highly efficient in both phases, and prover is highly efficient in the execution (generating the proof) and in the access to F
- be careful; a cheating prover can delete F after initialization

Proof of Space
(SpaceMint)

Graph Pebbling

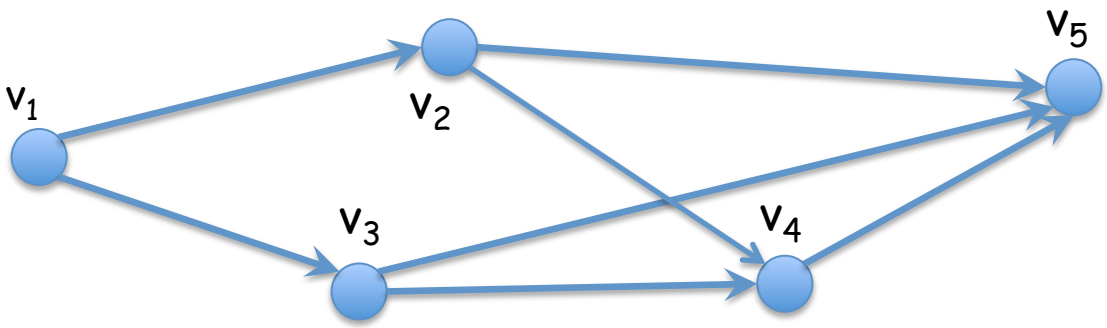
- consider a directed acyclic graph $G = (V\ E)$



Proof of Space
(SpaceMint)

Graph Pebbling

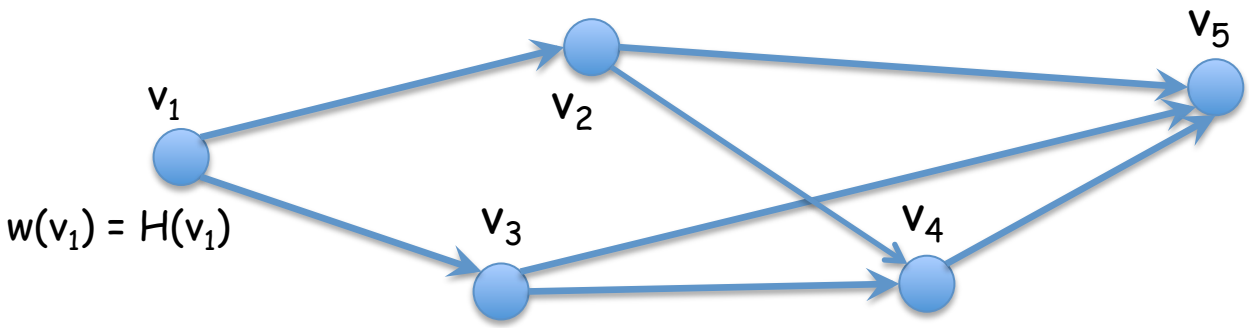
- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



Proof of Space
(SpaceMint)

Graph Pebbling

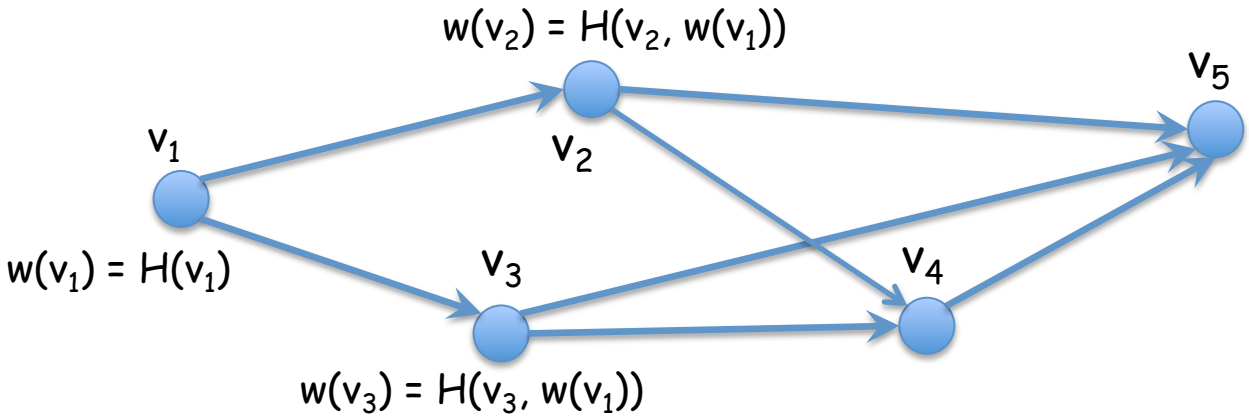
- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



Proof of Space
(SpaceMint)

Graph Pebbling

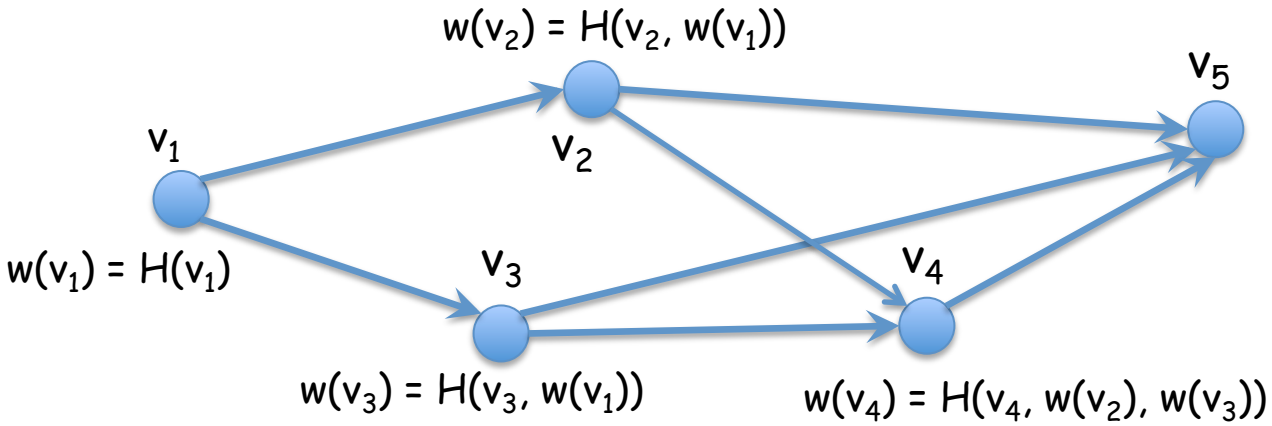
- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



Proof of Space
(SpaceMint)

Graph Pebbling

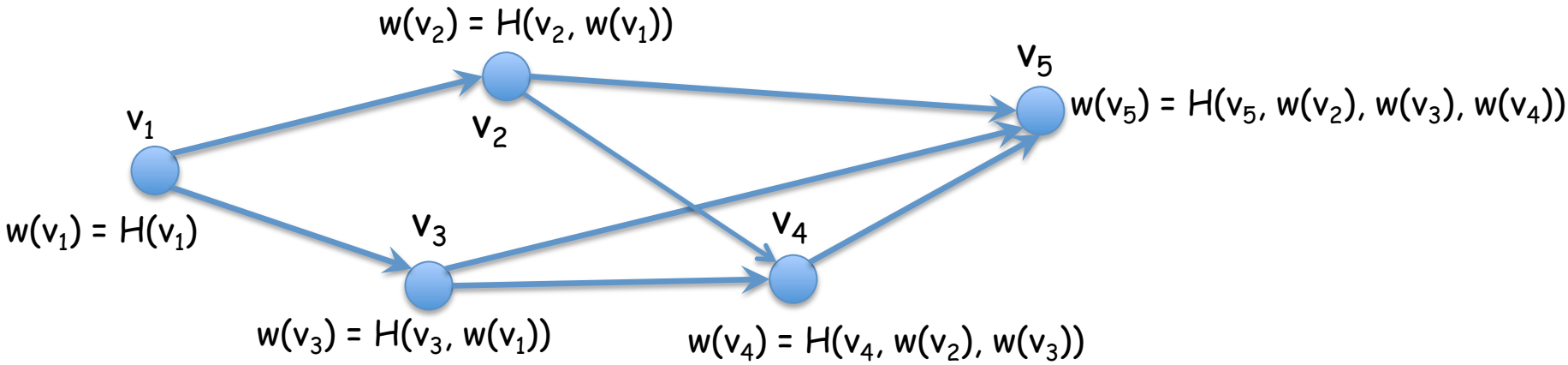
- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



Proof of Space
(SpaceMint)

Graph Pebbling

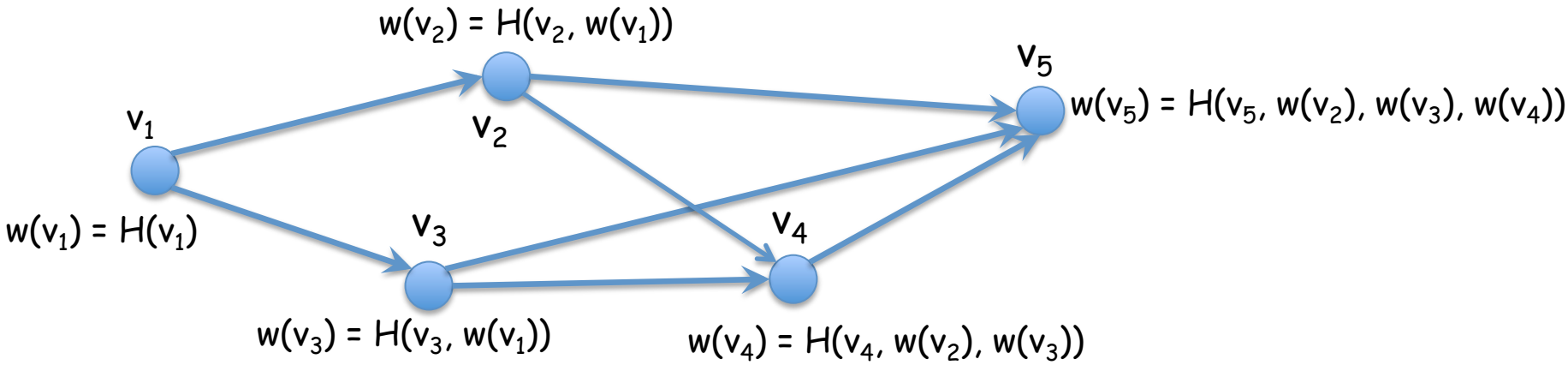
- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



Proof of Space
(SpaceMint)

Graph Pebbling

- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$

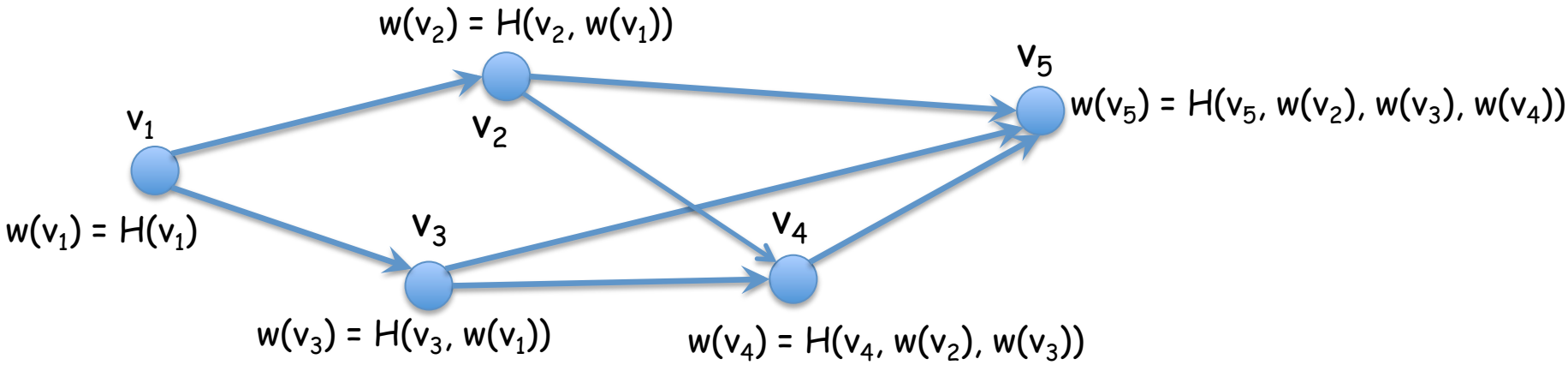


Parameter : $G = (V, E)$ s.t. $|V|=N$, D is efficiently samplable distribution over vertices

Proof of Space
(SpaceMint)

Graph Pebbling

- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



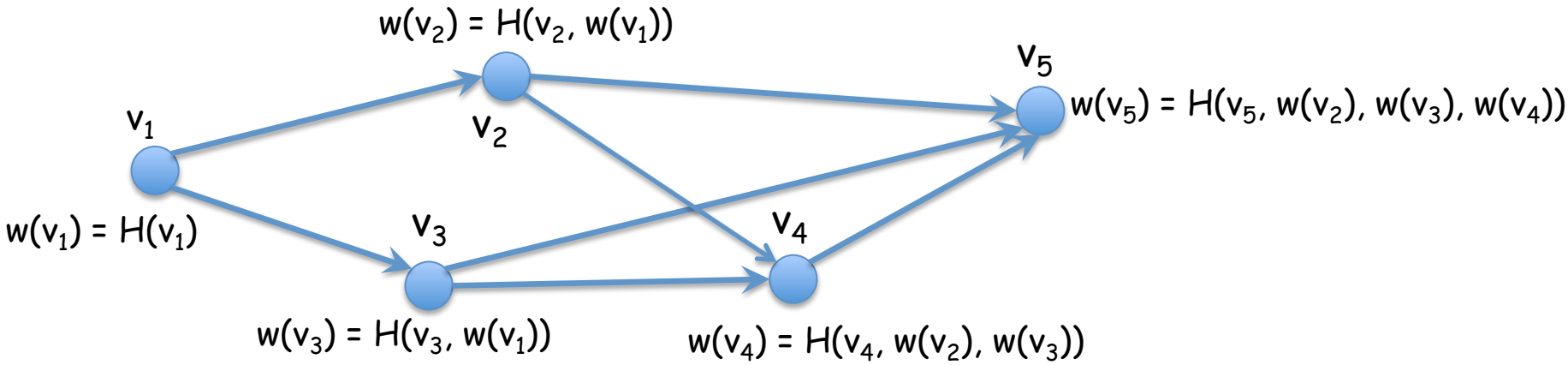
Parameter : $G = (V, E)$ s.t. $|V|=N$, D is efficiently samplable distribution over vertices

- First prover creates $S=w(V)$ and a short proof λ from S, then keeps the graph and gives λ to verifier

Proof of Space
(SpaceMint)

Graph Pebbling

- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



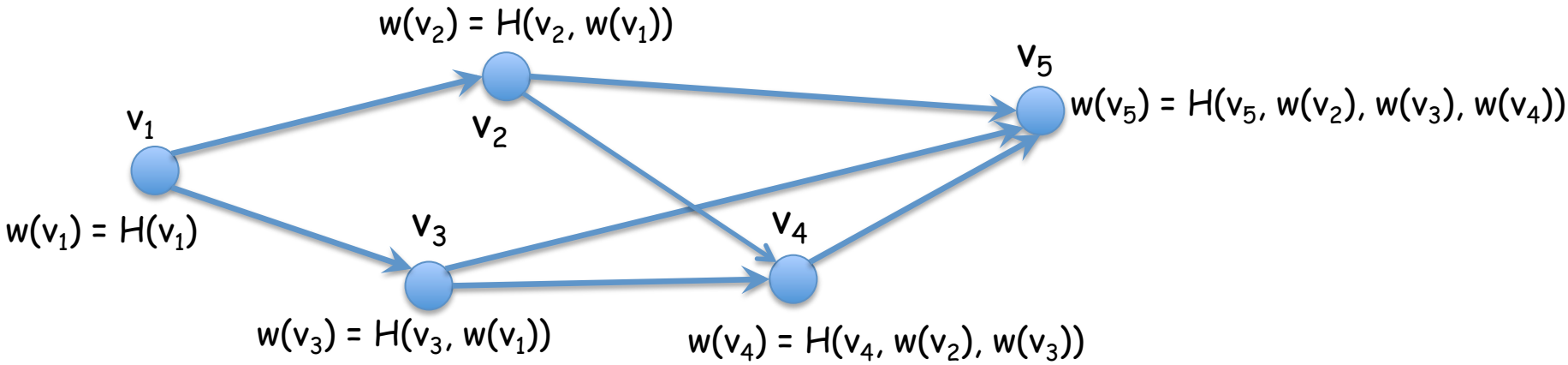
Parameter : $G = (V, E)$ s.t. $|V|=N$, D is efficiently samplable distribution over vertices

- First prover creates $S=w(V)$ and a short proof λ from S, then keeps the graph and gives λ to verifier
- Verifier samples a subset $C \leftarrow D$, and gives C to P

Proof of Space
(SpaceMint)

Graph Pebbling

- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$



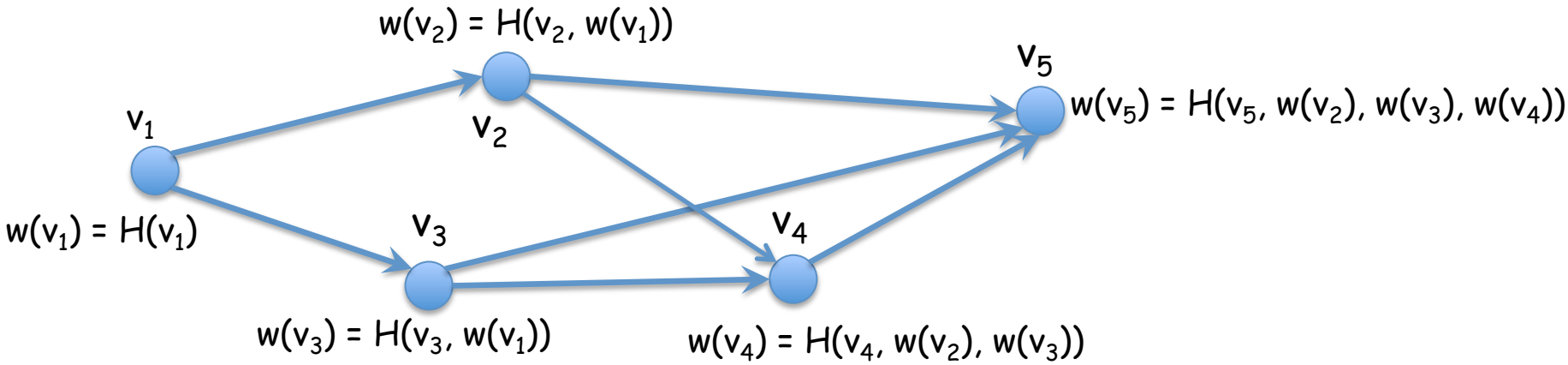
Parameter : $G = (V, E)$ s.t. $|V|=N$, D is efficiently samplable distribution over vertices

- First prover creates $S=w(V)$ and a short proof λ from S, then keeps the graph and gives λ to verifier
- Verifier samples a subset $C \leftarrow D$, and gives C to P
- Prover creates an answer $A=w(C)$ for C

Proof of Space
(SpaceMint)

Graph Pebbling

- consider a directed acyclic graph $G = (V\ E)$
- every vertex is associated with a value $w(v)$ in $\{0,1\}^L$

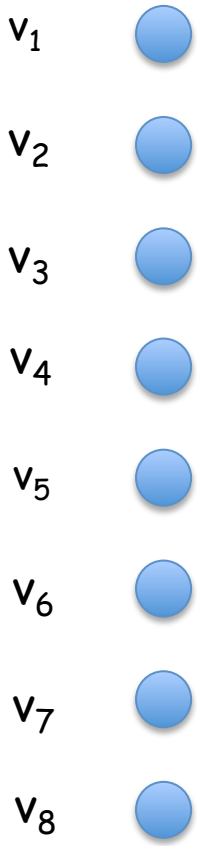


Parameter : $G = (V, E)$ s.t. $|V|=N$, D is efficiently samplable distribution over vertices

- First prover creates $S=w(V)$ and a short proof λ from S, then keeps the graph and gives λ to verifier
- Verifier samples a subset $C \leftarrow D$, and gives C to P
- Prover creates an answer $A=w(C)$ for C
- Verifier accepts if A is compatible with λ

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling

v_1 x_1

v_2 x_2

v_3 x_3

v_4 x_4

v_5 x_5

v_6 x_6

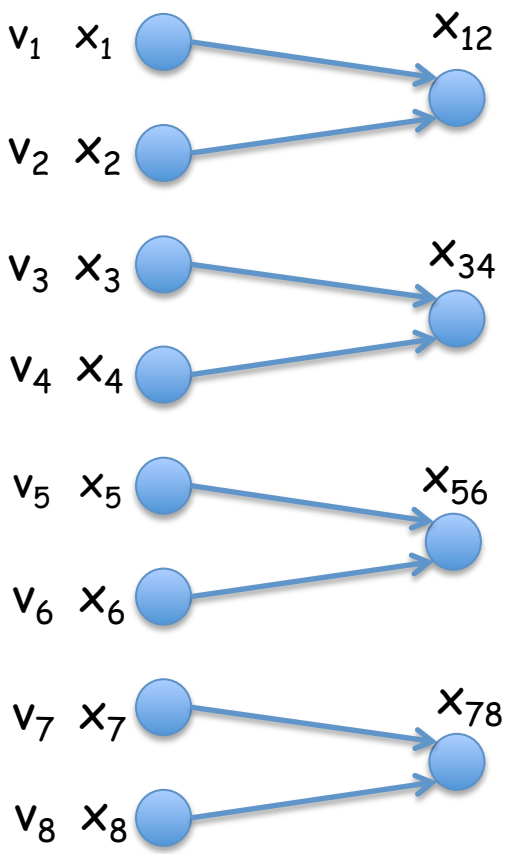
v_7 x_7

v_8 x_8

$x_a = H(v_a)$

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling

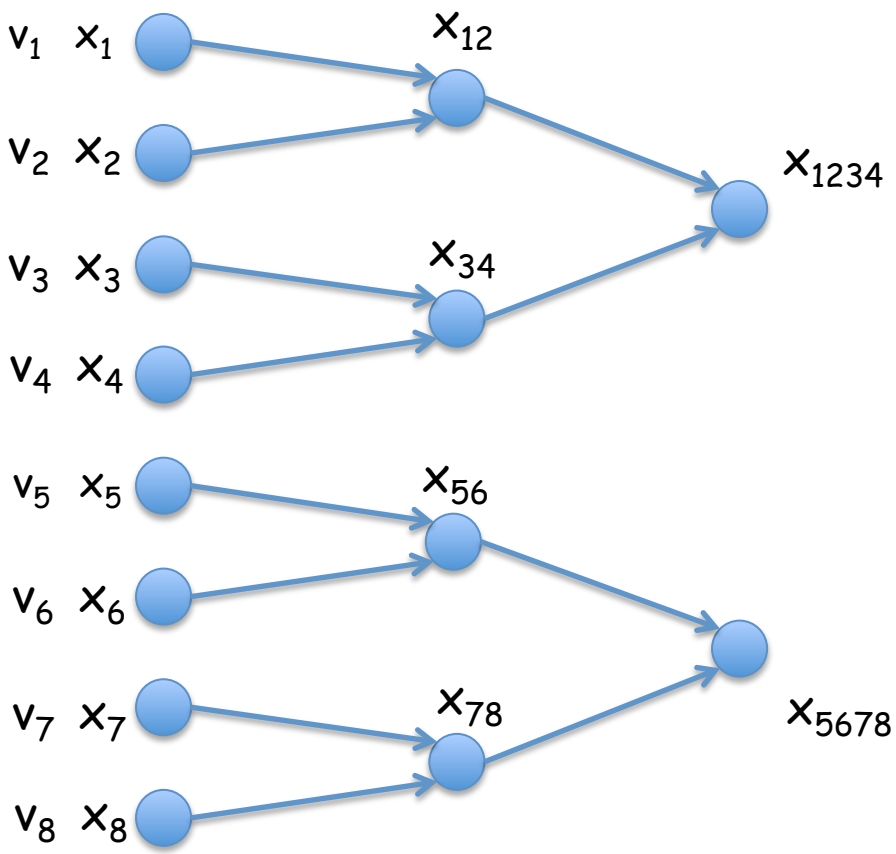


$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



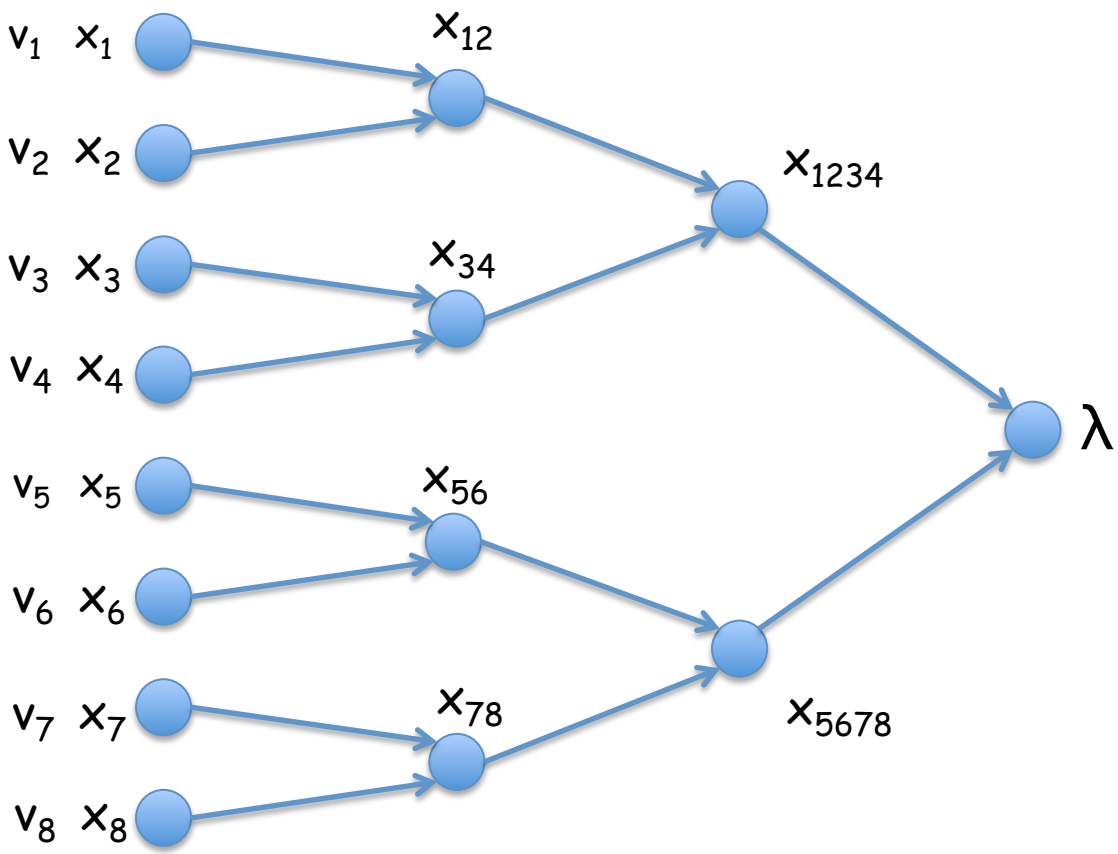
$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

$x_{abcd} = H(x_{ab}, x_{cd})$

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

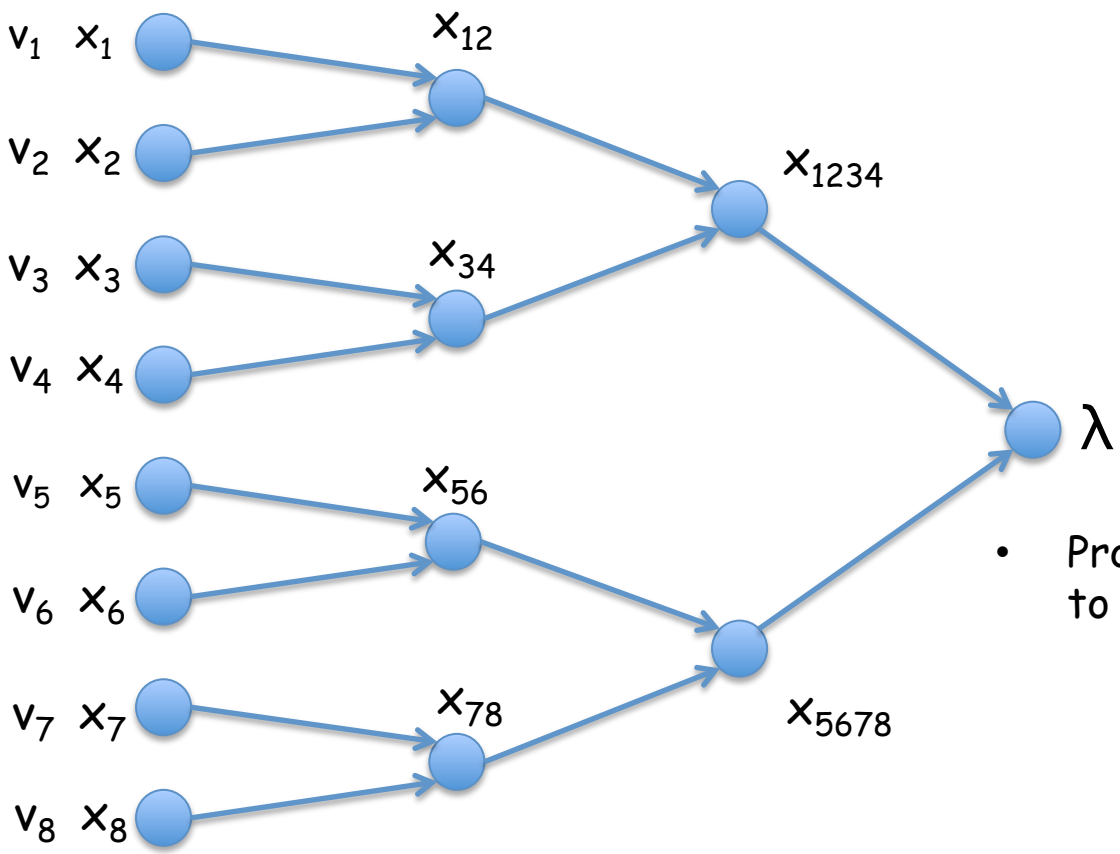
$x_{ab} = H(x_a, x_b)$

$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

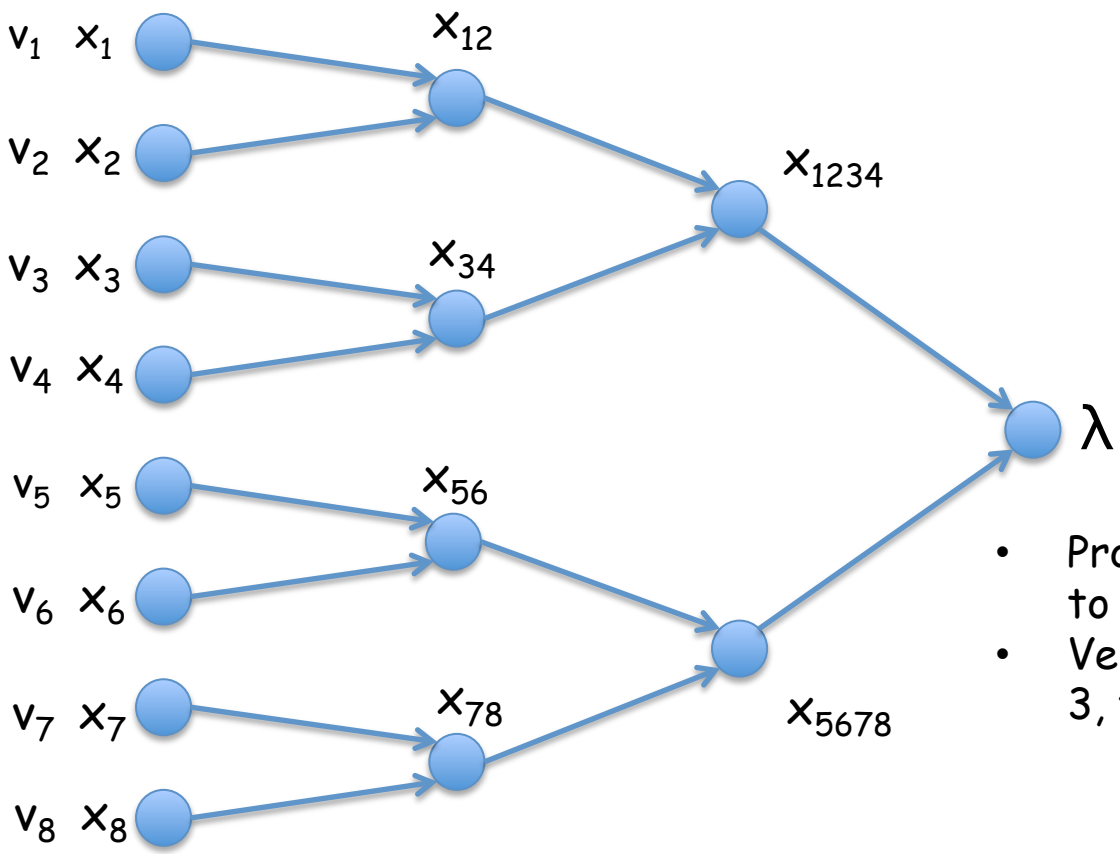
$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

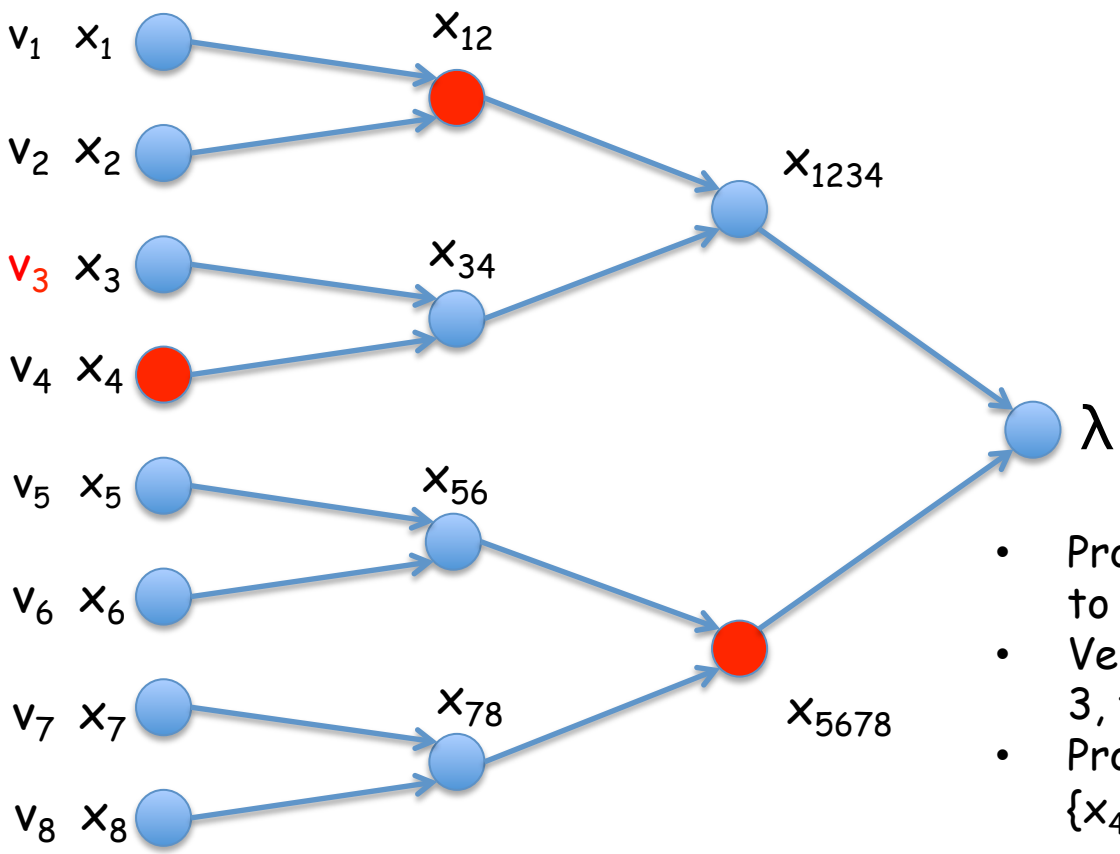
$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier
- Verifier sends a challenge, for instance 3, to prover

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

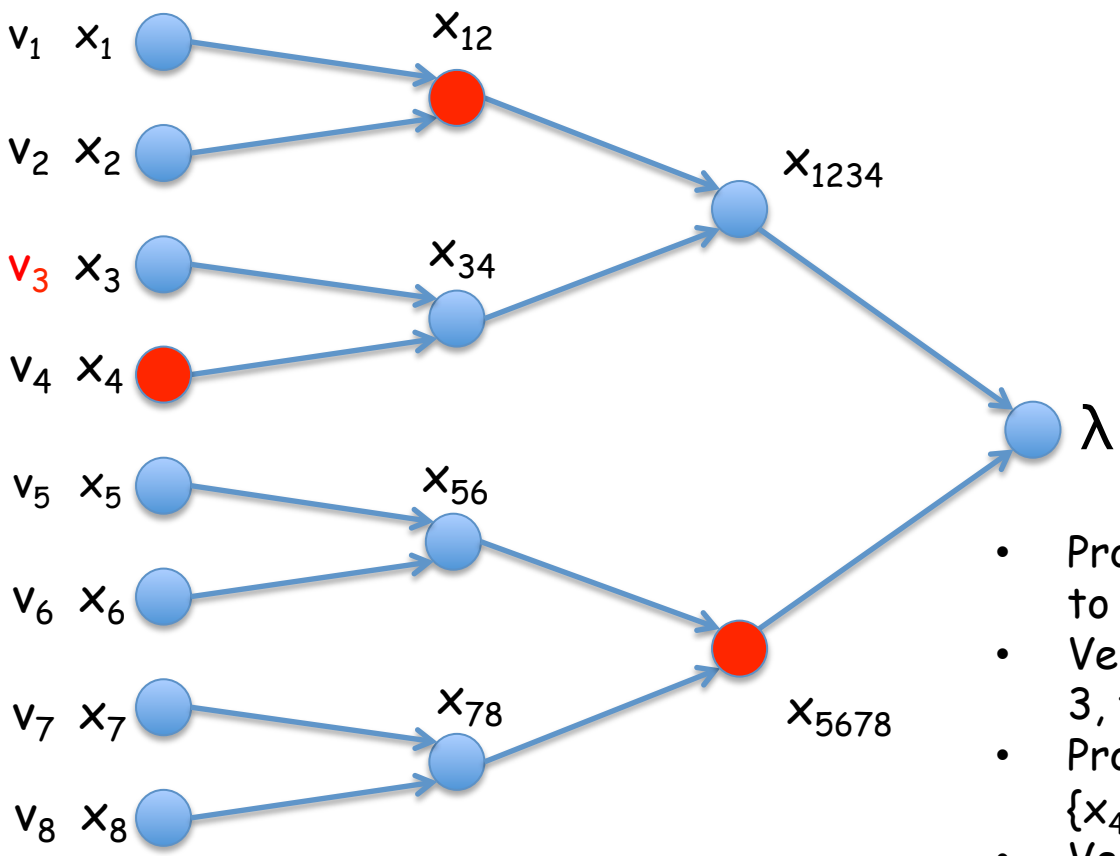
$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier
- Verifier sends a challenge, for instance 3, to prover
- Prover sends v_3 together with $\{x_4, x_{12}, x_{5678}\}$ to verifier

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

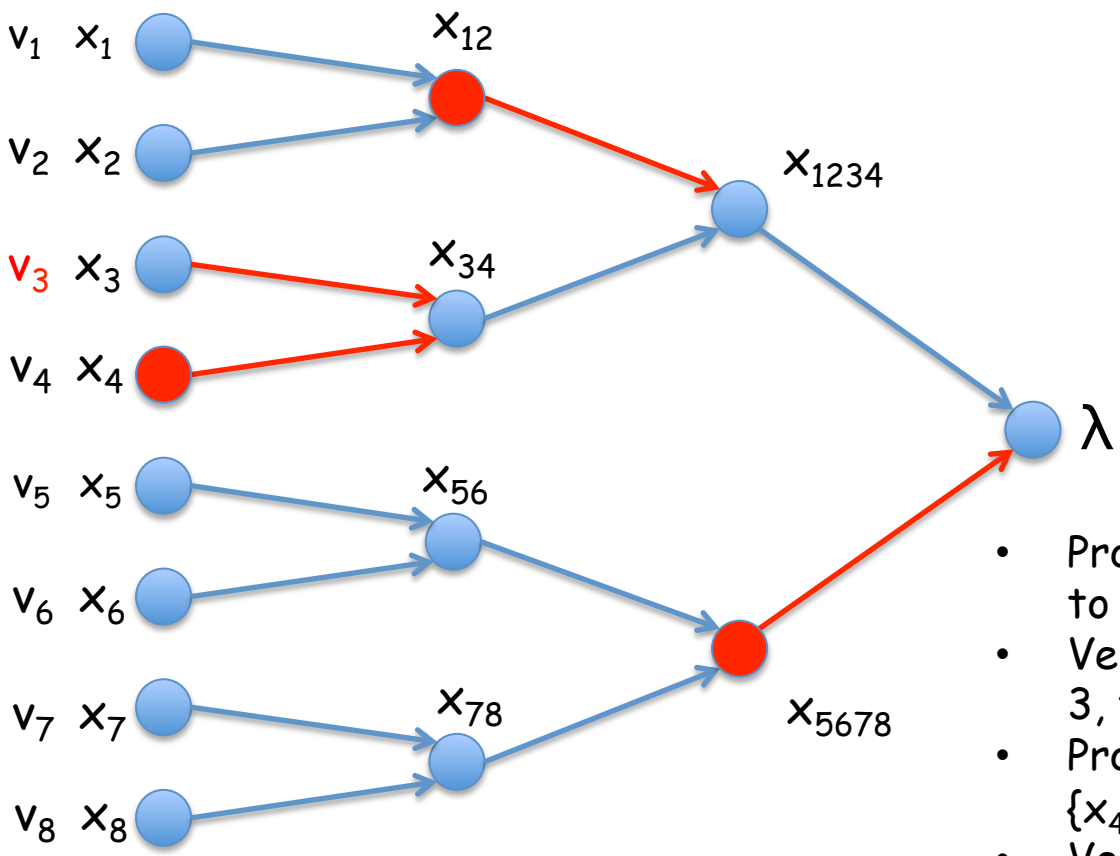
$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier
- Verifier sends a challenge, for instance 3, to prover
- Prover sends v_3 together with $\{x_4, x_{12}, x_{5678}\}$ to verifier
- Verifier accepts if

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

$x_{abcd} = H(x_{ab}, x_{cd})$

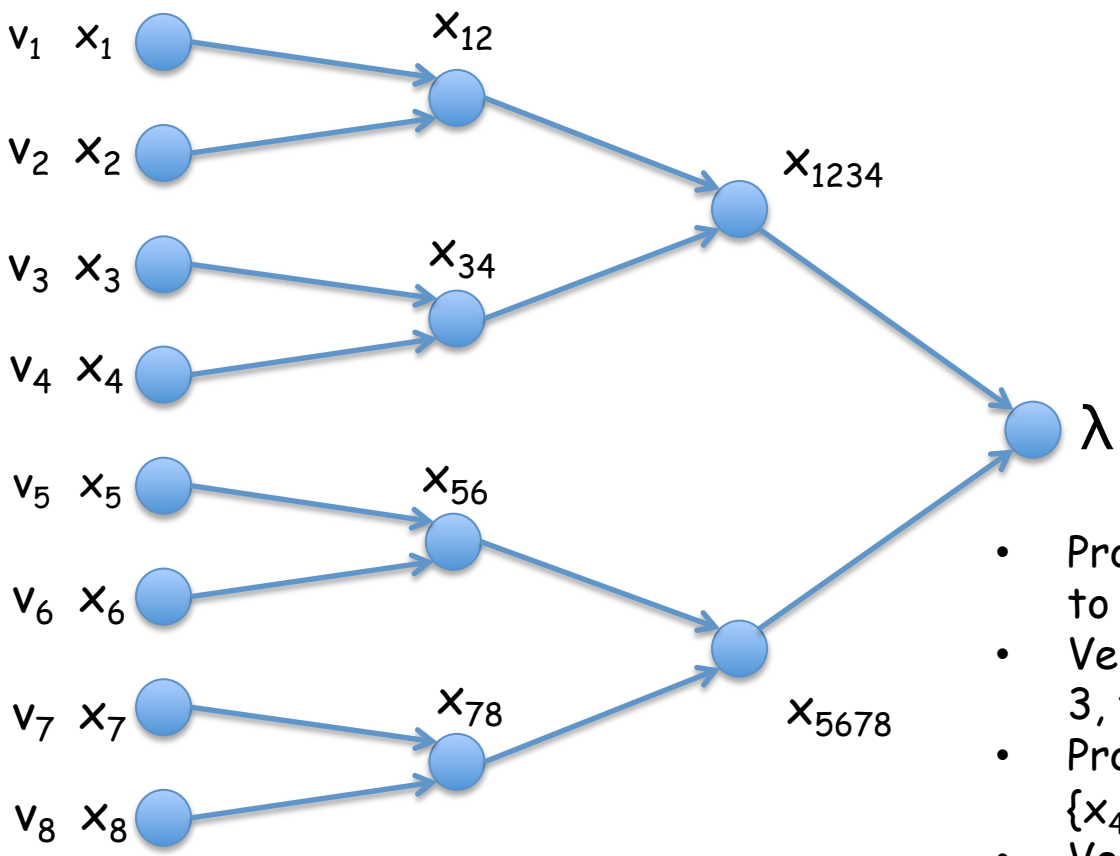
$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier
- Verifier sends a challenge, for instance 3, to prover
- Prover sends v_3 together with $\{x_4, x_{12}, x_{5678}\}$ to verifier
- Verifier accepts if

$$\lambda = H(H(x_{12}, H(H(v_3), x_4)), x_{5678})$$

Proof of Space
(SpaceMint)

Hash Trees for Graph Pebbling



$x_a = H(v_a)$

$x_{ab} = H(x_a, x_b)$

$x_{abcd} = H(x_{ab}, x_{cd})$

$\lambda = H(x_{1234}, x_{5678})$

- Prover creates hash tree and sends λ to verifier
- Verifier sends a challenge, for instance 3, to prover
- Prover sends v_3 together with $\{x_4, x_{12}, x_{5678}\}$ to verifier
- Verifier accepts if

$$\lambda = H(H(x_{12}, H(H(v_3), x_4)), x_{5678})$$

If H is collusion-resistant, then prover must keep all the data to be able to provide valid proof

Proof of Space
(SpaceMint)

Challenges

Proof of Space
(SpaceMint)

Challenges

- Proof of Space requires interactions, it's hard to adapt it to the blockchain settings

Proof of Space
(SpaceMint)

Challenges

- Proof of Space requires interactions, it's hard to adapt it to the blockchain settings

- Generating a PoSpace is computationally cheap. It needs some clever way to decide which of many proof wins

Proof of Space
(SpaceMint)

Challenges

- Proof of Space requires interactions, it's hard to adapt it to the blockchain settings

- Generating a PoSpace is computationally cheap. It needs some clever way to decide which of many proof wins

  a miner should learn if he is a winner without any interaction

Proof of Space
(SpaceMint)

Challenges

- Proof of Space requires interactions, it's hard to adapt it to the blockchain settings

- Generating a PoSpace is computationally cheap. It needs some clever way to decide which of many proof wins

  a miner should learn if he is a winner without any interaction

- nothing-at-stake : if mining is computationally cheap, then miners can mine on multiple chains, or try to create many different blocks with a single proof

Proof of Space
(SpaceMint)

Challenges

- Proof of Space requires interactions, it's hard to adapt it to the blockchain settings

- Generating a PoSpace is computationally cheap. It needs some clever way to decide which of many proof wins

  a miner should learn if he is a winner without any interaction

- nothing-at-stake : if mining is computationally cheap, then miners can mine on multiple chains, or try to create many different blocks with a single proof

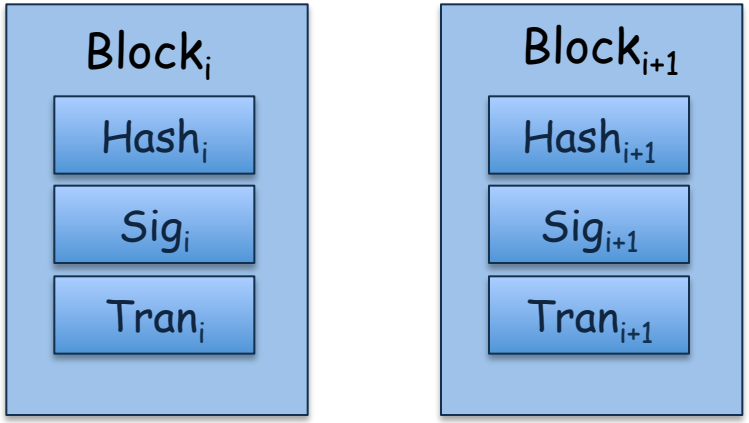
  slows down consensus

  gives chance to cheating miners to get a greater reward

  enables double-spending attacks by someone controlling less than 50% of the space

Proof of Space
(SpaceMint)

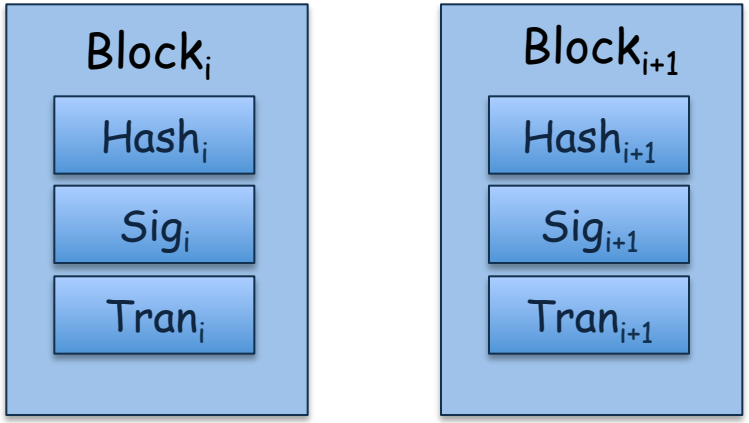
Construction



Proof of Space
(SpaceMint)

Construction

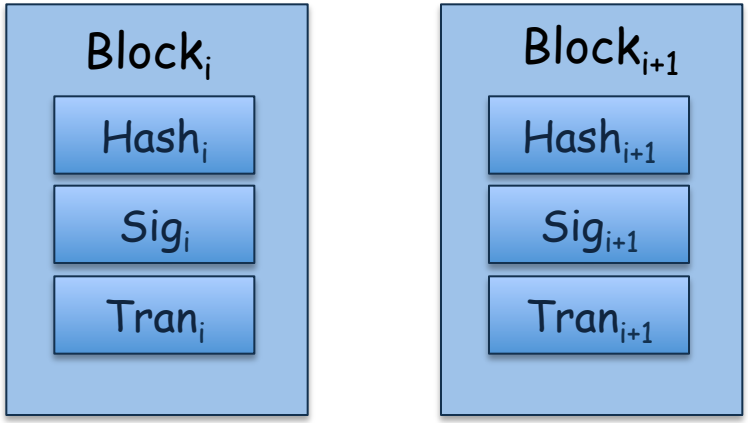
- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.



Proof of Space
(SpaceMint)

Construction

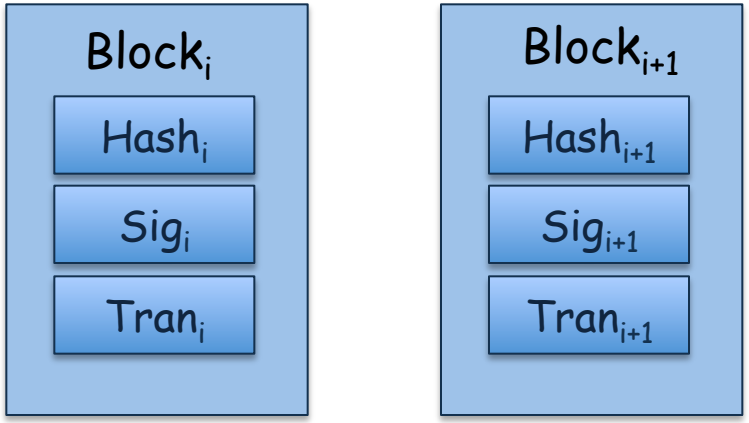
- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk),



Proof of Space
(SpaceMint)

Construction

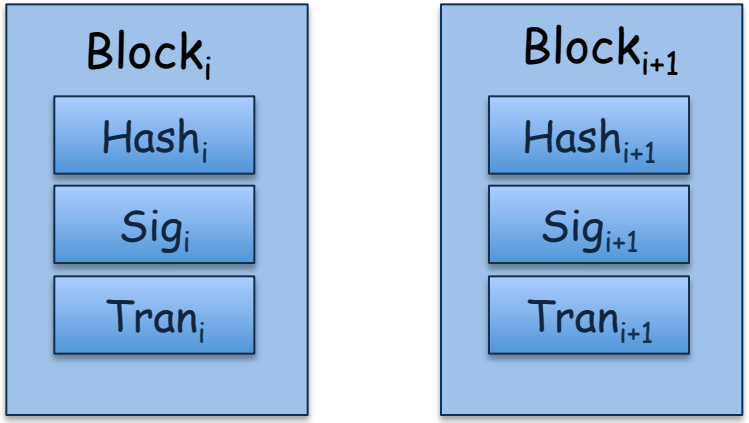
- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort

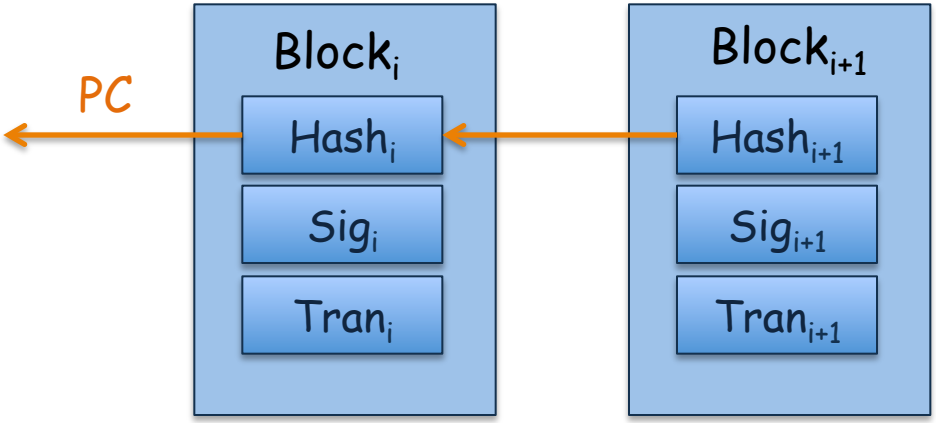


$Hash_i$: current block index i

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort

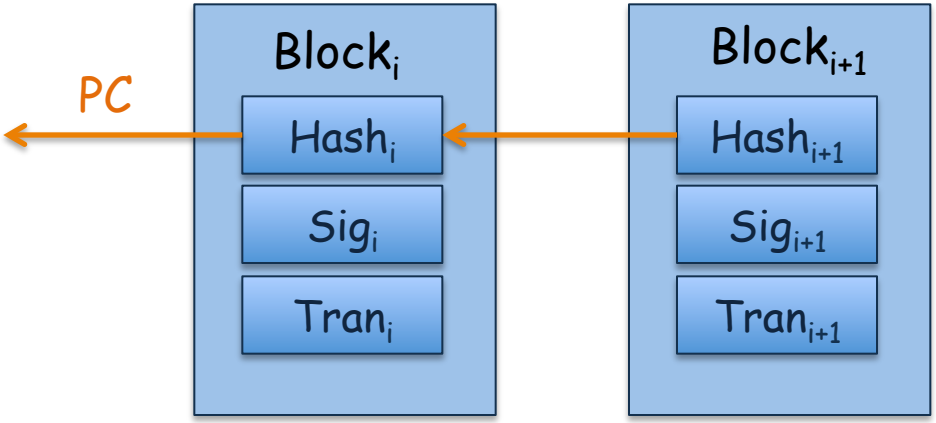


Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort

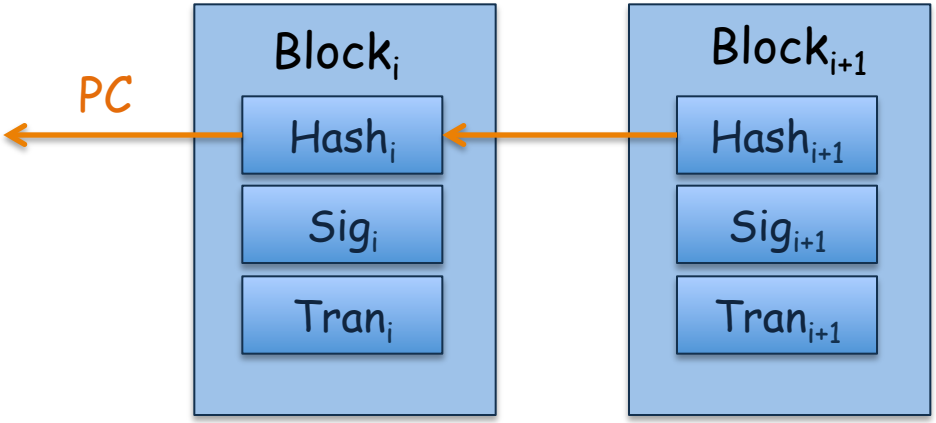


Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



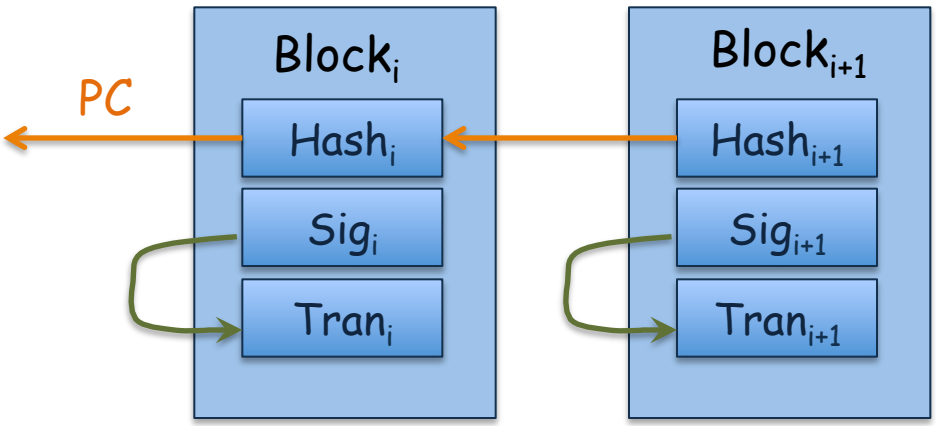
Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

Sig$_i$: current block index i

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



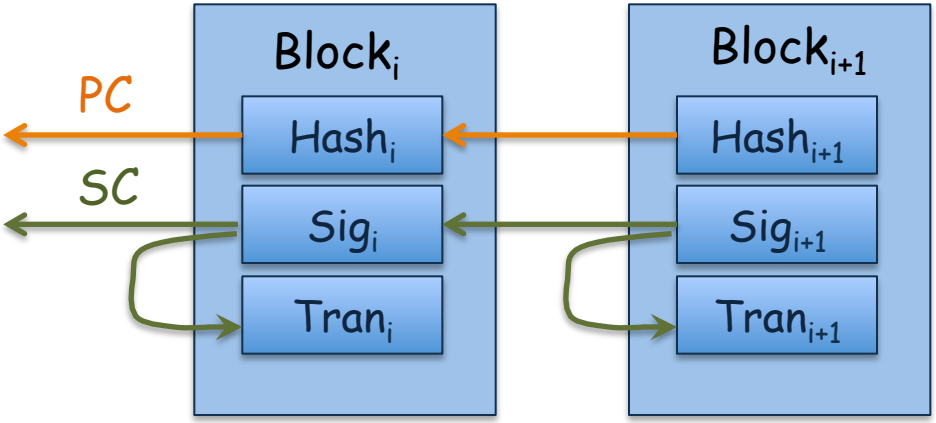
Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

Sig$_i$: current block index i
miner's signature on Tran$_i$

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



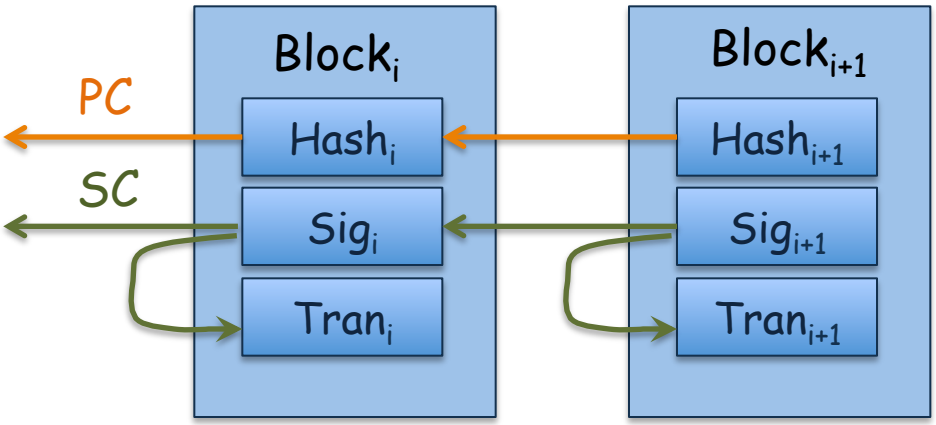
Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

Sig$_i$: current block index i
miner's signature on Tran$_i$
miner's signature on Sig$_{i-1}$

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

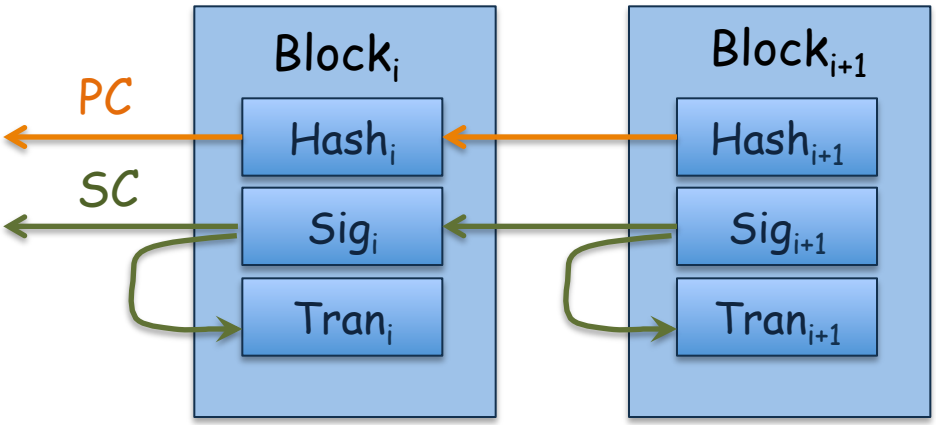
Sig$_i$: current block index i
miner's signature on Tran$_i$
miner's signature on Sig$_{i-1}$

Tran$_i$: current block index i
list of transactions

Proof of Space
(SpaceMint)

Construction

- a miner joins the network by announcing its space commitment (pk, λ) via a special transaction.
  Gen➔(pk,sk), Init(pk,N)➔(S,λ) where N is the size of the space in terms of bits that the miner contributes to the mining effort



Hash$_i$: current block index i
miner's signature on Hash$_{i-1}$
a space proof that contains pk

Sig$_i$: current block index i
miner's signature on Tran$_i$
miner's signature on Sig$_{i-1}$

Tran$_i$: current block index i
list of transactions

- Once an honest miner adds a new block to the chain, the transactions up to this block cannot be changed, even by someone that holds all secret keys of the miners that added all the previous blocks.

Proof of Space
(SpaceMint)

Construction

- three types of transactions :

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where
    $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where
    $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that
  - space commitments : has the form tx = (commit, txID, (pk, λ)) where

    $\mathrm{Init}(pk,N) \rightarrow (pk, \lambda)$

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that
  - space commitments : has the form tx = (commit, txID, (pk, λ)) where

    $\mathrm{Init}(pk,N) \rightarrow (pk, \lambda)$
  - penalties : has the form tx = (penalty, txID, pk, prf)

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that
  - space commitments : has the form tx = (commit, txID, (pk, λ)) where

    $\mathrm{Init}(pk,N) \rightarrow (pk, \lambda)$
  - penalties : has the form tx = (penalty, txID, pk, prf)

    pk is the public key of the transaction creator

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that
  - space commitments : has the form tx = (commit, txID, (pk, λ)) where

    $\mathrm{Init}(pk,N) \rightarrow (pk, \lambda)$
  - penalties : has the form tx = (penalty, txID, pk, prf)

    pk is the public key of the transaction creator

    prf is the proof indicating that two blocks of same index signed by the same signer

Proof of Space
(SpaceMint)

Construction

- three types of transactions :
  - regular payments : has the form tx = (payment, txID, in, out) where

    txID is a unique ID (no two tx in the blockchain can have same ID

    in is a list of input coins, in = $(in_1,\ldots,in_n)$ where $in_j = (txID_j, k_j, sig_j)$ such that k_j is an index that indicates the sender of $txID_j$

    out is a list of output coins, out = $(out_1,\ldots,out_m)$ where $out_i=(pk_i,v_i)$ such that
  - space commitments : has the form tx = (commit, txID, (pk, λ)) where

    $\text{Init}(pk,N) \rightarrow (pk, \lambda)$
  - penalties : has the form tx = (penalty, txID, pk, prf)

    pk is the public key of the transaction creator

    prf is the proof indicating that two blocks of same index signed by the same signer
- For regular payments; all signatures must be valid, any subsequent transaction must be used only one time in the blockchain(double-spendin), and the sum of the input values should be at least the sum of the output for the acceptance of tx

Proof of Space
(SpaceMint)

Construction

- Mining :

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v
  - compute challenges $Chal(n,u,C_u) = (c_1,\ldots,c_u)$

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v
  - compute challenges $Chal(n,u,C_u) = (c_1,\ldots,c_u)$
  - compute the proof of space $a = (a_1,\ldots,a_u)$

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v
  - compute challenges $Chal(n,u,C_u) = (c_1,\ldots,c_u)$
  - compute the proof of space $a = (a_1,\ldots,a_u)$
  - compute the quality $Q(pk,\lambda,c,a)$ of the proof

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v
  - compute challenges $Chal(n,u,C_u) = (c_1,\ldots,c_u)$
  - compute the proof of space $a = (a_1,\ldots,a_u)$
  - compute the quality $Q(pk,\lambda,c,a)$ of the proof
  - if the quality is high enough (there is a realistic chance to be the best proof in that period), compute the proof of correct commitment $b = (b_1, \ldots,b_v)$, create a block, send the block to the network

Proof of Space
(SpaceMint)

Construction

- Mining :
  - extract the hash value of the last block in the best chain so far, and a challenge c which is used to derive two long random strings C_u, C_v
  - [illegible]
  - [illegible]
  - [illegible] he best
  - [illegible] $b = (b_1,$

How do we create this challenge ?

(i-1)th block can be used to derive that challenge, but it can slow down the consensus.

- there may be many chains; rational miners can create different challenges for different chains, and try to create proofs for different chains since it is easy to do it

Derive the challenge from the hash of block i – Δ

- the probability of multiple chains surviving for more than Δ blocks decreases exponentially

Proof of Space
(SpaceMint)

Construction

- Mir
  - and a
  - compute the proof of space $a = (a_1,\ldots,a_u)$
  - compute the quality $Q(pk,\lambda,c,a)$ of the proof
  - if the quality is high enough (there is a realistic chance to be the best proof in that period), compute the proof of correct commitment $b = (b_1, \ldots,b_v)$, create a block, send the block to the network

For a set of valid proofs $\pi_1=(pk_1,\lambda_1,c_1, a_1),\ldots, \pi_m=(pk_m,\lambda_m,c_m, a_m)$, $Q(\pi_i)$ should be defined in a way that the probability that π_i has the best quality among $\pi_1,\ldots, \pi_m$ corresponds to ith miner's fraction of the total space in the network, which is

$$N_i / (N_1 + \ldots + N_m)$$

where N_i is the space committed to λ_i

Proof of Space
(SpaceMint)

Nothing-at-Stake

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on many different blocks (just try different transactions) till finding a good one that will allow them to generate good proofs for future

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on many different blocks (just try different transactions) till finding a good one that will allow them to generate good proofs for future
  - the challenge is derived from proof chain that does not contain transactions. Thus, a prover can create at most one valid proof per challenge

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on many different blocks (just try different transactions) till finding a good one that will allow them to generate good proofs for future
  - the challenge is derived from proof chain that does not contain transactions. Thus, a prover can create at most one valid proof per challenge
- Mining on mutiple chains:

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on many different blocks (just try different transactions) till finding a good one that will allow them to generate good proofs for future
  - the challenge is derived from proof chain that does not contain transactions. Thus, a prover can create at most one valid proof per challenge
- Mining on mutiple chains:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on all known chains in parallel to increase their profits, even to try double-spending and selfish-mining.

Proof of Space
(SpaceMint)

Nothing-at-Stake

- Grinding:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on many different blocks (just try different transactions) till finding a good one that will allow them to generate good proofs for future
  - the challenge is derived from proof chain that does not contain transactions. Thus, a prover can create at most one valid proof per challenge
- Mining on mutiple chains:
  - In PoSpace, it's computationally easy to generate proofs. So, miners can work on all known chains in parallel to increase their profits, even to try double-spending and selfish-mining.
  - the challenge is derived the hash of block i – Δ, and for any challenge there is a single proof. Besides, the protocol imposes a penalty via the penalty transactions (half of the reward for bad block is given to the creator of the penalty transaction, and other half is diminished)