Set Accumulators and their Applications to Blockchain Systems

Matteo Loporchio
University of Pisa
matteo.loporchio@di.unipi.it

IEEE SM
ICBC

Set accumulators

Cryptographic primitives that:

1) represent a large set of elements S with a single and constant-size value (*accumulator value*);

2) can be used to efficiently generate a proof of membership (*witness*) for an element in S.

Set accumulators

S = Set of elements

Set accumulator value A(S)

Check proof

Is  in S?

Yes + proof

Prover

Verifier

Set accumulators

The **prover** knows the entire content of the set (i.e., it stores the values explicitly).

The **verifier** knows only the accumulator value (sufficient to verify proofs).

Set accumulators

Classification

- **Static**: do not support element insertion and deletion.
- **Dynamic**: also support element insertion and deletion.

- **Asymmetric**: requires proof generation before verification.
- **Symmetric**: no proof needed, the accumulated value on its own is sufficient for verification.

- **Universal** accumulators also support **non-membership** proofs (i.e., prove that an element is not in the accumulated set).

Set accumulators

Batch membership proofs: demonstrate to a verifier that all elements of a set S are included in X (i.e., S is a subset of X).

Batch non-membership proofs: demonstrate to a verifier that all elements of a set S are not included in X (i.e., S is not a subset of X).

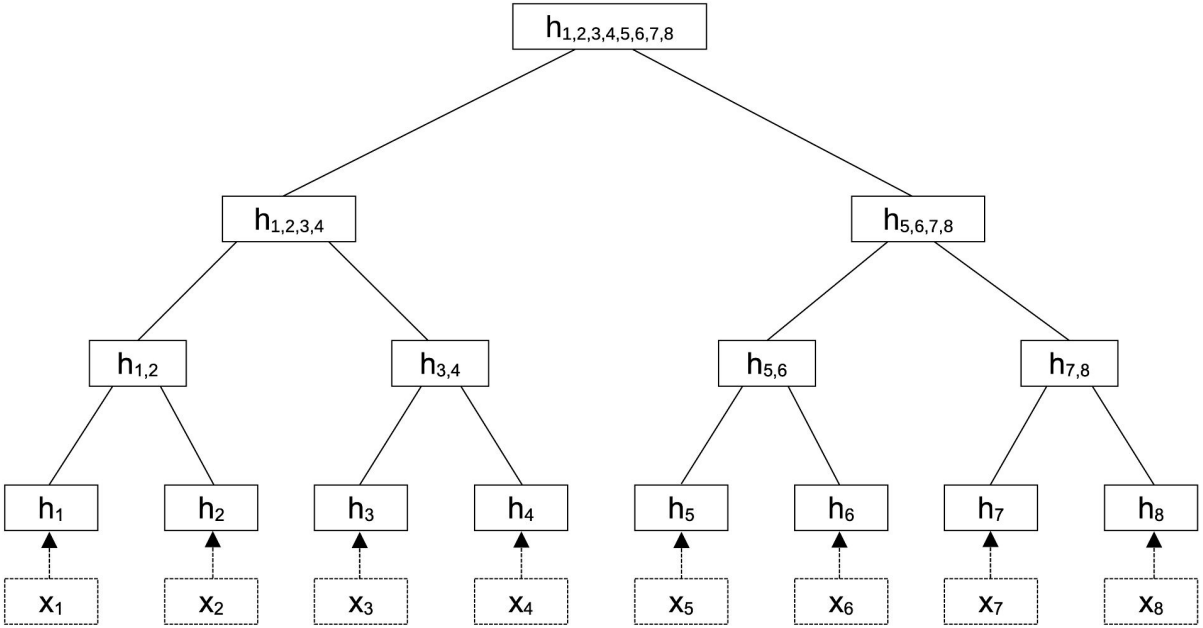
Set accumulators

- Merkle Tree
- RSA accumulator [BP97]
- Bilinear accumulator [Ngu05]
- Expressive accumulator [ZKP17]

Sources:
[BP97] N. Barić and B. Pfitzmann, “Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees,” Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 480–494, 1997.
[Ngu05] L. Nguyen, “Accumulators from Bilinear Pairings and Applications,” Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 275–292, 2005.
[ZKP17] Y. Zhang, J. Katz, and C. Papamanthou, “An Expressive (Zero-Knowledge) Set Accumulator,” 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, pp. 158–173, Apr. 2017.

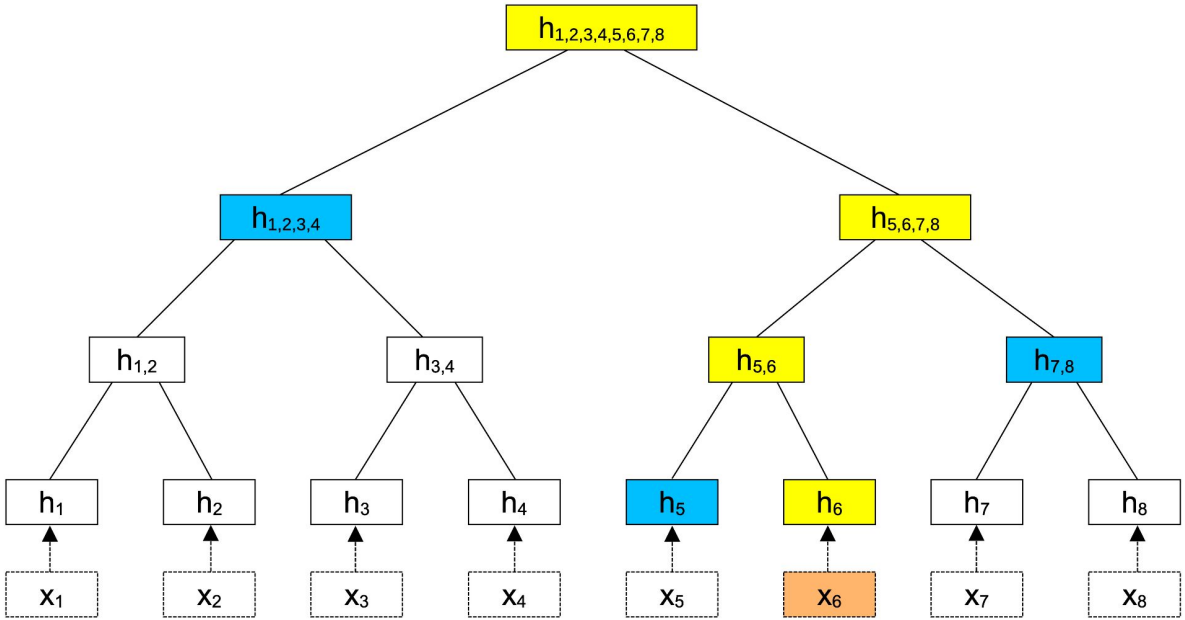
Merkle tree

Verify the integrity of a set of elements (e.g., TXs in a Bitcoin block).



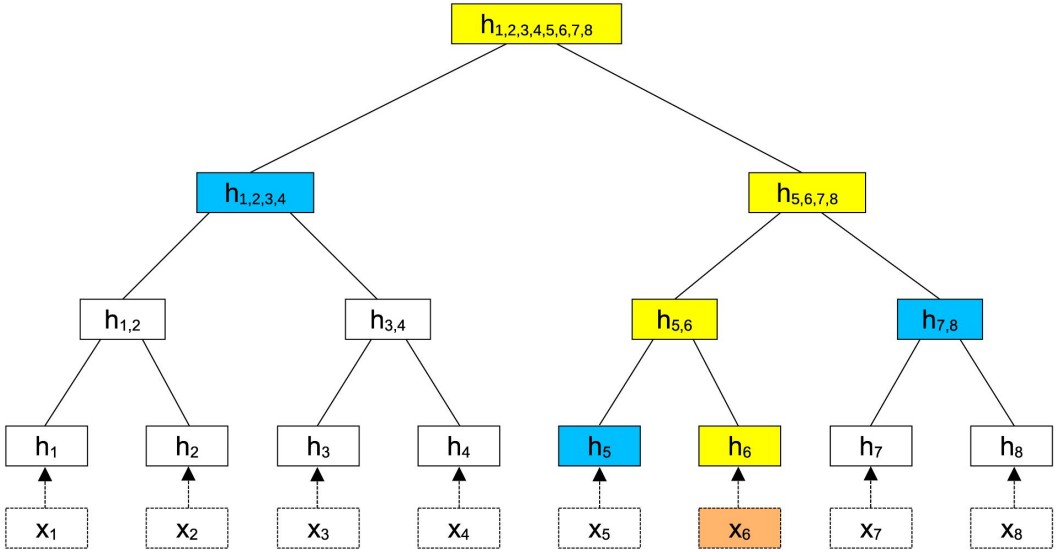
Merkle tree

Membership proof: an element x is included in a set S.



Proof

Merkle tree



1) Root hash represents the accumulator value.

2) Proof of membership is represented by nodes in the co-path from leaf to root.

Merkle tree

Security relies on the use of cryptographic hash functions (collision-resistance).

=> Adversaries cannot forge “fake” proofs (convincing a verifier that an element is in the set while in fact it is not).

RSA accumulator

Accumulates a set of **prime** numbers $X = \{x_1, \ldots, x_n\}$.

Let $N = p \cdot q$ be an RSA modulus, with p and q (large) prime numbers.

RSA accumulator

Accumulates a set of **prime** numbers $X = \{x_1, \ldots, x_n\}$.

Let $N = p \cdot q$ be an RSA modulus, with p and q (large) prime numbers.

The **accumulator value** of X is

$$A(X) = g^{(x_1 \cdot \ldots \cdot x_n)} \bmod N$$

where g is exponentiation base in $\mathbb{Z}_N = \{0, 1, \ldots, N\}$.

=> N, g and A(X) are **public**. p and q should be kept **secret**.

RSA accumulator

- What happens if data are not prime numbers?
  Use a prime-generating hash function (“hash-to-prime”).
  - Example: SHA-256 hash function + Miller-Rabin primality test.

- Security: RSA modulus size determines accumulator security.
  - Example: a 3072-bit RSA modulus gives ~ 128 bits of security.

RSA accumulator

Example:

We fix two primes p = 7, q = 11 and compute the RSA modulus N = p · q = 77.

Let X = {13, 31, 37, 59} and g = 2 be an exponentiation base in $\mathbb{Z}_{77}$.

Accumulator value: A(X) = 2^(13 · 31 · 37 ·59) = 39 mod 77

RSA accumulator

X = {13, 31, 37, 59}. A(X) = 39. g = 2.

Membership: A prover P wants to demonstrate to a verifier V that 37 ∈ X.

1) P computes the product of all elements in X - {37}, namely t = 13 · 31 ·59 = 23777.

2) P computes a witness w = g^t = 2^(23777) = 18 mod 7 and sends w to V.

3) V computes w^37 = 18^37 = 39 mod 77. Since A(X) = 39 mod 77 the proof is accepted as valid.

RSA accumulator

Non-membership protocol is made possible by Bezout's lemma:

"For any two integers x and y, the gcd(x, y) can be expressed as a linear combination of x and y".

Bezout cofactors (i.e., a and b s.t. ax+by=gcd(a,b)) can be computed with the Extended Euclidean Algorithm.

RSA accumulator

$X = \{13, 31, 37, 59\}$. $A(X) = 39$. $g=2$.

Non-membership: P wants to convince V that $61 \notin X$.

1) P computes $f = 13 \cdot 31 \cdot 37 \cdot 59 = 879749$.

2) P finds a and b such that $a \cdot f + b \cdot 61 = \gcd(f, 61) = \gcd(879749, 61) = 1$.

3) P sends the witness $w' = (a, b) = (-26, 374975)$ to V.

4) V verifies w' by checking if $A(X)^a \cdot (g^{(61)})^b = g \bmod 77$.
Indeed it is $39^{(-26)} \cdot 2^{(61 \cdot 374975)} = 53 \cdot 32 = 2 \bmod 77$.

RSA accumulator

Advantages:

- Supports batch proofs.
- Proof size is O(1) and independent from set size.

Limitations:

- **Trusted setup**: prover and verifiers have to agree on the RSA modulus N (but without disclosing p and q).
- Modular exponentiations are more expensive than hashing (bit operations).

RSA accumulator

Definition 3.3 (*Strong RSA Assumption*). When the RSA modulus N is sufficiently large and randomly generated, given a value $x \in \mathbb{Z}_N^*$ it is computationally infeasible to find $a, b \in \mathbb{Z}$, with $a \neq \pm 1$, such that $x \equiv b^a \mod N$.

The RSA accumulator is **collision-free** if it is computationally infeasible for an adversary to forge a valid membership (resp. non-membership) proof for an element that is not part (resp. is part) of the accumulated set.

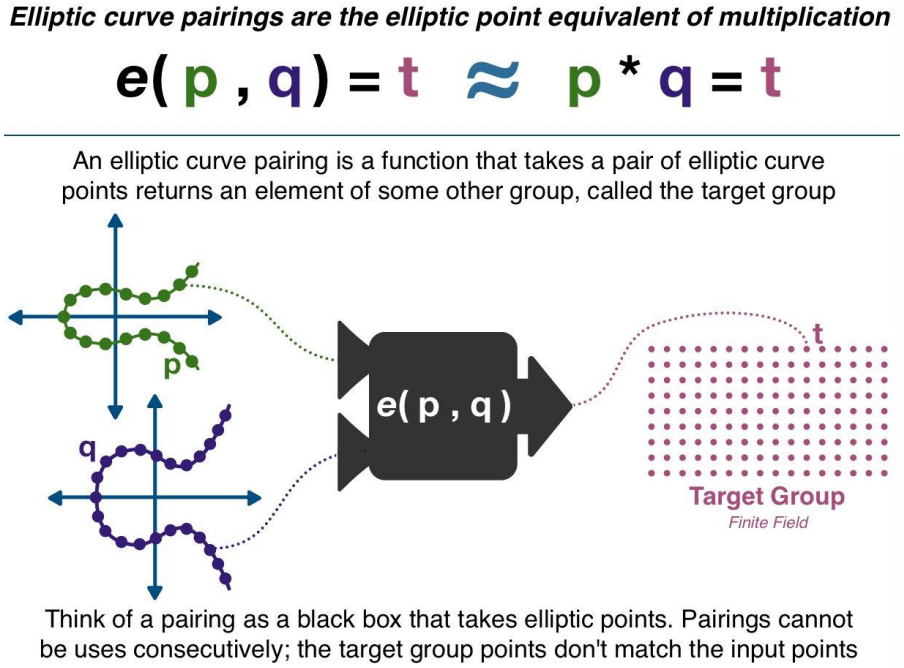
True as long as p and q (factors of N) are kept **secret**.

Bilinear accumulator

Based on **bilinear pairings** (defined over elliptic curves on finite fields)

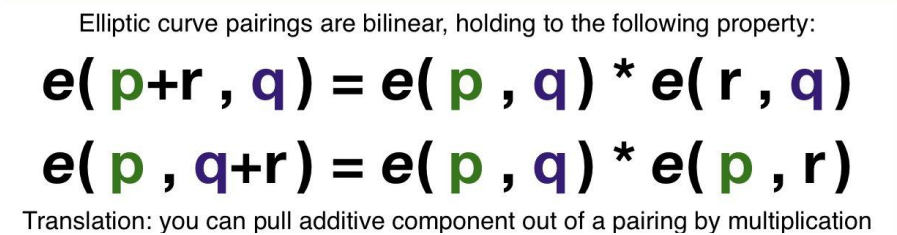
e: $G_1 \times G_2 \rightarrow G_T$

Symmetric pairings:
p and q belong to the same EC group G



Bilinear accumulator

Bilinearity property



If p is added a times to itself: e(p + p + ... + p, q) = e(p^a, q) = e(p, q)^a

If q is added b times to itself: e(p, q + q + ... + q) = e(p, q^b) = e(p, q)^b

Bilinear accumulator

The set $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{Z}_p$ is mapped to a **characteristic polynomial**

$$P_X(z) = (z + x_1) \cdot (z + x_2) \cdot \ldots \cdot (z + x_n)$$

The **accumulator value** of X is

$$A(X) = g^{(P_X(s))}$$

for a random **secret** value $s \in \mathbb{Z}_p$ and a group **generator** g.

Supports both membership and non-membership proofs.

Bilinear accumulator

Membership proof for an element v:

1. Let S = X - {v} and P_S(z) the characteristic polynomial of S.
2. The witness is w = g^(P_S(s)).
3. Witness is valid if and only if e(w, g^(s+v)) = e(A(X), g).

Notice that:

1) e(g^(P_S(s)), g^(s+v)) = e(g, g) ^ (P_S(s) * (s+v)) = e(g, g) ^ (P_X(s))
2) e(A(X), g) = e(g^(P_X(s)), g) = e(g, g) ^ (P_X(s))

Bilinear accumulator

Non-membership proof for an element v relies on the fact that if v is not in X, then gcd(P_X, z+v) = 1.

=> There exist two polynomials U and V s.t.
$U \cdot P_X + V \cdot (z+v) = 1$.

=> U and V can be computed with the Extended Euclidean Algorithm (for polynomials).

Bilinear accumulator

Definition 3.7 (*q-SBDH Assumption*). Let p be a prime number and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric bilinear pairing, where $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups. Let also g be a generator of $\mathbb{G}$ and s a random secret chosen in $\mathbb{Z}_p$. The q-strong bilinear Diffie–Hellman (q-SBDH) assumption holds if and only if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a pair $(e(g, g)^{1/(s+x)}, x)$ with $x \in \mathbb{Z}_p^*$ given a tuple $(g, g^s, g^{s^2}, \ldots, g^{s^q})$ as input is negligible.[2]

Trusted setup phase required.

- Trusted party generates secret value s (called trapdoor) and forgets it.
- Knowledge of s would allow a party to forge fake proofs.
- Public parameters: {g^(s^i) | 0 ≤ i ≤ q} with q fixed in advance.

Accumulated sets can have a maximum cardinality equal to q.

Expressive accumulator

- Based on bilinear pairings (and polynomials).

- Supports **verifiable operations** on accumulated sets.

- Zero-knowledge extension: no verifier may learn extra information about S beyond what it has queried.

Source: [ZKP17] Y. Zhang, J. Katz, and C. Papamanthou. “An Expressive (Zero-Knowledge) Set Accumulator”. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 158–173.

Expressive accumulator

Definition 3.8 (*Expressive Set Accumulator*). Let p be a prime number, g a generator of a cyclic multiplicative group $\mathbb{G}$ and $r, s \in \mathbb{Z}_p$ two randomly and uniformly chosen values. Given a set S, whose elements are drawn from a universe $\mathcal{U} = \{1, \ldots, \ell\}$, let

$$C_S(x) = \sum_{i \in S} x^i \qquad \text{and} \qquad D_S(x, y) = \sum_{i \in S} x^i y^{\ell - i}$$

denote its characteristic polynomials, where x and y are the unknowns. The accumulator value of S is the tuple $d_S = (g^{C_S(s)}, g^{D_S(s,r)}, g^{C_S(r)}, g^{D_S(r,s)})$.

Source: [ZKP17] Y. Zhang, J. Katz, and C. Papamanthou. “An Expressive (Zero-Knowledge) Set Accumulator”. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 158–173.

Expressive accumulator

Examples: Given a set X = {x_1, … , x_N}:

- SUM: prover can demonstrate that s = x_1 + … + x_N is actually the sum of the elements.

- COUNT: prover can demonstrate that N = |X|.

- MIN/MAX: prover can show that m (resp. M) is min(X) (resp. max(X)).

Source: [ZKP17] Y. Zhang, J. Katz, and C. Papamanthou. “An Expressive (Zero-Knowledge) Set Accumulator”. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 158–173.

Expressive accumulator

Examples: Given a set X = {x_1, … , x_N} and another set Y = {y_1, …, y_M}

- **Intersection**: prove that a set I is equal to X ∩ Y.

- **Union**: prove that a set U is equal to X ∪ Y.

Source: [ZKP17] Y. Zhang, J. Katz, and C. Papamanthou. “An Expressive (Zero-Knowledge) Set Accumulator”. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 158–173.

Expressive accumulator

The gain in terms of expressiveness is paid with a public key of size O(|U|^2), where U is the size of universe from which elements of the accumulated sets are drawn.

Source: [ZKP17] Y. Zhang, J. Katz, and C. Papamanthou. “An Expressive (Zero-Knowledge) Set Accumulator”. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. IEEE, 2017, pp. 158–173.

Accumulator comparison

Table 1
A comparison of different set accumulators for a set of n elements. For each construction, we detail: whether it needs a trusted setup phase and support batch proofs; the size of the public parameters (PK); the type of primitive operation it is built upon; the size of membership and non-membership proofs and the asymptotic complexity of generating and verifying them.

	Setup	Batch	PK	Operation	Membership			Non-Membership		
					Size	Gen.	Ver.	Size	Gen.	Ver.
RSA	Yes	Yes	$O(1)$	Modular Exp.	$O(1)$	$O(n)$	$O(1)$	$O(1)$	Ext. Euclid	$O(1)$
Bilinear	Yes	Yes	$O(q)$	EC mult.	$O(1)$	$O(n \log n)$	$O(1)$	$O(1)$	Ext. Euclid	$O(1)$
ESA	Yes	Yes	$O(\lvert\mathcal{U}\rvert^2)$	EC mult.	$O(1)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(1)$
Merkle	No	No	$O(1)$	Hashing	$O(\log n)$	$O(\log n)$	$O(\log n)$	n/a	n/a	n/a

Source: [LBDR23] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. "A survey of set accumulators for blockchain systems." Computer Science Review 49 (2023): 100570.

Accumulator comparison

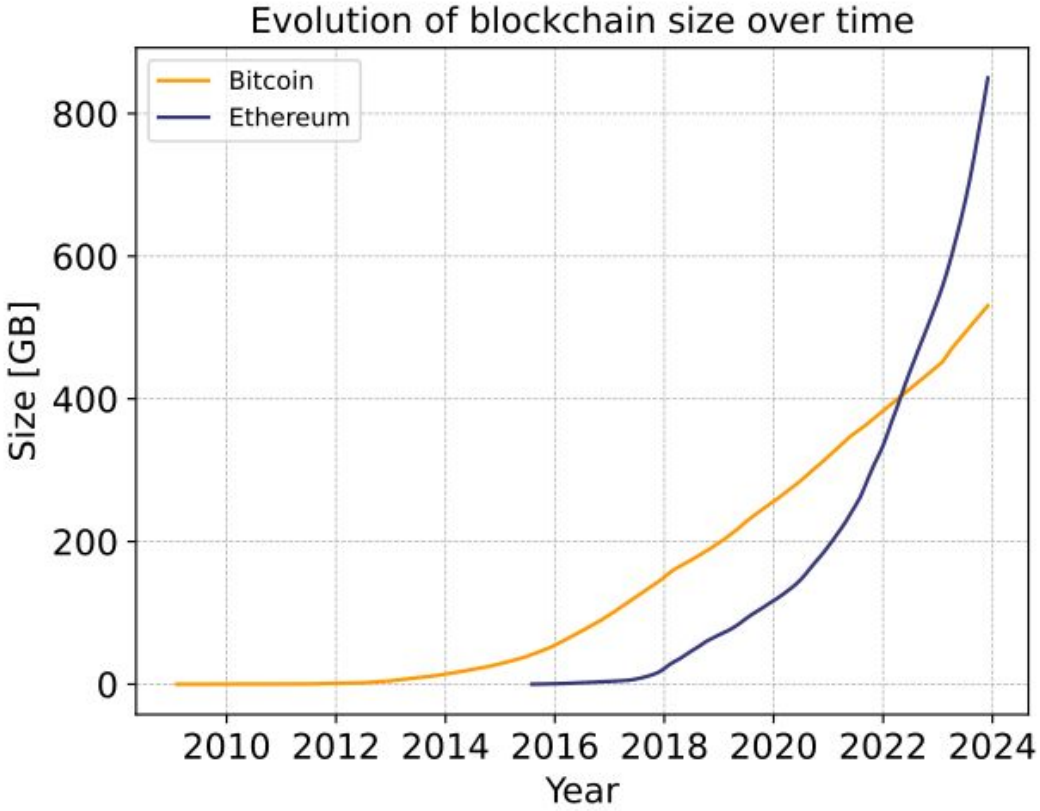
Table 1
A comparison of different set accumulators for a set of n elements. For each construction, we detail: whether it needs a trusted setup phase and support batch proofs; the size of the public parameters (PK); the type of primitive operation it is built upon; the size of membership and non-membership proofs and the asymptotic complexity of generating and verifying them.

	Setup	Batch	PK	Operation	Membership			Non-Membership		
					Size	Gen.	Ver.	Size	Gen.	Ver.
RSA	Yes	Yes	$O(1)$	Modular Exp.	$O(1)$	$O(n)$	$O(1)$	$O(1)$	Ext. Euclid	$O(1)$
Bilinear	Yes	Yes	$O(q)$	EC mult.	$O(1)$	$O(n \log n)$	$O(1)$	$O(1)$	Ext. Euclid	$O(1)$
ESA	Yes	Yes	$O(\lvert\mathcal{U}\rvert^2)$	EC mult.	$O(1)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$	$O(1)$
Merkle	No	No	$O(1)$	Hashing	$O(\log n)$	$O(\log n)$	$O(\log n)$	n/a	n/a	n/a

=> Hashing is less expensive than modular exponentiations and EC multiplications. No trusted setup, but proof size is O(log n).

Source: [LBDR23] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. "A survey of set accumulators for blockchain systems." Computer Science Review 49 (2023): 100570.

Query authentication


Evolution of blockchain size over time
Bitcoin
Ethereum
800
600
400
200
0
Size [GB]
2010
2012
2014
2016
2018
2020
2022
2024
Year

Query authentication

Resource-constrained light nodes need to query untrusted full nodes to retrieve on-chain data.

Challenges:

1) How to efficiently retrieve such data?
2) How to guarantee data integrity?

Query authentication

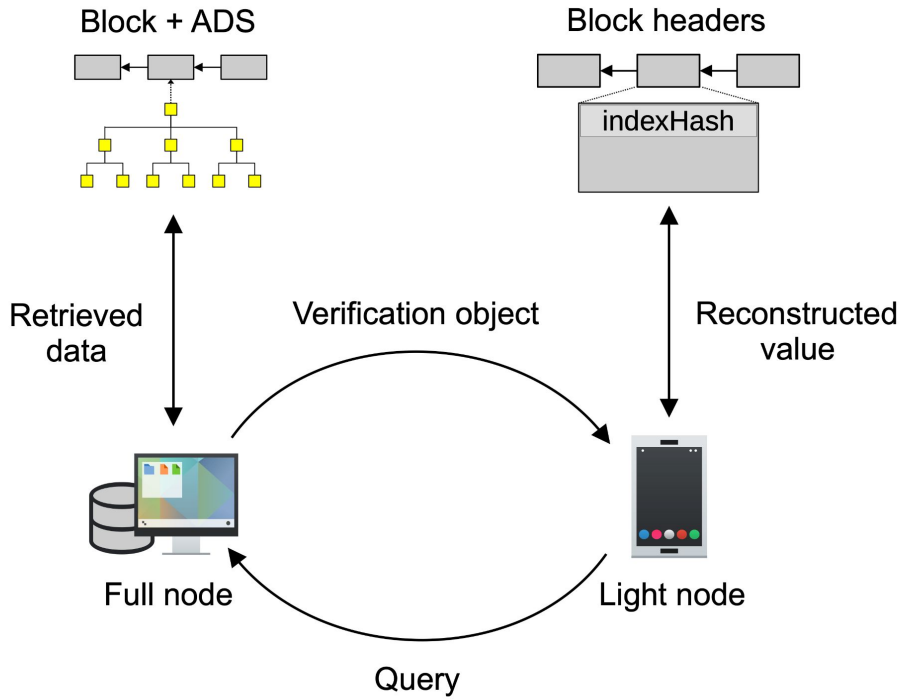
Set accumulators can be used to construct **Authenticated Data Structures (ADSs)**.

An **ADS** on a collection of data objects C incorporates cryptographic information so that:

1) an untrusted **prover** can carry out operations on C;
2) a **verifier** can efficiently check the **correctness** of the result of such operations.

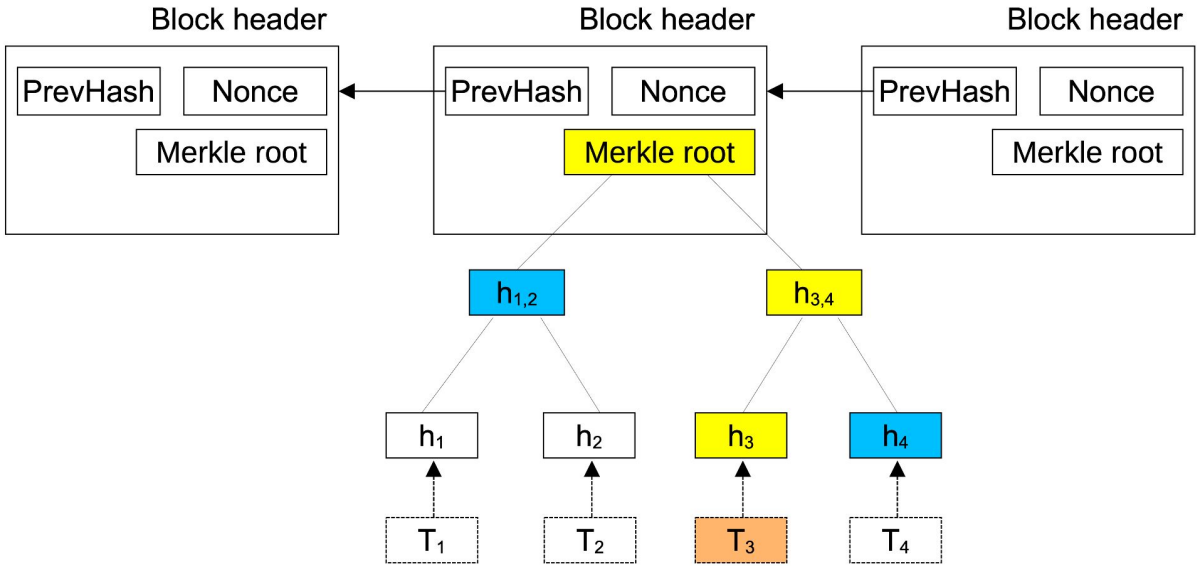
Query authentication

- Full node: **prover**
- Light node: **verifier**
- Verifier cannot access the data collection (only known to the prover).
- Verifier maintains only a small piece of authentication information summarizing the current collection state.



Bitcoin Simplified Payment Verification (SPV)

Is a given TX confirmed (i.e., included in a block)?



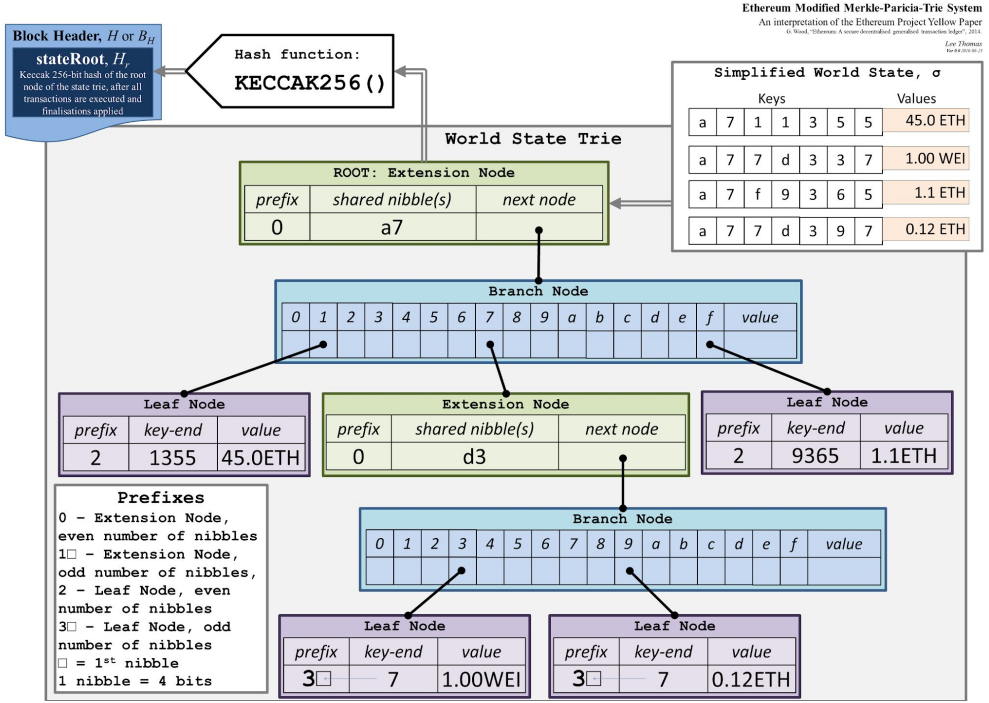
Ethereum Merkle Patricia Trie

ADS for managing state in Ethereum

Can build proofs as in Merkle Trees.

Key-value store:

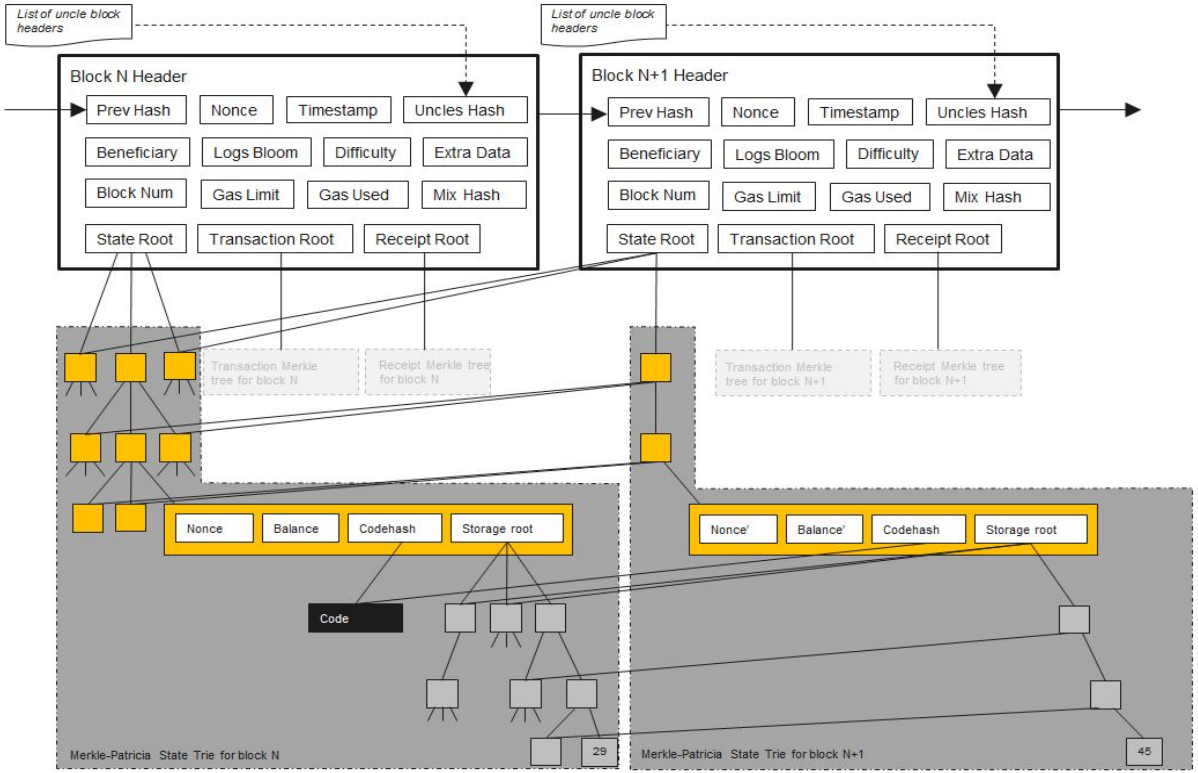
- Key is determined by path
- Value included in a leaf node



Ethereum Merkle Patricia Trie

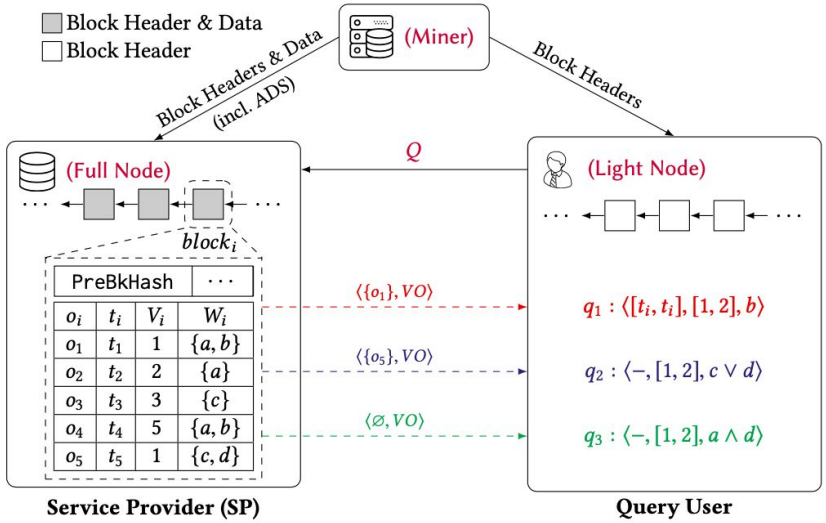
StateRoot (hash of MPT root) is embedded in each block header.

Represents a commitment to current Ethereum state.



vChain

- Enables **verifiable queries** on a blockchain.
- Supports both **intra-block** and **inter-block** queries.



- Guarantees **authenticity** and **completeness**.
  - Authenticity: results are not modified w.r.t. on-chain data.
  - Completeness: no matching element is omitted from the results.
- Relies on **bilinear accumulators** (or ESA-based construction).

Source: [XZX19] C. Xu, C. Zhang, and J. Xu, “vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases,” Proceedings of the 2019 International Conference on Management of Data. ACM, Jun. 25, 2019.

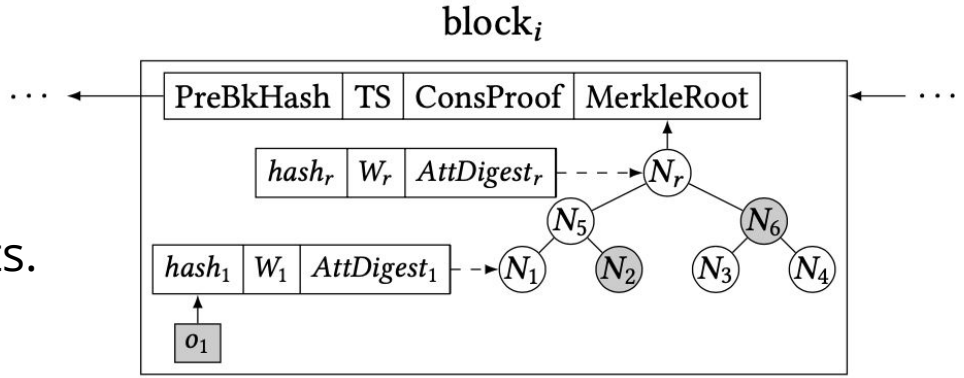
vChain

Each block stores a set of data objects.

Intra-block index

- Leaf node
  a. hash of data object;
  b. multiset with object attributes;
  c. accumulator.

- Internal node
  a. hash of child nodes;
  b. union of children multisets;
  c. accumulator.



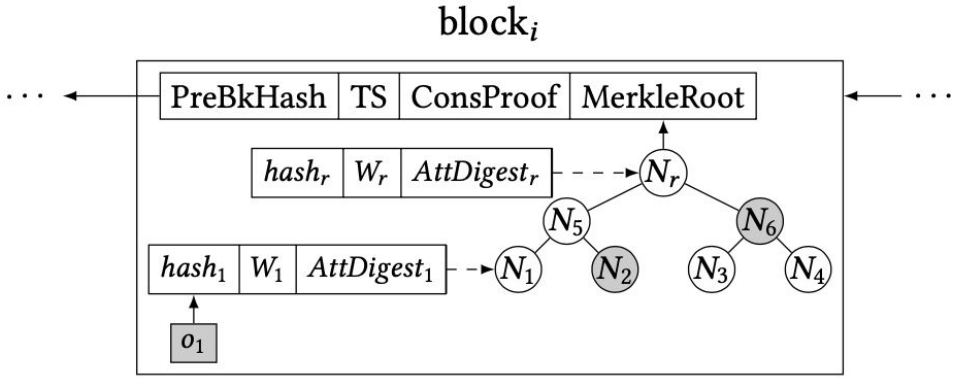
Node	Object	Set Attributes
N_1	o_1	W_1 = {"Sedan", "Benz"}
N_2	o_2	W_2 = {"Sedan", "Audi"}
N_3	o_3	W_3 = {"Van", "Benz"}
N_4	o_4	W_4 = {"Van", "BMW"}

Source: [XZX19] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases," Proceedings of the 2019 International Conference on Management of Data. ACM, Jun. 25, 2019.

vChain

Validators build ADS for each block and embed root digest into header.

Full node receives query and traverses the ADS in a top-down fashion to retrieve results and build VO.



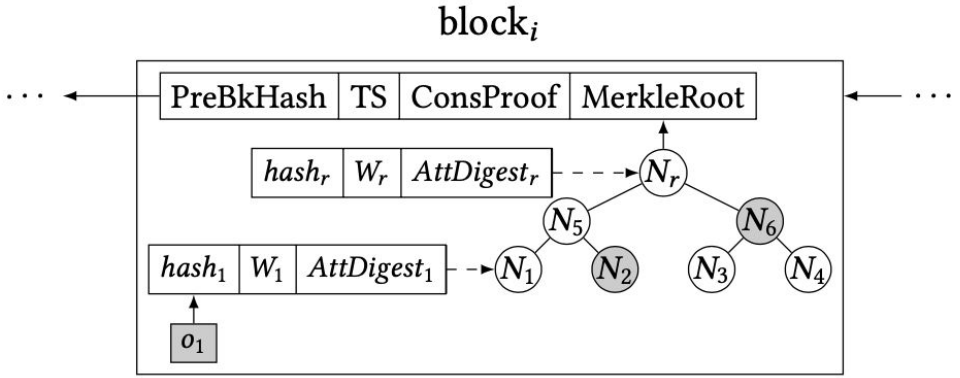
Node	Object	Set Attributes
N_1	o_1	W_1 = {"Sedan", "Benz"}
N_2	o_2	W_2 = {"Sedan", "Audi"}
N_3	o_3	W_3 = {"Van", "Benz"}
N_4	o_4	W_4 = {"Van", "BMW"}

Source: [XZX19] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases," Proceedings of the 2019 International Conference on Management of Data. ACM, Jun. 25, 2019.

vChain

When an internal node does not match the query, a non-membership proof is inserted in the VO.

=> All elements in the corresponding subtree do not match the query.



Node	Object	Set Attributes
N_1	o_1	W_1 = {“Sedan”, “Benz”}
N_2	o_2	W_2 = {“Sedan”, “Audi”}
N_3	o_3	W_3 = {“Van”, “Benz”}
N_4	o_4	W_4 = {“Van”, “BMW”}

Source: [XZX19] C. Xu, C. Zhang, and J. Xu, “vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases,” Proceedings of the 2019 International Conference on Management of Data. ACM, Jun. 25, 2019.

Skip index

A data structure for efficient **inter-block** queries (and authentication).

Use case: searching for events along the Ethereum blockchain.

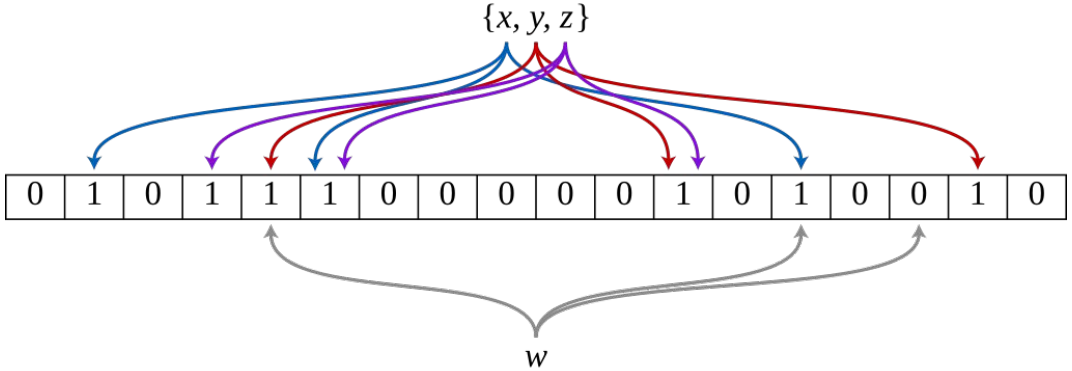
Membership queries: is a given event included in a certain block?

Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Skip index

Ethereum block header includes a secure Bloom filter (*logsBloom*) that summarizes block event logs.

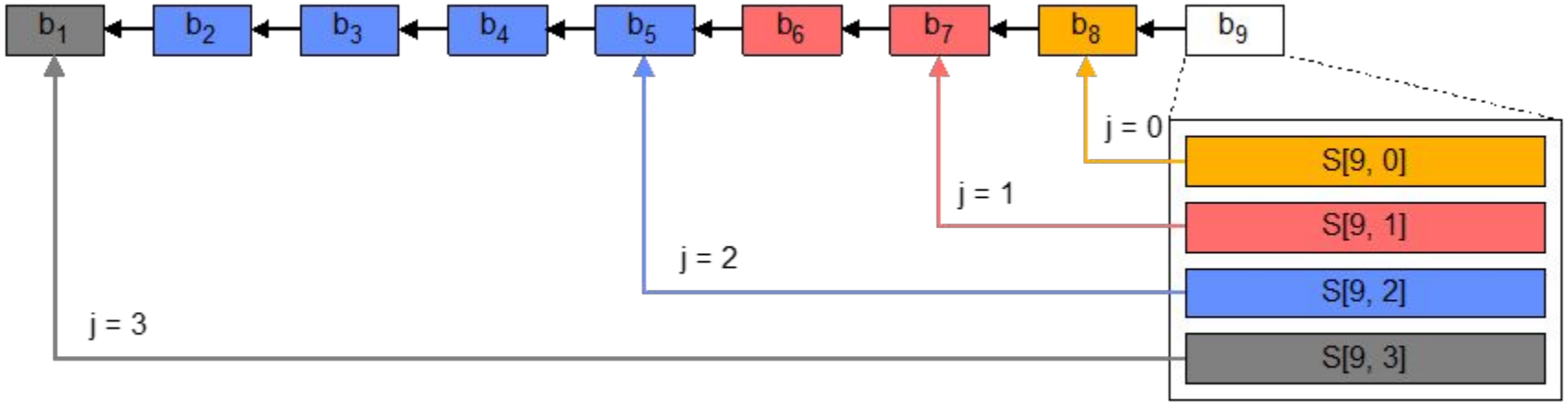
- Filter size: 256 bytes
- Uses Keccak-256 cryptographic hash function for insertion/lookup.



Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Skip index

Each block is enhanced with a sequence of Bloom filters summarizing the content of its predecessors at an exponentially increasing distance.



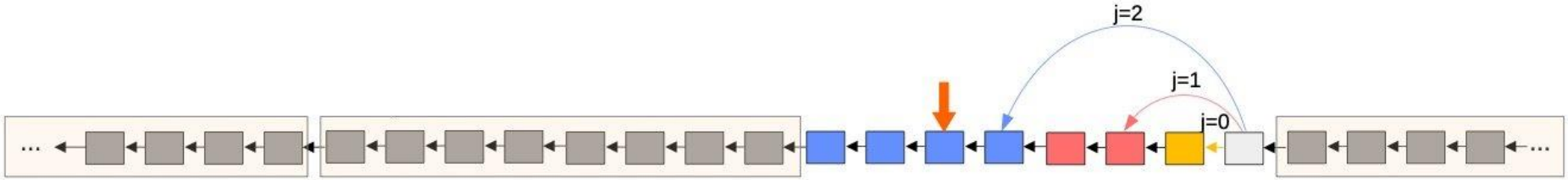
Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Skip index

Skip index with E entries (i.e., filters).

Goal: Find first (i.e., most recent) occurrence of event x within a range of blocks [l, u].

1. Partition [l, u] in chunks of 2^E blocks.
2. Recursively explore the range corresponding to the first matching filter.



Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Skip index

Query authentication process

1. Hash of skip index included in block header
2. Full node builds VO as the ordered sequence of indices of visited blocks, i.e., VO = (S_1, ..., S_v)
3. Light node uses VO to perform a guided search
   a. Compares hash(S_i) with header of block b_i
   b. Uses S_i to determine next block to be visited

=> Verification complexity = search complexity

Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Skip index

- Bloom filters are **symmetric** accumulators (i.e., no need to compute a witness to check membership).

- Optimal size also depends on the number of elements in the set
  => Bloom filters are less space-efficient w.r.t. accumulators.

- Hashing operations are more efficient than EC operations (and do not require trusted setup).

Source: [LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025

Hands-on session

GitHub repository

https://github.com/mloporchio/Tutorial-ICBC2025

Spatial data

Represent the position of people/objects
in a given coordinate space.

Usually represented as d-dimensional vectors
(x_1, ..., x_d)
with integer-valued or real-valued components.

Blockchains for spatial data

THE WALL STREET JOURNAL.

Latest World Business U.S. Politics Economy Tech **Markets & Finance** Opinion Arts Lifestyle Real Estate Personal Finance

PROPERTY REPORT

A Pioneer in Real Estate Blockchain Emerges in Europe

Sweden's Lantmäteriet will test using blockchain technology for property sales

LANTMÄTERIET

Blockchains for spatial data

Some relevant use cases:

- **Land registry:**
  - Collect information about the owner, location and size of a property and to manage transfers and purchases.
  - Perform real estate transfers.

- **Supply chain management:**
  - Tack goods and record information about their life cycle.

- **Proofs of location:**
  - Certify the presence of an entity in a geographic location at a given time
  - Grant right (e.g., access a resource) to users on the basis of their position, as it happens in access control systems.

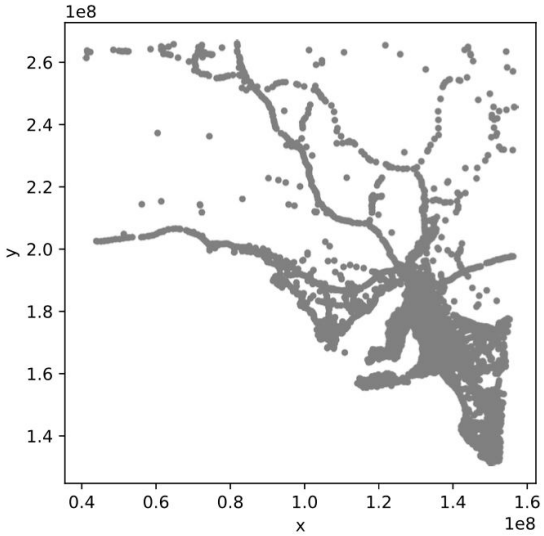
Blockchain for spatial data

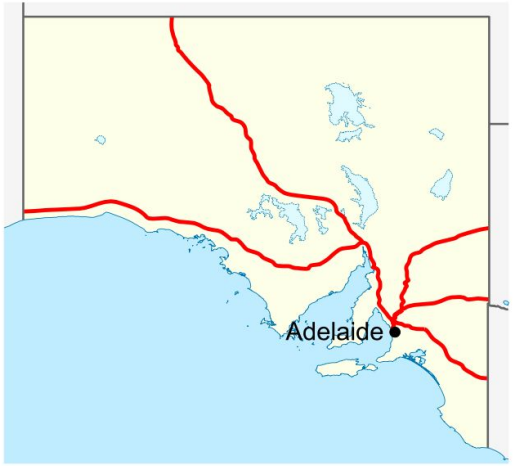
In all scenarios, there is a need for *efficient* and *authenticated* spatial data retrieval.

Spatial query authentication

Query authentication protocol for spatial blockchains

Data set:
South Australia Car Crashes (2012 - 2020)*





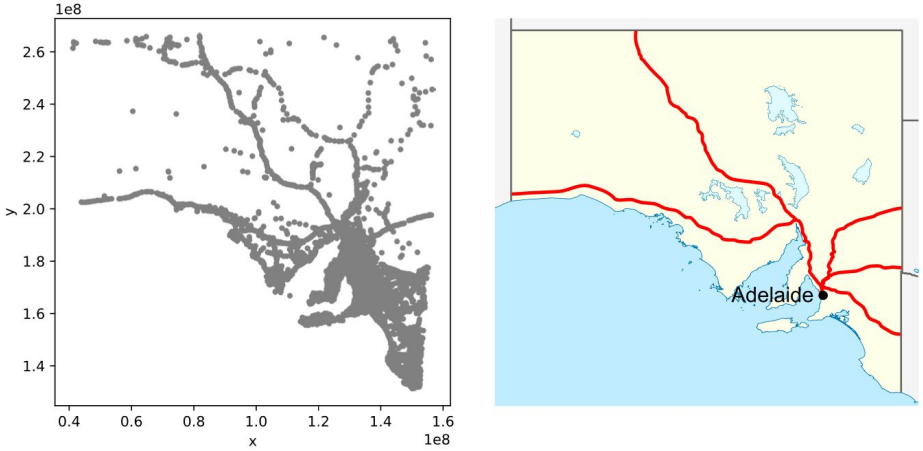
*Available at: https://data.sa.gov.au/data/dataset/road-crash-data

Source: [LBDR21] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. “Authenticating spatial queries on blockchain systems.” IEEE Access 9 (2021): 163363-163378.

Spatial query authentication

Range query: return all points falling within a given (axis-aligned) rectangle.

$Q = (lx, ux, ly, uy) = \{(x, y) \in \mathbb{R}^2 : lx \leq x \leq ux \wedge ly \leq y \leq uy\}$

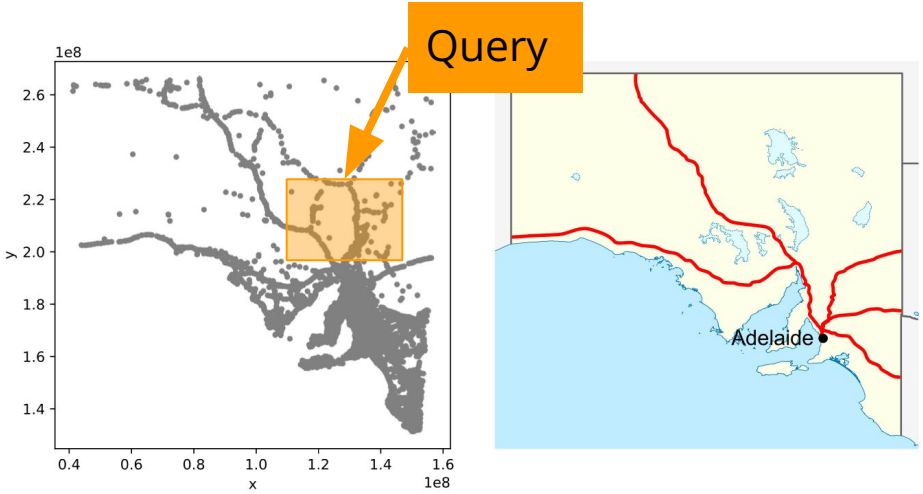


Source: [LBDR21] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. “Authenticating spatial queries on blockchain systems.” IEEE Access 9 (2021): 163363-163378.

Spatial query authentication

Range query: return all points falling within a given (axis-aligned) rectangle.

$Q = (lx, ux, ly, uy) = \{(x, y) \in \mathbb{R}^2 : lx \leq x \leq ux \wedge ly \leq y \leq uy\}$



Source: [LBDR21] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. “Authenticating spatial queries on blockchain systems.” IEEE Access 9 (2021): 163363-163378.

Spatial query authentication

Which notion of **correctness** for query results?

1) **Authenticity:** returned records are not modified w.r.t. data on the blockchain.

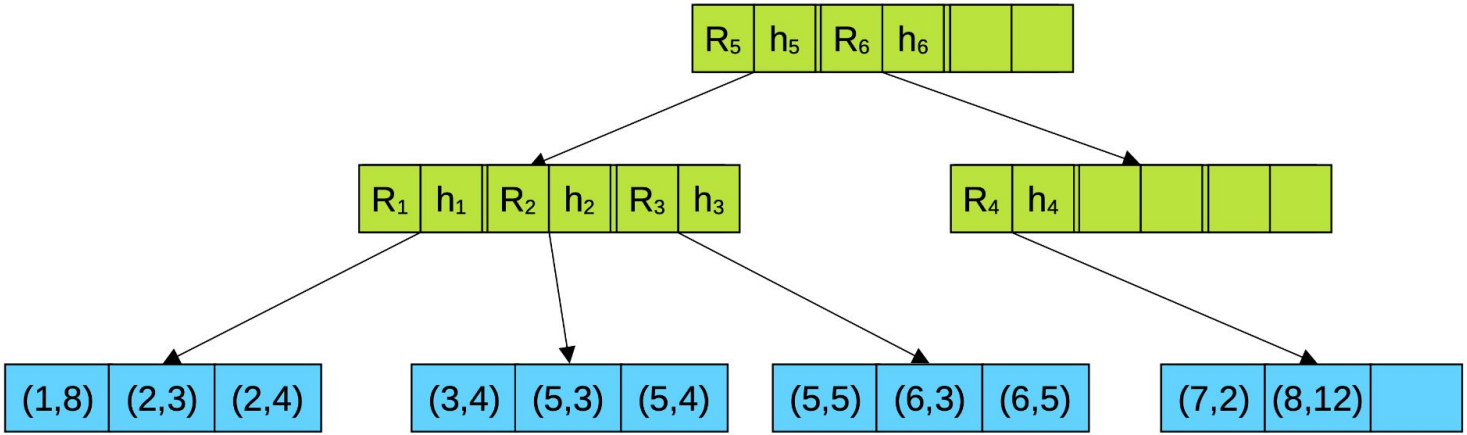
2) **Completeness:** no matching record is omitted from the result.

Source: [LBDR21] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. “Authenticating spatial queries on blockchain systems.” IEEE Access 9 (2021): 163363-163378.

Spatial query authentication

Merkle R-tree: ADS for range query authentication on multidimensional data.

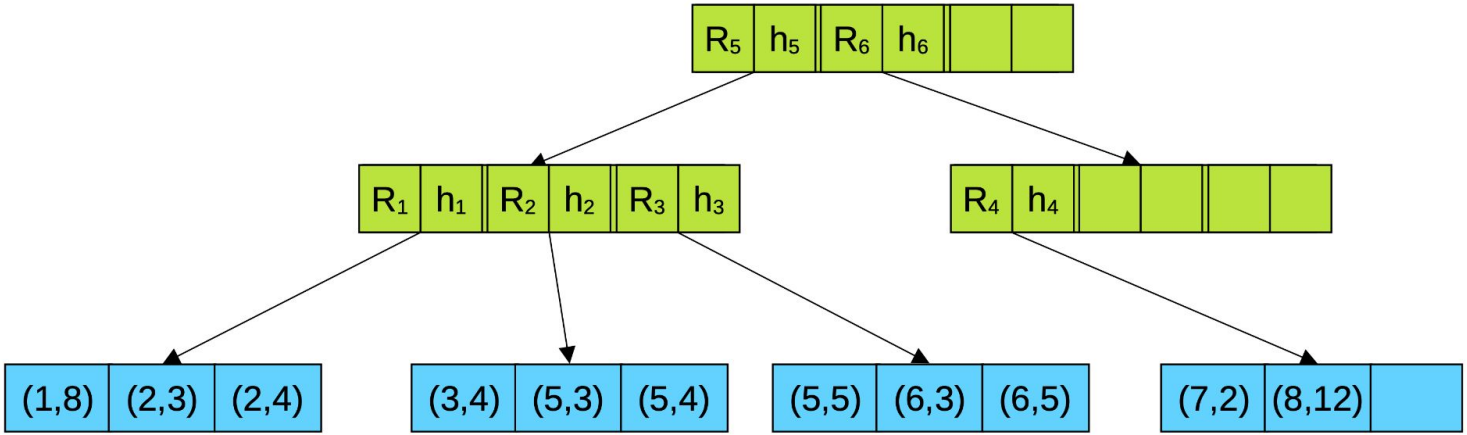
Combines Merkle trees and R-trees (c = page capacity).



Source: [YPPK09] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, “Authenticated indexing for outsourced spatial databases,” VLDB J., vol. 18, no. 3, pp. 631–648, Jun. 2009.

Spatial query authentication

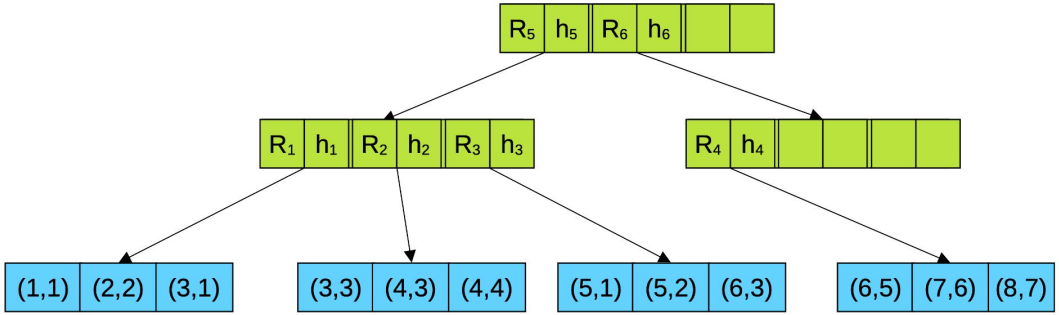
- **Leaf nodes:** at most c data points.
- **Internal nodes:** at most c entries (i.e., minimum bounding rectangle of subtree and cryptographic digest).

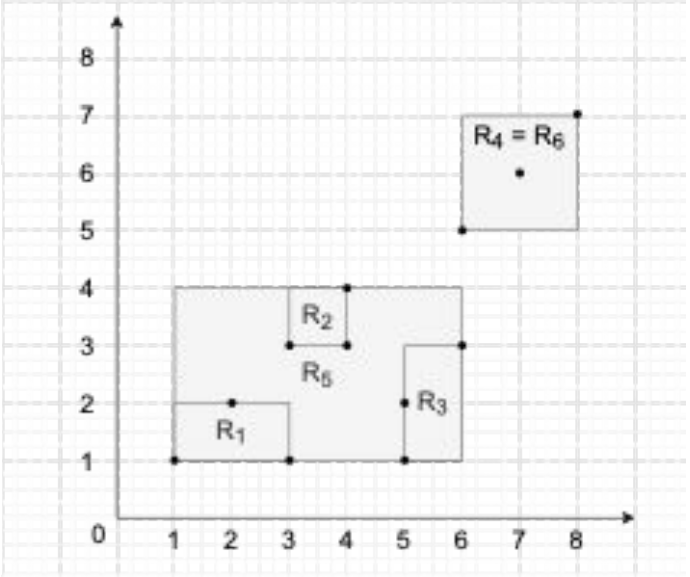


Source: [YPPK09] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, “Authenticated indexing for outsourced spatial databases,” VLDB J., vol. 18, no. 3, pp. 631–648, Jun. 2009.

Spatial query authentication

Bounding rectangle of internal node is computed as minimum area rectangle enclosing all subtree rectangles.

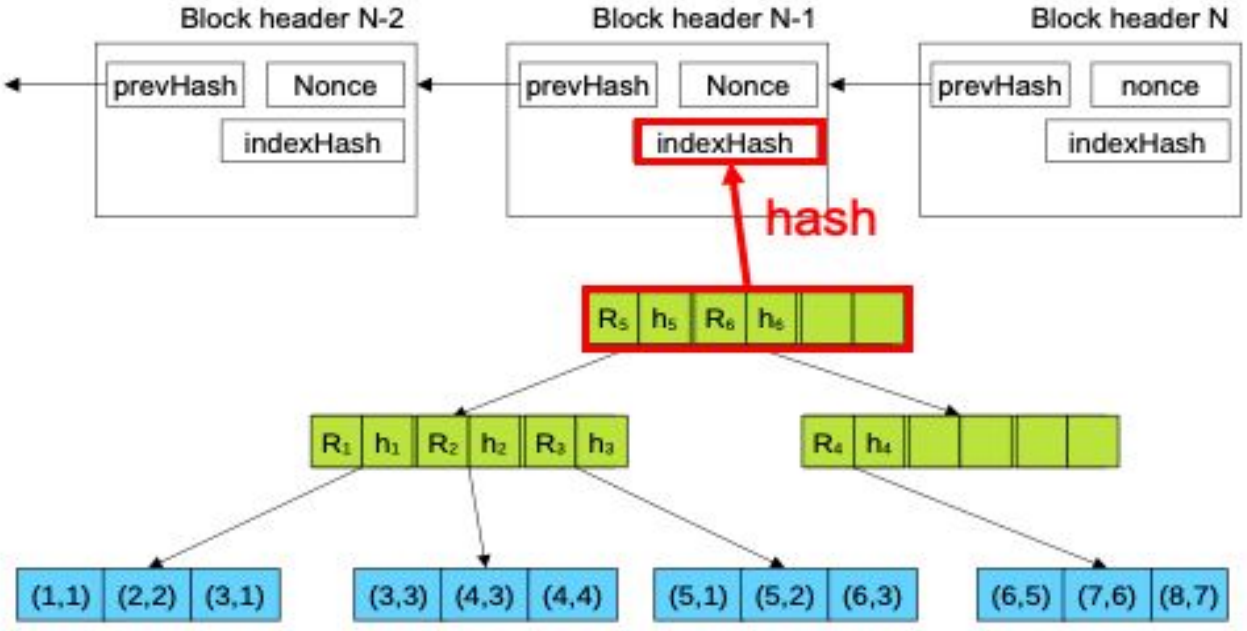




Source: [YPPK09] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, “Authenticated indexing for outsourced spatial databases,” VLDB J., vol. 18, no. 3, pp. 631–648, Jun. 2009.

Spatial query authentication

Embedding root hash in block header enables authentication.

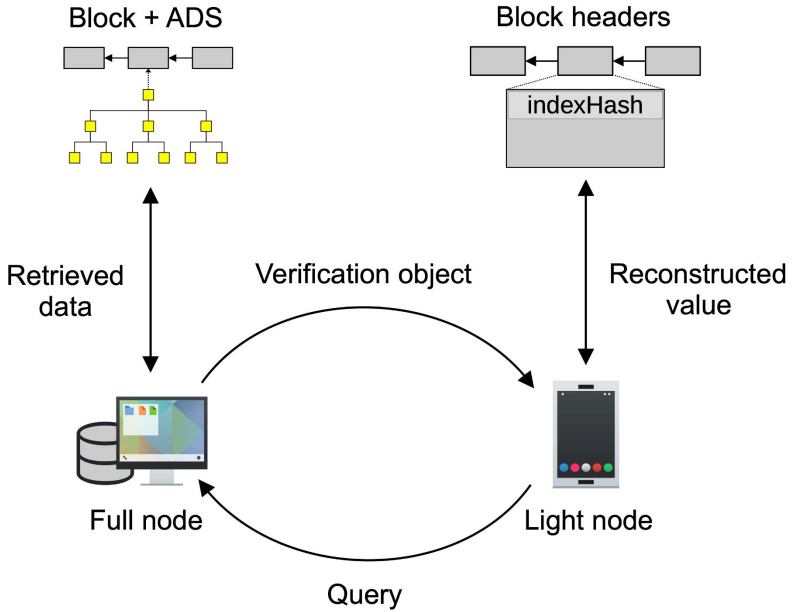


Spatial query authentication

1. **Construction** algorithm
2. **Query** algorithm
3. **Verification** algorithm

ADS is constructed by **validators** before adding the block to the chain.

Validator also embeds tree root hash in block header.



Spatial query authentication

Merkle R-tree construction algorithm

1) Sort all points in the set.

2) Create leaves (i.e., groups of c points).

3) Merge nodes iteratively until only one is left.

Algorithm 1 The Packed MR-Tree Algorithm

function PackedMRTree($\mathcal{P}$, c)
  Input: a set of points $\mathcal{P} = \{P_1, \ldots, P_m\}$, the page capacity c;
  Output: the root of the MR-tree;
  Require: $1 < c < m$;
  $N \leftarrow$ empty list;
  Sort the set $\mathcal{P}$;
  $L \leftarrow$ left fill partition $\mathcal{P}$ into $\lceil m/c \rceil$ groups of c elements;
  for each group $G \in L$ **do**
    $\ell \leftarrow$ create a leaf from the points in G;
    $R_\ell \leftarrow$ MBR of all points in G;
    $h_\ell \leftarrow$ hash of all points in G;
    Add (ℓ, R_ℓ, h_ℓ) to N;
  end for
  while $|N| > 1$ **do**
    $L \leftarrow$ left fill partition N into $\lceil |N|/c \rceil$ groups of c elements;
    $N' \leftarrow$ empty list;
    for each group $G \in L$ **do**
      Let $\{(n_1, R_1, h_1), \ldots, (n_k, R_k, h_k)\}$ be the content of G;
      $n \leftarrow$ create a node from the elements in G;
      $R_n \leftarrow R_1 \sqcup \ldots \sqcup R_k$;
      $h_n \leftarrow \mathcal{H}(R_1|h_1| \ldots |R_k|h_k)$;
      Add (n, R_n, h_n) to N';
    end for
    $N \leftarrow N'$;
  end while
  return the only element of N;
end function

Spatial query authentication

Complexity: If the page capacity c is constant, then the time complexity of Construction Algorithm is O(m log m), where m is the number of points.

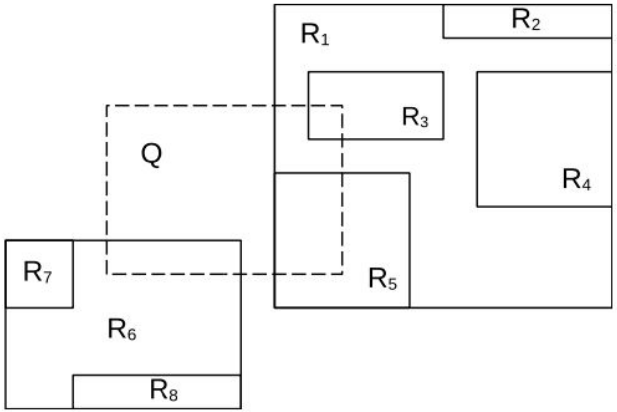
- Sorting is the most expensive phase.

- Different sorting strategies yield different trees.

Spatial query authentication

Query algorithm: recursively explore all subtrees whose rectangle overlaps with the query.

All records in a visited leaf are added to VO.



Algorithm 2 Spatial Query Algorithm

```
1:  function Query(z, Q)
2:      Input: a node z, a query Q;
3:      Output: the VO for the subtree rooted in z;
4:      VO ← empty list;
5:      if z is a leaf then
6:          Append all records in z to VO;
7:      else
8:          for all entries e in z do
9:              if e.R ∩ Q ≠ ∅ then
10:                 VO′ ← QUERY(e.subtree, Q);
11:                 Append VO′ to VO;
12:             else
13:                 Append [e.R, e.h] to VO;
14:             end if
15:         end for
16:     end if
17:     return VO;
18: end function
```

Spatial query authentication

Verification algorithm: reconstruct the Merkle R-tree root starting from the received proof.

Hash of the obtained root is compared against block header for authentication.

Algorithm 3 Spatial Verification Algorithm

```
1:  function Verify(Q, VO)
2:      R_r ← R_∅;
3:      RS ← empty list;
4:      s ← ϵ;
5:      for all lists L ∈ VO do
6:          if L contains records then
7:              Let P_L be the set of records;
8:              for each X ∈ P_L do
9:                  if X matches Q then
10:                     Add X to RS;
11:                 end if
12:                 R_r ← R_r ⊔ X;
13:                 s ← s | X;
14:             end for
15:         else
16:             if L contains pruned node information then
17:                 R_L ← MBR of the pruned node;
18:                 h_L ← digest of the pruned node;
19:             else
20:                 (RS_L, R_L, h_L) ← VERIFY(Q, L);
21:                 RS ← RS ∪ RS_L;
22:             end if
23:             R_r ← R_r ⊔ R_L;
24:             s ← s | R_L | h_L;
25:         end if
26:     end for
27:     h_r ← H(s);
28:     return (RS, R_r, h_r);
29: end function
```

Spatial query authentication

Different sorting strategies for construction:

1) Lexicographic sorting (L)

2) Sorting based on Z-order curves (Z)

3) Z-order with rank space mapping (ZR)

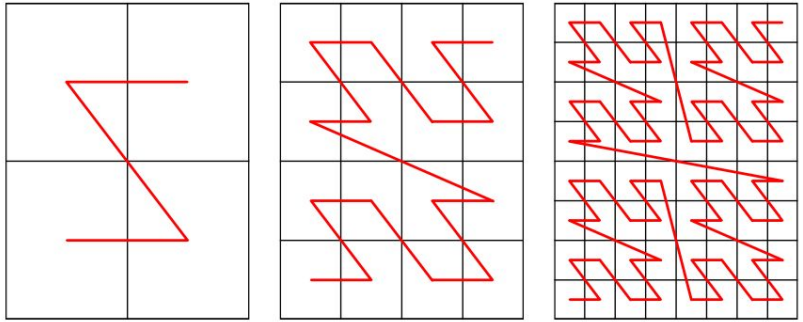


FIGURE 3. Z-order curves in $\mathbb{Z}_n^2$, with $n = 2, 4, 8$.

Source: [LBDR21] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. “Authenticating spatial queries on blockchain systems.” IEEE Access 9 (2021): 163363-163378.

Bilinear accumulator

Implemented in the **Cryptimeleon** library (https://cryptimeleon.github.io).

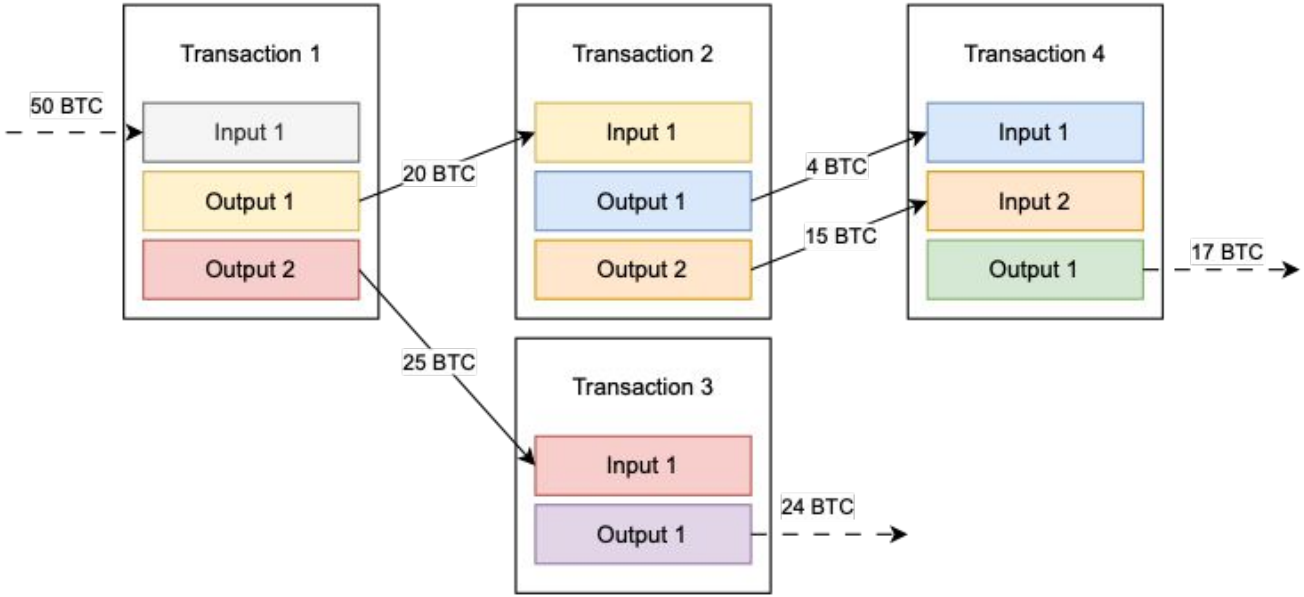
- Very simple
- Not fully optimized :-(

Source: Jan Bobolz et al. "Cryptimeleon: A Library for Fast Prototyping of Privacy-Preserving Cryptographic Schemes," in IACR Cryptol. ePrint Arch., vol. 2021, pp. 961, 2021.

Stateless Transaction Validation

Transactions

In the UTXO model, new transactions spend outputs of previous transactions.



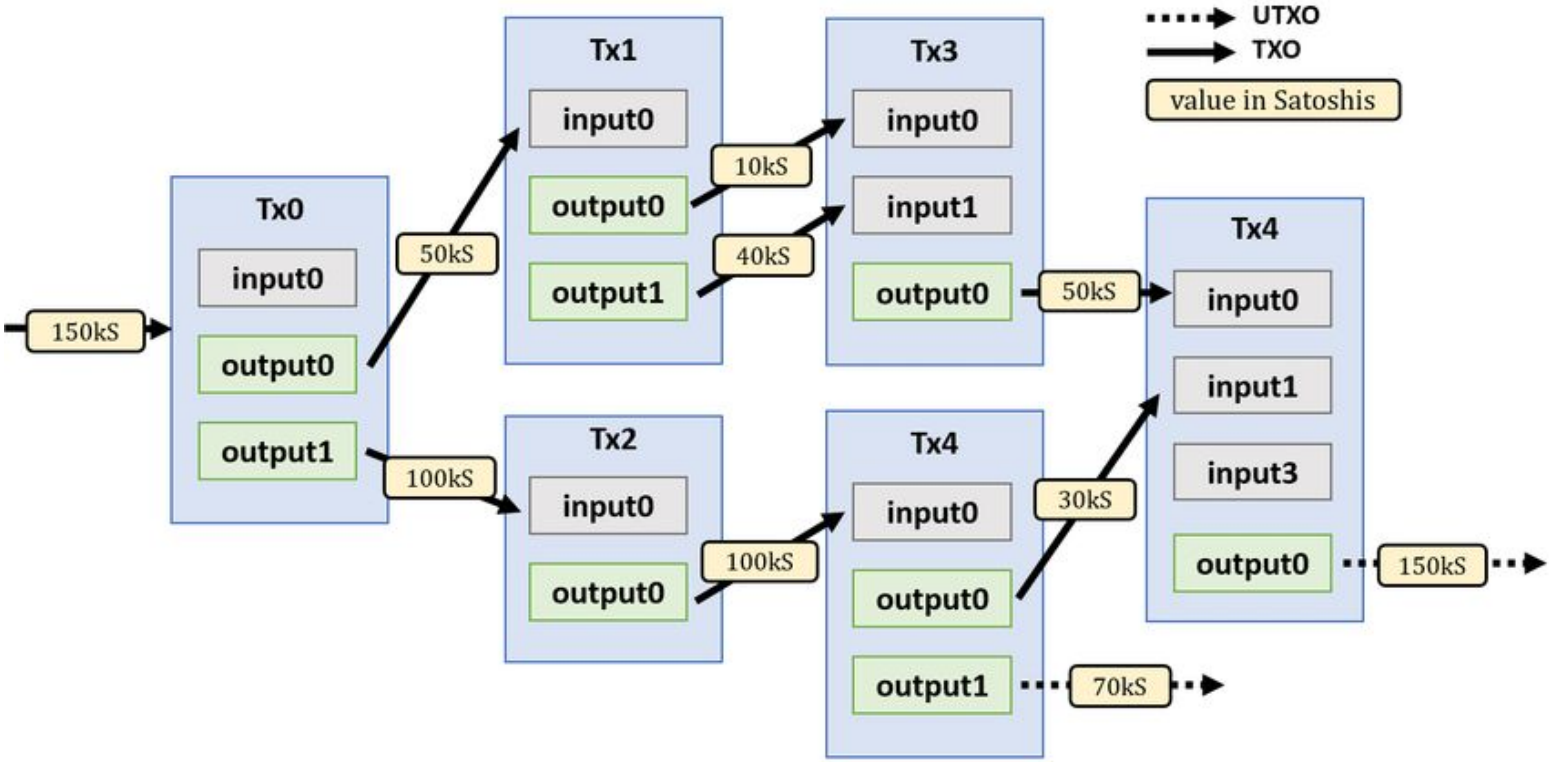
Transaction validation

Valid transaction: input funds correspond to unspent funds.

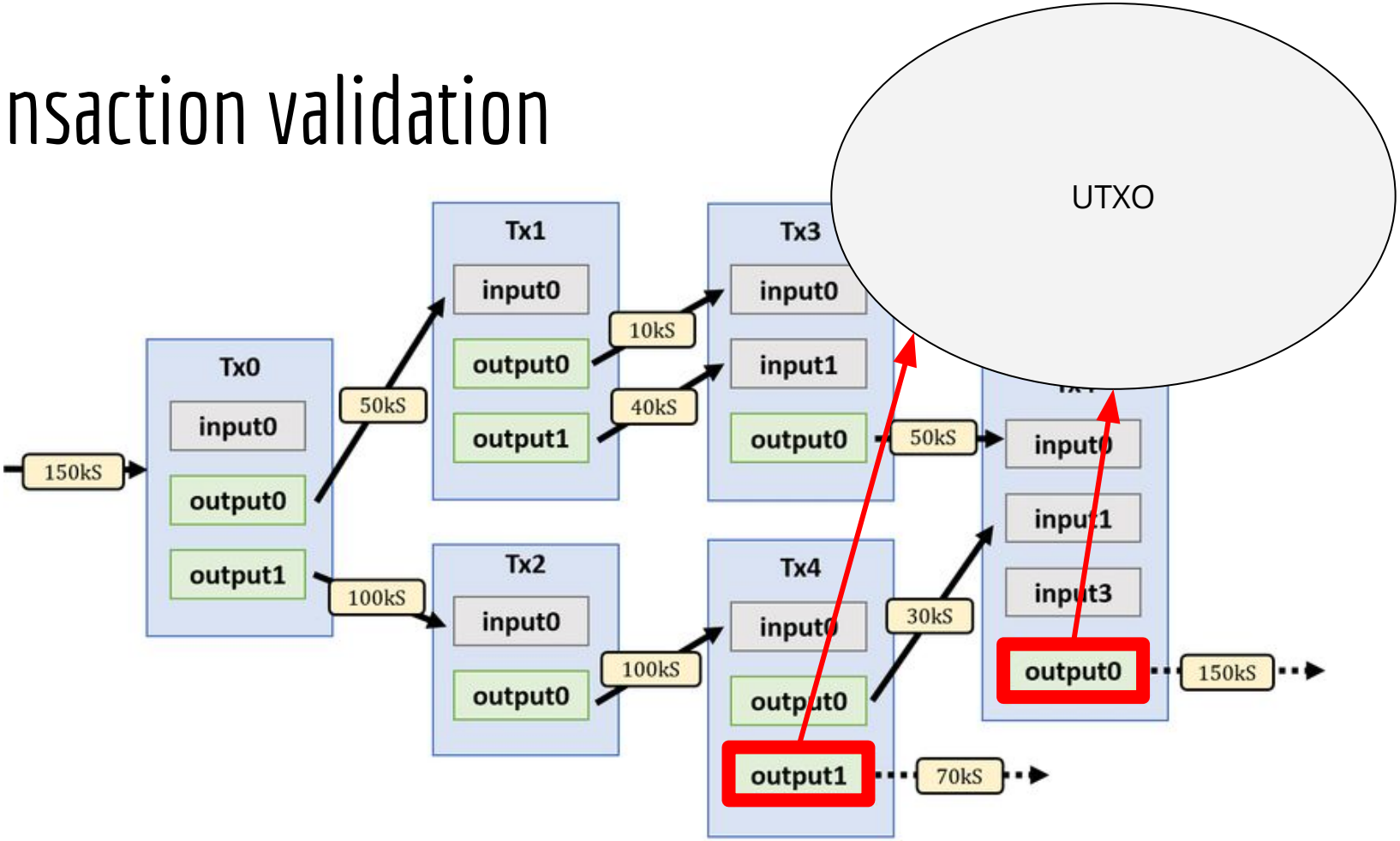
1) TX inputs coincide with outputs of previously confirmed transactions.

2) No other confirmed TX uses these funds as its inputs.

Transaction validation

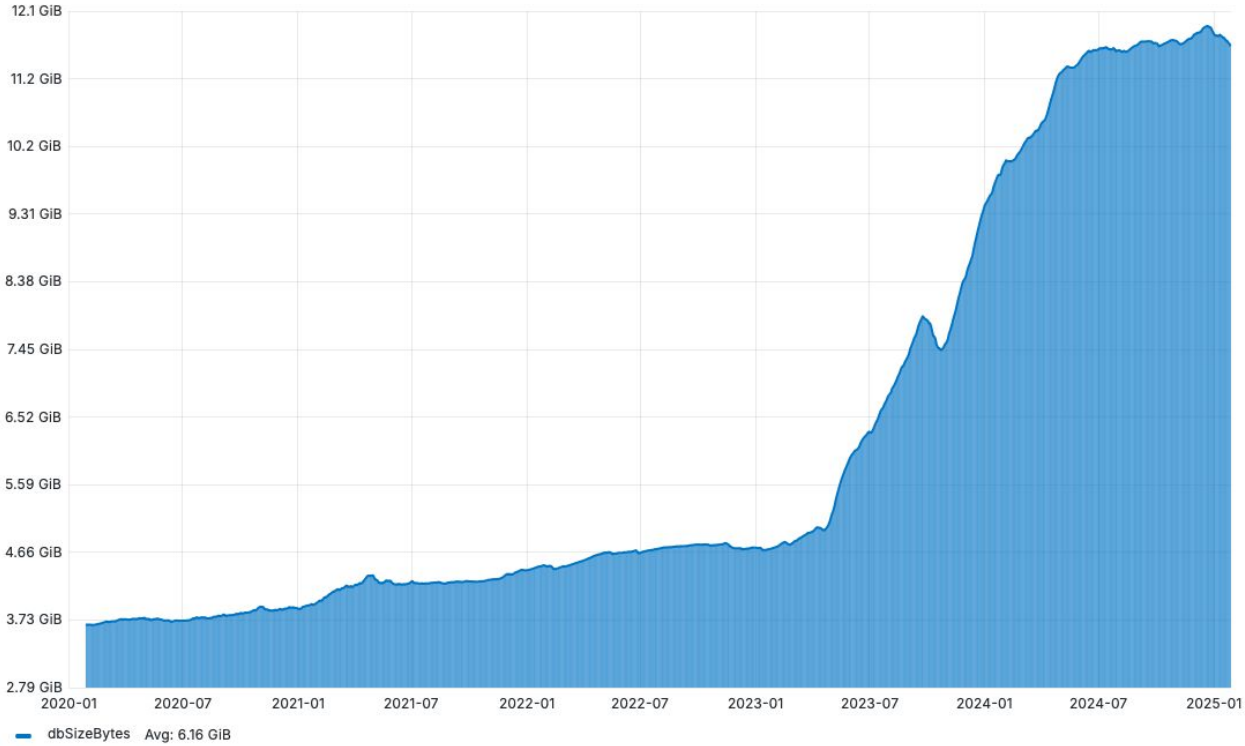


Transaction validation



Transaction validation

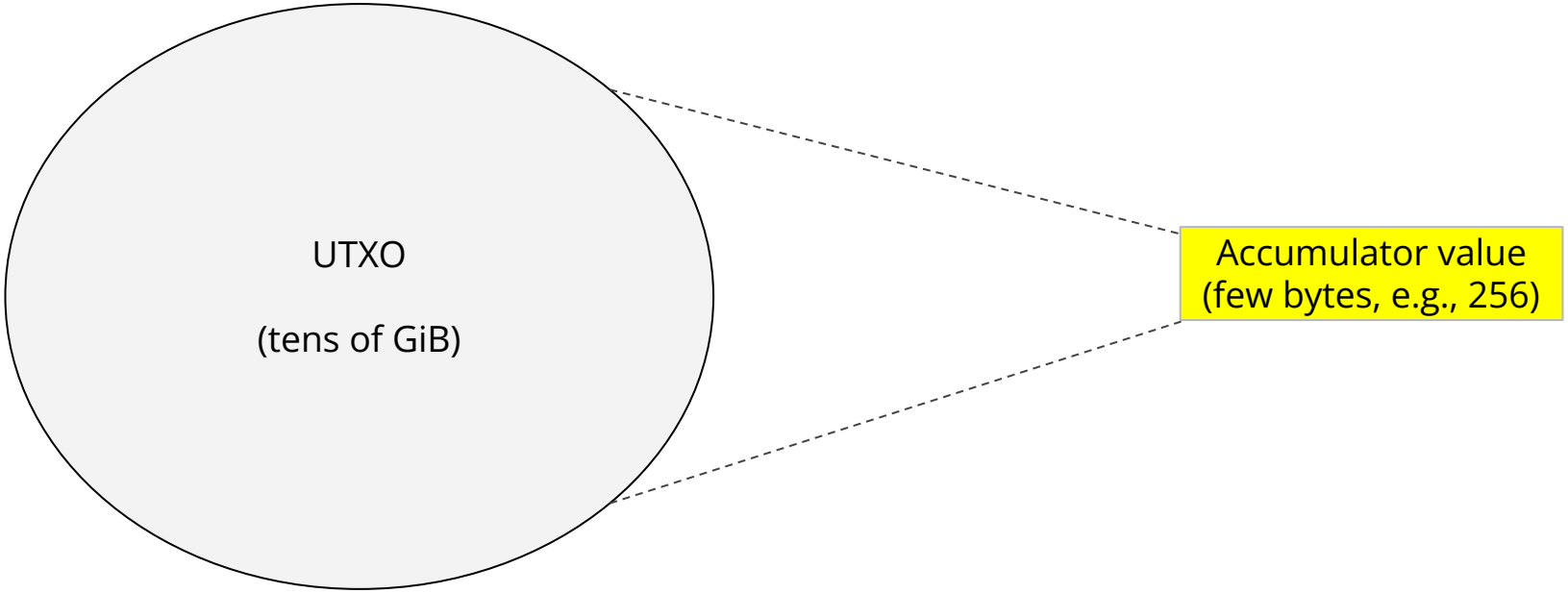
Size of serialized UTXO set (GiB)



Source: https://statoshi.info/d/000000009/unspent-transaction-output-set

Transaction validation

Key idea: Compress UTXO set using set accumulators.



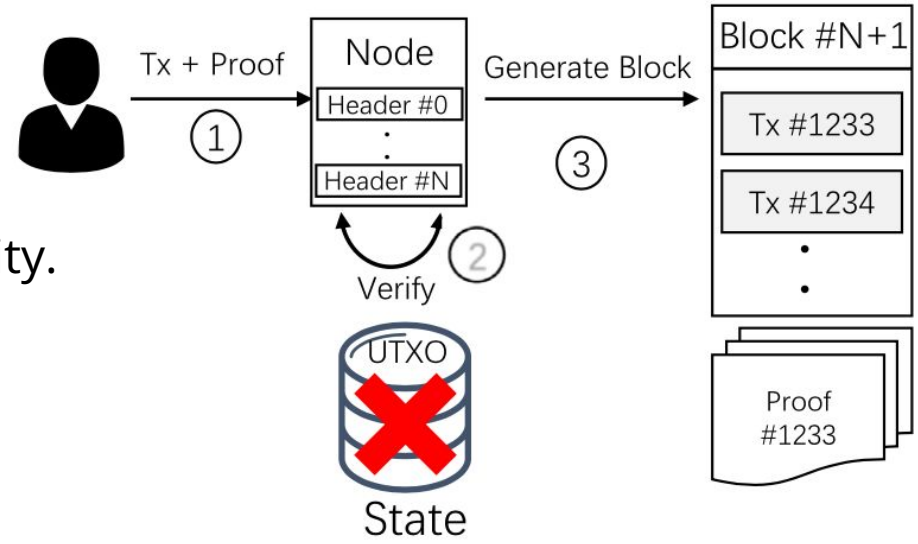
Transaction validation

Stateless design

Validator nodes verify the validity of a transaction by simply checking whether the inputs belong to the accumulator, instead of examining the entire set of unspent outputs.

=> Proof of transaction validity: proof of inputs membership in UTXO set.

Transaction validation



1) TX creators attach proof of validity.

2) Light validator node verifies TX
against the accumulator in block header.
If all inputs belong to the UTXO set, then the transaction is valid.

3) TX is added to the new block.

Source: [CW20] H. Chen, Y. Wang, “MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain”, J. Parallel Distrib. Comput. 143 (2020) 67–76.

Transaction validation

- Boneh et al. [BBF19] first proposed an RSA-based accumulator with **batch deletions** and **insertions**.

- No centralized trusted accumulator manager (perfect for untrusted environments).

- Accumulator value embedded in each block header (i.e., commitment to latest UTXO state).

Source: [BBF19] D. Boneh, B. Bünz, B. Fisch. “Batching techniques for accumulators with applications to IOPs and stateless blockchains”. In: A. Boldyreva, D. Micciancio (Eds.), Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I, in: Lecture Notes in Computer Science, vol. 11692, Springer, 2019, pp. 561–586.

Transaction validation

MiniChain protocol [CW20]

The UTXO set is split in two data structures:

1) **Spent Transaction Outputs (STXO)** set.
2) **Transaction Outputs (TXO)** set.

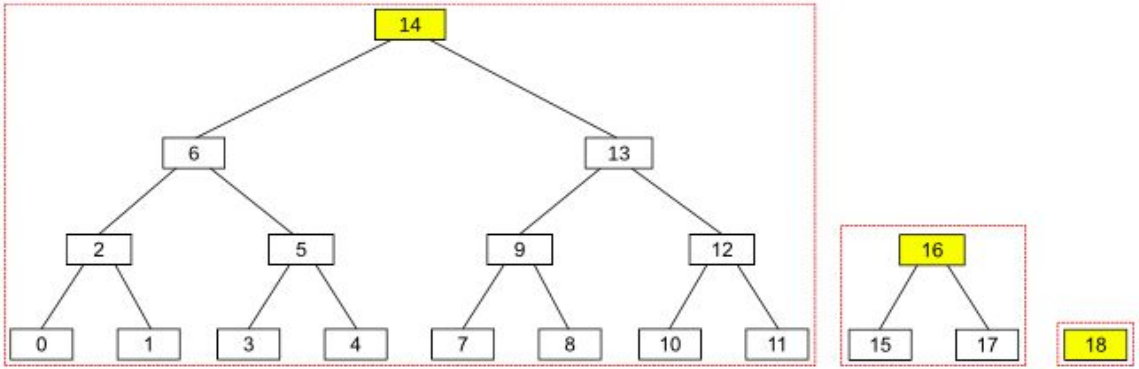
An output can be spent if and only if:

1) It belongs to TXO (=> membership proof)
2) It does not belong to STXO (=> non-membership proof)

Source: [CW20] H. Chen, Y. Wang, “MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain”, J. Parallel Distrib. Comput. 143 (2020) 67–76.

Transaction validation

TXO is managed with a **Merkle Mountain Range (MMR)** (i.e., append-only list of Merkle Trees)

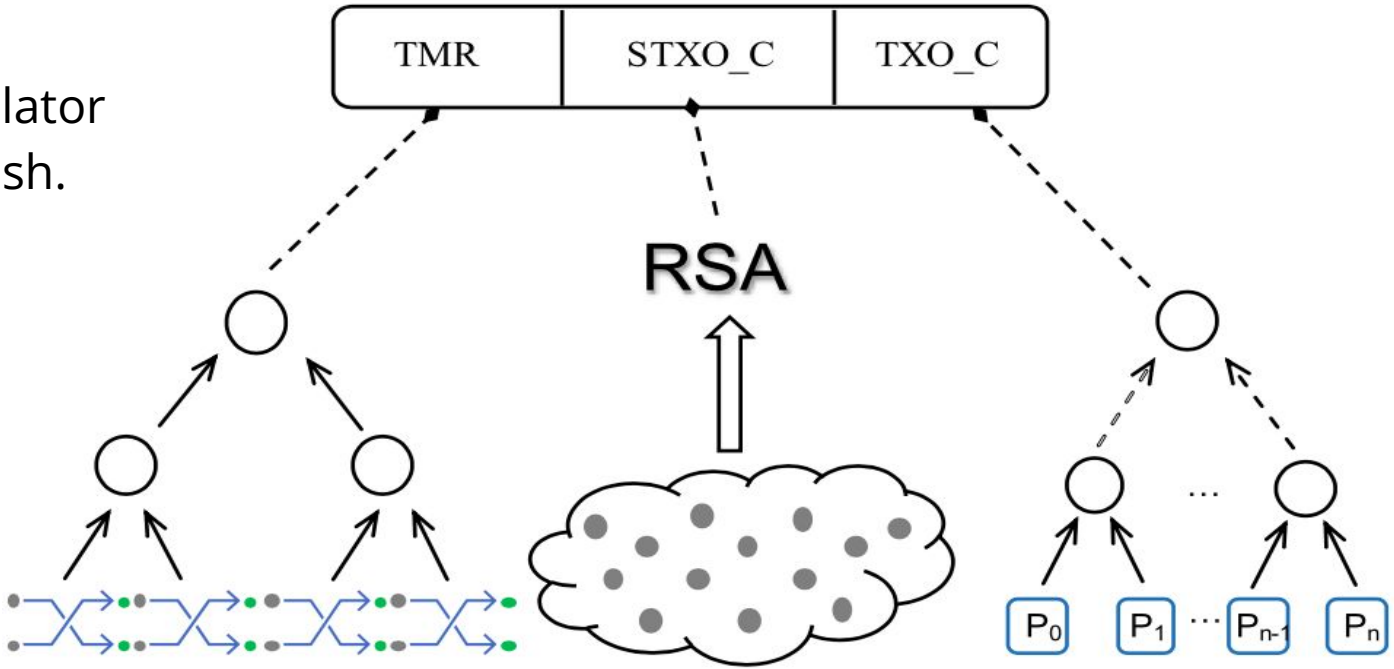


STXO is managed with a **dynamic RSA-based accumulator** (must support element insertion).

Source: [CW20] H. Chen, Y. Wang, “MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain”, J. Parallel Distrib. Comput. 143 (2020) 67–76.

Transaction validation

Each block header stores RSA accumulator and MMR peaks hash.



Source: [CW20] H. Chen, Y. Wang, “MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain”, J. Parallel Distrib. Comput. 143 (2020) 67–76.

Anonymity enhancement

Anonymity enhancement

Bitcoin is not really anonymous

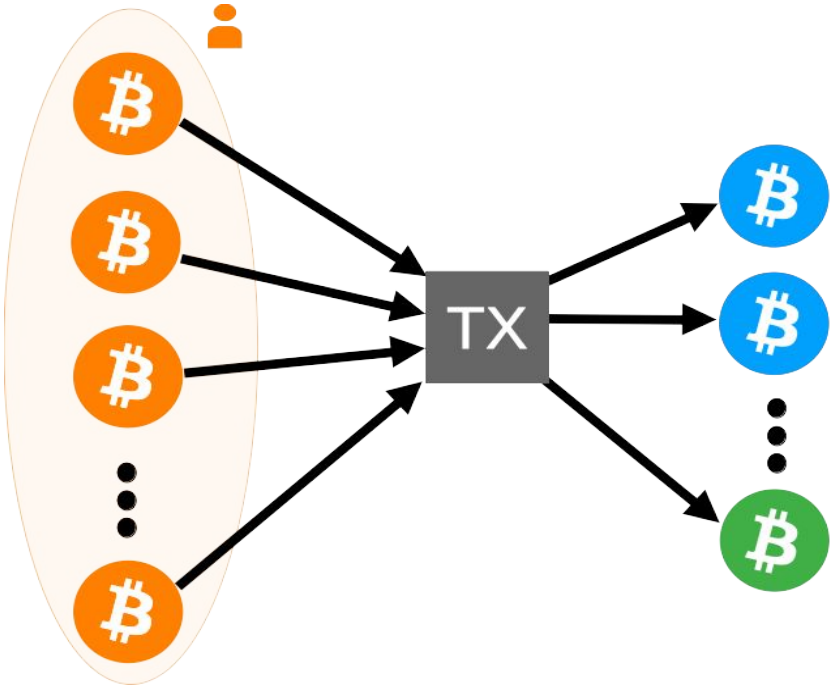
- Transactions are public
- Users control multiple addresses acting as pseudonyms
- Users can be **deanonymized** (attackers can also exploit off-network information)

Source: https://upload.bitpapa.com/uploads/blog_item/thumbnail/402/128_The_Ultimate_Guide_to_Buying_Crypto_with_Anonymous_Bitcoin.JPG

Anonymity enhancement

Multi-input heuristic rule

If two (or more) addresses are used as inputs to the same transaction, they are likely to be controlled by the same user ([Nak08]).



Anonymity enhancement

Bitcoin **laundries** mix together funds from different users to obfuscate TX history.

Drawbacks:

- Service must be trusted to return coins.
- Compromised or malicious laundry offers no anonymity.

Anonymity enhancement

Anonymity:

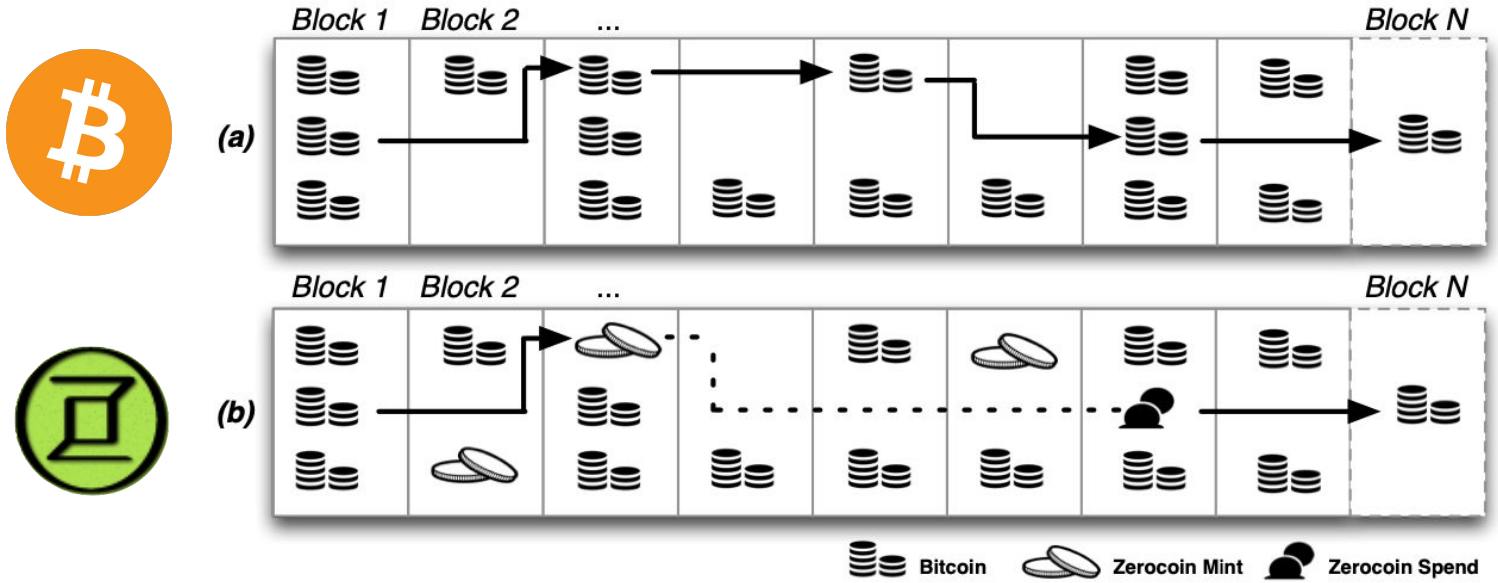
Neither the origin, destination, or amount of a payment should be revealed.

Zerocoin

- Decentralized and anonymous payment scheme.
- Born as an extension to the Bitcoin protocol (i.e., it is built on top of Bitcoin).
- Later evolved into Zerocash and Zcash.

Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

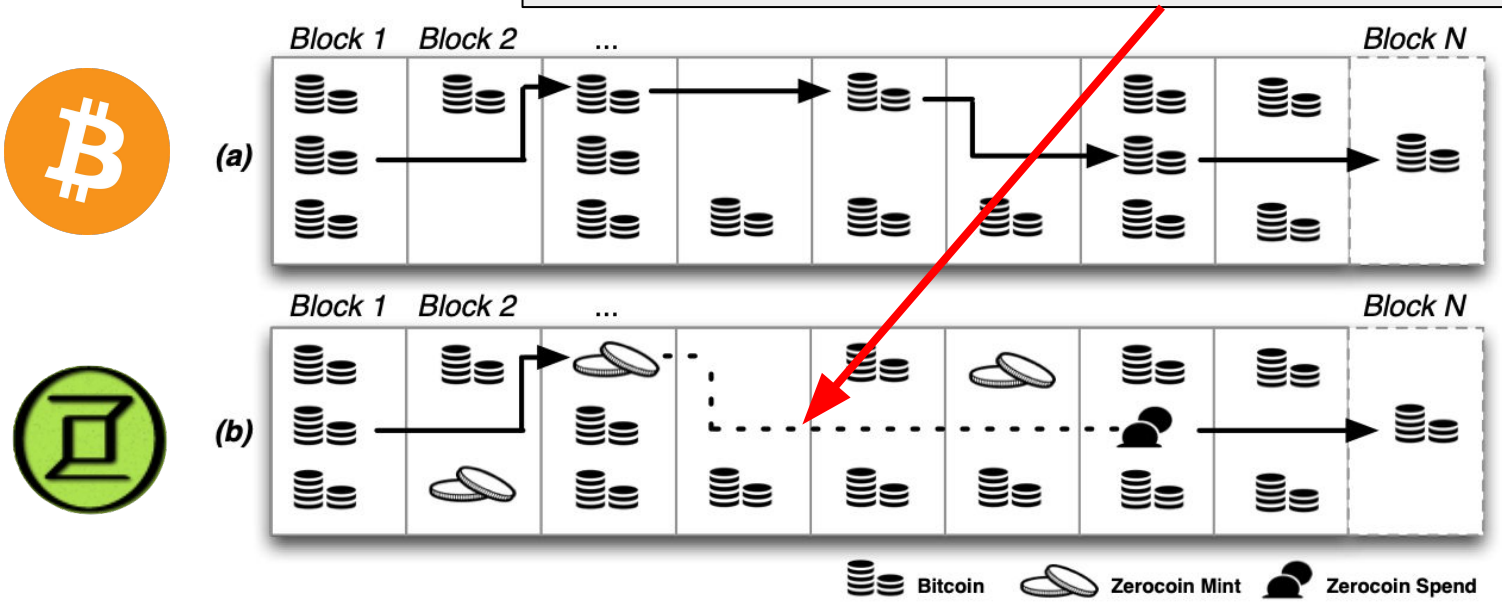
Zerocoin



Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

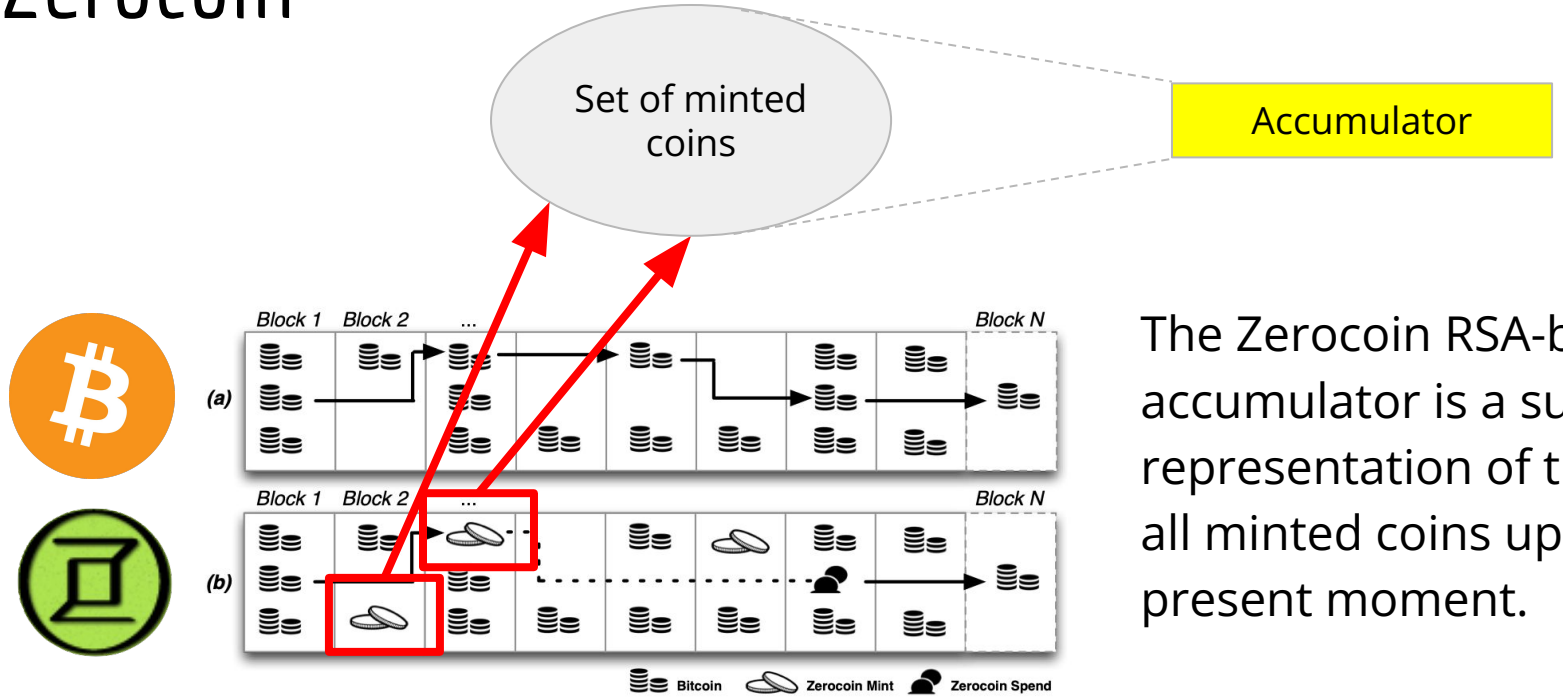
Zerocoin

True anonymity: Break link between the address that was used to create a certain coin and address used to redeem the same coin (without trusted parties).



Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

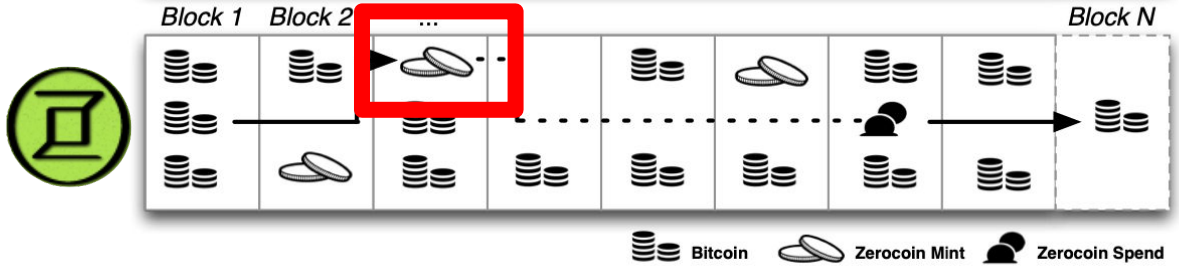
Zerocoin



The Zerocoin RSA-based accumulator is a succinct representation of the set of all minted coins up to the present moment.

Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

Zerocoin



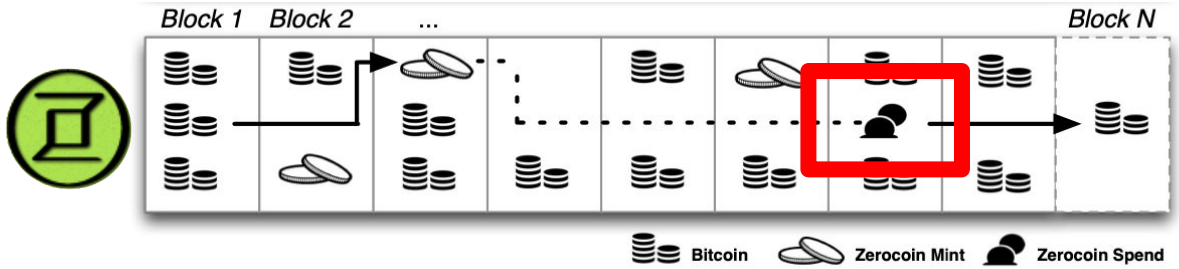
Minting a coin

1) User U generates random **serial number** S and a **nonce** r (secret);
2) C = commit(S, r) (with C prime);
3) Miners insert C into the accumulator.

Pedersen commitment scheme for step (2)
RSA-based accumulator for step (3)

Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

Zerocoin



Spending a coin

U produces two Zero-Knowledge proofs:

1) U has a commitment inserted into the accumulator.
2) U discloses S and proves the knowledge of r s.t. r and S correspond to a commitment (i.e., a coin).

Step (1) does not reveal C.

Step (2) does not reveal r.

Source: [MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

Zerocoin

Thanks to ZK proofs, the redeemed funds cannot be linked to a specific and previously minted coin.

To link coin C to the the serial number S used in the withdrawal, one must:

1) either know r (secret) ;

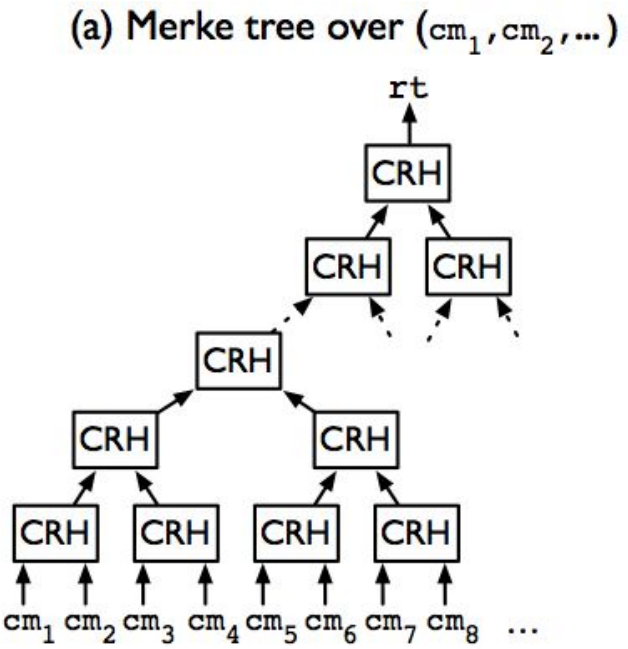
2) directly know which coin the user proved knowledge of, neither of which are revealed by the proof.

Zerocash

Zerocash

Nodes maintain a **Merkle tree** over all coin commitments seen so far.

Users demonstrate ownership of a coin commitment, via its decommitted values and a Merkle proof.



Source: [BCC+14] E. Ben Sasson et al., “Zerocash: Decentralized Anonymous Payments from Bitcoin,” 2014 IEEE Symposium on Security and Privacy. IEEE, May 2014.

State pruning

MiniLedger

- Distributed and anonymous payment system based on a shared ledger.

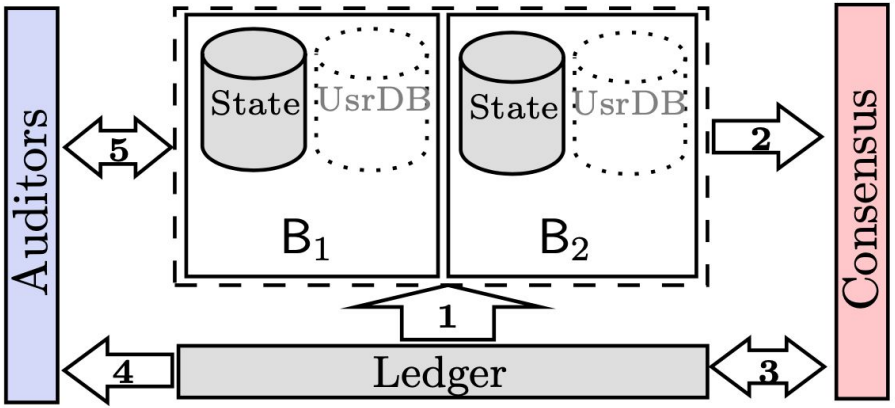
- **Privacy:** hide amount, sender and recipient of each transaction.

- **Auditability by consent:** any third-party auditor with ledger access can ask audit queries to a participant and verify the responses based on the public information on the ledger.

Source: [CB21] P. Chatzigiannis and F. Baldimtsi, “MiniLedger: Compact-Sized Anonymous and Auditable Distributed Payments,” Lecture Notes in Computer Science. Springer International Publishing, pp. 407–429, 2021.

MiniLedger

- N **banks** that transact with each other posting data on a common **ledger** L.



- L is maintained by consensus participants and is stored by all banks.

- Banks could be running consensus themselves (or outsource to any external set of consensus parties).

Source: [CB21] P. Chatzigiannis and F. Baldimtsi, “MiniLedger: Compact-Sized Anonymous and Auditable Distributed Payments,” Lecture Notes in Computer Science. Springer International Publishing, pp. 407–429, 2021.

MiniLedger

Problem: ledger storage space increases with number of recorded TXs.

Solution: prune old/unneeded TXs from the ledger.

- This could compromise auditability, as it is not possible audit data that no longer exists in the ledger.

Source: [CB21] P. Chatzigiannis and F. Baldimtsi, “MiniLedger: Compact-Sized Anonymous and Auditable Distributed Payments,” Lecture Notes in Computer Science. Springer International Publishing, pp. 407–429, 2021.

MiniLedger

(a) Ledger state before pruning, assuming B_1 had pruned before at tx_{10}.

	B_1	...	B_n
$\mathsf{tx}_1 \dots \mathsf{tx}_9$	$D_{9,1}, Q_{9,1}$		...
tx_{10}	$C_{10,1} = (c_1 = \mathsf{pk}_1^{r_{10,1}},\ c_2 = g^{v_{10,1}} h^{r_{10,1}})$ $\pi_{10,1}, \mathsf{cm}_{10,1}, Q_{10,1}$		...
tx_{11}	$C_{11,1} = (c_1 = \mathsf{pk}_1^{r_{11,1}},\ c_2 = g^{v_{11,1}} h^{r_{11,1}})$ $\pi_{11,1}, \mathsf{cm}_{11,1}, Q_{11,1}$		...

(b) Ledger state after B_1 prunes at tx_{12}. Digest $D_{11,1}$ represents $C_{10,1}, C_{11,1}$ and ciphertexts that were represented in $D_{9,1}$.

	B_1	...	B_n
$\mathsf{tx}_1 \dots \mathsf{tx}_{11}$	$D_{11,1}, Q_{11,1}$		...
tx_{12}	$C_{12,1} = (c_1 = \mathsf{pk}_1^{r_{12,1}},\ c_2 = g^{v_{12,1}} h^{r_{12,1}})$ $\pi_{12,1}, \mathsf{cm}_{12,1}, Q_{12,1}$		...

Each bank B periodically prunes all its transactions on L.

1) B publishes a digest D containing pruned TXs (keeping a local copy of such TXs).

=> D is an **RSA-based accumulator value**

2) The consensus layer verifies that D is indeed a valid digest.

3) Audit: B recovers TX from local copy and exhibits a membership proof.

Source: [CB21] P. Chatzigiannis and F. Baldimtsi, “MiniLedger: Compact-Sized Anonymous and Auditable Distributed Payments,” Lecture Notes in Computer Science. Springer International Publishing, pp. 407–429, 2021.

Identity management

Identity management

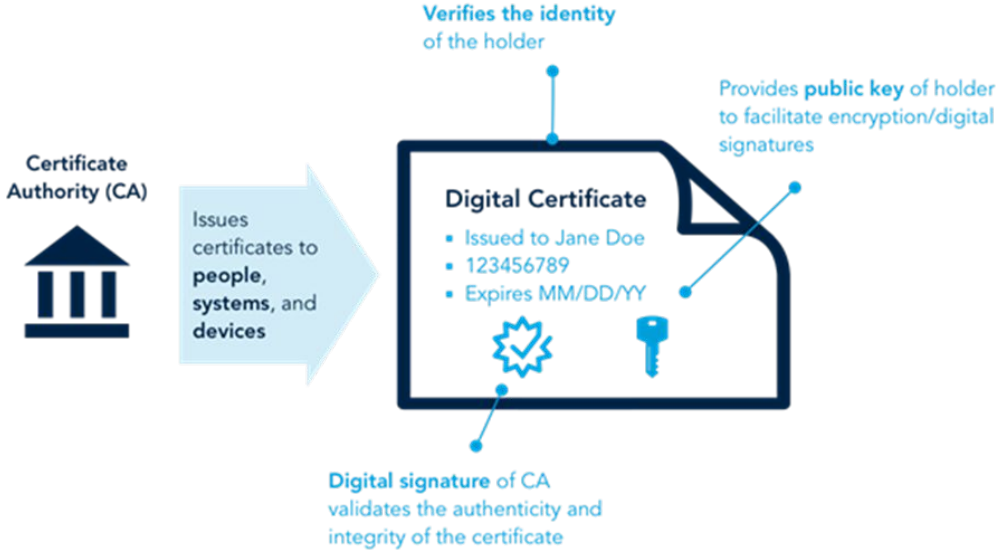
Blockchain technology plays a key role within two different paradigms.

1) **Decentralized Public Key Infrastructures (PKIs)**

2) **Self-Sovereign Identities (SSIs)**

Decentralized PKIs

Public key (PK) certificates associate identities (e.g., websites, individuals or organizations) to cryptographic keys.



Source: https://id4d.worldbank.org/sites/id4d-ms8.extcc.com/files/inline-images/18%20digital%20certificates.png

Decentralized PKIs

CAs should be trusted not to maliciously exploit all the collected information, but also to effectively protect it from external attacks.

=> Single point of failure!

Idea: Replace CAs (trusted parties) with DLT.

DLT is employed to maintain a database of (identity, public key) associations.

Decentralized PKIs

Certificates are recorded on chain => Storage overhead

How to perform identity authentication and assess certificate validity?

Not so efficient.

- As ledger size grows storage overhead is growing.

Should also be able to support certificate updates and revocation: PKI should (e.g., when public key becomes compromised)

Certcoin

Decentralized PKI protocol that uses the Namecoin blockchain as a public bulletin board.

Idea: Build a **Merkle tree** on the set of current (identity, public key) pairs.

- Assess pair validity through membership proofs.
- Complexity decreases from O(N) to O(log N) where N is the number of registered identities.

Source: [FVY14] C. Fromknecht, D. Velicanu, and S. Yakoubov. "A decentralized public key infrastructure with identity retention." Cryptology ePrint Archive, 2014.

Certcoin

- System maintains a global Merkle tree (current state is recorded on-chain).

- When a public key is created, updated or revoked the miner updates accumulator and corresponding witnesses.

- Users can check that accumulator correctly incorporates (resp. does not incorporate anymore) the newly added (resp. removed) values.

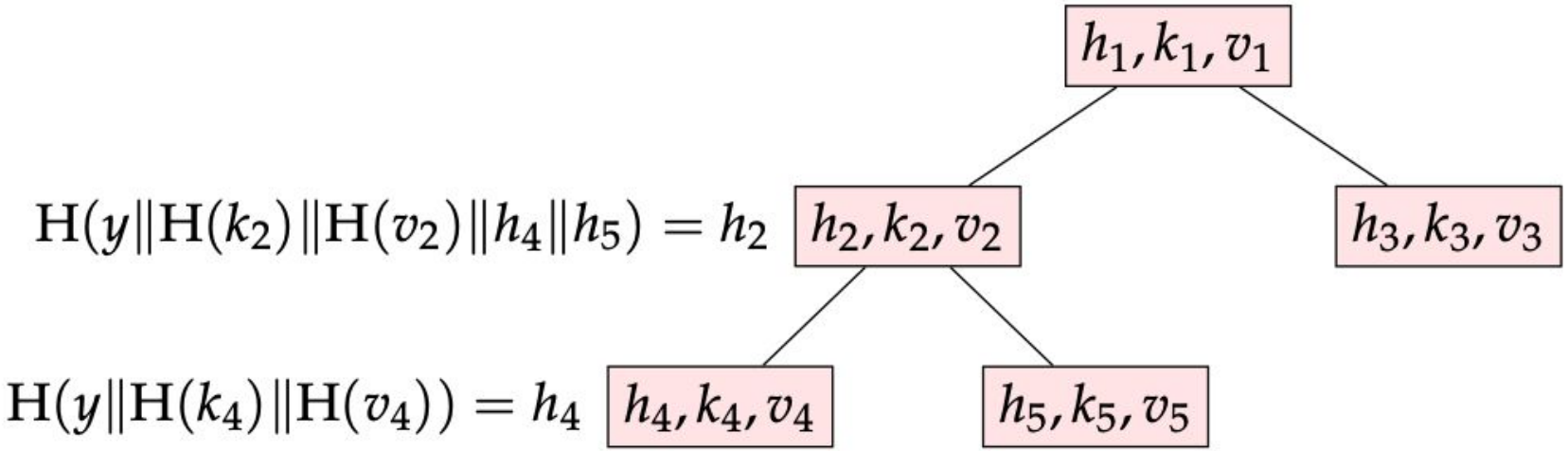
DPKIT

- Decentralized PKI for managing X.509 certificates (stored on chain).
- Ensures certificate transparency (publicly auditable, anyone can verify certificates or detect misbehavior).
- Provides a scheme for indexing certificates, proving their validity and occurred revocation.

Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

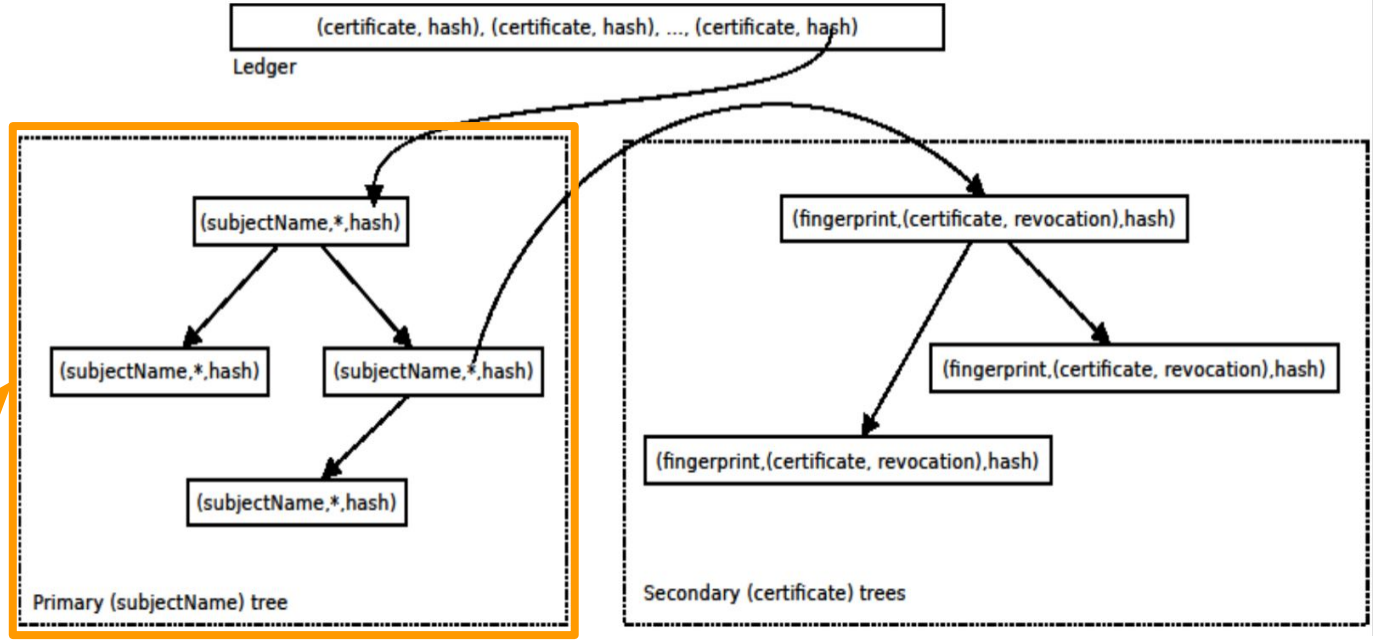
DPKIT

Merkle Binary Search tree (Merkle BST): stores key-value bindings.



Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

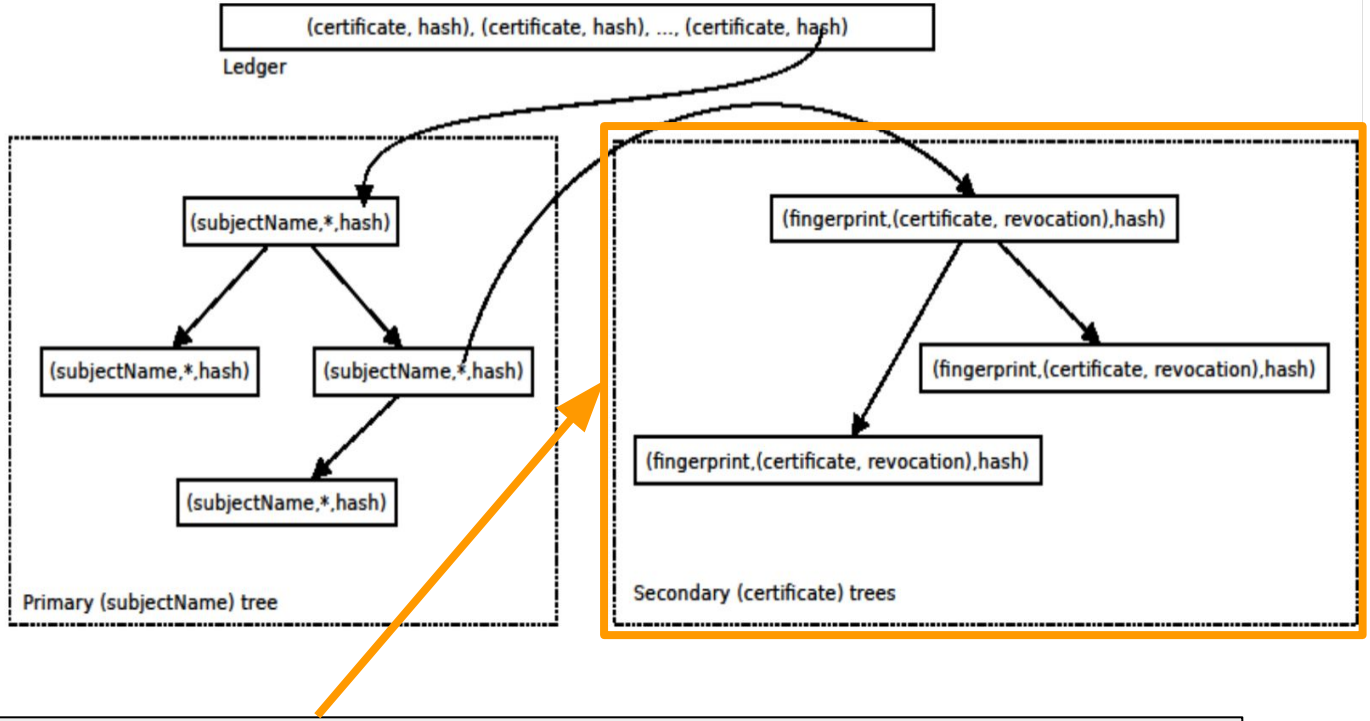
DPKIT



Primary tree: A Merkle BST which stores pointers to certificate trees, indexed by subject names.

Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

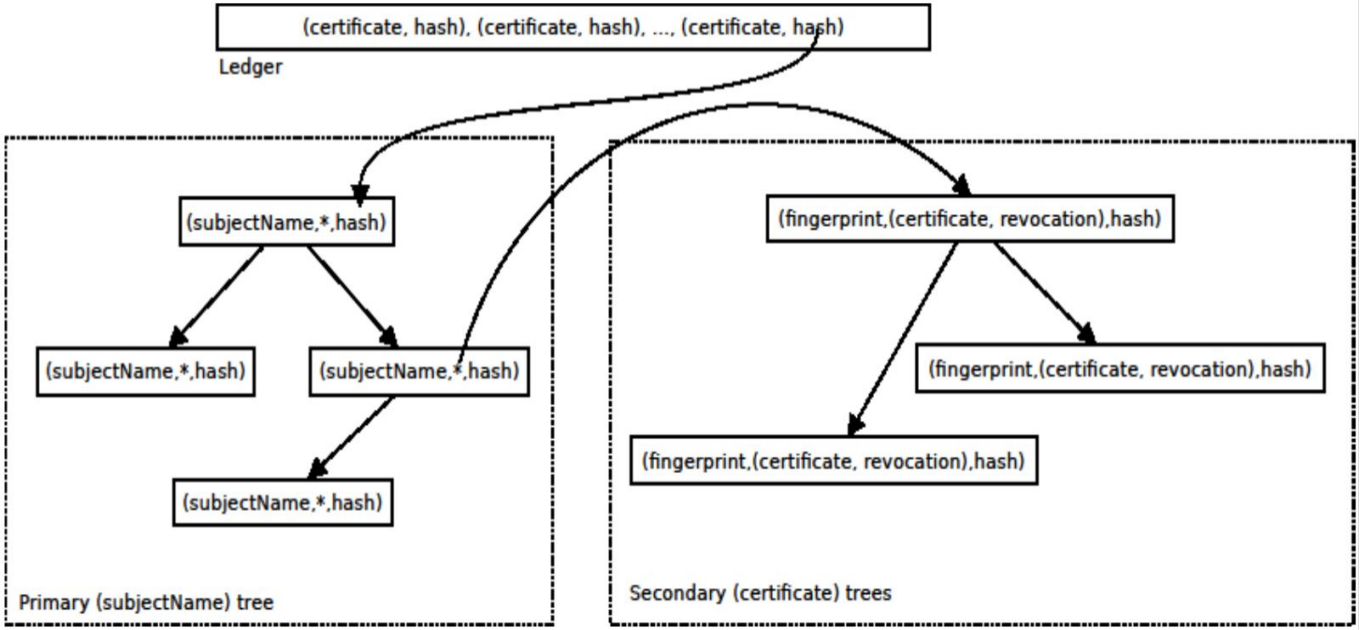
DPKIT



Secondary tree: stores (certificate, revocation) pairs with certificate fingerprints as keys

Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

DPKIT

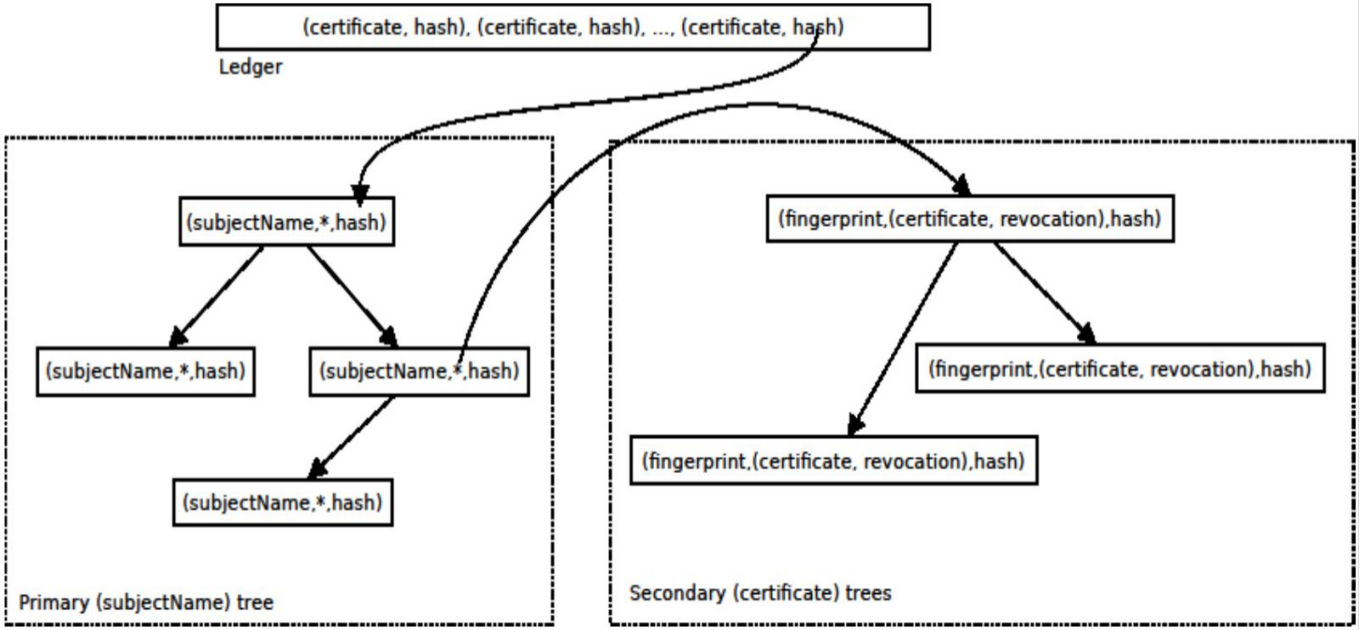


Certificate existence/validity is demonstrated through membership proofs.

1) the certificate tree in the primary BST
2) the one that identifies the certificate inside the secondary tree.

Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

DPKIT

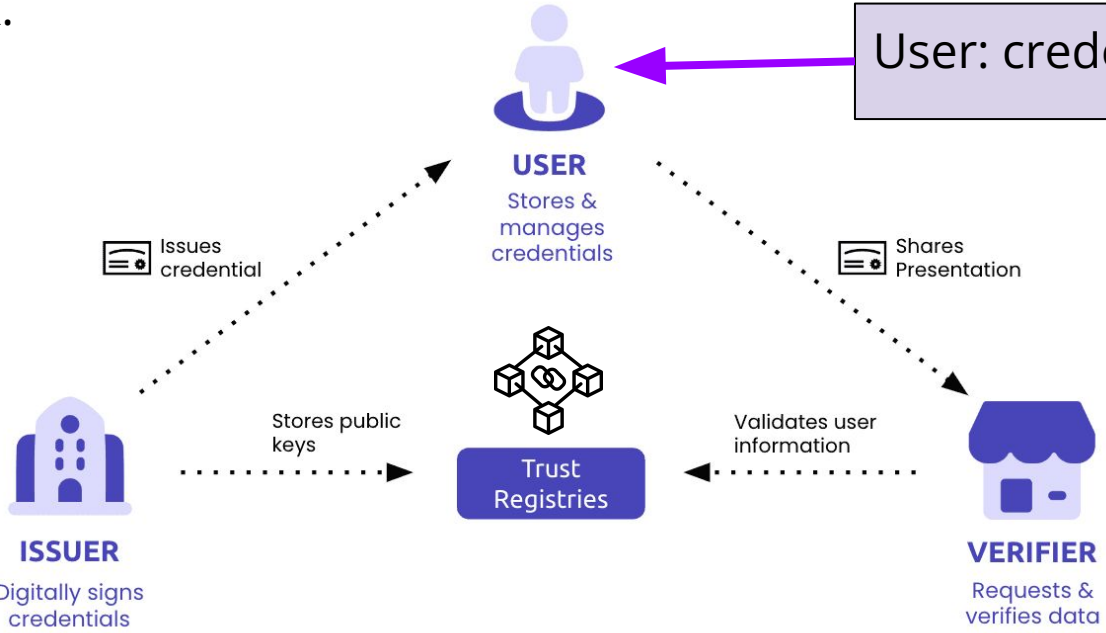


Proof of revocation is a membership proof in the secondary tree.

Source: [BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

Self-Sovereign Identity

Decentralized IDs: recognize individuals and organizations that participate to the network.



Sources: Flaticon.com

Hyperledger Indy

Open-source permissioned DLT for managing self-sovereign identities.

Records credential **schemas** and **definitions**.

- Schema: specifies credential name, version and attributes.
- Definition: signals issuer’s intention to create credentials with a given schema (and defines keys that used to sign such credentials).

Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Hyperledger Indy

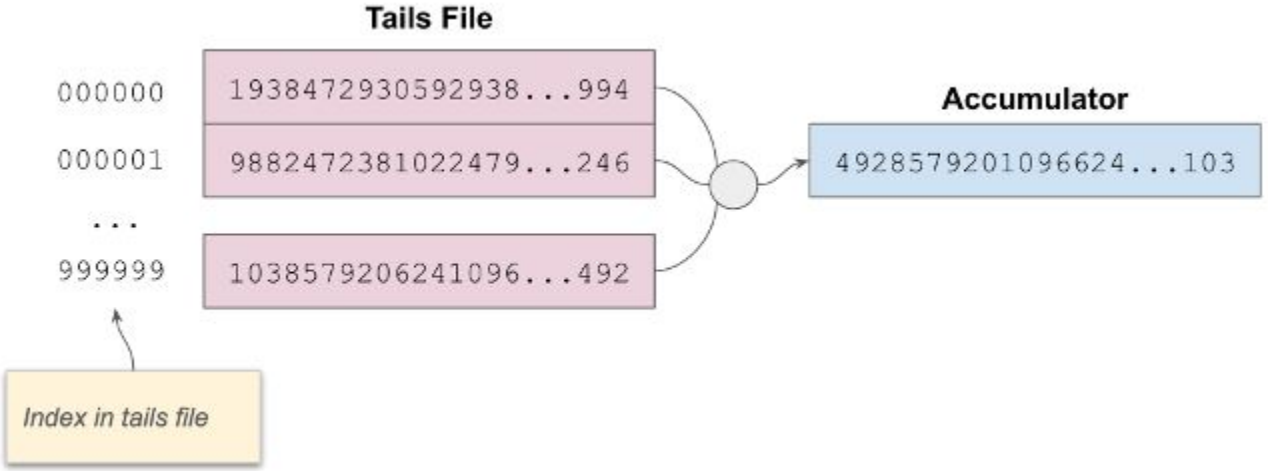
Indy supports **credential revocation**

- user frauds
- credential expiration
- issued by mistake
- guarantee the currency of credential data
- ...

Revocation is possible only if issuers publish a **revocation registry** on the ledger (i.e., metadata referencing a definition and specifying how to handle revocation).

Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Hyperledger Indy

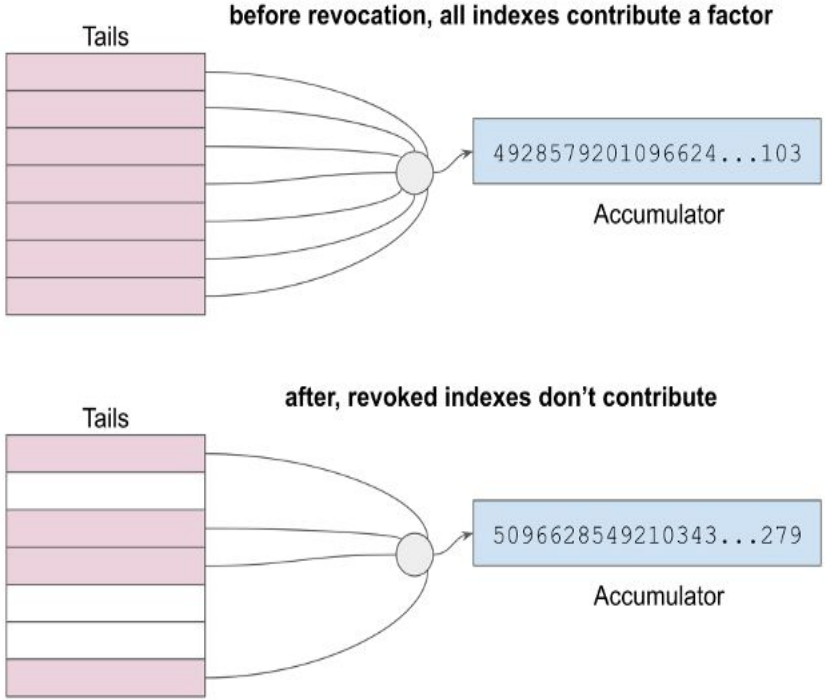


Public tails file: entries represent all possible credentials that could be issued.

Only valid (i.e., issued and not revoked) credentials contribute to the accumulator value.

Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Hyperledger Indy



Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Hyperledger Indy

When a credential is issued, the issuer transmits the following information to the holder (who will later become a prover):

1. actual **credential file**;
2. **private factor:** the index corresponding to the credential in the tails file;
3. **witness**: product of all other tails file entries.

Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Hyperledger Indy

Primary proof. Proving credential properties:
- “What is your birthdate?”
- “Please disclose your address”

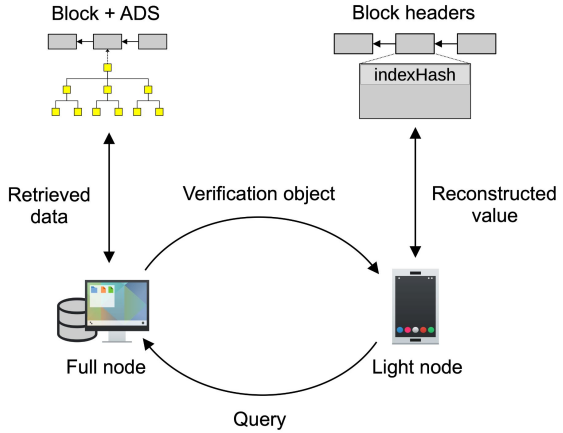
Proof of non-revocation. The prover must demonstrate that the credentials behind the primary proof have not been revoked.
=> Derive accumulator value from private factor and witness (just one multiplication).

Witness delta: mechanism for updating provers’ witnesses after accumulator has changed.

Source: https://hyperledger-indy.readthedocs.io/projects/sdk/en/latest/docs/concepts/revocation/cred-revocation.html

Discussion

Query authentication



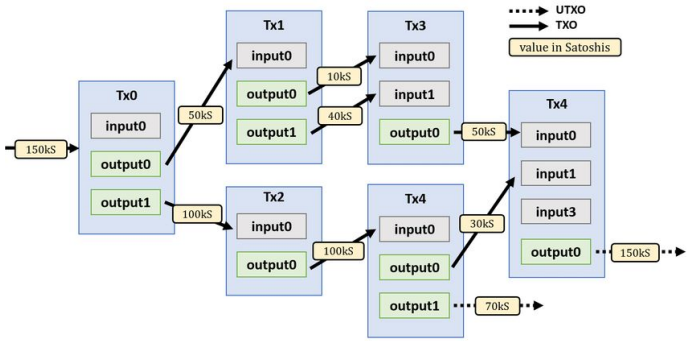
Pros:

- Enable authenticated queries.
- Reduce verification object size.
- Compress information included in multiple blocks.

Cons:

- Block structure modification required.
- A third party for trusted setup phase might be required.
- Arithmetic-based solutions might be slower than hashing operations.

Stateless transaction validation



Pros:

- Mitigate the impact of UTXO set growth (especially for resource-constrained nodes).

Cons:

- Increase network communication overhead (due to membership proofs attached to TX payloads).

  => Proof aggregation (i.e., batching) suggested as possible solution.

Anonymity enhancement

Pros:

- Decouple coin minting operations from spending transactions.
- Compress set of unspent coins (as in stateless TX validation).
- ...

Cons:

- Increasing burden on miners (because of proof validation).
- ...

Identity management

Pros:

- Reduce ledger size (in PKIs).
- Improve certificate search and validation (in PKIs).
- Improve credential revocation efficiency (in SSI).
- ...

Cons:

- Presence of a trusted party may not be desirable to achieve full decentralization (but may be fine in permissioned systems).
- ...

Set accumulators

- Compress large sets into a single digest, while ensuring the authenticity of operations performed on those sets.

- Perfect for scenarios where:
  - data are continuously appended to the blockchain;
  - the need for data retrieval collides with the difficulty of reading sequentially from the data structure.

- Common goals:
  - improving overall system scalability;
  - making the network more accessible to resource-constrained nodes.
  - ...

Table 6
A summary of the presented applications of set accumulators to blockchain systems. For each proposal, we indicate the use case, the accumulator type (specifying whether it is primarily based on cryptographic hash functions) and provide a brief description.

Use case	Proposal	Ref.	Accumulator type	Hash-based	Description
Query authentication	Bitcoin SPV	[1]	MHT	Yes	Transaction verification
	vChain	[45]	Bilinear	No	Boolean range query verification
	GCA^2-tree	[46]	ESA	No	Aggregate range query verification
	GEM^2-tree	[47]	Merkle B-tree	Yes	Range query verification
	Loporchio et al.	[48]	Merkle R-tree	Yes	Range query verification
	FalconDB	[49]	Bilinear	No	Collaborative database
Stateless transactions validation	Boneh et al.	[55]	RSA-based	No	UTXO and account-based validation
	EDRAX	[56]	Sparse MHT [41]	Yes	UTXO and account-based validation
	MiniChain	[57]	MMR, RSA-based	Yes	UTXO-based validation
	Bailey, Sankagiri	[58]	MHT	Yes	UTXO-based validation
	Utreexo	[59]	MHT forest	Yes	UTXO-based validation
	SecurePrune	[60]	RSA-based	No	Block pruning scheme
Anonymity enhancement	Zerocoin	[65]	RSA-based	No	Anonymity protocol
	Zerocash	[66]	MHT	Yes	Anonymity protocol
	Garman et al.	[67]	MHT	Yes	Anonymous payment scheme
	MiniLedger	[68]	RSA-based, MHT	Yes	Anonymous payment scheme
Identity management	Certcoin	[74]	Dynamic MHT [77]	Yes	Decentralized PKI
	DPKIT	[78]	Merkle BST	No	Decentralized PKI
	Feng et al.	[79]	RSA-based [80]	No	Decentralized PKI
	Hyperledger Indy	[81]	Bilinear-based [82]	No	Self-sovereign identity management

Source: [LBDR23] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa and L. Ricci. "A survey of set accumulators for blockchain systems." Computer Science Review 49 (2023): 100570.

Thank you
for your attention

Image credits: Freepik.com

References

[BCC+14] E. Ben Sasson et al., “Zerocash: Decentralized Anonymous Payments from Bitcoin,” 2014 IEEE Symposium on Security and Privacy. IEEE, May 2014.

[BP97] N. Barić and B. Pfitzmann, “Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees,” Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 480–494, 1997.

[BHMS21] X. Boyen, U. Herath, M. McKague, and D. Stebila, “Associative Blockchain for Decentralized PKI Transparency,” Cryptography, vol. 5, no. 2, p. 14, May 2021.

[CB21] P. Chatzigiannis and F. Baldimtsi, “MiniLedger: Compact-Sized Anonymous and Auditable Distributed Payments,” Lecture Notes in Computer Science. Springer International Publishing, pp. 407–429, 2021.

[CW20] H. Chen, Y. Wang, “MiniChain: A lightweight protocol to combat the UTXO growth in public blockchain”, J. Parallel Distrib. Comput. 143 (2020) 67–76.

[FVY14] C. Fromknecht, D. Velicanu, and S. Yakoubov. "A decentralized public key infrastructure with identity retention." Cryptology ePrint Archive. 2014.

[LBDR23] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci. "A survey of set accumulators for blockchain systems." Computer Science Review 49 (2023): 100570.

References

[LBDR25] M. Loporchio, A. Bernasconi, D. Di Francesco Maesa, and L. Ricci, “Skip index: Supporting efficient inter-block queries and query authentication on the blockchain,” Future Generation Computer Systems, vol. 164, p. 107556, Mar. 2025.

[MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin, “Zerocoin: Anonymous Distributed E-Cash from Bitcoin,” 2013 IEEE Symposium on Security and Privacy. IEEE, pp. 397–411, May 2013.

[Nak08] S. Nakamoto. “Bitcoin: A peer-to-peer electronic cash system.” 2008.

[Ngu05] L. Nguyen, “Accumulators from Bilinear Pairings and Applications,” Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 275–292, 2005.

[XZX19] C. Xu, C. Zhang, and J. Xu, “vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases,” Proceedings of the 2019 International Conference on Management of Data. ACM, Jun. 25, 2019.

[ZKP17] Y. Zhang, J. Katz, and C. Papamanthou, “An Expressive (Zero-Knowledge) Set Accumulator,” 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, pp. 158–173, Apr. 2017.