BlockOPE: Efficient Order-Preserving Encryption for Permissioned Blockchain

Zhihao Chen, Qingqing Li, Xiaodong Qi, Zhao Zhang, Cheqing Jin, Aoying Zhou

East China Normal University
Presented at ICDE 2022

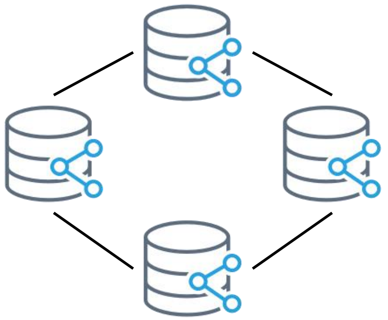
Background

- Blockchain, as a **tamper-evidence, traceable, multi-party jointly maintained** distributed ledger, can tackle trust problems among mutually distrusting parties.

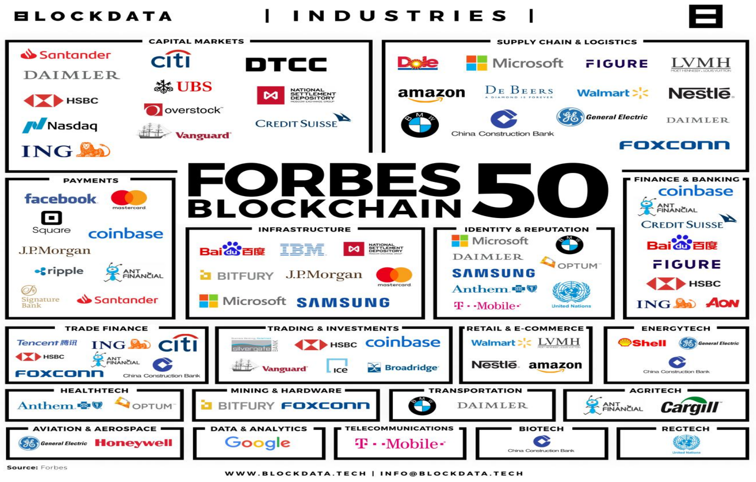
- Permissioned blockchain inherits the above advantages, which is evolving into a platform for **data management and sharing** in many collaborative scenarios due to its higher throughput.

Application Scenarios:

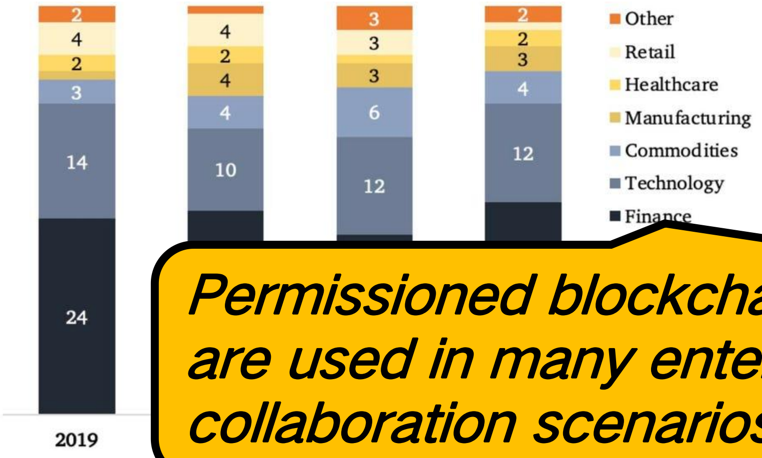
Cryptocurrency



Data Collaboration



Forbes Blockchain 50 (2020)



Permissioned blockchains are used in many enterprise collaboration scenarios

Forbes Blockchain 50 Categories (2019 ~ 2022)

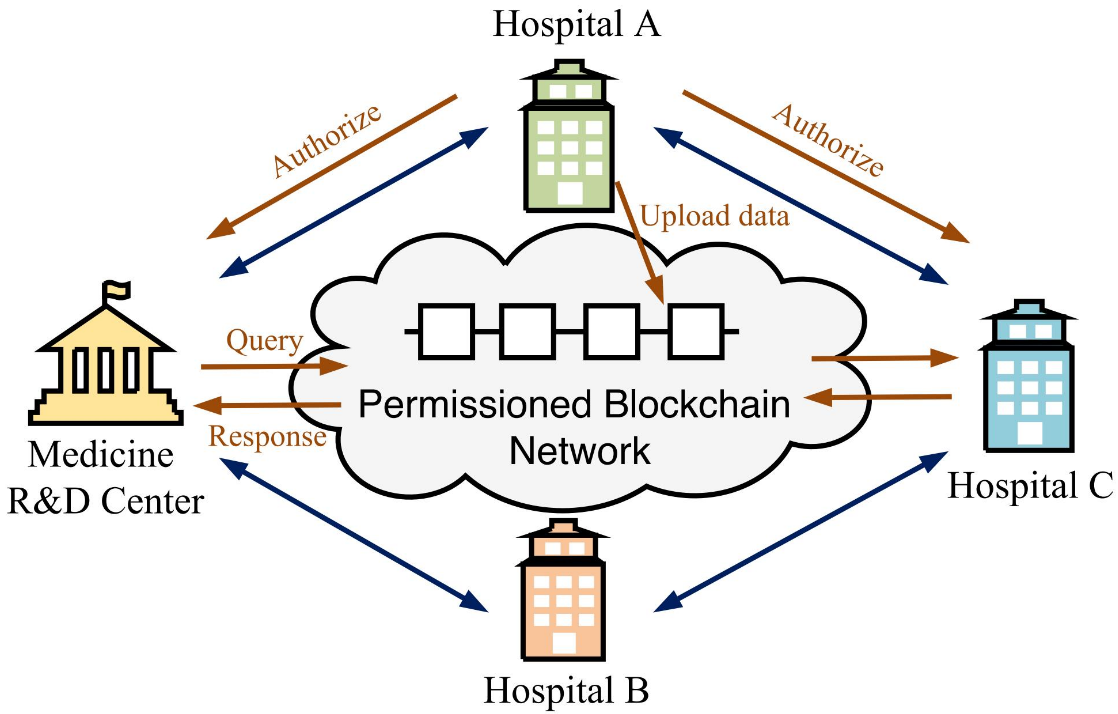
https://www.forbes.com/sites/javierpaz/2022/02/08/2022-forbes-blockchain-50-a-closer-look/

Background

- However, the data contents stored on blockchain are **plaintexts**, which can be exposed to the malicious nodes under the Byzantine environment, leading to **privacy breaches**.

*require **privacy protection** while **allowing efficient queries** to support blockchain-based data-sharing scenarios*

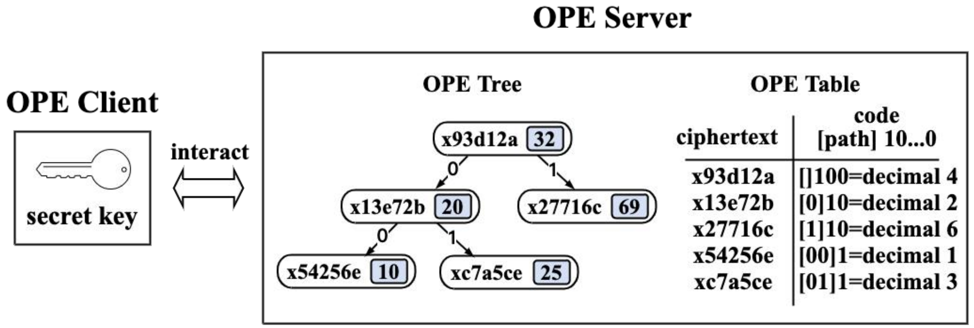
- Order-Preserving Encryption(OPE) is an encryption scheme that balances **data privacy** and **data usability**.



Blockchain-Based Collaborative Medicine R&D

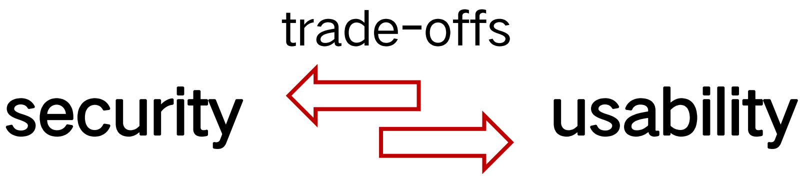
Related Works

- OPE schemes: widely used in outsourced databases and encrypted databases, allow the untrusted node to perform **order comparison over ciphertexts**.

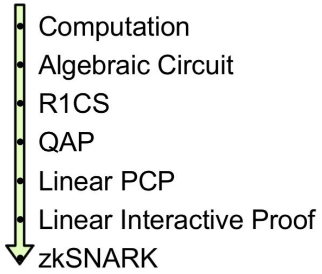


OPE Scheme

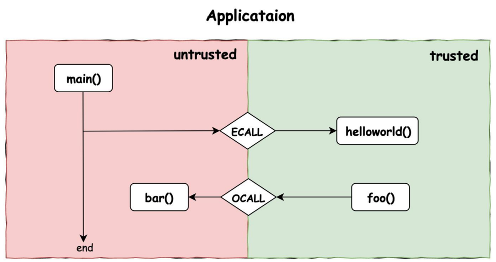
- Other cryptographic schemes: Searchable Encryption scheme (**SE**), Order Revealing Encryption scheme (**ORE**), Fully Homomorphic Encryption (**HE**) ...



- Blockchain-related: researches try to protect data confidentiality to blockchain, including Homomorphic encryption (**HE**), Zero Knowledge Proof (**ZKP**) and Trusted Execution Environment (**TEE**).



ZKP



TEE (e.g. SGX)

Related Works

- *constrained by limited use cases (the single client and single server model) and inherent performance limitations*

over

- *can not achieve higher security and lower searching complexity simultaneously than OPE*

ption

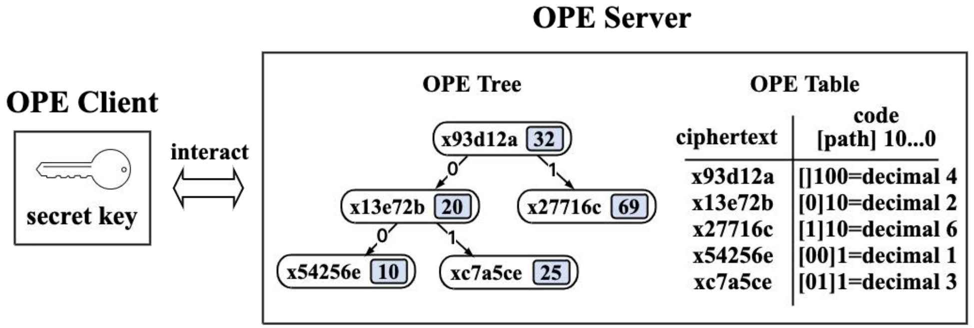
- *huge computation and memory costs / restricted by the hardware*

lata

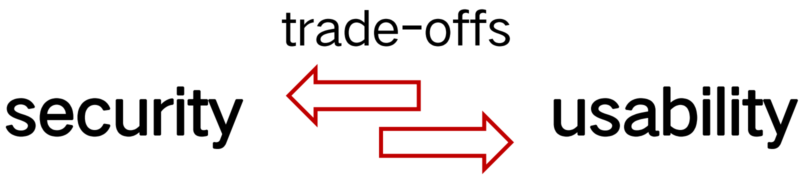
phic

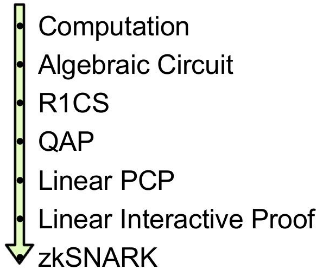
and

Trusted Execution Environment (TEE).

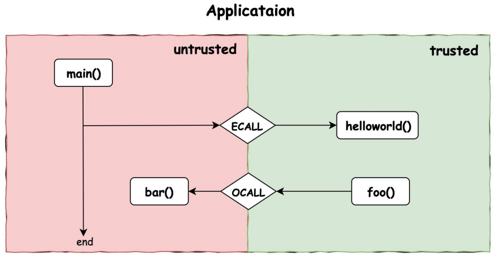


OPE Scheme





ZKP

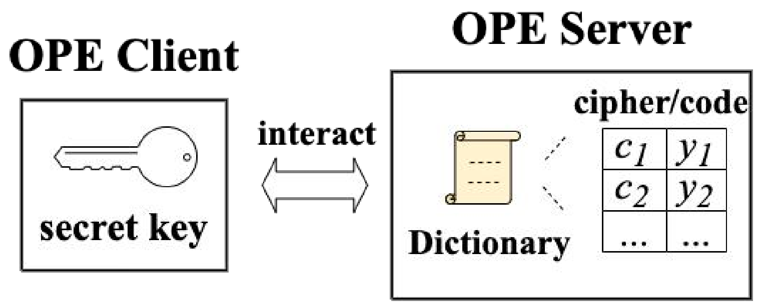


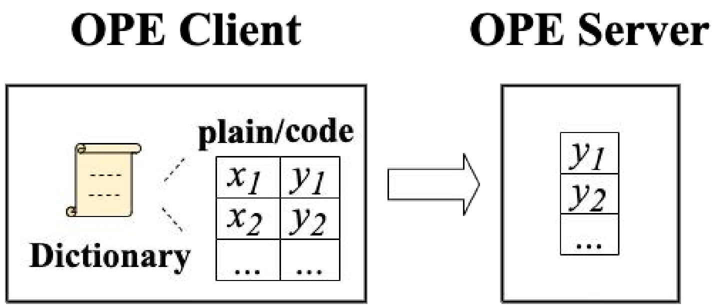
TEE (e.g. SGX)

Challenges

- Conventional OPE schemes only consider the cloud environments (single client and single server model), which are unfeasible for fully-replicated blockchain systems.
- The inherent serial encoding processing of OPE will burden the blockchain throughput, especially for permissioned blockchain where performance is one of the main concerns.
- Provide efficient queries over ciphertexts while tackling the above two points.

the dictionary is maintained
either on the client or
on the honest but curious server

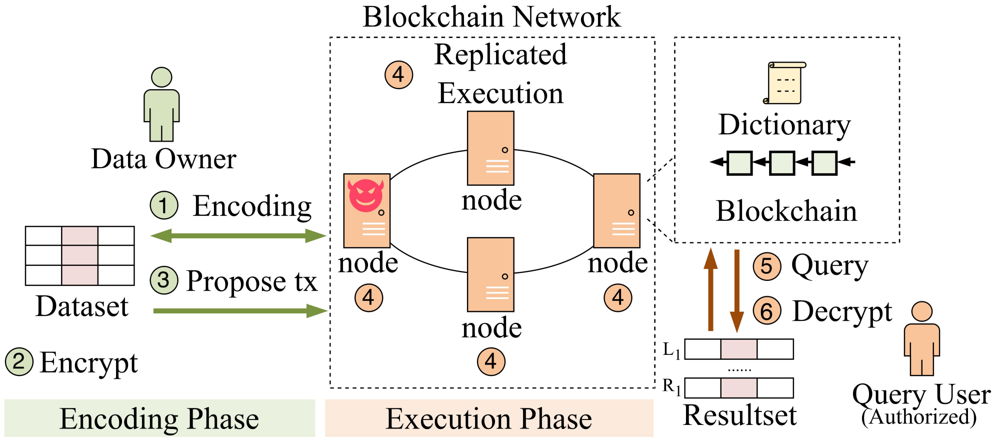




Conventional OPE Schemes

Contributions

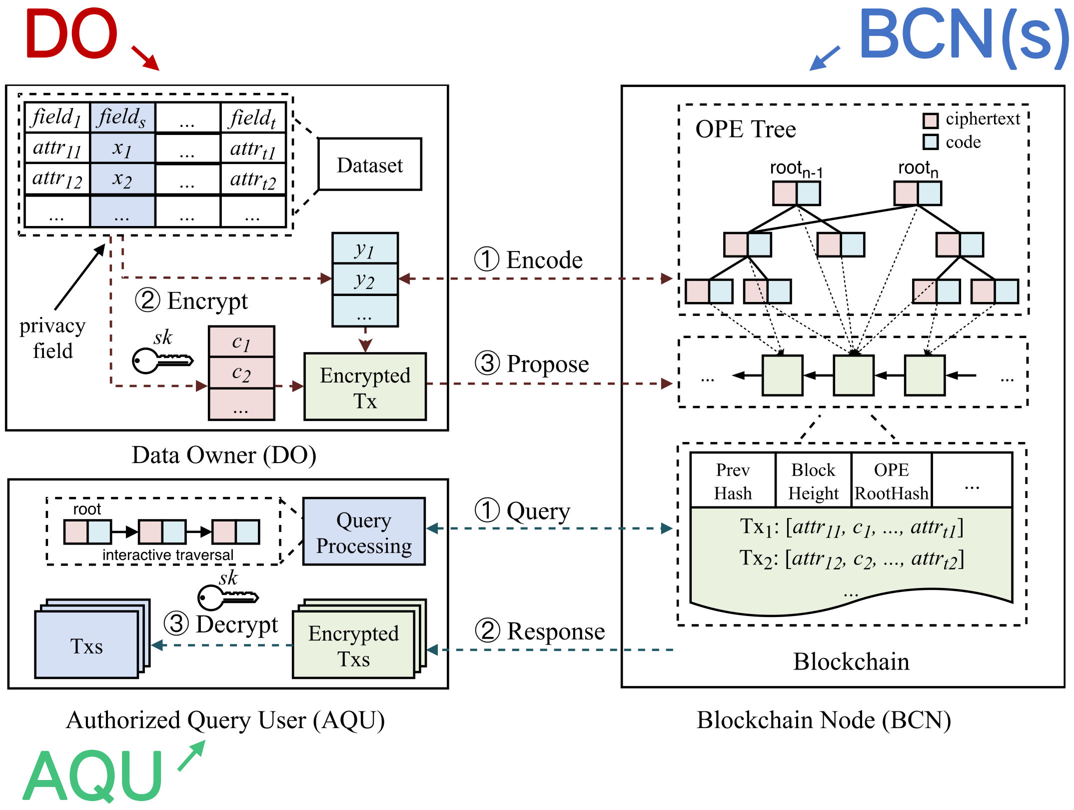
- BlockOPE: an efficient OPE scheme towards permissioned blockchain
  - Tech#1: Propose a novel two-phase scheme (encoding and execution phase)
  - Tech#2: Optimize the dictionary maintenance with parallel processing and conflict-reducing design
  - Tech#3: Leverage an adaptive cache to improve the query performance
- A prototype BlockOPE integrated with a permissioned mini-blockchain
- Experimental evaluations and theoretical analysis of the prototype
  - Verified proposed scheme is feasible and practical



Overview of BlockOPE

Threat model

- DO (Data Owner):
  - DO owns the data to be encrypted, thus is trusted.
  - DO uploads ciphertexts to the blockchain to store the data and claim the data authority.
- AQU (Authorized Query User):
  - AQU acquires the shared data by querying blockchain for ciphertexts provided by DO.
  - AQU and DO are mutually trusting while both of them share no trust with BCNs.
- BCN (Blockchain Node):
  - All BCNs cooperate to host the blockchain.
  - At most f BCN(s) can be malicious or Byzantine out of n BCNs such that n ⩾ 3f + 1.

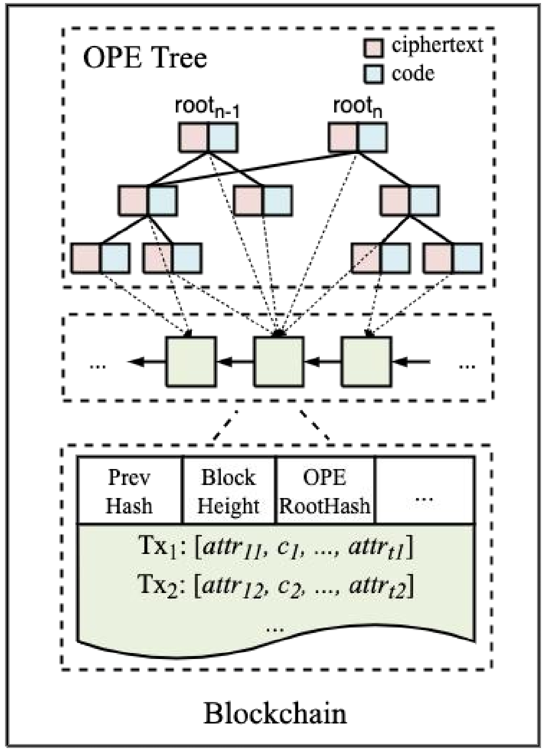


DO, AQU and BCNs of BlockOPE

Methodology

- OPE Tree:
  - The OPE Tree is a data structure of a collection of vertices in the form of **⟨c, y⟩**. It can be seen as an index structure built on blockchain transactions, meaning each vertex links to its corresponding transaction.

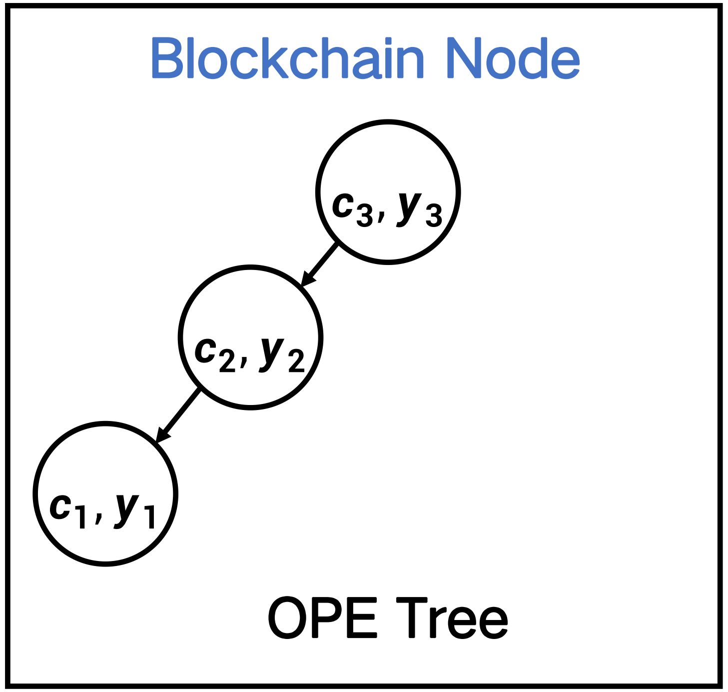
OPE Tree



Blockchain Node (BCN)

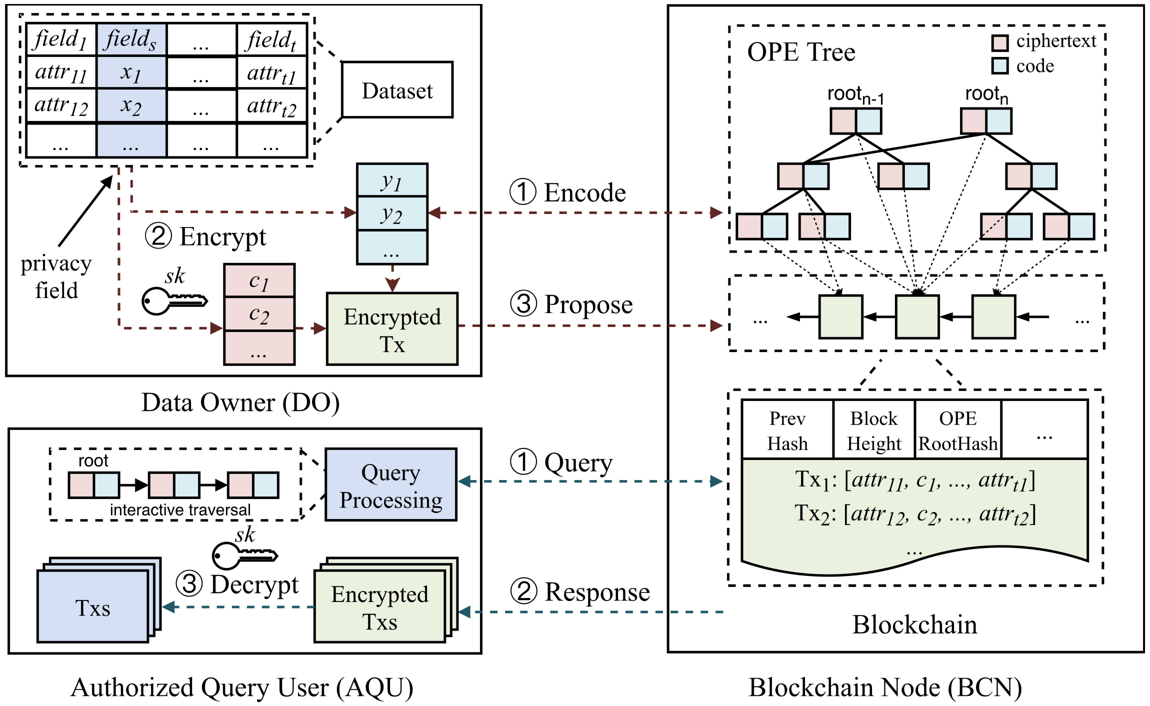
Methodology

- OPE Tree:
  - The OPE Tree is a data structure of a collection of vertices in the form of **⟨c, y⟩**. It can be seen as an index structure built on blockchain transactions, meaning each vertex links to its corresponding transaction.
  - *Definition:* The OPE Tree *T* is a binary search tree consisting of a set of vertices, **∀ vertex *v* = ⟨c,y⟩ ∈ *T*, c = Encrypt(x), y = Encode(x)**, where x is the plaintext of private data.



Methodology

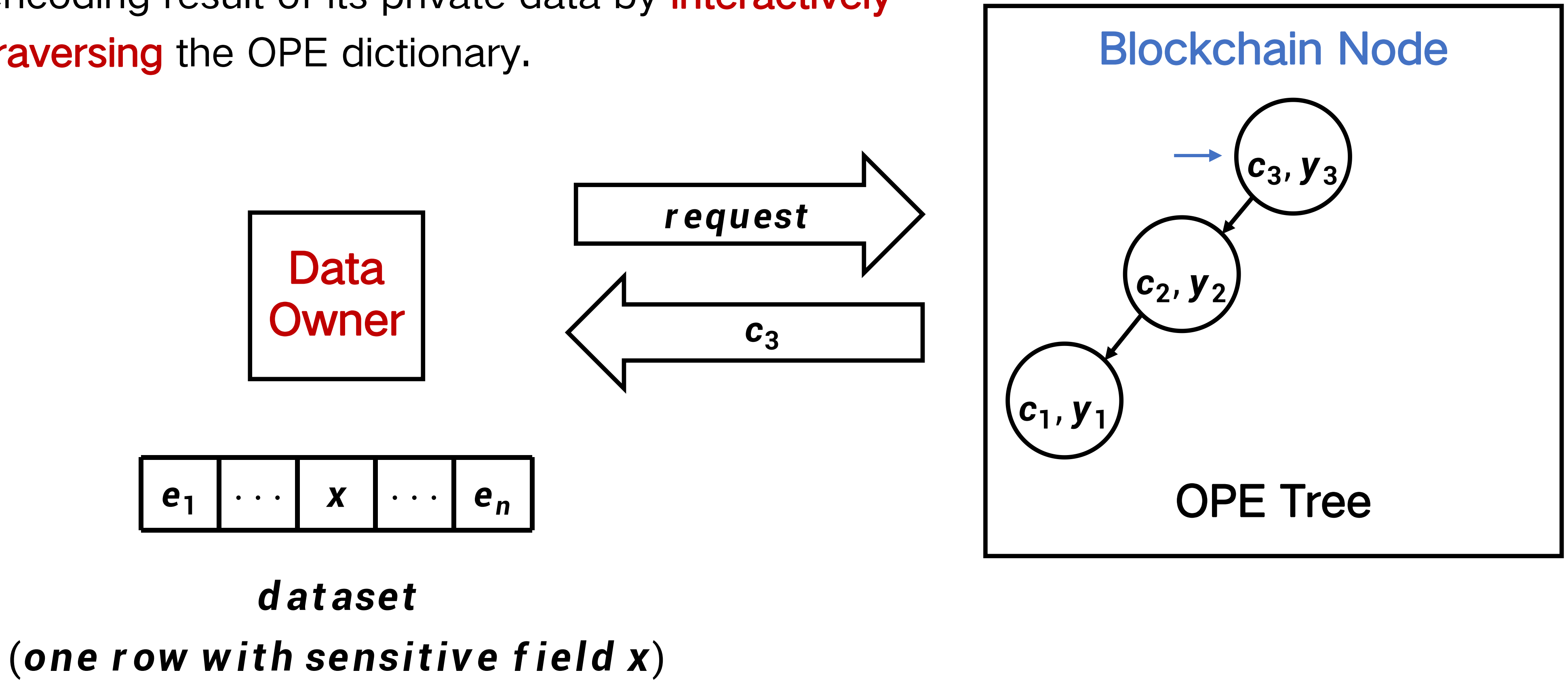
- Serial Two-phase BlockOPE:
  - Encoding Phase
    - DO interacts with BCN(s) to get the encoding result of its private data.
  - Execution Phase
    - DO sends the order-preserved codes to BCN(s) as a transaction.
    - BCNs execute agreed transactions to update the OPE Tree.



Processing of BlockOPE

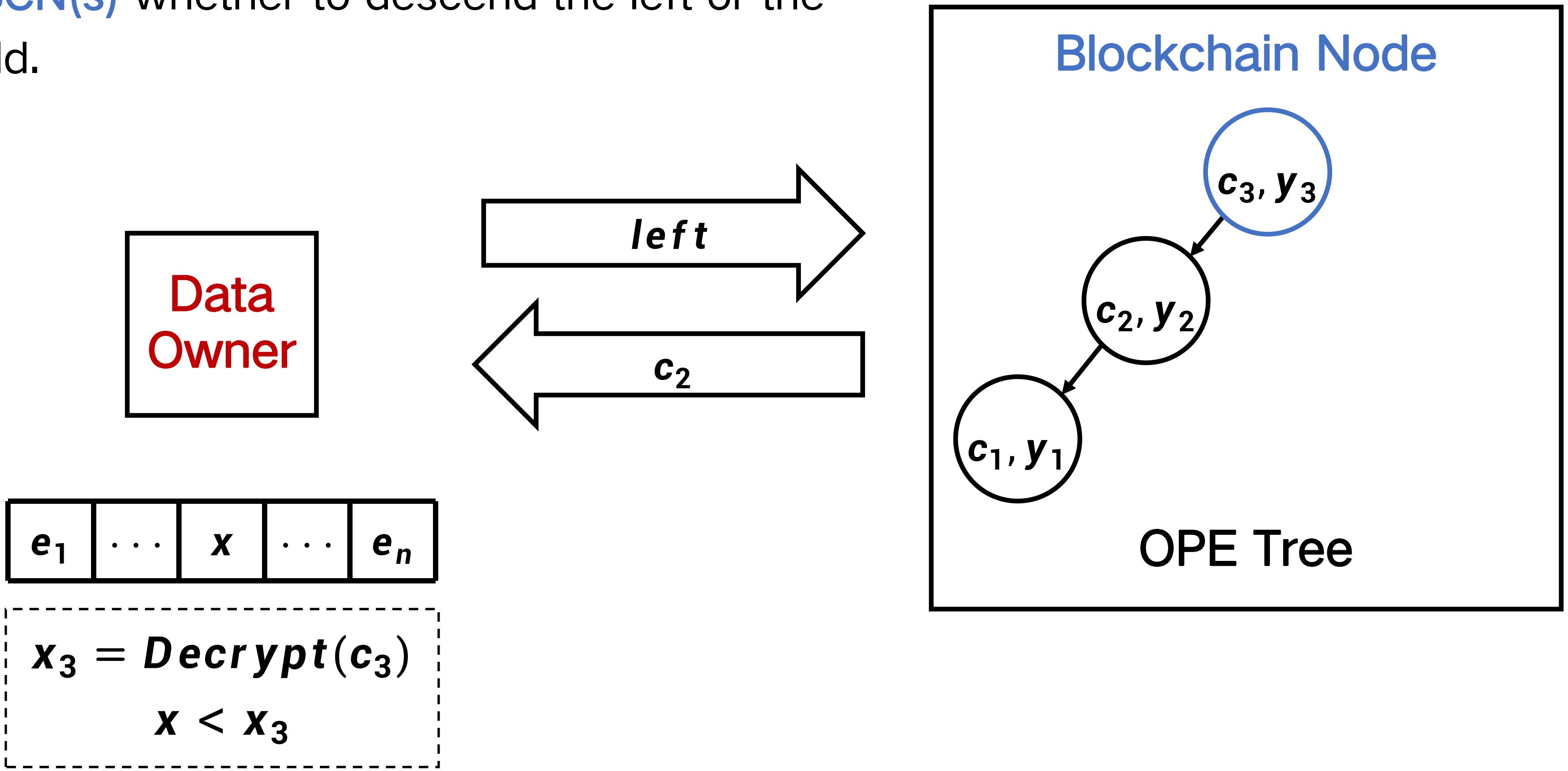
Methodology

- Encoding Phase:
  - DO sends an encoding request to BCN(s) to get the encoding result of its private data by interactively traversing the OPE dictionary.



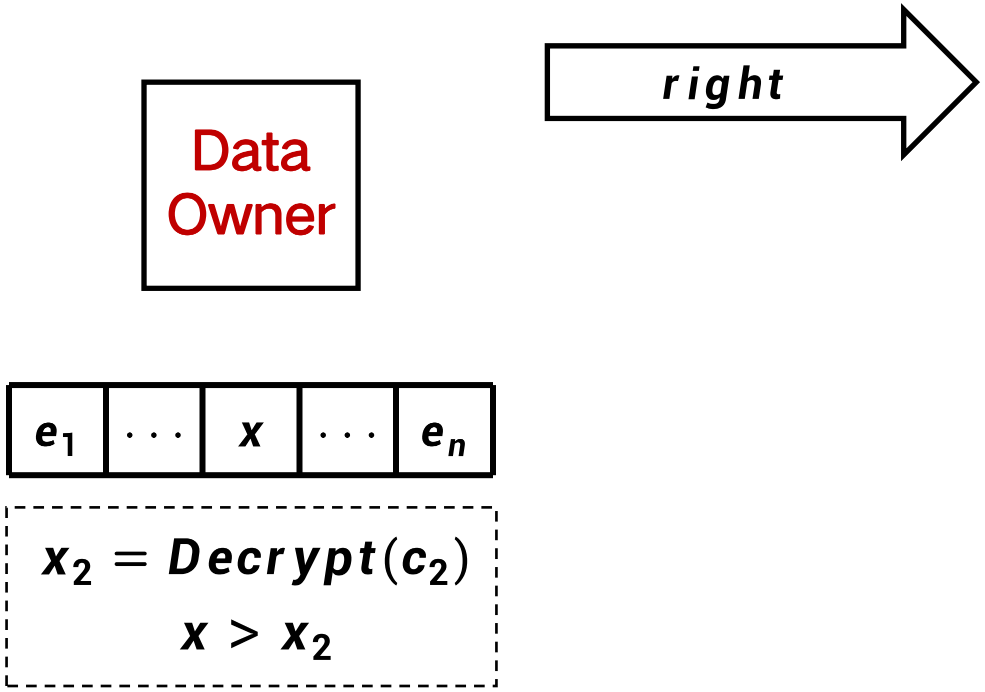
Methodology

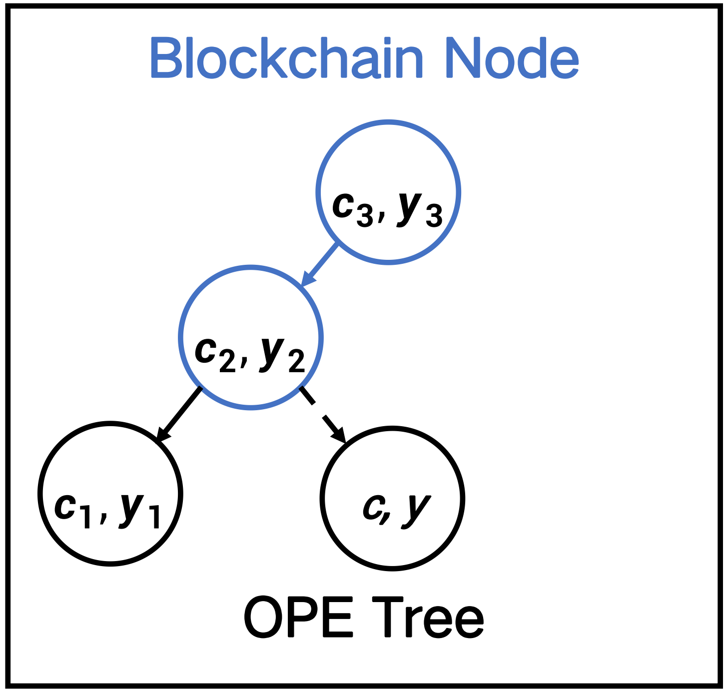
- Encoding Phase:
  - DO decrypts c_3 to x_3 and compares x_3 with x to inform BCN(s) whether to descend the left or the right child.



Methodology

- Encoding Phase:
  - Interactive operations end with arriving at **an empty vertex** or obtaining a vertex whose corresponding plaintext is equal to x.

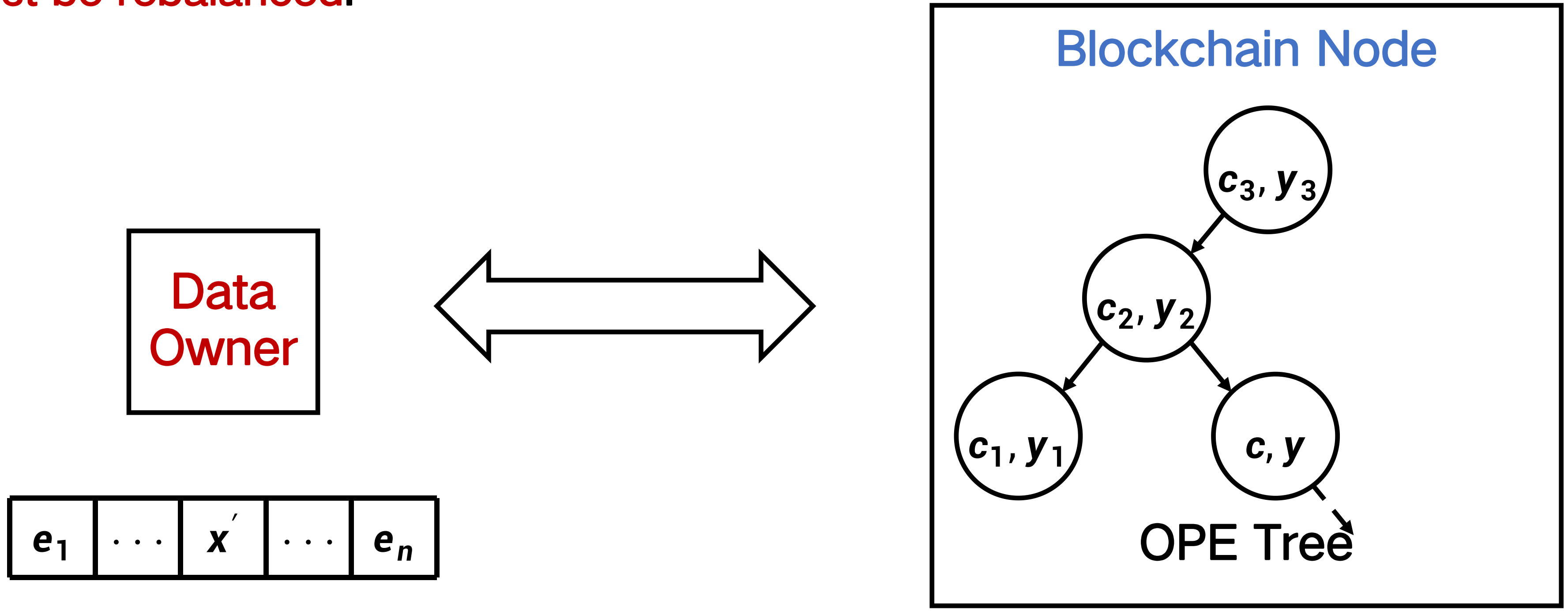




If $y_3 - y_2 > 1$, computes the arithmetic mean of y_3 and y_2 as the **code y**: $y = y_2 + \lceil \frac{y_3 - y_2}{2} \rceil$

Methodology

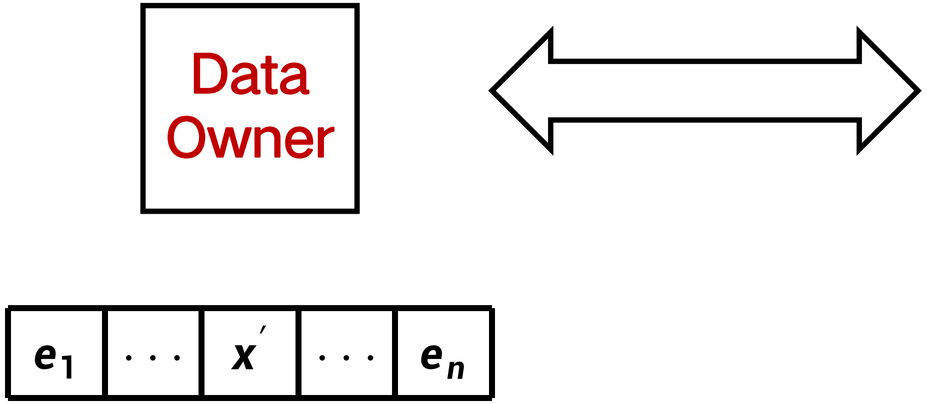
- Encoding Phase:
  - If the code space has been run out and **the OPE Tree must be rebalanced**.



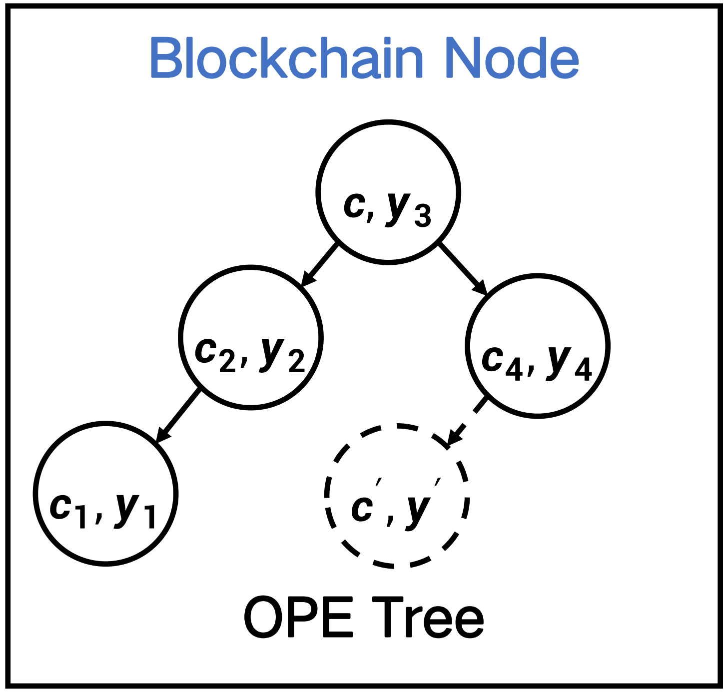
If $y_3 - y_2 = 1$,
the OPE Tree needs to be rebalanced

Methodology

- Encoding Phase:
  - Rebalance of OPE Tree: re-encoded all the existing codes to make them distributed more discretely.



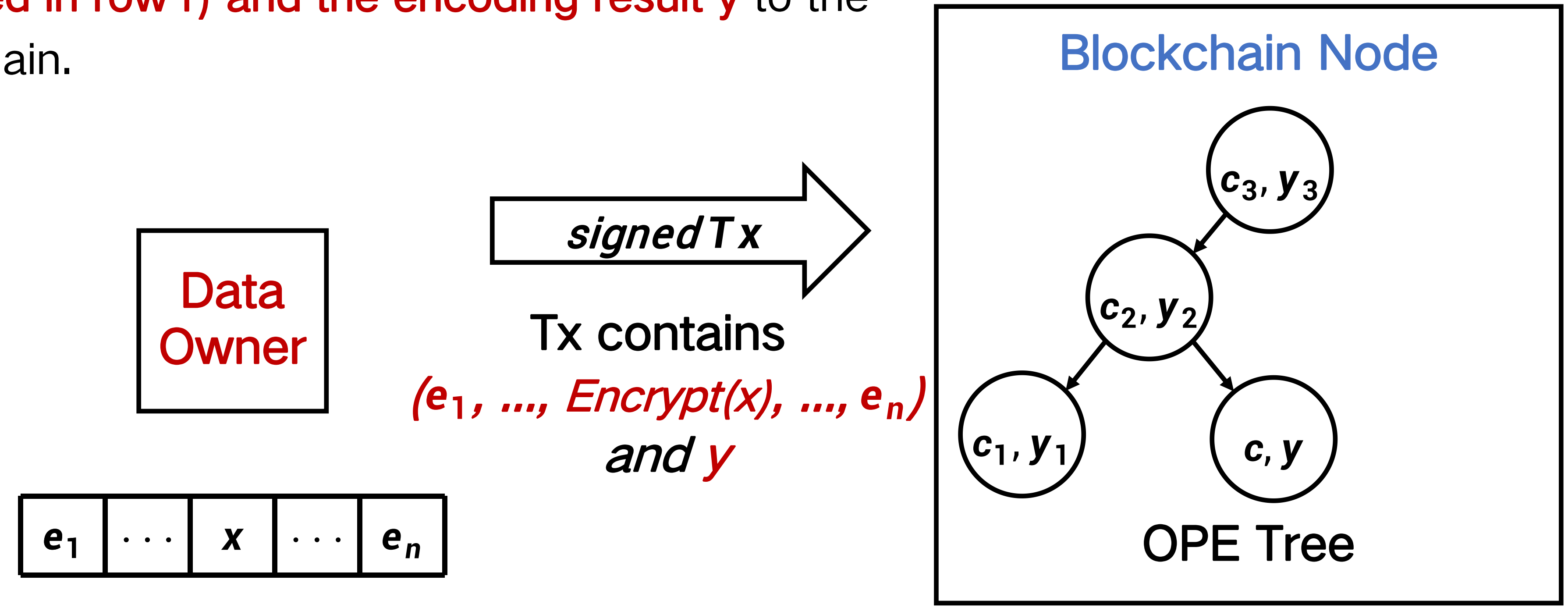
$$c' = Encrypt(x')$$



re-encode after tree rebalance

Methodology

- Execution Phase:
  - DO sends a transaction containing the ciphertext c (included in row r) and the encoding result y to the blockchain.

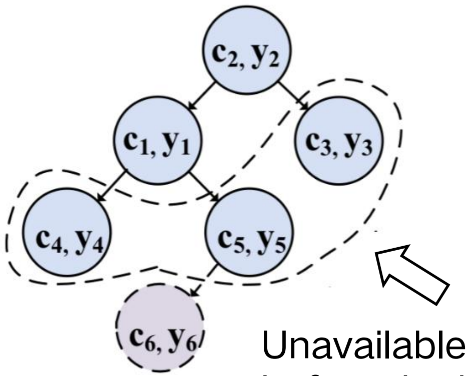


after consensus and replicated execution

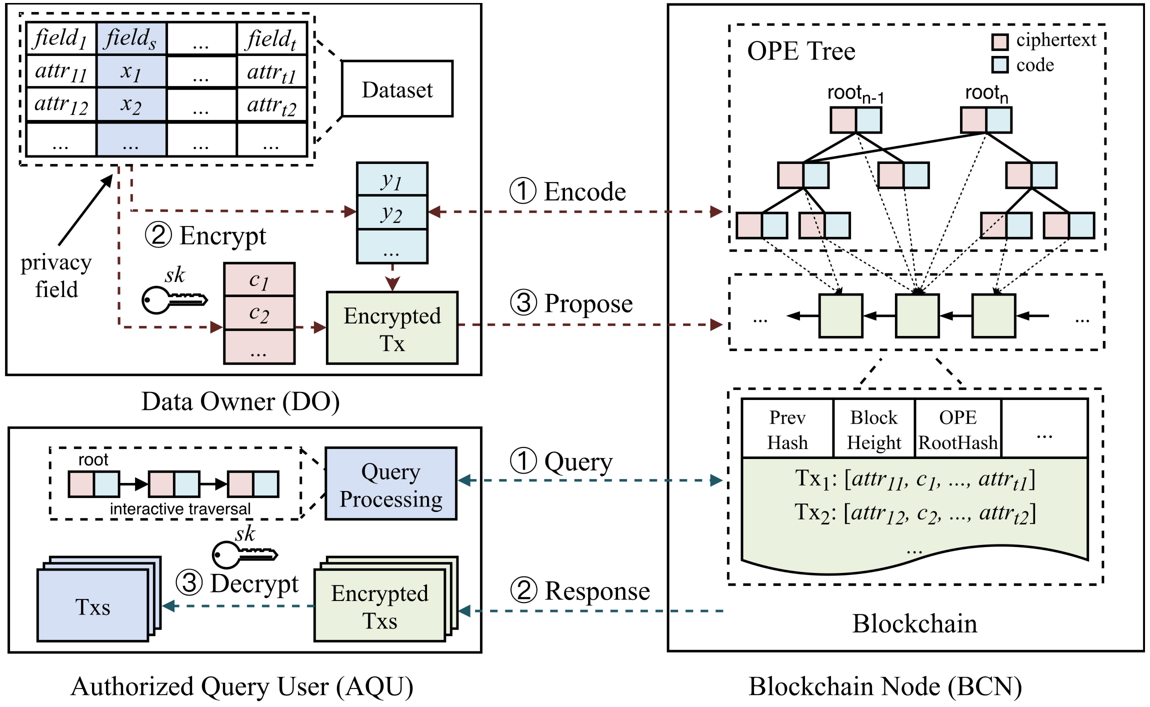
Optimization

- Parallelizing Two-phase BlockOPE:
  - Encoding Phase
    - encode multiple plaintexts **in parallel based on snapshots.**
  - Execution Phase
    - summon the updates and employ parallel execution.

Transaction execution writes the vertices **at the bottom of the OPE Tree**



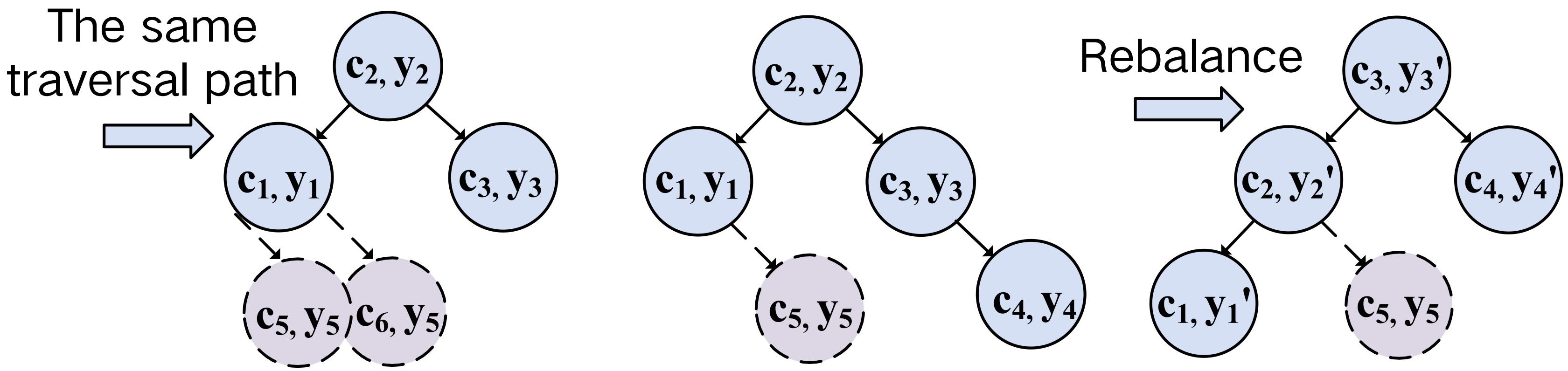
Unavailable for encoding before the block committed



Processing of BlockOPE

Optimization

- Reducing Conflicts:
  - The two-phase OPE scheme will **cause conflicts**, resulting in codes generated in the encoding phase cannot be appended in the execution phase.
    - **Same Code Conflict(SCC)**: multiple encoding requests for different plaintexts obtain the same traversal path in the OPE Tree.
    - **Update Conflict (UC)**: the OPE Tree is rebalanced before appending the codes.



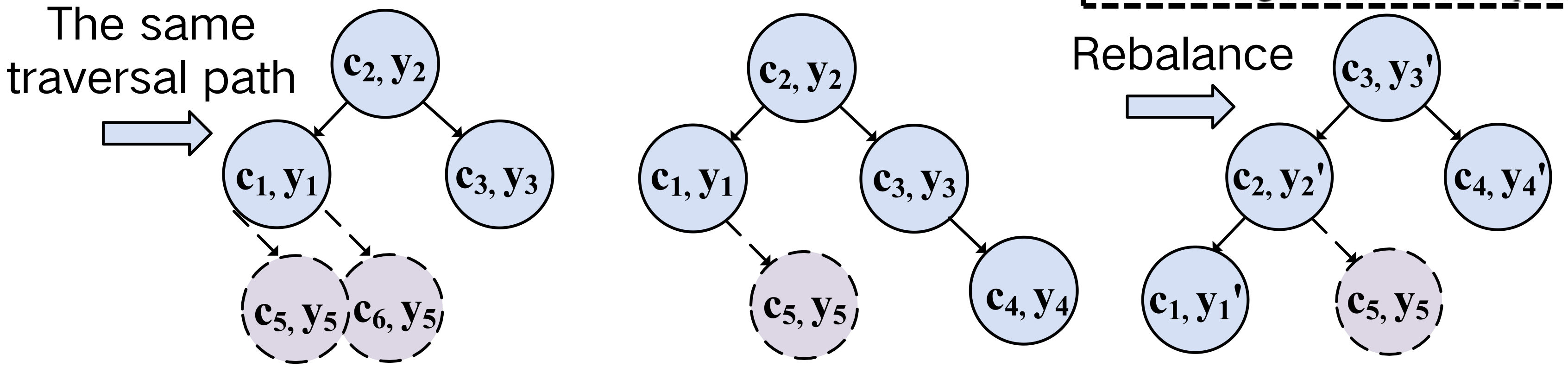
Same Code Conflict **Update Conflict** (before rebalance) **Update Conflict** (after rebalance)

Optimization

- Reducing Conflicts:
  - The two-phase OPE scheme will **cause conflicts**, resulting in codes generated in the encoding phase cannot be appended in the execution p
    - **Same Code Conflict(SCC)**: multiple encoding reque the same traversal path in the OPE Tree.
    - **Update Conflict (UC)**: the OPE Tree is rebalanced

Definition 3: Same Code Conflict (SCC). Given $x_1, x_2 \in \Phi$ and $x_1 \neq x_2$, if $\exists$ mappings $x_1 \rightarrow y$ and $x_2 \rightarrow y$, where $y \in \Upsilon$, it causes a *Same Code Conflict*.

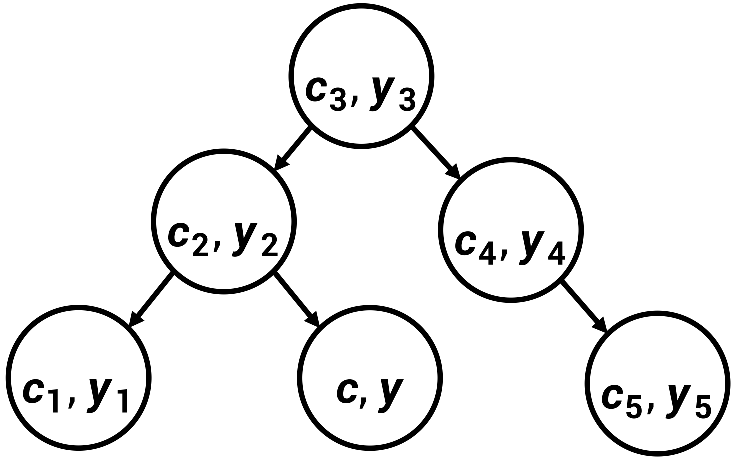
Definition 4: Update Conflict (UC). Given $x \in \Phi$, encoding x to $y \in \Upsilon$, this encoding relies on a set of vertices RS. Before appending vertex $\langle c, y \rangle$ to the OPE Tree, if any vertex in RS changes, it causes an *Update Conflict*.



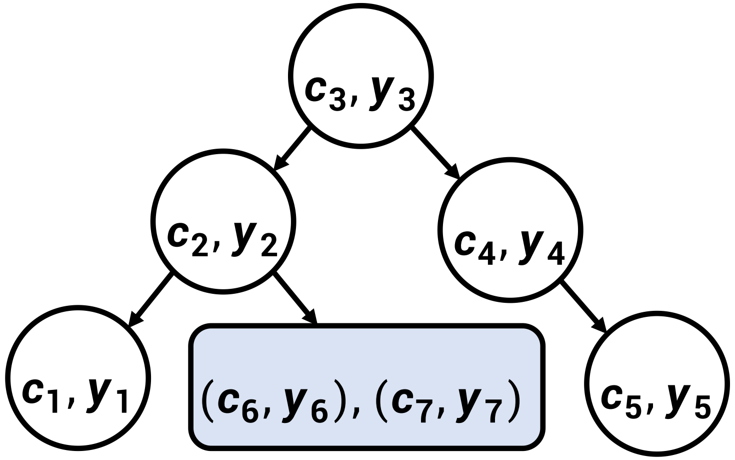
Same Code Conflict **Update Conflict** (before rebalance) **Update Conflict** (after rebalance)

Optimization

- Reducing Conflicts:
  - Randomized encoding
    - add a noise ε to eliminate the possibility of distinct plaintexts encoded to the same code.
  - Undecided-zone (UDZ) structure
    - a UDZ stores multiple vertices whose codes are non-order-preserving but within a certain range.



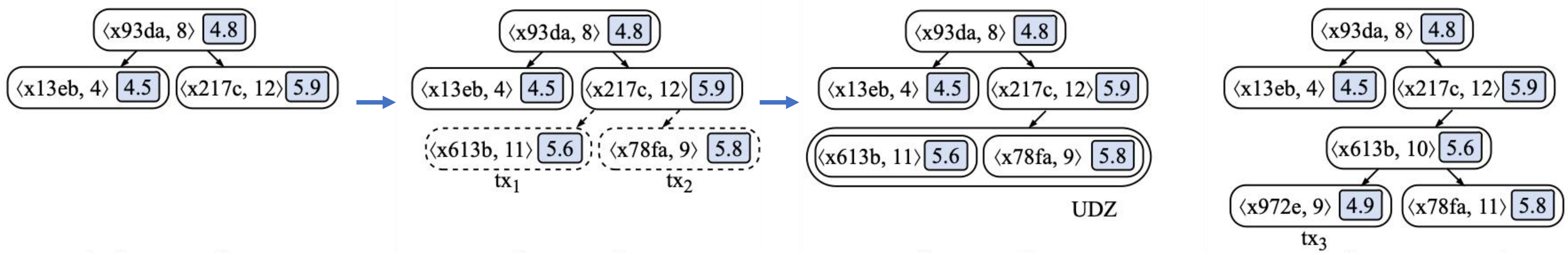
$$y = y_k + \lceil \frac{y_{k+1} - y_k}{2} \rceil + \varepsilon$$



undecided-zone

Optimization

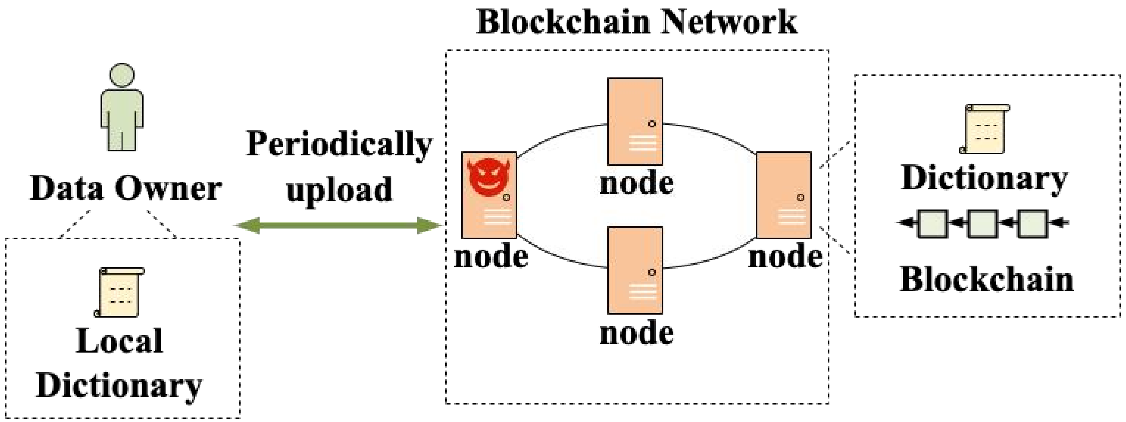
- Reducing Conflicts：
  - Randomized encoding
    - add a noise ε to eliminate the possibility of distinct plaintexts encoded to the same code.
  - Undecided-zone (UDZ) structure
    - a UDZ stores multiple vertices whose codes are non-order-preserving but within a certain range. (implicitly improve the encoding efficienncy due to reduced tree height)



before encoding　after encoding　after execution　case of rearrangement

Optimization

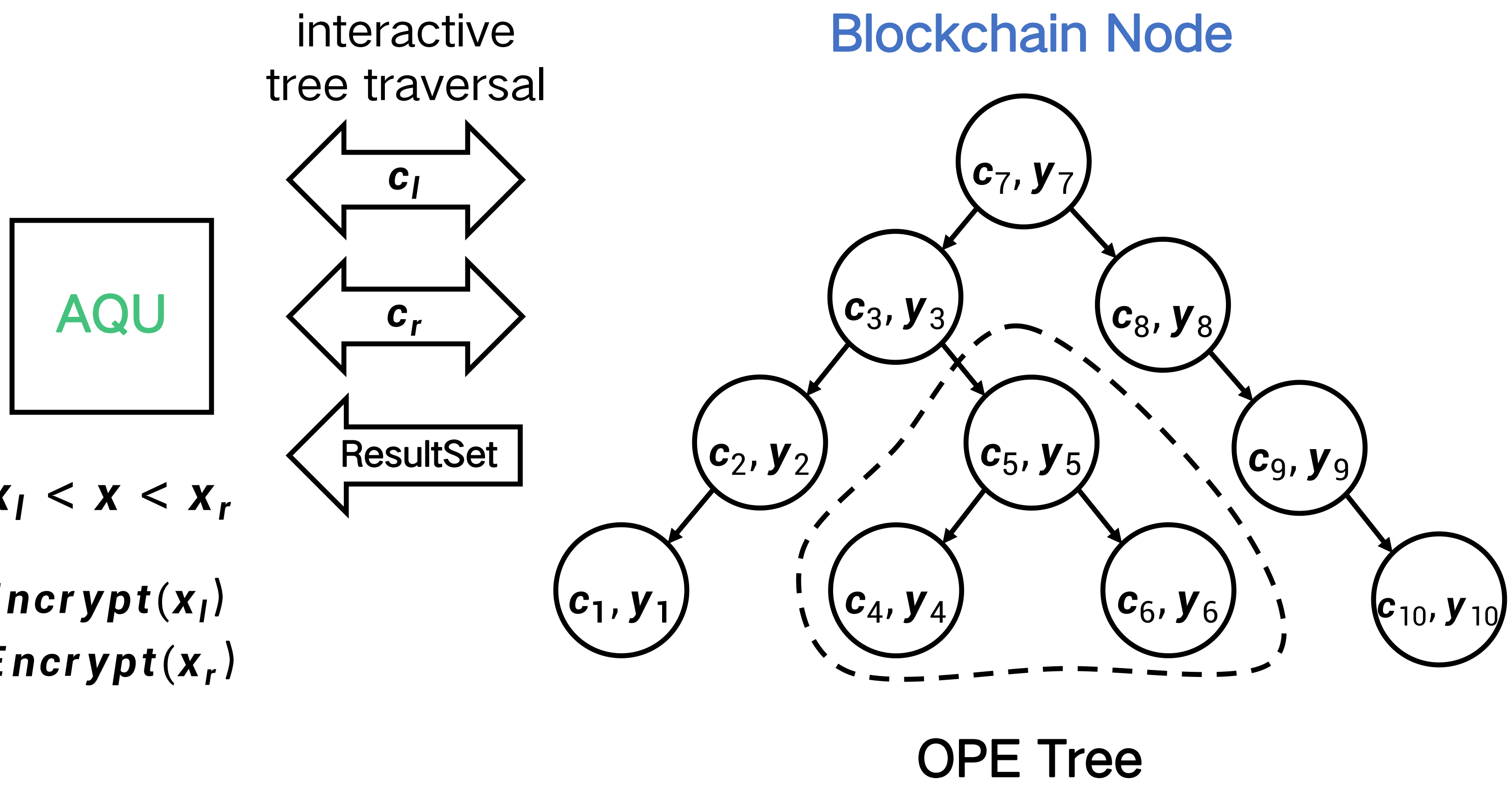
- Local Pre-processing:
  - An optional optimization to accelerate the processing
    - making **trade-offs between conflicts and client storage**.
    - on-chain insertion and resolution play the role of **final guards** to guarantee the correctness of BlockOPE.



Local Pre-processing

Query Processing

- Point/Range Query over ciphertexts:

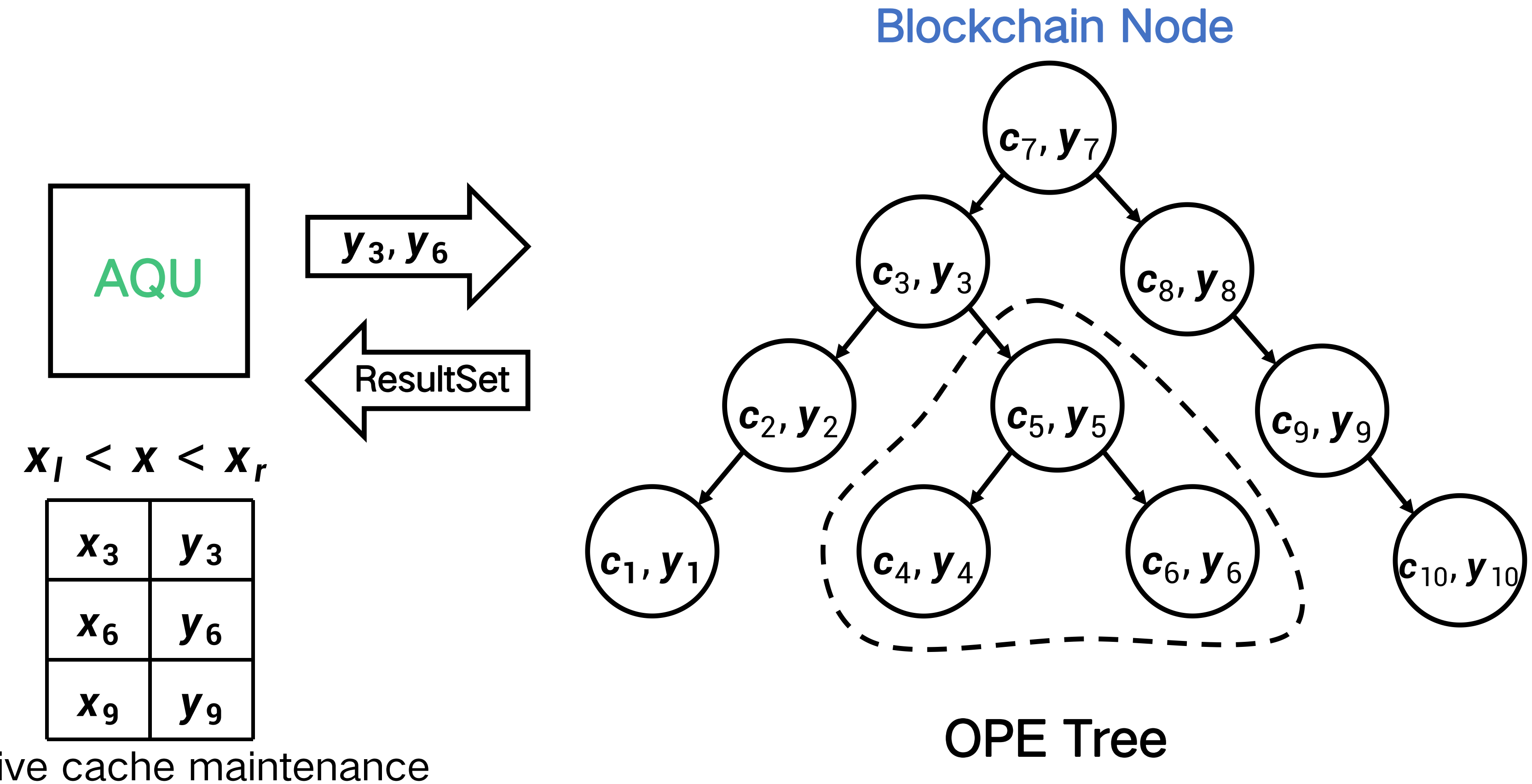


$$x_l < x < x_r$$

$$c_l = Encrypt(x_l)$$
$$c_r = Encrypt(x_r)$$

Query Optimization

- Point/Range Query over ciphertexts:
  - Adaptive Lightweight Cache
    - ➢ cache with uniform intervals -> adjust the cache according to query inputs



adaptive cache maintenance
(see Section VII.C)

Security

- IND-OCPA（indistinguishability under ordered chosen-plaintext attack）
  - Suppose there are two sequences of plaintexts: $S_0 = \{x_1^0, x_2^0, \dots, x_n^0\}$, $S_1 = \{x_1^1, x_2^1, \dots, x_n^1\}$,

  $x_i^0 < x_j^0 \Leftrightarrow x_i^1 < x_j^1$. Given the same initial state, if a malicious party cannot infer the corresponding plaintext sequence based on the encoding result, i.e., whether the encoding result is computed based on S_0 or S_1. Then the order-preserving encryption scheme is IND-OCPA.
- Frequency-based attacks
  - the relative order information in UDZ remains unknown.
- Plaintext guessing attacks
  - need auxiliary information.

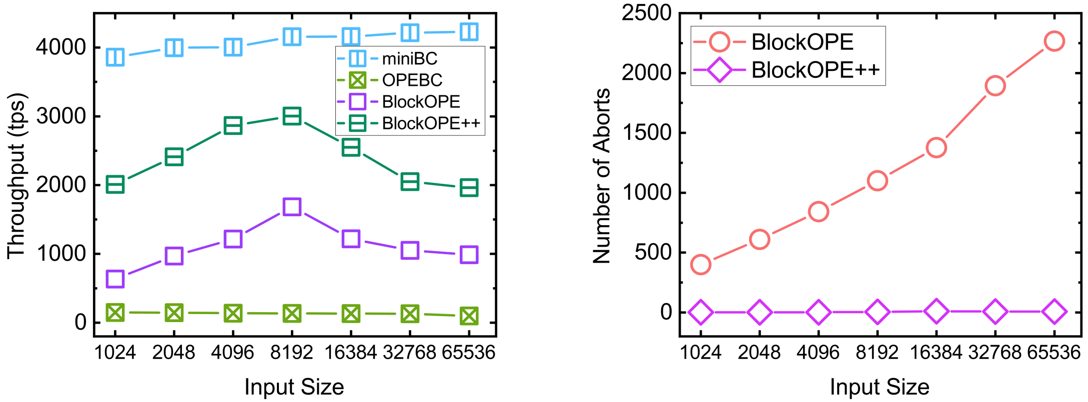
  (see Section VIII.A)

Implementation

- A prototype BlockOPE integrated with a permissioned mini-blockchain
  - Integrating an open-source PBFT component of Hyperledger Fabric 0.6.
  - A cluster of 4 permissioned blockchain nodes.
- Systems Compared
  - No OPE Baseline (**miniBC**): We build a mini permissioned blockchain as the baseline system which does not adopt any OPE scheme.
  - Serial OPE (**OPEBC**): We apply serial OPE scheme to miniBC where one block contains at most one transaction to simulate traditional OPE scheme.
  - Parallel BlockOPE (**BlockOPE**): This system is the parallel BlockOPE prototype without reducing conflicts.
  - Efficient Parallel BlockOPE (**BlockOPE++**): This system is the parallel BlockOPE prototype combined all optimizations.

Experiment

- Performance Evaluation
  - **Varying input size**
  - Varying block size
  - Varying number of executing threads
  - Varying UDZ capacity
  - Evaluation of Conflict Reduction



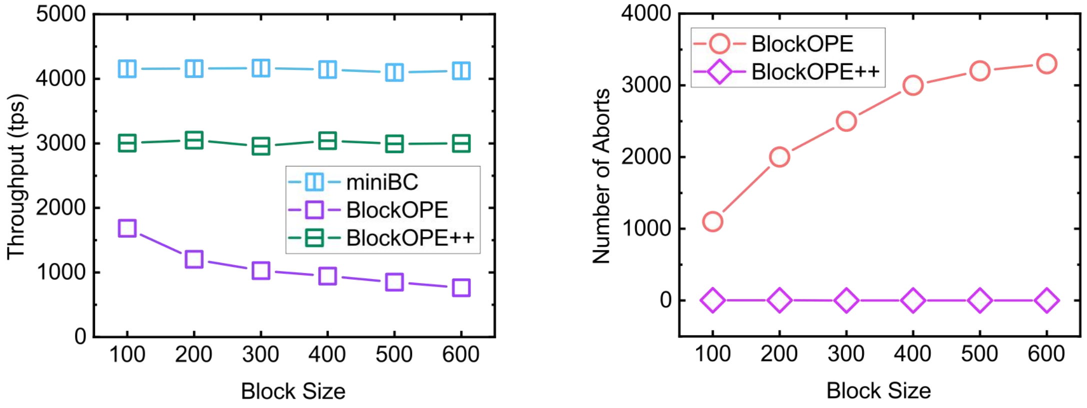
Varying input size

plaintext domain size $M = 2^{16}$
code domain size $N = M^3 = 2^{48}$

Parameter	Value	Parameter	Value
# of enc. threads	1 2 4 <u>8</u> ...	input size	4096 <u>8192</u> ...
# of exe. threads	1 2 4 <u>8</u> ...	block size	<u>100</u> 200 ...
UDZ capacity	<u>100</u> 200 ...	distribution	uniform

Experiment

- Performance Evaluation
  - Varying input size
  - **Varying block size**
  - Varying number of executing threads
  - Varying UDZ capacity
  - Evaluation of Conflict Reduction



Varying block size

plaintext domain size $M = 2^{16}$
code domain size $N = M^3 = 2^{48}$

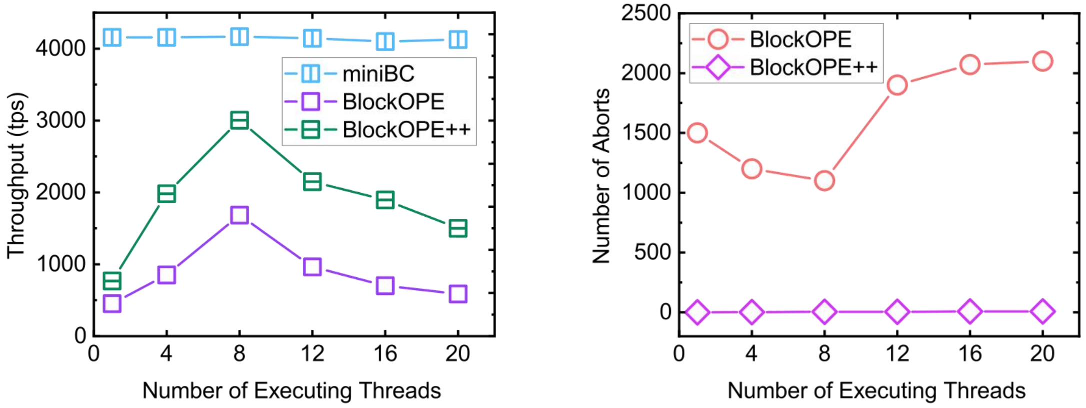
Parameter	Value	Parameter	Value
# of enc. threads	1 2 4 <u>8</u> ...	input size	4096 <u>8192</u> ...
# of exe. threads	1 2 4 <u>8</u> ...	block size	<u>100</u> 200 ...
UDZ capacity	<u>100</u> 200 ...	distribution	uniform

Experiment

- Performance Evaluation
  - Varying input size
  - Varying block size
  - **Varying number of executing threads**
  - Varying UDZ capacity
  - Evaluation of Conflict Reduction

plaintext domain size $M = 2^{16}$
code domain size $N = M^3 = 2^{48}$

Parameter	Value	Parameter	Value
# of enc. threads	1 2 4 8 ...	input size	4096 8192 ...
# of exe. threads	1 2 4 8 ...	block size	100 200 ...
UDZ capacity	100 200 ...	distribution	uniform



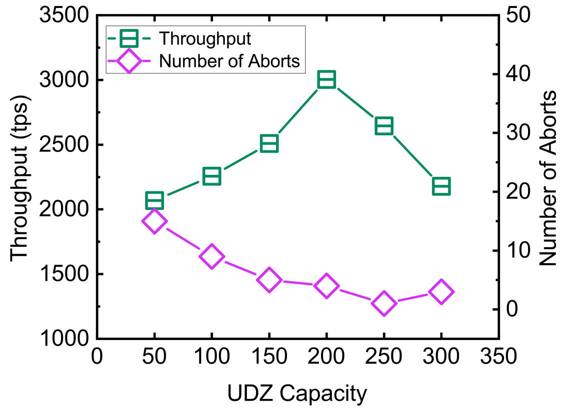
Varying number of executing threads

Experiment

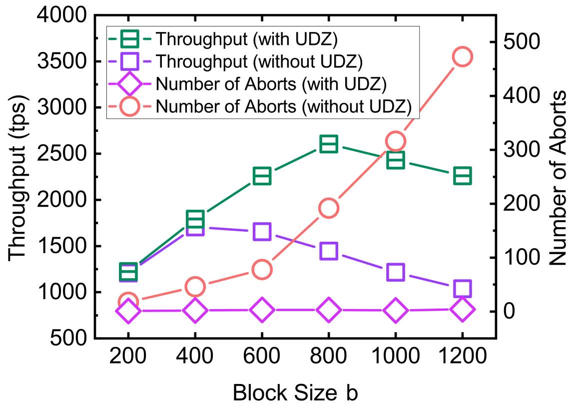
- Performance Evaluation
  - Varying input size
  - Varying block size
  - Varying number of executing threads
  - **Varying UDZ capacity**
  - **Evaluation of Conflict Reduction**

plaintext domain size $M = 2^{16}$
code domain size $N = M^3 = 2^{48}$

Parameter	Value	Parameter	Value
# of enc. threads	1 2 4 8 ...	input size	4096 8192 ...
# of exe. threads	1 2 4 8 ...	block size	100 200 ...
UDZ capacity	100 200 ...	distribution	uniform



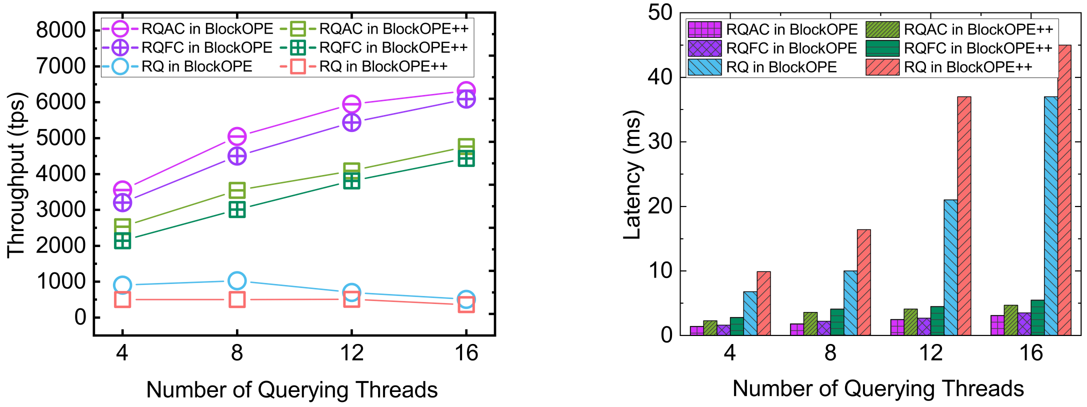
Varying UDZ capacity



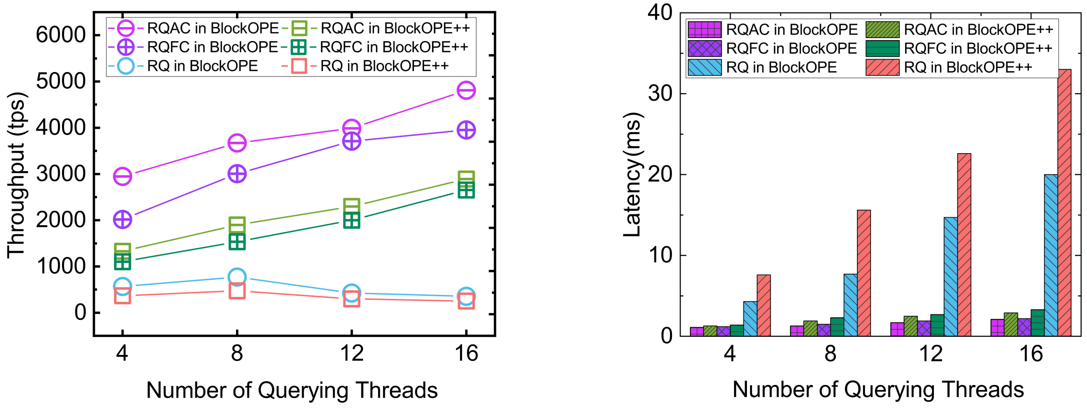
Evaluation of Conflict Reduction

Experiment

- Query Performance
  - Interactive range query (RQ)
  - Range query with fixed cache (RQFC)
  - Range query with adaptive cache (RQAC)



ResultSet size: 2000



ResultSet size: 4000

Conclusion

- **BlockOPE** is the **first blockchain-based OPE scheme** that brings privacy-protection to the blockchain while still preserving efficient order comparison primitives over ciphertexts.

- It provides efficient encoding through **parallelizing techniques and conflict reduction design**. It also utilizes an adaptive cache-based method for queries on ciphertexts.

- BlockOPE could be used to support **blockchain-based data sharing and collaboration scenarios** that require data privacy and efficient query ability.

THANKS !