IMCOM 2026
International Conference on Ubiquitous Information Management and Communication
January 4~6, 2026 – Hanoi, Vietnam

A Data Integration Platform with Identifiable Falsification Detection and its Evaluation in Automotive Parts Manufacturing

Presenter: Haruka Hori

ISBN: 979-8-3315-9017-8

Outline

Introduction

Approach and Design

Evaluation

Conclusion

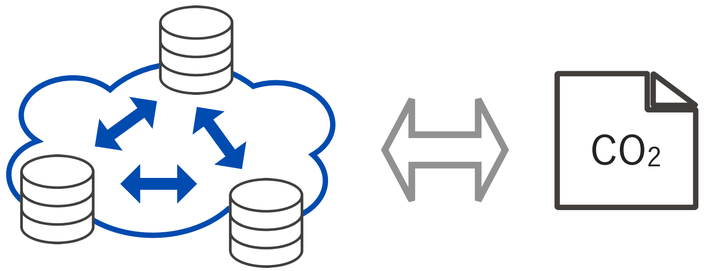
Introduction

Growing Importance of **Supply Chain CFP (Carbon Footprint of Products) Management** toward Decarbonization

- A key to identifying hotspots and selecting green products
- Increasing regulatory pressure in the automotive sector

Implementation Challenges

- **Data Falsification**
  Risk of fraud due to compliance pressure
- **Ensuring Traceability and Trust**
  Secure data integration across multiple enterprises in distributed systems



A distributed system
connecting each company

Introduction

Previous Work & Challenges

A study of blockchain-based metadata management and its use for data verification (Hori, Oguchi, the 12th International Symposium on Computing and Networking Workshops, 2024)

Realized a system to calculate, store, and verify CFP data using smart contracts.

Problem

- **Unidentifiable Falsification Location:** Can detect the *existence* of falsification, but cannot pinpoint the specific source.
- **Insufficient Security:** CFP data is processed and stored in plaintext.

Our Approach

- **Proposal: Hashed Component Tree**
  - A tree structure that integrates hash values using **XOR operations**.
  - **Leveraging XOR Properties**
    - Eliminates order information in integration.
    - Enables identification of the falsified location.
  - Handling **Duplicate parts**
- **System Design:** Modeled on the automotive parts manufacturing industry.

5

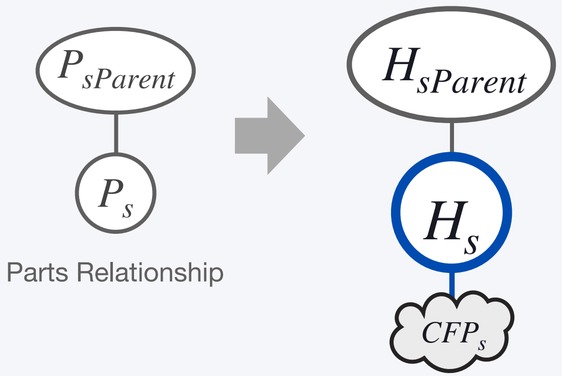
Definition: Hashed Component Tree

Determine a single hash value for each part by XORing hashed CFPs.

Single part

Hash value of its own CFP.

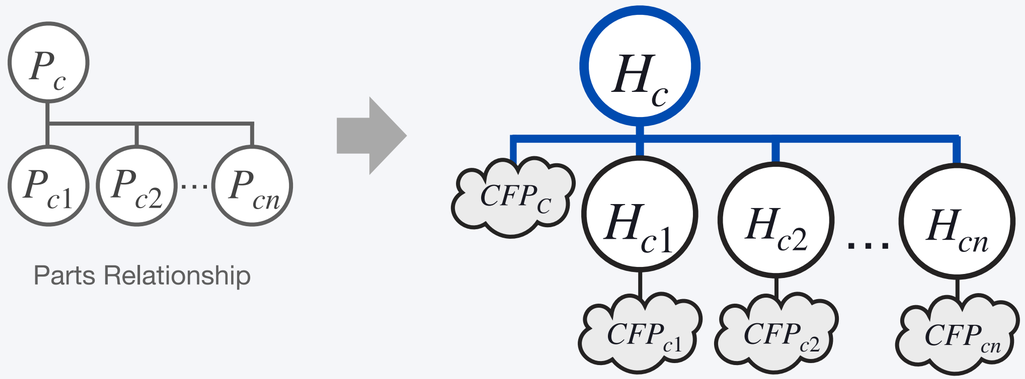
$$H_s = Hash(CFP_s)$$



Composite part

XOR combination of its own CFP and hash values of child parts.

$$H_c = Hash(CFP_c) \oplus H_{c1} \oplus H_{c2} \oplus \cdots \oplus H_{cn}$$



Root part is an automobile that contains CFP of **all** parts via XOR aggregation

6 Key Feature 1: XOR Properties

Advantages 1) Commutativity

Elimination of Order Information

$$a \oplus b = b \oplus a$$

- **No inherent ordering** among parts in this scenario.
- Parent hash remains unchanged even if the input order varies.

Advantages 2) Self-Inverse

Identification of Falsification

$$a \oplus a = 0$$

- Impact of falsification propagates toward the root part.
- **Clearly distinguishes** between the source of falsification and the affected parent nodes.

Key Feature 2: Handling Duplicate Parts

Issue Disappearance due to Self-Inverse Property

- Simple XOR causes **cancellation** in higher-level parts.
  → Loss of information for duplicated components.

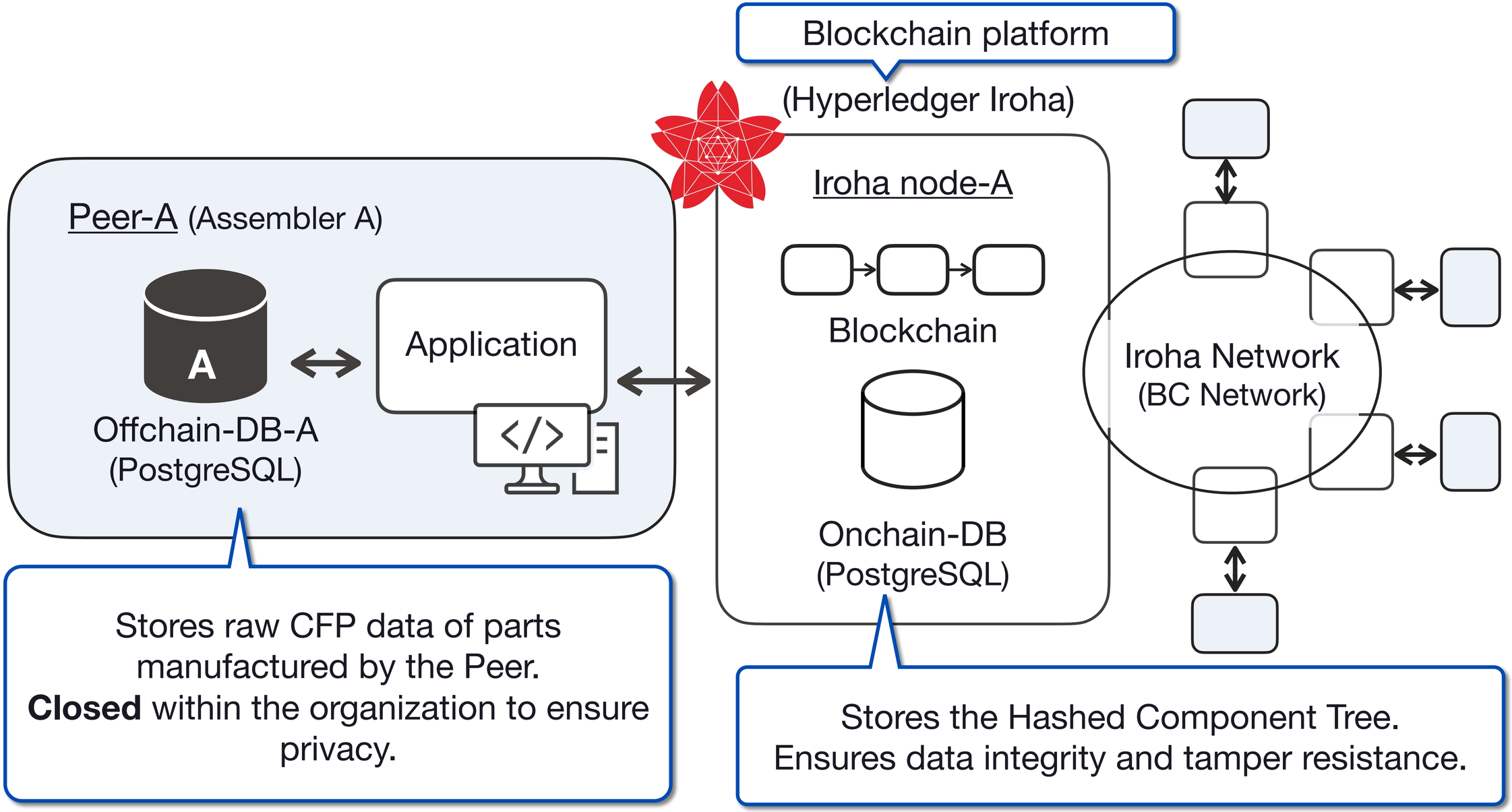
Solution

For a duplicate part H_{cd}, **concatenate the path to the root before hashing**.

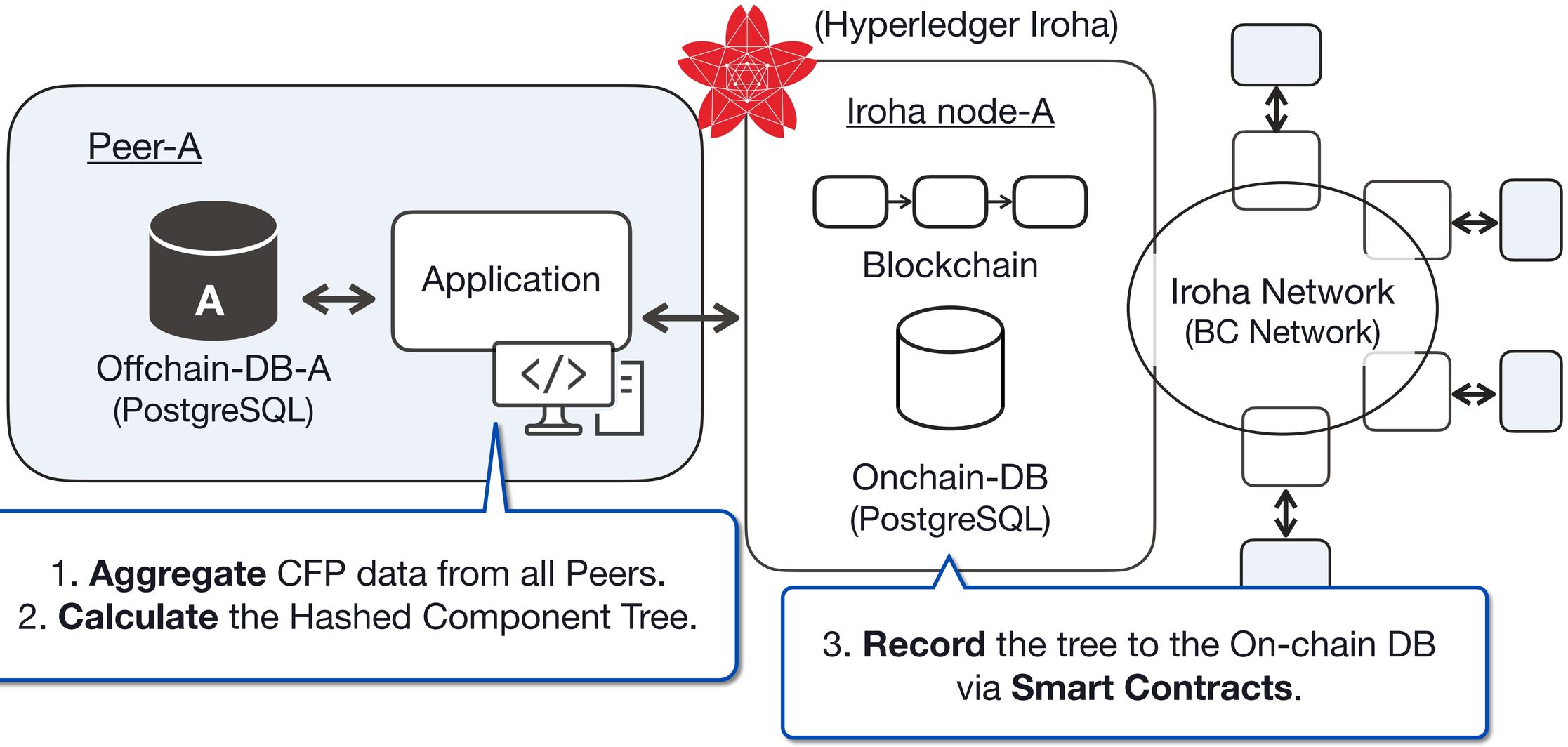
$$H_c = Hash(CFP_c) \oplus H_{c1} \oplus \cdots \oplus Hash(H_{cd} || Path(P_c, Root)) \oplus \cdots \oplus H_{cn}$$

- $Path(A, B)$: The specific path from node *A* to node *B*.
- **Benefit**: Ensures all hash values are unique, preventing cancellation caused by the self-inverse property.

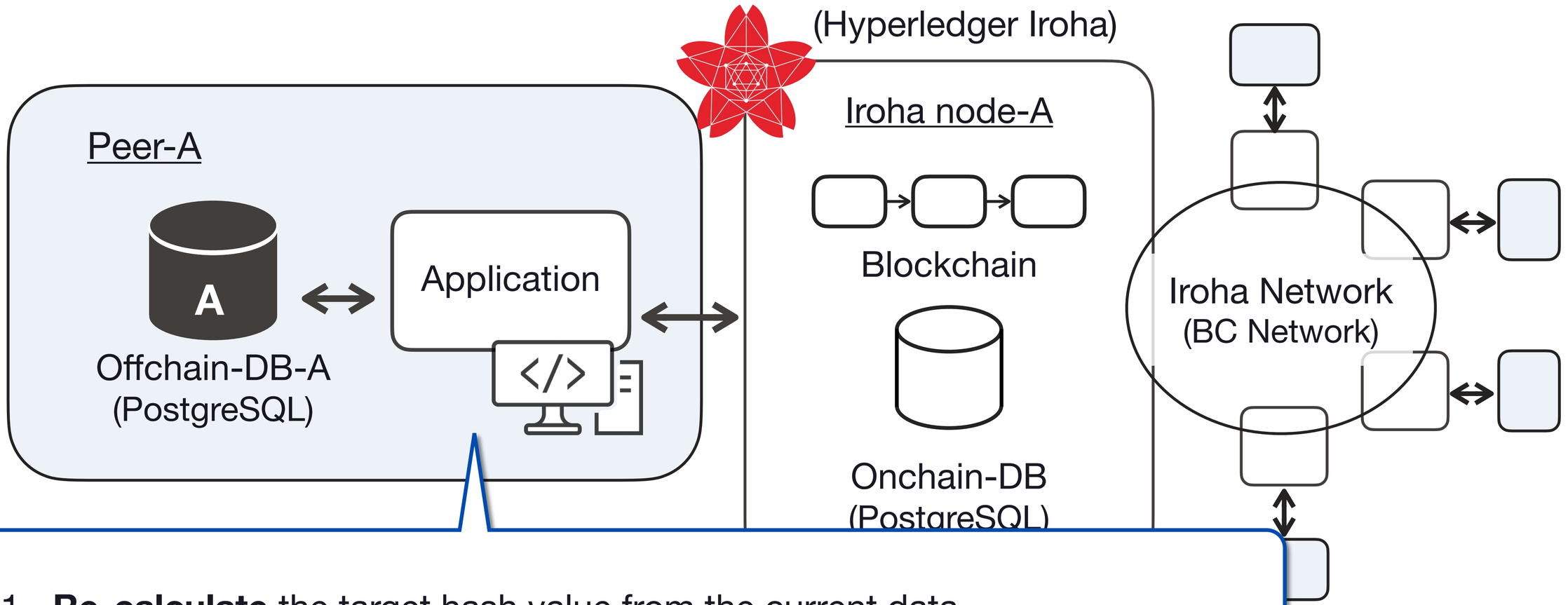
8 System Design



The Hashed Component Tree Generation Process



10 The Verification Process



1. **Re-calculate** the target hash value from the current data.
2. **Compare** it with the reference hash stored in the On-chain DB.
   → If a mismatch occurs, proceed to the Falsification Identification Process.

Falsification Localization Process

Purpose Verify if the part is the **actual source** of falsification using XOR's self-inverse property.

Process

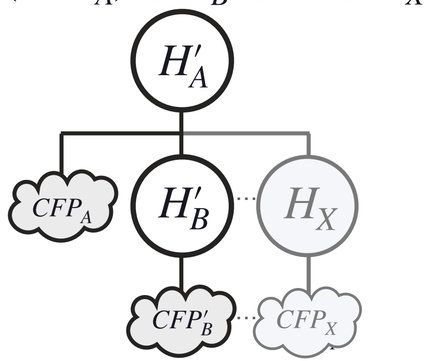
1. Search for hash discrepancies starting from the verification part (Root).
2. Difference Removal
3. Falsification Judgment

2. Calculation Method

Remove the child's influence (H'_B) from the parent (H'_A), and XOR with the correct child value retrieved from the On-chain DB.

$$Check = H'_A \oplus (H'_B \oplus H_B)$$

$$H'_A = Hash(CFP_A) \oplus H'_B \oplus \cdots \oplus H_X$$



12

Falsification Localization Process

Purpose Verify if the part is the **actual source** of falsification using XOR's self-inverse property.

Process

1. Search for hash discrepancies starting from the verification part (Root).
2. Difference Removal
3. Falsification Judgment

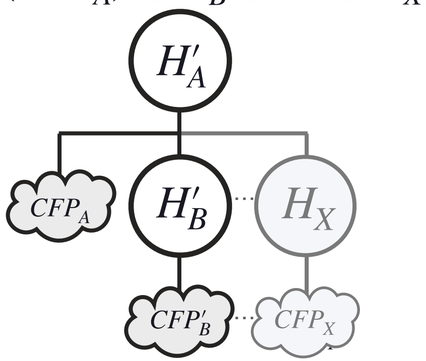
3. Judgement Logic

Does the calculated $Check$ value match the original correct hash (H_B)?

Match: Parent is **NOT** falsified (only affected by the child).

Mismatch: Parent **ITSELF** is falsified.

$$H'_A = Hash(CFP_A) \oplus H'_B \oplus \cdots \oplus H_X$$



13 Evaluation Setup

Investigated execution time, CPU usage, and memory usage verification process.

Build Environment

Ubuntu Container		PostgreSQL Container	
Blockchain Platform	Application	Onchain-DB	Offchain-DB
Hyperledger Iroha ver. 1	Python	PostgreSQL ver.16.3	
Virtual Environment	Docker		
OS	Ubuntu20.04LTS		
Physical Server	CPU : Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz MEMORY : 192GB, CORE : 16, THEAD : 32		

Evaluation Parameters

- Total number of parts : 300 / 3,000 / **30,000**
- Duplication rate of parts : 0 / 10 / 20 / 30 / 40 (%)
- Falsification rate of parts : 0 / 1 / 5 / 10 / 15 (%)

Equivalent to the total number of parts in a single vehicle.

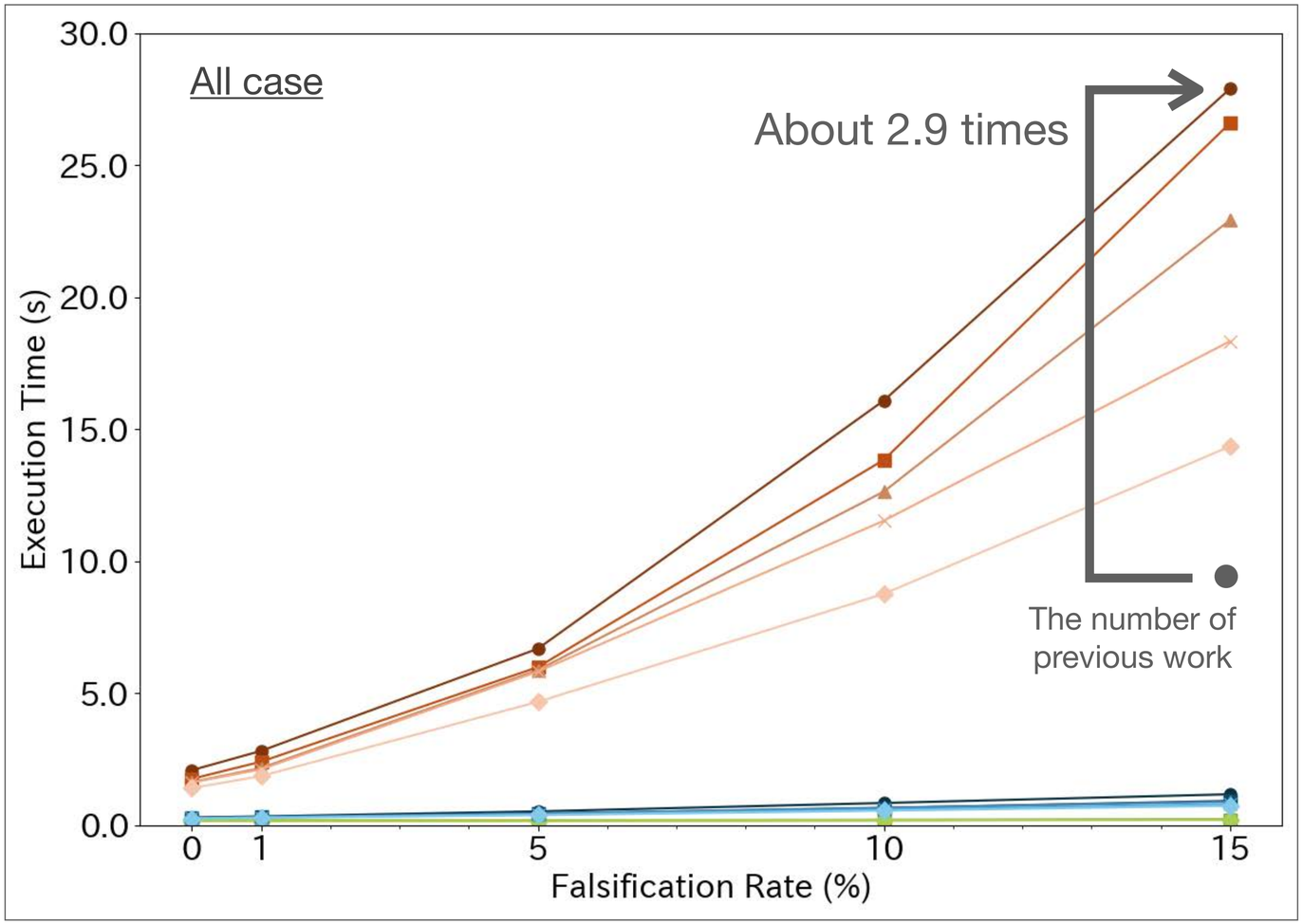
14 Summary of Evaluation Results

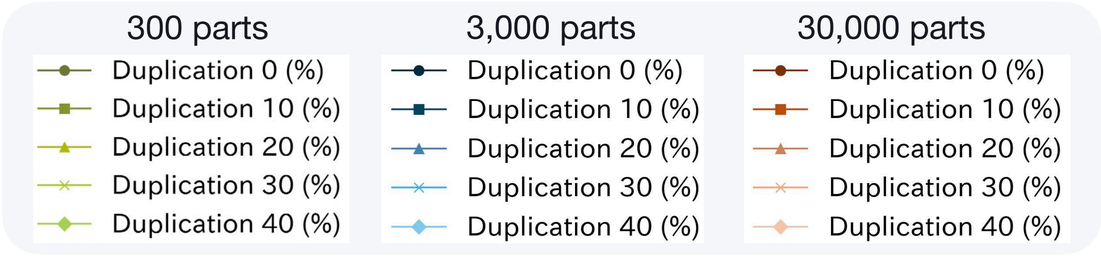
	Execution Time	CPU Usage	Memory Usage
Total number of parts	30,000 parts: 2.9 times the number of previous work Linear increase		
Duplication rate of parts	Lower is faster	largely unaffected	
Falsification rate of parts	Lower is faster		

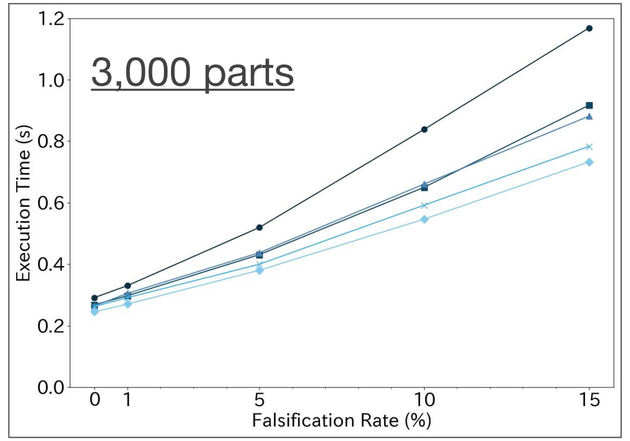
We achieved a 100% falsification identification rate, albeit with increased overhead.

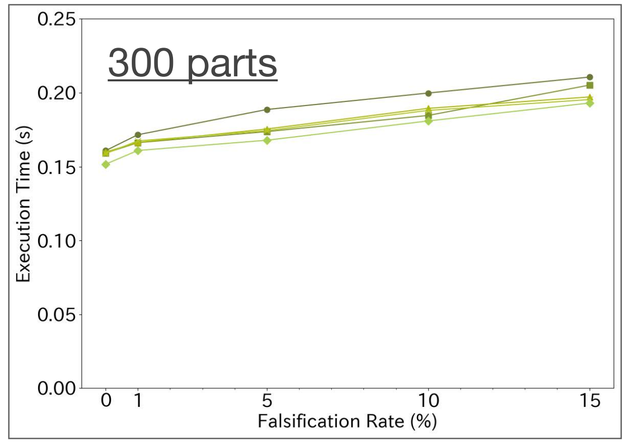
15

Results of Execution Time









Total number of parts

Linear increase

Duplication rate of parts

Lower is faster

Falsification rate of parts

Lower is faster

Conclusion

Conclusion

- Proposed a distributed CFP data integration platform for automotive parts manufacturing.
- **Developed a verification method** using Hashed Component Trees based on XOR properties:
  - **Commutativity Property:** Eliminates the need for component order information.
  - **Self-Inverse Property:** Identifies the specific location of falsification.
  - Handling duplication parts
- **Achieved a 100% identification rate** in evaluation experiments, confirming the system's expected behavior.

Future Work

- **Implement a data correction mechanism** for the identified falsified components.

Thank you for your attention.