一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

改ざんが特定可能なデータ連携基盤の構築と自動車部品製造業への応用における評価

堀　　遥† Le Hieu Hanh† 小口　正人†

† お茶の水女子大学　〒112–8610 東京都文京区大塚 2–1–1
E-mail: †haruka-h@ogl.is.ocha.ac.jp, ††{le,oguchi}@is.ocha.ac.jp

あらまし　脱炭素社会実現に向け，我々は自動車部品製造業を題材に，各部品のカーボンフットプリント (CFP) の管理，および CFP の改ざん発生を検知する検証機能をブロックチェーンベースで実装してきた．一方でこれまでの手法では，改ざんの具体的な発生箇所が特定不可であり，CFP を全て平文で扱っているという懸念が残されていた．この課題を可決すべく，我々はハッシュ木の階層的なハッシュ構造と XOR の性質に着目し，改ざん箇所の特定とデータの安全性を両立する手法を実装・評価した．

キーワード　分散システム，ブロックチェーン，サプライチェーン管理，カーボンフットプリント

Development of a Data Integration Platform with Identifiable Falsification Detection and Its Evaluation in Automotive Parts Manufacturing

Haruka HORI†, Hieu Hanh LE†, and Masato OGUCHI†

† Ochanomizu University　2–1–1 Otsuka, Bunkyou–ku, Tokyo 112–8610 Japan
E-mail: †haruka-h@ogl.is.ocha.ac.jp, ††{le,oguchi}@is.ocha.ac.jp

Abstract　To promote a decarbonized society, we develop a blockchain-based platform for the automotive parts manufacturing industry that enables management and representation of each part's carbon footprint (CFP) and detection of falsification attempts. Previous implementations, however, could not identify the exact location of tampering and required handling CFP data in plaintext, raising security concerns. To overcome these issues, we leverage the hierarchical hash structure of hash trees and the properties of XOR operations. Our proposed method enables both pinpoint identification of tampered components and secure management of CFP data, which we implement and evaluate on a prototype system.

Key words　Distributed System, Blockchain, Supply Chain Management, Carbon Footprint of Products

1. ま え が き

1.1 研 究 背 景

2050 年までにカーボンニュートラル達成という目標を表明する 150 以上の国々において，多様な取り組みが推進されており，その一つにカーボンフットプリントがあげられる．カーボンフットプリント (以降，CFP：Carbon Footprint of Products) とは，図 1 のように製品のライフサイクルの各過程で直接的・間接的に排出された温室効果ガスを CO_2 排出量に換算し，製品単位で表示する仕組みである．

そして現在，サプライチェーン全体で CFP を管理して，製品単体の CFP を管理・表現できるような分散データ基盤の開発が期待されている．このようなデータ基盤実現のメリットととして，一点目にホットスポットの特定により，効率的な削減につながること，二点目に表示により低炭素・脱炭素製品が積極的に選ばれること，三点目にサプライチェーン上の他事業者による排出削減が自社の削減とみなされるため，各社の削減の可能性が広がることが挙げられる．

図 1: カーボンフットプリント概要

CREST にて進行中のプロジェクト「検証可能なデータエコシステム」[1] では，データに付随する信頼度や来歴を第一級のデータとしてサポートし，任意のデータが検証可能であるようなデータエコシステムの研究開発を目指しており，実証実験として自動車製造業における CFP 管理に取り組んでいる．我々は中でも，データエコシステムの基盤となる分散データベースシ

ステムに，安全性を提供する役割を担当する．

1.2 研究目的

CFP 管理基盤では，複数企業を横断する分散 DB システムでトレーサビリティを表現し，CFP データを集約・算定・保管することが求められる．自動車製造業という実証実験シナリオにおいては，三つの課題が挙げられる．一点目は複数の異なる企業が分散システムに参加するため，異なる企業間の連携の信頼性を担保する必要がある．二点目に改ざんは外部攻撃だけでなく，自社内においても発生する可能性がある．近年，自動車の品質不正の事例も多く，CFP も例外ではない事から，改ざん検知機能の実現が求められる．三点目は部品は複数の部品を組み立てて作られるため，階層的な構成をしており，製品単位の CFP 値の計算には再帰的な探索が必要となる．

我々はこれまでに，一点目と二点目の課題に対して，Peer-to-peer ネットワーク上のデータ検証機能の導入を検討してきた．さらに，三点目に対しては，一度算出した CFP の値を保管し，検証時に再利用することで再帰処理の一部省略を図った．システムの実装にあたっては，実世界で進められている CFP 管理基盤を参考に，ブロックチェーンプラットフォーム Hyperledger Iroha を基盤とした．そして Hyperledger Iroha におけるスマートコントラクト技術である組み込みコマンドを用いて，分散システム上での高速なデータ検証を行い，改ざん検知するようなシステムを実装・評価してきた [2].

一方で，先行研究での手法には次の懸念が残されている．一点目に，データ検証機能では改ざんの発生を検知できるものの，発生箇所を特定することができない．二点目に，部品の重複が存在しない前提で設計されており，重複への考慮がなされていなく，現実に十分に反映されない．三点目に，CFP データを全て平文で処理・保管しており，情報セキュリティ上の配慮が不十分である．

これらの課題を解消し，先行研究より安全で高性能な手法を目指して，本稿では，データ構造の一種であるハッシュ木に着想を得た「ハッシュ部品木」を提案する．提案システムはハッシュ部品木の生成，検証の二つのメインプロセスから構成される．ここで，ハッシュ部品木では，ハッシュ木のようにハッシュ値の連結を繰り返して 1 つのハッシュ値を求めるのではなく，ハッシュ値の排他的論理和（XOR）演算を繰り返すことで 1 つのハッシュ値を決定する．重複する部品の場合，一意性のあるパスを連結することで，XOR の自己反転性による打ち消しの発生を防ぐ．さらに，検証プロセスでは，XOR の自己反転性を応用して改ざん特定を実現する．また，提案システムの実用性を評価するため，二つのメインプロセスそれぞれの性能を調査した．

1.3 本稿の構成

本稿は以下の通り構成される．第 2 章では本研究の前提となる技術概要を説明する．第 3 章では，ハッシュ部品木の定義とそれを用いたデータ検証機能を提案する．第 4 章では，提案手法の有効性を確認するための実験とその結果を述べる．第 5 章では提案手法の実用性に関する考察を行い，第 6 章でまとめる．

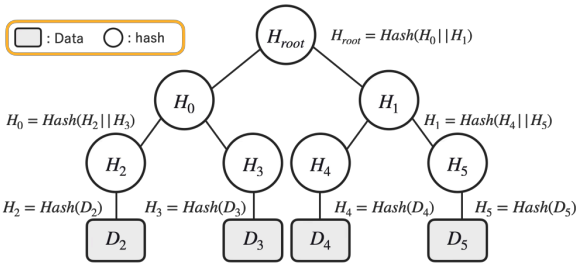


図 2: ハッシュ木の例

2. 関連知識

2.1 ブロックチェーン

ブロックチェーンは，データをブロック単位で時系列に連結した分散型台帳である．2008 年に Satoshi Nakamoto により投稿された論文 [3] に基づき発明された．各ブロックは前ブロックのハッシュ値を含み，チェーンのように連結するため，耐改ざん性に長けている．さらに，ブロックチェーンは参加者で分散管理するため，耐障害性と可用性に優れるといった特徴を持つ．

Ethereum に代表される一部のブロックチェーンには，スマートコントラクトが実装されている．スマートコントラクトとは，ブロックチェーン上で事前に指定されたルールを満たすと，自動的に契約が締結される仕組みである．仲介者を必要とせず，あらかじめプログラミングされた契約条件をもとに実行されるため，非常に効率的であるほか，高い透明性と耐改ざん性を持つ．本研究で使用する Hyperledger Iroha では組み込みコマンドとしてスマートコントラクトが導入されている．

2.2 ハッシュ木

ハッシュ木とは，ハッシュ値を格納する二分木のデータ構造である．Merkle-Tree とも呼ばれ，1979 年に Ralph C. Merkle によって発明された [4]．図 2 はデータ数 4 のハッシュ木の例である．葉ノード $H_2 - H_5$ にはデータ $D_2 - D_5$ のハッシュ値が格納される．中間ノードには子ノードのハッシュ値を連結しハッシュ化したものが葉から上方向に演算，格納される．図の例の場合，中間ノード H_0 の場合，H_2 と H_3 を連結してハッシュしたものとなる．最終的に Merkle ルート H_{root} が得られる．

ハッシュ木には，大規模データの整合性を高速に検証できるという特徴を持つ．n 個のデータから生成されたハッシュ木では，$log_2(n)$ 個のデータを用い，計算量 $O(log_2(n))$ で検証処理を行うことができる．

2.3 排他的論理和

排他的論理和 (以降, XOR) とは，二値のうち一方だけが真である場合のみ真を返す論理演算である．すなわち, $1 \oplus 1 = 0, 0 \oplus 0 = 0$ かつ $0 \oplus 1 = 1, 1 \oplus 0 = 1$ である．さらに，XOR は 4 つの性質を持つ．

（1） 単位元 : $a \oplus 0 = a$
（2） 自己反転性 : $a \oplus a = 0$
（3） 交換法則 : $a \oplus b = b \oplus a$
（4） 結合法則 : $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

中でも，本稿では (2) 自己反転性と (3) 交換法則を応用する．

3. 提 案 手 法

3.1 提案システム概要

図 3 は提案システムの全体像である．サプライチェーンに属する ASSEMBLER はローカル環境に Offchain-DB と Application を持つ．Offchain-DB には自社の製造部品の CFP と GHG が平文で保管される．Application では提案システムのメインプロセスを実行する．

ASSEMBLER はそれぞれの Iroha node で Hyperledger Iroha の Blockchain Network に参加する．Iroha node 上では Blockchain と Blockchain をもとに構成される Onchain-DB が動作しており，これらは全 Iroha node で同期される．Onchain-DB にはルートを自動車とするハッシュ部品木が保管される．

提案システムは，改ざんが特定可能であるような CFP 管理基盤を実現するための二つのメインプロセスを持つ．一つ目は，ハッシュ部品木の生成プロセスである．各部品のハッシュ部品木を生成し，CFP データを検証可能な形で保管する．二つ目は，ハッシュ部品木を用いた検証プロセスである．本研究の挑戦の一つである，Peer-to-peer 上でのデータ検証によって管理する CFP の改ざんを特定するプロセスである．次の節でそれぞれのプロセスの詳細を述べる．

3.2 定　　義

3.2.1 部　品　木

本稿では，製品を構成する部品の親子関係を木構造のように表現したものを部品木と呼ぶ．図 5 に示した例は部品 P_0 の部品木である．P_0 は $\{P_1, P_2\}$ で構成され，P_1 は $\{P_2, P_3(2\text{点})\}$ で構成される．P_2 のように，複数の構成部品に使用される部品を重複部品と呼ぶ．ここで，各部品は自身をルートとする部品木を形成しているが，全部品木を統合させるとルートは自動車となる．また，一つの部品木には異なる企業で製造された部品も含まれることに注意する．

3.2.2 CFP

本稿では自動車のライフラサイクルにおいて，製造部分にのみフォーカスする．よって，CFP は自動車部品製造に携わる企業の温室効果ガス排出量だけで算出する．図 4 は，本稿で定めた自動車部品製造業におけるスキーマであり，供給元 (SUPPLIER) から供給部品 (PARTSUPP) が供給され，組み立て工場 (ASSEMBLER) による組み立て工程 (ASSEMBLE PROCESS) を経て，親部品 (PARENTS PART) を構築する．供給部品 (PARTSUPP) から親部品 (PARENTS PART) を階層的に組み立てて，最上位部品の自動車を製造する．

部品 P_n の CFP を CFP_n，P_n の ASSEMBLER が製造過程を通して排出した温室効果ガス排出量を GHG_n とすると，図 5 に示した部品 P_0，P_1 の CFP_0，CFP_1 は次のように求める．

$$CFP_0 = GHG_0 + (CFP_1 + GHG_2)$$

$$CFP_1 = GHG_1 + (GHG_2 + GHG_3 \times 2)$$

3.2.3 ハッシュ部品木

ハッシュ部品木は，部品木に対応して生成される構造である．各部品にはハッシュ値が与えられるが，この値はハッシュ木の構成手順に着想を得て，複数のハッシュ値を XOR で組み合わせることで決定される．部品に対応する CFP 値やハッシュ値が変更された場合，他の部品におけるハッシュ値も必ず変化する．

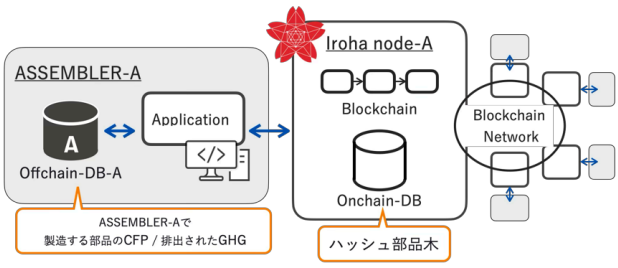


図 3: 提案システムの全体像

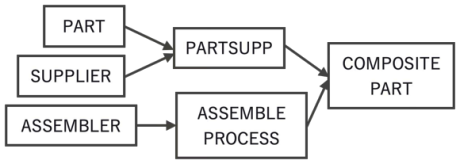


図 4: シナリオスキーマ

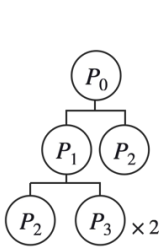


図 5: 部品木の例

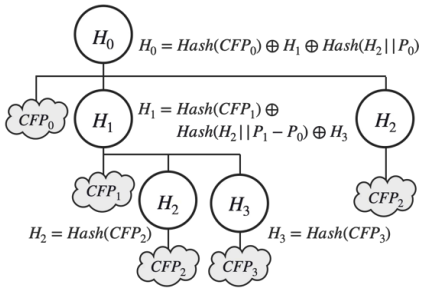


図 6: 図 5 に対するハッシュ部品木の値

一例として，図 5 の部品木に対し，図 6 のような対応するハッシュ部品木が構成される．ここで，ハッシュ部品木における各部品のハッシュ値の定義は，子部品を持たない単品部品 $\{P_2, P_3\}$ と，2 個以上の子部品を持つ構成部品 $\{P_0, P_1\}$ とで異なる．具体的なハッシュ値の定義は以下の通りである．ここで，$\boldsymbol{Hash}(x)$ は x にハッシュ関数 SHA256 を適用した結果を出力し，$\boldsymbol{Path}(x,y)$ は部品 x から部品 y までのパスを出力する．

- 任意の単品部品 P_s : $\boldsymbol{H_s = Hash(CFP_s)}$
- n 個の子部品 $\{P_{c-1}, P_{c-2}, \cdots, P_{c-n}\}$ $(2 \le i \le n)$

を持つ任意の構成部品 P_c :

$\boldsymbol{H_c = Hash(CFP_c) \oplus H_{c-1} \oplus H_{c-2} \oplus \cdots \oplus H_{c-n}}$

P_{c-i} が重複部品の場合，H_{c-i} を以下に置換する．

$\boldsymbol{Hash(H_{c-i}||Path(P_c, root))}$

ここで，2.2 節で紹介したハッシュ木では，2 つの子ノードのハッシュ値を順序を保った上で連結していく．しかしながら，本シナリオにおいては部品間に順序性が存在していないことを考慮し，ハッシュ値を順序に依存せずに統合する方法として，交換法則を持つ XOR を採用した．一方で，ハッシュ部品木に重複部品が含まれる場合，上位ノードにて重複部品同士で自己反転性が働いてしまう．そこで，重複部品の親部品からルート部品 (自動車) までのパスをハッシュ値に連結させる．このパスは一意性があるため，自己反転性を防ぎながら，改ざん特定も可能となる．

3.3 生成プロセス 処理手順

ハッシュ部品木生成プロセスは，CFP データを検証可能な形で保管するためのプロセスである．新規部品をシステムに追加する際など初期設定として実施する．

1. **集約**：システムに参加する全 ASSEMBLER の GHG を集約.
2. ***CFP* 算出**：1 の GHG から各部品の CFP を算出.
3. **ハッシュ部品木生成**：ハッシュ部品木の定義に従い，各部品のハッシュ値を算出.
4. **Onchain-DB へ write**：Iroha の組み込みコマンドで Onchain-DB に 3. のハッシュ値を格納．合意形成を経て Blockchain にブロックが追加される.
5. **Offchain-DB へ write**：4. が成功なら，2. の CFP を各 Offchain-DB に格納．失敗なら，プロセスの実行は undo される.

3.4 検証プロセス 処理手順

検証プロセスでは，任意の部品木の CFP データの改ざんの有無を検証する．Onchain-DB 上に格納されたハッシュ部品木を用いるため，前提としてハッシュ部品木生成プロセスが 1 度以上実行されている必要がある．また，改ざんは Offchain-DB でのみ発生するとする．

ハッシュ部品木の検証プロセスの流れを以下に示す.

1. **既存のハッシュ部品木取得**：Onchain-DB から比較対象のハッシュ部品木取得.
2. **新規ハッシュ部品木生成**：生成プロセスと同様にハッシュ部品を生成.
3. **比較**：1. と 2. のハッシュ部品木のルートハッシュのみを比較．一致の場合，検証成功で終了．不一致の場合，ハッシュ部品木全体を比較し，ハッシュ値が不一致の部品 (改ざん疑義部品と呼ぶ) を抽出.
4. **改ざん特定**：3. で得られた改ざん疑義部品を入力として Algorithm 1 を実行.

手順 4 の改ざん特定処理に進むとき，入力として手順 3 で得られた改ざん疑義部品を与える．改ざん疑義部品には，実際に改ざんが発生した部品と，ハッシュ部品木を生成する過程で改ざんの影響を受けた部品が存在する．この 2 種を判別すべく，XOR の自己反転性で改ざんの影響の除去を行う．ここで，改ざんの影響は，改ざん発生部品からルート部品までのパス上に存在する全ての部品に及ぶ．そのため改ざん疑義部品とは，改ざん発生部品からルート部品までのパスの集まりでもある.

以下に Algorithm 1 改ざん特定を示す．ここで，関数 $GetCorrectHash(x)$ は Onchain-DB に接続して部品 x のハッシュ値を取得する関数，関数 $Xor()$ は引数に与えた 16 進数同士を XOR する関数，関数 $Hash()$ は引数をハッシュ化する関数，関数 $Path(x, y)$ は部品 x から部品 y までのパスを返す関数である．さらに，図 7 には，Algorithm 1 の挙動例を示した.

Algorithm 1: 改ざん特定

入力: 改ざん疑義部品のパス列 P (二次元配列)
出力: 改ざん確定部品集合 C

レコード:
    PartInfo(文字列型 : part_id, バイナリ型 : hash, PartInfo : parents_part)
$C \leftarrow \emptyset$
$P_{next} \leftarrow [\,]$ // 次の疑義部品パス
while P **の要素数が** 0 **でない do**
    $P_{\text{now}} \leftarrow \text{pop}(P)$ // パスを一つ取得
    $target \leftarrow \text{pop}(P_{\text{now}})$ // パスの始点を取得
    $C \leftarrow C \cup \{target\}$ // 改ざん確定部品
    $ChackedHash \leftarrow \text{GetCorrectHash}(target)$
    $parent \leftarrow target.parents\_part$ // target の親部品
    repeat
        /* XOR で改ざん (target) の影響を除去 */
        if $target$ **が重複部品 then**
            $ChackedHash \leftarrow \text{Xor}\Big(parent.hash,$
            $\text{Hash}(target.hash \,\|\, \text{Path}(target, Root)),$
            $\text{Hash}(ChackedHash \,\|\, \text{Path}(target, Root))\Big)$
        else
            $ChackedHash \leftarrow$
            $\text{Xor}(parent.hash, target.hash, ChackedHash)$
        /* 改ざん (target) を除去後に正確な値と再比較 */
        if $ChackedHash \neq \text{GetCorrectHash}(parent)$ **then**
            $P_{next}.\text{append}(parent)$
        $parent.hash \leftarrow ChackedHash$
        $target \leftarrow parent$
    until $parent = Root$ **部品までパスを辿る**
    $P.\text{append}(P_{next})$
return C

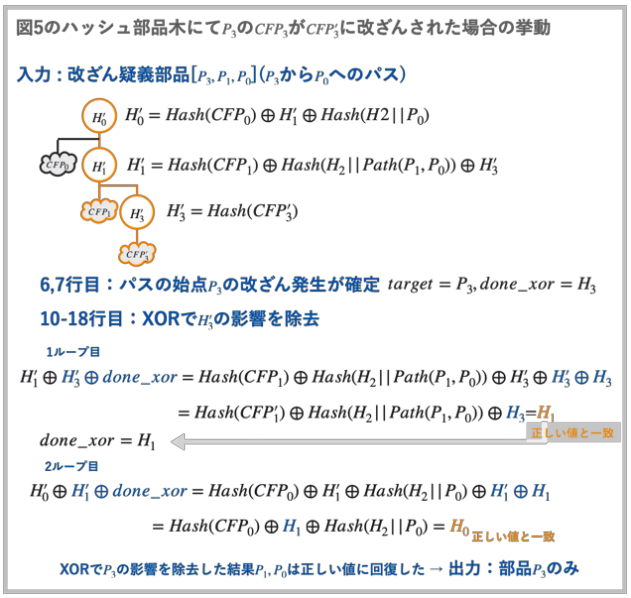


図 7: 図 6 を用いた Algorithm 1 の挙動例

4. 評　　価

本節では，提案システムの改ざん特定の性能を調査した.

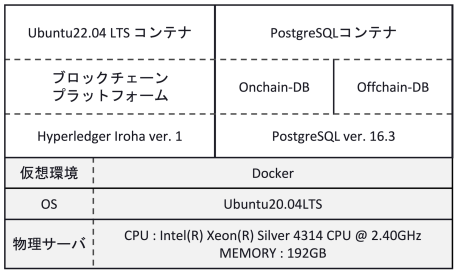


図 8: 構築環境

メインプロセスは，Application (Python) から Onchain-DB (PostgreSQL) 上の一時テーブルに SQL クエリを発行することで実装している．また，生成プロセスでの組み込みコマンド処理は，C++ で記述された Iroha の既存の組み込みコマンドを用いず，本提案手法に特化した新たなコマンドを開発した．

4.1 概　　要

制約のもとでランダム生成した部品木を対象に，二つのメインプロセスの実行時間を計測した．制約は以下の通りである．

- 総部品数は 30/300/3,000/30,000 点.
- 部品の重複率は 0/10/20/30/40% .

総部品数・重複率の組み合わせごとに 3 種ずつ部品木を生成した．ここで，総部品数 30,000 点は自動車一台に相当するとされる．

実装の構築環境とマシンの性能は図 8 に示した．より実運用に近い環境を求める場合は実ネットワークでの評価が妥当であるが，Iroha ネットワークの性能は実ネットワークと仮想ネットワークで大きな差が見られないため [5]，本稿では Docker 環境を採用している．また，参加する ASSEMBLER ごとに Iroha をビルドする Ubuntu 22.04 LTS コンテナと Onchain/Offchain-DB の機能を提供する PostgreSQL コンテナをセットで構築する．本実験では，システムに参加する ASSEMBLER 数を簡易的に 3 社とした．

4.2 生成プロセス 実行時間調査

まず，総部品数と重複率の組み合わせごとに各 3 種，計 60 個の部品木に対応するハッシュ部品木の生成プロセス実行時間を 3 回ずつ計測した．総部品数・重複率ごとに実行時間の平均値を算出し，図 9 に示した．

総部品数 30 と 300 点では重複率増加につれ実行時間は増加傾向で 30 と 300 点では重複処理が実行時間増加に影響を与えている．一方で，総部品数 3,000 と 30,000 点では重複率増加につれ実行時間は減少傾向であり，重複処理による実行時間増加が処理対象の部品数の減少の効果が上回っている．

次に，Iroha 組み込みコマンド処理のオーバーヘッドを調査した．Iroha 組み込みコマンド処理以外の処理を Application 処理とし，実行時間の内訳を図 10 に示した．

Iroha 組み込みコマンド処理は生成プロセス全体の約 70-81% を締める結果となった．よって，Iroha 組み込みコマンドのオーバーヘッドが，30 と 300 点の実行時間に差が生まれない原因として考えられる．さらに，Iroha 組み込みコマンド処理時間が生成プロセス実行時間全体に与える影響は大きいものの，生成プロセスの実行頻度は低いことから，実用性への影響は低いだろう．

4.3 検証プロセス 実行時間調査

まず，検証プロセスの結果が整合，すなわち検証対象の部品木に改ざんがない場合の実行時間を調査した．4.2 節で使用した部品木に対する検証プロセスの実行時間を 3 回ずつ計測した．総部品数・重複率ごとの平均値を図 11 に示す．

総部品数 30 と 300 点では，それぞれの実行時間の変動係数は 1.94%, 2.07% であり，値のばらつきは非常に小さい結果となった．これに対し，3,000 と 30,000 点では重複率に対して実行時間の減少傾向が見られる．また，同じ部品木の生成プロセス実行時間と比較すると，約 0.15-0.23 倍となった．この差の要因は，生成プロセスに Iroha 組み込みコマンド実行が含まれることが考えられる．

次に，検証プロセスの結果が不整合，すなわち検証対象の部品木に改ざんが発生した場合の実行時間を調査した．4.2 節で使用した部品木 (総部品数 30 点を除く) に含まれる部品のうち，1/5/10/15% の部品をランダムに選んで CFP データを別の数値に改ざんした．改ざん済みの部品木に対する検証プロセスの実行時間を 3 回ずつ計測し，重複率ごとの平均値を総部品数別に図 12 に示した．

図 12(a)(b)(c) いずれも，改ざん率の増加に対して実行時間も増大している．これは改ざん特定処理の繰り返す回数が増えるためと考えられる．また，同時に一度の実行での改ざん部品の特定率を調査したところ，いずれも特定率 100% となり，全ての改ざん部品を検出して期待通りの挙動を得られた．

5. 考　　察

本節では，提案手法についての考察を行う．

5.1 ハッシュ部品木の改ざん耐性に関する議論

本稿では XOR を応用しハッシュ部品木を構築することで，改ざん検知・特定を実現した．本節ではハッシュ部品木の耐改ざん性について考察する．前提としてハッシュ部品木が保管される Onchain-DB は，ブロックチェーンの性能により高い信頼性を持つ．以降の議論は Onchain-DB に保管されたハッシュ値の正当性が常に担保されているという前提で進める．

ハッシュ部品木の構造においては，各ノードのハッシュ値は，子ノードのハッシュ値と自身の CFP 値のハッシュとの XOR により構成される．この構造により，部品木内で CFP 値の改ざんが発生した場合，必ずそのノードおよび上位ノードのハッシュ値に差異が生じるため，改ざん検知が可能となる．特に，本手法では，任意ノードの改ざんはルートまでのパス上の全ノードに影響を与える．改ざんが伝播するパスを追跡することで，改ざん箇所を特定できるほか，パスの中間ノードの CFP が改ざんされた場合も，XOR の不整合から当該ノードにおける異常を検出することができる．

一方で，複数箇所の同時改ざんにより，XOR 結果が偶然一致し，上位ノードでハッシュ値の自己反転性が働くという懸念が考えられる．しかし，本構成では CFP 値が実数として表現されており，複数の改ざんを意図的に XOR で打ち消すには高精度の制御が必要であること，Onchain-DB 上のハッシュ値の改ざ

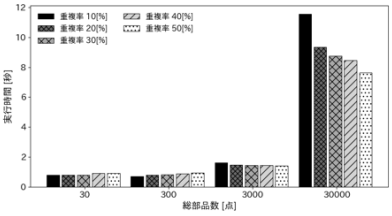


図 9: 生成プロセス 実行時間

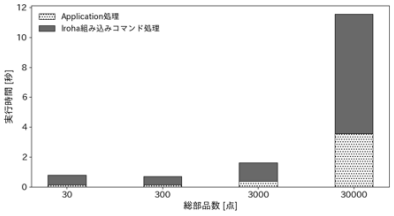


図 10: 生成プロセス 実行時間の内訳

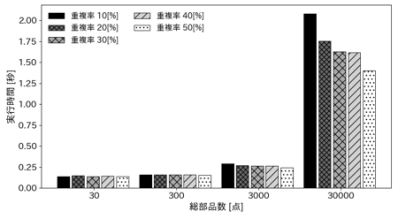
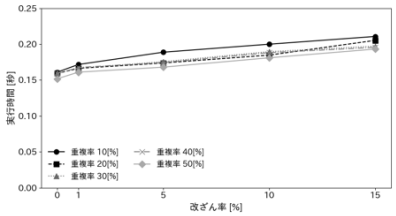
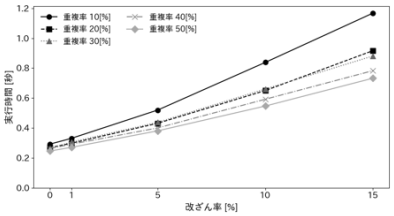


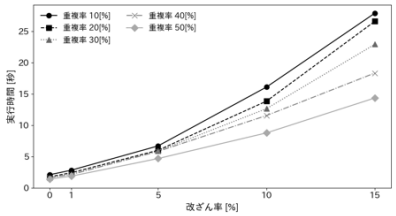
図 11: 検証プロセス 実行時間 (整合)



(a) 総部品数 300 点



(b) 総部品数 3,000 点



(c) 総部品数 30,000 点

図 12: 検証プロセス 実行時間 (不整合)．縦軸スケールは不統一であることに注意．

んが困難であることから，実用上その可能性は極めて低いと考えられる．また，衝突耐性を有する SHA-256 を用いており，異なる CFP から同一ハッシュが生成される可能性も現実的には無視できる．従って，ハッシュ値自体の信頼性も高い．

以上より，提案するハッシュ部品木構造は，XOR を活用することで改ざんの検出だけでなく箇所の特定まで実現可能であり，Onchain-DB に記録されたハッシュ値の正当性が担保されている限り，高い耐改ざん性を有する設計であると評価できる．

5.2 先行研究との比較

先行研究 [2] との比較を行う．先行研究では，部品木内の部品の CFP の算出手法，CFP 改ざんの有無を検知する手法を提案した．これに対し本稿では，より高い安全性と検証機能の高性能化を目指して，ハッシュ部品木を提案しており，先行研究では不可能であった CFP の改ざん特定が可能となった．実行速度の観点では，本稿第 4.3 節より検証プロセスが最大で 27.88 秒程度かかったが，同条件での先行研究の CFP 算出処理は 9.52 秒である．処理量の増加を考えれば妥当と考えられる．実装の観点では，先行研究の実装の大半は，1 つの Iroha 組み込みコマンド内で完結している．これに対し，本稿の実装では Python を利用した Application 処理を主としている．これにより，組み込みコマンド処理は開発の制約が多いが，Application での実装は自由度が高いことからメンテナンスが容易となった．以上より，本稿の提案システムは，先行研究と比較し，より実用性に優れていると考える．

6. まとめと今後の課題

本研究では，先行研究で開発した自動車部品の CFP を管理する分散データベースシステムの実用性と安全性を高めるべく，ハッシュ化した CFP を部品の構造に従って XOR で統合することで，各部品にハッシュ値を持たせる「ハッシュ部品木」を提案した．XOR の交換法則によって部品間の順序性の情報を省略できる他，改ざんが発生した場合に具体的な発生箇所を XOR の自己反転性で特定することが可能となった．重複部品に対しては，パスを連結させることで，自己反転性による打ち消しに対応した．評価実験より提案手法の実装は期待通りの挙動を示すことが確認され，特に検証プロセスの改ざん特定率はいずれも 100% となった．

今後は，Onchain-DB に保管したハッシュ部品木の分散を行っていく．現在の実装では，Onchain-DB は全 Iroha node で冗長であり，ルートを自動車とするハッシュ部品木が管理されている．すなわち，Onchain-DB のハッシュ部品木には，任意の ASSEMBLER の製造部品に関連する部品以外のデータも多く含まれており，不要なデータを保管することによる安全性の懸念，各分散ノードの負荷増加という側面において改善の余地がある．Blockchain より構築される Onchain-DB の耐改ざん性に優れるという利点をより活用するためにも，新たなアプローチの検討・実装を行っていく．

謝辞　本研究は一部，JST CREST JPMJCR22M2 の支援を受けたものである．

文　献

[1] CREST：検証可能なデータエコシステム．入手先 ⟨https://www.kde.cs.tsukuba.ac.jp/crest/index.html⟩ (参照 2024-12-30).
[2] 堀遥, Le Hieu Hanh, 小口正人. ブロックチェーンベースのカーボンフットプリントデータ管理基盤の構築と自動車部品製造業への応用における評価. In *DEIM2025 8C-01*, 2025.
[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
[4] Ralph C Merkle. A digital signature based on a conventional encryption function. *Conference on the Theory and Application of Cryptographic Techniques*, 1987.
[5] 坂本明穂, 小口正人. 実ネットワークで接続されたデータ検証可能な分散データベースの性能に関する検討. マルチメディア，分散，協調とモバイル (DICOMO2024) シンポジウム, 2024.