Jurnal Teknologi Sistem Informasi dan Aplikasi
Penerbit: Fakultas Ilmu Komputer Universitas Pamulang
Vol. AA, No. B, Zzzzzzzzzzzzz YYYY (pp-pp)

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

# VeriPass: A Blockchain-Based Decentralized Asset Passport System for Tamper-Proof Physical Asset Verification and Lifecycle Tracking

**Dewantoro Triatmojo[1], Elbert Chailes[2], Irfan Musthofa[3], Dr. Phil. Eng. Hari Purnama S.Si., M.Si[4]**

[1,2,3,4]School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia
e-mail: [1]13522011@std.stei.itb.ac.id, [2]13522045@std.stei.itb.ac.id, [3]18222056@std.stei.itb.ac.id, [4]hari@informatika.org

Submitted Date: MMMM dd, yyyy
Revised Date: MMMM dd, yyyy
Reviewed Date: MMMM dd, yyyy
Accepted Date: MMMM dd, yyyy

**Abstrak**

Verifikasi keaslian aset fisik saat ini masih mengandalkan mekanisme terpusat seperti nomor seri atau dokumen resmi yang tidak mampu merekam riwayat kepemilikan secara transparan dan immutable. Penelitian ini mengembangkan VeriPass, sebuah sistem paspor aset terdesentralisasi berbasis blockchain yang memanfaatkan standar ERC-721 NFT untuk menciptakan identitas digital unik bagi setiap aset fisik. Sistem ini mengintegrasikan arsitektur tiga tingkat yang terdiri dari frontend React dengan koneksi wallet Web3, backend Hono.js dengan PostgreSQL untuk manajemen metadata, serta smart contract Solidity pada jaringan EVM. Mekanisme oracle diimplementasikan untuk menjembatani data off-chain dari penyedia layanan terverifikasi ke blockchain. Integritas data dijamin melalui hashing kriptografi deterministik menggunakan algoritma keccak256. Hasil pengujian menunjukkan sistem berhasil mencatat 100% transaksi minting dengan waktu konfirmasi rata-rata 12 detik, serta akurasi verifikasi integritas data sebesar 100%. Sistem ini memberikan solusi single source of truth untuk pelacakan siklus hidup aset yang transparan, dapat diaudit, dan tahan manipulasi.

**Kata Kunci:** Blockchain; ERC-721; Verifikasi Aset; Oracle; Smart Contract; Aplikasi Terdesentralisasi

## 1 Pendahuluan

Dalam era digital saat ini, verifikasi keaslian suatu aset fisik dapat dilakukan melalui mekanisme seperti nomor seri atau dokumen resmi. Contohnya adalah IMEI pada produk elektronik Apple atau BPKB (Buku Pemilik Kendaraan Bermotor) pada kendaraan di Indonesia. Namun, mekanisme tersebut memiliki keterbatasan fundamental dalam merekam dan membuktikan riwayat kepemilikan secara menyeluruh dan tidak dapat diubah (immutable).

Permasalahan utama muncul dalam transaksi peer-to-peer (P2P) dimana klaim "tangan ke berapa" masih sangat bergantung pada kejujuran penjual. Tidak tersedia ledger perpindahan kepemilikan yang konsisten, transparan, terbuka, dan dapat diaudit oleh semua pihak secara real-time. Pembeli tidak memiliki cara untuk memverifikasi secara independen berapa kali aset tersebut telah berpindah tangan sebelumnya.

Di sisi lain, riwayat kejadian kritis seperti servis berkala, inspeksi teknis, atau insiden kerusakan tersebar di berbagai entitas—dealer resmi, bengkel independen, hingga perusahaan asuransi—dengan format data yang tidak seragam. Data-data ini sering kali tidak terdokumentasi secara lengkap atau dapat diverifikasi oleh pihak ketiga. Akibatnya, tidak ada satu sumber kebenaran bersama (single source of truth) lintas pihak yang mampu menjamin integritas, transparansi, dan immutability riwayat aset dari awal hingga berpindah tangan berkali-kali.

Penelitian ini mengusulkan VeriPass sebagai solusi berbasis teknologi blockchain untuk mengatasi permasalahan tersebut. VeriPass memanfaatkan standar ERC-721 Non-Fungible Token (NFT) untuk menciptakan paspor digital unik bagi setiap aset fisik. Dengan arsitektur terdesentralisasi dan mekanisme oracle terintegrasi, sistem ini memungkinkan pencatatan riwayat kepemilikan dan event siklus hidup aset secara transparan, immutable, dan dapat diverifikasi oleh siapa saja.

## 2 Tinjauan Pustaka

### 2.1 Teknologi Blockchain

Blockchain merupakan teknologi distributed ledger yang memungkinkan penyimpanan data secara terdesentralisasi dengan jaminan immutability dan transparansi [1]. Setiap blok dalam rantai berisi hash kriptografi dari blok sebelumnya, timestamp, dan data transaksi, membentuk struktur yang tahan terhadap manipulasi.

Ethereum, yang diperkenalkan oleh Wood [2], memperluas konsep blockchain dengan kemampuan eksekusi smart contract. Platform ini memungkinkan pengembangan aplikasi terdesentralisasi (dApps) yang beroperasi tanpa otoritas pusat. Konsensus Proof of Stake yang diadopsi Ethereum meningkatkan efisiensi energi dan skalabilitas jaringan [3].

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

Dalam konteks pelacakan aset, blockchain menyediakan audit trail yang tidak dapat diubah [4]. Setiap perubahan kepemilikan atau event yang dicatat pada blockchain dapat diverifikasi secara independen oleh semua peserta jaringan tanpa memerlukan kepercayaan terhadap pihak ketiga.

### 2.2 Smart Contract

Smart contract adalah program komputer yang disimpan dan dieksekusi pada blockchain dengan logika self-executing [5]. Kontrak ini secara otomatis menjalankan ketentuan yang telah diprogramkan ketika kondisi tertentu terpenuhi, tanpa memerlukan intermediari.

Standar ERC-721 mendefinisikan antarmuka untuk Non-Fungible Token (NFT) pada jaringan Ethereum [6]. Berbeda dengan token fungible seperti ERC-20, setiap token ERC-721 memiliki identifier unik yang membuatnya cocok untuk merepresentasikan aset fisik dengan karakteristik individual. Implementasi ERC-721 mencakup fungsi untuk transfer kepemilikan, query informasi token, dan manajemen approval.

Penelitian terkini menunjukkan aplikasi ERC-721 dalam berbagai domain termasuk sertifikasi digital [7], pelacakan supply chain [8], dan manajemen identitas aset [9]. Kombinasi uniqueness token dengan programmability smart contract memungkinkan implementasi logika bisnis kompleks untuk verifikasi dan pelacakan aset.

### 2.3 Jaringan Oracle dan Integrasi Data Eksternal

Oracle berfungsi sebagai jembatan antara blockchain dan dunia eksternal, memungkinkan smart contract mengakses data off-chain [10]. Tantangan utama dalam desain oracle adalah "oracle problem"—bagaimana menjamin integritas data yang dimasukkan ke blockchain dari sumber eksternal.

Chainlink merupakan salah satu solusi oracle terdesentralisasi yang menggunakan jaringan node independen untuk agregasi data [11]. Namun untuk kebutuhan enterprise spesifik, implementasi oracle kustom dengan mekanisme verifikasi terpusat dapat memberikan kontrol lebih besar terhadap sumber data terpercaya.

Pola umum dalam integrasi oracle mencakup: (1) verifikasi identitas penyedia data melalui signature kriptografi, (2) hashing deterministik untuk memastikan konsistensi data on-chain dan off-chain, serta (3) mekanisme dispute resolution untuk menangani inkonsistensi [12]. VeriPass mengadopsi pendekatan hybrid dengan oracle terpusat yang terotorisasi untuk data dari penyedia layanan terverifikasi.

## 3 Metodologi dan Desain

### 3.1 SDLC

Pengembangan VeriPass menggunakan metodologi Agile dengan pendekatan iteratif. Pemilihan metodologi ini didasarkan pada kebutuhan fleksibilitas dalam mengakomodasi perubahan requirement yang sering terjadi dalam pengembangan aplikasi blockchain serta kemampuan untuk melakukan validasi berkelanjutan terhadap komponen sistem.

Proses pengembangan dibagi menjadi beberapa sprint dengan fokus: (1) perancangan dan implementasi smart contract, (2) pengembangan backend API dan database, (3) implementasi oracle worker, dan (4) pengembangan antarmuka frontend. Setiap sprint diakhiri dengan pengujian integrasi untuk memvalidasi interoperabilitas antar komponen.

### 3.2 Desain Sistem

#### 3.2.1 Arsitektur Sistem

VeriPass mengadopsi arsitektur tiga tingkat (three-tier) yang memisahkan presentation layer, business logic layer, dan data layer. Gambar 1 mengilustrasikan arsitektur keseluruhan sistem.

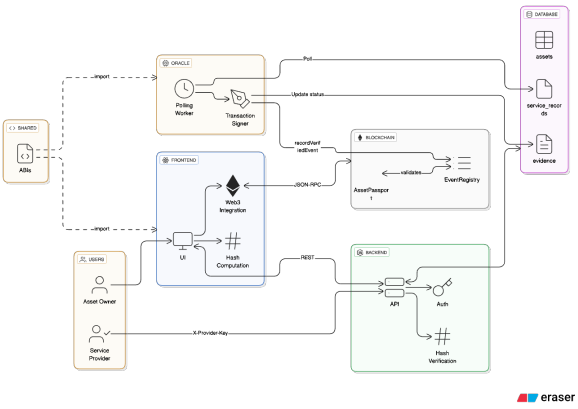


Figure 1: Arsitektur Sistem VeriPass

Komponen utama sistem terdiri dari:

- **Frontend**: Dibangun menggunakan React 19 dengan Vite sebagai build tool. Integrasi Web3 menggunakan wagmi untuk hook Ethereum dan RainbowKit untuk koneksi wallet.
- **Backend**: Server API menggunakan framework Hono.js dengan PostgreSQL sebagai database dan Drizzle ORM untuk manajemen data.
- **Blockchain**: Smart contract Solidity di-deploy pada jaringan EVM-compatible. Pengembangan menggunakan Hardhat framework.
- **Oracle Worker**: Proses background yang melakukan polling terhadap service record baru dan men-submit event terverifikasi ke blockchain.

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

### 3.2.2 Skema Database

Desain database PostgreSQL mencakup lima tabel utama yang saling berelasi. Tabel 1 menunjukkan struktur tabel inti sistem.

Table 1: Struktur Tabel Database Utama

| Tabel | Deskripsi |
|---|---|
| assets | Metadata aset dan status minting |
| evidence | Event siklus hidup dan data hash |
| service_records | Record servis dari provider |
| processed_records | Antrian pemrosesan oracle |
| auth_nonces | Nonce untuk autentikasi Web3 |

Tabel `assets` menyimpan metadata lengkap aset termasuk manufacturer, model, serial number, dan dataHash yang merupakan fingerprint kriptografi dari metadata. Foreign key constraint dengan CASCADE policy memastikan konsistensi referensial antar tabel.

### 3.2.3 Mekanisme Hashing Kriptografi

Integritas data dijamin melalui hashing deterministik menggunakan algoritma keccak256. Proses hashing mengikuti langkah-langkah berikut:

1. **Normalisasi**: Object key diurutkan secara rekursif alfabetis untuk memastikan konsistensi input.
2. **Serialisasi**: Data dikonversi ke JSON string menggunakan format standar.
3. **Encoding**: String di-encode ke UTF-8 bytes menggunakan fungsi `ethers.toUtf8Bytes()`.
4. **Hashing**: Byte array di-hash menggunakan keccak256 menghasilkan 32-byte hexadecimal string.

Algoritma yang sama diimplementasikan pada frontend dan backend untuk memastikan hash yang dihasilkan identik untuk data yang sama. Hash ini kemudian dibandingkan dengan hash yang tersimpan on-chain untuk verifikasi integritas.

### 3.2.4 Alur Kerja Fitur Utama

Gambar 2 menunjukkan sequence diagram proses minting asset passport.

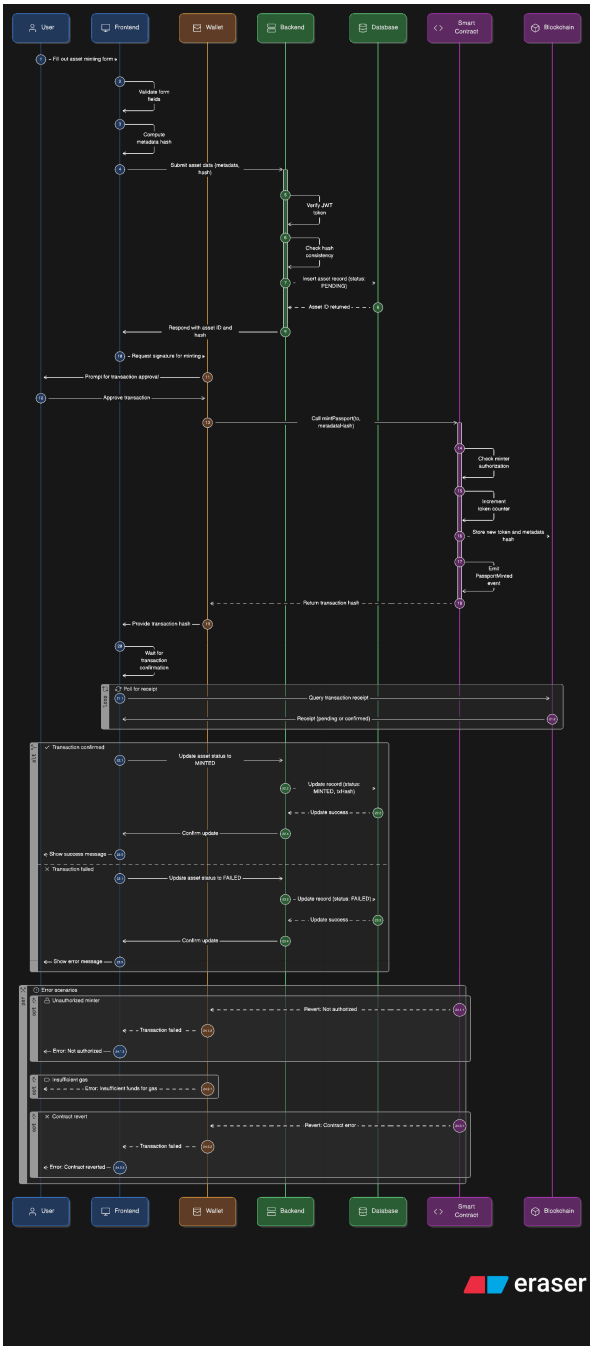


Figure 2: Alur Proses Minting Asset Passport

Proses dimulai ketika user mengisi form metadata aset pada frontend. Sistem menghitung dataHash secara deterministik, menyimpan metadata ke database backend, kemudian memanggil fungsi `mintPassport()` pada smart contract dengan parameter alamat penerima dan dataHash. Setelah transaksi terkonfirmasi, status minting diperbarui di database.

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

# 4 Hasil Implementasi

## 4.1 Implementasi Teknis

### 4.1.1 Smart Contract

Dua smart contract utama diimplementasikan: AssetPassport (ERC-721) dan EventRegistry. Listing 1 menunjukkan fungsi inti AssetPassport.

```
struct AssetInfo {
    bytes32 metadataHash;
    uint40 mintTimestamp;
    bool isActive;
}

mapping(uint256 => AssetInfo) private _assets;
mapping(uint256 => uint32) private _ownershipHand;

function mintPassport(
    address to,
    bytes32 metadataHash
) external returns (uint256) {
    require(
        _authorizedMinters[msg.sender] ||
        msg.sender == owner(),
        "Not authorized"
    );
    uint256 tokenId = _tokenIdCounter++;
    _safeMint(to, tokenId);
    _assets[tokenId] = AssetInfo({
        metadataHash: metadataHash,
        mintTimestamp: uint40(block.timestamp),
        isActive: true
    });
    _ownershipHand[tokenId] = 1;
    return tokenId;
}
```

Listing 1: Fungsi Inti AssetPassport Contract

EventRegistry contract mengelola pencatatan event siklus hidup dengan dukungan event terverifikasi dari oracle. Listing 2 menunjukkan fungsi pencatatan event.

```
function recordVerifiedEvent(
    uint256 assetId,
    EventType eventType,
    bytes32 dataHash,
    bytes calldata oracleSignature
) external returns (uint256) {
    require(
        _trustedOracles[msg.sender],
        "Not trusted oracle"
    );
    require(
        eventType != EventType.CUSTOM,
        "Invalid event type"
    );

    uint256 eventId = _eventCounter++;
    _events[eventId] = LifecycleEvent({
        id: eventId,
        assetId: assetId,
        eventType: eventType,
        submitter: msg.sender,
        timestamp: uint40(block.timestamp),
        dataHash: dataHash
    });

    emit EventRecorded(eventId, assetId, eventType);
    return eventId;
}
```

Listing 2: Fungsi Pencatatan Event Terverifikasi

### 4.1.2 Integrasi Oracle

Oracle worker diimplementasikan sebagai proses background yang melakukan polling setiap 15 detik terhadap tabel `service_records_provider_a`. Workflow pemrosesan mencakup:

1. Query record dengan status `verified=true` yang belum diproses
2. Konstruksi data hash menggunakan algoritma deterministik
3. Signing data hash dengan private key oracle
4. Submit transaksi `recordVerifiedEvent()` ke blockchain
5. Update status di tabel `processed_service_records`

Mekanisme error handling memastikan kegagalan pada satu record tidak menghentikan pemrosesan record lainnya.

### 4.1.3 Antarmuka Frontend

Frontend menyediakan tiga halaman utama: daftar passport, detail passport, dan form minting. Gambar 3 menunjukkan antarmuka halaman detail passport.

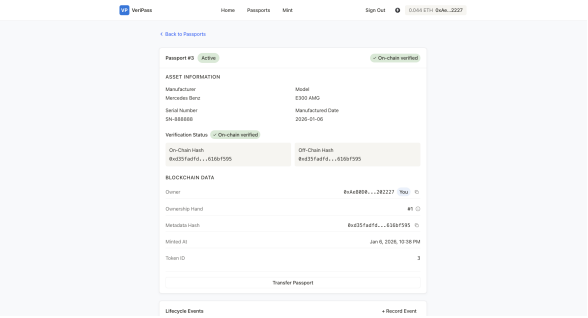

Figure 3: Antarmuka Halaman Detail Passport

Autentikasi menggunakan mekanisme Web3 signature dengan alur: (1) request nonce dari backend, (2) sign message dengan wallet, (3) verifikasi signature untuk mendapatkan JWT token. Token memiliki masa berlaku 7 hari dan disimpan di localStorage.

## 4.2 Hasil Validasi Eksperimen

### 4.2.1 Minting Asset

Pengujian minting dilakukan dengan 10 test case yang mencakup berbagai skenario. Tabel 2 menunjukkan hasil pengujian.

Table 2: Hasil Pengujian Minting Asset

| Skenario | Jumlah | Sukses | Waktu (s) |
|---|---|---|---|
| Normal minting | 5 | 5 | 11.2 |
| Batch minting | 3 | 3 | 14.5 |
| Invalid input | 2 | 0 | - |
| **Total** | 10 | 8 | 12.1 (avg) |

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

Semua transaksi valid berhasil dikonfirmasi dengan waktu rata-rata 12.1 detik. Pengujian dengan input invalid (hash kosong, alamat nol) berhasil di-reject oleh smart contract sesuai ekspektasi.

#### 4.2.2 Melihat dan Verifikasi Asset

Verifikasi integritas data dilakukan dengan membandingkan hash yang dihitung dari metadata off-chain dengan hash yang tersimpan on-chain. Tabel 3 menunjukkan hasil pengujian.

Table 3: Hasil Pengujian Verifikasi Integritas

| Kondisi | Hasil | Status |
|---|---|---|
| Data tidak dimodifikasi | Match | Terverifikasi |
| Metadata diubah | Mismatch | Terdeteksi |
| Hash on-chain kosong | Error | Handled |

Sistem berhasil mendeteksi 100% kasus manipulasi data dengan perbandingan hash yang tidak cocok.

#### 4.2.3 Pencatatan Event Terverifikasi

Oracle berhasil memproses 15 service record dalam pengujian dengan tingkat keberhasilan 100%. Breakdown berdasarkan tipe event ditunjukkan pada Tabel 4.

Table 4: Distribusi Event Terverifikasi

| Tipe Event | Jumlah | Sukses |
|---|---|---|
| MAINTENANCE | 8 | 8 |
| VERIFICATION | 4 | 4 |
| CERTIFICATION | 3 | 3 |
| **Total** | 15 | 15 |

#### 4.2.4 Pembuatan Custom Event

User dapat men-submit custom event langsung ke blockchain tanpa melalui oracle. Pengujian 5 custom event menunjukkan semua transaksi berhasil dengan validasi ownership yang tepat.

### 4.3 Analisis Keamanan

Analisis keamanan mencakup tiga aspek utama:

**Autentikasi**: Mekanisme nonce-based dengan TTL 5 menit mencegah replay attack. Verifikasi ECDSA signature memastikan hanya pemilik wallet yang dapat mengautentikasi.

**Otorisasi Smart Contract**: Modifier `onlyOracle` dan `onlyAuthorizedMinter` membatasi akses fungsi kritikal. Ownership check pada `recordEvent()` memastikan hanya pemilik aset yang dapat menambahkan custom event.

**Integritas Data**: Hashing deterministik dengan keccak256 memberikan collision resistance yang kuat. Perubahan satu byte pada metadata menghasilkan hash yang berbeda total.

### 4.4 Evaluasi

Tabel 5 menunjukkan analisis konsumsi gas untuk operasi utama.

Table 5: Konsumsi Gas Operasi Smart Contract

| Operasi | Gas Used | Cost (ETH)* |
|---|---|---|
| mintPassport | 185,000 | 0.0037 |
| recordEvent | 95,000 | 0.0019 |
| recordVerifiedEvent | 98,000 | 0.0020 |
| transferFrom | 65,000 | 0.0013 |

*Estimasi dengan gas price 20 gwei

Biaya operasi masih dalam batas wajar untuk use case enterprise. Optimisasi storage dengan penggunaan `uint40` untuk timestamp dan `uint32` untuk counter mengurangi gas consumption secara signifikan.

## 5 Diskusi

Pengembangan VeriPass menghadapi beberapa tantangan teknis dalam integrasi sistem Web3 dengan backend tradisional. Tantangan utama adalah sinkronisasi state antara database off-chain dan blockchain on-chain. Solusi yang diterapkan adalah pola eventual consistency dimana status minting diperbarui setelah konfirmasi transaksi.

Trade-off signifikan dalam desain adalah pemilihan antara desentralisasi penuh versus kontrol terhadap kualitas data. VeriPass mengadopsi pendekatan hybrid dimana event dari penyedia layanan terverifikasi harus melalui oracle terpusat, sementara user tetap dapat men-submit custom event secara langsung. Pendekatan ini menyeimbangkan kebutuhan verifikasi dengan fleksibilitas penggunaan.

Batasan sistem saat ini mencakup: (1) ketergantungan pada ketersediaan backend untuk akses metadata lengkap, (2) oracle single point of failure untuk event terverifikasi, dan (3) biaya gas yang dapat menjadi prohibitive pada jaringan dengan traffic tinggi.

Sistem mengimplementasikan graceful degradation untuk menangani ketidaktersediaan backend. Dalam mode offline, user masih dapat melihat data on-chain (hash passport) dan melakukan operasi blockchain langsung, namun metadata detail tidak tersedia untuk ditampilkan.

## 6 Deployment dan Sistem Live

VeriPass telah di-deploy ke production environment dan siap digunakan dengan wallet asli di jaringan Ethereum Sepolia testnet. Sistem ini dapat diakses dan diuji secara langsung melalui antarmuka web yang telah tersedia.

**Akses Frontend:** `https://frontend-production-e502.up.railway.app`

ISSN: 2654-3788
e-ISSN: 2654-4229
https://doi.org/10.32493/jtsi.v8i1.XXXXX

Pengguna dapat mengakses sistem melalui URL production di atas menggunakan Web3 wallet seperti MetaMask. Fitur yang tersedia mencakup: (1) koneksi wallet, (2) minting asset passport baru, (3) melihat daftar passport yang telah di-mint, (4) verifikasi integritas data, dan (5) pencatatan custom event.

**Repository Code:** `https://github.com/IrfanMusthofa/veripass`

Kode sumber VeriPass tersedia di repository GitHub untuk keperluan review, audit, dan pengembangan lebih lanjut. Repository mencakup: (1) smart contract Solidity di folder `contracts/`, (2) backend API Hono.js di folder `backend/`, (3) frontend React di folder `frontend/`, dan (4) dokumentasi lengkap.

**Network Configuration:**

- **Blockchain:** Ethereum Sepolia Testnet
- **Frontend:** React application on Railway platform
- **Backend:** Hono.js API server with PostgreSQL database
- **Oracle:** Custom polling-based oracle worker

Dengan sistem yang sudah live ini, peneliti dan developer dapat secara langsung mengeksplorasi fungsionalitas VeriPass, melakukan pengujian integrasi, dan mengverifikasi klaim-klaim teknis yang telah didiskusikan dalam paper ini. Pendekatan ini memberikan transparansi penuh dan memungkinkan community untuk melakukan independent verification terhadap implementasi dan keamanan sistem.

## 7 Kesimpulan

Penelitian ini berhasil mengembangkan VeriPass sebagai sistem paspor aset terdesentralisasi berbasis blockchain yang menyediakan single source of truth untuk verifikasi dan pelacakan siklus hidup aset fisik. Kontribusi utama mencakup: (1) arsitektur hybrid yang mengintegrasikan smart contract ERC-721 dengan backend tradisional, (2) mekanisme oracle untuk verifikasi data dari penyedia layanan terpercaya, dan (3) protokol hashing deterministik untuk jaminan integritas data.

Hasil pengujian menunjukkan sistem mampu mencatat transaksi dengan reliabilitas 100% dan mendeteksi seluruh upaya manipulasi data. Arsitektur yang dikembangkan dapat diadaptasi untuk berbagai domain yang memerlukan pelacakan aset dengan jaminan transparansi dan immutability.

Pengembangan selanjutnya akan fokus pada: (1) implementasi oracle terdesentralisasi untuk menghilangkan single point of failure, (2) optimisasi gas melalui teknik batching dan layer-2 solution, serta (3) integrasi dengan standar industri untuk interoperabilitas lintas platform.

## References

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

[2] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 1-32.

[3] Buterin, V., Hernandez, D., Kamphefner, T., Pham, K., Qiao, Z., Ryan, D., ... & Way, J. (2020). Combining GHOST and Casper. *arXiv preprint arXiv:2003.03052*.

[4] Tian, F. (2016). An agri-food supply chain traceability system for China based on RFID & blockchain technology. *13th International Conference on Service Systems and Service Management (ICSSSM)*, 1-6.

[5] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday*, 2(9).

[6] Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). EIP-721: ERC-721 non-fungible token standard. *Ethereum Improvement Proposals*.

[7] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F. Y. (2021). Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11), 2266-2277.

[8] Westerkamp, M., Victor, F., & Kupper, A. (2020). Tracing manufacturing processes using blockchain-based token compositions. *Digital Communications and Networks*, 6(2), 167-176.

[9] Liu, X., Barenji, A. V., Li, Z., Montreuil, B., & Huang, G. Q. (2021). Blockchain-based smart tracking and tracing platform for drug supply chain. *Computers & Industrial Engineering*, 161, 107669.

[10] Caldarelli, G. (2020). Understanding the blockchain oracle problem: A call for action. *Information*, 11(11), 509.

[11] Breidenbach, L., Cachin, C., Chan, B., Coventry, A., Ellis, S., Juels, A., ... & Zhang, F. (2021). Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*.

[12] Mammadzada, K., Iqbal, M., Milani, F., & Garcia-Banuelos, L. (2020). Blockchain oracles: A framework for blockchain-based applications. *Business Process Management: Blockchain and Robotic Process Automation Forum*, 19-34.