Jutisi: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi
https://ojs.stmik-banjarbaru.ac.id/index.php/jutisi/index
Jl. Ahmad Yani, K.M. 33,5 - Kampus STMIK Banjarbaru
Loktabat – Banjarbaru (Tlp. 0511 4782881), e-mail: puslit.stmikbjb@gmail.com
e-ISSN: 2685-0893

Perancangan dan Implementasi *Smart Contract* untuk Verifikasi Integritas Ijazah Digital pada Jaringan *Blockchain*

DOI: http://dx.doi.org/10.35889/jutisi.v15i3.3741

Ryan Bagus Bimantoro[1], Henni Endah Wahanani[2*], Muhammad Muharrom Al Haromainy[3]
Informatika, Universitas Pembangunan Nasional Veteran Jawa Timur, Surabaya, Indonesia
*e-mail *Corresponding Author:* henniendah.if@upnjatim.ac.id

Abstract

The authenticity of academic certificates such as diplomas remains a critical issue, as conventional verification still relies on centralized issuing institutions and is vulnerable to forgery. This study focuses on the design and implementation of a smart contract as the validation layer of a hash-based digital diploma verification system built on a hybrid blockchain-IPFS architecture. The smart contract, named DocumentVerification, was developed using the Solidity programming language and deployed on the Ethereum Sepolia testnet through the Hardhat framework with Alchemy as the RPC provider. The contract stores the SHA-256 hash of a document together with its IPFS Content Identifier (CID) inside an immutable mapping, enforcing input validation to reject empty hashes, empty CIDs, and duplicate registrations. Functional testing using the grey-box method through the Hardhat test runner shows that the storeDocument, getDocument, and verifyDocument functions executed correctly and that the contract successfully rejected all invalid-data scenarios. The results demonstrate that the smart contract is able to guarantee document integrity and enable independent public verification without depending on the issuing institution.

Keywords: *Smart Contract; Blockchain; Diploma Verification; Solidity; Document Integrity*

Abstrak

Keaslian dokumen akademik seperti ijazah masih menjadi persoalan penting karena proses verifikasi konvensional bergantung pada institusi penerbit yang bersifat terpusat dan rentan terhadap pemalsuan. Penelitian ini berfokus pada perancangan dan implementasi *smart contract* sebagai *validation layer* pada sistem verifikasi ijazah digital berbasis *hash* dengan arsitektur hybrid blockchain-IPFS. *Smart contract* bernama *DocumentVerification* dikembangkan menggunakan bahasa pemrograman *Solidity* dan dideploy pada jaringan *Ethereum Sepolia testnet* melalui framework *Hardhat* dengan *Alchemy* sebagai RPC provider. Kontrak menyimpan nilai *hash SHA-256* dokumen beserta *Content Identifier* (CID) IPFS ke dalam mapping yang bersifat *immutable*, serta menerapkan validasi input untuk menolak *hash* kosong, CID kosong, dan registrasi duplikat. Pengujian fungsional menggunakan metode grey-box melalui *Hardhat test runner* menunjukkan bahwa fungsi *storeDocument*, *getDocument*, dan *verifyDocument* berjalan sesuai rancangan dan kontrak berhasil menolak seluruh skenario data tidak valid. Hasil penelitian menunjukkan bahwa *smart contract* mampu menjamin integritas dokumen serta memungkinkan verifikasi publik secara mandiri tanpa bergantung pada institusi penerbit.

Kata kunci: *Smart Contract; Blockchain; Verifikasi Ijazah; Solidity; Integritas Dokumen*

1. Pendahuluan

Ijazah merupakan dokumen akademik resmi yang berfungsi sebagai bukti sah penyelesaian pendidikan formal sekaligus instrumen vital dalam menentukan mobilitas sosial serta profesional seseorang, seperti untuk melanjutkan studi ke jenjang yang lebih tinggi maupun dalam proses rekrutmen tenaga kerja. Seiring dengan masifnya arus digitalisasi di sektor pendidikan, pengelolaan ijazah kini telah bertransformasi ke dalam bentuk elektronik. Urgensi peralihan ini pun telah mendapatkan legitimasi hukum yang kuat di Indonesia melalui Undang-Undang Nomor 19 Tahun 2016 tentang Informasi dan Transaksi Elektronik [1]. Pengakuan hukum ini

menegaskan bahwa dokumen digital memiliki kedudukan yang setara dengan dokumen fisik, sehingga pengembangan sistem manajemen dan proteksi terhadap dokumen akademik digital menjadi sebuah kebutuhan yang sangat krusial demi menjaga marwah dan kredibilitas institusi pendidikan di era modern.

Meskipun payung hukum ijazah elektronik telah tersedia, infrastruktur verifikasi yang berjalan saat ini masih menghadapi kendala struktural yang signifikan. Sistem verifikasi eksis mayoritas masih bersifat terpusat *(centralized)* dan memiliki ketergantungan mutlak pada basis data internal institusi penerbit ijazah. Ketergantungan ini tidak hanya menciptakan kerentanan berupa *single point of failure* yang rawan terhadap serangan siber atau kerusakan data, tetapi juga membatasi transparansi serta menyulitkan pihak ketiga (seperti perusahaan atau instansi pemerintah) untuk melakukan verifikasi secara mandiri dan cepat [2]. Dampak dari lemahnya mekanisme verifikasi konvensional ini bermutasi menjadi persoalan hukum serius. Sebagai contoh, laporan media nasional mencatat ditemukannya 112 kasus penggunaan ijazah palsu oleh peserta seleksi CPNS di Kabupaten Simeulue, Aceh [3]. Fenomena ini menjadi bukti empiris bahwa sistem yang ada saat ini gagal mendeteksi dokumen ilegal sejak tahap awal (hulu), sehingga memicu kebutuhan mendesak akan sebuah sistem verifikasi yang transparan, andal, *tamper-proof* (kebal manipulasi), dan dapat diakses secara mandiri oleh pihak ketiga tanpa intervensi birokrasi yang rumit.

Untuk mengatasi masalah pemalsuan dokumen tersebut, sejumlah riset terdahulu telah mengusulkan model sistem verifikasi berkas digital berbasis teknologi blockchain. Suseno et al. mengembangkan sistem pencatatan dokumen akademik dengan mengombinasikan blockchain untuk menyimpan metadata serta nilai hash dokumen melalui *smart contract,* sementara *InterPlanetary File System* (IPFS) digunakan sebagai media penyimpanan terdistribusi [4]. Wijayanto merancang aplikasi verifikasi sertifikat berbasis website yang memanfaatkan transparansi blockchain untuk menjamin ketertelusuran *(traceability)* data sertifikat [5]. Di sisi lain, Swari dan Suartana mengembangkan aplikasi terdesentralisasi (dApp) untuk manajemen sertifikat digital menggunakan konsep *Non-Fungible Token* (NFT) berbasis standar ERC-721 pada jaringan Ethereum [6], yang diperkuat oleh prototipe implementasi *smart contract* untuk penyimpanan dokumen oleh Pangidoan et al [7]. Namun, gap (celah penelitian) yang masih tersisa dari riset-riset tersebut adalah pemanfaatan blockchain umumnya masih terfokus pada pencatatan data secara pasif, di mana *smart contract* hanya diposisikan sebagai repositori penyimpanan data desentralisasi tanpa adanya logika pemrosesan yang ketat di tingkat jaringan. Akibatnya, sistem belum mampu menyaring kualitas data dokumen secara otonom sebelum data tersebut masuk ke dalam buku besar *(ledger)*.

Sebagai solusi *State of the Art* untuk mengatasi keterbatasan tersebut, penelitian ini mengusulkan arsitektur *hybrid* yang mengintegrasikan jaringan IPFS secara *off-chain* untuk mengatasi tantangan skalabilitas komputasi dan biaya gas fee dari penyimpanan file besar, dengan jaringan blockchain khusus untuk pencatatan jejak kriptografis [8]. Kebaruan (novelty) utama dalam penelitian ini terletak pada transformasi peran *smart contract* dari yang sebelumnya hanya sebagai komponen penyimpan pasif, kini ditingkatkan menjadi sebuah *validation layer* yang proaktif dan otonom. Melalui pengikatan otentikasi antara nilai *hash* dan *Content Identifier* (CID) dari IPFS, *smart contract* yang dirancang menerapkan sistem kontrol berlapis *(multi-layered control)* yang secara deterministik mampu menolak *hash* kosong dan mencegah registrasi ganda sejak tahap awal transaksi sebelum pembentukan blok dilakukan. Pendekatan simulatif pada jaringan *Ethereum Sepolia testnet* digunakan untuk mengevaluasi kelayakan teknis arsitektur ini. Melalui fungsi inti dan kontrol validasi yang ketat tersebut, penelitian ini memberikan kontribusi berupa fondasi teknis yang andal untuk mewujudkan ekosistem verifikasi ijazah digital yang kebal manipulasi *(tamper-proof)*, mandiri, dan dapat diaudit oleh publik secara independen [9].

2. Metodologi

Penelitian ini menggunakan pendekatan simulatif dengan fokus pada perancangan, implementasi, dan pengujian *smart contract* sebagai komponen validasi pada sistem verifikasi ijazah digital. Pendekatan ini dipilih karena memungkinkan setiap tahapan logika kontrak, mulai dari pencatatan *hash* dokumen hingga verifikasi keberadaan data, diuji dalam lingkungan terkontrol pada jaringan blockchain pengujian *(testnet)* [10]. Tahapan penelitian meliputi analisis kebutuhan, perancangan arsitektur dan struktur data kontrak, implementasi pada bahasa *Solidity*, *deployment* ke jaringan *Ethereum Sepolia,* verifikasi kontrak, serta pengujian fungsional menggunakan metode grey-box. Alur tahapan penelitian ditunjukkan pada Gambar 1.

Gambar 1. Alur Penelitian

2.1. Analisis Kebutuhan Sistem

Sebagai langkah awal, penelitian ini melakukan analisis kebutuhan sistem secara komprehensif untuk merumuskan spesifikasi dasar dari arsitektur verifikasi ijazah digital yang diusulkan. Proses analisis ini mengidentifikasi bahwa sistem membutuhkan sebuah *validation layer* berbasis *smart contract* yang mampu mencatat identitas unik dokumen berupa hash *SHA-256* dan *Content Identifier* (CID) secara *immutable*. Di samping itu, tahap ini juga menetapkan kebutuhan infrastruktur pengembangan, yang mencakup pemilihan bahasa *Solidity,* pemanfaatan *framework Hardhat* untuk proses pengujian *grey-box*, dan penggunaan jaringan *Ethereum Sepolia testnet* agar pengujian dapat disimulasikan secara realistis tanpa membebani biaya transaksi nyata.

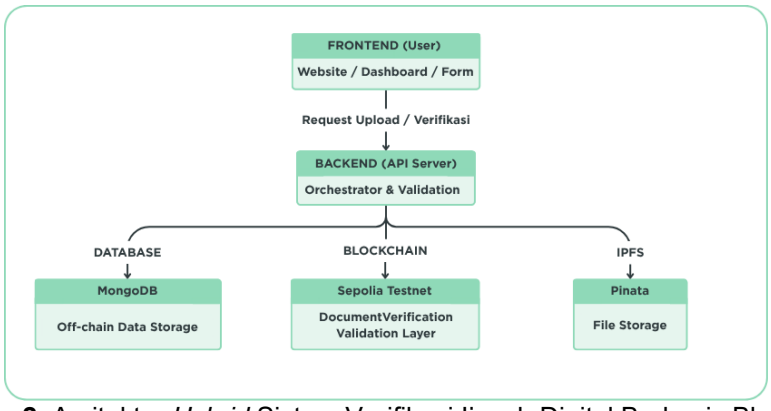
Table 1. Kebutuhan Sistem

Kategori Kebutuhan	Parameter	Spesifikasi dan Deskripsi
Kebutuhan Fungsional	Penyimpanan Data Terdesentralisasi	Sistem *(smart contract)* harus mampu menerima dan mencatat identitas unik dokumen berupa nilai *hash* kriptografi SHA-256 beserta *Content Identifier* (CID) IPFS ke dalam pemetaan *(mapping)* yang bersifat *immutable.*
	Logika Validasi *(Validation Layer)*	*Smart contract* wajib memiliki aturan validasi bawaan *(require)* untuk menolak transaksi yang mengandung hash kosong (nol), CID kosong, serta secara otomatis mencegah registrasi dokumen ganda (duplikasi).
	Fungsi Verifikasi Publik	Sistem harus menyediakan fungsi pembacaan *(view)* yang memungkinkan publik atau pihak ketiga memverifikasi keaslian dan status keberadaan dokumen secara mandiri tanpa biaya transaksi *(gas fee)*.
Kebutuhan Infrastruktur & Teknis		Pengembangan *smart contract* menggunakan bahasa pemrograman *Solidity*, serta memanfaatkan *framework Hardhat* + untuk memfasilitasi kompilasi dan simulasi pengujian *grey-box*.
	Jaringan Deployment	Implementasi kontrak pintar dilakukan pada lingkungan simulasi menggunakan jaringan *Ethereum Sepolia testnet,* dengan memanfaatkan layanan *Alchemy*

Kategori Kebutuhan	Parameter	Spesifikasi dan Deskripsi
		sebagai *Remote Procedure Call* (RPC) provider untuk menghubungkan sistem dengan jaringan blockchain.
	Penyimpanan *Off-chain*	Dokumen fisik (PDF ijazah) disimpan secara terdistribusi menggunakan jaringan *InterPlanetary File System* (IPFS) melalui layanan Pinata, sementara metadata administratif disimpan menggunakan basis data *MongoDB*.

2.2. Arsitektur Hybrid dan Posisi Smart Contract

Sistem dirancang menggunakan *arsitektur hybrid* yang memisahkan penyimpanan data *on-chain* dan *off-chain*. Pada lapisan *off-chain*, dokumen ijazah dalam format PDF disimpan pada jaringan *InterPlanetary File System* (IPFS) melalui layanan Pinata sehingga diperoleh *Content Identifier* (CID) sebagai identitas file, sedangkan metadata administratif disimpan pada basis data *MongoDB*. Seluruh proses pengelolaan data dilakukan melalui *backend* yang berperan sebagai *orchestration layer* dan penghubung antara *frontend*, basis data, layanan IPFS, serta jaringan blockchain. Pada lapisan *on-chain*, *smart contract* berfungsi sebagai *validation layer* yang mencatat nilai *hash SHA-256* dokumen beserta CID secara permanen pada blockchain *Ethereum Sepolia* [11]. Pemisahan antara penyimpanan *off-chain* dan validasi *on-chain* bertujuan untuk mengurangi beban penyimpanan pada blockchain sekaligus mempertahankan integritas, keaslian, dan sifat *immutable* dari dokumen melalui pencatatan bukti digital yang tidak dapat dimodifikasi [12]. Dengan posisi tersebut, *smart contract* menjadi sumber acuan utama (*source of truth*) dalam proses verifikasi dokumen. Gambaran posisi *smart contract* pada arsitektur sistem ditunjukkan pada Gambar 2.



Gambar 2. Arsitektur *Hybrid* Sistem Verifikasi Ijazah Digital Berbasis Blockchain

2.3. Perancangan Struktur Data dan Fungsi Kontrak

Smart contract dirancang dengan nama *DocumentVerification* dan menggunakan sebuah struktur data Document untuk merepresentasikan metadata setiap dokumen yang dicatat. Struktur ini menyimpan empat atribut, yaitu cid sebagai *Content Identifier* dokumen pada IPFS, *issuer* sebagai alamat penerbit, *timestamp* sebagai waktu pencatatan, dan *exists* sebagai penanda keberadaan dokumen. Data dokumen disimpan menggunakan struktur mapping dengan *key* berupa *hash* dokumen bertipe *bytes32*, sehingga setiap *hash* hanya dapat diregistrasikan satu kali dan berperan sebagai identitas digital yang unik. Rincian rancangan komponen kontrak ditunjukkan pada Tabel 2.

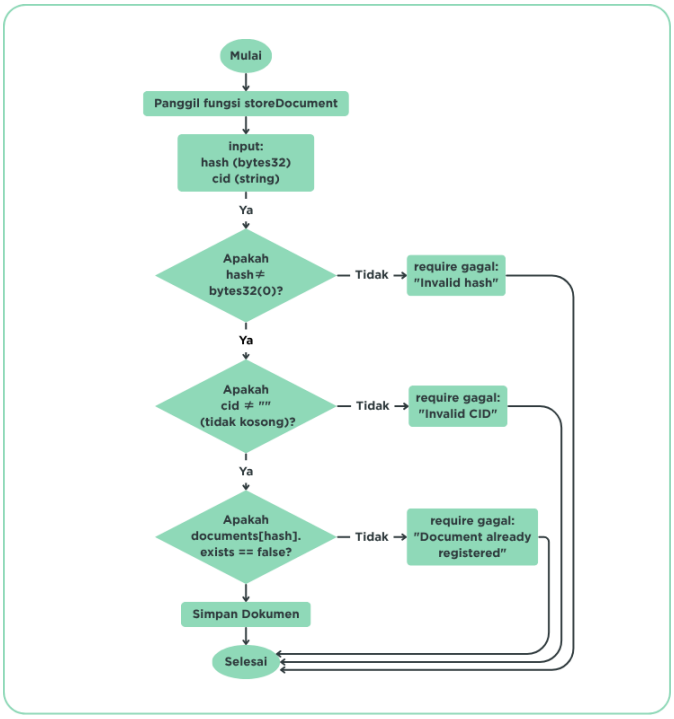
Table 2. Rancangan Komponen *Smart Contract*

Komponen	Fungsi
struct Document	Struktur data metadata dokumen: cid, *issuer, timestamp, dan exists*
mapping(bytes32 => Document)	Penyimpanan data dokumen dengan *key hash* unik sebagai identitas dokumen
event DocumentStored	Event yang dipancarkan ketika dokumen berhasil disimpan ke blockchain
storeDocument(bytes32, string)	Mencatat satu dokumen ke blockchain setelah melewati validasi *input*

Kontrak menyediakan tiga fungsi utama. Fungsi *storeDocument* digunakan untuk mencatat dokumen baru, fungsi *getDocument* digunakan untuk mengambil metadata dokumen yang telah tersimpan, dan fungsi *verifyDocument* digunakan untuk memeriksa keberadaan dokumen pada blockchain. Selain itu, kontrak memancarkan event *DocumentStored* setiap kali dokumen berhasil disimpan sebagai catatan aktivitas pada blockchain.

2.4. Mekanisme Validasi dan Pencegahan Duplikasi

Untuk menjaga integritas data, fungsi *storeDocument* menerapkan tiga aturan validasi sebelum data disimpan. Pertama, nilai *hash* tidak boleh kosong (bukan bytes32(0)). Kedua, nilai CID tidak boleh berupa *string* kosong. Ketiga, dokumen dengan *hash* yang sama tidak boleh telah terdaftar sebelumnya, yang diperiksa melalui atribut *exists* pada *mapping*. Apabila salah satu aturan tidak terpenuhi, transaksi akan dibatalkan melalui pernyataan *require* sehingga tidak ada perubahan *state* yang tercatat pada blockchain [13]. Mekanisme ini memastikan setiap dokumen bersifat unik dan mencegah penimpaan data yang telah dipublikasikan. Alur logika fungsi *storeDocument* beserta titik validasinya ditunjukkan pada Gambar 3.



Gambar 3. Diagram Alir Fungsi *storeDocument*

2.5. Lingkungan *Implementasi* dan *Deployment*

Smart contract diimplementasikan menggunakan bahasa *Solidity* versi ^0.8.28 dan dikembangkan dengan framework *Hardhat* untuk proses development, deployment, dan pengujian[14]. Kontrak dikompilasi menjadi *bytecode* yang dieksekusi pada *Ethereum Virtual*

Machine (EVM) [15]. Proses deployment memanfaatkan layanan *Alchemy* sebagai RPC provider untuk menghubungkan aplikasi dengan jaringan *Ethereum Sepolia testnet*. Setelah berhasil *dideploy, smart contract* diverifikasi pada blockchain explorer *Etherscan* sehingga *source code*, ABI, serta fungsi kontrak dapat diakses dan diaudit secara publik. Penggunaan IPFS sebagai penyimpanan *off-chain* juga sejalan dengan praktik untuk menekan beban penyimpanan dan biaya transaksi pada kontrak[16]. Penggunaan jaringan testnet dipilih agar pengujian dapat dilakukan tanpa biaya transaksi nyata, dengan konsekuensi bahwa analisis biaya gas *(gas fee)* berada di luar ruang lingkup penelitian ini.

2.6. Metode Pengujian

Pengujian *smart contract* menggunakan metode *grey-box testing,* yaitu pendekatan yang menggabungkan pengujian *black-box* dan *white-box* dengan memanfaatkan pengetahuan sebagian terhadap struktur internal kontrak namun tetap berfokus pada kesesuaian keluaran fungsi [17], [18]. Metode ini dipilih karena pengujian tidak hanya menilai keluaran fungsi, tetapi juga mempertimbangkan proses internal seperti validasi *hash*, penolakan data tidak valid, dan pencegahan duplikasi. Pengujian dijalankan pada jaringan lokal *(local development network)* menggunakan *Hardhat test runner* berbasis Node.js. Skenario pengujian mencakup keberhasilan penyimpanan dokumen, pengambilan metadata, verifikasi keberadaan dokumen, serta penanganan kondisi kesalahan berupa *hash* kosong, CID kosong, dan penyimpanan dokumen duplikat.

3. Hasil dan Pembahasan

3.1. Implementasi Struktur Smart Contract

Struktur utama *smart contract DocumentVerification* ditunjukkan pada Kode Program Gambar 4. Kontrak mendeklarasikan struktur *Document* beserta *mapping* penyimpanan dokumen dan *event DocumentStored.* Atribut *issuer* dan *hash* pada event ditandai dengan kata kunci *indexed* agar dapat dijadikan parameter pencarian *(filter)* pada log blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.28;

contract DocumentVerification {
    struct Document {
        string cid;
        address issuer;
        uint256 timestamp;
        bool exists;
    }

    mapping(bytes32 => Document) private documents;

    event DocumentStored(
        bytes32 indexed hash,
        string cid,
        address indexed issuer
    );
}
```

Gambar 4. Struktur *Smart Contract DocumentVerification*

3.2. Implementasi Fungsi Penyimpanan Dokumen

Fungsi *storeDocument* pada Kode Program Gambar 5 menerima parameter berupa *hash* dokumen dan CID IPFS. Sebelum data disimpan, fungsi menjalankan tiga validasi *require* untuk memastikan hash valid, CID tidak kosong, dan dokumen belum terdaftar. Setelah validasi terpenuhi, data dokumen disimpan ke *mapping* dengan menyertakan alamat pengirim *(msg.sender)* sebagai issuer dan waktu blok *(block.timestamp)* sebagai *timestamp,* kemudian *event DocumentStored* dipancarkan.

```
function storeDocument(bytes32 _hash, string memory _cid)
    public
{
    require(_hash != bytes32(0), "Hash tidak valid");
    require(bytes(_cid).length > 0,
       "CID tidak boleh kosong");
    require(!documents[_hash].exists,
       "Dokumen sudah terdaftar");

     documents[_hash] = Document({
        cid: _cid,
        issuer: msg.sender,
        timestamp: block.timestamp,
        exists: true
     });

     emit DocumentStored(_hash, _cid, msg.sender);
}
```

Gambar 5. Fungsi Penyimpanan Dokumen

3.3. Implementasi Fungsi Pengambilan dan Verifikasi Dokumen

Fungsi *getDocument* pada Kode Program Gambar 6 digunakan untuk mengambil metadata dokumen berdasarkan *hash.* Fungsi terlebih dahulu memastikan dokumen terdaftar melalui *validasi require,* kemudian mengembalikan nilai cid, *issuer,* dan *timestamp.* Fungsi *getDocument* dideklarasikan sebagai view sehingga tidak mengubah *state* dan tidak memerlukan biaya gas ketika dipanggil.

```
function getDocument(bytes32 _hash)
    public
    view
    returns (
       string memory cid,
       address issuer,
       uint256 timestamp
    )
{
     require(documents[_hash].exists,
        "Dokumen tidak ditemukan");

     Document memory doc = documents[_hash];
     return (doc.cid, doc.issuer, doc.timestamp);
}
```

Gambar 6. Fungsi Pengambilan Dokumen

Fungsi *verifyDocument* pada Kode Program Gambar 7 merupakan fungsi verifikasi inti yang mengembalikan nilai *boolean.* Fungsi ini mengembalikan nilai *true* apabila *hash* dokumen ditemukan pada kontrak dan *false* apabila tidak terdaftar. Karena bersifat *view,* fungsi ini dapat diakses publik tanpa biaya untuk memeriksa keaslian dokumen secara mandiri.

```
function verifyDocument(bytes32 _hash)
    public
    view
    returns (bool)
{
    return documents[_hash].exists;
}
```

Gambar 7. Fungsi Verifikasi Dokumen

3.4. Implementasi Testing Fungsional

1. Store Document

```
it("should store document", async function () {
  const contract = await deployContract();

  const hash = ethers.keccak256(
    ethers.toUtf8Bytes("ijazah-1")
  );

  const cid = "QmTestCid123";

  await contract.storeDocument(hash, cid);

  const isValid =
    await contract.verifyDocument(hash);

  assert.equal(isValid, true);
});
```

Gambar 8. Testing Fungsi Store Dokumen

Pengujian ini bertujuan untuk memvalidasi fungsionalitas utama penyimpanan dokumen pada kondisi normal. Skrip menyimulasikan pembuatan nilai *hash* menggunakan fungsi ethers.keccak256 dari sebuah string ("ijazah-1"). Setelah fungsi *storeDocument* dipanggil dengan parameter *hash* dan CID yang valid, pengujian melakukan asersi *(assertion)* dengan memanggil fungsi *verifyDocument.* Pengujian dinyatakan berhasil *(pass)* apabila fungsi verifikasi mengembalikan nilai *boolean true*, yang membuktikan bahwa dokumen telah benar-benar tercatat secara permanen pada blockchain.

2. Get Store Document

```
it("should get stored document", async function () {
  const contract = await deployContract();

  const [issuer] =
    await ethers.getSigners();

  const hash = ethers.keccak256(
    ethers.toUtf8Bytes("ijazah-2")
  );

  const cid = "QmTestCid456";

  await contract.storeDocument(hash, cid);

  const document =
    await contract.getDocument(hash);

  assert.equal(document[0], cid);
  assert.equal(document[1], issuer.address);
  assert.ok(document[2] > 0n);
});
```

Gambar 9. *Testing* Fungsi *Get Store Dokumen*

Pengujian ini dilakukan untuk memastikan bahwa metadata dokumen yang telah diregistrasikan dapat diambil kembali secara akurat melalui fungsi *getDocument.* Setelah satu dokumen contoh disimulasikan tersimpan, skrip akan mengambil data tersebut dan melakukan tiga asersi utama. Asersi memastikan bahwa variabel luaran *document[0]* sesuai dengan nilai CID awal, document sesuai dengan alamat dompet pengirim *(issuer),* dan document memiliki nilai *timestamp* blok yang lebih besar dari nol.

3. Penyimpanan dengan *hash* kosong

```
it("should reject empty hash", async function () {
  const contract = await deployContract();

  await assert.rejects(
    contract.storeDocument(
      ethers.ZeroHash,
      "QmTestCid"
    ),
    /Hash tidak valid/
  );
});
```

Gambar 10. Testing Fungsi Penyimpanan dengan *hash* kosong

Kode program ini merupakan skenario kondisi error untuk menguji ketahanan logika *validation layer* pada *smart contract.* Pengujian mengirimkan nilai *ethers.ZeroHash* yang merepresentasikan hash bernilai nol (kosong). Fungsi *assert.rejects* digunakan untuk memastikan bahwa sistem beroperasi sesuai rancangan, yakni membatalkan transaksi *(reverted)* secara otomatis dan memunculkan pesan kesalahan "*Hash* tidak valid" pada log eksekusi.

4. Penyimpanan dengan CID kosong

```
it("should reject empty cid", async function () {
  const contract = await deployContract();

  const hash = ethers.keccak256(
    ethers.toUtf8Bytes("ijazah-3")
  );

  await assert.rejects(
    contract.storeDocument(hash, ""),
    /CID tidak boleh kosong/
  );
});
```

Gambar 11. Testing Fungsi Penyimpanan dengan hash kosong

Serupa dengan pengujian sebelumnya, skenario ini berfokus pada validasi kelengkapan *Content Identifier* (CID). Sebuah hash yang valid dikirimkan, namun parameter CID sengaja diisi dengan *string* kosong (""). Skrip pengujian mengekspektasikan bahwa *smart contract* akan menolak transaksi tersebut dan mengembalikan pesan validasi "CID tidak boleh kosong", sehingga mencegah masuknya data tidak bermakna ke dalam media penyimpanan blockchain.

5. Penyimpanan dokumen duplikat

Pengujian ini sangat krusial untuk membuktikan bahwa *smart contract* mampu mencegah duplikasi registrasi (menghindari penimpaan data). Pada skenario ini, satu dokumen yang sama diregistrasikan sebanyak dua kali dengan nilai CID yang berbeda. Sistem pengujian *(assert.rejects)* memvalidasi bahwa eksekusi pemanggilan fungsi *storeDocument* yang kedua akan secara paksa digagalkan oleh jaringan dengan pesan *revert* "Dokumen sudah terdaftar".

```
it("should reject duplicate document", async function () {
  const contract = await deployContract();

  const hash = ethers.keccak256(
    ethers.toUtf8Bytes("ijazah-4")
  );

  await contract.storeDocument(
    hash,
    "QmTestCidA"
  );

  await assert.rejects(
    contract.storeDocument(
      hash,
      "QmTestCidB"
    ),
    /Dokumen sudah terdaftar/
  );
});
```

Gambar 12. Testing Fungsi Dokumen Duplikat

6. Verifikasi dokumen tidak terdaftar

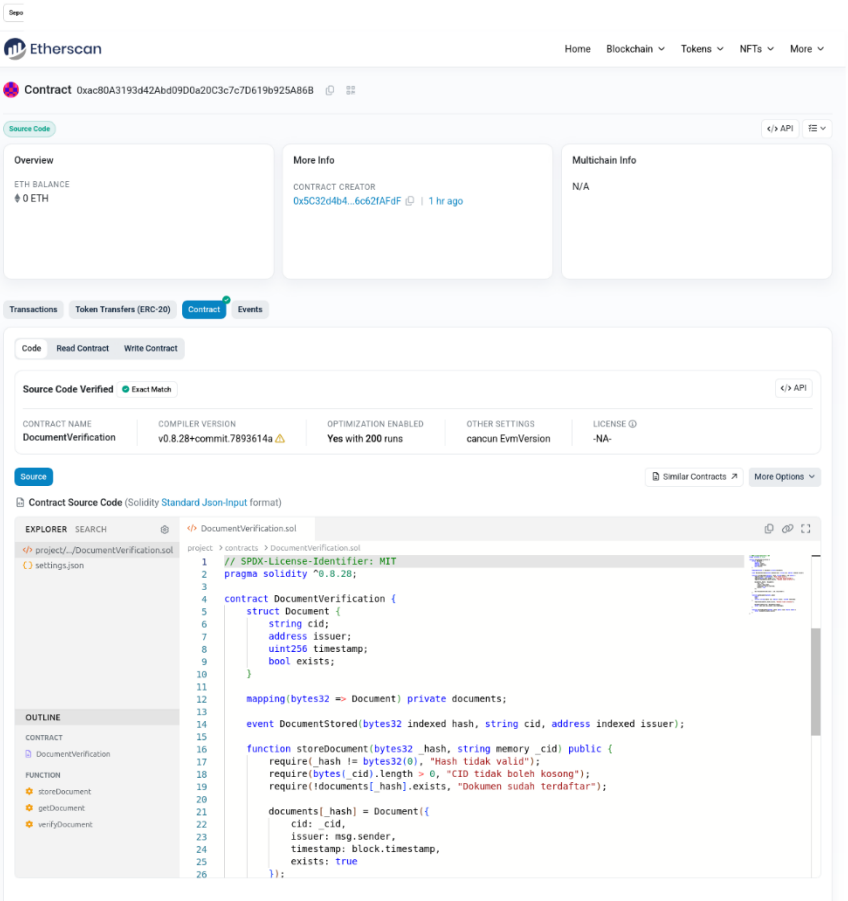
```
it("should return false for unknown document", async function () {
  const contract = await deployContract();

  const hash = ethers.keccak256(
    ethers.toUtf8Bytes("unknown")
  );

  const isValid =
    await contract.verifyDocument(hash);

  assert.equal(isValid, false);
});
```

Gambar 13. Testing Fungsi dokumen tidak terdaftar

Pengujian terakhir ini memvalidasi keandalan fungsi pembacaan publik saat mendeteksi dokumen yang tidak sah atau belum pernah diterbitkan. Skrip menyimulasikan pengecekan *(query)* menggunakan *hash* acak *("unknown")* yang tidak ada di dalam *mapping smart contract*. Pengujian ini memastikan fungsi *verifyDocument* mengembalikan nilai *boolean false* dengan akurat, membuktikan bahwa sistem tidak memberikan validasi positif palsu *(false positive)*.

3.5. Hasil *Deployment* dan Verifikasi Kontrak

Smart contract berhasil dideploy pada jaringan *Ethereum Sepolia testnet* menggunakan framework *Hardhat* dengan *Alchemy* sebagai RPC provider. Setelah proses *deployment,* kontrak diverifikasi secara publik pada blockchain *explorer Etherscan* sehingga *source code,* ABI, dan seluruh fungsi kontrak dapat diakses dan diaudit oleh siapa pun, sebagaimana ditunjukkan pada Gambar 14. Proses verifikasi ini membuktikan bahwa *bytecode* yang berjalan pada blockchain sesuai dengan *source code* implementasi. Dengan demikian, kontrak dapat berfungsi sebagai media penyimpanan hash dokumen ijazah secara terdesentralisasi yang transparan, sulit dimanipulasi, dan dapat diverifikasi publik tanpa bergantung pada satu server pusat.



Gambar 14. Verifikasi *Smart Contract* pada *Ethereum Sepolia (Etherscan*)

3.6. Hasil Pengujian Smart Contract

Pengujian fungsional dijalankan menggunakan *Hardhat test runner* pada jaringan lokal untuk memvalidasi seluruh logika kontrak. Hasil pengujian menunjukkan bahwa seluruh skenario berhasil dijalankan tanpa kesalahan, baik untuk kondisi normal maupun kondisi kesalahan. Rincian hasil pengujian disajikan pada Tabel 3, sedangkan keluaran eksekusi pengujian pada terminal ditunjukkan pada Gambar 15.

Tabel 3. Hasil Pengujian Sistem

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil
1	Penyimpanan dokumen valid *(storeDocument)*	Dokumen tersimpan dan *event DocumentStored* dipancarkan	Sesuai
2	Pengambilan metadata dokumen *(getDocument)*	Mengembalikan cid, *issuer,* dan *timestamp* dokumen	Sesuai
3	Verifikasi dokumen terdaftar *(verifyDocument)*	Mengembalikan nilai *true*	Sesuai
4	Verifikasi dokumen tidak terdaftar	Mengembalikan nilai *false*	Sesuai
5	Penyimpanan dengan *hash* kosong	Transaksi ditolak: "*Hash* tidak *valid*"	Sesuai
6	Penyimpanan dengan CID kosong	Transaksi ditolak: "CID tidak boleh kosong"	Sesuai
7	Penyimpanan dokumen duplikat	Transaksi ditolak: "Dokumen sudah terdaftar"	Sesuai

Berdasarkan Tabel 3, fungsi *storeDocument* berhasil menyimpan data hash dan CID, fungsi *getDocument* berhasil menampilkan metadata dokumen, dan fungsi *verifyDocument* berhasil

mengembalikan status keberadaan dokumen secara tepat. Pengujian terhadap kondisi kesalahan juga menunjukkan bahwa kontrak mampu menolak hash kosong, CID kosong, dan dokumen duplikat sesuai pesan validasi yang dirancang. Seluruh skenario pengujian memperoleh status *passing*, sehingga dapat disimpulkan bahwa *smart contract* telah berjalan stabil dan sesuai rancangan.

```
ryan@ryan:~/Documents/projects/skripsi/portal-ijazah-digital$ pnpm hardhat test
nodejs
No contracts to compile

Running node:test tests

  DocumentVerification
    ✔ should store document (92ms)
    ✔ should get stored document
    ✔ should reject empty hash
    ✔ should reject empty cid
    ✔ should reject duplicate document
    ✔ should return false for unknown document


  6 passing (1369ms)
```

Gambar 15. Hasil Eksekusi Pengujian *Smart Contract* Menggunakan Hardhat

3.7. Pembahasan

Hasil pengujian fungsional menunjukkan bahwa implementasi *smart contract* bernama *DocumentVerification* berhasil bertindak sebagai *validation layer* yang solid. Pengujian *grey-box* menggunakan *Hardhat test runner* memperlihatkan bahwa seluruh logika operasional seperti fungsi *storeDocument, getDocument,* dan *verifyDocument* mendapatkan status kelulusan *(passing)* secara absolut tanpa anomali. Capaian ini secara langsung menjawab tujuan awal penelitian, yaitu membangun sistem verifikasi dokumen akademik yang independen dan terdesentralisasi, sebagaimana urgensi desentralisasi keamanan data yang diusulkan oleh Abdullah. Sistem ini terbukti mampu menolak secara otomatis pendaftaran dokumen tidak valid, seperti hash atau CID kosong, serta secara ketat mencegah duplikasi data sejak awal transaksi pembentukan blok.

Temuan teknis tersebut sangat selaras dengan prinsip fundamental teknologi blockchain yang mengedepankan atribut desentralisasi dan *immutability*, sebagaimana ditinjau secara komprehensif terkait prinsip dasar blockchain oleh Tripathi et al. [18]. Penerapan arsitektur *hybrid* dalam sistem ini membuktikan efisiensi komputasi, di mana penyimpanan *file* berukuran masif dialihkan secara off-chain menuju *InterPlanetary File System* (IPFS). Sementara itu, jaringan blockchain difokuskan murni sebagai pencatat bukti kriptografis berupa *hash SHA-256.* Sensitivitas algoritma kriptografi modern ini memungkinkan sistem mengamankan dokumen dengan presisi tinggi, karena manipulasi sekecil apa pun pada level file fisik akan mendistorsi nilai *hash* dan secara otomatis menggagalkan proses otentikasi.

Jika disandingkan dengan literatur terdahulu, penelitian ini mengonfirmasi sekaligus mengekspansi batasan pada riset sebelumnya. Sistem ini mendukung arsitektur yang digagas oleh Suseno et al. [4] terkait kolaborasi blockchain dan IPFS, serta memperkuat temuan Kartiko et al. [9] bahwa penggunaan IPFS efektif menekan beban biaya *(gas fee)* pada jaringan *Ethereum.* Namun, penelitian ini berhasil menyuguhkan kebaruan *(novelty)* yang signifikan dengan merekayasa *smart contract* menjadi validation layer yang proaktif dalam menjamin keunikan dan integritas dokumen [17]. Hal ini menjadi pembeda utama dengan arsitektur pada penelitian Pangidoan et al. [7], Wijayanto [5], maupun Swari dan Suartana [6] yang memposisikan *smart contract* sekadar sebagai fasilitas penyimpanan data secara pasif.

Sempurnanya proses validasi dan penanganan kesalahan *(error handling)* pada simulasi testnet ini dilatarbelakangi oleh desain kode deterministik dari bahasa pemrograman *Solidity* pada *Ethereum Virtual Machine* (EVM). Fungsi pernyataan *require* pada fungsi *storeDocument* bertindak layaknya gerbang logika mutlak yang akan membatalkan *(revert)* transaksi jika prasyarat tidak terpenuhi, sehingga mencegah penumpukan data sampah pada state jaringan. Di samping itu, strategi pemetaan menggunakan struktur mapping (bytes32 => Document) dengan hash sebagai kunci utama secara bawaan memastikan bahwa setiap hash dokumen

ijazah terepresentasikan oleh identitas digital tunggal yang eksklusif. Hal ini membuktikan bahwa mekanisme kontrak dapat sangat andal dalam mendeteksi dan mencegah duplikasi sejak di level fundamental.

Dari kacamata akademik dan rekayasa perangkat lunak, penelitian ini mendemonstrasikan kelayakan *smart contract* sebagai penjamin integritas data administratif publik, sekaligus menawarkan landasan teknis bagi pengembangan ekosistem pendidikan terdesentralisasi. Meskipun demikian, penelitian ini tidak lepas dari berbagai keterbatasan teknis. Logika pemrograman yang ada baru dirancang untuk mengeksekusi dokumen tunggal *(single record)*, sehingga diestimasikan kurang optimal secara *gas fee* untuk penerbitan ijazah massal. Selain itu, sistem belum memiliki protokol pembatalan *(revocation)* untuk ijazah yang dicabut statusnya, dan seluruh pengujian baru dilakukan dalam ekosistem terkendali *Ethereum Sepolia testnet.* Oleh karena itu, terbuka peluang riset lanjutan untuk mengembangkan fitur penyimpanan jamak *(batch storage)* secara *on-chain,* merancang mekanisme pencabutan akses, serta mengevaluasi kinerja dan biaya transaksi secara langsung di jaringan *mainnet*.

4. Simpulan

Penelitian ini berhasil merancang dan mengimplementasikan *smart contract DocumentVerification* sebagai *validation layer* pada sistem verifikasi ijazah digital berbasis hash dengan arsitektur hybrid blockchain-IPFS. Kontrak yang dikembangkan menggunakan *Solidity* dan dideploy pada jaringan *Ethereum Sepolia testnet* mampu menyimpan nilai *hash SHA-256* dokumen beserta CID IPFS secara immutable melalui struktur mapping, serta menerapkan validasi untuk menolak hash kosong, CID kosong, dan registrasi duplikat. Hasil pengujian fungsional dengan metode *grey-box* menunjukkan bahwa seluruh fungsi (*storeDocument, getDocument, dan verifyDocument*) berjalan sesuai rancangan dan seluruh skenario kesalahan berhasil ditangani dengan benar. Dengan demikian, *smart contract* terbukti mampu menjamin integritas dokumen dan memungkinkan verifikasi publik secara mandiri tanpa bergantung pada institusi penerbit. Penelitian selanjutnya dapat diarahkan pada penambahan fungsi penyimpanan *batch on-chain,* mekanisme *revocation* dokumen, serta evaluasi biaya gas dan kinerja transaksi pada jaringan *mainnet*.

Daftar Referensi

[1] Republik Indonesia, “Undang-Undang Republik Indonesia Nomor 19 Tahun 2016 tentang Perubahan atas Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik.” Pemerintah Republik Indonesia, 2016.

[2] S. Abdullah, “Implementasi Blockchain untuk Keamanan Data Akademik dalam Sistem Informasi Perguruan Tinggi,” *Go Infotech J. Ilm. STMIK AUB*, vol. 31, no. 1, pp. 149–160, 2025, doi: 10.36309/goi.v31i1.371.

[3] M. H. S. Agus, “Ditemukan 112 ijazah palsu CPNS di Simeulue, Ombudsman dorong polisi segera usut.” 2023. [Online]. Available: https://aceh.antaranews.com/berita/341475/ditemukan-112-ijazah-palsu-cpns-di-simeulue-ombudsman-dorong-polisi-segera-usut

[4] T. R. D. Suseno, I. Afrianto, and S. Atin, “Strengthening Data Integrity in Academic Document Recording with Blockchain and InterPlanetary File System,” *Int. J. Electr. Comput. Eng.*, vol. 14, no. 2, pp. 1759–1769, 2024, doi: 10.11591/ijece.v14i2.pp1759-1769.

[5] H. Wijayanto, “Aplikasi Verifikasi Sertifikat Berbasis Website Menggunakan Blockchain,” *Infact*, vol. 8, no. 2, pp. 35–42, 2024, doi: 10.61179/jurnalinfact.v8i02.586.

[6] B. A. I. Swari and I. M. Suartana, “Pengembangan Decentralized Application (DApp) untuk Manajemen Sertifikat Digital menggunakan Non-Fungible Tokens (NFT) berbasis Standar ERC-721 di Jaringan Ethereum,” *J. Inform. Comput. Sci.*, vol. 6, no. 3, pp. 661–668, 2024, doi: 10.26740/jinacs.v6n03.p661-668.

[7] A. M. Pangidoan, P. W. Buana, and F. Purnama, “Prototype of Implementation of Smart Contract for Blockchain-Based Document Storage,” *J. Appl. Inform. Comput.*, vol. 9, no. 4, pp. 1242–1253, 2025, doi: 10.30871/jaic.v9i4.9493.

[8] R. A. Suparlan, “Implementasi Smart Contract Blockchain Ethereum Pada Aplikasi E-voting,” *Inform. Digit. Expert*, vol. 7, no. 1, pp. 66–70, 2025, doi: 10.36423/index.v7i1.2177.

[9] H. S. Kartiko, T. Rismawan, and I. Ruslianto, “Implementasi IPFS untuk Mengurangi Gas Fee Smart Contract Ethereum pada Aplikasi Penggalangan Dana,” *J. Edukasi Dan Penelit. Inform.*, vol. 9, no. 2, pp. 195–203, 2023, doi: 10.26418/jp.v9i2.61876.

[10] K. Ramadahni, “Penerapan Teknologi Blockchain dalam Sistem Manajemen Kesehatan Elektronik,” *J. Sos. Dan Teknol.*, vol. 4, no. 2, pp. 116–119, 2024, doi: 10.59188/jurnalsostech.v4i2.1097.
[11] A. N. Sari and T. Gelar, “Blockchain: Teknologi dan Implementasinya,” *J. Mnemon.*, vol. 7, no. 1, pp. 63–70, 2024, doi: 10.36040/mnemonic.v7i1.6961.
[12] H. E. Wahanani and M. H. P. Swari, “Usability Testing pada Sistem Kearsipan Dokumen Dosen,” *Krisnadana J.*, vol. 2, no. 3, pp. 424–431, 2023, doi: 10.58982/jkdn.v2i3.336.
[13] H. D. Silitonga, R. A. Windari, and S. N. Ardhya, “Analisis Keabsahan (Smart Contract) Transaksi Aset Digital di Platform Ethereum dalam Teknologi Blockchain,” *J. Komunitas Yust.*, vol. 7, no. 1, pp. 37–49, 2024, doi: 10.23887/jatayu.v7i1.94160.
[14] Hardhat, “Getting Started with Hardhat 3.” 2025. [Online]. Available: https://hardhat.org/docs/getting-started
[15] M. H. S. Fedianto, F. P. Aditiawan, and M. M. A. Haromainy, “Pengujian Sistem Jaringan Dokumentasi Dan Informasi Menggunakan Black Box Testing Dan White Box Testing,” *J. Publ. Sist. Inf. Dan Manaj. Bisnis*, vol. 3, no. 1, pp. 213–221, 2024, doi: 10.55606/jupsim.v3i1.2447.
[16] J. Skalka and M. Drlik, “Development of Automatic Source Code Evaluation Tests Using Grey-Box Methods: A Programming Education Case Study,” *IEEE Access*, vol. 11, pp. 106772–106792, 2023, doi: 10.1109/ACCESS.2023.3317694.
[17] S. Avasthi, K. Jain, A. Prakash, and T. Sanwal, “A Blockchain Architecture for Secure Decentralized System and Computing,” in *2024 3rd International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC)*, 2024, pp. 415–420. doi: 10.1109/PARC59193.2024.10486648.
[18] G. Tripathi, M. A. Ahad, and G. Casalino, “A Comprehensive Review of Blockchain Technology: Underlying Principles and Historical Background with Future Challenges,” *Decis. Anal. J.*, vol. 9, p. 100344, 2023, doi: 10.1016/j.dajour.2023.100344.
[19] Republik Indonesia, “Undang-Undang Republik Indonesia Nomor 19 Tahun 2016 tentang Perubahan atas Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik.” Pemerintah Republik Indonesia, 2016.