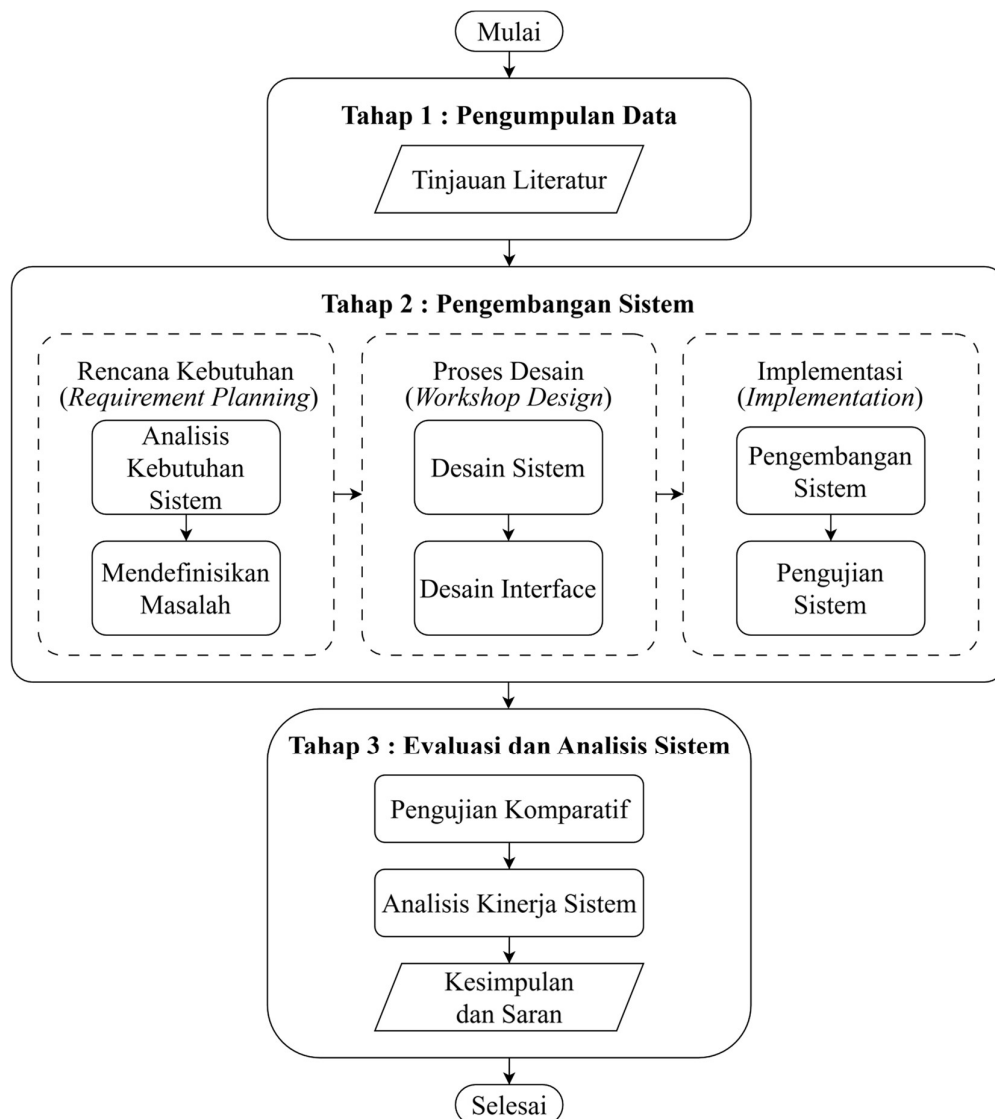


BAB III

METODOLOGI

Penelitian dan pengembangan "Optimalisasi Penyimpanan Arsip Digital Berbasis *Blockchain* dengan *Smart Contract* untuk Meningkatkan Integritas dan Transparansi Data" akan dilaksanakan dalam 3 (tiga) tahapan utama meliputi:



Gambar 3.1 Tahapan Metodologi Penelitian

3.1 Pengumpulan Data

Tahap awal penelitian akan berfokus pada pengumpulan data yang diperlukan sebagai landasan informasi terkait manajemen arsip digital berbasis *blockchain* dengan *smart contract*.

3.1.1 Tinjauan Literatur

Pengkajian dilakukan terhadap berbagai sumber literatur meliputi penelitian terdahulu seperti jurnal, skripsi, dan prosiding konferensi. Penelaahan juga dilakukan terhadap buku-buku referensi dan situs website resmi yang relevan untuk memperkuat landasan teoritis penelitian.

3.2.1 Rencana Kebutuhan (*Requirement Planning*)

3.3.1.1 Analisis Kebutuhan Sistem

Tahap ini dilakukan untuk mengidentifikasi dan menganalisis secara mendalam berbagai tantangan dalam sistem yang ada, khususnya terkait aspek integrasi, transparansi, dan kinerja sistem. Hasil analisis ini digunakan sebagai dasar pengembangan solusi yang sesuai dengan kebutuhan sistem manajemen yang dikembangkan.

3.3.1.2 Mendefinisikan Masalah

Perumusan masalah difokuskan pada identifikasi kendala-kendala utama dalam sistem saat ini. Hasil perumusan masalah dijadikan acuan dalam menentukan solusi yang tepat guna dan efektif untuk meningkatkan sistem manajemen arsip digital.

3.2 Pengembangan Sistem

Metodologi yang digunakan dalam pengembangan sistem manajemen arsip digital berbasis *blockchain* dengan *smart contract* pada penelitian ini adalah metodologi RAD yang terdiri dari 3 (tiga) tahapan berikut.

3.2.1 Proses Desain (*Workshop Design*)

3.2.2.1 Desain Sistem

Perancangan struktur dan alur kerja sistem dilakukan secara komprehensif.

Perancangan dituangkan dalam beberapa diagram sebagai berikut:

1. Arsitektur Sistem
2. Flowchart
3. Use Case Diagram
4. Class Diagram
5. Sequence Diagram

3.2.2.2 Desain Interface

Antarmuka pengguna dirancang agar intuitif dan mudah digunakan oleh operator arsip maupun pihak penandatangan dokumen. Desain memperhatikan prinsip keteraksesan, konsistensi navigasi, serta kejelasan penyajian data. Tujuannya adalah memastikan interaksi pengguna terhadap sistem dapat berjalan lancar tanpa membutuhkan pelatihan teknis mendalam.

3.2.2 Implementasi (*Implementation*)

3.2.3.1 Pengembangan Sistem

Pengembangan sistem dimulai dari penulisan *smart contract* menggunakan bahasa Solidity versi 0.8.19. *Smart contract* dirancang untuk mendukung fungsi-fungsi utama seperti registrasi pengguna, pengelolaan peran (*user*, *archivist*, *signer*), unggah dan pembaruan dokumen, serta permintaan dan validasi tanda tangan digital. Sistem juga mengimplementasikan pengendalian akses berbasis peran (RBAC) secara *on-chain*.

Antarmuka pengguna dibangun dengan ReactJS dan Tailwind CSS, serta terintegrasi dengan *wallet* Web3 melalui ConnectKit dan EthersJS. Dokumen yang diunggah disimpan di jaringan IPFS, dengan *hash*-nya direkam ke *blockchain* untuk menjamin integritas file. Selain itu, kontrak juga dioptimalkan untuk efisiensi penggunaan gas agar biaya transaksi tetap terjangkau.

3.2.3.2 Pengujian Sistem

a. Lingkungan Pengujian

Lingkungan pengujian dipersiapkan agar dapat memenuhi kebutuhan evaluasi sistem secara menyeluruh melalui pengujian black box, pengujian white box, dan pengujian komparatif.

b. Pengujian Black Box

Pada pengujian black box, dilakukan pengujian dari sisi antarmuka pengguna untuk memastikan bahwa sistem memberikan respons yang sesuai terhadap berbagai interaksi pengguna. Pengujian mencakup fitur registrasi, unggah dokumen, berbagi akses, permintaan tanda tangan, serta pelacakan histori arsip.

c. **Pengujian White Box**

White Box *testing* diterapkan untuk menguji logika internal *smart contract* termasuk validasi input, alur kondisi, dan pembatasan akses berdasarkan peran. Pengujian dilakukan menggunakan framework Hardhat dan pustaka Chai.

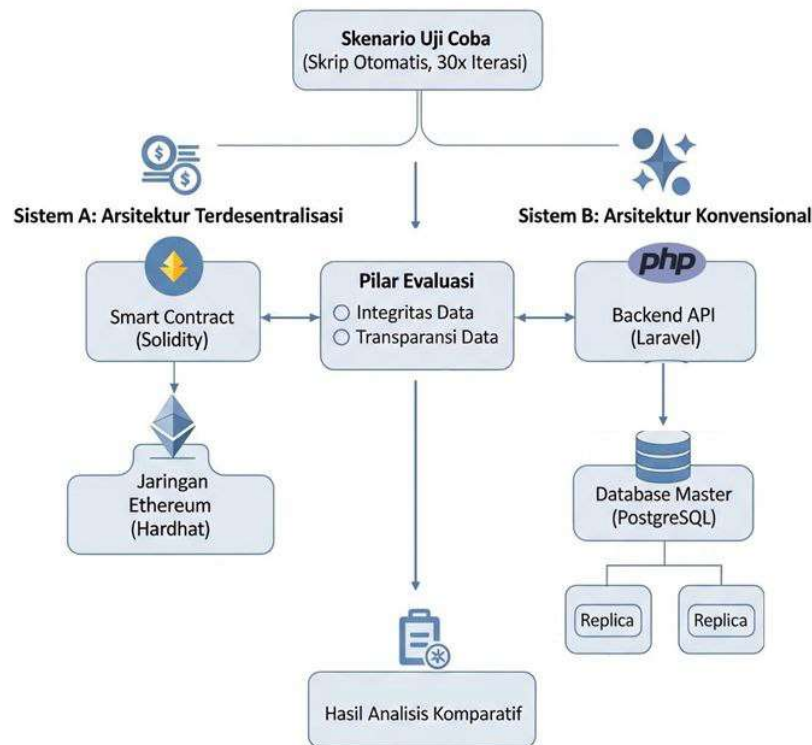
3.3 **Evaluasi dan Analisis Sistem**

Setelah melakukan pengembangan sistem berbasis blockchain, maka dilanjutkan dengan mengevaluasi dan menganalisa kinerja sistem manajemen arsip digital berbasis blockchain dengan manajemen arsip digital berbasis database dengan API melalui pengujian komparatif.

3.3.1 **Pengujian Komparatif**

Tahap ini merupakan inti dari pembuktian hipotesis penelitian, yang bertujuan untuk mengukur dan membandingkan secara objektif kapabilitas sistem berbasis *blockchain* dengan sistem berbasis arsitektur terpusat. Fokus utama pengujian ini adalah untuk memvalidasi klaim bahwa arsitektur *blockchain* menawarkan tingkat integritas dan transparansi data yang lebih optimal untuk kebutuhan manajemen arsip digital.

Pengujian komparatif dilakukan secara sistematis dengan melakukan 30 iterasi untuk setiap metrik pengujian guna memastikan validitas dan reliabilitas hasil (Montgomery, 2019). Metode pengujian yang digunakan terdiri dari beberapa tahapan utama yang dijelaskan dalam flowchart berikut:



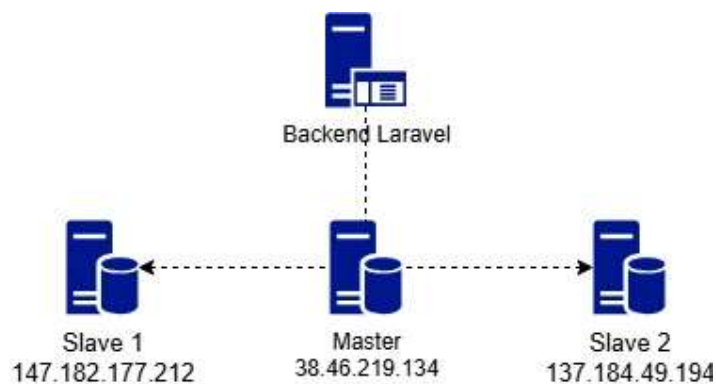
Gambar 3.2 Skenario Pengujian

3.3.1.1 Kerangka dan Lingkungan Pengujian

Untuk memastikan perbandingan yang adil dan dapat dipertanggungjawabkan secara ilmiah, penelitian ini menerapkan Kerangka Pengujian Komparatif Berbasis Skenario. Setiap skenario pengujian yang telah dirancang dieksekusi secara paralel pada kedua arsitektur sistem dalam lingkungan yang terkontrol. Jumlah replikasi atau iterasi untuk setiap skenario pengujian ditetapkan sebanyak 30 kali. Pemilihan jumlah replikasi ini didasarkan pada prinsip dalam desain eksperimen, di mana ukuran sampel $n \geq 30$ dianggap memadai untuk menerapkan Teorema Batas Sentral (*Central Limit Theorem*), sehingga

memungkinkan penggunaan uji statistik parametrik yang valid untuk menganalisis hasil (Montgomery, 2019).

Seluruh skenario pengujian diotomatisasi menggunakan serangkaian skrip yang ditulis dalam JavaScript dengan *framework* pengujian Mocha dan Chai, yang dijalankan melalui Hardhat untuk berinteraksi dengan *smart contract* dan pustaka Axios untuk berinteraksi dengan API sistem arsitektur terpusat.



Gambar 3.3 Arsitektur Sistem Pembanding

3.3.1.2 Arsitektur Sistem Pembanding

Pengujian dilakukan pada dua arsitektur sistem yang berbeda secara fundamental namun dirancang untuk menyediakan fungsionalitas manajemen arsip yang identik dari sudut pandang pengguna.

1. Sistem A: Arsitektur Terdesentralisasi (Berbasis *Blockchain*)

Sistem ini merupakan solusi yang diusulkan dalam penelitian. Komponen teknologinya meliputi:

Tabel 3.1 Komponen Arsitektur Terdesentralisasi

Komponen Sistem	Deskripsi Teknologi
Antarmuka Pengguna	Dibangun menggunakan ReactJS.

Logika Bisnis & Penyimpanan State	Diatur sepenuhnya oleh Smart contract yang ditulis dalam bahasa Solidity dan diterapkan (deploy) pada jaringan Ethereum (dalam lingkungan pengujian lokal Hardhat).
Penyimpanan File	Menggunakan protokol <i>InterPlanetary File System</i> (IPFS) untuk menyimpan file dokumen secara terdistribusi.

2. Sistem B: Arsitektur Terpusat

Sistem ini dibangun untuk merepresentasikan arsitektur aplikasi web modern yang umum digunakan. Tujuannya adalah untuk menjadi tolok ukur (benchmark) yang realistis.

Tabel 3.2 Komponen Arsitektur Terpusat

Komponen Sistem	Deskripsi Teknologi
Antarmuka Pengguna	Menggunakan antarmuka ReactJS yang sama persis dengan Sistem A untuk memastikan konsistensi pengalaman pengguna.
Logika Bisnis & Penyimpanan State	Dikelola oleh Backend API yang dibangun menggunakan framework Laravel (PHP).
Penyimpanan File	Menggunakan sistem manajemen database PostgreSQL dengan arsitektur replikasi. Arsitektur ini terdiri dari satu server database Master yang menangani semua operasi tulis (<i>write</i>), dan dua server <i>database</i> Replika yang menangani operasi baca (<i>read</i>). Mekanisme replikasi ini umum digunakan untuk meningkatkan ketersediaan (<i>high availability</i>) dan distribusi beban baca. Namun, penting untuk dipahami bahwa model ini akan

	menyinkronkan <i>semua</i> data dari <i>master</i> ke replika, termasuk data yang mungkin telah dimanipulasi secara tidak sah pada server <i>master</i> , sehingga tidak secara inheren meningkatkan integritas data.
--	---

3.3.1.3 Pilar dan Metrik Evaluasi

Sesuai dengan judul dan tujuan penelitian, evaluasi komparatif difokuskan pada dua pilar utama: Integritas Data dan Transparansi Data. Setiap pilar diuraikan menjadi beberapa metrik yang dapat diukur melalui skenario uji coba yang spesifik.

a. Pilar 1 Integritas Data

Integritas data dalam konteks ini adalah jaminan bahwa informasi arsip terlindungi dari modifikasi yang tidak sah dan aturan aksesnya ditegakkan secara konsisten. Berikut merupakan metrik dari integritas data:

1) Metrik 1.1 Imutabilitas Data dan Pencegahan Manipulasi

Tabel 3.3 Metrik 1.1 Imutabilitas Data dan Pencegahan Manipulasi

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur ketahanan inheren sistem terhadap upaya modifikasi data di luar alur bisnis yang telah ditentukan.
Skenario Uji Coba	Disimulasikan seorang pengguna dengan hak akses rendah (<i>attacker</i>) mencoba mengubah metadata sebuah dokumen (misalnya, judul) secara langsung. Skenario ini bertujuan menguji apakah sistem memiliki celah yang memungkinkan penimpaan data (<i>data override</i>).
Implementasi Teknis:	Skrip pengujian (<i>integrityTest.js</i>) menggunakan token otentikasi milik <i>attacker</i> untuk mengirim

Sistem Terpusat	permintaan PUT ke sebuah <i>endpoint</i> API imajiner (.../direct_update). <i>Endpoint</i> ini sengaja disimulasikan sebagai potensi celah keamanan yang mungkin ada akibat kesalahan konfigurasi atau pengembangan.
Implementasi Teknis: Sistem <i>Blockchain</i>	Skrip pengujian menjalankan skenario serupa dengan mencoba memanggil fungsi yang tidak ada atau memanipulasi parameter. Skenario ini dijalankan untuk membuktikan secara empiris bahwa setiap upaya untuk berinteraksi dengan <i>smart contract</i> di luar fungsi yang telah didefinisikan akan selalu ditolak oleh jaringan <i>blockchain</i> pada level protokol.
Pengukuran	Hasil pengujian dicatat sebagai keberhasilan (skor 100%) jika sistem berhasil mencegah aksi manipulasi, dan kegagalan (skor 0%) jika sebaliknya.

2) Metrik 1.2 Imutabilitas Riwayat Revisi

Tabel 3.4 Metrik 1.2 Imutabilitas Riwayat Revisi

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur jaminan bahwa riwayat perubahan sebuah dokumen bersifat permanen dan tidak dapat dihapus
Skenario Uji Coba	Sebuah dokumen diunggah, kemudian diperbarui untuk menciptakan riwayat revisi. Setelah itu, disimulasikan sebuah upaya untuk menghapus catatan riwayat revisi pertama tersebut untuk menghilangkan jejak

Implementasi Teknis: Sistem Terpusat	Skrip pengujian (<i>integrityTest.js</i>) mencoba mengirimkan permintaan DELETE ke <i>endpoint</i> API yang ditujukan untuk menghapus data revisi spesifik (<i>.../revisions/{id}</i>).
Implementasi Teknis: Sistem <i>Blockchain</i>	Skenario pengujian tetap dijalankan untuk membuktikan bahwa tidak ada fungsi <i>deleteRevision</i> atau sejenisnya dalam <i>smart contract</i> . Ini mendemonstrasikan bahwa imutabilitas bukan hanya kebijakan, melainkan batasan teknis yang melekat, di mana setiap upaya pemanggilan fungsi yang tidak ada akan gagal pada level transaksi.
Pengukuran	Hasil dicatat sebagai keberhasilan (skor 100%) jika upaya penghapusan gagal, dan kegagalan (skor 0%) jika berhasil.

3) Metrik 1.3 Efektivitas Kontrol Akses Berbasis Peran (RBAC)

Tabel 3.5 Metrik 1.3 Efektivitas Kontrol Akses Berbasis Peran (RBAC)

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur kemampuan sistem dalam menegakkan aturan akses yang telah ditetapkan berdasarkan peran pengguna.
Skenario Uji Coba	Seorang pengguna dengan peran 'user' mencoba melakukan aksi yang secara bisnis hanya diizinkan untuk peran 'archivist', yaitu mengunggah dokumen baru ke dalam sistem
Implementasi Teknis: Sistem Terpusat	Skrip pengujian (<i>integrityTest.js</i>) menggunakan token otentikasi milik 'user' untuk mengirim permintaan POST ke endpoint API <i>.../documents</i> .
Implementasi Teknis:	Skrip pengujian memanggil fungsi <i>uploadDocument</i> pada <i>smart contract</i>

Sistem <i>Blockchain</i>	menggunakan akun dompet digital (<i>wallet</i>) milik 'user'.
Pengukuran	Keberhasilan (skor 100%) jika sistem menolak transaksi tersebut, dan kegagalan (skor 0%) jika transaksi berhasil dieksekusi.

4) Metrik 1.4 Integritas Proses Pencabutan Akses

Tabel 3.6 Metrik 1.4 Integritas Proses Pencabutan Akses

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur apakah sistem dapat secara efektif dan final mencabut hak akses yang telah diberikan, serta secara konsisten menolak upaya akses setelahnya.
Skenario Uji Coba	Hak akses seorang 'user' terhadap sebuah dokumen dibagikan, kemudian langsung dicabut oleh 'archivist'. Setelah itu, 'user' tersebut mencoba kembali untuk mengakses dokumen yang sama.
Implementasi Teknis: Sistem Terpusat	Skrip pengujian (<i>integrityTest.js</i>) secara berurutan memanggil <i>endpoint</i> API untuk berbagi akses (.../share), mencabut akses (.../revoke), dan terakhir mencoba mengakses kembali dokumen tersebut (GET .../documents/{id}) menggunakan token milik 'user'.
Implementasi Teknis: Sistem <i>Blockchain</i>	Skrip pengujian memanggil fungsi <i>shareDocument</i> , kemudian <i>revokeAccess</i> , dan terakhir memanggil fungsi <i>accessDocument</i> menggunakan akun dompet digital milik 'user'.
Pengukuran	Hasil dicatat sebagai keberhasilan (skor 100%) jika sistem secara konsisten menolak upaya akses

	setelah hak dicabut, dan kegagalan (skor 0%) jika pengguna masih dapat mengakses dokumen.
--	---

b. Pilar 2 Transparansi Data

Transparansi dalam konteks ini adalah kemampuan sistem untuk menyediakan jejak aktivitas (*audit trail*) yang lengkap, dapat diandalkan, dan tidak dapat disangkal (*non-repudiable*). Berikut merupakan metrik dari transparansi data:

1) Metrik 2.1 Kelengkapan Jejak Audit Trail

Tabel 3.7 Metrik 2.1 Kelengkapan Jejak Audit Trail

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur sejauh mana sistem mampu mencatat secara lengkap setiap aktivitas yang terjadi pada sebuah dokumen, mulai dari pembuatan hingga akses terakhir.
Skenario Uji Coba	Sebuah dokumen diunggah, kemudian serangkaian aksi standar dilakukan terhadapnya: pembaruan (<i>update</i>), pembagian (<i>share</i>), dan akses oleh pengguna lain. Setelah itu, sistem akan diperiksa untuk memverifikasi apakah semua (empat) aksi tersebut tercatat dalam riwayatnya.
Implementasi Teknis: Sistem Terpusat	Skrip pengujian (<i>transparencyTest.js</i>) secara berurutan mengirimkan permintaan ke API untuk melakukan aksi upload (POST <i>/documents</i>), <i>update</i> (PUT <i>/documents/{id}</i>), <i>share</i> (POST <i>/documents/{id}/share</i>), dan <i>access</i> (GET <i>/documents/{id}</i>). Setelah itu, skrip meminta data riwayat melalui <i>endpoint</i> GET <i>/audit-logs/{id}</i> dan membandingkan jumlah log yang diterima dengan jumlah aksi yang seharusnya tercatat (yaitu 4).

Implementasi Teknis: Sistem <i>Blockchain</i>	Skrip pengujian memanggil serangkaian fungsi pada <i>smart contract</i> : <code>uploadDocument()</code> , <code>updateDocument()</code> , <code>shareDocument()</code> , dan <code>accessDocument()</code> . Setelah semua transaksi berhasil, skrip memanggil fungsi <code>getFullDocumentHistory()</code> untuk mengambil seluruh riwayat aktivitas dokumen tersebut. Kelengkapan dinilai dengan membandingkan panjang array riwayat yang dikembalikan dengan jumlah transaksi yang dijalankan (yaitu 4).
Pengukuran	Skor 100% diberikan jika semua empat aksi tercatat dengan benar, dan skor lebih rendah jika ada aksi yang tidak tercatat.

2) Metrik 2.2 Imutabilitas Proses Persetujuan (Tanda Tangan)

Tabel 3.8 Metrik 2.2 Imutabilitas Proses Persetujuan (Tanda Tangan)

Aspek Pengujian	Deskripsi
Tujuan Pengujian	Untuk mengukur finalitas dan ketidakberubahan dari sebuah keputusan atau persetujuan yang telah tercatat.
Skenario Uji Coba	Sebuah permintaan tanda tangan untuk dokumen dikirim kepada seorang 'signer', dan 'signer' tersebut menolaknya. Setelah keputusan 'tolak' ini tercatat secara final, disimulasikan upaya dari seorang pengguna dengan hak akses istimewa (disimulasikan sebagai admin sistem) untuk mengubah status tersebut menjadi 'disetujui' secara paksa.
Implementasi Teknis:	Skrip pengujian (<code>transparencyTest.js</code>) memanggil sebuah <i>endpoint</i> API khusus

Sistem Terpusat	(.../signatures/{id}/override) yang disimulasikan sebagai hak akses super-admin yang mampu menimpa status persetujuan.
Implementasi Teknis: Sistem <i>Blockchain</i>	Pengujian dilakukan dengan mencoba memanggil fungsi <code>signDocument</code> untuk kedua kalinya dengan parameter berbeda atau mencoba memanggil fungsi lain yang tidak dirancang untuk mengubah status. Tujuannya adalah untuk membuktikan bahwa setelah <i>state</i> keputusan diatur, tidak ada jalur eksekusi dalam <i>smart contract</i> yang dapat mengubahnya kembali.
Pengukuran	Hasil dicatat sebagai keberhasilan (skor 100%) jika status keputusan tetap tidak berubah, dan kegagalan (skor 0%) jika dapat diubah.

c. **Analisis Biaya Kinerja (*Performance Cost*)**

Untuk memberikan konteks yang seimbang terhadap keunggulan integritas dan transparansi yang ditawarkan *blockchain*, dilakukan pula analisis biaya kinerja. Pengukuran ini tidak bertujuan untuk menentukan sistem mana yang "lebih cepat", melainkan untuk mengkuantifikasi *trade-off* atau "biaya" komputasi yang harus dibayar untuk mendapatkan jaminan keamanan yang lebih tinggi.

1) **Metrik 3.1 Waktu Respons Operasi**

Tabel 3.9 Metrik 3.1 Waktu Respon Operasi

Aspek Pengujian	Deskripsi
Waktu Respon	Satuan waktu yang digunakan adalah milidetik dalam tiga aksi fundamental yaitu unggah (<i>upload</i>), akses (<i>access</i>), dan perbarui (<i>update</i>) dokumen.

3.3.2 Analisis Sistem

Analisis dalam penelitian ini berfokus pada evaluasi efektivitas implementasi teknologi *blockchain* dalam sistem manajemen arsip digital. Pembahasan mencakup interpretasi hasil pengujian, evaluasi sistem berdasarkan indikator fungsionalitas dan non-fungsionalitas, serta tinjauan kesesuaian dengan standar keamanan informasi dan pengelolaan arsip.