Sistem Validasi Integritas Dokumen Digital Menggunakan *Merkle Tree Hashing*

Kartika Dwi Prayoga[1], I Made Suartana[2]

[1,2] Program Studi Teknik Informatika/Teknik Informatika, Universitas Negeri Surabaya
[1]kartikadwi.22010@mhs.unesa.ac.id
[2]madesuartana@unesa.ac.id

***Abstrak*— Peralihan pengelolaan dokumen fisik menuju dokumen digital menuntut adanya mekanisme yang dapat menjamin keaslian serta keutuhan berkas secara teknis, mengingat banyak sistem yang masih hanya mengandalkan penyimpanan berkas dan pengaturan hak akses tanpa metode kriptografis yang mampu mendeteksi perubahan isi secara presisi. Penelitian ini bertujuan merancang dan membangun platform berbasis web untuk penerbitan serta validasi dokumen digital dengan menerapkan struktur *Merkle Tree* yang dipadukan fungsi *hash* SHA-256, sehingga sekumpulan dokumen dapat diringkas menjadi satu nilai *Merkle root* dan diverifikasi melalui mekanisme *Merkle proof*. Pendekatan yang digunakan bersifat eksperimental dengan tahapan *rekayasa perangkat lunak*, mencakup perancangan arsitektur sistem berbasis *framework Laravel*, pembentukan struktur pohon *hash* secara *batch*, serta pengujian performa pada variasi jumlah dokumen. Hasil uji menunjukkan bahwa algoritma SHA-256 terbukti unggul dibandingkan MD5 dalam mencegah *collision*, tetap konsisten menghasilkan *Merkle root* yang sama pada kondisi jumlah dokumen genap maupun ganjil, serta mampu mendeteksi manipulasi sekecil satu karakter dengan *avalanche effect* mendekati 50%. Mekanisme *Merkle proof* mencatatkan tingkat akurasi verifikasi sebesar 100%, dengan waktu pembentukan pohon yang meningkat seiring jumlah dokumen namun waktu verifikasi yang tetap stabil karena kedalaman pohon hanya bertumbuh secara logaritmik. Dapat disimpulkan bahwa sistem yang dikembangkan efektif menjaga integritas dokumen digital secara efisien meski volume dokumen terus bertambah.**

***Kata Kunci*— Integritas Dokumen, *Merkle Tree*, SHA-256, *Merkle proof*, Validasi Digital**

I. PENDAHULUAN

Perkembangan transformasi digital telah mendorong peralihan pengelolaan dokumen dari bentuk fisik menuju dokumen elektronik yang didukung oleh sistem informasi. Berbagai aktivitas administratif, seperti penerbitan surat, kontrak, laporan, sertifikat, dan dokumen pendukung lainnya, kini semakin mengandalkan platform berbasis web untuk meningkatkan efisiensi kerja, kecepatan layanan, serta kemudahan distribusi informasi [1]. Seiring dengan meningkatnya penggunaan dokumen digital, kebutuhan akan mekanisme yang mampu menjamin keaslian dan keutuhan dokumen juga menjadi semakin penting, mengingat dokumen tersebut sering digunakan sebagai dasar pengambilan keputusan maupun alat pembuktian yang memiliki nilai administratif dan legal.

Permasalahan yang masih banyak dijumpai pada sistem pengelolaan dokumen digital saat ini adalah minimnya mekanisme teknis yang mampu menjamin bahwa dokumen tidak mengalami perubahan sejak pertama kali diterbitkan. Sebagian besar sistem hanya mengandalkan penyimpanan berkas dan pengaturan hak akses tanpa dilengkapi metode kriptografis yang mampu mendeteksi perubahan isi dokumen secara presisi [2]. Kondisi ini menyebabkan modifikasi dokumen, baik yang disengaja maupun tidak disengaja, sulit terdeteksi setelah dokumen didistribusikan kepada pihak lain, sehingga membuka peluang terjadinya manipulasi maupun pemalsuan dokumen [3].

Salah satu pendekatan yang dapat digunakan untuk menjamin integritas dokumen digital adalah penerapan struktur data kriptografis berbasis *Merkle Tree* hashing. Mekanisme ini bekerja dengan menghitung nilai hash dari setiap dokumen, kemudian menyusunnya ke dalam struktur pohon hingga menghasilkan satu nilai hash tunggal yang disebut *Merkle root*. Melalui pendekatan tersebut, sekumpulan dokumen dapat direpresentasikan oleh satu nilai ringkas yang berfungsi sebagai bukti integritas kolektif. Selain itu, proses verifikasi dapat dilakukan secara efisien melalui mekanisme *Merkle proof* tanpa perlu mengakses seluruh dokumen yang tersimpan [4].

Berbagai penelitian menunjukkan bahwa *Merkle Tree* memiliki kemampuan yang baik dalam menjaga integritas data serta mendukung proses verifikasi yang efisien pada lingkungan dengan volume data yang besar. Ketika dipadukan dengan fungsi hash yang kuat, struktur ini juga memiliki ketahanan yang tinggi terhadap collision dan berbagai bentuk manipulasi data [5]. Meskipun demikian, penerapan *Merkle Tree* secara langsung dan terintegrasi pada sistem penerbitan serta validasi dokumen digital berbasis web masih relatif terbatas. Sebagian besar penelitian berfokus pada aspek teoritis maupun implementasi pada domain lain, sehingga masih terdapat peluang untuk mengembangkan penerapan *Merkle Tree* sebagai mekanisme validasi integritas dokumen digital yang praktis dan mudah digunakan [6].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan merancang dan mengembangkan sebuah website penerbitan dan validasi dokumen digital yang menerapkan struktur *Merkle Tree* hashing untuk menjamin integritas dokumen secara efisien. Sistem yang dikembangkan mampu mengonsolidasikan sekumpulan dokumen ke dalam satu nilai *Merkle root*, menyediakan mekanisme verifikasi melalui *Merkle proof*, serta mengevaluasi kinerja sistem berdasarkan aspek akurasi validasi, sensitivitas terhadap manipulasi dokumen, dan skalabilitas terhadap variasi jumlah dokumen. Kebaruan penelitian ini terletak pada integrasi struktur *Merkle Tree* sebagai mekanisme penyimpanan bukti integritas yang efisien dalam proses penerbitan dokumen digital berbasis web, sehingga sistem tetap mampu beroperasi secara ringan dan responsif meskipun menangani dokumen dalam jumlah besar.

Penelitian ini dibatasi pada dokumen berformat PDF yang diterbitkan melalui sistem yang dikembangkan. Fokus penelitian diarahkan pada penjaminan integritas dokumen menggunakan struktur *Merkle Tree* dan mekanisme *Merkle proof* tanpa implementasi enkripsi dokumen maupun tanda tangan digital. Pengujian dilakukan terhadap kemampuan sistem dalam mendeteksi manipulasi dokumen, memvalidasi keanggotaan dokumen dalam *batch* penerbitan, serta mengukur performa sistem berdasarkan variasi jumlah dokumen yang diproses.

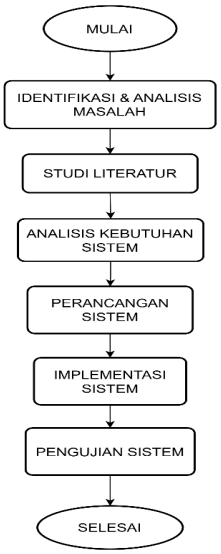
II. METODELOGI PENELITIAN

A. Pendekatan Penelitian

Penelitian ini menggunakan desain eksperimental untuk mengukur kemampuan *Merkle Tree* dalam mengidentifikasi indikasi manipulasi pada dokumen digital, sekaligus menilai efisiensi waktu komputasi (T) pada dua proses utama, yaitu pembentukan pohon hash dan verifikasi dokumen. Desain ini dipilih karena mampu menghasilkan bukti terukur secara kuantitatif mengenai kinerja algoritma pada lingkungan sistem nyata melalui parameter yang dapat diamati secara empiris [7]. Secara operasional, riset ini mengikuti kaidah rekayasa perangkat lunak yang meliputi penetapan kebutuhan, perumusan arsitektur, tahap pembangunan sistem, hingga pengujian terkendali atas keabsahan *Merkle proof*, sehingga konsisten dengan prinsip pengembangan sistem informasi yang menitikberatkan pada desain sistematis dan evaluasi kinerja yang terukur guna membuktikan bahwa struktur *Merkle Tree* efektif menjaga keutuhan dokumen melalui jalur pembuktian yang efisien.

B. Alur Penelitian

Tahapan penelitian disusun secara berurutan untuk menggambarkan proses pengembangan dan pengujian sistem penerbitan serta validasi dokumen digital berbasis *Merkle Tree*, mulai dari perumusan masalah hingga analisis hasil akhir, sebagaimana diperlihatkan pada Gambar 1. Susunan ini dimaksudkan agar setiap langkah riset dapat dipertanggungjawabkan secara ilmiah dan teknis melalui kerangka metodologis yang runtut. Skema tersebut menunjukkan rangkaian proses pengembangan yang dirancang berkelanjutan untuk menjamin keutuhan dokumen digital melalui penerapan teknik hashing, sekaligus menekan potensi kesalahan teknis dan menjaga akurasi data pengujian. Riset ini secara khusus diarahkan pada perancangan dan uji coba algoritma *Merkle Tree* untuk mengolah dokumen secara kelompok (*batch*) sehingga diperoleh nilai *Merkle root* beserta mekanisme *Merkle proof* untuk keperluan validasi, sejalan dengan pandangan bahwa keterpaduan antara tahap perancangan dan pengujian memegang peran penting dalam menjamin sistem berjalan optimal serta memenuhi standar kinerja yang ditetapkan. Tingkat keberhasilan sistem dinilai dari efisiensi waktu pemrosesan dan ketahanan integritas data terhadap upaya manipulasi.



Gambar 1. Flowchart Alur Penelitian

C. Analisis Kebutuhan Sistem

Tahap analisis kebutuhan sistem menjadi langkah awal yang menentukan dalam perancangan platform berbasis web untuk penerbitan dan validasi dokumen digital yang mengandalkan teknik *Merkle Tree* hashing. Tahap ini berfungsi mengenali serta merumuskan keseluruhan fungsi yang wajib dimiliki sistem agar mampu menopang tujuan riset, yakni menjamin keutuhan dokumen digital sekaligus menghadirkan mekanisme verifikasi yang cermat dan efisien. Analisis kebutuhan sistem dipandang sebagai bagian krusial dalam rekayasa perangkat lunak karena berperan menetapkan layanan pokok sistem beserta batasan teknis yang harus terpenuhi agar sistem dapat berfungsi sesuai kebutuhan pengguna serta tujuan operasionalnya, di mana rumusan kebutuhan yang jelas akan berdampak langsung terhadap mutu, keamanan, dan keberhasilan penerapan sistem berbasis web [8]. Kebutuhan sistem pada penelitian ini dibagi menjadi dua kategori, yaitu kebutuhan fungsional dan kebutuhan non-fungsional, yang masing-masing mencakup layanan pokok sistem serta karakteristik kualitas yang menyertainya.

D. Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan spesifikasi layanan inti yang harus tersedia pada sistem agar mampu mendukung keseluruhan proses, mulai dari penerbitan hingga validasi dokumen digital berbasis *Merkle Tree* hashing, sekaligus merepresentasikan bentuk interaksi antara pengguna dengan sistem dalam menjaga keutuhan dokumen [9]. Rancangan kebutuhan fungsional disusun untuk mencakup keseluruhan rangkaian proses, mulai dari penerbitan dokumen hingga proses verifikasi keaslian yang dapat dilakukan secara mandiri oleh pengguna. Butir-butir kebutuhan fungsional yang harus dipenuhi sistem antara lain tersedianya mekanisme autentikasi untuk mengatur hak akses admin atau petugas penerbit dokumen; kemampuan menerima unggahan dokumen digital secara *batch* dalam format PDF; kemampuan menghitung nilai hash kriptografis secara otomatis untuk setiap dokumen; kemampuan menyusun struktur *Merkle Tree* secara berjenjang

hingga dihasilkan nilai *Merkle root*; kemampuan menyimpan data integritas dokumen berupa hash, *Merkle proof*, *Merkle root*, dan metadata penerbitan; tersedianya layanan validasi dokumen melalui pengunggahan ulang dan pencocokan hash; kemampuan mengidentifikasi perubahan atau manipulasi dokumen berdasarkan hasil verifikasi; serta kemampuan menampilkan hasil verifikasi secara informatif kepada pengguna.

E. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional berkaitan dengan karakter kualitas sistem yang menentukan performa, keamanan, dan kenyamanan dalam penggunaan platform, bukan pada fungsi tertentu melainkan pada cara sistem menjalankan fungsi-fungsi tersebut secara optimal dan andal. Aspek pertama adalah keamanan sistem, yang menuntut perlindungan data dokumen, nilai hash, dan *Merkle root* dari akses tidak berwenang melalui mekanisme autentikasi yang membatasi akses pada fitur penerbitan serta menjaga keutuhan data dalam basis data. Aspek kedua, kinerja dan efisiensi waktu proses, mensyaratkan sistem mampu membangun struktur *Merkle Tree* dan melaksanakan verifikasi dalam rentang waktu relatif singkat meskipun jumlah dokumen dalam satu *batch* tergolong besar, demi menjaga responsivitas dan kenyamanan pengguna. Aspek ketiga, keandalan sistem (*reliability*), mensyaratkan sistem dapat berjalan stabil tanpa kegagalan proses pada saat pengunggahan, pembentukan struktur hash, maupun validasi dokumen, dengan data integritas yang tersimpan tetap konsisten.

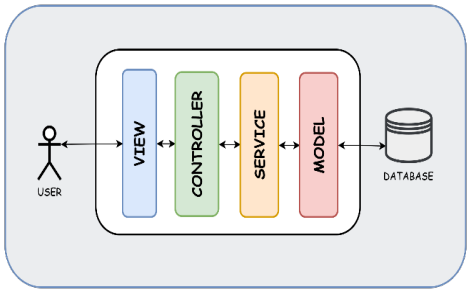
Aspek keempat, kemudahan penggunaan (*usability*), menuntut antarmuka platform dirancang sederhana dan mudah dipahami baik oleh admin maupun pengguna umum, di mana proses unggah dan validasi dapat dilakukan tanpa pemahaman teknis kriptografi yang mendalam. Aspek kelima, skalabilitas sistem, mengharuskan sistem mampu mengakomodasi pertambahan jumlah dokumen dalam proses *batch* tanpa penurunan performa yang berarti, dengan struktur *Merkle Tree* mendukung skalabilitas tersebut melalui mekanisme verifikasi berbasis jalur hash yang efisien. Aspek terakhir, kompatibilitas dan aksesibilitas, menuntut platform dapat diakses melalui beragam perangkat dan peramban modern tanpa gangguan fungsionalitas, sehingga layanan validasi dokumen dapat dimanfaatkan secara luas.

F. Data Pengujian

Dataset yang digunakan terdiri atas dokumen digital berformat PDF dengan ragam jenis dan konten untuk mencerminkan kondisi penggunaan yang lebih luas, meliputi buku digital berbahasa Indonesia dan Inggris, artikel jurnal bidang teknologi informasi, kesehatan, dan pendidikan dalam bahasa Inggris, Indonesia, dan Jepang, serta dokumen hasil pembuatan mandiri yang disiapkan khusus untuk keperluan pengujian. Pemanfaatan dokumen hasil pembuatan mandiri dimaksudkan untuk memudahkan proses manipulasi konten selama pengujian sehingga kemampuan sistem dalam mengidentifikasi perubahan data dapat diuji secara terkendali. Proses hashing dan pembentukan *Merkle Tree* pada sistem tidak bergantung pada makna semantik dokumen, melainkan pada representasi biner berkasnya, sehingga keragaman dataset tetap relevan untuk menguji performa dan keabsahan integritas dokumen pada sistem yang dikembangkan.

G. Perancangan Arsitektur Sistem

Tahap perancangan sistem menerjemahkan kebutuhan fungsional dan non-fungsional ke dalam model teknis yang akan diwujudkan, meliputi rancangan arsitektur, alur algoritma, pemodelan proses, hingga struktur penyimpanan data. Arsitektur sistem dirancang dengan pendekatan *monolithic architecture* berbasis *framework* Laravel yang bersifat terpusat guna memadukan logika antarmuka, pemrosesan kriptografi, dan pengelolaan data dalam satu lingkungan server sehingga latensi komunikasi dapat ditekan. Sistem mengadopsi pola desain *Model-View-Controller (MVC)* yang dilengkapi lapisan layanan (*service layer*) untuk menangani komputasi berat algoritma *Merkle Tree*, sebagaimana diperlihatkan pada Gambar 2.

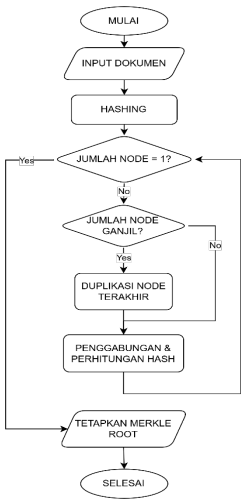


Gambar 2. Rancangan Arsitektur Sistem

Rancangan arsitektur tersebut terdiri atas empat komponen utama. Komponen pertama adalah lapisan *view* yang memanfaatkan *engine* Laravel Blade untuk menangani interaksi pengguna. Komponen kedua, lapisan *controller*, berperan sebagai *orchestrator* yang menerima permintaan, melaksanakan validasi awal, dan mendelegasikan tugas pemrosesan algoritma ke lapisan layanan. Komponen ketiga, *Merkle Engine Service*, menjadi inti sistem yang berdiri modular dan menangani logika kriptografi mulai dari konversi dokumen ke hash SHA-256 hingga kalkulasi *Merkle root* dan *Merkle proof*. Komponen keempat, lapisan model/basis data, mengelola interaksi dengan basis data melalui *Eloquent ORM* untuk menyimpan metadata dokumen dan nilai *Merkle root* agar dapat diakses kembali pada proses verifikasi.

H. Perancangan Algoritma Merkle Tree (Pembentukan)

Algoritma *Merkle Tree* dirancang untuk mengubah kumpulan dokumen dalam satu *batch* menjadi struktur pohon hash yang menghasilkan satu nilai *Merkle root* sebagai representasi integritas kolektif sekaligus identitas kriptografis seluruh dokumen. Algoritma hash SHA-256 dipilih karena memiliki karakteristik *collision-resistant* dan *avalanche effect*, sehingga perubahan kecil pada dokumen berdampak signifikan terhadap nilai hash dan *Merkle root*, sebagaimana digambarkan pada Gambar 3, di mana proses pembentukan dilaksanakan secara *bottom-up* dari penghitungan hash setiap dokumen sebagai *leaf node* hingga penggabungan berlapis pada tingkat yang lebih tinggi.

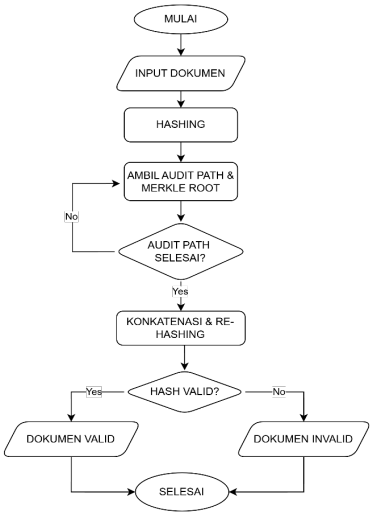


Gambar 3. Flowchart Penerbitan Dokumen

Proses pembentukan *Merkle Tree* diawali dengan penerimaan dokumen PDF dalam satu *batch*, kemudian setiap dokumen dihitung hash-nya menggunakan SHA-256 untuk menghasilkan *leaf node*. Apabila hanya terdapat satu dokumen, nilai hash tersebut langsung menjadi *Merkle root*; jika jumlah *node* lebih dari satu dan ganjil, *node* terakhir digandakan agar penggabungan dapat dilakukan berpasangan, dan proses tersebut diulang hingga tersisa satu nilai hash tunggal sebagai *Merkle root*. Algoritma ini menjamin keamanan integritas dokumen karena perubahan pada satu dokumen akan merambat hingga mengubah *Merkle root* secara keseluruhan, sekaligus efisien dari sisi penyimpanan karena sistem hanya perlu menyimpan *leaf hash* dan satu nilai *Merkle root* per *batch*.

I. Perencanaan Algoritma Merkle proof (Verifikasi)

Algoritma verifikasi *Merkle proof* dirancang untuk membuktikan integritas dokumen secara individual dengan mencocokkannya terhadap nilai *Merkle root* yang tersimpan, memungkinkan verifikasi dilaksanakan secara efisien melalui rekonstruksi jalur pembuktian (*audit path*) yang tersusun dari sejumlah *sibling hash* yang relevan, sebagaimana diperlihatkan pada Gambar 4.

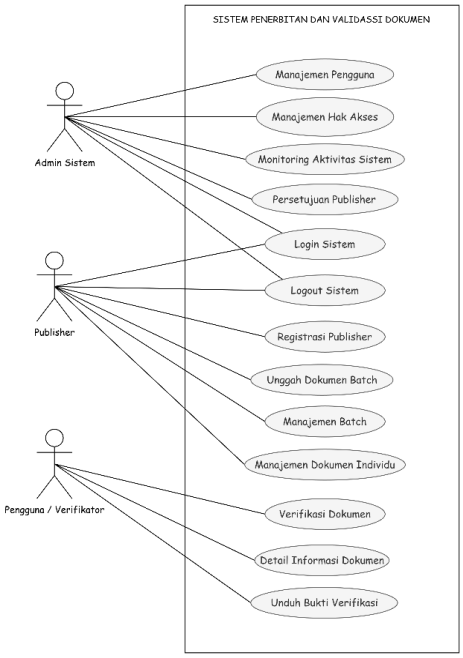


Gambar 4. Flowchart Verifikasi Dokumen

Proses verifikasi dimulai ketika pengguna mengunggah dokumen yang hendak divalidasi, kemudian sistem menghitung hash SHA-256 sebagai nilai awal, menggabungkannya dengan *sibling hash* sesuai jalur *Merkle proof*, dan menghitung ulang hash secara bertahap hingga diperoleh nilai puncak pohon yang kemudian dibandingkan dengan *Merkle root* tersimpan; kesamaan nilai menandakan dokumen valid, sementara perbedaan menandakan dokumen telah mengalami perubahan. Proses verifikasi bersifat efisien karena komputasi hanya bergantung pada tinggi pohon ($\log_2 n$), bukan pada keseluruhan jumlah dokumen.

J. Perancangan UML (Unified Modeling Language)

Pemodelan UML digunakan untuk mendokumentasikan, merinci spesifikasi, dan membangun artefak sistem, yang pada penelitian ini mencakup *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram*. *Use Case Diagram* memodelkan interaksi antara aktor eksternal dengan fungsi sistem, membagi hak akses ke dalam tiga kategori aktor, sebagaimana diperlihatkan pada Gambar 5.



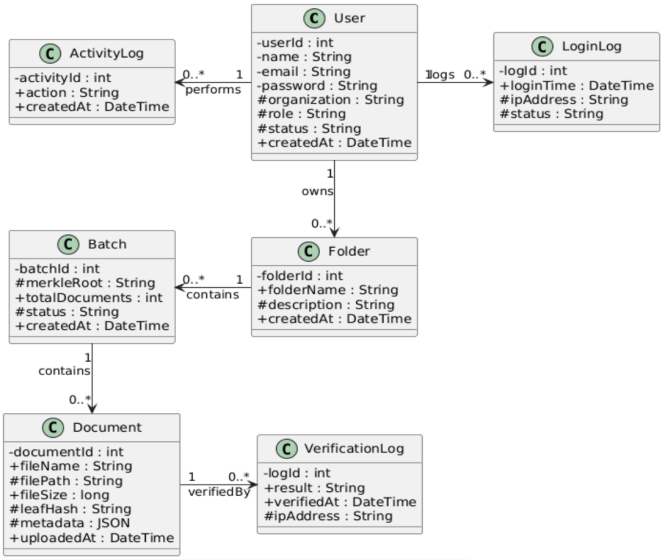
Gambar 5. Use Case Diagram

Admin Sistem berperan sebagai aktor dengan otoritas tertinggi yang mengelola pengguna, mengatur hak akses, serta mengawasi aktivitas sistem demi stabilitas dan keamanan layanan. Publisher memiliki kewenangan menerbitkan dokumen dalam bentuk *batch* serta mengelola data dokumen pada sistem, sementara Pengguna/Verifikator merepresentasikan pihak umum yang dapat melakukan verifikasi keaslian dokumen tanpa harus terdaftar sebagai pengguna tetap.

K. Perancangan Data

Perancangan data mengusung pendekatan *object-oriented* dengan *Class Diagram* sebagai model konseptual utama yang

menghubungkan tahap analisis kebutuhan dengan tahap implementasi, sebagaimana diperlihatkan pada Gambar 6.



Gambar 6. Class Diagram Validasi Dokumen

Struktur data dimodelkan melalui beberapa kelas utama yang saling berasosiasi. Kelas *User* memiliki relasi satu-ke-banyak terhadap kelas *Folder*, *Batch*, *ActivityLog*, dan *LoginLog*, menunjukkan satu pengguna dapat memiliki beberapa folder, menerbitkan beberapa *batch*, serta memiliki riwayat aktivitas dan login. Kelas *Folder* berfungsi sebagai pengelompokan administratif dokumen, sedangkan kelas *Batch* merepresentasikan unit proses yang menghasilkan nilai *merkleRoot*. Kelas *Batch* berasosiasi dengan kelas *Document* yang menyimpan atribut *leafHash*, yang selanjutnya berhubungan dengan kelas *VerificationLog* pencatat riwayat validasi, sementara kelas *ActivityLog* dan *LoginLog* memperkuat pencatatan aktivitas sistem secara keseluruhan.

L. Metode Pengujian dan Evaluasi

Bagian ini memaparkan metodologi pengujian kinerja dan fungsionalitas algoritma *Merkle Tree*, yang difokuskan pada konsistensi struktur pohon, sensitivitas terhadap perubahan data, dan keandalan mekanisme *Merkle proof*.

M. Skenario Pengujian

Skenario pertama membandingkan algoritma MD5 dan SHA-256 dengan mengunggah dua dokumen berbeda pada sistem yang menerapkan masing-masing algoritma secara terpisah, lalu membandingkan *leaf hash* dan *Merkle root* yang dihasilkan guna mengidentifikasi kemungkinan collision. Skenario kedua, konsolidasi *batch*, menguji pengunggahan dokumen berjumlah genap dan ganjil, dengan duplikasi hash terakhir mengikuti prinsip $H_{right} = H_{left}$ pada kondisi ganjil agar pembentukan *parent node* tetap berpasangan. Skenario ketiga, determinisme *Merkle root*, membandingkan nilai *Merkle root* dari dua kali unggahan *batch* identik (R_1 *dan* R_2) untuk memastikan sifat deterministik algoritma. Skenario keempat, sensitivitas dan deteksi manipulasi, mengubah satu karakter pada dokumen tersimpan lalu membandingkan hash asli dengan hash hasil manipulasi melalui *Hamming Distance*. Skenario kelima, validitas *Merkle proof*, menguji dokumen valid, dokumen yang telah dimanipulasi, dan dokumen dengan *batch* yang tidak sesuai untuk memastikan sistem menghasilkan status verifikasi yang sesuai dengan kondisi yang diharapkan. Skenario keenam, waktu eksekusi algoritma, mencatat T_{finish} dan T_{start} dalam milidetik. Skenario terakhir, skalabilitas, menguji variasi jumlah dokumen 1, 10, 25, 50, dan 100 berkas untuk mengamati performa verifikasi terhadap peningkatan volume data.

N. Instrumen Pengukuran

Instrumen pengukuran disesuaikan dengan indikator integritas data, akurasi verifikasi, efisiensi waktu, dan skalabilitas. Pengukuran *collision* menilai keunikan nilai hash MD5 dan SHA-256 antardokumen berbeda. Instrumen konsolidasi *batch* mengukur konsistensi pembentukan struktur pohon pada jumlah dokumen ganjil melalui duplikasi *node* terakhir. Instrumen determinisme *Merkle root* menilai kestabilan algoritma dengan kriteria $R^1 = R^2$ yang menandakan presisi 100%. Instrumen sensitivitas deteksi manipulasi menghitung *Hamming Distance* (D_H) dengan target *avalanche effect* (AE) mendekati 50%. Instrumen akurasi verifikasi mengukur tingkat keberhasilan sistem dalam menentukan status valid atau tidak valid berdasarkan kesesuaian hasil rekonstruksi *Merkle root* dengan *Merkle root* yang tersimpan pada sistem. Instrumen waktu eksekusi menghitung selisih T_{finish} dan T_{start}, sementara instrumen skalabilitas memetakan hubungan jumlah dokumen (n) dan waktu verifikasi berdasarkan pola pertumbuhan logaritmik $O(log^2 n)$.

O. Lingkungan Pengujian Sistem

Pengujian dilaksanakan menggunakan perangkat keras dan perangkat lunak yang mendukung implementasi dan evaluasi sistem secara lokal (*localhost*), sebagaimana dirangkum pada Tabel 1.

TABEL 1
SPESIFIKASI PERANGKAT KERAS DAN PERANGKAT LUNAK PENGUJIAN

Kategori	**Komponen**	**Spesifikasi**
Perangkat Keras	Prosesor	AMD Ryzen 7 6800H with Radeon Graphics (3.20 GHz)
	RAM	16 GB
	Kartu Grafis	AMD Radeon™ Graphics (2 GB)
	Media Penyimpanan	SSD 512 GB
	Arsitektur Sistem	64-bit Operating System, x64-based Processor
Perangkat Lunak	Sistem Operasi	Microsoft Windows 11 Pro
	Development Environment	Laragon Full 6.0
	Bahasa Pemrograman	PHP versi 8.2.28
	Framework Web	Laravel *Framework* 12.58.0
	Database Management System	MySQL

	Peramban Pengujian	Google Chrome

Lingkungan tersebut digunakan untuk menjalankan keseluruhan skenario pengujian yang telah dirancang, sehingga hasil yang diperoleh dapat dipertanggungjawabkan secara konsisten dan terukur.

III. Hasil Dan Pembahasan

Penelitian ini menghasilkan sebuah sistem berbasis web untuk memvalidasi integritas dokumen digital yang dibangun di atas struktur data *Merkle Tree* dengan fungsi hash *SHA-256*, dikembangkan menggunakan *framework Laravel*. Pemilihan *Laravel* didasarkan pada keunggulannya dalam menyediakan struktur kerja terorganisir melalui pola *Model-View-Controller (MVC)*, sebagaimana ditunjukkan oleh kajian terhadap penggunaan *framework* ini yang membuktikan kemampuannya mempercepat pengembangan aplikasi web melalui fitur *Eloquent ORM* dan *Blade templating engine* sekaligus menjaga responsivitas serta skalabilitas sistem berbasis PHP [9]. Sistem ini memiliki tiga fitur inti, yaitu pengelolaan dokumen melalui folder dan *batch*, visualisasi pembentukan struktur *Merkle Tree*, dan mekanisme verifikasi dokumen melalui *Merkle proof*. Setiap berkas yang diunggah terlebih dahulu diolah menggunakan SHA-256 untuk memperoleh nilai *leaf hash*, yang selanjutnya disusun bertingkat hingga terbentuk satu nilai *Merkle root* sebagai representasi tunggal dari keseluruhan dokumen yang diwakilinya.

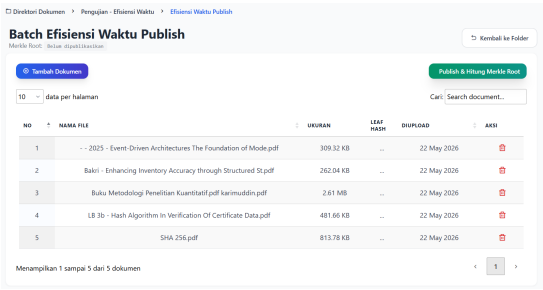
A. *Manajemen Folder dan Batch*

Pengelolaan dokumen pada sistem ini diorganisasikan melalui dua tingkatan, yakni folder sebagai wadah penyimpanan utama dan *batch* sebagai unit pengelompokan dokumen yang akan diproses menjadi satu struktur *Merkle Tree*. Pengguna dapat membentuk folder dan *batch* baru, kemudian menambahkan dokumen ke dalamnya selama status *batch* masih berada pada kondisi terbuka (*open*). Pada tahap penyimpanan, sistem membentuk nama berkas unik berbasis UUID guna mencegah duplikasi nama, disertai metadata berupa nama berkas, ukuran, lokasi penyimpanan, dan waktu unggah. Tahapan ini sejalan dengan pandangan bahwa proses penerbitan dokumen digital bukan sekadar penyimpanan berkas, melainkan juga titik perekaman nilai representasi matematis dokumen yang dijadikan rujukan validasi pada tahap berikutnya [7]. Implementasi proses penyimpanan dokumen ke dalam *batch* ditunjukkan pada Gambar 7.

```php
$hashName = Str::uuid() . '.' . $extension;
$path = $file->storeAs('documents/' . $batch->id, $hashName, 'public');

$document = Document::create([
    'batch_id' => $batch->id,
    'file_name' => $originalName,
    'hash_name' => $hashName,
    'file_path' => $path,
    'file_size' => $file->getSize(),
    'leaf_hash' => null,
    'uploaded_at' => now(),
    'metadata' => json_encode(['original_name' => $originalName]),
]);
```

Gambar 7. Implementasi Penyimpanan Dokumen ke dalam *Batch*

Hasil pengujian terhadap fitur ini menunjukkan bahwa sistem dapat menjalankan pengelolaan dokumen secara terstruktur, di mana pengguna mampu mengunggah berkas ke *batch* tertentu sebelum proses *hashing* dan pembentukan pohon dilakukan. Antarmuka sistem juga menampilkan jumlah dokumen serta status *batch* untuk memudahkan pemantauan pengelolaan dokumen digital, sebuah pendekatan yang konsisten dengan temuan bahwa sistem berbasis web mendukung pengelolaan dokumen secara lebih efektif dan terintegrasi dibandingkan pengelolaan konvensional [10]. Tampilan halaman *batch* beserta detail dokumen yang telah diunggah ditunjukkan pada Gambar 8.



Gambar 8. Tampilan *Batch* dan Detail Dokumen

B. *Visualisasi Merkle Tree*

Fitur visualisasi dikembangkan untuk memperlihatkan tahapan pembentukan struktur pohon hash, mulai dari *leaf hash*, *parent node*, hingga *Merkle root*. Untuk mengatasi kondisi jumlah node ganjil pada suatu level, sistem menerapkan mekanisme penambahan *duplicate node* agar struktur pohon tetap berbentuk biner seimbang (*balanced tree*). Visualisasi ini hanya tersedia pada *batch* yang telah dipublikasikan, karena sistem memerlukan data dokumen beserta nilai *leaf hash* yang telah tersimpan untuk membangun struktur pohon secara dinamis. Implementasi proses tersebut ditunjukkan pada Gambar 9.

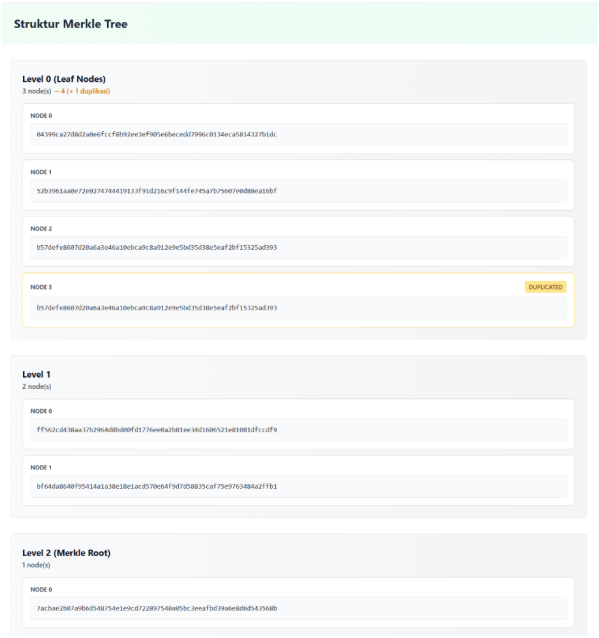
```php
$documents = $batch->documents()->orderBy('id', 'asc')
->get(['id', 'file_name', 'leaf_hash']);

$levels = $this->buildLevels($leafHashesBinary);

return [
    'merkle_root' => $merkleRootHex,
    'levels' => $levelDetails,
    'duplicated_nodes' => $duplicatedNodes,
];
```

Gambar 9. Implementasi Kode Visualisasi *Merkle Tree*

Pengujian terhadap fitur ini membuktikan bahwa sistem berhasil menampilkan struktur pohon secara bertingkat dari *leaf hash* hingga *Merkle root*, di mana setiap dokumen yang telah dihitung nilai SHA-256-nya digabungkan secara berpasangan sampai diperoleh satu nilai akar sebagai representasi integritas kumpulan dokumen.

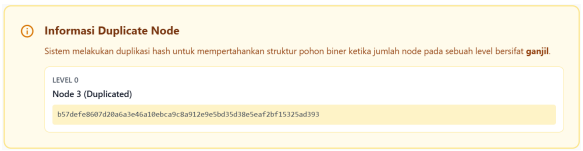
Gambar 10, memperlihatkan hasil visualisasi struktur *Merkle Tree* yang terbentuk dari kumpulan dokumen dalam satu *batch*. Pola hierarkis semacam ini sesuai dengan prinsip bahwa setiap simpul internal merupakan hasil hash dari

kombinasi anak-anaknya yang berpasangan, sehingga perubahan pada satu data akan merambat naik hingga ke puncak struktur [9].



Gambar 10. Hasil Visualisasi Pembentukan *Merkle Tree*

Pada kondisi jumlah dokumen ganjil, sistem secara otomatis menggandakan node terakhir pada level tersebut sebelum proses penggabungan menuju *parent node* dilanjutkan. Node hasil duplikasi diberi label "*Duplicated*" untuk membedakannya dari node asli, sehingga pengguna dapat membedakan struktur sebenarnya dari hasil penyesuaian system. Seperti yang ditunjukan pada Gambar 11.



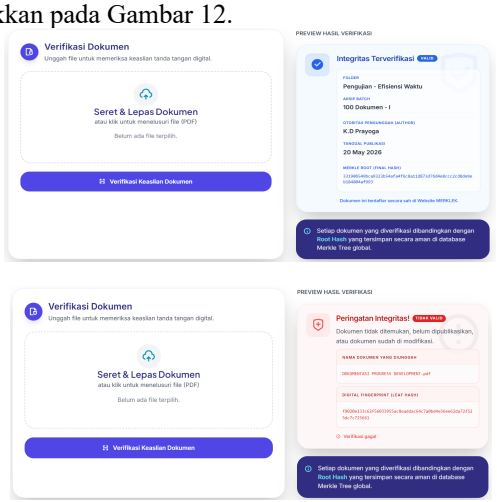
Gambar 11. Visualisasi Node Duplikat

C. *Verifikasi Dokumen*

Mekanisme verifikasi pada sistem ini dirancang untuk membuktikan keaslian dokumen melalui pendekatan *Merkle proof*. Proses dimulai ketika pengguna mengunggah berkas yang ingin diverifikasi, kemudian sistem menghitung nilai SHA-256 dari berkas tersebut dan mencocokkannya dengan kumpulan *leaf hash* yang tersimpan pada *batch* terkait. Pendekatan validasi berbasis hash semacam ini sejalan dengan prinsip bahwa nilai hash yang dihitung ulang harus identik dengan nilai yang tersimpan pada saat penerbitan agar dokumen dinyatakan utuh [11].

Apabila nilai hash dokumen tidak ditemukan dalam *batch*, atau *batch* yang dirujuk belum berstatus dipublikasikan, sistem akan langsung menetapkan status *INVALID* tanpa melanjutkan proses rekonstruksi. Sebaliknya, apabila pencocokan berhasil, sistem akan menggabungkan hash berdasarkan posisi *sibling node* (kiri/kanan) hingga diperoleh kembali nilai *Merkle root* yang kemudian dibandingkan dengan root yang tersimpan pada system [10].

Pengujian fungsional terhadap mekanisme ini menunjukkan dua kondisi keluaran yang berbeda. Pada dokumen yang sesuai dengan *batch* rujukannya, hasil rekonstruksi hash identik dengan *Merkle root* yang tersimpan sehingga sistem mengeluarkan status *VALID*. Sebaliknya, dokumen yang telah dimodifikasi atau tidak berkaitan dengan *batch* tersebut menghasilkan status *INVALID*. Sistem juga dilengkapi fitur pencatatan log verifikasi (*verification log*) untuk merekam riwayat setiap proses pengecekan yang dilakukan pengguna. Hasil proses verifikasi dokumen valid dan tidak valid ditunjukkan pada Gambar 12.



Gambar 12. Hasil Verifikasi Dokumen Valid dan Tidak Valid

Penerapan algoritma *Merkle Tree* bertumpu pada dua tahapan utama, yaitu pembentukan nilai hash tiap dokumen menggunakan SHA-256 dan penggabungan nilai-nilai tersebut secara berpasangan hingga terbentuk satu *Merkle root*. Pada tahap penghitungan hash, sistem membaca berkas dokumen yang tersimpan kemudian menghasilkan nilai SHA-256 menggunakan fungsi *hash file()* sebagai *leaf hash*. Nilai hash ini bersifat deterministik sebagai konsekuensi dari karakteristik fungsi hash kriptografis yang mengubah data berukuran arbitrer menjadi nilai ringkas berukuran tetap secara satu arah, sehingga input yang sama akan selalu menghasilkan output identik namun tidak dapat dikembalikan ke bentuk semula [5].

D. *Pembentukan Merkle Tree dan Merkle proof*

Proses pembentukan struktur pohon dilakukan ketika *batch* dipublikasikan, di mana sistem menghitung *Merkle root* melalui fungsi *computeMerkleRoot()* dan menyusun *Merkle proof* untuk setiap dokumen melalui fungsi *generateProofs()*. Pada level dengan jumlah node ganjil, sistem menduplikasi node terakhir terlebih dahulu sebelum melakukan penggabungan hash secara berpasangan untuk membentuk *parent node* pada level berikutnya.

Adapun pembentukan *Merkle proof* dilakukan setelah struktur pohon terbentuk sepenuhnya, di mana sistem menyusun jalur audit untuk setiap dokumen berdasarkan posisi nodenya pada tiap level melalui fungsi *getProofPath()*. Penentuan pasangan *sibling node* pada level yang sama

dilakukan menggunakan operasi XOR (^) terhadap indeks node, sehingga sistem dapat menyimpan nilai sibling hash beserta arah node sebagai bahan rekonstruksi pada proses verifikasi berikutnya.

Tahap akhir verifikasi memanfaatkan fungsi *verifyUploadedFile()* untuk mencocokkan kembali hash dokumen yang diunggah dengan *leaf hash* tersimpan, kemudian merekonstruksi *Merkle root* menggunakan sibling hash dan arah node yang tercatat pada *Merkle proof*. Perbandingan hasil rekonstruksi dengan root yang tersimpan dilakukan melalui fungsi *hash_equals()* untuk menentukan validitas dokumen, sebuah pendekatan kriptografis yang menurut kajian tentang verifikasi sertifikat digital mampu mendeteksi manipulasi dokumen secara andal dalam berbagai alur validasi [12].

E. Pengujian Sistem

Pengujian dilaksanakan pada lingkungan lokal (*localhost*) untuk menilai kinerja algoritma *Merkle Tree* dalam mendukung validasi integritas dokumen digital. Cakupan pengujian meliputi tujuh aspek, yaitu uji *collision* MD5 dan SHA-256, uji konsolidasi *batch*, uji determinisme *Merkle root*, uji sensitivitas terhadap manipulasi dokumen, uji validitas *Merkle proof*, uji waktu eksekusi algoritma, serta uji skalabilitas sistem.

Pengujian Collision MD5 dan SHA-256. Pengujian ini membandingkan ketahanan dua algoritma hash terhadap kemungkinan *collision*, yaitu kondisi ketika dua dokumen berbeda menghasilkan nilai hash identik. Data uji bersumber dari repositori publik berjudul *Collision* yang dipublikasikan oleh GeopJr melalui platform GitHub. Hasil pengujian disajikan pada Tabel 2.

TABEL 2
HASIL PENGUJIAN COLLISION MD5 DAN SHA-256

Parameter	MD5	SHA-256
Dokumen 1	doc_collision_a.pdf	doc_collision_a.pdf
Dokumen 2	doc_collision_b.pdf	doc_collision_b.pdf
Leaf hash Dokumen 1	bcc4be725bbc03c6c3dfc42f59b3df96	5c694a1291411928d7b3dd679c0754e98ecd12f5feed77b8ca2ddc51f56cb0c3
Leaf hash Dokumen 2	bcc4be725bbc03c6c3dfc42f59b3df96	9e7dc6c1a1fe594937e4e8e77a70dfc1c88a4e02b3e0b26d5e82023111f4096b
Status Pengujian	Collision	Aman

Hasil pengujian pada Tabel 2 memperlihatkan bahwa MD5 menghasilkan nilai hash identik pada dua dokumen yang berbeda, kondisi yang berkaitan dengan panjang keluaran MD5 yang relatif singkat (128-bit) sehingga rentan terhadap serangan collision attack meski memiliki performa pemrosesan yang cepat [7]. Sebaliknya, SHA-256 dengan panjang keluaran 256-bit tetap menghasilkan nilai hash yang berbeda pada kedua dokumen uji, sehingga dikategorikan aman dari ancaman collision pada skenario pengujian ini.

Pengujian Konsolidasi *Batch*. Pengujian ini menilai kemampuan sistem dalam mengintegrasikan sekumpulan dokumen digital terpisah ke dalam satu struktur hierarkis guna menghasilkan satu nilai komitmen tunggal. Hasilnya menunjukkan bahwa pada jumlah dokumen genap, seluruh *leaf node* dapat dipasangkan langsung tanpa penyesuaian tambahan, sementara pada kondisi ganjil sistem berhasil menjalankan mekanisme duplikasi node terakhir secara otomatis yang ditandai label DUPLICATED, sehingga proses pembentukan *parent node* tetap dapat berlanjut hingga diperoleh satu *Merkle root* yang sah.

Pengujian Determinisme *Merkle root*. Pengujian ini membuktikan konsistensi nilai *Merkle root* terhadap dokumen yang identik. Hasil pengujian disajikan pada Tabel 3.

TABEL 3
HASIL PENGUJIAN DETERMINISME

Kondisi	Jumlah Doc	Uji ke-	Nilai *Merkle root*	Status
Ganjil	3	1	73f33b819e02a2f1ce5c7fda4b6dd6a2f7a632a4c6a0c04750225a0b1cc1a254	Identik
Ganjil	3	2	73f33b819e02a2f1ce5c7fda4b6dd6a2f7a632a4c6a0c04750225a0b1cc1a254	Identik
Genap	4	1	6dcb7e0551ebef4ebd1de5a0cc4a6ff3ec1515900d2593faaef2f701265e9ef1	Identik
Genap	4	2	6dcb7e0551ebef4ebd1de5a0cc4a6ff3ec1515900d2593faaef2f701265e9ef1	Identik

Sebagaimana tersaji pada Tabel 3, pengunggahan *batch* dengan tiga dokumen (kondisi ganjil) maupun empat dokumen (kondisi genap) sebanyak dua kali menghasilkan nilai *Merkle root* yang identik pada masing-masing pengujian. Temuan ini menegaskan bahwa mekanisme duplikasi node pada kondisi ganjil tidak memengaruhi konsistensi nilai akhir yang dihasilkan.

Pengujian Sensitivitas dan Deteksi Manipulasi. Pengujian ini mengukur karakteristik *avalanche effect* pada SHA-256 melalui perhitungan Hamming Distance antara hash dokumen asli dan hash dokumen yang telah dimodifikasi sebagian kecil isinya. Hasil pengujian disajikan pada Tabel 4 .

TABEL 4
HASIL PENGUJIAN SENSITIVITAS DAN DETEKSI MANIPULASI

Parameter	Uji-01	Uji-02	Uji-03
Jenis Perubahan	Perubahan 1 karakter	Perubahan 1 kata	Perubahan 1 huruf
Hash Asli	393bd667cf9767727f3d8cac2bd3aa9475693462b07c24115b7a79aa1e6fee39	393bd667cf9767727f3d8cac2bd3aa9475693462b07c24115b7a79aa1e6fee39	393bd667cf9767727f3d8cac2bd3aa9475693462b07c24115b7a79aa1e6fee39
Hash Manipulasi	06b8864d0fc31c093802d35298d2fa7e83a97ceef6d5d990546d45a9fec0f4c0	ae107a7ebac20f448a40a83dc421ac552d928dc78f2013d315dbc91dc01ed740	72fc9968e894aeb7e777df84368827e385b3ea83bb4e9f1f8a0b1398c63c8131

Hamming Distance	128 bit	138 bit	127 bit
Avalanche effect	50.00%	53.91%	49.61%
Status	**Terdeteksi**	**Terdeteksi**	**Terdeteksi**

Berdasarkan Tabel 4, tiga skenario perubahan satu karakter, satu kata, dan satu huruf masing-masing menghasilkan nilai *avalanche effect* sebesar 50,00%, 53,91%, dan 49,61%. Hasil ini sejalan dengan ketentuan bahwa keandalan fungsi hash dalam mendeteksi manipulasi dinilai ideal apabila memiliki tingkat perubahan bit yang mendekati 50 persen [8], sehingga membuktikan sensitivitas tinggi SHA-256 terhadap perubahan data sekecil apa pun [13].

Pengujian Validitas *Merkle proof*. Pengujian ini mengevaluasi kemampuan sistem menerima jalur bukti yang sah dan menolak dokumen yang tidak valid melalui proses rekonstruksi *Merkle root*. Hasil pengujian disajikan pada Tabel 5.

TABEL 5
HASIL PENGUJIAN VALIDITAS *MERKLE PROOF*

Parameter	**Uji-01**	**Uji-02**	**Uji-03**	**Uji-04**
Jenis Pengujian	Dokumen Valid	Manipulasi 1 Huruf	Manipulasi 1 Kata	*Batch* Tidak Sesuai
Hasil Rekonstruksi Root	Sesuai	Tidak Sesuai	Tidak Sesuai	Tidak Sesuai
Status Verifikasi	VALID	INVALID	INVALID	INVALID
Status Pengujian	**Berhasil**	**Berhasil**	**Berhasil**	**Berhasil**

Sebagaimana terlihat pada Tabel 5, seluruh skenario menghasilkan status verifikasi yang sesuai dengan hasil yang diharapkan, sehingga tingkat keberhasilan verifikasi mencapai 100%. Hasil ini menunjukkan bahwa sistem hanya menyatakan dokumen valid apabila hasil rekonstruksi *Merkle root* identik dengan *Merkle root* yang tersimpan pada sistem.

Pengujian Waktu Eksekusi Algoritma. Pengukuran dilakukan terhadap waktu pembentukan pohon (build time) dan waktu verifikasi (verify time) pada variasi jumlah dokumen 1, 10, 25, 50, dan 100 berkas, masing-masing diulang lima kali. Hasil pengujian disajikan pada Tabel 6.

TABEL 6
HASIL PENGUJIAN WAKTU EKSEKUSI ALGORITMA

ID *Batch*	**Jumlah Dokumen**	**Jumlah Pengujian**	**Build Time**	**Verify Time**
BATCH-01	1	5	37.84 ms	21.75 ms
BATCH-02	10	5	268.28 ms	20.40 ms
BATCH-03	25	5	428.96 ms	27.80 ms
BATCH-04	50	5	654.67 ms	24.50 ms
BATCH-05	100	5	1413.36 ms	21.31 ms

Tabel 6 menunjukkan bahwa build time meningkat secara proporsional seiring bertambahnya jumlah dokumen, dari 37,84 ms pada satu dokumen hingga 1.413,36 ms pada seratus dokumen, sejalan dengan bertambahnya jumlah operasi hashing yang harus dijalankan. Sebaliknya, verify time relatif stabil pada kisaran 20–28 ms terlepas dari jumlah dokumen.

Pengujian Skalabilitas Sistem. Pengujian ini menelusuri hubungan antara jumlah dokumen, kedalaman pohon (tree depth), dan waktu verifikasi. Hasil pengujian disajikan pada Tabel 7.

TABEL 7
HASIL PENGUJIAN SKALABILITAS SISTEM

ID *Batch*	**Jumlah Dokumen**	**Kedalaman Pohon**	**Verify Time**
BATCH-01	1 Dokumen	0	21.75 ms
BATCH-02	10 Dokumen	4	20.40 ms
BATCH-03	25 Dokumen	5	27.80 ms
BATCH-04	50 Dokumen	6	24.50 ms
BATCH-05	100 Dokumen	7	21.31 ms

Tabel 7 menunjukkan bahwa kedalaman pohon bertambah dari 0 pada satu dokumen menjadi 7 pada seratus dokumen, namun peningkatan tersebut tidak diikuti kenaikan signifikan pada waktu verifikasi. Pola ini membuktikan keunggulan utama penggunaan pohon hash dalam sistem distribusi data, yaitu skalabilitas yang bersifat logaritmik, di mana penambahan jumlah dokumen secara eksponensial hanya memberikan dampak minimal terhadap panjang jalur bukti yang harus diproses verifikator.

F. Pembahasan Umum

Secara keseluruhan, rangkaian pengujian menunjukkan bahwa algoritma SHA-256 lebih unggul dibandingkan MD5 dalam mencegah collision, sekaligus mampu mempertahankan sifat deterministik pada proses pembentukan *Merkle root* baik pada kondisi jumlah dokumen genap maupun ganjil. Sensitivitas tinggi terhadap perubahan data memungkinkan sistem mendeteksi manipulasi dokumen secara akurat, sementara mekanisme *Merkle proof* memberikan akurasi verifikasi sebesar 100% pada skenario pengujian yang dilakukan. Pada aspek performa, waktu pembentukan pohon meningkat seiring jumlah dokumen, tetapi waktu verifikasi tetap stabil dan kedalaman pohon bertumbuh logaritmik, sehingga sistem ini dinilai mampu menjaga efisiensi proses verifikasi meskipun jumlah dokumen dalam *batch* terus bertambah.

G. Kelebihan dan Keterbatasan Sistem

Sistem yang dikembangkan menunjukkan keunggulan dalam mendeteksi perubahan dokumen secara akurat berkat sifat sensitif SHA-256, di samping karakter deterministiknya yang menjamin konsistensi hash pada dokumen identik. Pendekatan ini sejalan dengan kajian mengenai pelestarian arsip digital yang menegaskan bahwa integritas data menjadi

pilar utama untuk memastikan dokumen digital tetap otentik dan dapat diuji kebenarannya sepanjang waktu penyimpanan [10]. Hasil pengujian skalabilitas turut memperkuat bahwa waktu verifikasi tetap stabil meskipun jumlah dokumen bertambah, dan fitur visualisasi yang disediakan membantu pengguna memahami proses pembentukan hash secara lebih intuitif.

Di sisi lain, sistem ini masih memiliki sejumlah keterbatasan. Dukungan format dokumen saat ini terbatas pada berkas PDF, sehingga belum dapat menangani jenis dokumen digital lainnya maupun dokumen fisik hasil pemindaian. Waktu pembentukan pohon juga cenderung meningkat seiring bertambahnya jumlah dokumen dalam satu *batch* karena bertambahnya beban komputasi hashing. Selain itu, fokus penelitian ini terbatas pada mekanisme validasi integritas melalui struktur *Merkle Tree* dan SHA-256, tanpa mencakup penerapan enkripsi dokumen atau tanda tangan digital, dan pengujian keamanan yang dilakukan hanya menyasar skenario manipulasi isi dokumen tanpa menjangkau aspek keamanan infrastruktur server maupun basis data sistem.

IV. KESIMPULAN

Riset ini berhasil mengembangkan platform berbasis web untuk validasi integritas dokumen digital dengan memanfaatkan struktur *Merkle Tree* dan fungsi hash SHA-256. Hasil implementasi menunjukkan bahwa *Merkle Tree* mampu mengonsolidasikan sekumpulan dokumen dalam satu *batch* menjadi satu nilai *Merkle root* yang merepresentasikan integritas kolektif dokumen. Pengujian juga membuktikan bahwa SHA-256 menghasilkan nilai hash yang unik dan sensitif terhadap perubahan data, sehingga manipulasi sekecil apa pun pada dokumen dapat terdeteksi melalui perubahan hash dan *Merkle root* yang dihasilkan.

Mekanisme *Merkle proof* berhasil diterapkan untuk melakukan verifikasi dokumen secara efisien tanpa memerlukan perhitungan ulang seluruh hash dalam satu *batch*. Selain itu, sistem menunjukkan konsistensi dalam menghasilkan nilai *Merkle root* pada dokumen yang identik serta mampu mempertahankan akurasi verifikasi pada seluruh skenario pengujian. Dari sisi performa, waktu pembentukan pohon hash meningkat seiring bertambahnya jumlah dokumen, namun proses verifikasi tetap stabil karena kedalaman pohon bertumbuh secara logaritmik. Temuan ini menunjukkan bahwa pendekatan *Merkle Tree* dan SHA-256 efektif diterapkan sebagai mekanisme penjamin integritas dokumen digital yang akurat, efisien, dan skalabel.

REFERENSI

[1] M. R. Anwar, D. Apriani, and I. R. Adianita, “Hash Algorithm In Verification Of Certificate Data Integrity And Security,” vol. 3, no. 2, pp. 181–188, 2021.

[2] H. K. Albahadily, I. A. Jabbar, and A. A. Altaay, “Issuing Digital Signatures for Integrity and Authentication of Digital Documents,” vol. 34, no. 3, pp. 3–8, 2023.

[3] A. Djajadi, K. S. Lestari, L. E. Englista, and A. Destaryana, “Blockchain-Based E-Certificate Verification and Validation Automation Architecture to Avoid Counterfeiting of Digital Assets in Order to Accelerate Digital Transformation,” vol. 16, no. 1, pp. 68–85, 2023.

[4] C. Gilbert and M. A. Gilbert, “Exploring Secure Hashing Algorithms for Data Integrity Verification,” vol. 7, no. 11, pp. 373–390, 2025.

[5] Z. Gong, Z. Zheng, Z. Hu, K. Tian, Y. Zhang, and Z. Oleksiy, “Authenticated Private Set Intersection : A *Merkle Tree*-Based Approach for Enhancing Data Integrity,” 2025.

[6] O. Kuznetsov, A. Rusnak, A. Yezhov, K. Kuznetsova, D. Kanonik, and O. Domin, “Evaluating the Security of *Merkle Tree*s : An Analysis of Data Falsification Probabilities,” 2024.

[7] M. R. Zayana, “Penerapan Message Diggest Algorithm MD5 untuk Pengamanan Data Karyawan PT. Swifect Berbasis Desktop,” vol. 6, no. 3, 2022.

[8] A. F. Wulandari and A. Suryadi, “Media Teknologi dan Informatika Implementasi Arsip Digital Berbasis Web Dengan Integrasi QR Code Untuk Berkas Fisik Media Teknologi dan Informatika,” vol. 2, pp. 161–167, 2025.

[9] C. Liang, J. Zhang, S. Ma, Y. Zhou, Z. Hong, and J. Fang, “Study on data storage and verification methods based on improved Merkle mountain range in IoT scenarios,” J. King Saud Univ. - Comput. Inf. Sci., vol. 36, no. 6, p. 102117, 2025, doi: 10.1016/j.jksuci.2024.102117.

[10] D. D. P. Sari, “Analysis of Electronic Document Management System (EDMS) Implementation at Company X,” pp. 1133–1147, 2020.

[11] J. C. Santillán-lima et al., Preservation of Digital Documents : Strategic Models and International Standards for Institutional Management, no. Icomta 2025. Atlantis Press International BV, 2025. doi: 10.2991/978-94-6463-868-4.

[12] O. Kuznetsov, “*Merkle Tree*s in Blockchain : A Study of Collision Probability and Security Implications,” pp. 1–17, 2021.

[13] G. Szyjewski, “Securing Digital Copies of the Documents to Ensure Documents ’ Integrity,” vol. XXVI, no. 4, pp. 718–726, 2023.