

Perancangan Manajemen Dokumen Cloud yang Aman, Auditabel, dan Mempertahankan Integritas Menggunakan Laravel dan Google Drive

Indri Anjar Kartika Sari^{1*}, Agung Budi Prasetyo², Onno W. Purbo³

¹ Magister Ilmu Komunikasi, Universitas Pembangunan Nasional “Veteran” Jawa Timur, Indonesia

² Faculty of Computer Science, Institut Teknologi Tangerang Selatan, Indonesia

³ Independent Researcher, Indonesia

*Corresponding email: indri.cs2026@ids.ac.id; agung@itts.ac.id; onno@indo.net.id

Abstrak: Pengelolaan dokumen berbasis cloud memberikan kapasitas dan akses yang fleksibel, tetapi pemindahan penyimpanan ke layanan eksternal meningkatkan kebutuhan terhadap otorisasi, integritas, akuntabilitas, dan auditabilitas. Penelitian ini merancang arsitektur manajemen dokumen yang mengintegrasikan Laravel dan Google Drive dengan role-based access control, izin tingkat dokumen, metadata, fingerprint SHA-256, audit trail tamper-evident, dan mekanisme berbagi terkontrol. Metode penelitian menggunakan perancangan arsitektur, pemodelan ancaman, perumusan skenario pengujian keamanan, serta rancangan evaluasi kinerja dan biaya. Artefak implementasi diperlakukan sebagai dasar rancangan; nilai kinerja empiris belum diisi sebelum pengujian dijalankan. Kontribusi penelitian terletak pada integrasi kontrol keamanan pada lapisan aplikasi dan protokol evaluasi yang dapat direproduksi.

Kata kunci: manajemen dokumen cloud, Laravel, Google Drive, kontrol akses, audit trail, integritas data

Abstract: Cloud-based document management provides scalable storage and remote access, but externalizing storage increases the need for authorization, integrity verification, accountability, and auditability. This study designs a secure document-management architecture integrating Laravel and Google Drive with role-based access control, document-level permissions, metadata, SHA-256 fingerprints, tamper-evident audit trails, and controlled sharing. The method combines architecture design, threat modeling, security-test design, and a reproducible framework for performance and cost evaluation. The implementation artifact is treated as the basis for the proposed architecture; empirical performance values are intentionally not fabricated before execution. The contribution is an application-layer security architecture and an evaluation protocol that can be tested end-to-end.

Keywords: cloud document management, Laravel, Google Drive, access control, audit trail, data integrity

PENDAHULUAN

Organisasi semakin menggunakan penyimpanan cloud untuk memperoleh kapasitas yang elastis, akses jarak jauh, dan pengurangan kebutuhan pemeliharaan infrastruktur. Perpindahan dokumen ke penyedia eksternal, bagaimanapun, mengubah batas pengendalian keamanan karena file tidak lagi sepenuhnya berada pada infrastruktur yang dikelola organisasi. Das et al. (2024) menegaskan bahwa “Secure storage and access to data in the cloud is critical.” Pernyataan tersebut menunjukkan bahwa manfaat skalabilitas cloud harus disertai kontrol yang menjaga kerahasiaan, integritas, serta ketepatan akses. Risiko tersebut juga berkaitan dengan berkurangnya kendali langsung pemilik data; Wang et al. (2024) menyatakan bahwa “data stored in the cloud is beyond the control of users, posing various security risks.” Karena itu, tanggung jawab aplikasi tetap mencakup autentikasi, otorisasi, pengelolaan metadata, kebijakan berbagi, verifikasi integritas, dan penyediaan bukti aktivitas pengguna.

Integrasi Laravel dengan Google Drive dapat menyelesaikan kebutuhan dasar unggah dan unduh dokumen, tetapi integrasi penyimpanan semata belum cukup untuk membentuk kontribusi penelitian yang kuat. Persoalan yang lebih penting adalah bagaimana lapisan aplikasi mempertahankan kerahasiaan, integritas, ketepatan otorisasi, akuntabilitas, dan kinerja ketika objek biner disimpan pada layanan cloud eksternal. Literatur menunjukkan bahwa mekanisme berbagi memerlukan kontrol yang lebih granular. Liu et al. (2024) menjelaskan bahwa mekanisme key-aggregate sebelumnya “can only control what data users are allowed to receive ... but not what data users can send.” Keterbatasan ini menguatkan

kebutuhan keputusan akses pada tingkat objek dan tindakan. Almasian dan Shafieinejad (2024) juga menyatakan bahwa “Blockchain enables us to implement access control as a smart contract between data owner and users.” Penelitian ini tidak mengadopsi blockchain sebagai komponen inti, tetapi prinsip pemisahan kebijakan akses dari penyimpanan menjadi pijakan konseptual untuk menempatkan Laravel sebagai policy enforcement layer.

Penelitian ini merancang arsitektur manajemen dokumen yang menempatkan Laravel sebagai lapisan kebijakan dan logika aplikasi, sedangkan Google Drive berfungsi sebagai penyedia penyimpanan eksternal. Arsitektur mencakup metadata dokumen, hubungan kepemilikan dan peran, izin tingkat dokumen, catatan audit, fingerprint integritas, serta mekanisme berbagi terkontrol. Fokus tersebut selaras dengan kecenderungan penelitian cloud auditing yang tidak hanya menilai keamanan secara teoritis, tetapi juga memeriksa implementabilitas dan biaya. Chen et al. (2024) menegaskan bahwa banyak skema audit terdahulu bersifat “theoretical in nature” dan “not fit for existing mainstream cloud storage service practices.” Kesenjangan inilah yang menjadi dasar perlunya rancangan yang dapat diimplementasikan dan dievaluasi pada lingkungan aplikasi nyata.

Rumusan masalah penelitian dirumuskan sebagai berikut:

RQ	Rumusan masalah
RQ1	Bagaimana arsitektur Laravel–Google Drive dapat menyediakan operasi dokumen yang aman, akuntabel, dan auditabel?
RQ2	Risiko kontrol akses, integritas, berbagi, dan audit apa yang masih terdapat pada integrasi penyimpanan cloud konvensional?
RQ3	Berapa overhead kinerja yang ditimbulkan oleh otorisasi, verifikasi integritas, dan audit logging?
RQ4	Bagaimana arsitektur yang diusulkan dibandingkan dengan konfigurasi baseline dalam keamanan, kinerja, keandalan, dan biaya?

Kontribusi penelitian diposisikan pada pertemuan antara keamanan aplikasi dan penyimpanan cloud. Penelitian tidak mengklaim penciptaan primitive kriptografi baru, melainkan mengusulkan arsitektur yang dapat diimplementasikan serta protokol evaluasi yang memungkinkan bukti empiris diperoleh dari artefak perangkat lunak.

LANDASAN TEORI

Manajemen dokumen cloud. Dalam arsitektur ini, penyimpanan cloud diperlakukan sebagai lapisan infrastruktur eksternal, sedangkan basis data aplikasi mempertahankan metadata, identitas, hubungan otorisasi, dan bukti audit. Pemisahan tersebut membuat keputusan keamanan tidak diserahkan kepada lokasi penyimpanan objek. Chen et al. (2024) mendefinisikan fungsi audit secara ringkas: “Cloud storage auditing is a technique that enables a user to remotely check the integrity of the outsourced data.” Konsep ini mendukung kebutuhan untuk menyimpan bukti yang memungkinkan aplikasi melakukan pemeriksaan terhadap objek yang telah dialihdayakan ke Google Drive tanpa memperlakukan penyedia cloud sebagai satu-satunya sumber kepercayaan.

Kontrol akses. Role-based access control (RBAC) digunakan sebagai dasar pengaturan hak aplikasi, tetapi manajemen dokumen memerlukan keputusan pada tingkat objek karena pengguna dengan peran yang sama dapat memiliki hak berbeda terhadap dokumen tertentu. RBAC karena itu dikombinasikan dengan izin tingkat dokumen, aturan kepemilikan, scope, serta masa berlaku berbagi. Liu et al. (2024) menekankan perlunya kontrol dua arah dengan tujuan “control both what the user can receive and what the data owner can send.” Perspektif tersebut memperkuat desain fine-grained authorization agar hak upload, download, delete, dan share tidak ditentukan hanya oleh label peran global.

Integritas data. Fingerprint SHA-256 digunakan sebagai indikator bahwa objek yang diambil kembali masih sesuai dengan fingerprint yang dicatat saat unggah. Nilai hash tidak membuktikan asal-usul atau siapa yang membuat dokumen; fungsinya terbatas pada pemeriksaan kesesuaian konten. Wen dan Deng (2024) menyatakan bahwa “Data integrity auditing provides a method for checking the integrity of outsourced data in cloud storage.” Wang et al. (2024) menambahkan bahwa audit integritas dapat dilakukan “without the necessity of data downloading.” Penelitian ini menggunakan prinsip pemeriksaan integritas tersebut secara lebih sederhana pada lapisan aplikasi melalui fingerprint SHA-256, bukan mengklaim sebagai protokol public auditing kriptografis baru.

Audit trail. Setiap event audit dihubungkan dengan hash event sebelumnya sehingga perubahan terhadap event lama dapat terdeteksi ketika rantai divalidasi kembali. Mekanisme ini bersifat tamper-evident, bukan jaminan bahwa penyimpanan audit tidak dapat dihapus oleh administrator dengan hak istimewa tinggi. Kumar dan Bhatia (2024) menyebut bahwa “The cloud auditing model was attractive for the cloud storage system to check the integrity of the stored user files.” Audit pada penelitian ini diperluas dari pemeriksaan file menjadi pencatatan aktor, tindakan, objek, outcome, waktu, dan keterkaitan hash agar aktivitas keamanan dapat ditelusuri. Gan et al. (2024) juga menunjukkan pentingnya efisiensi audit melalui pemisahan proses, karena “the data auditing process ... is divided to online and offline phases,” sehingga overhead audit menjadi bagian yang perlu diukur secara eksplisit.

METODE

Penelitian menggunakan pendekatan perancangan dan evaluasi sistem. Artefak yang menjadi dasar adalah proyek Laravel dengan integrasi Google Drive yang direpresentasikan dalam repository sebagai laravel-google-drive-storage-main.zip. Pada tahap penyusunan naskah ini, isi ZIP belum tersedia untuk pemeriksaan source-code secara baris per baris sehingga klaim implementasi diposisikan sebagai rancangan yang harus diverifikasi terhadap kode sebelum publikasi. Pendekatan ini dipilih karena studi audit cloud yang berorientasi praktik perlu menghubungkan rancangan dengan kebutuhan sistem nyata. Chen et al. (2024, Highlights) melaporkan, “We conducted real-world requirement analysis of a cloud storage auditing system.” Prinsip yang sama diterapkan melalui identifikasi aset, ancaman, kontrol, skenario uji, metrik kinerja, dan biaya yang harus dibuktikan melalui implementasi aktual.

Model ancaman mencakup pengguna eksternal tanpa otorisasi, akun sah yang telah dikompromikan, insider yang berbahaya atau ceroboh, serta penyerang yang mengeksploitasi aplikasi web. Aset yang dilindungi meliputi isi dokumen, metadata, identitas pengguna, kredensial atau token OAuth, catatan izin, audit log, dan konfigurasi sistem. Pemodelan ancaman juga mempertimbangkan kebocoran informasi ketika fungsi berbagi digunakan. Guan et al. (2024) mengingatkan bahwa dalam proses berbagi, “unauthorized entities may have access to these privacy fields.” Ancaman tersebut diterjemahkan menjadi skenario pengujian berupa broken access control, privilege escalation, unauthorized write/delete, sharing terlalu luas, kebocoran metadata, serta manipulasi log dan kredensial.

ID	Ancaman	Permukaan serangan	Kontrol utama
T01	Unduh tanpa otorisasi	Broken access control	Middleware otorisasi + pemeriksaan izin objek + audit
T02	Privilege escalation	Manipulasi role/permission	Kebijakan server-side + pemisahan administrasi
T03	Penggantian dokumen	Unauthorized write/delete	Pemeriksaan kepemilikan + fingerprint
T04	Manipulasi audit	Penghapusan/perubahan log	Rantai audit tamper-evident + pembatasan akses

T05	Kebocoran kredensial	Source code/log	Secret management + redaksi log + rotasi
T06	Sharing terlalu luas	Permission sharing	Scope eksplisit + expiry/revocation
T07	Kesalahan pemetaan objek	Object/path confusion	ID dokumen kanonik + metadata server-side
T08	Replay/duplikasi operasi	Request berulang	Idempotency pada operasi tulis tertentu
T09	Kehilangan data	Kegagalan cloud/API	Recovery + versioning/backup bila tersedia
T10	Kebocoran metadata	Search/list endpoint	Authorization filtering pada query/service

Gambar 1 menggambarkan arsitektur konseptual: pengguna mengakses aplikasi melalui HTTPS; Laravel menjalankan autentikasi, otorisasi, business logic, dan audit; MySQL menyimpan metadata; layanan integritas menghitung SHA-256; dan Google Drive menyimpan objek biner. Evaluasi dirancang untuk mengisolasi dampak kontrol keamanan terhadap sistem. Gan et al. (2024) menjelaskan bahwa pembagian proses audit dapat “greatly relieving the online computational burden,” sehingga rancangan eksperimen penelitian ini secara khusus memasukkan authorization overhead, audit overhead, dan integrity overhead sebagai metrik terpisah, bukan hanya latency total.

Komponen	Fungsi
Browser/Client	Antarmuka pengguna melalui HTTPS
Laravel	Autentikasi, otorisasi, business logic, dan audit
MySQL	Users, roles, documents, folders, permissions, audit
Integrity Service	Fingerprint dan verifikasi SHA-256
Google Drive	Penyimpanan objek biner eksternal

Gambar 1. Arsitektur keamanan yang diusulkan

HASIL DAN PEMBAHASAN

Hasil tahap perancangan adalah model arsitektur yang memisahkan identitas, otorisasi, metadata dokumen, hierarki folder, dan bukti audit. Model data inti terdiri atas users, roles, user_roles, folders, documents, document_permissions, dan audit_logs. Record documents menyimpan referensi objek Google Drive dan fingerprint SHA-256, sedangkan audit_logs menyimpan actor, action, object, outcome, timestamp, previous_hash, dan event_hash. Hasil ini merepresentasikan prinsip bahwa keamanan data cloud tidak cukup bergantung pada autentikasi akun. Das et al. (2024) menunjukkan bahwa controlled access dapat gagal menjaga privasi akibat “insider attacks, breaches of data security, or any other types of unauthorized access.” Struktur data yang diusulkan karena itu menyediakan konteks server-side untuk memutuskan bukan hanya siapa pengguna, tetapi juga objek apa yang boleh diakses dan tindakan apa yang diperbolehkan.

Entitas	Tujuan	Atribut representatif
users	Identitas dan status akun	id, name, email, status, timestamps
roles	Peran aplikasi	id, name, description
user_roles	Relasi user-role	user_id, role_id
folders	Hierarki folder	id, parent_id, owner_id, name
documents	Metadata dan	drive_file_id, name, mime_type, size,

	referensi cloud	sha256
document_permissions	Otorisasi tingkat objek	document_id, principal_id, permission, expiry
audit_logs	Bukti aktivitas keamanan	actor_id, action, object_id, outcome, previous_hash, event_hash

Tabel 1. Entitas inti system

Urutan keputusan otorisasi dirancang berlapis: autentikasi principal, pemeriksaan status akun, resolusi dokumen dari metadata server-side, evaluasi hak berbasis peran, evaluasi kepemilikan atau permission dokumen, penerapan scope dan expiry sharing, pemberian izin hanya jika seluruh kondisi terpenuhi, serta pencatatan event audit. Pendekatan ini mencegah aplikasi mempercayai file identifier atau keputusan izin yang dikirimkan oleh klien. Almasian dan Shafieinejad (2024) menunjukkan pentingnya kebijakan akses eksplisit karena pemilik data dapat menetapkan “access policy to the file as well as preserving user anonymity.” Pada arsitektur yang diusulkan, fungsi tersebut diwujudkan melalui kombinasi RBAC dan document_permissions sehingga keputusan akses tetap berada pada server aplikasi.

Operasi	Administrator	Manager	Staff	Auditor
View metadata	Ya	Ya	Ya	Ya
Download	Ya	Ya*	Ya*	Terkontrol
Upload	Ya	Ya	Ya	Tidak
Delete	Ya	Owner/policy	Owner/policy	Tidak
Share	Ya	Policy	Policy	Tidak
Manage users	Ya	Tidak	Tidak	Tidak
View audit	Ya	Selected	Own scope	Read-only scope

Tabel 2. Matriks otorisasi ilustratif (*harus diverifikasi dan ditetapkan secara eksplisit pada implementasi)

Untuk integritas dokumen, sistem menghitung $H = \text{SHA-256}(D)$ saat unggah. Ketika verifikasi dilakukan, sistem menghitung H' dari objek yang diambil kembali. Kesamaan H dan H' menunjukkan kesesuaian konten dengan fingerprint yang direkam; perbedaan menunjukkan anomali integritas. Pengujian harus mencakup modifikasi satu byte, truncation, penggantian objek, ketidakcocokan metadata, dan objek yang hilang. Wen dan Deng (2024) menekankan bahwa data outsourcing dapat membawa “risks of exposure during data sharing,” sehingga pemeriksaan integritas perlu berjalan bersama kontrol kerahasiaan dan sharing. Fingerprint pada rancangan ini tidak menggantikan skema certificateless auditing, tetapi menjadi kontrol implementatif yang ringan untuk mendeteksi perubahan objek.

Untuk audit, event ke- i dirangkai sebagai $H_i = \text{SHA-256}(\text{Event}_i \parallel H_{i-1})$. Perubahan terhadap event terdahulu akan mengubah rantai dan dapat terdeteksi saat validasi. Mekanisme ini tidak menghalangi administrator yang memiliki hak untuk menghapus seluruh penyimpanan audit sehingga lingkungan dengan kebutuhan assurance tinggi tetap memerlukan pembatasan otoritas tulis dan, bila diperlukan, retensi log eksternal. Literatur juga menunjukkan bahwa audit yang aman harus mempertimbangkan pihak pemeriksa. Wang et al. (2024) mengingatkan bahwa “malicious third-party auditors can compromise data privacy.” Temuan tersebut mendukung desain audit yang memisahkan kemampuan membaca log, mengubah log, dan menjalankan validasi rantai.

Rancangan Evaluasi

Evaluasi empiris dirancang untuk membandingkan empat konfigurasi agar overhead setiap mekanisme dapat diisolasi. Baseline A adalah Laravel dengan penyimpanan lokal dan autentikasi dasar; B adalah Laravel + Google Drive; C adalah Laravel + Google Drive + RBAC + audit terstruktur; D adalah arsitektur yang diusulkan dengan otorisasi fine-grained, verifikasi integritas, dan audit tamper-evident. Desain pembandingan diperlukan karena penguatan keamanan dapat menambah biaya komputasi dan penyimpanan. Yan et al. (2024), misalnya, melaporkan bahwa pada skema yang mereka usulkan “the storage overhead for accountability is also extremely low.” Klaim efisiensi semacam itu hanya dapat dibandingkan secara bermakna apabila penelitian ini mengukur overhead secara eksplisit pada workload dan konfigurasi yang terdokumentasi.

Faktor	Level yang disarankan
Ukuran file	1 MB, 10 MB, 50 MB, 100 MB, 500 MB; disesuaikan lingkungan
Concurrency	1, 5, 10, 25, 50 request bersamaan
Operasi	Upload, download, metadata read, search, permission check
Authorization	Allowed, denied, expired share, wrong owner
Integrity	Normal, mutated, replaced, missing
Failure	Network interruption, provider timeout, application restart
Replikasi	Minimal 20–30 pengukuran per sel jika memungkinkan

Tabel 3. Faktor eksperimen

Metrik kinerja meliputi upload latency, download latency, throughput, CPU/RAM, authorization overhead, audit overhead, integrity overhead, unauthorized detection, audit completeness, integrity detection, recovery, dan biaya. Statistik deskriptif mencakup ukuran pemusatan dan dispersi; median dan interquartile range dapat digunakan ketika latency cloud tidak berdistribusi normal. Pengukuran biaya juga relevan karena keamanan yang kuat harus tetap layak dijalankan. Chen et al. (2024, Highlights) menyatakan bahwa model biaya dapat membantu pengguna “balance security, efficiency, and economic cost.” Karena itu, evaluasi tidak berhenti pada keberhasilan kontrol keamanan, tetapi juga menilai konsekuensi performa dan biaya dari setiap konfigurasi.

Hasil Pengujian yang Harus Diisi

Berdasarkan berkas kerja yang menjadi sumber, nilai numerik hasil eksperimen belum tersedia. Oleh karena itu, bagian ini tidak boleh diisi dengan angka yang dibuat-buat. Tabel berikut merupakan struktur pelaporan yang harus diisi setelah implementasi dijalankan.

Konfigurasi	Ukuran	Concurrency	Median upload (ms)	Median download (ms)	CPU (%)	RAM (MB)
A	—	—	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED
B	—	—	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED
C	—	—	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED
D	—	—	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED	TO BE MEASURED

Tabel 4. Template hasil kinerja; seluruh nilai harus berasal dari pengukuran aktual.

Kontrol	Jumlah uji	Pass	Fail	Detection rate	Bukti
Authorization	TO BE MEASURED	—	—	—	Raw test log
Integrity	TO BE MEASURED	—	—	—	Mutation dataset
Audit completeness	TO BE MEASURED	—	—	—	Event reconciliation
Audit-chain validation	TO BE MEASURED	—	—	—	Chain verification log

Tabel 5. Template validasi keamanan.

Pembahasan

Interpretasi hasil harus menekankan trade-off antara keamanan dan kinerja, bukan sekadar mengulang angka. Jika hashing menghasilkan tambahan latency, analisis perlu menentukan apakah overhead terutama berasal dari ukuran file, CPU, jaringan, atau I/O. Jika audit logging meningkatkan penggunaan storage, penambahan evidence per operasi dan strategi retensi harus dilaporkan. Gan et al. (2024) menyatakan bahwa rancangan mereka bertujuan “achieve the integrity verification for cloud data efficiently,” sehingga efisiensi perlu dibaca sebagai karakteristik yang diuji, bukan diasumsikan. Prinsip yang sama digunakan dalam penelitian ini: mekanisme keamanan baru dianggap layak apabila peningkatan kontrol dapat ditunjukkan bersama overhead yang terukur dan dapat diterima.

Perbandingan dengan penelitian terdahulu perlu memperhatikan perbedaan arsitektur, workload, cloud provider, mekanisme keamanan, dan metodologi pengukuran. Perbandingan numerik hanya bermakna apabila kondisi eksperimen cukup sebanding. Temuan Liu et al. (2024) bahwa skema mereka menjamin “security against unauthorized receivers and malicious senders” menunjukkan bahwa evaluasi akses tidak cukup hanya menguji pengguna yang tidak login; pengujian juga perlu mencakup pengguna sah dengan hak yang salah, pemilik yang mencoba berbagi di luar scope, serta share yang telah kedaluwarsa. Literatur cloud auditing yang praktis selanjutnya menunjukkan pentingnya memasukkan biaya komputasi, storage, bandwidth, dan request dalam evaluasi.

Implikasi desain yang utama adalah bahwa Google Drive sebaiknya diperlakukan sebagai storage provider, bukan sebagai pengganti lapisan kebijakan aplikasi. Keputusan otorisasi harus tetap berada di server, metadata harus menjadi sumber keputusan, dan kredensial provider tidak boleh diekspos kepada browser. Guan et al. (2024) menggarisbawahi bahwa data sharing pada cloud dapat menghadirkan kebutuhan untuk menyembunyikan informasi sensitif, sedangkan Almasian dan Shafieinejad (2024) menunjukkan bahwa kebijakan akses perlu melekat pada proses berbagi. Rancangan Laravel–Google Drive karena itu memusatkan policy enforcement, audit, dan integrity checking pada aplikasi, sementara provider hanya menerima operasi penyimpanan yang telah lolos keputusan keamanan server-side.

SIMPULAN

Penelitian ini merancang arsitektur manajemen dokumen cloud yang mengintegrasikan Laravel dan Google Drive dengan kontrol akses, fingerprint SHA-256, audit trail tamper-evident, controlled sharing, dan protokol evaluasi yang dapat direproduksi. Kontribusi utama bukan integrasi teknologi semata, melainkan pemisahan lapisan penyimpanan, kebijakan keamanan, bukti audit, dan evaluasi empiris. Tahap berikutnya adalah membuka dan mengaudit source code aktual, melakukan deployment terkontrol, menjalankan pengujian

keamanan dan kinerja, serta mengganti seluruh placeholder hasil dengan observasi yang dapat ditelusuri.

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih kepada pihak yang mendukung pengembangan artefak dan proses penelitian. Kontribusi yang bersifat kepenulisan tercantum pada daftar penulis.

DAFTAR PUSTAKA

- Almasian, M., & Shafieinejad, A. (2024). Secure cloud file sharing scheme using blockchain and attribute-based encryption. *Computer Standards & Interfaces*, 87, 103745. <https://doi.org/10.1016/j.csi.2023.103745>
- Chen, F., Meng, F., Li, Z., Li, L., & Xiang, T. (2024). Public cloud object storage auditing: Design, implementation, and analysis. *Journal of Parallel and Distributed Computing*, 189, 104870. <https://doi.org/10.1016/j.jpdc.2024.104870>
- Das, S., Mishra, M., Priyadarshini, R., Barik, R. K., & Saikia, M. J. (2024). A secure, privacy-preserving, and cost-efficient decentralized cloud storage framework using blockchain. *Journal of King Saud University—Computer and Information Sciences*, 36(10), 102260. <https://doi.org/10.1016/j.jksuci.2024.102260>
- Gan, Q., Wang, X., Huang, D., Li, J., Wang, C., & Liu, Z. (2024). Online/offline remote data auditing with strong key-exposure resilience for cloud storage. *Computer Standards & Interfaces*, 88, 103798. <https://doi.org/10.1016/j.csi.2023.103798>
- Guan, X., Chang, J., & Zhang, W. (2024). Secure data sharing scheme with privacy-preserving and certificateless integrity auditing in cloud storage. *Computer Communications*, 224, 285–301. <https://doi.org/10.1016/j.comcom.2024.06.013>
- Kumar, R., & Bhatia, M. P. S. (2024). An intelligent optimized secure blockchain mechanism for cloud auditing. *Expert Systems with Applications*, 124593. <https://doi.org/10.1016/j.eswa.2024.124593>
- Liu, J., Qin, J., Wang, W., Mei, L., & Wang, H. (2024). Key-aggregate based access control encryption for flexible cloud data sharing. *Computer Standards & Interfaces*, 88, 103800. <https://doi.org/10.1016/j.csi.2023.103800>
- Wen, J., & Deng, L. (2024). Certificateless integrity auditing scheme for sensitive information protection in cloud storage. *Journal of Systems Architecture*, 156, 103267. <https://doi.org/10.1016/j.sysarc.2024.103267>
- Wang, H., Zhang, Y., Wang, X. A., & Yang, X. (2024). An improved identity-based public audit protocol for cloud storage. *Heliyon*, 10(16), e36273. <https://doi.org/10.1016/j.heliyon.2024.e36273>
- Yan, L., Wang, G., Feng, H., Liu, P., Gao, H., Zhang, W., Hu, H., & Pan, F. (2024). Efficient and accountable anti-leakage attribute-based encryption scheme for cloud storage. *Heliyon*, 10(12), e32404. <https://doi.org/10.1016/j.heliyon.2024.e32404>