EVALUASI ARSITEKTUR SMART CONTRACT UNTUK PRA-VERIFIKASI PORT STATE CONTROL

Senri Ali Said*1, Asnur2, Kamaruddin3

[1] Politeknik Ilmu Pelayaran Balikpapan, Kalimantan Timur, Indonesia
[2] Politeknik Ilmu Pelayaran Makassar, Sulawesi Selatan, Indonesia
[3] Universitas Handayani Makassar, Sulawesi Selatan, Indonesia

[1]senrialisaid@pip-balikpapan.ac.id ,
[2]asnur@pipmakassar.ac.id,
[3]k4m4.1t@handayani.ac.id

ABSTRAK

Verifikasi administratif dokumen dan sertifikat merupakan instrumen pendukung dalam kerangka Port State Control (PSC), sedangkan wewenang pemeriksaan fisik, penetapan defisiensi, dan tindakan penegakan hukum tetap berada sepenuhnya pada Port State Control Officer (PSCO). Penelitian ini merancang dan mengevaluasi tiga arsitektur smart contract berbasis Ethereum Virtual Machine (EVM) untuk alur pra-verifikasi administratif: otorisasi berbasis peran (RBAC), komitmen hash, dan skema mock verification yang mengadopsi alur interface Zero-Knowledge Proof (ZKP). Ketiga smart contract diimplementasikan menggunakan Solidity 0.8.24 dengan optimasi 200 runs dan dievaluasi pada lingkungan Hardhat EVM lokal. Pengujian performa dilakukan melalui 1.000 transaksi sekuensial per arsitektur untuk memetakan kebutuhan komputasi dan latensi eksekusi. Hasil empiris menunjukkan median execution gas masing-masing sebesar 48.429 gas (konvensional), 71.507 gas (komitmen hash), dan 117.686 gas (mock verification), dengan overhead sebesar 143,00% pada arsitektur mock verification terhadap baseline konvensional. Sementara itu, median wall-clock latency pada lingkungan lokal tercatat sebesar 2,211 ms, 2,232 ms, dan 2,515 ms. Evaluasi fungsional mengonfirmasi efektivitas penegakan kontrol akses, validasi predicate, serta pencegahan transaksi berulang (anti-replay) melalui mekanisme nullifier. Penelitian ini menyajikan baseline empiris mengenai alur kontrol dan beban komputasi on-chain guna mendukung integrasi sistem administrasi pelabuhan di masa depan.

Kata kunci: Port State Control; blockchain; smart contract; pra-verifikasi administratif; EVM

EXPERIMENTAL EVALUATION OF SMART CONTRACT ARCHITECTURES FOR ADMINISTRATIVE PRE-VERIFICATION IN PORT STATE CONTROL

ABSTRACT

Administrative verification of ship certificates and statutory documents provides essential operational support for Port State Control (PSC), whereas physical inspection, deficiency determination, and detention authority strictly reside with certified Port State Control Officers (PSCOs). This study designs and experimentally evaluates three Ethereum Virtual Machine (EVM)-based smart contract architectures for administrative pre-verification: role-based access control (RBAC), cryptographic hash commitment, and a mock verification scheme inspired by the Zero-Knowledge Proof (ZKP) control flow. All contracts were developed using Solidity 0.8.24 with 200 optimizer runs and evaluated on a local Hardhat EVM environment. Performance benchmarks comprising 1,000 sequential transactions per architecture were conducted to quantify computational overhead and execution latency. Empirical measurements demonstrate median execution gas costs of 48,429 gas (conventional), 71,507 gas (hash commitment), and 117,686 gas (mock verification), representing a 143.00% overhead for the mock architecture relative to the baseline. Median local wall-clock latencies were measured at 2.211 ms, 2.232 ms, and 2.515 ms, respectively. Functional testing verified the robust enforcement of role-based permissions, verification predicate integrity, and replay attack prevention using a one-time nullifier mechanism. This work establishes a rigorous empirical baseline for on-chain state transition costs and control flows to inform future blockchain-based maritime administrative integration.

Keywords: Port State Control; blockchain; smart contract; administrative pre-verification; EVM

1. PENDAHULUAN

Port State Control (PSC) merupakan rezim inspeksi terpadu yang dilaksanakan oleh otoritas negara pelabuhan untuk memastikan bahwa kapal-kapal berbendera asing yang memasuki yurisdiksinya memenuhi standar kelaiklautan, keselamatan jiwa di laut, keamanan maritim, pencegahan pencemaran lingkungan, serta kelayakan kondisi kerja dan hidup awak kapal sesuai konvensi internasional. Sebagaimana ditegaskan dalam resolusi International Maritime Organization melalui Procedures for Port State Control, 2025 (Resolution A.1206(34)), harmonisasi prosedur inspeksi, identifikasi ketidaksesuaian (deficiencies), serta penetapan tindakan penegakan hukum menuntut akurasi data administratif yang tinggi. Keputusan PSC memiliki implikasi hukum dan komersial yang signifikan, sehingga penerapan teknologi informasi mutlak diposisikan sebagai instrumen pendukung verifikasi administratif dokumen dan pembentukan jejak audit yang reliabel, bukan sebagai pengambil alih diskresi profesional Port State Control Officer (PSCO) [1].

Dalam lanskap operasional pelabuhan modern, proses pra-kedatangan dan verifikasi administratif melibatkan interaksi kompleks antara beragam entitas, termasuk agen pelayaran, otoritas pelabuhan, badan klasifikasi (Recognized Organization), operator terminal, sistem kepabeanan, serta platform pertukaran data terpadu seperti Maritime Single Window (MSW) dan Port Community System (PCS). Fragmentasi pangkalan data antarorganisasi sering kali menimbulkan redundansi pemeriksaan dokumen sertifikasi, meningkatkan friksi administratif, dan menyulitkan pelacakan integritas jejak audit. Teknologi blockchain menawarkan kapabilitas state machine terdesentralisasi dengan eksekusi deterministik dan pencatatan riwayat transaksi yang tahan manipulasi (tamper-evident) [2], [3]. Penerapan arsitektur distributed ledger dalam ekosistem maritim berpotensi memperkuat integritas data, menjamin ketertelusuran dokumen statutoria, serta memfasilitasi koordinasi lintas pemangku kepentingan [4], [5].

Meskipun demikian, transparansi penuh pada buku besar terdistribusi dapat menimbulkan tantangan privasi komersial. Dokumen kepatuhan kapal sering memuat informasi operasional sensitif. Konsep Zero-Knowledge Proof (ZKP) secara teoretis menawarkan mekanisme pembuktian kepatuhan suatu pernyataan tanpa mengekspos data saksi privat (private witness) kepada publik [6]–[8]. Namun, penerapan sistem pembuktian kriptografis ZKP on-chain secara penuh memerlukan sumber daya komputasi yang substansial. Sebelum melangkah pada implementasi sirkuit kriptografis yang kompleks, evaluasi arsitektural terhadap desain antarmuka smart contract, alur kontrol logika, manajemen komitmen, serta mitigasi serangan pemutaran ulang (replay attacks) menjadi langkah fundamental yang diperlukan.

Penelitian ini membandingkan tiga arsitektur smart contract pada ekosistem EVM untuk alur pra-verifikasi administratif PSC: (1) arsitektur konvensional berbasis otorisasi peran (RBAC), (2) arsitektur berbasis komitmen hash dokumen, dan (3) arsitektur mock verification yang memodelkan alur antarmuka verifikasi proof dan pencegahan replay menggunakan nullifier. Penelitian ini menetapkan baseline empiris terkait alokasi gas eksekusi, latensi komputasi, serta keandalan penegakan aturan akses pada smart contract.

1.1 Kesenjangan Penelitian dan Kebaruan

Kajian literatur mengenai penerapan blockchain dalam sektor maritim sebagian besar berfokus pada kerangka adopsi konseptual, pertukaran informasi kargo, dan rantai pasok maritim secara umum (Tabel 1). Sebaliknya, literatur ZKP berfokus pada efisiensi konstruksi kriptografis tingkat rendah. Masih sangat jarang ditemukan investigasi eksperimental yang mengkaji secara spesifik struktur biaya komputasi on-chain dan desain alur kontrol smart contract yang disesuaikan dengan kebutuhan tata kelola administratif Port State Control.

Kebaruan penelitian ini terletak pada perancangan dan evaluasi komparatif arsitektur smart contract PSC yang mengintegrasikan: (a) kontrol akses berbasis peran (RBAC) bagi agen kapal yang terotorisasi, (b) skema penyimpanan komitmen hash dokumen untuk mencegah pemaparan dokumen penuh on-chain, (c) mekanisme registri nullifier satu kali guna mencegah serangan pengajuan berulang (anti-replay), dan (d) benchmarking komparatif terhadap konsumsi gas eksekusi serta latensi pada lingkungan EVM yang terstandarisasi.

Studi	Fokus Utama	Metodologi / Pendekatan	Batasan / Pembeda terhadap Studi Ini
Jović et al. (2020)	Pertukaran informasi maritim berbasis blockchain	Tinjauan konseptual koordinasi dan keberlanjutan sektor pelabuhan	Tidak mengimplementasikan atau menguji smart contract verifikasi PSC.
Pu dan Lam (2021)	Adopsi teknologi blockchain maritim	Kerangka konseptual tata kelola dan kesiapan organisasi	Fokus pada aspek manajerial, tanpa evaluasi komputasi smart contract.

Studi	Fokus Utama	Metodologi / Pendekatan	Batasan / Pembeda terhadap Studi Ini
Sun et al. (2021)	Zero-Knowledge Proof dalam arsitektur blockchain	Survei komprehensif skema kriptografis dan aplikasinya	Kajian umum kriptografi, tidak spesifik pada proses administratif maritim.
Chi et al. (2023)	Protokol ZKP berorientasi privasi pada blockchain	Konstruksi kriptografis ZKP aktual dan efisiensi sirkuit	Fokus pada sirkuit matematis murni, tanpa integrasi alur kerja PSC.
Pacheco et al. (2023)	Analisis empiris performa transaksi Ethereum	Studi empiris faktor latensi pada jaringan publik	Mengevaluasi jaringan publik dengan beban umum, bukan alur domain maritim.
Penelitian ini	Pra-verifikasi administratif smart contract PSC	Eksperimen komparatif 3 arsitektur (RBAC, Commitment, Mock ZKP Interface)	Fokus pada baseline empiris biaya gas, kontrol alur EVM, dan anti-replay.

Tabel 1. Posisi penelitian terhadap literatur terdahulu.

1.2 Tujuan dan Pertanyaan Penelitian

Penelitian ini diarahkan untuk menjawab tiga pertanyaan penelitian utama (Research Questions):

• RQ1: Bagaimana arsitektur smart contract memisahkan peran aktor secara efektif antara agen kapal, smart contract, ledger EVM, otoritas pelabuhan, dan PSCO dalam pra-verifikasi administratif PSC?

• RQ2: Berapa overhead biaya komputasi (execution gas) dan latensi lokal yang dihasilkan oleh penambahan komitmen hash serta skema antarmuka mock verification dibandingkan otorisasi konvensional pada lingkungan EVM yang seragam?

• RQ3: Bagaimana efektivitas penegakan kontrol akses berbasis peran, validasi predikat hash, dan proteksi serangan replay berbasis nullifier saat dievaluasi melalui skenario pengujian fungsional terdefinisi?

1.3 LANDASAN TEORETIS DAN DESAIN KONSEPTUAL

1.3.1 Port State Control dan Batasan Otomatisasi

Port State Control bertindak sebagai benteng pertahanan terakhir terhadap operasional kapal-kapal substandar. Prosedur standar PSC mencakup pemeriksaan kelengkapan sertifikat statutoria kapal (seperti sertifikat SOLAS, MARPOL, STCW, dan MLC 2006). Meskipun pemeriksaan dokumen awal bersifat administratif, kewenangan penuh untuk melakukan inspeksi mendalam (more detailed inspection), menilai signifikansi defisiensi, hingga menerbitkan surat perintah penahanan kapal (detention order) mutlak berada pada diskresi PSCO yang berkualifikasi. Oleh karena itu, sistem smart contract dirancang strictly sebagai lapisan pra-verifikasi dokumenter dan penyimpanan bukti digital, sehingga memitigasi risiko kesalahan administratif pra-kedatangan tanpa mengintervensi otoritas hukum petugas berwenang.

1.3.2 Smart Contract, Integritas Data, dan Auditabilitas

Smart contract pada platform EVM beroperasi sebagai program deterministik yang mengikat secara komputasi sesuai logika bytecode yang dideploy. Pencatatan status pra-verifikasi ke dalam ledger menciptakan jejak audit yang tidak dapat diubah secara sepihak. Namun, penempatan dokumen berukuran besar langsung ke dalam state penyimpanan on-chain sangat tidak efisien dan berbiaya tinggi. Pendekatan penyimpanan cryptographic hash commitment memungkinkan smart contract memverifikasi integritas representasi data sertifikat tanpa mengekspos isi dokumen utuh ke dalam blockchain.

1.3.3 Alur Verifikasi Predikat dan Konsep Nullifier

Dalam perancangan sistem privasi modern, ZKP memisahkan peran pembukti (prover) dan pemverifikasi (verifier). Arsitektur mock verification dalam studi ini memodelkan relasi komitmen deterministik melalui antarmuka predikat hash internal: proofCommitment = keccak256(abi.encodePacked(documentCommitment, domainSeparator)). Domain separator disusun dari parameter runtime kontrak untuk memastikan isolasi konteks eksekusi. Guna mencegah transaksi valid diajukan ulang oleh pihak yang tidak bertanggung jawab (replay attack), diterapkan struktur pemetaan usedNullifiers. Nullifier unik yang disertakan dalam muatan transaksi dicatat status penggunaannya secara permanen saat eksekusi pertama berhasil, sehingga setiap upaya pengajuan lanjutan dengan nullifier yang sama secara otomatis ditolak (revert) oleh kontrak.

2. METODE PENELITIAN

2.1 Desain Eksperimental dan Ruang Lingkup

Penelitian ini menerapkan metode rekayasa perangkat lunak eksperimental (experimental software engineering). Eksperimen dirancang untuk mengisolasi dan mengukur variabel performa komputasi smart contract secara kuantitatif. Variabel bebas dalam penelitian ini adalah jenis arsitektur smart contract (Konvensional, Komitmen Hash, dan Mock Verification). Variabel terikat meliputi konsumsi deployment gas, execution gas, wall-clock latency pada lingkungan pengujian lokal, serta tingkat keberhasilan eksekusi skenario uji fungsional. Pengukuran difokuskan pada interaksi client-to-EVM receipt dalam lingkungan lokal terkontrol guna mendapatkan baseline komparatif yang deterministik.

2.2 Arsitektur Sistem dan Matriks Peran Aktor

Arsitektur pra-verifikasi administratif dirancang dengan memisahkan domain tanggung jawab setiap entitas maritim secara tegas guna menjamin akuntabilitas operasional, sebagaimana diilustrasikan pada matriks peran Tabel 2.

Aktor / Komponen	Peran Utama	Model Asumsi Ancaman	Mekanisme Kontrol / Batasan
Agen Kapal	Membentuk komitmen dokumen, menyusun nullifier, dan mengirimkan transaksi pra-verifikasi.	Partially trusted; potensi kesalahan input data atau transmisi transaksi berulang.	Pendaftaran otorisasi peran (RBAC), validasi format parameter, batas kedaluwarsa, dan nullifier anti-replay.
Aplikasi Klien / DApp	Menyusun muatan transaksi (payload) dan mengelola penandatanganan digital.	Presentation layer rentan terhadap gangguan integritas antarmuka.	Penandatanganan kriptografis berbasis kunci privat agen dan skema encoding deterministik.
Smart Contract	Menjalankan evaluasi predikat otorisasi, integritas komitmen, dan pencatatan riwayat event.	Dipercaya penuh sebatas logika bytecode yang telah dideploy pada EVM.	Pengujian fungsional menyeluruh, validasi guard condition, dan manajemen versi bytecode.
Ledger EVM	Menyimpan status pembaruan state clearance dan memancarkan log audit trail.	Keterbukaan metadata transaksi; kebutuhan konsistensi konsensus.	Penerapan jaringan terkelola (permissioned ledger) pada implementasi operasional.
Otoritas Pelabuhan	Mengelola registri agen terdaftar, mencabut hak akses, dan memantau transparansi log.	Penyalahgunaan hak administratif atau kompromi kunci privat tunggal.	Penggunaan arsitektur multi-signature (multisig) dan audit trail berbasis log event on-chain.
PSCO	Melaksanakan inspeksi fisik di atas kapal dan menetapkan keputusan hukum kepatuhan.	Keputusan keselamatan maritim menuntut pertimbangan profesional manusia.	Aktivitas penilaian profesional sepenuhnya berada di luar eksekusi otomatis smart contract.

Tabel 2. Matriks pemisahan peran aktor, asumsi ancaman, dan batas kewenangan sistem.

2.3 Spesifikasi Implementasi Tiga Arsitektur

Ketiga arsitektur smart contract dikembangkan secara modular dalam bahasa Solidity untuk merepresentasikan evolusi fitur keamanan:

1. Arsitektur A (ConventionalPSC): Berperan sebagai baseline efisiensi biaya. Smart contract mengevaluasi pemetaan authorizedAgents dan validitas hash identitas kapal (imoHash), kemudian mencatat status clearance dasar beserta batas masa berlaku (validUntil).

2. Arsitektur B (HashCommitmentPSC): Memperluas kontrak konvensional dengan menambahkan parameter documentCommitment ke dalam state dan memancarkan log event verifikasi komitmen dokumen. Hal ini memungkinkan pihak berwenang mengonfirmasi keaslian berkas digital kapal tanpa membebani kapasitas blok EVM.

3. Arsitektur C (MockProofPSC): Menambahkan lapisan predikat verifikasi bukti (proofCommitment), parameter domainSeparator yang terikat pada alamat kontrak dan chain ID, serta struktur pemetaan usedNullifiers. Logika kontrak secara ketat menolak transaksi dengan IMO hash nol, komitmen kosong, masa berlaku kedaluwarsa, kegagalan pencocokan predikat, atau penggunaan ulang nullifier yang telah tercatat.

Kategori Ancaman	Mekanisme Kontrol	Hasil Evaluasi Fungsional	Mitigasi Lanjutan
Pengajuan Tanpa Otorisasi	Pemfilteran pengirim via mapping authorizedAgents	Transaksi dari akun tidak sah dibatalkan (revert NotAuthorizedAgent)	Rotasi kunci publik berkala dan sertifikasi agen digital.
Serangan Pemutaran Ulang (Replay)	Pencatatan konsumsi pada mapping usedNullifiers	Pengajuan kedua dengan nullifier identik dibatalkan (revert NullifierAlreadyUsed)	Pemberian format derivasi berbasis timestamp dan sesi pelayaran.
Pemalsuan Parameter Predikat	Validasi kecocokan predikat hash deterministik	Payload predikat tidak valid dibatalkan (revert MockProofVerificationFailed)	Penerapan skema kriptografis zk-SNARK verifier pada fase produksi.
Manipulasi Berkas Administratif	Penyimpanan cryptographic document commitment	Integritas hash terverifikasi terhadap representasi data off-chain	Standarisasi skema kanonikalisasi berkas XML/JSON statutoria.
Eksposur Metadata Transaksi	Penyimpanan dokumen utuh dilakukan off-chain	Tidak ada konten dokumen statutoria terbuka yang disimpan di ledger	Enkripsi metadata transaksi dan integrasi relayer privat.
Penyalahgunaan Akun Otoritas	Pemisahan peran kepemilikan kontrak	Fungsi tata kelola agen dibatasi hanya untuk alamat pemilik otoritas	Integrasi modul smart contract multi-signature (Gnosis Safe).

Tabel 3. Model ancaman keamanan, mekanisme penanganan, dan mitigasi lanjutan.

2.4 Konfigurasi Lingkungan Eksperimen

Pengujian eksperimental dijalankan pada lingkungan komputasi terkontrol untuk menjamin stabilitas observasi. Parameter konfigurasi lingkungan pengujian dirangkum secara sistematis pada Tabel 4.

Parameter Komponen	Konfigurasi Eksperimen
Ethereum Virtual Machine (EVM)	Hardhat local network runtime; Chain ID: 31337; mode auto-mining aktif; block mining interval 0 ms; konfigurasi gasPrice otomatis.
Framework Pengembangan	Hardhat framework v2.22.x dengan integrasi @nomicfoundation/hardhat-toolbox.
Lingkungan Runtime Klien	Node.js JavaScript runtime engine LTS (v18.x / v20.x) dengan Ethers.js transaction management.
Kompiler Smart Contract	Solidity Compiler v0.8.24; target EVM kompatibel Cancun / Shanghai.
Optimasi Bytecode	Solidity optimizer diaktifkan dengan konfigurasi 200 runs.
Protokol Transaksi Benchmark	25 transaksi warm-up (diabaikan dari analisis) diikuti 1.000 transaksi terukur per arsitektur (total N = 3.000 transaksi evaluasi).
Spesifikasi Host Lingkungan Uji	Server Linux virtual x86_64 dengan multi-core processing environment terkontrol.

Tabel 4. Konfigurasi lingkungan eksperimen dan parameter benchmark EVM.

2.5 Prosedur Pengukuran dan Analisis Data

Prosedur benchmark dieksekusi melalui skrip otomatisasi yang menjalankan deployment kontrak secara berurutan: ConventionalPSC, HashCommitmentPSC, dan MockProofPSC. Pada setiap arsitektur, satu akun agen didaftarkan secara resmi ke dalam registri, dilanjutkan dengan fase pemanasan (warm-up) sebanyak 25 transaksi untuk mengeliminasi variabilitas awal caching pustaka dan inisialisasi node. Selanjutnya, sebanyak 1.000 transaksi terukur dieksekusi secara sekuensial. Waktu eksekusi diukur menggunakan pewaktu presisi tinggi (process.hrtime) yang mencatat selisih durasi tepat sebelum transmisi panggilan hingga penerimaan transaction receipt yang valid.

Seluruh data transaksi dicatat ke dalam berkas dataset dengan atribut: arsitektur kontrak, indeks iterasi, transaction hash, nomor blok, gas used, status receipt, dan latensi (milidetik). Analisis statistik konsumsi gas diringkas melalui parameter tendensi sentral (mean dan median), dispersi (standar deviasi), serta persentil ke-5

(P5) dan ke-95 (P95). Sementara itu, distribusi latensi lokal dievaluasi menggunakan median, interval kepercayaan bootstrap 95% (5.000 iterasi resample), persentil ke-95, serta uji statistik non-parametrik Kruskal-Wallis dan Mann-Whitney U. Overhead komputasi relatif dihitung melalui formulasi: Overhead (%) = ((Nilai Model / Nilai Baseline) - 1) × 100%.

3. HASIL DAN PEMBAHASAN

3.1 Analisis Biaya Deployment dan Execution Gas

Konsumsi gas merupakan metrik utama dalam mengevaluasi efisiensi biaya operasional komputasi pada arsitektur EVM. Tabel 5 menyajikan ringkasan statistik konsumsi gas deployment dan eksekusi untuk ketiga arsitektur berdasarkan 1.000 transaksi per perlakuan.

Arsitektur Smart Contract	Deployment Gas	Mean Gas	Median Gas	Standar Deviasi	Persentil 5 (P5)	Persentil 95 (P95)
Konvensional (Baseline)	289.063	48.427,24	48.429	4,39	48.417	48.429
Komitmen Hash	301.404	71.503,80	71.507	6,07	71.495	71.507
Mock Verification (ZKP Interface)	363.917	117.679,68	117.686	8,83	117.662	117.686

Tabel 5. Statistik konsumsi execution gas dan deployment gas pada 1.000 transaksi per arsitektur.

Hasil pengukuran empiris memperlihatkan bahwa arsitektur konvensional memerlukan median sebesar 48.429 gas. Penerapan komitmen hash meningkatkan median biaya eksekusi menjadi 71.507 gas, sedangkan arsitektur mock verification mencapai median 117.686 gas. Standar deviasi yang sangat rendah (berkisar antara 4,39 hingga 8,83 gas) menegaskan bahwa transisi state EVM berjalan secara sangat deterministik. Sebagaimana dirinci pada Tabel 6, overhead komputasi dari komitmen hash mencapai 47,65% di atas baseline, yang diatribusikan pada operasi penulisan storage variabel bytes32 dokumen dan emisi event log. Sementara itu, arsitektur mock verification menimbulkan peningkatan overhead sebesar 143,00% dibandingkan baseline konvensional (atau 64,58% di atas komitmen hash), yang bersumber dari pembacaan dan penyimpanan nullifier (SSTORE/SLOAD), evaluasi predikat keccak256, serta penegakan batas waktu masa berlaku.

Perbandingan Arsitektural	Overhead Rata-rata Gas (%)	Interpretasi Komputasi dan Beban State
Komitmen Hash vs. Konvensional	+47,65%	Penambahan penyimpanan parameter documentCommitment pada storage slot dan emisi log event audit.
Mock Verification vs. Konvensional	+143,00%	Akumulasi penulisan mapping nullifier baru, evaluasi fungsi hashing internal, validasi expiry, dan pembaruan struct clearance.
Mock Verification vs. Komitmen Hash	+64,58%	Beban komputasi spesifik untuk mekanisme anti-replay dan verifikasi predikat integritas komitmen bukti.

Tabel 6. Analisis perbandingan overhead execution gas relatif antarsistem.

3.2 Evaluasi Wall-Clock Latency Lokal

Pengukuran waktu eksekusi dilakukan pada node Hardhat lokal dengan mode auto-mining guna memetakan durasi pemrosesan pada tingkat komputasi instan tanpa distorsi antrean mempool jaringan publik. Karakteristik distribusi latensi disajikan pada Tabel 7.

Arsitektur Smart Contract	Median Latensi (ms)	Bootstrap 95% CI (ms)	Persentil 95 (ms)	Maksimum (ms)	Overhead Relatif
Konvensional	2,211	2,184 - 2,241	3,119	13,008	Baseline (Referensi)
Komitmen Hash	2,232	2,215 - 2,252	2,965	4,984	+0,95% vs. Baseline

Arsitektur Smart Contract	Median Latensi (ms)	Bootstrap 95% CI (ms)	Persentil 95 (ms)	Maksimum (ms)	Overhead Relatif
Mock Verification	2,515	2,494 - 2,539	3,449	6,100	+13,75% vs. Baseline

Tabel 7. Karakteristik distribusi wall-clock latency lokal pada 1.000 transaksi per arsitektur.

Data menunjukkan median latensi lokal untuk kontrak konvensional tercatat sebesar 2,211 ms, komitmen hash sebesar 2,232 ms, dan mock verification sebesar 2,515 ms. Pengujian bootstrap dengan 5.000 resample menghasilkan rentang estimasi interval kepercayaan 95% yang sangat rapat, mengonfirmasi stabilitas eksekusi pada lingkungan lokal. Meskipun mock verification meningkatkan median latensi sebesar 13,75% terhadap baseline konvensional, nilai absolut durasi pemrosesan tetap berada pada skala sub-milidetik tambahan (+0,304 ms), yang menunjukkan bahwa penambahan logika kontrol alur dan pencatatan nullifier tidak memberikan hambatan latensi yang memberatkan pada tingkat mesin klien lokal.

3.3 Pengujian Fungsional dan Validasi Keamanan Alur

Evaluasi fungsional menyeluruh dirancang untuk menguji respons smart contract terhadap berbagai kondisi transaksi, mencakup jalur eksekusi valid (happy path) serta skenario penolakan kondisi batas (negative scenarios). Tabel 8 memetakan hasil pengujian fungsional pada arsitektur MockProofPSC.

Skenario Pengujian	Ekspektasi Perilaku Kontrak	Mekanisme Penanganan / Kode Error	Status Evaluasi
Pengirim Tidak Terotorisasi	Transaksi dibatalkan sebelum mutasi state	Custom Error NotAuthorizedAgent()	Sukses (Revert Terverifikasi)
Proof Commitment Tidak Cocok	Pemeriksaan predikat gagal dan transaksi dibatalkan	Custom Error MockProofVerificationFailed ()	Sukses (Revert Terverifikasi)
Identitas IMO Kosong (Zero Hash)	Validasi guard condition menolak identifier kosong	Custom Error InvalidImoHash()	Sukses (Revert Terverifikasi)
Pengajuan Parameter Valid	Verifikasi predikat berhasil, pemutakhiran state clearance	Emisi Event ClearanceIssued(imoHash, validUntil) & isCleared = true	Sukses (State Mutated)
Penggunaan Ulang Nullifier (Replay)	Deteksi pemakaian nullifier sebelumnya dan penolakan instan	Custom Error NullifierAlreadyUsed() pada pengajuan kedua	Sukses (Revert Terverifikasi)
Pengajuan Pasca Pencabutan Agen	Penolakan akses setelah pemanggilan fungsi revokeAgent	Custom Error NotAuthorizedAgent()	Sukses (Revert Terverifikasi)

Tabel 8. Hasil pemetaan skenario pengujian fungsional dan integritas alur logika smart contract.

3.4 Sintesis Jawaban atas Pertanyaan Penelitian

Berdasarkan analisis hasil empiris dan fungsional, temuan penelitian dapat disintesis untuk menjawab pertanyaan penelitian sebagai berikut:

1. Jawaban RQ1: Arsitektur smart contract berhasil menegakkan pemisahan wewenang yang tegas antara entitas maritim. Agen kapal hanya memiliki hak mengajukan muatan komitmen, smart contract memverifikasi predikat secara otonom, buku besar EVM menjamin keabadian jejak audit, otoritas pelabuhan memegang kendali administratif registri agen, dan kewenangan evaluasi kelaiklautan fisik kapal tetap sepenuhnya berada di tangan PSCO.

2. Jawaban RQ2: Penerapan komitmen hash dokumen menghasilkan overhead gas sebesar 47,65% terhadap baseline konvensional, sedangkan penambahan skema antarmuka mock verification beserta registry nullifier menimbulkan overhead gas sebesar 143,00% terhadap baseline. Pada sisi wall-clock latency lokal, overhead berkisar antara 0,95% hingga 13,75%, yang membuktikan bahwa beban operasional tambahan masih sangat terjangkau pada ekosistem EVM.

3. Jawaban RQ3: Evaluasi fungsional membuktikan bahwa seluruh guard condition, pembatasan otorisasi pengirim, penolakan parameter nol, serta mekanisme proteksi pemutaran ulang (anti-replay) melalui struktur nullifier bekerja secara sempurna sesuai spesifikasi perancangan.

3.5 Pembahasan

3.5.1 Analisis Trade-off Komputasi dan Karakteristik State EVM

Peningkatan biaya gas sebesar 69.257 gas antara arsitektur konvensional dan mock verification mencerminkan trade-off inheren pada mesin EVM. Operasi penulisan storage permanen (SSTORE) untuk pencatatan status pemakaian nullifier (usedNullifiers[nullifier] = true) dan pembuatan entri clearance baru merupakan kontributor utama pembengkakan gas, jauh melampaui biaya komputasi primitif hashing keccak256 dalam memori. Hasil ini memberikan gambaran kuantitatif yang realistis bagi perancang sistem maritim: sebelum mempertimbangkan verifier kriptografis ZKP yang sesungguhnya, overhead antarmuka dan manajemen state anti-replay telah menyerap alokasi gas tertentu yang harus diperhitungkan dalam rancangan batas blok jaringan pelabuhan.

3.5.2 Manajemen Privasi, Auditabilitas, dan Batasan Komitmen Dokumen

Penyimpanan komitmen hash dokumen secara efektif melindungi isi dokumen fisik dari keterbukaan umum pada ledger. Namun, komitmen hash deterministik tanpa entropi tambahan rentan terhadap serangan kamus (dictionary attack) apabila format sertifikat memiliki ruang nilai yang sempit dan dapat diprediksi. Selain itu, metadata transaksi publik—seperti alamat pengirim agen, waktu transmisi, dan frekuensi pembaruan—tetap dapat dianalisis oleh pihak ketiga. Implementasi ZKP tingkat produksi di masa depan, seperti sirkuit berbasis Groth16 atau PLONK, akan memungkinkan agen membuktikan kepatuhan dokumen statutoria (misalnya: masa berlaku sertifikat > tanggal kedatangan, dan penerbit terdaftar pada database IMO) tanpa mengekspos identifier sensitif secara langsung ke dalam ledger.

3.5.3 Implikasi Hukum, Tata Kelola Pelabuhan, dan Prinsip Human-in-the-Loop

Dalam kerangka regulasi pelayaran internasional, keabsahan hukum sertifikat kapal diatur oleh konvensi IMO dan hukum maritim nasional. Status clearance digital yang dihasilkan oleh smart contract merupakan sertifikasi administratif tingkat sistem (pre-clearance verifikasi berkas). Sistem ini tidak menggantikan dan tidak dapat menganulir hak diskresi PSCO untuk melakukan intervensi apabila ditemukan indikasi kondisi kapal yang tidak memenuhi standar keselamatan (clear grounds). Prinsip human-in-the-loop harus dipertahankan secara eksplisit dalam SOP pelabuhan: smart contract menyediakan ringkasan integritas berkas yang terverifikasi secara cepat sebelum kapal sandar, sementara PSCO memegang otoritas validasi lapangan.

3.5.4 Interoperabilitas Sistem Maritim: MSW, PCS, dan Smart Port

Smart contract pra-verifikasi ini paling optimal diposisikan sebagai lapisan integritas data (trust layer) yang terintegrasi dengan platform maritim yang telah beroperasi, seperti Maritime Single Window (MSW) mandat IMO dan Port Community System (PCS). Melalui antarmuka API terstandarisasi, data dokumen yang diunggah oleh agen kapal pada portal MSW dapat diverifikasi komitmen hash-nya secara otomatis oleh smart contract, sehingga memangkas redundansi pemeriksaan dokumen manual antarinstansi pemerintah di pelabuhan.

3.5.5 Peta Jalan Kesiapan Teknologi Menuju Implementasi Operasional

Berdasarkan kerangka Technology Readiness Level (TRL), prototipe smart contract yang dievaluasi pada penelitian ini berada pada tingkat TRL 3 (pembuktian konsep eksperimental pada lingkungan laboratorium). Tabel 9 memetakan jalur transisi komprehensif menuju implementasi penuh pada ekosistem operasional pelabuhan.

Tingkat Kesiapan (TRL)	Tahapan Transisi	Target Validasi dan Kebutuhan Teknis
TRL 3 (Status Saat Ini)	Proof-of-Concept Eksperimental	Desain arsitektur kontrol alur, benchmarking biaya gas lokal, dan verifikasi fungsional anti-replay.
TRL 4	Pengembangan Sirkuit Kriptografis Laboratorium	Konstruksi sirkuit ZKP aktual (Circom/Noir), kompilasi verifier on-chain (Groth16/PLONK), dan audit formal keamanan kode.
TRL 5	Uji Coba Lingkungan Jaringan Relevan	Penerapan pada testnet konsorsium permissioned (Hyperledger Besu / Quorum), integrasi adapter API MSW/PCS, dan uji beban konkurensi tinggi.
TRL 6	Demonstrasi Operasional Terbatas	Implementasi pilot project mode bayangan (shadow-mode pilot) bersama konsorsium otoritas pelabuhan, PSCO, dan asosiasi agen pelayaran.

Tingkat Kesiapan (TRL)	Tahapan Transisi	Target Validasi dan Kebutuhan Teknis
TRL 7 - 8	Integrasi Penuh dan Adopsi Kebijakan	Penyusunan regulasi hukum pengakuan bukti on-chain, protokol respons insiden, SOP terharmonisasi, dan peluncuran skala komersial.

Tabel 9. Peta jalan transisi kesiapan teknologi (TRL) dari prototipe eksperimental menuju operasional.

3.6 Keterbatasan Penelitian

Penelitian ini memiliki sejumlah batasan yang perlu diperhatikan dalam menginterpretasikan hasil yang diperoleh:

1. Lingkungan Eksekusi: Pengukuran performa dilakukan pada lingkungan Hardhat EVM lokal dengan auto-mining aktif. Oleh karena itu, latensi yang tercatat adalah wall-clock latency komputasi lokal, bukan latensi jaringan desentralisasi terdistribusi, waktu propagasi antrean mempool publik, proses konsensus validator, atau waktu finalitas blok.

2. Karakteristik Beban Uji: Transaksi benchmark dieksekusi secara sekuensial pada satu proses komputasi. Meskipun variabilitas gas terbukti deterministik, dinamika persaingan gas pada jaringan multi-user dengan lalu lintas tinggi memerlukan pengujian konkurensi lebih lanjut pada lingkungan testnet publik atau privat terdistribusi.

3. Batasan Domain Operasional: Penelitian ini berfokus secara ketat pada evaluasi komputasi smart contract untuk verifikasi administratif dan jejak audit digital. Penelitian ini tidak mengukur durasi fisik inspeksi kapal di pelabuhan atau penurunan dwelling time logistik secara langsung, karena parameter tersebut melibatkan variabel operasional lapangan yang berada di luar lingkup perangkat lunak on-chain.

4. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang, mengimplementasikan, dan mengevaluasi secara empiris tiga arsitektur smart contract untuk pra-verifikasi administratif Port State Control pada ekosistem EVM. Hasil benchmark 1.000 transaksi per perlakuan mengonfirmasi bahwa arsitektur konvensional mencatatkan median 48.429 gas, arsitektur komitmen hash sebesar 71.507 gas (+47,65%), dan arsitektur mock verification sebesar 117.686 gas (+143,00%), dengan median wall-clock latency lokal yang konsisten pada kisaran 2,211 ms hingga 2,515 ms. Evaluasi fungsional membuktikan bahwa pemisahan hak akses berbasis peran, validasi predikat integritas komitmen dokumen, serta pencegahan serangan pemutaran ulang (replay attack) menggunakan struktur nullifier berjalan secara kokoh sesuai rancangan. Temuan ini menetapkan landasan empiris yang solid mengenai biaya transisi state dan alur kontrol komputasi smart contract, membuka peluang bagi pengembangan sirkuit Zero-Knowledge Proof aktual, serta mendukung integrasi sistem pelabuhan cerdas masa depan.

PERNYATAAN ETIKA DAN KONFLIK KEPENTINGAN

Penelitian ini tidak melibatkan subjek manusia, hewan coba, ataupun penggunaan data operasional pelayaran yang bersifat privat. Seluruh pengujian dilakukan pada lingkungan perangkat lunak lokal menggunakan data sintetis deterministik. Para penulis menyatakan tidak memiliki konflik kepentingan finansial maupun non-finansial yang dapat memengaruhi objektivitas penelitian ini. Penelitian ini dilaksanakan secara independen tanpa pendanaan eksternal yang mengikat.

KETERSEDIAAN DATA DAN KODE

Seluruh kode sumber smart contract (ConventionalPSC.sol, HashCommitmentPSC.sol, MockProofPSC.sol), skrip pengujian fungsional, skrip benchmarking otomatis, serta skrip analisis data telah didokumentasikan secara terstruktur dan tersedia secara terbuka pada repositori riset untuk mendukung transparansi ilmiah dan keterulangan pengujian (reproducibility).

DAFTAR PUSTAKA

[1] International Maritime Organization, Procedures for Port State Control, 2025, Resolution A.1206(34). London: IMO Publishing, 2025.

[2] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292–2303, 2016, doi: 10.1109/ACCESS.2016.2566339.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Yellow Paper, 2014. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[4] M. Jović, E. Tijan, D. Žgaljić, and S. Aksentijević, "Improving maritime transport sustainability using blockchain-based information exchange," Sustainability, vol. 12, no. 21, 2020.

[5] S. Pu and J. S. L. Lam, "Blockchain adoptions in the maritime industry: A conceptual framework," Maritime Policy & Management, vol. 48, no. 6, pp. 777–794, 2021.

[6] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," SIAM Journal on Computing, vol. 18, no. 1, pp. 186–208, 1989.

[7] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in 2016 IEEE Symposium on Security and Privacy, 2016.

[8] X. Sun et al., "A survey on zero-knowledge proof in blockchain," IEEE Network, vol. 35, no. 4, pp. 198–205, 2021.

[9] E. Ben-Sasson et al., "Zerocash: Decentralized anonymous payments from Bitcoin," in 2014 IEEE Symposium on Security and Privacy, 2014.

[10] P.-W. Chi, Y.-H. Lu, and A. Guan, "A privacy-preserving zero-knowledge proof for blockchain," IEEE Access, vol. 11, pp. 85108–85117, 2023.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[12] Nomic Foundation, "Hardhat documentation: Ethereum development environment for professionals," 2026. [Online]. Available: https://hardhat.org/docs

[13] M. Pacheco et al., "What makes Ethereum blockchain transactions be processed fast or slow? An empirical study," Empirical Software Engineering, 2023.

[14] Solidity Team, "Solidity documentation (Version 0.8.24)," 2026. [Online]. Available: https://docs.soliditylang.org/

Lampiran C. Dokumentasi Paket Rekonstruksi Artefak Reproduksibilitas

Lampiran ini menyelaraskan naskah dengan hardhat.config.js, package.json, README.md, tiga source contract, benchmark.js, analyze.py, dan MockProofPSC.functional.test.js. Source telah diperiksa secara statis; raw benchmark, laporan analisis, log kompilasi, dan log eksekusi test belum tersedia. Paket mereproduksi prosedur, bukan menjamin reproduksi angka hasil naskah.

C.1 Struktur Direktori

```
psc-smart-contracts/
├── contracts/
│   ├── ConventionalPSC.sol      # Arsitektur A: otorisasi berbasis peran
│   ├── HashCommitmentPSC.sol    # Arsitektur B: komitmen hash dokumen
│   └── MockProofPSC.sol         # Arsitektur C: proof commitment dan nullifier
├── scripts/
│   ├── benchmark.js             # Warm-up dan transaksi benchmark
│   └── analyze.py               # Analisis deskriptif dan inferensial eksploratif
├── test/
│   └── MockProofPSC.functional.test.js  # Enam skenario uji fungsional
├── hardhat.config.js            # Solidity 0.8.24; optimizer 200 runs
├── package.json
└── README.md
```

C.2 Kesesuaian Paket dengan Metode Penelitian

Item dalam naskah	Implementasi dalam paket
Solidity 0.8.24; optimizer 200 runs	hardhat.config.js; dikonfirmasi tersedia
25 warm-up dan 1.000 transaksi terukur per arsitektur	scripts/benchmark.js; source diperiksa, output run belum tersedia
Kolom raw benchmark	Tujuh kolom CSV dikonfirmasi pada scripts/benchmark.js; CSV belum tersedia
proofCommitment = keccak256(documentCommitment || domainSeparator)	MockProofPSC._mockVerify; source diperiksa
Nullifier satu kali untuk pencegahan replay	MockProofPSC.usedNullifiers; source diperiksa
Enam uji fungsional pada Tabel 8	MockProofPSC.functional.test.js; enam definisi test diperiksa, output run belum tersedia
Overhead gas relatif	scripts/analyze.py: gas_overhead(); source diperiksa
Kruskal-Wallis, Mann-Whitney U, rank-biserial, bootstrap CI 95%	scripts/analyze.py; source diperiksa, raw data/output belum tersedia

Metadata package.json: nama paket psc-smart-contract-zkp-inspired-benchmark; versi 1.0.0; lisensi MIT; private true; scripts compile = "hardhat compile", test = "hardhat test", benchmark = "hardhat run scripts/benchmark.js --network hardhat", dan analyze = "python3 scripts/analyze.py". Rentang dependensi pengembangan adalah Hardhat mulai 2.22.0 hingga sebelum 3.0.0 dan @nomicfoundation/hardhat-toolbox mulai 5.0.0 hingga sebelum 6.0.0.

C.3 Cara Menjalankan Paket

README menetapkan Node.js versi 18 atau lebih baru dan Python 3 dengan pandas, numpy, serta scipy. package.json menyediakan perintah compile, test, benchmark, dan analyze. Karena package-lock.json belum tersedia, versi aktual Hardhat, toolbox, Ethers, dan dependensi transitif tidak dapat dipastikan hanya dari rentang package.json; seluruh versi terpasang, sistem operasi, dan spesifikasi host harus direkam bersama hasil benchmark.

```
# Instal dependensi dan kompilasi
npm install
npx hardhat compile

# Jalankan enam uji fungsional
npx hardhat test

# Jalankan 25 warm-up dan 1.000 transaksi terukur x 3 arsitektur
npx hardhat run scripts/benchmark.js --network hardhat
```

```
# Instal dependensi analisis dalam virtual environment, lalu analisis CSV
python3 -m venv .venv
source .venv/bin/activate
python3 -m pip install pandas numpy scipy
python3 scripts/analyze.py benchmark_raw.csv
```

Review analyze.py mengonfirmasi bahwa skrip memvalidasi tujuh kolom CSV, menyaring receipt_status = 1, menghitung statistik gas dan latency, menjalankan bootstrap median 5.000 resample dengan seed 42, Kruskal-Wallis, Mann-Whitney U, serta rank-biserial, lalu menulis benchmark_analysis_report.json. Karena CSV belum tersedia, hasil numerik belum direplikasi dari skrip yang diunggah.

C.4 Status Artefak dan Tindakan Sebelum Submission

1. Dataset benchmark. Source benchmark.js tersedia dan lolos pemeriksaan sintaks, tetapi benchmark_raw.csv belum disertakan. Dataset harus berasal dari eksekusi nyata dan tidak boleh direkonstruksi dari angka tabel naskah.
2. **Hash integritas.** Hash SHA-256 dataset dan kode harus dihitung ulang setelah benchmark dan finalisasi source code. Nilainya dapat berbeda dari naskah terdahulu akibat perbedaan versi atau implementasi.
3. **Repositori dan DOI.** Folder proyek, dataset, laporan analisis, keluaran pengujian, README, lisensi, dan manifest versi harus dipublikasikan pada repositori Git. Buat rilis bertag, arsipkan melalui Zenodo, OSF, figshare, atau layanan setara, lalu cantumkan DOI dan URL rilis pada pernyataan ketersediaan data.
4. **Pengujian keamanan.** Paket hanya menyediakan enam uji fungsional. Coverage, fuzzing Echidna/Foundry, Slither, expiry-boundary, canonicalization/collision vectors, authority rotation/compromise, dan concurrent submissions harus dijalankan serta dilaporkan sebelum klaim keamanan diperluas.

Perintah pemeriksaan integritas setelah seluruh artefak tersedia:

```
sha256sum benchmark_raw.csv
sha256sum contracts/*.sol
```

C.5 Batas Interpretasi

- MockProofPSC bukan implementasi ZKP. Tidak terdapat circuit, private witness, proving key, verification key, atau verifier kriptografis on-chain; _mockVerify hanya menjalankan predicate hash deterministik.
- Wall-clock latency adalah waktu client sampai receipt pada Hardhat auto-mining lokal, bukan latency jaringan, konsensus, finality, inspeksi PSC, clearance kapal, atau dwelling time pelabuhan.
- Nilai gas dan latency bergantung pada source code, versi compiler/Hardhat, state, urutan transaksi, proses host, dan konfigurasi eksekusi. Prosedur identik tidak menjamin nilai absolut identik.
- Sebanyak 1.000 transaksi per arsitektur merupakan repeated benchmark measurements. Tanpa replikasi batch independen dan randomisasi urutan, inferensi statistik tetap rentan terhadap autokorelasi, efek urutan, noise host, dan pseudoreplikasi.
- Enam definisi uji fungsional konsisten dengan Tabel 8, tetapi tanpa log eksekusi tidak dapat disebut sebagai enam hasil PASS. Cakupannya bukan audit keamanan menyeluruh.

C.6 Lisensi

Paket rekonstruksi menggunakan Lisensi MIT dan dapat digunakan kembali untuk kepentingan akademik dan reproduksi, dengan tetap mencantumkan atribusi yang sesuai. Lisensi tidak mengubah kewajiban penulis untuk memastikan integritas data, akurasi dokumentasi, serta kepatuhan terhadap kebijakan jurnal dan repositori.