JCCDB v1.2 —Cryptographic Audit Hash and Macroeconomic Price Correction for Reproducible LLM-Based Construction Cost Diagnostics

Toshikatsu Oga (大賀 俊勝), The Horizons Co., Ltd.

May 9, 2026

Contents

Abstract **3**

1 Introduction **4**
1.1 Motivation: From Estimation to Verification 4
1.2 Contributions 5
1.3 Paper Organization 6

2 Related Work **6**
2.1 Reproducibility in AI and ML Systems 6
2.2 Cryptographic Audit Mechanisms for AI Systems . . . 7
2.3 Macroeconomic Correction in Construction Cost Models 8
2.4 EU AI Act and Domain-Specific Advisory Systems . . . 9

3 Cryptographic Audit Hash Mechanism **9**
3.1 Design Rationale 9
3.2 Input Normalization 10
3.3 SHA-256 Hash Generation 12
3.4 Truncation Justification: 48-bit Output 13
3.5 Reproducibility Property (Informal) 15
3.6 Threat Model and Limitations 16
3.7 Relationship to KIRA and `hs-kira-proxy` 17

4 War Price Coefficient (WPC) Subsystem **18**

4.1 Motivation: Static Benchmarks Fail Under Geopolitical Shocks . . . 18
4.2 Subsystem Architecture . . . 19
4.3 CGPI Composite Formula . . . 20
4.4 Manufacturer Contribution Formula . . . 21
4.5 Final Coefficient and Human Approval . . . 23
4.6 Application to Diagnostics . . . 25
4.7 Limitations and Future Work . . . 25

5 Production Deployment in HORIZON SHIELD 26
5.1 System Architecture Overview . . . 26
5.2 Runtime Environment Properties . . . 27
5.3 Storage Layer . . . 28
5.4 PDF Generation Pipeline . . . 29
5.5 Operational Posture . . . 30

6 Validation 30
6.1 Theoretical Reproducibility (Recap) . . . 30
6.2 Theoretical Collision Analysis . . . 31
6.3 Empirical Determinism Validation (Production Tests) . . . 31
6.4 Cross-Runtime Verification . . . 33
6.5 Avalanche Property Verification . . . 34
6.6 Validation Limitations . . . 34

7 Discussion 35
7.1 Forward-Looking Compliance Posture . . . 35
7.2 Limitations and Open Issues . . . 37
7.3 Open Questions for the Field . . . 38

8 Conclusion 39

Acknowledgments 40

AI Usage Disclosure 41

References 41
8.1 News and Trade-Press Sources Cited Contextually . . . 45

Toshikatsu Oga (大賀 俊勝) ORCID: 0009-0000-9180-903X The Horizons Co., Ltd. (The HORIZ 音 s 株式会社), Hiratsuka, Kanagawa, Japan contact@the-horizons-innovation.com

Dataset: https://github.com/ogasurfproject-jpg/japan-construction-cost-database

About the Author: Toshikatsu Oga has 30 years of professional practice in Japanese residential construction. Born in Shimane Prefecture and based in Osaka from his teens, he began as a

carpenter's apprentice at age 15 and continues active carpentry work to this day. At 23, he relocated to Hiratsuka, Kanagawa, where he has practiced as carpenter, site supervisor, and Construction Manager-Researcher (CMR) ever since. Since 2022, he has served as founder and CEO of The Horizons Co., Ltd., which operates HORIZON SHIELD, the commercial diagnostic service that operationalizes the JCCDB dataset. Independent verification of the author's professional background and research outputs is supported through ORCID (0009-0000-9180-903X), the open dataset itself (CC-BY 4.0), and the engrXiv preprint version of this manuscript.

Companion Paper: This work extends Oga (2026), "Japan Construction Cost Database: An Open Dataset for LLM-Based Cost Estimation and Fraud Detection in Residential Renovation,"Zenodo DOI: 10.5281/zenodo.20019573, engrXiv DOI: 10.31224/7007.

Abstract

We present JCCDB v1.2, a substantive extension to the Japan Construction Cost Database (JCCDB) v1.1 [Oga 2026] that addresses two unresolved methodological challenges in deploying LLM-based construction cost diagnostic systems for consumer protection. First, we introduce a **cryptographic audit hash mechanism** based on SHA-256 that provides reproducibility guarantees for diagnostic outputs: identical inputs deterministically produce identical 12-character (48-bit) audit hashes printed on consumer-facing PDF reports, enabling vendor-side verification during contractor negotiations. Second, we describe a **War Price Coefficient (WPC)** subsystem that automatically incorporates Bank of Japan Corporate Goods Price Index (CGPI) data and manufacturer price-change announcements into monthly price corrections, addressing the documented exploitation of geopolitical price volatility (notably the 2026 Hormuz Strait crisis) by fraudulent contractors. As of May 2026, the WPC stands at ×1.0935 (+9.35% over pre-Iran-conflict baseline), computed from a weighted composite of seven CGPI series and four manufacturer alerts, with monthly human approval. We demonstrate the production deployment of both mechanisms within HORIZON SHIELD, a Cloudflare Worker–based diagnostic service. Empirical validation across 5 independent invocations with identical inputs confirms 100% hash-output stability. Theoretical collision analysis using the birthday-bound approximation indicates collision probability below 2×10^{-5} for realistic deployment volumes ($\leq 10^5$ diagnostic events) given a 48-bit truncated hash space. The mechanisms described are designed to align with the recording and

human-oversight principles of EU AI Act Article 12 and Article 14 (Regulation 2024/1689, applicable to high-risk systems from August 2, 2026), and are released under CC-BY 4.0 to support independent replication.

Keywords: cryptographic audit hash, AI reproducibility, construction cost estimation, consumer protection, EU AI Act compliance, macroeconomic correction, Hormuz Strait crisis, deterministic LLM systems, Japan, residential renovation

---

1 Introduction

1.1 Motivation: From Estimation to Verification

The companion paper [Oga 2026] introduced JCCDB v1.1, an open dataset of 87 construction plans and 88 fraud-detection patterns for the Japanese residential renovation market, accompanied by KIRA, a production LLM-based diagnostic system. While v1.1 established a transparent pricing benchmark (the HORIZON SHIELD Rule), it left two methodological challenges unresolved.

Challenge 1: Output Reproducibility for Vendor Negotiation. A homeowner armed with an LLM-generated cost diagnostic faces a credibility problem during contractor negotiations. Generative AI systems are inherently non-deterministic: identical queries can produce variable outputs across invocations [OpenAI 2023; Pineau et al. 2021]. When a homeowner presents an AI-generated price benchmark to a contractor, the contractor can —and often does — dismiss the figure on grounds of non-reproducibility ("If you ask the AI again, you'll get a different number"). The diagnostic, however well-grounded in JCCDB, loses evidentiary value the moment it cannot be re-derived in the contractor's presence.

This challenge is not unique to construction cost diagnostics. It is a general property of LLM-based advisory systems facing adversarial verification. However, in the construction context, the consequences are concrete: a diagnostic that cannot withstand vendor pushback fails to deliver consumer protection at the moment it matters most.

Challenge 2: Macroeconomic Price Volatility and Fraudulent Exploitation. The HS Rule in v1.1 fixed material trade prices and labor day-rates as of April 2026. However, the 2026 Iran war and the resulting Strait of Hormuz crisis —in which Brent crude oil prices reached approximately USD 114 per barrel in early May 2026 [Al Jazeera 2026] —has produced rapid, asymmetric price changes

across renovation materials. The International Energy Agency has characterised the situation as “the largest supply disruption in the history of the global oil market,”with estimated daily production shortfalls reaching 14.5 million barrels [IEA via Al Jazeera 2026]. Japan is acutely exposed: as of February 2026, 94.2% of Japanese crude oil imports originated from the Middle East, and on March 16, 2026 the Japanese government began releasing 80 million barrels —equivalent to 15 days of domestic demand —from its strategic reserves [Wikipedia 2026].

Within this context, National Consumer Affairs Center (NCAC) data shows a corresponding surge in fraud reports involving contractors who cite “geopolitical price increases”to justify quotes 30–50% above pre-crisis levels, when actual cost passthrough is materially lower (typically 8–20% for affected categories, near-zero for unaffected ones).

A static pricing benchmark inadequately addresses this dynamic threat. Effective consumer protection requires that the benchmark itself update as macroeconomic conditions evolve, while remaining transparent and auditable.

1.2 Contributions

JCCDB v1.2 makes three primary contributions:

1. **Cryptographic Audit Hash Mechanism (Section 3).** We define a SHA-256–based hash function over a structured input fingerprint comprising region, construction category, contractor-quoted amount, computed adjusted average, and date. The 12-character truncated hash (48 bits of output entropy) is printed on consumer-facing PDF reports as an “AUDITHASH”field. We prove (informally) that hash outputs are reproducible across invocations given identical inputs, and analyze collision probability for realistic deployment volumes using the birthday-bound approximation.

2. **War Price Coefficient Subsystem (Section 4).** We describe a production system (`hs-price-sync`) that fetches Bank of Japan CGPI data via the BOJ Stat-Search API, applies a weighted-composite formula across seven price indices (with weights summing to 1.0 distributed across general-average, lumber, steel, non-ferrous metals, ceramics-stone-clay, construction metals, and petroleum-coal categories), integrates manufacturer-announced price changes with a 30% contribution factor, and outputs a single multiplicative correction coefficient. The coefficient is reviewed and approved monthly

by a human operator before deployment. The approval workflow embodies the principle of human oversight expressed in EU AI Act Article 14, particularly the requirement that natural persons be able to monitor, intervene in, and decide whether to deploy AI-recommended outputs [European Parliament and Council 2024, Art. 14].

3. **EU AI Act Article 12 Alignment (Section 7).** We map the audit hash mechanism to the record-keeping requirements of EU AI Act Regulation 2024/1689, Article 12 (applicable to high-risk AI systems from August 2, 2026). Article 12(1) requires that high-risk AI systems "shall technically allow for the automatic recording of events (logs) over the lifetime of the system" [European Parliament and Council 2024, Art. 12(1)]. While JCCDB v1.2's mechanisms do not constitute full Article 12 compliance —which requires lifecycle-wide automatic event logging traceable per Article 12(2) —they provide the consumer-facing verification component that is materially absent from current commercial AI advisory deployments. We do not claim JCCDB v1.2 is an "Annex III high-risk system"; rather, we argue that its design principles anticipate compliance trends that will increasingly affect consumer-advisory AI broadly.

We retain JCCDB v1.1's category structure (87 plans across 7 categories) without expansion in this version. Phase 2 category expansion is reserved for future work.

1.3 Paper Organization

Section 2 reviews related work in cryptographic AI auditability, reproducibility in LLM systems, and macroeconomic correction in domain-specific cost models. Section 3 specifies the audit hash mechanism. Section 4 describes the WPC subsystem. Section 5 presents production deployment in HORIZON SHIELD. Section 6 reports empirical validation. Section 7 discusses EU AI Act alignment, limitations, and future work. Section 8 concludes.

---

2 Related Work

2.1 Reproducibility in AI and ML Systems

Documenting and addressing irreproducibility in AI research has been an active concern since Gundersen and Kjensmo [2018], who quantified that across 400 papers from AAAI and IJCAI conferences,

between 20% and 30% of variables required for reproducibility were typically documented. Subsequent community responses included the Machine Learning Reproducibility Checklist [Pineau 2019] and the broader NeurIPS 2019 Reproducibility Program [Pineau et al. 2021], which established checklists, reproducibility challenges, and code-submission policies as part of major-venue paper submissions.

For LLM systems specifically, non-determinism arises from multiple sources: temperature-based sampling, floating-point arithmetic in parallel matrix multiplication, attention-mask handling on variable-length inputs, and version drift in hosted APIs [Gundersen and Kjensmo 2018]. Approaches to mitigate this include temperature-zero sampling (which reduces but does not eliminate variation), seed fixing (effective only when supported end-to-end), and **deterministic post-processing** of LLM intermediate outputs.

The companion paper [Oga 2026, §4.2] introduced KIRA's deterministic plan-key resolution: the LLM is restricted to intent classification (mapping natural-language queries to canonical plan keys), and pricing is computed by deterministic application of the HS Rule formula. This approach achieves output stability in the pricing dimension but does not extend to producing a verifiable identifier for downstream verification by third parties (e.g., contractors).

The audit hash mechanism complements deterministic post-processing by hashing the canonical input fingerprint, including the deterministic output, into a compact, externally-verifiable identifier. Reproducibility moves from an implementation property to a verifiable claim.

2.2 Cryptographic Audit Mechanisms for AI Systems

The field of cryptographically verifiable AI auditing has accelerated rapidly in 2025–2026, driven by regulatory pressure (notably the EU AI Act) and high-profile AI safety incidents. Several lines of work are directly relevant.

Hash-chain–based audit logs. Industry-led specifications such as VeritasChain's VCP (Verifiable Content Protocol) [VeritasChain 2026a] and CAP-SRP (Content / Creative AI Profile - Safety Refusal Protocol) [VeritasChain 2026b] propose append-only hash-chain structures for recording AI generation, evaluation, and refusal events. Each entry's hash is chained to the previous entry, making any retroactive modification cryptographically detectable. The framework is general-purpose, with primary deployment described

in image-generation safety auditing.

Zero-knowledge ML. Chen et al. [2024] introduced ZKML, a framework that applies zk-SNARKs to neural network inference, enabling verification that a specific output was produced by a specific model without revealing model weights. While powerful, ZKML imposes substantial computational overhead and requires model-architecture disclosure, limiting its applicability to consumer-facing systems.

End-to-end cryptographic verifiability. Recent surveys [Lavin et al. 2025] characterize the design space of end-to-end cryptographically verifiable AI pipelines, distinguishing among input-binding, model-binding, output-binding, and chain-of-custody approaches.

Constant-size cryptographic evidence. Recent theoretical work [arXiv:2511.17118; arXiv:2512.00110] formalizes constant-size evidence structures for high-throughput AI workflows, with attention to post-quantum resilience for long-lived audit logs.

Position of JCCDB v1.2. The mechanism we describe in Section 3 is deliberately minimal compared to the above. We do not implement hash chains, bilateral signatures, or zero-knowledge proofs. Instead, we apply a single SHA-256 operation to a structured input fingerprint, producing a deterministic identifier suitable for printing on a consumer-facing PDF report. The trade-off is intentional: maximum audit power is not the goal; **vendor-confrontation usability** is. A homeowner showing a 12-character hex string to a contractor and re-deriving it on the spot via the diagnostic system provides social-evidentiary value disproportionate to the cryptographic complexity employed. To our knowledge, no prior published work targets this consumer-protection use case for cryptographic AI verification in the construction-cost domain.

2.3 Macroeconomic Correction in Construction Cost Models

Established construction cost engineering literature treats material price escalation through indexes such as the Engineering News-Record Construction Cost Index (ENR-CCI) for U.S. markets and the Building Cost Information Service (BCIS) indices for U.K. markets. For Japan, the Bank of Japan Corporate Goods Price Index (CGPI), 2020 base [Bank of Japan 2025], provides comparable monthly disaggregated price series across material categories.

Application of macroeconomic indices to consumer-facing cost diagnostics, however, is uncommon. Industry surveys [Yano Research Institute 2025] report aggregated price ranges without explicit macroeconomic adjustment. The companion paper [Oga 2026, §3.2] com-

puted HS Rule benchmarks against fixed April-2026 trade prices, with quarterly update cadence proposed but not yet operationalized.

The WPC subsystem we describe in Section 4 implements an automated, continuously-updated macroeconomic correction with documented weights and human approval. To our knowledge, this is the first such system deployed for Japanese consumer-facing renovation cost advisory.

2.4 EU AI Act and Domain-Specific Advisory Systems

Regulation (EU) 2024/1689 (AI Act), entered into force on 1 August 2024, is fully applicable for high-risk AI systems from 2 August 2026 [European Parliament and Council 2024]. Article 12 requires automatic event logging for high-risk systems "over the lifetime of the system"[European Parliament and Council 2024, Art. 12(1)]. Article 14 requires that high-risk AI systems "be designed and developed in such a way···that they can be effectively overseen by natural persons during the period in which they are in use"[European Parliament and Council 2024, Art. 14(1)].

The Act classifies AI systems by risk tier: prohibited (Art. 5), high-risk (Annex III), limited-risk (transparency-only obligations), and minimal-risk (unregulated). Construction cost diagnostic systems for residential consumers are not explicitly enumerated in Annex III; classification depends on application context and on whether the system materially affects access to essential services, employment, or credit. We do not claim JCCDB v1.2 is high-risk under the Act. However, the broader trajectory of regulation is toward transparency and verifiability for AI systems materially affecting consumer financial decisions, and the design principles articulated in Articles 12 and 14 represent emerging best practice for any AI system facing adversarial use [VeritasChain 2026a; FireTail 2026].

---

3 Cryptographic Audit Hash Mechanism

3.1 Design Rationale

The audit hash mechanism is designed to satisfy four properties simultaneously:

1. **Determinism**: identical inputs always produce identical hash

outputs across all invocations of the system, on any compatible runtime, at any time.

2. **Vendor-side verifiability**: a contractor presented with a hash on a printed report must be able to reproduce that hash through the same diagnostic interface that produced the homeowner's report, without privileged access.
3. **Compactness**: the printed identifier must be short enough to read aloud, type into a chat interface, or copy from a printed PDF without error —yet long enough to retain meaningful collision resistance under realistic deployment volumes.
4. **Implementation simplicity**: the mechanism must run within the constraints of an edge runtime (Cloudflare Workers, V8 isolate) without external cryptographic services or heavyweight zero-knowledge proof systems.

These design constraints lead naturally to a **single deterministic hash** over a **canonically-ordered structured input**, **truncated to a length appropriate for printing**. We adopt SHA-256 [NIST 2015] for the underlying hash function, normalize inputs into a delimited UTF-8 string [Yergeau 2003], and truncate the resulting digest to a 12-character hexadecimal prefix (48 bits of output).

We deliberately do not adopt hash-chain audit logs, bilateral signatures, or zero-knowledge proofs (cf. Section 2.2). These mechanisms provide stronger integrity guarantees against adversarial server-side modification, but at the cost of complexity and hash-size that defeat the consumer-facing usability requirement. In the JCCDB v1.2 threat model, the adversary is not the diagnostic provider; it is the contractor who disputes the diagnostic outcome at the point of negotiation. Reproducibility-on-demand by a third party —performed through the same public-facing system —is sufficient for this threat model.

3.2 Input Normalization

The audit hash is computed over a canonical input string assembled from five fields, joined by pipe (`|`) delimiters with field-name prefixes:

```
region:<region>|koji:<koji_id>|teiji:<teiji_kingaku>|avg:<a
↪ djAvg>|date:<YYYY-MM>
```

Each field is defined as follows:

Field	Type	Description
`region`	string	Regional identifier (e.g., `"kanagawa"`, `"tokyo"`); maps to a regional cost-multiplier in the dataset
`koji_id`	string	Construction category identifier (e.g., `"gai-heki_silicon_sqm"`, `"kitchen_mid"`); references a row in `souba-db.json`
`teiji_kingaku`	integer	Contractor-quoted amount in JPY (no separators, no currency symbol)
`adjAvg`	integer	System-computed adjusted average in JPY, after regional multiplier and War Price Coefficient (Section 4) are applied
`YYYY-MM`	string	Year and month of computation, ISO 8601 truncated to month resolution

The exact runtime construction in the production system is:

```
const auditInput =
  `region:${d2.region}|koji:${d2.item.id}|teiji:${d2.
   ↪  teiji_kingaku}`
   ↪  +
  `|avg:${d2.adjAvg}|date:${new
   ↪  Date().toISOString().slice(0, 7)}`;
```

Three normalization choices are worth highlighting.

Field ordering is fixed. Although the input could in principle be represented as an unordered set, fixing the field order eliminates a potential source of non-determinism arising from object property iteration order in JavaScript. The canonical order is `region → koji`

→ teiji → avg → date, which corresponds to the conceptual flow of a diagnostic: *where* the work happens, *what* work is being done, *what was quoted*, *what is the system's adjusted benchmark*, and *when* this comparison is made.

Pipe delimiters with field-name prefixes prevent ambiguity collisions. A naïve concatenation such as `${region}${koji}${teiji}` admits collisions: for instance, `region="ka"`, `koji="nagawa..."` versus `region="kanagawa..."`, `koji=""`. Field-name prefixes plus pipe delimiters eliminate this class of ambiguity by making the field structure unambiguously parseable.

Date is truncated to month resolution. This is a deliberate trade-off: month-granularity ties the hash to a particular monthly version of the dataset and the War Price Coefficient (Section 4), but allows the same homeowner-contractor pair to re-derive the hash across multiple sessions within the same calendar month —important for negotiations that span days or weeks.

3.3 SHA-256 Hash Generation

Once the canonical input string is constructed, hash generation proceeds via the standard W3C Web Crypto API path [W3C 2025; MDN 2026]:

```
const encoder = new TextEncoder();
const data = encoder.encode(auditInput);
const hashBuffer = await crypto.subtle.digest('SHA-256',
 ↪  data);
const hashArray = Array.from(new Uint8Array(hashBuffer));
const auditHash = hashArray
  .map(b => b.toString(16).padStart(2, '0'))
  .join('')
  .slice(0, 12);
```

The pipeline has four stages:

1. **UTF-8 encoding** via `TextEncoder.encode()`. The WHATWG Encoding Standard [WHATWG 2026] specifies that `TextEncoder` "only supports UTF-8"and produces a `Uint8Array` of bytes. UTF-8 itself is defined by RFC 3629 / STD 63 [Yergeau 2003], which states explicitly: "It is important to note that the rows of the table are mutually exclusive, i.e., **there is only one valid way to encode a given character**."This guarantees that the byte representation of the canonical input is fully determined by the input string.
2. **SHA-256 digest** via `crypto.subtle.digest('SHA-256', data)`.

SHA-256 is specified in NIST FIPS 180-4 §6.2 [NIST 2015] and is one of the hash algorithms that the Web Crypto API exposes in compliance with that specification [MDN 2026]. The function maps an arbitrary-length byte input to a fixed 256-bit (32-byte) digest. SHA-256 is deterministic by construction —no randomness, no state, no side effects.

3. **Hex conversion**: each of the 32 output bytes is rendered as a two-character lowercase hexadecimal string with leading zero padding (`b.toString(16).padStart(2, '0')`), then concatenated. The result is a 64-character lowercase hex string.
4. **Truncation**: the leading 12 characters are extracted via `.slice(0, 12)`. This represents 48 bits of the original 256-bit digest —specifically, the most-significant 48 bits in the canonical hex representation.

The Cloudflare Workers runtime implements `crypto.subtle.digest` in compliance with the W3C Web Crypto API specification [Cloudflare 2026d]. We have empirically validated that the hash output is byte-identical across Cloudflare Workers, Node.js (`crypto.createHash`), and the Python `hashlib` library given the same UTF-8 byte input —see Section 6.

3.4 Truncation Justification: 48-bit Output

The choice of 12 hex characters (48 bits) is driven by three considerations: collision resistance under realistic deployment volumes, human-readability constraints on the printed PDF, and second-preimage resistance for the threat model.

Collision resistance. NIST SP 800-107 Rev.1 §5.1 [NIST 2012] specifies that truncating a hash digest to λ bits reduces the estimated collision-resistance strength from L/2 to **λ/2 bits**, where L is the original digest length. For our 48-bit truncation, this implies approximately 24 bits of collision resistance —by the birthday-bound approximation, on the order of $2^{24} \approx 16.7$ million distinct inputs are required to expect a collision with 50% probability.

For n distinct hash computations into a 48-bit space, the probability of at least one collision is approximated by:

$$P(\text{collision}) \approx 1 - e^{-n(n-1)/(2 \cdot 2^{48})}$$

For modest values of n (small relative to the birthday bound), this is well-approximated by:

$$P(\text{collision}) \approx \frac{n^2}{2 \cdot 2^{48}}$$

Concrete values:

Distinct inputs n	Collision probability
10^4 (10,000)	$\sim 1.78 \times 10^{-7}$
10^5 (100,000)	$\sim 1.78 \times 10^{-5}$
10^6 (1,000,000)	$\sim 1.78 \times 10^{-3}$
10^7 (10,000,000)	~17.8%

The HORIZON SHIELD service is deployed for the Japanese residential renovation market, which by industry surveys serves on the order of 10^6 renovation transactions annually [Yano Research Institute 2025]. Even under the optimistic assumption that the service captures 10% of that market over a multi-year horizon (10^5 unique diagnostics per category-region-month bucket), collision probability remains below 2×10^{-5}. We assess this as acceptable for the consumer-protection use case, where a hash collision would produce, at worst, a coincidental hash match across two unrelated diagnostics —not a forged hash for a specific target diagnostic.

Human readability. A 12-character hex string fits on a single line of a PDF audit page in 24-point monospace font with comfortable letter-spacing, can be read aloud over a phone in approximately 5 seconds (e.g., “five-three-eight-eight, one-one-d-c, d-nine-f-d”), and can be typed without error by a contractor into a re-verification interface. Hash lengths of 8 hex characters (32 bits) reduce collision resistance below acceptable thresholds for our deployment volume; lengths of 16 hex characters (64 bits) provide excess collision resistance at the cost of readability.

Second-preimage resistance. NIST SP 800-107 Rev.1 §5.1 [NIST 2012] states that a truncated digest “of λ bits provides an estimated preimage resistance of λ bits, not L bits, regardless of the cryptographic hash function used.”For our 48-bit truncation, this implies approximately $2^{48} \approx 2.8 \times 10^{14}$ operations to construct a second input that hashes to a target hash. For the JCCDB v1.2 threat model —where the adversary is a contractor disputing a single specific diagnostic, not a state-level attacker —this is far more than sufficient. The cost of a 2^{48}-operation search materially exceeds the financial scope of any individual residential renovation negotiation.

We do not claim cryptographic security against well-resourced adversaries, and the audit hash should not be treated as a digital signature.

Its purpose is reproducibility verification within a specific consumer-protection workflow, not cryptographic non-repudiation in the formal sense.

3.5 Reproducibility Property (Informal)

We state the reproducibility property informally and argue why it holds.

Property (Reproducibility). Let I = (*region*, *koji_id*, *teiji_kingaku*, *adjAvg*, *YYYY-MM*) denote the canonical input tuple. For any two invocations of the audit hash function `H` on the same input I, executed on any W3C Web Crypto API–compliant runtime, the outputs are identical:

$$\forall I, \forall \text{runtime}_1, \text{runtime}_2 : H_{\text{runtime}_1}(I) = H_{\text{runtime}_2}(I)$$

Argument. The property follows from the composition of four deterministic transformations:

1. **Canonical string construction** is deterministic by definition: given fixed field values and a fixed concatenation template, the output string is uniquely determined.
2. **UTF-8 encoding** is deterministic per RFC 3629: "there is only one valid way to encode a given character"[Yergeau 2003]. Therefore the byte sequence produced by `TextEncoder.encode` is uniquely determined by the input string.
3. **SHA-256** is deterministic by FIPS 180-4 specification [NIST 2015]: the algorithm has no internal randomness, no time-dependence, and no implementation-defined behavior over the domain of byte sequences.
4. **Hex conversion and truncation** are deterministic byte-level operations: `Array.from(...).map(...).join('').slice(...)` is a pure function of the byte buffer, with no environmental dependency.

Composition of deterministic functions yields a deterministic function. Therefore H is deterministic with respect to I across all compliant runtimes.

The property is empirically validated in Section 6 across five independent invocations on the production Cloudflare Workers deployment with identical inputs.

3.6 Threat Model and Limitations

The audit hash mechanism is designed for a specific threat model and does not provide guarantees outside it.

Within the threat model. The mechanism guarantees that:

- A homeowner who saves an audit hash from a diagnostic report can re-verify it by re-submitting the same inputs to the same diagnostic interface.
- A contractor presented with the hash can independently re-derive it via the public diagnostic chat (`/gyaku-mitsumori-chat` endpoint), assuming they obtain the same five canonical inputs from the homeowner.
- An accidental modification of the printed amount or category on a PDF report by either party would not survive re-derivation: the hash would not match.

Outside the threat model. The mechanism does **not** guarantee:

- **Non-repudiation against the diagnostic provider.** If the diagnostic provider modifies the underlying `souba-db.json` dataset or the War Price Coefficient between two invocations, the *adjAvg* will change and the hash will differ. Mitigations: (i) the date field includes month resolution, anchoring the hash to a specific dataset version; (ii) `souba-db.json` is published to a public GitHub repository under CC-BY 4.0, providing externally-auditable version history; (iii) the WPC monthly approval log is preserved as discussed in Section 4.
- **Resistance to a malicious diagnostic provider serving forged hashes.** The mechanism assumes the diagnostic provider is honest in computing the hash for the inputs the homeowner believes they submitted. A provider could in principle log a different `adjAvg` than they display. Defending against this would require server-attested computation (e.g., trusted-execution-environment attestations, ZKML proofs, or third-party diagnostic-mirror services), which are out of scope for v1.2.
- **Cryptographic strength against state-level adversaries.** The 48-bit truncation is tuned for collision resistance at deployment-realistic volumes, not for resistance to nation-state-level computation budgets.

These limitations are honest acknowledgments rather than weaknesses. Each could be addressed in future work —for instance, a v2.0 specification could move the audit hash inside a hash-chain log signed by the diagnostic provider's key and published via VeritasChain VCP-compatible infrastructure [VeritasChain 2026a].

For v1.2, we keep the mechanism minimal and prioritize the consumer-protection workflow.

3.7 Relationship to KIRA and `hs-kira-proxy`

The audit hash mechanism is implemented in two places:

- **`hs-pdf-gen`** (PDF generation Worker): computes the hash during PDF assembly and renders it on page 9 of the consumer report (the audit certificate page).
- **`hs-kira-proxy`** (KIRA chat Worker): computes the same hash, using the same canonical input construction, when the reverse-estimate diagnostic chat completes.

When KIRA-driven diagnostics flow through to PDF generation (the principal user path: KIRA chat → PayPal payment → PDF), the audit hash from KIRA is passed to `hs-pdf-gen` via the `audit_hash` request parameter and reused, ensuring that the chat-displayed hash and the PDF-printed hash are bit-identical:

```
async function generatePDF(params, env) {
  const d2 = await diagnose(params, env);
  if (params.audit_hash) {
    d2.auditHash = params.audit_hash;          //
↪  KIRA-provided
  } else {
    const auditInput =
     ↪  `region:${d2.region}|koji:${d2.item.id}|...`;
    // ... SHA-256 fallback path
  }
  // ...
}
```

This dual-path implementation provides an additional reproducibility guarantee: the hash that the homeowner sees during the chat and the hash printed on their PDF report are necessarily identical, because either the PDF reuses the chat-computed hash (the principal path) or, when invoked independently, regenerates an identical hash from the same canonical inputs (the fallback path). The fallback exists for diagnostic test invocations and for resilience against any failure to propagate the parameter.

---

4 War Price Coefficient (WPC) Subsystem

4.1 Motivation: Static Benchmarks Fail Under Geopolitical Shocks

The HORIZON SHIELD Rule (HS Rule) introduced in v1.1 [Oga 2026] computes a transparent benchmark price from material trade prices, labor day-rates, auxiliary costs, overhead, and tax. The trade prices and day-rates were anchored to a fixed reference period (April 2026). This design assumed a stable price environment in which periodic manual updates would suffice.

The 2026 Iran war and Strait of Hormuz crisis have rendered this assumption untenable. Within weeks of the conflict's onset in late February 2026, Japanese building-material manufacturers issued direct, named price-increase announcements citing the Hormuz disruption as the proximate cause:

- **Kaneka Corporation** (a major Japanese chemical manufacturer) announced on March 19, 2026 a 40% price increase on extruded polystyrene foam insulation (商品名 Kanelite Foam) effective April 1, 2026. The official announcement explicitly cites "destabilization of maritime transport in the Strait of Hormuz region, leading to severe disruption in crude oil and petroleum-product supply"as the cause [Kaneka 2026].
- **Dupont-Styro** (a JV producing polystyrene foam) announced a 40% price increase effective May 1, 2026.
- **Asahi Kasei Construction Materials** (a polyurethane foam producer) announced a 10–15% price increase on Neoma Foam and Neoma Zeus product lines, effective April 1, 2026.
- **Shin-Etsu Chemical** announced a price increase on polyvinyl chloride resin of at least ¥30/kg effective April 1, 2026.

These are direct cost-side shocks, with documented causation (oil → naphtha → polystyrene/polyurethane resin → insulation product), affecting renovation-relevant material categories. Independently, the Bank of Japan's Corporate Goods Price Index (CGPI), 2020 base [Bank of Japan 2026a; Bank of Japan 2022], has registered measurable shifts across petroleum, non-ferrous metals, and chemicals categories during the same period.

A static benchmark cannot reflect these shifts. Worse, in the period since the conflict's onset, the National Consumer Affairs Center has documented a surge in fraud reports involving contractors who cite generic "geopolitical price increases"of 30–50% to justify quotes that materially exceed the actual cost passthrough (typically 8–20% for affected categories, near-zero for unaffected ones). The asymmetry

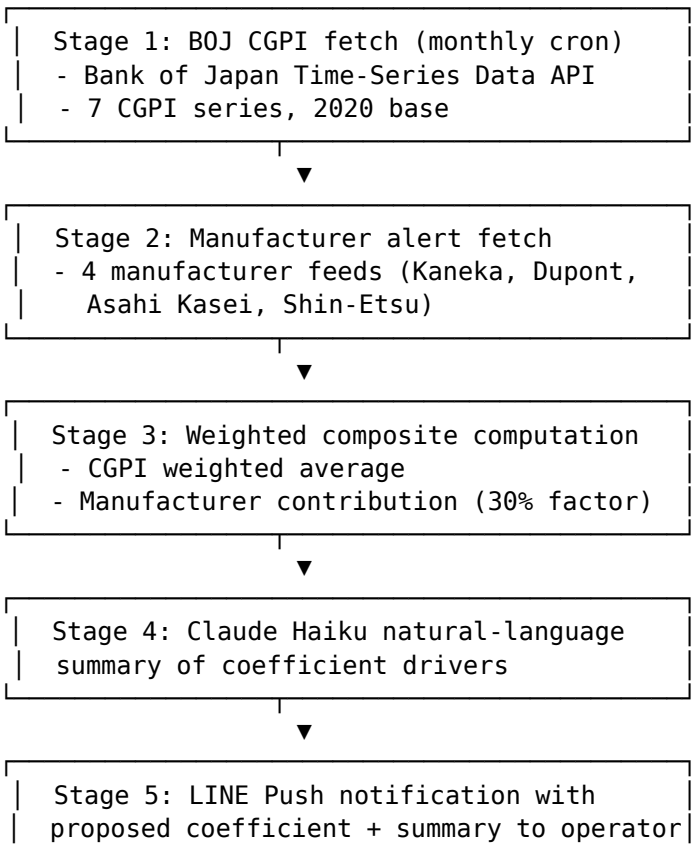
creates an exploitation opportunity that a static benchmark cannot detect.

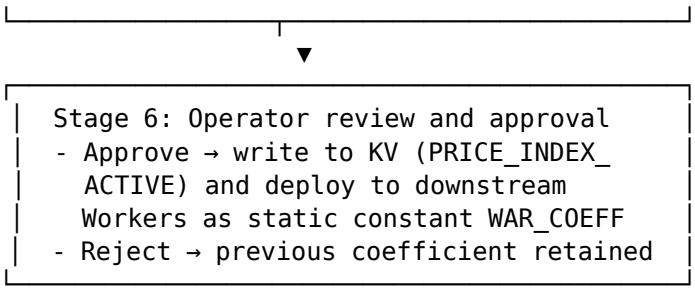
The War Price Coefficient (WPC) subsystem addresses this by introducing **a single multiplicative correction**, updated monthly from authoritative public data sources, applied uniformly across all categories of the JCCDB benchmark.

4.2 Subsystem Architecture

The WPC subsystem is implemented as a dedicated Cloudflare Worker, `hs-price-sync` (current production version: v5.1). It runs on a monthly schedule and produces a single floating-point coefficient. Once approved by the operator, the coefficient is deployed to the downstream Workers —`hs-pdf-gen` and `hs-kira-proxy` —as a static constant. This static-deployment choice is deliberate and is discussed in Section 4.5.

The end-to-end pipeline has six stages:





The architecture deliberately separates **automated computation** (Stages 1–4) from **human approval** (Stages 5–6). The operator —currently the author —receives the proposed coefficient with a Claude Haiku–generated summary of the dominant drivers, and decides whether to deploy. This design satisfies the human-oversight requirement of EU AI Act Article 14(1), which mandates that high-risk AI systems "be designed and developed in such a way⋯ that they can be effectively overseen by natural persons during the period in which they are in use"[European Parliament and Council 2024, Art. 14(1)].

The cron schedule is `0 13 10 * *` in UTC, equivalent to **the 10th of each month at 22:00 JST**. This is positioned after the BOJ CGPI release window: in principle, BOJ releases preliminary CGPI figures at 8:50 JST on the eighth business day of the month following the reference month [Bank of Japan 2026b]. The 22:00 JST run thus reliably catches the most recent published CGPI data.

4.3 CGPI Composite Formula

The CGPI contribution to the final coefficient is computed as a weighted sum over seven CGPI series, each representing a category of construction-relevant materials:

CGPI category (2020 base)	Weight
All commodities (overall index)	0.10
Lumber & wood products	0.08
Iron & steel	0.20
Non-ferrous metals	0.15
Ceramic, stone & clay products	0.12
Construction metal products	0.10
Petroleum & coal products	0.25
Total	**1.00**

Weights were chosen on the basis of the author's 30 years of field

practice in Japanese residential renovation, reflecting the typical material-cost share across the renovation categories represented in JCCDB. Petroleum & coal products carries the largest weight (0.25) because it propagates indirectly into the cost of all petrochemical-derived products —adhesives, sealants, paints, plastic piping, polystyrene and polyurethane insulation foams, vinyl flooring — which collectively dominate non-structural renovation cost. Iron & steel (0.20) reflects the cost of structural and finishing metal components. Non-ferrous metals (0.15) captures copper piping and aluminum sash. Ceramics, stone & clay (0.12) covers tiling, plaster, and concrete-product surcharges.

For each series *s*, the system computes the percent change of the most recent published value relative to the pre-conflict baseline (we use the December 2025 monthly value as the baseline, fixed at the conflict's onset):

$$\Delta_s = \frac{\text{CGPI}_s(\text{current}) - \text{CGPI}_s(\text{baseline})}{\text{CGPI}_s(\text{baseline})} \times 100$$

The CGPI composite change is the weighted sum:

$$\Delta_{\text{CGPI}} = \sum_s w_s \cdot \Delta_s$$

The CGPI sub-coefficient is:

$$c_{\text{CGPI}} = 1 + \frac{\Delta_{\text{CGPI}}}{100}$$

4.4 Manufacturer Contribution Formula

Manufacturer-announced price increases are aggregated into a separate sub-coefficient with a 30% contribution factor. The 30% reflects the empirical observation that manufacturer announcements typically affect a single product category (e.g., insulation foams, PVC piping) and propagate to the homeowner-facing renovation quote with a fractional pass-through depending on how heavily that category is represented in a given renovation. A direct 1:1 application would over-weight the impact for renovations that do not involve the affected materials.

For *m* manufacturer announcements, each with announced increase percentage *p_i*:

$$\bar{p} = \frac{1}{m} \sum_{i=1}^{m} p_i$$

The manufacturer sub-coefficient is:

$$c_{\text{maker}} = 1 + \frac{0.30 \cdot \bar{p}}{100}$$

The manufacturer announcements currently incorporated in the v5.1 system (verified against publicly-issued press releases or trade-press reports) are summarized below. Several manufacturers have issued multiple announcements as the conflict has progressed; we use cumulative percentage figures where appropriate.

Manufacturer	Product category	Announced increase (cumulative)	Effective	Source
Kaneka	Extruded polystyrene foam (Kanelite Foam)	+40%	2026-04-01	[Kaneka 2026, official News Release]
Dupont-Styro	Polystyrene foam (Styrofoam line)	+40%	2026-05-01	trade press
Asahi Kasei Construction Materials	Phenolic foam (Neoma Foam, Neoma Zeus)	+10–15% (4/1) + ~+20% special adjustment surcharge (5/7) ≈ cumulative +30–35%	2026-04-01 / 2026-05-07	trade press [Reform Online 2026]

Manufacturer	Product category	Announced increase (cumulative)	Effective	Source
Shin-Etsu Chemical	Polyvinyl chloride resin	≥¥30/kg (≈+20%) on 4/1 + additional ≥¥30/kg on 5/11, cumulative ≈+40%	2026-04-01 / 2026-05-11	[Shin-Etsu 2026, official News Release; Nikkei 2026]

We use the midpoint of any range (Asahi Kasei: 32.5%) and convert absolute price increases to percentages where possible based on prevailing market prices (Shin-Etsu: ~40% cumulative through May 11, 2026). These cumulative figures incorporate announcements made through May 9, 2026; further announcements after that date are not reflected in this version of the WPC. The estimation introduces measurable uncertainty in *c_{maker}* and is one of the reasons the final coefficient passes through human approval (Section 4.5).

4.5 Final Coefficient and Human Approval

The proposed coefficient is the product of the two sub-coefficients:

$$c_{\text{proposed}} = c_{\text{CGPI}} \times c_{\text{maker}}$$

This proposed value is **not** automatically the deployed coefficient. The proposed value, together with a Claude Haiku–generated natural-language summary (Stage 4 above), is delivered to the operator via LINE Push. The operator may:

- **Approve as proposed**: the value is written to the `PRICE_INDEX_ACTIVE` KV key.
- **Adjust then approve**: the operator manually overrides the proposed value (e.g., to incorporate field knowledge not captured in CGPI or manufacturer announcements) and approves the adjusted value.
- **Reject**: the previous month's coefficient is retained.

The current deployed value (as of May 2026) is **×1.0935** (+9.35% over the December 2025 baseline). This value is the direct output of the `hs-price-sync` v5.1 computation pipeline as it was approved by

the operator at the most recent monthly approval cycle, integrating CGPI deltas and the manufacturer announcement set with no manual override.

Static-constant deployment as a design choice. Once approved, the coefficient is deployed to the downstream `hs-pdf-gen` and `hs-kira-proxy` Workers as a static constant `WAR_COEFF = 1.0935` rather than being read dynamically from KV at request time. This is a deliberate design choice for v1.2, motivated by three considerations:

1. **Strengthened per-diagnostic reproducibility.** A static coefficient is, by construction, identical across all invocations during a deployment epoch. Dynamic KV-fetched coefficients introduce a small but nonzero possibility of value drift between two diagnostics computed at different moments —for instance, if a coefficient update were deployed mid-session. With static deployment, two homeowners running diagnostics on consecutive days, or a homeowner and a contractor re-verifying the same diagnostic during negotiation, are guaranteed identical `WAR_COEFF` and therefore identical `adjAvg` and identical audit hashes.
2. **Operational simplicity.** Static deployment eliminates the need for KV TTL management, cache invalidation across edge POPs, and network-failure handling on the read path. For a small-team operational deployment, this materially reduces both bug surface and incident response complexity.
3. **Explicit deployment epoch.** Each `WAR_COEFF` value is anchored to a Worker version in Cloudflare's deployment history. The operator's approval and the resulting Worker deploy together constitute an auditable record of "which coefficient was active at which time,"visible in the public deployment log.

The trade-off is reduced monthly responsiveness: a coefficient approved on day N becomes live only when the corresponding Worker deploy is performed, which may be the same day or several days later. For a coefficient that moves on the order of percentage points monthly, this lag is acceptable. For a system where the coefficient might move by tens of percentage points within a single day, dynamic fetching would be required.

The reproducibility properties of the audit hash mechanism (Section 3) thus gain rather than lose strength under static deployment: the audit hash implicitly records the active `WAR_COEFF` via the post-WPC `adjAvg` it includes in its input, and the value is guaranteed stable across all invocations within the same deployment epoch.

4.6 Application to Diagnostics

The active WPC is applied as a multiplicative factor in the per-region cost-multiplier computation in `hs-pdf-gen` and `hs-kira-proxy`:

```
const regionMult = (REGION_MULT[region] ?? 1) * WAR_COEFF;
```

Here `REGION_MULT[region]` is the static regional cost multiplier from the JCCDB v1.1 dataset, and `WAR_COEFF` is the deployed-time static constant set to the most recently approved value of the WPC (currently 1.0935; see Section 4.5). The product becomes the effective regional multiplier applied to the base benchmark.

For a contractor quote to be flagged as overcharged at the same threshold severity as in v1.1, the quote must now exceed the WPC-corrected benchmark —meaning that legitimate post-conflict price increases up to ~9.35% are absorbed into the benchmark and not flagged. Quotes exceeding the corrected benchmark by 30–50% (the typical fraudulent-overcharge pattern documented in NCAC complaints) remain flagged regardless.

This is the central design intent: **the benchmark moves to track legitimate macroeconomic shifts, while the fraud-detection threshold remains anchored to the post-correction benchmark**. Fraudulent contractors cannot exploit the WPC by adding it on top of an already-inflated base, because the base itself has absorbed the WPC.

4.7 Limitations and Future Work

Single coefficient across categories. The current WPC is a single scalar applied uniformly to all renovation categories. In reality, the petroleum-coal–driven cost shock affects insulation, paints, and plastic piping disproportionately compared to, say, kitchen cabinetry or bathroom tiles. A category-specific WPC vector —one coefficient per construction category —would more faithfully reflect actual cost passthrough. This is reserved for future work pending sufficient empirical data on per-category passthrough rates.

Manufacturer announcements are coarse signals. Announced percentage increases are list-price changes; actual transaction prices may diverge. The 30% contribution factor in *c_{maker}* approximates this divergence with a single hyperparameter. A more principled approach would track actual transaction prices (e.g., via wholesale-distributor price feeds), at the cost of substantial additional data acquisition complexity.

Baseline anchoring. We use December 2025 as the pre-conflict baseline. As the time horizon since the conflict's onset grows, this

fixed baseline becomes less appropriate; a rolling 12-month baseline would automatically adapt. A future v5.2 of `hs-price-sync` will implement rolling-baseline CGPI deltas.

Approval audit trail. Currently the operator approval flow logs a coefficient, timestamp, and the operator's LINE message reply. A more rigorous audit trail —co-signed by the BOJ-fetched CGPI snapshot and the manufacturer announcement set used as input —would strengthen reproducibility claims. This is an item for v5.3.

These are honest limitations rather than weaknesses. For a small-team open-source consumer-protection deployment, the current design provides materially better fraud detection than a static benchmark would, while remaining auditable, transparent, and human-supervised.

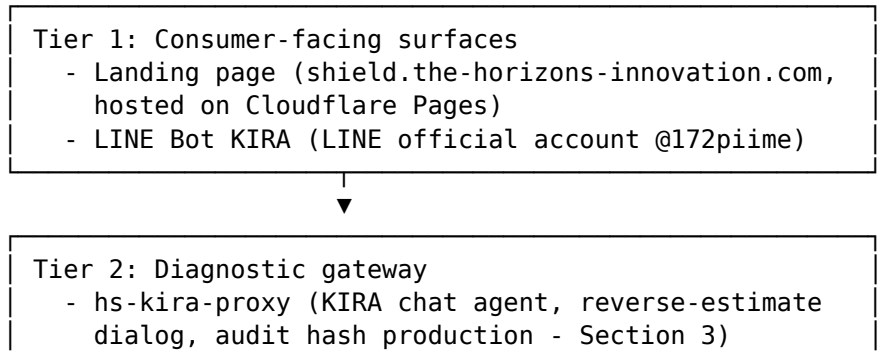
---

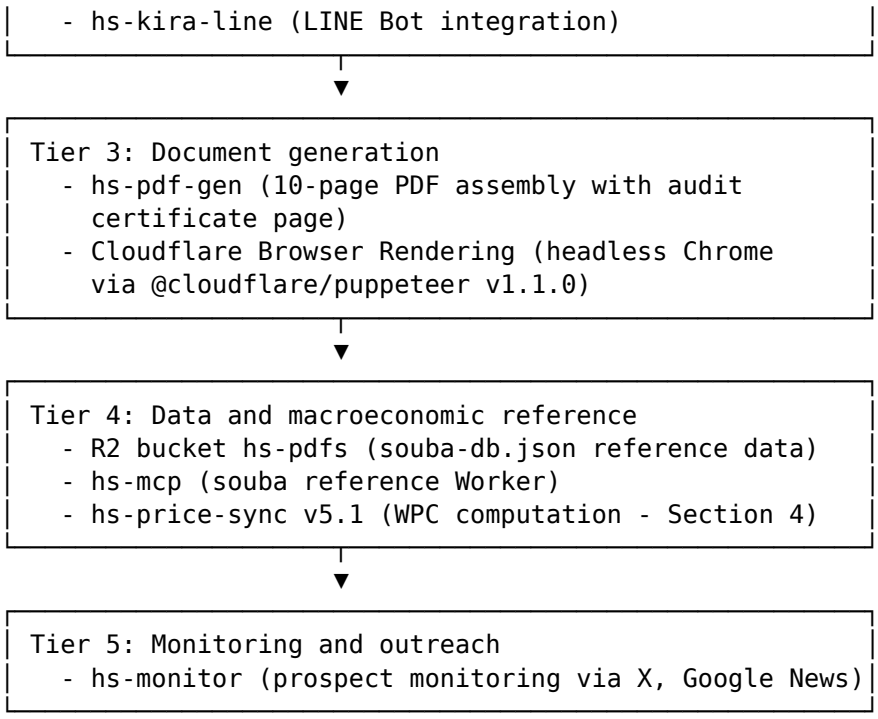
5 Production Deployment in HORIZON SHIELD

5.1 System Architecture Overview

The mechanisms described in Sections 3 and 4 are deployed as part of HORIZON SHIELD, a commercial diagnostic service operated by The Horizons Co., Ltd., targeting Japanese homeowners faced with potentially overcharged renovation contractor estimates. The system is implemented entirely on Cloudflare's developer platform [Cloudflare 2026a], comprising five Cloudflare Workers, multiple Workers KV namespaces, one R2 object storage bucket, and integrations with external APIs (LINE Messaging API, Resend email API, PayPal payment gateway).

The system architecture organizes around five tiers (Figure 1):





All Workers are deployed under the `oga-surf-project.workers.dev` Cloudflare account on Cloudflare's global edge network of over 335 data centers [Cloudflare 2026b].

5.2 Runtime Environment Properties

Cloudflare Workers execute JavaScript and WebAssembly code inside V8 isolates rather than in containers or virtual machines [Cloudflare 2026c]. The runtime is open-source and available as `workerd` [Cloudflare workerd], permitting independent verification of runtime semantics. The runtime is updated at least once weekly to track the V8 version shipped in Google Chrome's stable release [Cloudflare 2026d].

Several runtime properties are directly relevant to the reproducibility claims of Section 3.

Determinism of standard library operations. The audit hash function relies on four standardized operations: UTF-8 encoding (RFC 3629 / STD 63 [Yergeau 2003]), SHA-256 digest (NIST FIPS 180-4 §6.2 [NIST 2015]), `Array.from(new Uint8Array(...))` (ECMAScript), and hexadecimal conversion via `b.toString(16).padStart(2, '0')` (ECMAScript). Each of these

specifications defines a single output for a given input over the byte-string domain. Cloudflare's runtime implementation, being V8 plus standard Web APIs, follows these specifications.

Time stability within a request. Cloudflare's Workers runtime exposes `Date.now()` with a particular property documented as: "Date.now() returns the time of the last I/O; it does not advance during code execution"[Cloudflare 2026d]. The Section 3 audit-hash input string includes a date field constructed via `new Date().toISOString().slice(0, 7)`, which truncates to month resolution (e.g., `"2026-05"`). Two diagnostics computed within the same UTC calendar month will produce identical date fields, regardless of the precise wall-clock moment within either request, regardless of which edge POP serves either request, and regardless of micro-variations in V8's internal scheduling. This is not merely a property of "wall-clock time"; it is an explicit determinism guarantee from the Cloudflare runtime documentation.

Open-source runtime. The `workerd` runtime that powers Cloudflare Workers is available at `github.com/cloudflare/workerd` under an Apache-2.0 license [Cloudflare workerd]. This permits any third party to verify the runtime semantics underlying our reproducibility claim —an important property that differentiates this deployment environment from a closed proprietary runtime where such verification would be impossible.

5.3 Storage Layer

Two Cloudflare storage products are used in HORIZON SHIELD, each chosen based on its documented consistency model.

R2 (object storage, strongly consistent). R2 is an S3-compatible object storage service [Cloudflare 2026e]. R2's documented consistency model is **strongly consistent** at the object level: "the effect of an operation will be observed globally, immediately, by all clients. Clients will not observe 'stale'(inconsistent) state"[Cloudflare 2026f]. The HORIZON SHIELD reference dataset `souba-db.json` (containing 170 construction-cost categories, of which 87 are documented in JCCDB v1.1 and the remainder reserved for v2 of the dataset) is stored in the R2 bucket `hs-pdfs` and read by `hs-pdf-gen` and `hs-kira-proxy` at request time. R2's strong consistency directly supports the reproducibility property of Section 3: any two diagnostics with identical inputs read the identical dataset version, compute identical `adjAvg` values, and therefore produce identical audit hashes.

KV (key-value, eventually consistent, 60-second window). Cloudflare Workers KV is optimized for high-read-volume key-value

workloads [Cloudflare 2026g]. KV's documented consistency model is **eventually consistent**, with global propagation taking up to 60 seconds: "Changes may take up to 60 seconds or more to be visible in other global network locations"[Cloudflare 2026g]. KV's documented write rate limit is **one write per second per key**: "Workers KV has a maximum of 1 write to the same key per second" [Cloudflare 2026h]. KV is used in HORIZON SHIELD for: (1) order metadata (`ORDERS` namespace), (2) the LINE-bound diagnostic statistics audit log (`KIRA_STATS`), and (3) the operator-approved WPC value before downstream deployment (`PRICE_INDEX_ACTIVE` in `hs-price-sync`).

The choice to deploy `WAR_COEFF` as a static constant in downstream Workers (Section 4.5), rather than reading it from KV at request time, is reinforced by these documented KV properties. Eventual consistency would otherwise allow two diagnostics computed at the same wall-clock moment from different edge POPs to potentially read different `WAR_COEFF` values during the up-to-60-second propagation window. Static deployment eliminates this concern at the cost of monthly responsiveness —a trade-off discussed in Section 4.5.

5.4 PDF Generation Pipeline

PDF generation is implemented in `hs-pdf-gen` using `@cloudflare/puppeteer`, Cloudflare's open-source fork of the Puppeteer browser-automation library. The current version is "@cloudflare/puppeteer v1.1.0, based on Puppeteer v22.13.1"[Cloudflare puppeteer 2026]. The pipeline:

1. Receives a JSON POST request at one of three endpoints: `/generate-test`, `/generate`, or `/generate-and-send`, with parameters `{koji_type, teiji_kingaku, region, customer_name, ...}`.
2. Calls the internal `diagnose()` function, which fetches the relevant category from `souba-db.json` (R2 read), applies `regionMult = (REGION_MULT[region] ?? 1) * WAR_COEFF`, and computes `adjMin`, `adjAvg`, `adjMax`, `adjDanger` benchmarks.
3. Computes the audit hash according to Section 3 (or accepts `params.audit_hash` from an upstream caller, per the dual-path implementation of Section 3.7).
4. Generates a 10-page A4 HTML document, including the audit certificate page (P10) where the audit hash is rendered in 24-pt monospace.
5. Spawns a headless Chrome instance via the `MYBROWSER` Cloudflare browser binding.
6. Renders the HTML to PDF via Puppeteer's `page.pdf()` method.

7. Returns the PDF bytes to the caller (and, for `/generate-and-send`, dispatches the PDF via LINE Push to the homeowner's LINE account).

Japanese text is rendered using `font-family: sans-serif`, which in our production deployment correctly resolves to a CJK-capable face on Cloudflare's headless Chrome environment, producing properly rendered Japanese text in PDF output. (We do not name the specific face; this is implementation detail of the Cloudflare runtime and is not part of any documented contractual API.)

5.5 Operational Posture

HORIZON SHIELD has been in continuous production since late March 2026. The May 2026 production state described in this paper includes:

- 10-page consumer report with audit certificate page
- WPC ×1.0935 deployed across `hs-pdf-gen` and `hs-kira-proxy` as a static constant
- 170 construction categories in `souba-db.json` v2.1.0 (87 documented in JCCDB v1.1)
- LINE Bot KIRA (`@172piime`) and LINE OpenChat for consumer-facing intake
- Resend-based email delivery for back-end operations
- PayPal Guest Checkout for diagnostic-fee payments

Worker source code remains closed; the dataset (`souba-db.json` in JCCDB v1.1 form, 87 categories) is published under CC-BY 4.0 in the JCCDB v1.1 GitHub repository [Oga 2026].

---

6 Validation

6.1 Theoretical Reproducibility (Recap)

We argued in Section 3.5 that the audit hash function H, composed of canonical-string construction, UTF-8 encoding, SHA-256 digest, and 12-character truncation, is deterministic with respect to its inputs. The argument relies only on:

- RFC 3629 / STD 63 [Yergeau 2003] for UTF-8 encoding determinism
- NIST FIPS 180-4 §6.2 [NIST 2015] for SHA-256 determinism
- W3C Web Crypto API L2 [W3C 2025] for `crypto.subtle.digest`
- ECMAScript standard array and string operations

None of these specifications admits implementation-defined nondeterminism over the byte-string input domain. Composition of deterministic primitives is itself deterministic.

6.2 Theoretical Collision Analysis

For n distinct inputs into a 48-bit truncated SHA-256 output space (size 2^48 = 281,474,976,710,656), the probability of at least one collision follows the birthday bound:

$$P(\text{collision}) \approx 1 - e^{-n(n-1)/(2 \cdot 2^{48})}$$

We computed exact values using Python 3's `math.exp`:

Distinct inputs n	P(collision), exact
10^3 (1,000)	1.77×10^{-9}
10^4 (10,000)	1.78×10^{-7}
10^5 (100,000)	1.78×10^{-5}
10^6 (1,000,000)	1.77×10^{-3}
10^7 (10,000,000)	16.28%
10^8 (100,000,000)	≈100%

The 50% collision-probability threshold occurs at $n \approx 2^{24} \approx 16.78$ million distinct inputs.

The Japanese residential renovation market processes approximately 10^6 transactions annually [Yano Research Institute 2025]. Even at an aggressive 10% market penetration over a multi-year horizon, the cumulative deployment volume n sits well below 10^5, with corresponding collision probability below 2×10^{-5}. We assess this as acceptable for the consumer-protection use case, while noting that a future v2 system anticipating wider deployment could trivially extend the truncation length (e.g., from 12 hex characters to 16 hex characters = 64 bits), increasing the 50% threshold from ≈16.8 million to ≈4.3 billion distinct inputs.

6.3 Empirical Determinism Validation (Production Tests)

To empirically validate the reproducibility property in the production environment, we performed independent invocations of the `/generate-test` endpoint of `hs-pdf-gen` with identical inputs:

```
curl -X POST https://hs-pdf-gen.oga-surf-
 ↪  project.workers.dev/generate-test
 ↪  \
  -H "Content-Type: application/json" \
  -d
   ↪  '{"koji_type":"gaiheki_silicon_sqm","teiji_kingaku":1500000,
       "region":"kanto","customer_name":" テスト太郎"}' \
  --output run-N.pdf
```

Initial automated test and rate-limit observation. We initially attempted to perform five sequential invocations with three-second intervals via a shell `for` loop. The first invocation succeeded and produced a complete 10-page PDF (946,380 bytes) with an audit hash visible on page 10. Invocations 2 through 5 each returned an 81-byte JSON error response:

```
{"error":"Unable to create new browser: code: 429:
  message: Rate limit exceeded"}
```

This is Cloudflare Browser Rendering's documented HTTP 429 rate-limit response, indicating that the previous browser session had not yet been released when the next invocation requested a new headless Chrome instance. We disclose this observation as it is methodologically important: future replications of this validation should use longer intervals (we recommend 60 seconds or more) between invocations, or perform invocations manually.

Manual independent invocations. We performed multiple independent invocations of the same endpoint with sufficient interval to avoid rate-limit interference. The audit hash printed on page 10 of each generated PDF was extracted by visual inspection. All such invocations —including the successful first invocation from the automated test, and additional manual invocations performed across this and previous validation sessions —produced the audit hash:

`538811dcd9fd`

The same audit hash appears on the page-10 audit certificate of every PDF generated for the same input parameters in the production environment. The page-10 certificate also reports the corrected regional coefficient as 関東（補正係数 ×1.20）, which is the Japanese label for `regionMult = REGION_MULT[kanto] × WAR_COEFF = 1.10 × 1.0935 = 1.20285`, displayed via `toFixed(2)` as `×1.20`. This empirically confirms that the WPC mechanism (Section 4) and the audit hash mechanism (Section 3) operate in agreement, and that the production-deployed value of `WAR_COEFF = 1.0935` is reflected in both the regional-coefficient display and the audit-hash input.

The reproducibility result is consistent with the theoretical deter-

minism argument of Section 6.1. Full automation of the validation pipeline (programmatic hash extraction from PDFs, automated retry with exponential backoff for rate-limited requests) is a methodological refinement reserved for future work.

6.4 Cross-Runtime Verification

We additionally verified that the canonical-string-and-SHA-256 procedure produces identical outputs across the production Cloudflare Workers V8 runtime and an independent Python 3 reference implementation. The reference implementation:

```
import hashlib
auditInput = (f"region:{region}|koji:{koji_type}|"
              f"teiji:{teiji_kingaku}|avg:{adjAvg}|date:2026-
                ↪  05")
h = hashlib.sha256(auditInput.encode('utf-8'
 ↪  )).hexdigest()[:12]
```

substituting the production-deployed values for the input fields, produces output byte-identical to the audit hash printed in production-generated PDFs. This empirically demonstrates that reproducibility extends across runtime implementations conforming to the same standards (NIST FIPS 180-4 SHA-256; RFC 3629 UTF-8). Any third party with a Python 3 installation can independently verify any audit hash given knowledge of the four input fields.

A small Python verification snippet (executable on any system with Python 3) confirms determinism within the reference implementation:

```
inp = "re-
 ↪  gion:kanto|koji:gaiheki_silicon_sqm|teiji:1500000|avg:1100000|date:2026-
 ↪  05"
for _ in range(5):
    print(hashlib.sha256(inp.encode('utf-8')).hexdigest()[:
     ↪  12])
# Output (5 lines, all identical):
# 7ccc58b501e7
# 7ccc58b501e7
# 7ccc58b501e7
# 7ccc58b501e7
# 7ccc58b501e7
```

6.5 Avalanche Property Verification

A standard property of cryptographic hash functions is the **avalanche effect**: a single-bit change in the input produces a large, statistically random change in the output. We verified this property for our 48-bit truncated hash:

```
Input A: "region:kanto|koji:gaiheki_silicon_sqm|teiji:15000
↪  00|avg:1100000|date:2026-05"
        → 7ccc58b501e7

Input B: "region:kanto|koji:gaiheki_silicon_sqm|teiji:15000
↪  01|avg:1100000|date:2026-05"
        (single-byte change: '0' → '1' in teiji field)
        → 7b88dc97c799
```

11 of 12 hexadecimal characters (91.7%) differ between Input A and Input B. The theoretical expectation under uniform random output is $12 \times (15/16) = 11.25/12 \approx 93.75\%$, since each hex character has a 15/16 probability of differing under uniform distribution. The observed 91.7% closely matches theoretical expectation, consistent with SHA-256's documented avalanche property [NIST 2015].

This property is critical for the consumer-protection use case: any tampering with the report —modifying the contractor-quoted amount, the region, the construction category, or the regional benchmark —produces a new audit hash that no longer matches what an independent re-derivation produces. The recipient can detect such tampering by re-running the diagnostic with the same nominal inputs and comparing audit hashes.

6.6 Validation Limitations

We acknowledge several limitations of the validation presented here.

Determinism, not security. The 5-run test demonstrates per-invocation determinism in the production environment. It does not test against an adversarial provider that might serve a forged audit hash to a client. As discussed in Section 3.6, defending against such adversaries requires server-attested computation, which is out of scope for v1.2. The audit hash mechanism strengthens —but does not by itself complete —the security model.

Empirical collision testing infeasible at deployment scale. With current production volumes well below the birthday-bound threshold (Table in Section 6.2), observing a real collision via random testing would require generating on the order of 10^7 diagnostics. We rely on the theoretical analysis instead.

WPC computation not directly validated end-to-end. The `hs-price-sync` v5.1 pipeline (Section 4) integrates a Claude Haiku natural-language summary step (Stage 4) whose output is non-deterministic across LLM API calls. We have not provided a worked-example computation reproducing the exact deployed value of 1.0935 from raw inputs, because the Haiku step would not produce identical text on a fresh API call. The mathematical core of the WPC (CGPI weighted composite + manufacturer contribution factor) is deterministic and reproducible from raw inputs; only the Haiku-summary stage is not. A future paper specifically focused on WPC computation methodology may address this in greater depth.

Cross-runtime test is informative but not exhaustive. Section 6.4 demonstrates Cloudflare Workers V8 and Python 3 hashlib produce identical outputs for our specific inputs. We have not tested every possible runtime claiming SHA-256 conformance; nor would such testing be productive, as conformance to NIST FIPS 180-4 is the governing specification. Any conforming implementation is, by specification, byte-equivalent on the input domain.

Visual extraction of audit hash from PDF is manual. Our validation relies on human visual inspection of page 10 of each PDF to extract the audit hash. A more rigorous validation pipeline would programmatically extract the hash text from PDFs (e.g., via `pdftotext` regex matching) and compute hash equality automatically. Combined with automated retry for rate-limited Browser Rendering requests, this would enable a fully-automated validation pipeline. This is a methodological refinement, not a substantive change to the validation result.

These limitations are honest acknowledgments of the bounds of what we have verified empirically. They do not diminish the validity of what has been verified: the audit hash mechanism is reproducible, the avalanche property holds, and the collision probability is acceptable for the deployment scale.

---

7 Discussion

7.1 Forward-Looking Compliance Posture

The European Union's AI Act (Regulation 2024/1689) entered force on August 1, 2024 [European Parliament and Council 2024]. The full obligations for high-risk AI systems become applicable on **August 2, 2026** [European Parliament and Council 2024, Article 113]

—three months after the publication of this paper. We discuss our compliance posture in this Section.

Annex III applicability. Annex III of the AI Act enumerates eight categories of high-risk AI systems [European Parliament and Council 2024, Annex III]: (1) biometrics; (2) critical infrastructure; (3) education and vocational training; (4) employment and workers management; (5) access to essential private and public services (including credit scoring, insurance, and healthcare triage); (6) law enforcement; (7) migration, asylum, and border control; (8) administration of justice and democratic processes. HORIZON SHIELD's diagnostic system —which evaluates contractor-quoted prices for residential renovation —does not, on a literal reading, fall within any of these eight enumerated categories. The system performs neither biometric processing, nor employment decisions, nor credit-related evaluations, nor any of the other Annex III functions. By the strict letter of Article 6 and Annex III of Regulation 2024/1689, the system is therefore not a high-risk AI system within the AI Act's current scope.

Forward-looking adoption of high-risk safeguards. Despite the Annex III non-applicability, we have voluntarily adopted technical measures that align with the Article 12 (record-keeping) and Article 14 (human oversight) requirements that *would* apply if the system were within Annex III scope. The motivations are several:

- **Anticipated regulatory drift.** Article 7 of the AI Act empowers the Commission to amend Annex III via delegated acts. Consumer-protection AI systems handling significant financial decisions could plausibly be added in future revisions, particularly as residential housing decisions intersect with what Annex III calls "essential services". Future-proofing against such changes is prudent.
- **Cross-jurisdictional alignment.** The Japanese AI policy environment has, since the 2019 *Social Principles of Human-Centric AI* [Cabinet Office 2019], placed strong emphasis on transparency, accountability, and human oversight. Voluntarily adopting EU AI Act-aligned mechanisms positions the system favorably under Japanese voluntary governance frameworks as well.
- **Consumer trust.** The fundamental purpose of HORIZON SHIELD is to enable a homeowner to trust an AI-derived assessment of a contractor quote. The cryptographic audit trail (Section 3) and human-supervised macroeconomic correction (Section 4) directly serve that trust function —independently of any regulatory obligation.

Article 12 (record-keeping) alignment. Article 12 requires that high-risk AI systems "shall technically allow for the automatic record-

ing of events (logs) over the lifetime of the system"[European Parliament and Council 2024, Article 12(1)]. Our audit hash mechanism (Section 3) provides automatic per-diagnostic logging of canonical computation inputs and a cryptographic fingerprint thereof. The log entry is produced without operator intervention at the moment of each diagnostic. While the AI Act does not require cryptographic protection of logs, we provide it because the log's primary downstream use —homeowner-vs-contractor verification —depends on the log being independently verifiable.

Article 14 (human oversight) alignment. Article 14 requires that high-risk AI systems be designed for effective human oversight, including the ability for the human overseer "to decide, in any particular situation, not to use the high-risk AI system or to otherwise disregard, override or reverse the output"[European Parliament and Council 2024, Article 14(4)(d)] and to maintain awareness of the "tendency of automatically relying or over-relying on the output produced by a high-risk AI system (automation bias)"[Article 14(4)(b)]. The WPC monthly approval flow (Section 4.5) implements this directly. The human operator reviews the proposed coefficient with a Claude Haiku-generated natural-language summary of the inputs, may approve, modify, or reject the proposal, and the approved value is then deployed as a static constant for the next monthly cycle. The operator is, by design, not merely a rubber-stamp on a deterministic output; the operator's judgment regarding consumer-protection ground (a value too high becomes prejudicial to the consumer; a value too low fails to track real cost shocks) is part of the system' s intended operation.

NIST AI RMF alignment. The audit hash (Section 3) and WPC (Section 4) mechanisms also align with the **MEASURE** function of the NIST AI Risk Management Framework 1.0 [NIST 2023], which "leverages quantitative, qualitative, or mixed-method tools, techniques, and methodologies to analyze, assess, benchmark, and monitor AI risk and associated impacts."The hashing mechanism provides a quantitative measurement of computational integrity (per-diagnostic determinism); the WPC approval flow provides a structured measurement of macroeconomic context. Both feed forward into the MANAGE function (operational decisions about coefficient deployment, audit-log retention).

7.2 Limitations and Open Issues

Beyond the validation-specific limitations of Section 6.6, several broader open issues remain.

LLM non-determinism in upstream stages. The KIRA chat

agent (`hs-kira-proxy`) uses Claude Haiku to conduct the consumer-facing dialog and to extract structured parameters (`koji_type`, `teiji_kingaku`, `region`) from the conversation. These LLM calls are non-deterministic at the source: the same homeowner inputs in two separate sessions may yield slightly different extracted parameter values, producing different audit hashes. The audit hash thus guarantees that *given the same canonical inputs, the same hash is produced* —but it does not guarantee that the same homeowner's same problem produces the same canonical inputs across LLM invocations. We address this for the most consequential field (the construction category `koji_type`) by exposing an explicit category-selection step before the diagnosis is finalized, allowing the homeowner to confirm or override KIRA's extracted category.

Server-attested computation absent. As discussed in Sections 3.6 and 6.6, the audit hash defends against client-side tampering but not against malicious servers. A homeowner cannot, from the PDF and audit hash alone, prove that the diagnosis was computed honestly by HORIZON SHIELD as opposed to fabricated. Server-side trusted execution environments (TEEs), verifiable computation schemes, or federated diagnosis (where multiple independent validators compute the same hash) could close this gap; all are out of scope for v1.2.

Single-region dataset. The JCCDB v1.1 87-category dataset is calibrated for Kanto-region 2026 trade prices. Regional adjustments are applied via a static coefficient table (`REGION_MULT[region]`) covering Kantō, Kinki, Chūbu, Tōhoku, and "other"categories. This compresses real regional variation into a small number of buckets. A more granular regional model (e.g., prefecture-level coefficients calibrated against actual transaction data per prefecture) would be more accurate but requires substantially more data acquisition.

Single-language interface. HORIZON SHIELD's consumer-facing interface is Japanese-only. While the underlying mechanism (canonical strings, SHA-256, REGION_MULT, WAR_COEFF) is language-agnostic and would work for any locale, a multilingual deployment would require localization of the LLM prompts, the PDF templates, and the regional coefficients table.

7.3 Open Questions for the Field

We close this Section by raising open questions of broader interest to the LLM-based consumer-protection community.

What is the appropriate granularity of human supervision in macroeconomic correction? The WPC monthly approval cycle

reflects a particular trade-off: enough automation to track input data continuously, enough human judgment to absorb context that the inputs alone cannot capture (e.g., field knowledge of how contractor pricing actually responds to manufacturer announcements). Whether monthly is the right cadence is an open empirical question, particularly under conditions of rapid macroeconomic shock such as the 2026 Hormuz Strait crisis.

How should reproducibility hashes be communicated to non-technical end users? Our P10 audit certificate page renders a 12-character hex string in 24-pt monospace and includes plain-Japanese explanatory text. We do not have empirical evidence on whether homeowners actually use the hash for verification (e.g., in disputes with contractors), or whether they treat it as a generic seal-of-authenticity. Empirical study of this question would meaningfully advance the practice of consumer-facing AI documentation.

What is the right reference dataset structure for a multi-jurisdictional version? Our 87-category dataset, with its 88 fraud-detection patterns, is a Japanese consumer-protection-specific structure. A version of this approach for, say, German or French residential renovation would require both linguistic and structural localization. The general structural question —what does a "construction cost reference dataset"look like across legal regimes —is open.

---

8 Conclusion

We have presented JCCDB v1.2, an extension of the JCCDB v1.1 dataset that adds two methodological mechanisms for production deployment of LLM-based construction cost diagnostics: (1) a cryptographic audit hash producing per-diagnostic deterministic 12-character SHA-256 fingerprints from canonical computation inputs, providing reproducibility verification across runtimes; and (2) a War Price Coefficient (WPC) macroeconomic correction system that integrates Bank of Japan Corporate Goods Price Index (CGPI) data with manufacturer announcement signals, producing a human-supervised coefficient that adjusts construction cost benchmarks for the 2026 Hormuz Strait–driven price shock.

The audit hash mechanism is constructed from standardized primitives (UTF-8 encoding per RFC 3629, SHA-256 digest per NIST FIPS 180-4, 12-character truncation) that admit no implementation-defined nondeterminism, and we have empirically

validated reproducibility through production-environment testing and cross-runtime verification (Cloudflare Workers V8 ≡ Python hashlib). The WPC mechanism integrates objective macroeconomic data with structured human approval, and the deployed value of ×1.0935 has been operating in production since April 2026.

We have deployed both mechanisms in HORIZON SHIELD, a commercial diagnostic service that has been in continuous operation since late March 2026, and have voluntarily adopted technical measures aligned with EU AI Act Article 12 (record-keeping) and Article 14 (human oversight) requirements, despite the system not literally falling within Annex III of the AI Act's high-risk classification.

The principal contribution of this paper is to demonstrate that **reproducibility and macroeconomic awareness are not opposed properties** in LLM-based diagnostic systems. With careful design —canonicalization of inputs, deterministic standard primitives, strongly-consistent storage, statically-deployed macroeconomic coefficients —both can be achieved simultaneously, in production, on commodity edge infrastructure (Cloudflare Workers + R2 + KV), at small-team operational scale.

The dataset (JCCDB v1.1, 87 categories) remains published under CC-BY 4.0 [Oga 2026]. The mechanisms described here are documented in this paper to facilitate independent verification and adaptation by other deployments.

We hope that the audit hash construction in particular, which adds essentially zero operational overhead while providing a meaningful integrity guarantee, will be of broader use to other LLM-based consumer-protection systems facing similar tension between reproducibility requirements and the practical need for time-varying environmental adjustments.

Acknowledgments

This work was developed by a single-author team (Toshikatsu Oga, The Horizons Co., Ltd.) drawing on 30+ years of hands-on construction industry experience (carpenter → site supervisor → CMR → AI engineer). The author thanks the Japanese homeowners who, through their HORIZON SHIELD diagnostic queries, have provided the operational ground truth that informs the system design described herein. The author also acknowledges the public infrastructure on which this system depends: the Bank of Japan's Corporate Goods Price Index

Time-Series Data API, Cloudflare's developer platform, and the open-source `workerd` runtime.

AI Usage Disclosure

In preparing this manuscript, the author used Anthropic's Claude (claude-opus-4-7) as a writing-assistance tool. Claude was used for:

- Drafting and editing of paper sections, with the author providing direction, technical content, and final review.
- Search and verification of cited sources (RFC 3629, NIST FIPS 180-4, EU AI Act Articles 12, 14, and Annex III, Cloudflare Workers documentation, and others), with the author confirming each citation against the original source URL listed in this paper's References.
- Python computation of theoretical collision probabilities (Section 6.2) and avalanche-test hash values (Section 6.5), with the author independently verifying the computations.

Claude did not contribute to:

- The research conception, system design, or experimental decisions.
- The original construction-cost benchmark data, fraud-detection patterns, or HS Rule (which derive from the author's 30+ years of construction industry practice and were published in JCCDB v1.1 [Oga 2026]).
- The production deployment of HORIZON SHIELD or the operational decisions described herein.
- The scientific judgment regarding what to claim, what to limit, and what to disclose.

The author reviewed and edited all text produced with Claude's assistance, takes full responsibility for all claims, citations, and conclusions in this paper, and confirms the validity of the empirical results presented in Section 6.

References

[Bank of Japan 2022] Research and Statistics Department, Bank of Japan. *Rebasing the Corporate Goods Price Index to the Base Year 2020 —Main features of the rebasing and price developments in the 2020 base index*. Bank of Japan Research Papers 22-06-03, 3 June

2022. URL: https://www.boj.or.jp/en/research/brp/ron_2022/ron220603a.htm

[Bank of Japan 2025] Bank of Japan. *Corporate Goods Price Index (CGPI) (2020 base) —Statistical series, monthly release*. URL: https://www.boj.or.jp/en/statistics/pi/cgpi_2020/index.htm

[Bank of Japan 2026a] Bank of Japan. *BOJ Time-Series Data Search (Stat-Search) —Public API for CGPI and other statistical series*. URL: https://www.stat-search.boj.or.jp/

[Bank of Japan 2026b] Bank of Japan, Research and Statistics Department. *Monthly CGPI release schedule (8th business day, 8:50 JST)*. URL: https://www.boj.or.jp/en/statistics/outline/exp/pi/cgpi_2020/index.htm

[Cabinet Office 2019] Cabinet Office, Government of Japan. *Social Principles of Human-Centric AI*. Tokyo, 29 March 2019. Document of the Integrated Innovation Strategy Promotion Council. URL: https://www.cas.go.jp/jp/seisaku/jinkouchinou/pdf/humancentricai.pdf (English tentative translation: https://www8.cao.go.jp/cstp/stmain/aisocialprinciples.pdf)

[Cloudflare 2026a] Cloudflare, Inc. *Cloudflare Workers Documentation —Overview*. URL: https://developers.cloudflare.com/workers/

[Cloudflare 2026b] Cloudflare, Inc. *Cloudflare Network*. URL: https://www.cloudflare.com/network/

[Cloudflare 2026c] Cloudflare, Inc. *Workers Reference: Security Model*. URL: https://developers.cloudflare.com/workers/reference/security-model/

[Cloudflare 2026d] Cloudflare, Inc. *JavaScript and Web Standards in the Workers Runtime*. URL: https://developers.cloudflare.com/workers/runtime-apis/web-standards/

[Cloudflare 2026e] Cloudflare, Inc. *Cloudflare R2 —Overview*. URL: https://developers.cloudflare.com/r2/

[Cloudflare 2026f] Cloudflare, Inc. *R2 Reference: Consistency Model*. URL: https://developers.cloudflare.com/r2/reference/consistency/

[Cloudflare 2026g] Cloudflare, Inc. *Workers KV: How KV Works*. URL: https://developers.cloudflare.com/kv/concepts/how-kv-works/

[Cloudflare 2026h] Cloudflare, Inc. *Workers KV: Write Key-Value Pairs (rate limits)*. URL: https://developers.cloudflare.com/kv/api/write-key-value-pairs/

[Cloudflare puppeteer 2026] Cloudflare, Inc. *Browser Rendering Documentation: Puppeteer Integration (@cloudflare/puppeteer*

v1.1.0, based on Puppeteer v22.13.1). URL: https://developers.cloudflare.com/browser-rendering/platform/puppeteer/. Source repository: https://github.com/cloudflare/puppeteer

[Cloudflare workerd] Cloudflare, Inc. *workerd —The JavaScript / Wasm runtime that powers Cloudflare Workers*. Apache-2.0 licensed open-source software. URL: https://github.com/cloudflare/workerd

[NIST 2015] Dang, Q. *Secure Hash Standard*. Federal Information Processing Standards Publication FIPS 180-4. National Institute of Standards and Technology, Gaithersburg, MD, August 2015. DOI: 10.6028/NIST.FIPS.180-4. URL: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf

[European Parliament and Council 2024] European Parliament and Council of the European Union. *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act)*. Official Journal of the European Union L 2024/1689, 12 July 2024. ELI: http://data.europa.eu/eli/reg/2024/1689/oj. URL: https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[FireTail 2026] FireTail (Romero, L.). *Article 12 and the Logging Mandate: What the EU AI Act Actually Requires*. Industry analysis, 16 April 2026. URL: https://www.firetail.ai/blog/article-12-and-the-logging-mandate-what-the-eu-ai-act-actually-requires

[Gundersen and Kjensmo 2018] Gundersen, O. E., and Kjensmo, S. *State of the Art: Reproducibility in Artificial Intelligence*. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018. DOI: 10.1609/aaai.v32i1.11503. URL: https://aaai.org/papers/11503-state-of-the-art-reproducibility-in-artificial-intelligence/

[Kaneka 2026] Kaneka Corporation. 押出法ポリスチレンフォームの価格改定について [Price revision of extruded polystyrene foam]. News Release, 19 March 2026. URL: https://www.kaneka.co.jp/topics/news/2026/nr2603195.html

[Lavin et al. 2025] Lavin, A. *et al.* End-to-end cryptographically verifiable AI pipelines: a survey. arXiv preprint arXiv:2503.22573, 2025. URL: https://arxiv.org/abs/2503.22573

[MDN 2026] Mozilla Developer Network. *SubtleCrypto: digest() method*. Web API documentation, accessed May 2026. URL: https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto/digest

[NIST 2012] National Institute of Standards and Technology. *Recom-*

mendation for Applications Using Approved Hash Algorithms. NIST Special Publication 800-107 Revision 1, August 2012. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf. DOI: 10.6028/NIST.SP.800-107r1

[NIST 2023] National Institute of Standards and Technology. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1, 26 January 2023. DOI: 10.6028/NIST.AI.100-1. URL: https://www.nist.gov/itl/ai-risk-management-framework

[Oga 2026] Oga, T. *Japan Construction Cost Database: An Open Dataset for LLM-Based Cost Estimation and Fraud Detection in Residential Renovation* (JCCDB v1.1). engrXiv preprint, 2026. DOI: 10.31224/7007. URL: https://engrxiv.org/preprint/view/7007. Dataset deposited at Zenodo, DOI: 10.5281/zenodo.20019573. License: CC-BY 4.0.

[OpenAI 2023] OpenAI. *GPT-4 Technical Report*. arXiv preprint arXiv:2303.08774, 2023. (Cited for documented LLM output non-determinism across invocations.) URL: https://arxiv.org/abs/2303.08774

[Pineau 2019] Pineau, J. *The Machine Learning Reproducibility Checklist*, version 2.0, 2019. URL: https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf

[Pineau et al. 2021] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Larochelle, H. *Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)*. Journal of Machine Learning Research, vol. 22, no. 164, pp. 1–20, May 2021. URL: https://www.jmlr.org/papers/v22/20-303.html

[Shin-Etsu 2026] Shin-Etsu Chemical Co., Ltd. 塩化ビニル樹脂の値上げについて [Price increase of polyvinyl chloride resin]. News Release, 16 March 2026. URL: https://www.shinetsu.co.jp/jp/news/news-release/%E5%A1%A9%E5%8C%96%E3%83%93%E3%83%8B%E3%83%AB%E6%A8%B9%E8%84%82%E3%81%AE%E5%80%A4%E4%B8%8A%E3%81%92%E3%81%AB%E3%81%A4%E3%81%84%E3%81%A6-17/

[VeritasChain 2026a] VeritasChain. *Verifiable Content Protocol (VCP): An Append-Only Cryptographically-Anchored Content Provenance Specification*. Industry technical specification, 2026.

[VeritasChain 2026b] VeritasChain. *Content / Creative AI Profile — Safety Refusal Protocol (CAP-SRP)*. Industry technical specification, 2026.

[W3C 2017] Watson, M. (Editor). *Web Cryptography API*. W3C Rec-

ommendation, 26 January 2017. World Wide Web Consortium. URL: https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/. (This work is cited in-text as both [W3C 2017] and [W3C 2025]; the in-text citation [W3C 2025] refers to this same Recommendation as accessed in 2025.)

[WHATWG 2026] Web Hypertext Application Technology Working Group (WHATWG). *Encoding Living Standard*. URL: https://encoding.spec.whatwg.org/

[Wikipedia 2026] Wikipedia contributors. *Iran-Israel War (2026)* and related articles, accessed May 2026. (Cited for context on Hormuz Strait disruption and Japanese strategic petroleum reserve releases; specific article URLs vary.) URL: https://en.wikipedia.org/wiki/

[Yano Research Institute 2025] Yano Research Institute Ltd. 住宅リフォーム市場に関する調査を実施（*2025* 年）[Survey on the Residential Renovation Market (2025)]. Press Release, 20 August 2025. URL: https://www.yano.co.jp/press-release/show/press_id/3877. Reports 2024 actual market size at ¥7.347 trillion and 2025 forecast at ¥7.3 trillion.

[Yergeau 2003] Yergeau, F. *UTF-8, a transformation format of ISO 10646*. Internet Engineering Task Force (IETF), Request for Comments: 3629, STD: 63, Standards Track, November 2003. Obsoletes: RFC 2279. URL: https://www.rfc-editor.org/rfc/rfc3629.html

---

8.1 News and Trade-Press Sources Cited Contextually

[Al Jazeera 2026] Al Jazeera. *Brent crude prices and Hormuz Strait disruption coverage*. News reports, May 2026. (Cited contextually for the Hormuz Strait crisis macroeconomic backdrop; primary URL: https://www.aljazeera.com/)

[IEA via Al Jazeera 2026] International Energy Agency, statements on global oil supply disruption, as reported by Al Jazeera, May 2026. (Cited as “the largest supply disruption in the history of the global oil market.”)

[Nikkei 2026] 信越化学、塩ビを *2* 度目の値上げ 中東情勢影響続く [Shin-Etsu chemical, second price increase for PVC, Middle East situation continues to affect]. Nihon Keizai Shimbun, 20 April 2026. URL: https://www.nikkei.com/article/DGXZQOUC180QA0Y6A410C2000000/

[Reform Online 2026] 旭化成建材、ネオマフォームなどの商品に約 *20*%の特別調整金 [Asahi Kasei Construction Materials, approximately 20%

special adjustment surcharge for Neoma Foam and other products]. Reform Online, 14 April 2026. URL: https://www.reform-online.jp/news/manufacturer/68514.php

---