ARGON-V: Deterministic State Attestation via Multi-Surface Coherence in Hostile Client Environments

Dante Ielceanu[1]
Independent Researcher

Abstract

We present ARGON-V, a protocol for verifiable state attestation in uncooperative, untrusted client environments. Traditional web-based evidence collection relies on hardware-locked Trusted Execution Environments (TEEs) or fragile, UI-dependent extraction, both susceptible to local forgery. ARGON-V introduces a five-stage pipeline that enforces **economic exhaustion** on a rational adversary by requiring semantic coherence across k independent observation surfaces. By binding ephemeral server-issued nonces (η) to deterministic DOM canonicalization, the protocol ensures that sustaining a logically consistent forgery is more costly than the expected utility of the forged state. We describe a reference implementation and analyze resilience against scripted DOM-injection and coordinated multi-surface spoofing attempts.

Keywords: State Attestation; Adversarial Security; Web Protocols; Merkle-Based Verification; Economic Security; DOM Canonicalization

Introduction

ARGON-V (pronounced "argon-five"), short for **Adversarial-Resistant Ground-truth ObservatioN — Verification**, is a deterministic verification protocol designed to extract high-confidence attestations from untrusted client environments. The protocol addresses a fundamental limitation of modern web systems: while servers can sign transmitted data, they lack visibility into how state is rendered, manipulated, or falsified at the client edge.

ARGON-V was originally developed to verify educational course completions for Oasis, but is intentionally designed as a generalizable framework for state attestation without requiring platform-side API cooperation.

Related Work and the Gap

Client-side state attestation is commonly approached through hardware-centric isolation or network-layer notarization.

Hardware-Centric Enclaves

Trusted Execution Environments (TEEs) such as Intel SGX and ARM TrustZone offer strong isolation guarantees but are often unavailable or impractical within standard consumer browser contexts. Even when supported, deployment constraints and UX friction limit TEEs for mass-market verification.

[1]Correspondence: `dante.ielceanu@gmail.com`

Network-Layer Notarization

TLSNotary-style systems and DECO provide cryptographic binding of TLS transcripts [4]. However, in modern Single Page Applications (SPAs), the transmitted data (e.g., JSON payloads) can differ substantially from the user-visible state (rendered DOM after client-side hydration and transformation). ARGON-V operates at the **application layer**, targeting the rendered state as observed, rather than only raw transport-layer artifacts.

The Adversarial Model

We define the adversary $\mathcal{A}$ as a rational actor with full control over the local execution environment. We consider three tiers of capability:

1. **Naive forgery:** manual DOM manipulation via developer tools.
2. **Scripted forgery:** automation that injects or masks DOM state on triggers.
3. **Sophisticated simulation:** a "Truman Show" attack that virtualizes the browser environment to serve coherent fraudulent states.

The safety objective of ARGON-V is:

$$\mathrm{Cost}(\mathcal{A}_{\mathrm{forgery}}) > \mathrm{Utility}(\mathcal{A}_{\mathrm{fraud}}). \tag{1}$$

Formal Protocol Specification

ARGON-V consists of five stages that map hostile observations into a signed cryptographic commitment.

Formal State Definition

We model the client-visible state σ as a directed acyclic graph over the DOM at time t. The observation process maps raw state to a fact set: $\Phi : \sigma \to F$, where F is a finite set of extracted facts. Because σ contains non-deterministic noise (session artifacts, layout shifts, ads), we define a stable subgraph $\sigma_s \subset \sigma$ representing semantically meaningful content.

A canonicalizer f_{can} aims to isolate σ_s such that:

$$\forall t_1, t_2 \in \Delta t : \ f_{can}(\sigma_{t_1}) = f_{can}(\sigma_{t_2}), \tag{2}$$

i.e., equivalent platform states map deterministically within the challenge window Δt.

Stage I: Temporal Challenge

The Verifier $\mathcal{V}$ issues an ephemeral nonce η, bound to a Page Plan $\mathcal{L}$ and a constraint set C:

$$\mathcal{V} \xrightarrow{\eta,\mathcal{L},C} \mathcal{P}. \tag{3}$$

C defines semantic invariants (e.g., "the Grade on Surface A must equal the Grade on Surface B").

Stage II: Multi-Surface Observation

The Prover traverses k surfaces and injects η into each DOM prior to serialization, producing snapshots:

$$\sigma_i = \{\mathrm{DOM}_i, \eta, \mathrm{URL}_i\}. \tag{4}$$

Client-provided timestamps may be recorded for debugging, but freshness is enforced cryptographically via nonce binding and server TTL.

Stage III: Semantic Intersection

The verifier models the extracted fact set $F(\sigma_i)$ from each surface as a point in a constraint-defined semantic space. An attestation is considered coherent only if all extracted facts satisfy the verifier's constraint set C simultaneously. Formally, we require:

$$\forall i \in \{1,\dots,k\} : \ \mathrm{Dist}(F(\sigma_i), C) = 0. \tag{5}$$

Any violation implies semantic inconsistency and results in immediate rejection.

Formalization of Semantic Invariants

Stage rests on enforcing invariants across heterogeneous representations. Let F be key-value facts extracted from a surface. In our case study, the verifier enforces a dual-surface constraint between UI-layer facts F_{ui} and telemetry-layer (shadow) facts F_{tel} when available. An attestation is valid if:

$$\forall k \in \mathrm{CoreFacts} : \ \delta(F_{\mathrm{ui}}[k], F_{\mathrm{tel}}[k]) < \epsilon, \tag{6}$$

where δ is a domain-specific distance function and ϵ tolerates non-semantic variation (e.g., whitespace, formatting).

Stage IV: Deterministic Canonicalization

A provider-specific canonicalizer f_{can} maps noisy HTML into a stable leaf set L. This stage removes non-deterministic artifacts that would otherwise invalidate hashing under benign UI changes.

Stage V: Cryptographic Finality

Leaves L are hashed into a Merkle tree:

1. **Root hash (R):** a commitment to the canonical observation.
2. **Selective disclosure:** third parties can verify individual facts via Merkle proofs without requiring raw HTML.

The final receipt is issued as:

$$\mathcal{A}_{tt} = \mathrm{Sign}_{\mathcal{V}}(R, \eta, \mathrm{SubjectID}),$$

where SubjectID is a verifier-defined identifier (optionally pseudonymous) bound to the claim.

Adversarial Resiliency Analysis

The core innovation of ARGON-V is enforced **economic friction**: forgery may be possible, but not economically rational.

The Truman Show Attack

In a simulation attack, $\mathcal{A}$ serves synchronized faked surfaces. To succeed, $\mathcal{A}$ must maintain semantic consistency across k surfaces while reacting to unpredictable η within Δt. A coarse complexity proxy is:

$$C(\mathcal{A}) \propto k \cdot \sum_{i=1}^{k} H_i, \tag{7}$$

where H_i captures the maintenance burden of accurately spoofing the canonicalized facts on surface i under platform drift.

Probabilistic Success Rates

Table 1 illustrates undetected forgery probability P_f under a simplifying independence assumption, with per-surface spoof success probability p:

$$P_f = p^k.$$

This analysis is illustrative rather than empirical. It assumes independent spoofing events across surfaces and is intended only to demonstrate scaling behavior as k increases. In practice, spoofing attempts may be correlated (e.g., shared data dependencies or a single instrumentation layer affecting multiple surfaces), in which case the effective reduction in P_f can be weaker than p^k. ARGON-V therefore prioritizes surface heterogeneity and the use of shadow facts to reduce correlation across observations.

Table 1: Illustrative forgery success probability ($P_f = p^k$) under an independence assumption.

k (Surfaces)	$p = 0.5$	$p = 0.1$	$p = 0.01$
1	0.50	0.10	0.01
3	0.125	0.001	10^{-6}
6	0.0156	10^{-6}	10^{-12}

Implementation: Coursera Case Study

We implemented a reference observer for Coursera[2].

Shadow Fact Extraction

The implementation targets *shadow facts*: structured data embedded in attributes used for telemetry (e.g., `data-click-value`). These often remain stable across UI reskins and are less likely to be spoofed by simple DOM scripts.

[2]Coursera is a registered trademark of Coursera Inc. This research is independent and has not been authorized, sponsored, or approved by Coursera Inc.

```
// Shadow Fact Extraction (illustrative)
const raw = el.getAttribute("data-click-value");
const obj = JSON.parse(raw);
const slug = obj?.product?.slug;
```

Quorum-Based Verification

A $k = 6$ page plan was selected to cross-check public verification, internal dashboards, course progress surfaces, and certificate renderers.

Engineering Trade-offs

The 60-Second Window (Δt)

We empirically tested Δt across network conditions. $\Delta t \approx 60$s was found to balance liveness and adversarial friction: long enough for scripted traversal of $k = 6$ surfaces, but too short for manual multi-surface manipulation.

Isolated World Protections

By executing in an isolated extension context, page scripts are less able to intercept the attestation logic. This preserves an "observation gap" between the page and observer within the browser.

Evaluation Methodology and Prototype Status

The design and analysis presented in this paper are grounded in a working reference prototype of the ARGON-V protocol. The prototype implements all five stages of the pipeline described in Section , including nonce issuance, multi-surface traversal, semantic invariant enforcement, deterministic canonicalization, and Merkle-based commitment generation.

At the time of writing, ARGON-V has not undergone formal experimental evaluation under controlled benchmarks or large-scale deployment conditions. No quantitative performance measurements, adversarial success rates, or longitudinal studies are reported. Instead, this work should be understood as a protocol specification and systems analysis validated through implementation correctness and adversarial reasoning.

The claims made in this paper fall into three categories:

1. **Correctness claims**, established by end-to-end execution of the prototype on real platforms and manual verification that valid states produce successful attestations while inconsistent states fail.
2. **Security claims**, derived analytically from the adversarial model defined in Section and the enforced invariants of the protocol.
3. **Liveness claims**, argued qualitatively based on observed traversal behavior of the prototype under typical network conditions, without formal latency guarantees.

Accordingly, the evaluation presented in this section is *analytical and synthetic* rather than empirical. Probability estimates, complexity proxies, and success-rate tables are illustrative and intended to characterize relative scaling behavior, not to report measured attack frequencies.

A comprehensive empirical evaluation—including adversarial automation, performance benchmarks across platforms, and user-scale deployment—is deferred to future work once the protocol specification stabilizes.

Sensitivity to Quorum Size k

Security increases with k but liveness degrades beyond a provider-dependent threshold due to latency and rate-limits. For complex Single Page Applications (SPAs), we observed diminishing practical returns as k increased beyond a moderate range. In informal prototype runs on Coursera using the reference implementation, traversal reliability degraded for larger page plans due to cumulative navigation latency, asynchronous hydration delays, and intermittent rate-limiting. While these observations were not collected under controlled experimental conditions and should not be interpreted as statistically rigorous measurements, they suggest that increasing k past a certain threshold yields marginal security benefits while disproportionately impacting liveness.

Based on these observations and architectural considerations, we treat $k \in [4, 7]$ as a practical operating range for complex SPAs. A comprehensive empirical study quantifying security gains, traversal latency, and failure rates as a function of k is deferred to future work.

Estimated Computational Overhead

We estimate client-side canonicalization overhead as approximately linear in extracted nodes, with total runtime dominated by network traversal rather than hashing or parsing. This suggests viability even on low-power devices.

Scope and Status of This Work. This document presents a *v0.1 protocol and systems design draft* for ARGON-V. It specifies the threat model, protocol stages, and security intuition, and is grounded in a working reference prototype. This paper is not an empirical security evaluation: all quantitative models, probability tables, and complexity expressions are analytical and illustrative, intended to reason about scaling behavior and adversarial cost rather than to report measured attack success rates. No controlled experiments, benchmarks, or large-scale adversarial studies have been conducted. Claims are intentionally scoped to design rationale, correctness-by-construction, and economic-security intuition, with formal proofs and empirical validation explicitly deferred to future work.

Limitations and Ethical Considerations

Local Proxy Attacks

A motivated adversary could use a TLS-terminating local proxy to modify traffic before rendering. ARGON-V raises the difficulty via Δt and multi-surface coherence, but does not claim absolute prevention.

Data Privacy

ARGON-V should follow least-privilege extraction: canonicalizers emit only facts necessary for the attestation. Persisted artifacts are minimized (e.g., store Merkle roots and signed receipts; avoid raw HTML retention).

Future Direction: Self-Healing Upgrades

Selector drift remains a practical challenge. Future work includes autonomous canonicalization using vision-language models, enabling semantic extraction resilient to UI overhauls.

Conclusion

ARGON-V transforms hostile browser sessions into auditable observations by binding multi-surface semantic coherence to an ephemeral challenge, then committing the result via Merkle hashing and verifier signatures. The protocol shifts verification from "trust the client" toward "make coherent forgery uneconomical."

References

[1] Merkle, R. C. (1980). Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*.

[2] Jones, M., Bradley, J., & Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519.

[3] Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

[4] Zhang, F., et al. (2020). DECO: Liberating web data using decentralized oracles. In *Proceedings of CCS '20*.

[5] Google Developers. (2026). Chrome Extension Manifest V3 Documentation.

[6] Coursera Inc. (2026). Coursera Verified Certificates. `https://www.coursera.org`.