DOM-Based Automatic Signature Detection and Verification

Ramprasad Anandam Gaddam
Vouch Protocol Project
https://vouch-protocol.com
ram@vouch-protocol.com

January 2026

Abstract

This paper presents a browser-based method for automatically detecting and verifying cryptographic signatures embedded in web content. The system, termed "Smart Scan," traverses the Document Object Model (DOM) to locate signature blocks, extracts signed content and detached signatures, performs cryptographic verification, and displays verification results inline. This enables ambient verification—users can observe at a glance which content on a page has been cryptographically attested without manual verification steps. We specify the detection algorithm, define the signature embedding format, and analyze privacy properties.

This paper is part of the Vouch Protocol Defensive Disclosure Series. Full index: https://vouch-protocol.com/docs/disclosures/

Keywords: browser extension, DOM traversal, signature verification, web security, ambient verification

1 Introduction

The proliferation of AI-generated content has intensified the need for content authenticity verification. Cryptographic signatures can provide such verification, but existing approaches require users to manually copy content into verification tools. This friction reduces adoption.

We present Smart Scan, a browser extension that automates signature detection and verification. Upon page load, the extension:

1. Scans the DOM for signature markers
2. Extracts signed content and signatures
3. Performs local cryptographic verification
4. Displays inline verification badges

1.1 Contributions

- Specification of in-content signature embedding format
- DOM traversal algorithm for signature detection
- Privacy-preserving local verification architecture
- Browser extension reference implementation

2 Background

2.1 Document Object Model

Definition 1 (DOM). *The Document Object Model represents an HTML document as a tree of nodes, where each element, attribute, and text segment is a node accessible via standard APIs.*

Browser extensions can traverse and modify the DOM, enabling content analysis and augmentation.

2.2 Detached Signatures

Definition 2 (Detached Signature). *A detached signature σ is computed over content c but stored separately:*

$$\sigma = \mathsf{Sign}(sk, c), \quad \textit{stored independently of } c \tag{1}$$

Unlike embedded signatures, detached signatures allow content to remain human-readable while maintaining verifiability.

3 Signature Embedding Format

3.1 Marker Syntax

Signed content is delimited with explicit markers:

```
--- vouch:start ---
[signed content]
--- vouch:signature ---
[base64-encoded JWS]
--- vouch:end ---
```


3.2 Format Specification

Definition 3 (Vouch Block). *A Vouch block $B = (m_s, c, m_\sigma, \sigma, m_e)$ consists of:*

- m_s*: Start marker (*`-- vouch:start --`*)*
- c*: Signed content (arbitrary text)*
- m_σ*: Signature delimiter (*`-- vouch:signature --`*)*
- σ*: Base64-encoded JWS*
- m_e*: End marker (*`-- vouch:end --`*)*

4 Detection Algorithm

4.1 DOM Traversal

The extension performs the following on page load:

1. Query: `document.body.innerText.matchAll(/-- vouch:start --/g)`
2. For each match, extract block boundaries
3. Parse content between m_s and m_σ
4. Parse signature between m_σ and m_e
5. Queue for verification

4.2 Verification Process

For each detected block:

1. Decode JWS to extract header and payload
2. Extract issuer DID from `iss` claim
3. Resolve DID to obtain public key pk
4. Verify: $\mathsf{Ed25519.Verify}(pk, c, \sigma) \stackrel{?}{=} 1$
5. Generate verification result

4.3 Result Injection

Verification results are displayed by injecting DOM elements:

```
{
  "status": "verified",
  "signer": "did:web:alice.example.com",
  "verified_at": "2026-01-10T12:00:00Z"
}
```

Visual indicators:

- Green checkmark: Signature valid
- Red X: Signature invalid or content modified
- Yellow warning: Verification failed (network error)

5 Privacy Analysis

Property 1 (Local Verification)**.** *All cryptographic operations occur within the browser. Content is never transmitted to external servers.*

Property 2 (Minimal Network Activity)**.** *The only network requests are DID resolution to fetch public keys. These requests reveal that verification is occurring but not the content being verified.*

Mitigation for DID resolution privacy: Cache DID documents locally with configurable TTL.

6 Security Considerations

Marker Injection: Malicious pages could inject fake markers with invalid signatures. The extension must clearly distinguish verification failure from verification success.

Content Modification After Signing: Any modification between m_s and m_σ invalidates the signature. This is the intended behavior.

7 Related Work

DKIM for Email provides header-based email authentication but does not support arbitrary web content.

Signed HTTP Exchanges (SXG) provide origin-signed web bundles but require server-side implementation.

Content Authenticity Initiative (CAI) focuses on media provenance; our approach is content-agnostic.

8 Implementation

Reference implementation available at https://github.com/vouch-protocol/vouch.

Browser extension available for:

- Chrome (Manifest V3)
- Firefox
- Edge

9 Conclusion

This paper presented Smart Scan, a browser-based system for automatic signature detection and verification. By embedding signatures within content using a simple marker format, and providing ambient verification through a browser extension, we reduce the friction of content authenticity verification. The system preserves privacy through local verification while providing clear visual feedback to users.

Prior Art Declaration

This document is published as a defensive prior art disclosure under the Creative Commons CC0 1.0 Universal Public Domain Dedication. The methods and systems described herein are hereby released into the public domain to prevent patent monopolization.
Any party implementing similar functionality after the publication date of this document cannot claim novelty for patent purposes.
Reference Implementation: https://github.com/vouch-protocol/vouch

Cryptographically Signed Document

Signed by: Ramprasad Anandam Gaddam (`github:rampyg`)
Verify: https://v.vouch-protocol.com/p/pad004

This document's authenticity can be verified by computing its SHA-256 hash and checking against the signature registered at the verification URL above.

References