Independent Submission
Internet-Draft
Intended status: Informational
Expires: 24 August 2026

R. T. Surampudi
Nylo Project

20 February 2026

WTX-1: Cross-Domain Context Preservation Protocol
draft-surampudi-wtx1-00

Abstract

This document defines the WTX-1 (WaiTag Transfer Protocol, version 1) for preserving pseudonymous user context across unrelated web domains without third-party cookies, browser fingerprinting, login requirements, or personal data collection. The protocol uses cryptographically generated pseudonymous identifiers (WaiTags), URL hash fragment transport, server-side verification, and DNS-based domain authorization to enable privacy-respecting cross-domain analytics.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 24 August 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

Third-party cookies have been the primary mechanism for cross-domain user identification since the 1990s. With Safari's Intelligent Tracking Prevention (ITP [ITP]) deployed in 2017, Firefox Enhanced Tracking Protection (ETP) in 2019, and Chrome's Privacy Sandbox initiative beginning in 2024, this mechanism is being eliminated across all major browsers.

When a user navigates from one domain to an unrelated domain (different effective top-level domain plus one label, or eTLD+1), analytics systems lose the ability to understand that the same visitor made both visits. This creates blind spots in patient journey analytics across healthcare providers, financial customer experience across banking portals, government service usage across agency domains, and multi-brand retail analytics.

Existing solutions have significant limitations: browser fingerprinting is ethically problematic and legally risky; login-based identity excludes anonymous visitors; first-party data sharing requires business partnerships and PII exchange; Privacy Sandbox Topics API is coarse-grained, advertising-focused, and Chrome-only.

WTX-1 addresses these limitations by defining a protocol that:

1. Preserves visitor context across unrelated domains

2. Never collects, transmits, or derives personal information

3. Works without cookies, fingerprinting, or login

4. Resists tracking by unauthorized third parties via DNS-based domain authorization

5. Degrades gracefully when consent is denied

6. Operates within GDPR, CCPA, and ePrivacy frameworks

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

WaiTag A pseudonymous identifier generated per-visitor using cryptographic randomness. Format: wai_<timestamp_b36>_<random><domain_hash>. Contains no personally identifiable information (PII).

Origin Domain The domain where the user’s session begins and the WaiTag is generated.

Destination Domain The domain the user navigates to, which receives and verifies the WaiTag via a cross-domain token.

Cross-Domain Token A time-limited, server-signed token encoding the WaiTag for transfer between domains.

DNS Authorization Domain ownership verification via DNS TXT records that authorizes a domain to participate in WTX-1 identity sharing.

Verification Server The server-side component that issues, signs, and verifies cross-domain tokens.

Anonymous Mode A degraded operating mode where no WaiTag is generated, no identity is preserved, and identity-related API calls are no-ops.

3. Protocol Overview

The WTX-1 protocol operates in seven steps across three participants: the origin domain, the destination domain, and the verification server.

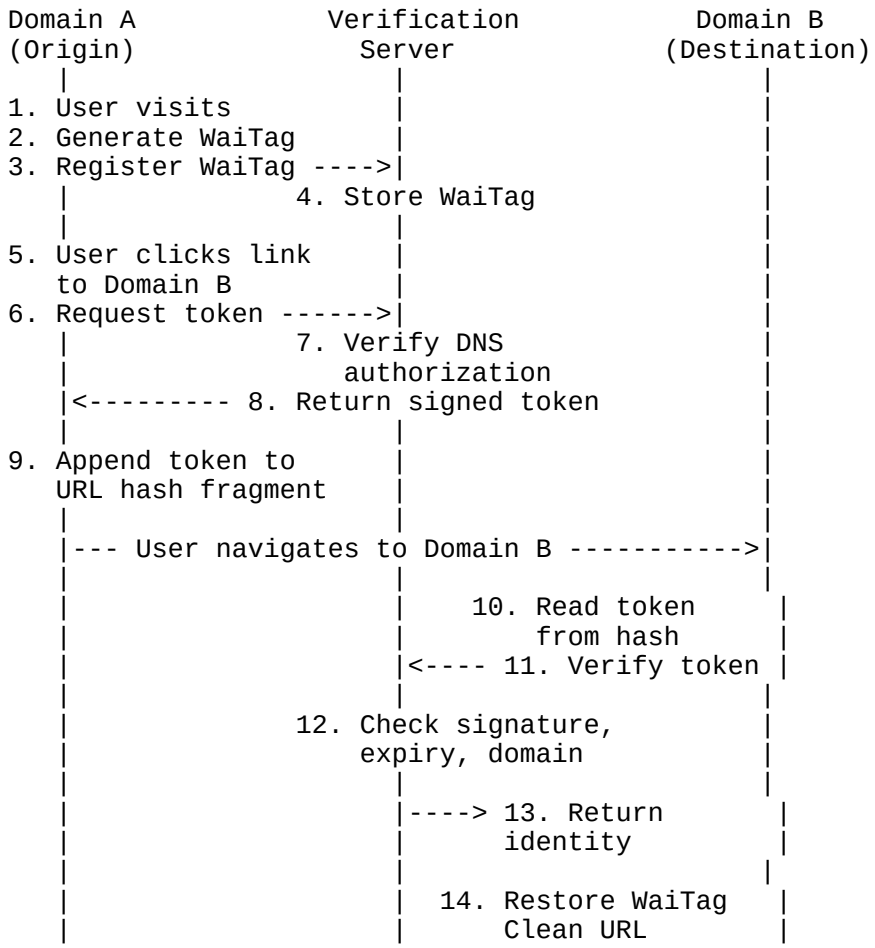

Figure 1: WTX-1 Protocol Flow

3.1. Step-by-Step

1. *WaiTag Generation:* When a user first visits a participating domain, the client generates a WaiTag using the Web Crypto API (crypto.getRandomValues). The WaiTag format is wai_<timestamp_base36>_<random_id><domain_hash>.

2. *Identity Registration:* The client registers the WaiTag with the verification server. The server stores the WaiTag, session ID, originating domain, and timestamp.

3. *Cross-Domain Navigation:* When the user navigates to another participating domain, a cross-domain token is appended to the destination URL. The mechanism for link decoration is implementation-defined (e.g., server-side link rewriting, client-side click handlers, or manual URL construction).

4. *Token Generation:* The verification server generates a signed, time-limited token encoding the user's WaiTag and session context. The server SHOULD verify that the destination domain is DNS-authorized before issuing the token.

5. *Hash Fragment Transport:* The token is appended to the destination URL as a hash fragment (#nylo_token=<token>). Hash fragments are never sent to the server in HTTP requests, providing an additional privacy layer.

6. *Token Verification:* On the destination domain, the client reads the token from the hash fragment and sends it to the verification server for validation.

7. *Identity Restoration:* If verification succeeds, the server returns the original WaiTag and session context. The client restores the user's pseudonymous identity on the new domain and removes the token from the URL.

4. WaiTag Format

4.1. Structure

A WaiTag has the following structure:

wai_<timestamp>_<random><domain_hash>

Component	Encoding	Length	Source
Prefix	ASCII	4 chars	Literal "wai_"
Timestamp	Base-36	~8 chars	Current time in milliseconds
Separator	ASCII	1 char	Literal "_"
Random ID	Base-36	11 chars	crypto.getRandomValues (8 bytes)
Domain Hash	Hex	8 chars	One-way hash of hostname

Table 1: WaiTag Components

4.2. Properties

A WaiTag has the following properties:

* Not reversible: No component can be reversed to identify a natural person.

* Not derived from PII: No personal information is used as input.

* Domain-scoped: The domain hash binds the WaiTag to its origin, but the hash is one-way and cannot reveal the domain to a third party.

* Collision-resistant: 64 bits of cryptographic randomness provides sufficient uniqueness for analytics use cases.

* Not a fingerprint: No hardware, software, or behavioral signals are used.

* Not deterministic: The same user on the same device will receive different WaiTags across sessions unless identity is explicitly restored via cross-domain token.

4.3. Generation Requirements

Implementations MUST generate WaiTags using crypto.getRandomValues [W3C-WebCryptoAPI] or an equivalent cryptographically secure pseudorandom number generator (CSPRNG). If the Web Crypto API is not available, implementations MAY fall back to Math.random, but SHOULD log a warning indicating reduced randomness quality.

5. Token Transport

5.1. Primary: URL Hash Fragment

The cross-domain token MUST be transported via URL hash fragment as the primary mechanism:

https://destination.example.com/page#nylo_token=<signed_token>

Hash fragments (the portion of a URL after "#") are processed entirely client-side per [RFC3986] Section 3.5. They are:

* Never included in HTTP requests to the server
* Never logged in server access logs
* Never sent in the Referer header
* Not visible to network intermediaries (proxies, CDNs, WAFs)

This provides a stronger privacy guarantee than URL query parameters, which are transmitted to the server and commonly logged.

5.2. Fallback: URL Query Parameters

If hash fragment transport is not feasible (e.g., the destination URL uses hash-based client-side routing), the token MAY be transported as a query parameter:

https://destination.example.com/page?nylo_token=<signed_token>

Implementations using query parameter transport SHOULD remove the token from the URL immediately after reading it, using the History API (history.replaceState) to clean the URL without triggering a page reload.

5.3. Parameter Names

Implementations MUST accept both of the following parameter names for cross-domain tokens:

* "nylo_token" (primary)
* "wai_token" (alternative)

5.4. Token Cleanup

After reading a cross-domain token from the URL, the client MUST remove it from the URL to prevent:

* Accidental sharing of token-bearing URLs
* Token replay from browser history
* Token exposure in analytics tools that capture full URLs

The client SHOULD use history.replaceState to remove the token without causing a navigation event or page reload.

6. Token Format and Verification

6.1. Token Contents

A cross-domain token encodes the following fields:

Field	Description
waiTag	The pseudonymous identifier to transfer
sessionId	The session identifier from the origin domain
originDomain	The domain that issued the token
destinationDomain	The intended recipient domain
issuedAt	UTC timestamp of token creation
expiresAt	UTC timestamp of token expiry
signature	Server-generated signature (HMAC or implementation-defined)

Table 2: Cross-Domain Token Fields

The token encoding format is implementation-defined. Common choices include JSON Web Tokens (JWT) [RFC7519] and opaque server-side tokens with a lookup key.

6.2. Verification Requirements

The verification server MUST check:

1. Signature validity: The token has not been tampered with.

2. Expiry: The token has not expired. The expiry window is configurable; the recommended default is 5 minutes.

The verification server SHOULD also check:

1. Domain authorization: The destination domain is DNS-authorized to receive identities.

2. Origin matching: The token was issued for the domain making the verification request.

6.3. Verification Endpoint

The verification endpoint accepts a POST request with Content-Type application/json containing the token, the requesting domain, a customer identifier, and the HTTP Referer value.

Request body:

```
{
  "token": "<signed_token>",
  "domain": "destination.example.com",
  "customerId": "<customer_id>",
  "referrer": "https://origin.example.com/page"
}
```

Success response:

```
{
  "success": true,
  "identity": {
    "sessionId": "<session_id>",
    "waiTag": "<wai_tag>",
    "userId": null
  }
}
```

Failure response:

```
{
  "success": false,
  "error": "TOKEN_EXPIRED"
}
```

7. DNS Domain Authorization

7.1. Purpose

Before a domain can receive cross-domain identities, it SHOULD prove ownership via a DNS TXT record. This prevents unauthorized domains from requesting or receiving WaiTags and ensures that only domains controlled by the same organization (or participating partners) can share visitor context.

7.2. TXT Record Format

The DNS TXT record uses the following format:

```
_nylo-verify.example.com. IN TXT
    "nylo-domain-verify=<verification_code>"
```

The verification code is a unique value generated by the verification server for each domain registration.

7.3. Verification Flow

1. The domain owner registers the domain with the verification server and receives a verification code.

2. The domain owner adds the TXT record to their DNS configuration.

3. The domain owner calls the verification endpoint.

4. The server performs a DNS lookup for _nylo-verify.<domain>.

5. If the TXT record matches the expected verification code, the domain is authorized.

6. Authorization is cached server-side and periodically re-verified.

7.4. Subdomain Inheritance

If a parent domain (e.g., example.com) is DNS-verified, subdomains (e.g., blog.example.com, shop.example.com) inherit authorization automatically. Implementations MAY provide configuration to exclude specific subdomains from inheritance.

8. Client-Side Storage

8.1. Three-Layer Strategy

The client SHOULD store identity data using multiple mechanisms for resilience against browser storage restrictions:

Layer	Mechanism	Persistence	ITP Impact
1	First-party cookie	24h (JS-set under ITP)	Capped at 7 days
2	localStorage	Until cleared	Partitioned by eTLD+1
3	sessionStorage	Tab lifetime	Unaffected

Table 3: Storage Layers

8.2. Storage Read Priority

When restoring identity, the client reads in order: cookie, localStorage, sessionStorage. The first valid result is used.

8.3. Data Stored

The stored identity record contains:

```
{
  "sessionId": "<secure_id>",
  "waiTag": "<wai_tag>",
  "userId": null,
  "domain": "example.com",
  "createdAt": "2026-02-20T12:00:00.000Z",
  "integrity": "<hash>"
}
```

The "integrity" field is a one-way hash of the session ID, domain, WaiTag, and customer ID, used to detect tampering.

8.4. Data Obfuscation

Data stored in localStorage SHOULD be encoded with a customer-specific salt and timestamp to prevent casual inspection. This is obfuscation, not cryptographic security. The stored data contains no PII regardless of encoding.

9. Consent and Anonymous Mode

9.1. Consent API

Implementations MUST provide a consent API that integrates with Consent Management Platforms (CMPs). The API MUST support the following operations:

* setConsent({analytics: false}) - Deny consent, switch to anonymous mode
* setConsent({analytics: true}) - Grant consent, restore identity tracking
* getConsent() - Query current consent state, returns {analytics: boolean}

9.2. Anonymous Mode Behavior

When consent is denied, the client MUST operate in anonymous mode with the following behavior:

* Event tracking remains active but with no identity attached.
* WaiTag generation is disabled; waiTag is null.
* The identify() API call is a no-op and is silently ignored.
* getSession() returns null for waiTag and userId fields.
* Cross-domain identity is disabled; no tokens are generated or accepted.
* Only a session-scoped anonymous identifier is used (format: anon_<timestamp>_<random>).
* No persistent identity data is written to cookies, localStorage, or sessionStorage.

9.3. Consent Restoration

When consent is granted after starting in anonymous mode:

1. The client checks for previously stored identity data.
2. If found, the existing WaiTag and session are restored.
3. If not found, a new WaiTag is generated.
4. Cross-domain token checking is activated.
5. Full identity tracking resumes.

9.4. Regulatory Considerations

WaiTags are designed to fall outside the GDPR [GDPR] definition of "personal data" as defined in Article 4(1) of Regulation (EU) 2016/679 because they contain no information derived from a natural person, cannot be reversed or cross-referenced to identify someone, and are not deterministic across sessions.

However, the classification of pseudonymous identifiers under data protection law varies by jurisdiction and is subject to evolving regulatory interpretation. Implementers SHOULD consult local legal counsel. The anonymous mode provides a conservative fallback for jurisdictions where pseudonymous identifiers may be considered personal data.

10. Security Considerations

10.1. Token Security

* Tokens MUST be signed server-side using HMAC [RFC2104] or an equivalent mechanism.
* Tokens MUST have a configurable expiry window. The recommended default is 5 minutes.
* Tokens SHOULD be single-use where feasible to prevent replay attacks.
* Token verification MUST validate the signature and expiry.

10.2. Transport Security

* All API communication between the client and the verification server MUST use HTTPS [RFC2818].

* Hash fragment transport prevents server-side token logging by design, as fragments are not included in HTTP requests per [RFC3986].

* When query parameter fallback is used, the client MUST clean the token from the URL after reading.

10.3. Input Validation

* All client-supplied strings MUST be sanitized to prevent cross-site scripting (XSS) attacks. HTML entity encoding is RECOMMENDED.

* String fields MUST be length-limited. The recommended maximum is 1,000 characters.

* Domain values MUST be validated before use in DNS lookups or token generation.

10.4. Rate Limiting

* API endpoints MUST implement rate limiting. The recommended default is 100 requests per IP address per minute.

* Rate limit status MUST be communicated via X-RateLimit-Limit, X-RateLimit-Remaining, and X-RateLimit-Reset response headers.

* Exceeded limits MUST return HTTP 429 (Too Many Requests).

10.5. Response Headers

Servers implementing WTX-1 SHOULD set the following security-related response headers:

* X-Content-Type-Options: nosniff

* X-Frame-Options: SAMEORIGIN

* Referrer-Policy: strict-origin-when-cross-origin

* Permissions-Policy: camera=(), microphone=(), geolocation=()

* Content-Security-Policy: Appropriate policy for the deployment

* Strict-Transport-Security: max-age=31536000; includeSubDomains (when serving over HTTPS)

11. IANA Considerations

This document has no IANA actions.

If this protocol achieves broader adoption, the following registrations may be requested in future revisions:

* Registration of the "_nylo-verify" DNS underscore name prefix in the "Underscored and Globally Scoped DNS Node Names" registry per RFC 8552.

* Registration of the "nylo_token" and "wai_token" URI fragment parameter names if a suitable registry is established.

12. Privacy Considerations

WTX-1 is designed from the ground up to preserve user privacy while enabling cross-domain analytics. This section details the privacy properties, guarantees, and limitations of the protocol.

12.1. Core Privacy Commitments

1. No PII collection: The protocol never collects, transmits, or derives personal information such as names, email addresses, phone numbers, or physical addresses.

2. No fingerprinting: No hardware, software, or behavioral signals (screen resolution, installed fonts, GPU rendering, etc.) are used to generate or correlate identifiers.

3. No third-party cookies: Cross-domain identity does not use the Set-Cookie/Cookie HTTP mechanism for cross-domain transfer.

4. No login required: Identity preservation works for anonymous visitors without requiring authentication or account creation.

5. Consent-respecting: The protocol degrades to fully anonymous tracking when consent is denied, ensuring compliance with opt-out requirements.

6. Domain-authorized: Only DNS-verified domains can participate in identity sharing, preventing unauthorized cross-domain tracking.

7. Time-limited tokens: Cross-domain tokens expire (default 5 minutes), preventing indefinite tracking or token reuse.

8. Client-side token reading: Hash fragment transport ensures tokens are never sent to destination servers in HTTP requests. The token is subsequently sent to the verification server via a dedicated API call for validation.

12.2. Pseudonymous Identifiers

WaiTags are pseudonymous identifiers generated using cryptographically secure random number generators (window.crypto.getRandomValues). They contain no embedded personal information, device characteristics, or derivable real-world identity. A WaiTag cannot be reversed to identify a natural person without access to a separate mapping table, which the protocol does not define or require.

Under GDPR, pseudonymous identifiers are still considered personal data when they can be attributed to a natural person by using additional information. Implementors MUST ensure that any additional information that could link a WaiTag to a natural person is kept separately and subject to appropriate technical and organizational measures.

12.3. User Control and Consent

Implementations SHOULD provide users with clear mechanisms to:

1. Opt out of cross-domain identity sharing while retaining single-domain analytics.

2. Clear their WaiTag and session data at any time.

3. View what data has been collected about their pseudonymous identity.

4. Request deletion of all data associated with their WaiTag.

When a user denies consent, the protocol MUST degrade gracefully to anonymous mode where no cross-domain tokens are generated or accepted, and only aggregate, non-identifiable analytics are collected.

12.4. Data Minimization

The protocol follows the principle of data minimization as required by GDPR Article 5(1)(c). Cross-domain tokens contain only the minimum fields necessary for identity verification: a WaiTag, session identifier, optional user identifier, source domain, expiration timestamp, and cryptographic signature. No behavioral data, page content, or interaction history is included in the token.

12.5. Cross-Domain Tracking Scope

Cross-domain identity sharing is limited to domains that have been explicitly authorized via DNS TXT records. This prevents a domain from being included in a tracking network without the domain owner's explicit consent. The DNS verification mechanism ensures that only organizations with administrative control over a domain can authorize it for cross-domain identity sharing.

Implementations MUST NOT allow wildcard domain authorization or open enrollment of domains into a tracking network.

12.6. Regulatory Compliance

WTX-1 is designed to operate within the following regulatory frameworks:

* GDPR (EU): Pseudonymous processing with data minimization, purpose limitation, and right to erasure support.

* CCPA/CPRA (California): No sale of personal information; supports opt-out mechanisms.

* ePrivacy Directive (EU): No use of cookies or similar technologies for cross-domain transfer; hash fragment transport does not trigger cookie consent requirements.

* HIPAA (US Healthcare): No collection or transmission of Protected Health Information (PHI).

* COPPA (US Children): Protocol does not collect age information; implementors in child-directed services MUST implement additional safeguards.

Implementors are responsible for ensuring their specific deployment complies with applicable local regulations, as the protocol alone does not guarantee regulatory compliance.

12.7. Limitations and Residual Risks

While WTX-1 provides strong privacy guarantees, the following residual risks should be acknowledged:

* Server-side correlation: Organizations operating multiple domains may correlate WaiTags server-side. The protocol limits this to DNS-authorized domains but cannot prevent the authorizing organization from performing correlation.

* Token interception: Although hash fragment transport prevents tokens from being sent in HTTP requests, tokens may be visible to browser extensions or client-side JavaScript on the destination page before cleanup occurs.

* Referrer leakage: Some browsers may include hash fragments in Referer headers under certain conditions. Implementations SHOULD set Referrer-Policy headers to mitigate this risk.

* Long-lived sessions: If WaiTags are stored in localStorage without expiration, they persist indefinitely. Implementations SHOULD implement WaiTag rotation or expiration policies.

13. References

13.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[W3C-WebCryptoAPI] W3C, "Web Cryptography API", https://www.w3.org/TR/WebCryptoAPI/, W3C Recommendation , January 2017.

[GDPR] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)", https://eur-lex.europa.eu/eli/reg/2016/679/oj, April 2016.

[ITP] Apple WebKit, "Intelligent Tracking Prevention", https://webkit.org/blog/7675/intelligent-tracking-prevention/, June 2017.

Comparison with Existing Approaches

Property	3P Cookies	Fingerprinting	Login-Based	WTX-1
Cross eTLD+1	Yes (deprecated)	Yes	Yes	Yes
Requires login	No	No	Yes	No
Collects PII	Varies	Yes	Yes	No
Browser support	Declining	Declining	Universal	Universal
Consent required	Yes	Yes	Varies	Recommended
Works on Safari	No (ITP)	Partially	Yes	Yes (degraded)

Table 4: Cross-Domain Identity Approaches

Acknowledgments

WTX-1 was developed as part of the Nylo project to address the analytics gap created by the deprecation of third-party cookies, with a focus on privacy-respecting solutions for regulated industries including healthcare, finance, and government.

Author’s Address

Ravi Teja Surampudi
Nylo Project
Email: ravisurampudi@outlook.com
URI: https://github.com/tejasgit/nylo