

Authenticated Browsing: Reconciling Application-Layer Flooding Defense with Data Confidentiality

FELIX STÖGER, ETH Zurich, Switzerland

JELTE VAN BOMMEL, ETH Zurich, Switzerland

JONGHOON KWON, ETH Zurich, Switzerland

PATRICK BAMERT, Zürcher Kantonalbank, Switzerland

STEFANIE BOSS, University of Amsterdam, Netherlands

ALEXANDER EGGERTH, ETH Zurich, Switzerland

ADRIAN PERRIG, ETH Zurich, Switzerland

Organizations in regulated sectors, including financial institutions subject to banking secrecy, healthcare providers handling patient data, and government services processing citizen information, face a structural obstacle in deploying effective denial-of-service (DoS) defenses against application-layer flooding: the most capable commercial solutions require third-party TLS termination, which conflicts with their data protection obligations. We systematically analyze existing DoS mitigation approaches, both industry-deployed and academic, and show that none simultaneously achieve application-layer traffic differentiation and compatibility with stringent confidentiality requirements at line rate, without per-connection state or network changes. We present *Authenticated Browsing*, a system that separates traffic differentiation from traffic inspection by making prior application-layer authentication verifiable at the packet level. Authenticated Browsing repurposes widely deployed identity infrastructure for DoS mitigation: after a user is verified as enrolled with the protected service, an Identity Provider (IdP) issues short-lived cryptographic credentials from which a user agent derives per-packet authentication tags. Forward-deployed validators verify these tags at line rate using only symmetric cryptography, without inspecting payload content or learning user identities. The IdP's role is limited to credential issuance and need not coincide with the application's login system. We implement Authenticated Browsing and evaluate it at scale. A single validator sustains more than 70 million authenticated packets per second on commodity hardware, achieving line rates of 160 Gbps. Internet measurements show that Authenticated Browsing authentication headers are preserved across a substantial fraction of real-world paths. These results show that application-layer flooding attacks can be mitigated under strong confidentiality requirements.

CCS Concepts: • **Networks** → **Denial-of-service attacks**; • **Security and privacy** → *Authentication*; *Privacy protections*.

Additional Key Words and Phrases: Denial-of-service defense, application-layer flooding, packet authentication, federated identity, privacy-preserving mitigation, data confidentiality, GDPR

ACM Reference Format:

Felix Stöger, Jelte van Bommel, Jonghoon Kwon, Patrick Bamert, Stefanie Boss, Alexander Eggerth, and Adrian Perrig. 2026. Authenticated Browsing: Reconciling Application-Layer Flooding Defense with Data Confidentiality. *Proc. ACM Netw.* 4, CoNEXT4, Article 46 (December 2026), 25 pages. <https://doi.org/10.1145/3856957>

Authors' Contact Information: Felix Stöger, ETH Zurich, Zurich, Switzerland, felix.stoeger@inf.ethz.ch; Jelte van Bommel, ETH Zurich, Zurich, Switzerland, jelte.vanbommel@inf.ethz.ch; Jonghoon Kwon, ETH Zurich, Zurich, Switzerland, jong.kwon@inf.ethz.ch; Patrick Bamert, Zürcher Kantonalbank, Zurich, Switzerland, patrick.bamert@zkb.ch; Stefanie Boss, University of Amsterdam, Amsterdam, Netherlands, s.boss@uva.nl; Alexander Eggerth, ETH Zurich, Zurich, Switzerland, aeggerth@ethz.ch; Adrian Perrig, ETH Zurich, Zurich, Switzerland, adrian.perrig@inf.ethz.ch.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2834-5509/2026/12-ART46

<https://doi.org/10.1145/3856957>

1 Introduction

Financial institutions, healthcare providers, and government services share a common challenge: they must defend against application-layer flooding attacks while keeping application payloads within the confidentiality boundaries imposed by laws such as banking secrecy statutes [44], medical confidentiality rules [1, 19], and the GDPR [20].

The challenge is architectural: *application-layer flooding attacks* send a high volume of well-formed HTTP(S) requests from hosts that complete TCP handshakes and establish valid TLS sessions. Header-based filtering therefore cannot distinguish attack requests from legitimate ones. Flooding differs from asymmetric application-layer DoS attacks, which exploit expensive application operations at low request rates and are detected by application-level defenses [15]; this paper focuses on flooding. Commercial defenses that can classify flooding requests, including CDN/WAF-based mitigation services, terminate TLS to inspect request content. TLS termination grants the defense provider access to decrypted application-layer traffic and pulls it into the regulated processing chain, where it must satisfy the confidentiality, audit, and data-handling obligations attached to that role. Where institutional or statutory confidentiality requirements prohibit such disclosure, these mitigations cannot be deployed at all. Increasingly large HTTP flooding attacks [11] exacerbate the trade-off: the services that most need request-aware filtering are often the ones least able to use it.

Section 3 surveys the DoS defense landscape against this constraint. Defenses below the application layer cannot distinguish semantically legitimate requests from attack requests; defenses that can generally require plaintext visibility. The two properties are coupled in deployed defenses, but they need not be: deciding *whether a packet comes from a legitimate user* is a different question from *what that packet contains*. Decoupling them opens the design space for systems that authenticate users without inspecting their traffic.

A useful place to look for such a decoupling is the browser itself. Users of regulated web services already authenticate through federated Identity Provider (IdP) infrastructure such as commercial single sign-on services or governmental eID platforms; the application-layer machinery for user identification already exists and is trusted by users and operators. What is missing is a way to carry the result of that authentication down into the network, so that the validity of a packet can be cryptographically checked by a third party without requiring access to plaintext application-layer data or user identities.

Motivated by this gap, we developed *Authenticated Browsing (AB)* in collaboration with a major Swiss systemically important financial institution whose regulatory obligations prohibit third-party access to plaintext customer traffic. AB extends federated identity infrastructure to provision short-lived cryptographic credentials at the start of a user's browsing session; their browser, via a local user agent, then derives a per-packet authentication tag and attaches it to every outgoing packet. Forward-deployed *validators*, deployed at Internet exchange points or scrubbing centers, verify these tags using symmetric cryptography, without terminating TLS or inspecting payload content. Under attack, operators reserve capacity for authenticated browser traffic and rate-limit or drop the rest. The result is application-layer user identification combined with network-layer verification: a single validator sustains more than 70 million packets per second at line rates of 160 Gbps on commodity hardware. Unlike capability-based systems, AB credentials derive from the user's pre-existing IdP enrollment and are issued without involving the application under attack.

This paper introduces the following contributions:

- (1) We introduce AB, a packet-level authentication architecture for organizations that cannot expose plaintext traffic to a third-party mitigator. AB's core insight is to lift federated user authentication from the application layer into per-packet verification at the network layer, separating traffic differentiation from traffic inspection (Section 4).

- (2) We design the cryptographic mechanisms that make stateless, line-rate validation compatible with these confidentiality constraints: short-lived packet credentials, per-application and per-user key derivation, rotating pseudonymous user identifiers, and symmetric-only verification. Validators authenticate packets without learning user identities, while the protected service retains the information needed for accountability and revocation (Section 4).
- (3) We implement and evaluate AB end-to-end. Our validator sustains over 70 million authenticated validations per second on commodity hardware, and Internet measurements show that authenticated browser traffic traverses up to 86% of measured paths intact (Sections 6 and 7).

2 Design Goals and Threat Model

In this section, we formalize the design goals and threat model for DoS mitigation in environments where application-layer traffic differentiation must be achieved without exposing plaintext traffic to the filtering infrastructure. Section 3 evaluates existing approaches against these requirements and shows that none meets all six design goals. AB builds on OAuth 2.0/OIDC [23, 49], where the authorization server acts as an Identity Provider (IdP) that issues identity assertions alongside access tokens; this infrastructure already operates at web scale through commercial Single Sign-On (SSO) services and governmental eID platforms.

2.1 Design Goals

We formalize six design goals, which reflect the legal constraints and technical requirements of a Denial of Service (DoS) mitigation system operating in a highly regulated sector:

- G1: No plaintext exposure to filtering infrastructure.** Filtering infrastructure authenticates packets without access to decrypted application-layer traffic.
- G2: Differentiated service under attack.** During DoS attacks against an AB-protected service, validators reserve a configured share of forwarding capacity for authenticated traffic; unauthenticated traffic can be deprioritized or dropped when capacity is scarce.
- G3: Accountability without disclosure.** Misbehaving user accounts are identifiable and revocable without revealing the user's true identity to filtering infrastructure. The protected resource can still identify any misbehaving user.
- G4: No network changes.** The system requires no router or network-equipment modifications.
- G5: Line-rate validation.** Packets are authenticated at line rate without per-connection state.
- G6: Resilience.** The system continues authenticating users while at least one linked IdP and one validator remain reachable.

2.2 Threat Model

Adversary Capabilities. We consider an adversary whose goal is to deny service to authenticated users of an application using AB. The adversary controls a botnet, can spoof source-IPs, and launches volumetric floods, protocol-level state exhaustion attacks (e.g., SYN floods, TLS renegotiation), and application-layer flooding attacks using well-formed requests.

Compromised Accounts. We assume IdPs are trusted and secure entities; a breach of an IdP would have implications far beyond AB. However, the adversary may compromise a fraction of accounts at the IdP through credential stuffing, phishing, or similar attacks. The packet flow toward a validator therefore comprises three categories: (i) unauthenticated packets, (ii) authenticated packets from benign users, and (iii) authenticated packets from compromised accounts.

Sybil Resistance. We assume IdPs prohibit the creation of large numbers of bot accounts through strong verification, such as multi-factor authentication or know your customer processes. AB is not intended for environments where IdPs permit weakly verified account creation.

Curious Validators. Validators may be operated by third parties such as DoS mitigation providers, and may be *honest-but-curious*: they follow the protocol but attempt to learn identities or track users across sessions. AB must ensure that validators can verify packet authenticity without learning which user generated a packet or linking packets across key bundle epochs.

Compromised User Agents. An adversary who compromises a user's device gains access to the user's current key bundle and can generate authenticated packets for the remainder of the bundle lifetime; this is equivalent to one compromised account, and is bounded by the same revocation.

Out of Scope. We do not claim protection against volumetric attacks that saturate links upstream of deployed validators; such attacks must be handled by existing DoS mitigations. Nor do we claim that AB by itself is enough to mitigate general application-layer DoS attacks (Section 8.4). Instead, we discuss how AB can enhance existing mitigations against application-layer DoS in Section 8.1. We also do not address side-channel attacks which use metadata, such as packet timing and size patterns, to perform traffic analysis, since they are orthogonal to the privacy and authentication guarantees AB provides. Finally, we assume the KDF and MAC are secure PRFs, and that the keyed construction used for the rotating user identifier *ruid* is pseudorandom for parties not holding K_a^{id} .

3 Existing Approaches and Their Limitations

Table 1. Evaluation of DoS mitigation approaches against design goals (formalized in Section 2).

✓ = satisfied, ~ = partially satisfied, × = not satisfied.

Approach	G1 (No Plaintext.)	G2 (Diff. Srv.)	G3 (Acc./Discl.)	G4 (No Infra.)	G5 (Perf.)	G6 (Resil.)
BGP-Based Filtering & Diversion	✓	×	×	✓	✓	✓
L3/L4 Filtering	✓	~	×	✓	✓	~
Rate Limiting & Admission Control	✓	~	×	✓	~	~
In-Path Authenticated Middleboxes	~	✓	~	✓	~	~
· MiddlePolice	~	✓	~	✓	~	✓
TLS-Terminating App-Layer Defenses	×	✓	×	✓	~	~
· PreAcher	×	~	~	✓	~	✓
Network Capabilities	✓	✓	×	×	~	×
Transport- & Network-Layer Auth.	~	~	~	✓	~	✓
Anonymous Authentication	✓	~	✓	✓	×	~
Authenticated Browsing	✓	✓	✓	✓	✓	✓

Existing DoS defenses span the protocol stack and make distinct trade-offs between visibility, deployability, and filtering precision. Under the goals of Section 2, however, none of the approaches we surveyed provides the combination required by web services in regulated sectors: metadata-based defenses preserve payload confidentiality but lack application semantics; TLS-terminating defenses provide request-level control but expose plaintext; and cryptographic or capability-based alternatives introduce deployment, state, trust, or performance costs. We survey these trade-offs in Table 1, grouping approaches by the information they use to classify traffic.

3.1 Metadata-Based Network and Transport Defenses

These defenses classify traffic using signals available without decrypting application traffic, such as routing state, packet headers, timing and rate information, and resource pressure. This makes them compatible with payload confidentiality and effective against volumetric floods, spoofing,

malformed packets, and some concentrated-source attacks. Their limitation is that these signals do not capture application semantics: they cannot reliably distinguish legitimate encrypted requests from abusive requests with the same network-level shape.

BGP-Based Filtering and Diversion. Networks utilize BGP to (i) null-route prefixes [29], (ii) push header-based filters to upstream ASes [38], or (iii) divert traffic to scrubbing centers [26] that apply the data-plane filters described below. These mechanisms react quickly to volumetric floods, but their prefix- and header-level signatures are too coarse for per-packet differentiation of well-formed traffic. A further line of work has investigated rerouting-based defenses, which use BGP to steer the traffic of selected source ASes around flooded links [30, 51]. Like the mechanisms above, they differentiate traffic only coarsely, and their effectiveness further depends on the routing behavior of ASes outside the defender's control [50, 56].

L3/L4 Packet Filtering. Many attack classes can be identified from network and transport headers alone. TCP SYN cookies [16] defer state allocation until handshake completion, uRPF rejects spoofed source addresses, and header sanity checks discard malformed packets. Recent work moves such defenses onto programmable switches [2, 37, 61], sustaining filtering at line rates of up to 380 Gbps [37].

Rate Limiting and Admission Control. Token-bucket policing keyed on source identifiers, combined with origin-side load shedding, contains attacks that concentrate load on few sources, and operates entirely on packet headers. Large distributed botnets weaken this approach: each bot can remain below its individual threshold while aggregate traffic exceeds the target's capacity. FineLame [15] detects asymmetric application-layer DoS attacks, in which individual requests trigger expensive operations at the origin, a threat distinct from application-layer flooding with large volumes of well-formed requests.

3.2 In-Path Authenticated Middleboxes and TLS-Terminating Defenses

In-Path Authenticated Middleboxes. A separate line of work designs middleboxes that filter traffic with destination-defined or cryptographically scoped authority. MiddlePolice [36] delegates per-sender rate limiting to cloud middleboxes that issue capabilities and enforce destination-defined policies, but its HTTPS source-validation mechanism either requires TLS termination at the middlebox, or pushes validation back to the protected application, leaving the capability-request path susceptible to denial-of-capability attacks. Umbrella [35] places middleboxes that enforce congestion accountability into ISPs. SENS [46] signals upstream filtering requests across ASes but operates only at the network and transport layer. mcTLS [42] scopes TLS so middleboxes decrypt only granted segments, but any middlebox needing application-layer visibility still holds the relevant read keys, so the confidentiality trade-off remains; it also requires TLS-stack changes at both endpoints.

TLS-Terminating Application-Layer Defenses. Among widely deployed defenses, CDN/WAF proxies provide the most direct request-level controls against HTTP(S) floods and expensive-endpoint attacks. These services terminate TLS and inspect the decrypted traffic, both to identify potential attacks, for example through attack signatures [12], and to apply bot challenges (JavaScript, CAPTCHA, proof-of-work), WAF rules, and reputation filtering; CDN caching of static content additionally absorbs flash crowds. The price is direct visibility of unencrypted traffic by the defense provider. DynaShield [62] has the server issue capabilities as HTTP cookies, which cloud proxies verify after terminating TLS with the server's private key. PreAcher [32] narrows this exposure for user passwords, while mitigating DoS attacks targeting expensive password hashing functions.

OPRF-based pre-authentication is deployed at CDN edges, allowing the CDN to filter credential-stuffing attempts without learning user passwords; the CDN remains on the traffic path, and the technique does not generalize to arbitrary application-layer attacks.

3.3 Network Capabilities and Cryptographic Authentication

Network Capabilities. A substantial body of work explores network capabilities as a DoS defense. SIFF [59] and TVA [60] use capabilities that senders request from receivers and routers verify in-network; NetFence [34] adds network-wide per-sender fairness; and Portcullis [45] addresses denial-of-capability attacks at the request channel itself. Passport [33] uses MAC-based packet authentication for source-AS validation and traceback. OPT [28] lets routers authenticate the source and path of each packet using MAC keys re-derived on the fly from a local secret. Because any host can obtain such keys, a valid tag attests only the origin of a packet in the network rather than the identity of the user who sent it. In APIP [41], a trusted delegate vouches for senders and keeps their identities from verifiers; verifiers query the delegate for each new flow, and only the delegate can identify a misbehaving sender. In path-aware networks like SCION [4], reservation systems such as SIBRA [5] and Hummingbird [57] offer inter-domain bandwidth reservations that shield reserved traffic from floods, DoCile [58] protects the setup of AS-to-AS reservations from denial-of-capability attacks, and EPIC [31] authenticates the source and path of every packet with keys that any host can obtain. These systems preserve confidentiality and differentiate traffic in the network, but require new in-network functionality or a new architecture and identify hosts or ASes rather than enrolled users; capability systems on today's Internet also remain exposed to denial-of-capability attacks, at least during initial setup.

Transport- and Network-Layer Authentication. Mutual TLS (mTLS) [13, 48] authenticates the client only to the endpoint that terminates TLS: subsequent records are protected with keys that only the two endpoints hold, and TLS 1.3 encrypts the client certificate. TLS-SRP [52] likewise authenticates the client only to the TLS endpoint, which must hold a password verifier for every user, and sends the username in the clear. Both can thus be checked only where TLS terminates, exposing plaintext and user identities to the filter. IPsec AH [27] authenticates every packet without decryption, but a filter would need the key and replay window of every security association. TCP-AO [55] authenticates each segment with keys that are bound to socket pairs and derived from each connection's initial sequence numbers, so a filter needs per-connection state and can attribute segments only to connection endpoints.

Anonymous Authentication. A natural direction to achieve authentication without identity disclosure is anonymous cryptographic authentication: group signatures [8], BBS+ credentials [3], and related schemes [9]. Existing constructions rely on public-key operations and often zero-knowledge proofs, making verification substantially more expensive than symmetric-key MACs. In our per-packet validator model, these costs are incompatible with the line-rate target and would expose a denial-of-service vector by letting adversaries trigger expensive verifications at scale.

3.4 Summary

Table 1 consolidates the analysis. The surveyed approaches expose a three-way gap: none simultaneously provides (i) application-level differentiation of traffic from enrolled users, (ii) line-rate filtering by the mitigation provider without plaintext access, per-client or per-connection state, or user identities, and (iii) deployment without changes to Internet routers or endpoint TLS stacks. AB is designed to provide this combination, while leaving one secondary trade-off: users without pre-established credentials remain in the unauthenticated traffic class during an ongoing attack. In turn, AB depends on trusted and available IdPs (Section 8.4), which are off the packet path

and issue short-lived credentials; if one becomes unavailable, users linked only to it fall back to unauthenticated service, and if one is compromised, its bundles admit attack traffic until validators reject its identifier. By comparison, a TLS-terminating provider is on the packet path and sees plaintext, capability systems depend on modified routers and an available request channel, and APIP verifiers must reach the delegate for each new flow; mTLS needs no online issuer, but a compromised CA can likewise issue client certificates, and timely revocation needs status checks.

4 System Design

This section specifies the design and operations of AB, a system that leverages existing Identity Provider (IdP) infrastructure to enable packet-level authentication for DoS mitigation. By decoupling user authentication from traffic inspection, AB allows in-network validators to prioritize legitimate traffic without compromising payload confidentiality. We begin with an overview of the AB's lifecycle.

4.1 Overview

To illustrate the operation of AB, consider an end-to-end example featuring Alice, a customer of bank.example. The bank utilizes AB for DoS defense and trusts a specific set of governmental and commercial IdPs. Figure 1 shows AB's deployment and operation overview.

Before her first AB session, Alice performs a one-time linkage of her bank account to her chosen IdP(s) via a standard OAuth 2.0 flow. When she later begins a browsing session, she logs into one or more IdPs (①), and her AB user agent retrieves a list of all accounts linked to each IdP. On navigating to the bank website, her user agent transparently requests a short-lived *key bundle* from the IdP (②) and appends an AB header to every packet (③). As these packets traverse the network, a forward-deployed *validator* verifies them using keys previously provisioned by the bank (④–⑤). Because authentication is decoupled from traffic inspection, authenticated packets are forwarded with priority, while unauthenticated traffic, e.g., originated from a botnet (⑥), is rate-limited or discarded. This lets Alice maintain access to her banking services even under heavy adversarial load.

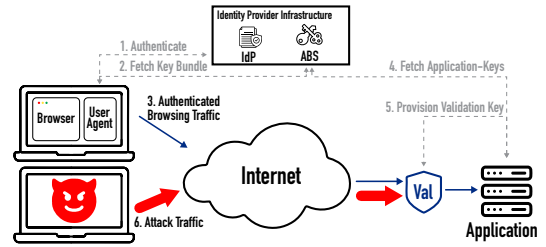


Fig. 1. Authenticated Browsing deployment and operation overview. Traffic toward applications is routed through validators, which forward authenticated traffic while dropping or rate-limiting unauthenticated traffic. Dashed arrows denote logical interactions; the IdP/ABS is not on the packet path.

4.2 System Architecture

AB comprises four components: IdPs, Applications, Validators, and User Agents.

Identity Providers (IdPs). The IdP is an OAuth 2.0/OIDC-compatible service extended with the ABS, a companion service that manages user-application linkages and provisions cryptographic key material to applications and user agents. The ABS exposes six endpoints organized into three functional groups: Application Registration, Linkage Management, and Key Provisioning. Table 2 summarizes the six ABS endpoints.

The ABS maintains three persistent stores:

Table 2. *AB Server* (ABS) API endpoints. UA = user agent; Auth column shows OAuth scope or credential type. The Link token implicitly encodes u and aid; they are listed explicitly for clarity.

Endpoint	Caller	Auth	Effect / Output
Register(idp-id) $\rightarrow$ (aid, asec)	App	None	Creates entry in $\mathcal{R}$
Link(token, u , aid, u_a) $\rightarrow$ σ	App	ab:link	Creates (u, a, u_a, σ) in $\mathcal{L}$
Revoke(σ)	App	Bearer σ	Removes entry from $\mathcal{L}$
Fetch(aid, asec) $\rightarrow$ (K_a^{id} , K_a^{mac} , T_k)	App	App creds	Updates $\mathcal{K}$; subsequent bundles derived from new pair
List(token) $\rightarrow$ [$a_1, \dots, a_n$]	UA	ab:list	List all linked applications for that user from $\mathcal{L}$
Bundle(token, aid) ($K_{a,u}^{\text{mac}}$, ruid, idp-id, T_b) $\rightarrow$	UA	ab:bundle	Returns short-lived key bundle (expiry time T_b)

- **Application Registry ($\mathcal{R}$):** maps enrolled applications to their credentials, $\mathcal{R} : \text{aid} \rightarrow (\text{asec}, \text{idp-id})$, with entries created via the Register endpoint.
- **Linkage Store ($\mathcal{L}$):** records authorized user-application relationships, $\mathcal{L} \subseteq \mathcal{U} \times \mathcal{A} \times \mathcal{U}^{\mathcal{A}} \times \Sigma$, where each entry $(u, a, u_a, \sigma) \in \mathcal{L}$ indicates that user u has authorized application a , is known to a under the application-specific identifier u_a , and that σ is a revocation token allowing a to unilaterally dissolve the linkage.
- **Key Store ($\mathcal{K}$):** holds current cryptographic key material per application, $\mathcal{K} : \mathcal{A} \rightarrow (K_a^{\text{id}}, K_a^{\text{mac}}, T_k)$, where K_a^{id} and K_a^{mac} are the application's identity-resolution and authentication keys used to derive user-application keys and T_k is the expiration time.

Applications. Applications deploy one or more *validator* nodes that authenticate inbound packets using keys periodically fetched from each registered IdP, and maintain two persistent stores:

- **User Mapping ($\mathcal{M}^{\mathcal{U}}$):** associates application-specific user identifiers with internal identifiers and revocation tokens, $\mathcal{M}^{\mathcal{U}} : u_a \rightarrow (\text{internal-id}, \sigma)$; this indirection lets the application use arbitrary internal identifiers without revealing them to the IdP.
- **IdP Mapping ($\mathcal{M}^{\mathcal{I}}$):** associates each registered IdP with the application's credentials and current key material, $\mathcal{M}^{\mathcal{I}} : \text{idp-id} \rightarrow (\text{aid}, \text{asec}, K_a^{\text{id}}, K_a^{\text{mac}}, T_k)$; only K_a^{mac} and T_k are forwarded to validators.

Validators. Validators are forward-deployed packet filters that cryptographically verify AB headers at line rate. They use authentication keys K_a^{mac} provided by the application to verify that incoming packets originate from legitimately linked users, without learning user identities. A validator maintains two data stores.

- **Key Mapping ($\mathcal{K}^{\mathcal{I}}$):** associates each idp-id with the current authentication key, its expiration time, and the previous key (retained for a grace period after key rotation), $\mathcal{K}^{\mathcal{I}} : \text{idp-id} \rightarrow (K_a^{\text{mac}}, T_k, K_a^{\text{mac,old}})$.
- **Replay Suppression Filter ($\mathcal{D}$):** a set-associative table indexed by a keyed hash of the MAC-tag τ in the AB header to detect replayed packets, $\mathcal{D} : \tau \rightarrow \{\top, \perp\}$; a tag is retained until evicted in round-robin order by subsequent insertions into the same set, giving protection against replays of any authentic packet recently seen.

User Agents. The user agent is a local proxy on user devices that intercepts browser traffic and appends AB headers to packets destined for linked applications. It maintains three data stores:

- **Linkage Mapping ($\mathcal{L}^u$):** records, for each application the user is linked to, the IdPs through which the linkage exists, $\mathcal{L}^u : \text{aid} \rightarrow [\text{idp-id}]$.
- **Bundle Cache ($\mathcal{K}^u$):** stores the current key bundle for each application, $\mathcal{K}^u : \text{domain} \rightarrow (\text{idp-id}, T_b, \text{ruid}, K_{a,u}^{\text{mac}})$, fetched on demand and refreshed before expiry.
- **Flow Table ($\mathcal{F}$):** maps active network flows to their corresponding key bundles, $\mathcal{F} : \text{flow-id} \rightarrow (\text{idp-id}, T_b, \text{ruid}, K_{a,u}^{\text{mac}})$, keyed by the flow's 5-tuple.

4.3 Protocol Flow

AB operations are organized into an initial linkage phase, session authorization, and continuous packet validation.

Phase I: Linkage & Application Registration (One-Time). Applications enroll with the ABS via the Register endpoint to obtain credentials. A user links their account at an application to one or more IdPs trusted by the application, via a standard OAuth 2.0 flow. The application obtains an access token with the `ab:link` scope and calls the Link endpoint, passing a freshly generated application-specific user identifier u_a . Upon success, the ABS returns a revocation token σ with scope `ab:revoke`, and the application stores the mapping in its user mapping $\mathcal{M}^u$. The IdP only learns that a linkage exists; it does not learn the user's account details at the application.

Phase II: Session Initialization and Key Provisioning. To fetch fresh keys from an IdP, the application calls the ABS' Fetch endpoint. The ABS returns fresh keys and expiration time $(K_a^{\text{id}}, K_a^{\text{mac}}, T_k)$. The application then pushes K_a^{mac} and T_k to its validators over a secure channel. Crucially, K_a^{id} is *not* forwarded to validators.

At the start of each AB session, the user authorizes the user agent to access ABS endpoints on their behalf, completing an OAuth 2.0 flow granting the `ab:list` and `ab:bundle` scope. The user agent then queries each IdP's List endpoint to discover which applications the user is linked to, populating the linkage mapping $\mathcal{L}^u$.

Phase III: Packet Interception and Authentication. When the browser resolves a domain, the user agent checks whether the domain appears in $\mathcal{L}^u$. If so, and no valid bundle exists in $\mathcal{K}^u$, the user agent calls the ABS' Bundle endpoint. The ABS derives a per-user authentication key $K_{a,u}^{\text{mac}}$ and a rotating user identifier ruid from the current application keys, and returns a bundle valid until T_b .

For packets destined to linked applications, the user agent: (i) retrieves the key bundle from $\mathcal{F}$ using the packet's flow identifier, (ii) computes the authentication tag τ over the packet using $K_{a,u}^{\text{mac}}$, (iii) assembles the AB header containing `idp-id`, `ruid`, `T_b`, and τ , and (iv) transmits the authenticated packet.

Phase IV: Packet Validation and Identity Extraction. Packet validation is the validator's primary function and must execute at line rate. The validator queries the replay suppression filter $\mathcal{D}$ to detect replayed packets. It then extracts from the AB header: the `idp-id`, rotating user identifier ruid , bundle expiration time T_b , and authentication tag τ . The validator derives $K_{a,u}^{\text{mac}}$ from K_a^{mac} using header fields as KDF input, binding the user's identity to the IdP without revealing it. The validator verifies τ using $K_{a,u}^{\text{mac}}$. After τ is verified, it is inserted into $\mathcal{D}$. Authenticated packets are forwarded to the application.

Identity resolution is not on the validator path. Under ordinary operation the application identifies users through existing application-layer mechanisms such as session cookies or OIDC tokens; only when attribution is required, such as anomalous behavior of authenticated traffic, does the application resolve ruid by using K_a^{id} to recompute $\text{PRF}(K_a^{\text{id}}, u_a \parallel T_b)$ for candidate linked users. The application then looks up the user's internal identifier and revocation token σ .

Fig. 2. AB Header Format

Next Hdr (1B)	Hdr. Ext. Len (1B)	Opt. Type (1B)	Opt. Len. (1B)	Opt. Data (20B)				
TCP/UDP	0x02	0x1E	0x14	IDPID (1B)	Expiry (2B)	RUID (7B)	Pad (2B)	Tag (8B)

4.4 Cryptographic Construction

AB employs a three-level key hierarchy. This hierarchy enables efficient validation: validators derive $K_{a,u}^{\text{mac}}$ from K_a^{mac} using only information present in the AB header, without contacting the IdP.

Application Key Derivation. The ABS derives two application keys from its master secret SV_{IdP} :

$$K_a^{\text{id}} \leftarrow \text{KDF}(\text{SV}_{\text{IdP}}, T_k \| \text{aid} \| \text{"id"}) \quad K_a^{\text{mac}} \leftarrow \text{KDF}(\text{SV}_{\text{IdP}}, T_k \| \text{aid} \| \text{"mac"})$$

The expiration T_k and application-identifier aid serve as domain separators.

User-Application Key Derivation. Per bundle, the ABS computes a compact rotating user identifier and derives the per-user MAC key from it:

$$\text{ruid} \leftarrow \text{PRF}(K_a^{\text{id}}, u_a \| T_b) \quad K_{a,u}^{\text{mac}} \leftarrow \text{KDF}(K_a^{\text{mac}}, \text{ruid} \| T_b)$$

The expiration T_b encodes seconds before application-key expiry. Since T_b changes across bundle epochs, the same user receives different *ruid* values over time.

Packet Authentication. The user agent authenticates packets by computing a MAC over the IPv6 header of the packet and its payload. Mutable IPv6 header fields are masked to zero before the MAC is computed. Given key bundle $(\text{idp-id}, T_b, \text{ruid}, K_{a,u}^{\text{mac}})$ and the masked IP packet P :

$$\tau \leftarrow \text{MAC}(K_{a,u}^{\text{mac}}, P)$$

The validator extracts *ruid* and T_b from the AB header, applies the same masking to the IPv6 header to obtain P , derives $K_{a,u}^{\text{mac}}$ as described above, and accepts the packet iff $\text{MAC}(K_{a,u}^{\text{mac}}, P) = \tau$.

AB Header Format. AB header placement depends on the transport protocol. For UDP/QUIC, the user agent computes τ over the original masked packet with the UDP payload, then appends the AB header to the payload. Since TCP payloads cannot be extended without breaking the byte stream, the AB header is carried in an IPv6 Destination Options extension header (Fig. 2).

5 Security Analysis

We analyze AB's security properties under the threat model of Section 2.2, assuming standard cryptographic primitives, i.e., a secure PRF (KDF and MAC) [39, 40], and the keyed construction used for *ruid* is pseudorandom for parties not holding K_a^{id} . We map the six resulting security properties to our design goals (Section 2.1).

P1: Packet Authentication. (G2, G5). *Only entities holding a valid key bundle can produce authentication tags that a validator accepts.*

Packet authentication reduces to MAC unforgeability: an adversary without $K_{a,u}^{\text{mac}}$ must forge a valid tag. For a PRF-based MAC, the forgery advantage is $\epsilon_{\text{forge}} \leq q/2^t$. At 64-bit tags ($t = 64$) and a bundle lifetime of 5 min, a brute-force adversary at full validator throughput (256 byte packet size) produces at most $q \approx 2^{34.4}$ attempts, giving $\epsilon_{\text{forge}} \leq 2^{34.4}/2^{64} \approx 2^{-29.6}$. This bound is conservative: rate limiting of failed verifications reduces q in practice. Replays of captured authentic packets are caught by the validator's duplicate filter (Section 4.2).

P2: No Stable Cross-Epoch Identifier. (G1, G3). *AB does not expose a stable protocol-level identifier that links a user's packets across key bundle epochs.*

This property follows from two design decisions in the key hierarchy (Section 4.4). The validator receives only the per-application authentication key K_a^{mac} , never K_a^{id} , which is used to compute the rotating user identifier *ruid*. Thus, while the validator can observe that packets carrying the same *ruid* belong to the same authenticated context within a bundle epoch, it cannot recover u_a from *ruid* or test whether two different *ruid* values correspond to the same user: since *ruid* is the truncated output of a PRF keyed by K_a^{id} , it is computationally indistinguishable from a random string to any party not holding K_a^{id} . Since T_b changes across bundle epochs, the *ruid* values a validator observes in different epochs are unlinkable without K_a^{id} ; any cross-epoch correlation must therefore rely on ordinary network metadata, not on a stable identifier introduced by AB.

P3: Confidentiality Preservation. (G1). *Only the application learns plaintext traffic content.*

The AB header contains only (idp-id, T_b , *ruid*, τ) (which are independent from payload content), and TLS/QUIC is not terminated by any intermediary node.

P4: Bounded Compromise. (G2, G3). *A compromised user account yields authenticated access for at most one key bundle lifetime after detection and revocation.*

If an adversary obtains a user's IdP credentials (e.g., via phishing), it can call the Bundle endpoint to obtain key bundles and generate authenticated packets. Upon detection, the application revokes the linkage (P5), so subsequent Bundle requests fail; bundles obtained before revocation remain valid until T_b . A compromised user device yields the current $K_{a,u}^{\text{mac}}$ and *ruid* and is bounded by the same window. Shorter bundle lifetimes shrink this window but increase IdP load and sensitivity to clock synchronization.

P5: Revocation Completeness. (G3). *After revocation, a user cannot obtain new key bundles for the revoked linkage.*

Revocation is immediate at the IdP: $\text{Revoke}(\sigma)$ deletes the corresponding entry from $\mathcal{L}$, and any subsequent Bundle request for that linkage is rejected.

Design Trade-Off. AB does not support validator-side revocation: pushing a revocation list to validators would require per-user state, violating the stateless validation model that enables line-rate operation (G5). The resulting exposure window is bounded by one bundle lifetime.

P6: Resilience. (G6). *Failure of an individual IdP or validator does not prevent a user linked to multiple IdPs from authenticating.*

AB allows users to link their accounts to multiple IdPs per application, and validators accept bundles from any IdP trusted by the application: if one IdP is unavailable, the user agent requests a bundle from another. Validators can likewise be deployed redundantly. As long as one linked IdP and one validator remain reachable, authenticated service continues.

6 Implementation

We implemented full end-to-end AB functionality comprising all four system components. The validator is the performance-critical component and is highly optimized for line-rate packet processing in C as a DPDK application; the application, IdP, and user agent are functional prototypes that demonstrate the complete protocol flow.

6.1 Cryptographic Operations

All cryptographic operations in our implementation use AES-based primitives, which enables efficient implementation via AES-NI hardware acceleration: AES-CBCMAC for key derivation and

for the compact *ruid* construction on single-block, fixed-length inputs [6], and AES-PMAC for the per-packet authentication tags [7]. PMAC is chosen because it is a provably secure, parallelizable MAC built from a block cipher: blocks of the message are XORed with precomputed offsets and encrypted independently, and only the results are combined in a final AES call. Unlike CMAC, whose CBC-style chain forces the AES encryptions to run strictly sequentially, PMAC lets us feed multiple independent AES-NI pipelines, improving throughput on modern CPUs.¹

6.2 Prototype Components

A key design objective is to minimize the changes AB requires to existing IdP infrastructure. We implemented a two-process Python prototype: an Identity Provider, built on Flask and Authlib, runs a standard OAuth 2.0 Authorization Code flow against user accounts and signs JWT access tokens, while a separate Authenticated Browsing Server, built on FastAPI, holds the master secret and keeps the registry, linkage store, and key store in a small SQLite database. The application side is also implemented in Python and handles registration with IdPs, periodic key retrieval (every 12 h), key distribution to validators, and user identity extraction from AB headers.

6.3 Validator

The validator must verify authentication tags at line rate without introducing significant latency. Hardware-accelerated cryptography (e.g., AES-NI [22]) together with high-speed packet processing frameworks (e.g., DPDK [54]) make this possible on commodity hardware. Figure 3 gives an overview of the validator pipeline.

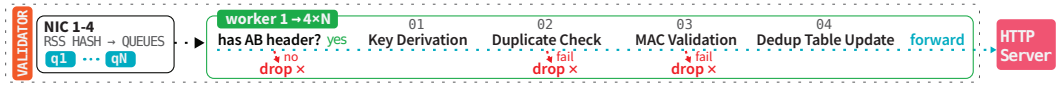


Fig. 3. The pipeline of the validator. The pipeline is configurable using policies, e.g., to allow forwarding of unauthenticated packets.

Packet Processing Pipeline. Each worker runs a poll-mode driver on a dedicated logical core and processes packets in bursts, following the verification procedure of Section 4.3. The PMAC lets us feed eight independent AES-NI pipelines per worker, masking the ≈ 7 -cycle latency of `aesenc` on our hardware; CBCMAC is retained for the one-block key derivation, where parallelism yields no benefit.

RSS and Duplicate Detection. The duplicate filter is implemented as a 4-way set-associative table per worker. The table is indexed by an 18-bit keyed hash of the MAC tag, for a total of 1 M slots per worker. A lookup is a single cache-line load and 4 tag compares; if any way matches, the packet is dropped at the validator. Since MAC tags are uniformly random, an entry in the table is expected to survive roughly $4 \times 2^{18} \approx 2^{20}$ packets on its worker before being evicted, which across 32 workers at 10 Mpps aggregate corresponds to about 3.4 seconds.

Per-worker tables catch a replay only if it reaches the same worker as the original packet. The NICs therefore apply receive-side scaling (RSS) on the IPv6 five-tuple (source address, destination address, and transport-layer ports) only. An exact replay has the same five-tuple and reaches the same worker, whose filter catches it; a replay with a modified IPv6 header may reach another

¹Poly1305 is another high-throughput MAC candidate. We choose PMAC because most commodity hardware provides AES-NI acceleration, whereas the (wider SIMD) support needed to make Poly1305 similarly advantageous is not yet as prevalent.

worker but fails the MAC check, since the tag covers the IPv6 header; and reusing a tag under a different five-tuple requires forging the MAC. Per-worker tables thus suffice, without cross-core synchronization or a distributor that steers packets by tag.

6.4 User Agent

A user-space SOCKS5 proxy is the sole interface between the browser and AB. On each new connection it checks whether the target domain is linked to an IdP, ensures a valid key bundle is available (fetching one on demand if not, and refreshing it proactively shortly before expiration), and assigns a unique socket mark that identifies the flow for downstream AB processing. A kernel-space eBPF interceptor identifies marked packets and redirects them to a cryptographic processing module, which appends the AB header using AES-PMAC over the masked packet. We also implemented an AF_XDP alternative that bypasses the kernel network stack for improved throughput on supported hardware.

7 Evaluation

AB's value as a DoS mitigation rests on three claims, which we evaluate in turn: that authenticated packets carrying the AB header reach the validator (Section 7.1), that the validator keeps up with attack traffic without becoming a bottleneck itself (Sections 7.2 and 7.3), and that legitimate users pay no noticeable latency cost for the added authentication step (Section 7.4). We do not evaluate the user agent or the IdP/ABS in this work: the user agent runs cryptographic operations comparable to those at the validator on a far smaller volume of traffic, and the IdP/ABS is located off the packet path and is exercised only at session start and bundle refresh; both are functional prototypes whose performance under load is sufficient to demonstrate the scalability of the system.

7.1 Packet Survivability

AB adds an AB header to browsing packets, either as IPv6 Destination Options extension header, or in the UDP payload if QUIC is used as the transport protocol. Because some middleboxes drop packets containing IPv6 extension headers, it is necessary to evaluate the survivability of these packets across real-world Internet paths. A prior measurement study between cloud-hosted vantage points reports approximately 90% survivability for a 24-byte Destination Options header, the same size and type as the AB header [25].

To complement these findings, we ran a RIPE Atlas measurement between January and May 2026 with 200 probes each in North America and Europe. We chose North America because a 2022 APNIC measurement of server-to-client paths [24] shows low Destination Options drop rates in many countries but rates above 70% in the United States and comparable rates in Canada; we chose Europe since its stringent confidentiality and data protection laws make AB particularly relevant. The probes sent TCP traceroute packets with a 24-byte Destination Options header to a host in an OVH data center in Roubaix, France, which captured inbound traffic and recorded the unique source addresses.² Since our probes cover only two regions and a single destination, survivability may differ for other regions and destinations.

After filtering out non-participating or offline probes, 191 probes remained in North America and 190 in Europe; the destination received packets with the extension header from 143 (74.9%) and

²We found two bugs in the RIPE Atlas platform: probes did not log successful traceroutes when extension headers were present, and a deterministic error in the platform's IPv6 pseudo-header construction caused malformed checksums. We verified that these faults lie within RIPE's infrastructure rather than in middleboxes: manually crafted packets with identical extension header size, sent from a custom node in the same data center as an affected probe, arrived uncorrupted. We have disclosed both faults to RIPE.

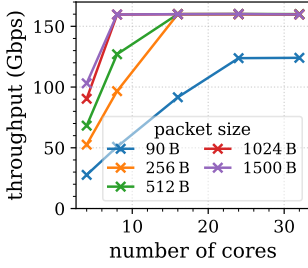


Fig. 4. Validator throughput with varying number of cores and packet sizes.

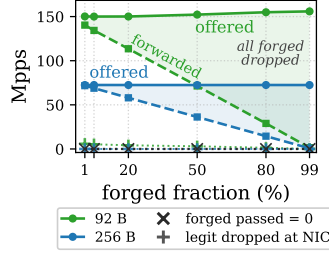


Fig. 5. Forged-traffic triage at line rate. The shaded region is forged traffic dropped by MAC verification.

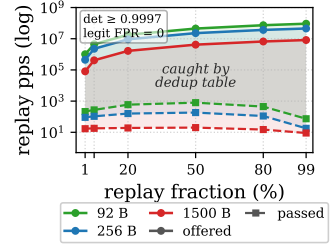


Fig. 6. Replay detection at line rate. The shaded region is caught replay traffic.

133 (70.0%), respectively. Across all TCP runs with randomly sampled probe sets and the 24-byte header, survivability ranged from 72% to 75% in North America and 67% to 86% in Europe.

Although survivability varies across vantage points, a substantial fraction of measured paths preserve a 24-byte Destination Options header. These results should be read within AB's threat model: the system is designed for attack-time deployment, so the relevant baseline is not the near-100% survivability of regular packets during normal operation, but whether enrolled users can reach a protected resource during application-layer flooding attacks.

7.2 Throughput

Setup. We evaluate the DPDK validator using two commodity servers. Each machine is equipped with two Intel Xeon E5-2695 v4 CPUs (18 cores / 36 threads per socket, 2.10 GHz), 64–128 GiB of DDR4 memory, and four Intel XL710 40 GbE QSFP+ ports (160 Gbps aggregate per host), running Debian bookworm. The load generator runs MoonGen [17]; the device under test runs our DPDK validator against an IPv6 workload carrying an AB destination-options header. Each measurement point runs for 60 s and we report the mean forwarded rate during the steady-state window.

Parameters. We sweep the number of validator worker cores $c \in \{4, 8, 16, 24, 32\}$ (one DPDK lcore per worker, each pinned to its own physical core, RX/TX queues allocated per worker), and the packet size $s \in \{90, 256, 512, 1024, 1500\}$ B (Ethernet frame size). Because every packet carries the AB header, the usual 64 B minimum frame is not possible: the smallest frame is 90 B, made up of a 14 B Ethernet header, 40 B IPv6, 24 B AB header, 8 B UDP and a 4 B FCS, with an empty payload.

Results. Figure 4 shows the forwarded throughput as a function of the number of validator cores. With 1500 B and 1024 B frames, the validator saturates the link capacity of 160 Gbps with only 8 cores (13.1 and 19.1 Mpps, respectively); with 512 B and 256 B frames, it does so with 16 cores (37.6 and 72.5 Mpps). Line rate is sustained for all larger core counts. Smaller frames are limited by packet rate rather than link capacity: at 90 B, the validator reaches 140.7 Mpps (123.8 Gbps) at 24 cores. This limit is imposed by the NIC, not the validator: a minimal program that only reflects packets does not exceed it either. Up to this limit, the 90 B curve grows nearly linearly with the number of cores, indicating that the per-core pipeline (key derivation, MAC verification, and duplicate detection) scales without contention.

7.3 Attack Triage

The validator's two filter mechanisms (MAC verification and the duplicate detection table) should drop attack traffic without introducing any packet drops for legitimate traffic. We verify both

mechanisms at line rate by running the load generator with mixes of legitimate traffic and one of two attack classes: *forged* packets (well-formed AB packets, where the MAC is replaced with a random 8-byte tag) and *replay* packets (identical retransmissions of an authentic packet).

Setup. We use the testbed of Section 7.2. The load generator sends 160 Gbps (line rate on 4×40 GbE) with $t=16$ TX threads per device for 30 s per measurement point, sweeping packet size $s \in \{92, 256, 1500\}$ B and attack fraction $f \in \{1, 5, 20, 50, 80, 99\}$ % of the offered load. A 2-byte marker in each packet identifies its traffic class, and the validator counts incoming and outgoing packets per marker.

Forged Traffic. Figure 5 contrasts the rate offered to the validator with the rate forwarded to the egress port; the shaded region in-between represents forged traffic the validator dropped after PMAC verification. In every iteration, no forged packets passed verification, which is expected given the forgery bound of 2^{-64} per packet. For 256 B packets the legitimate throughput tracks the legitimate load offered. For 92 B packets, the offered load (~ 150 Mpps) exceeds the packet rate the validator’s NIC can receive and retransmit, so legitimate packets are dropped before reaching the validator; these losses are independent of the attack and do not reflect verification errors.

Replay Traffic. Figure 6 shows the same for the duplicate-detection table. Detection rates lie in $[0.999779, 1]$ across the sweep; the residual leak of 9 to 795 packets per second is partly unavoidable, since the first copy of a packet is always accepted. No fresh authentic packet was mistaken for a replay (FPR = 0), regardless of table fill level.

7.4 End-to-End

Setup. To measure the validator’s effect on application traffic, we extend the testbed of Section 7.2 with a third host running `nginx`, which serves an 8 KiB static body over IPv6. The validator runs as a transparent L2 bridge (bump in the wire) between the load generator and the server, so AB traffic and the replies both traverse it. The load generator runs `wrk2` [53] to drive a constant HTTP load with latency measurement, and MoonGen to inject attack traffic.

Parameters. We compare three operating modes of the validator. *Raw forwarding* is the baseline, which disables the entire AB pipeline (no header insertion or strip, no MAC validation), isolating the cost of the network alone. *Validate, no attack* runs the full MAC verify, duplicate-filter, and header-strip pipeline against the legitimate (`wrk2`) HTTP traffic only. *Validate, under attack* is the same, but in addition, the load generator injects forged AB traffic over four 40 Gbps links: 145 Mpps (128 Gbps) at 90 B and 72.4 Mpps (160 Gbps, near line rate) at 256 B. At 90 B, line rate exceeds the packet rate the validator’s NIC can receive (Section 7.2); we therefore cap the attack on the link that also carries the HTTP traffic at 28.7 Mpps, a rate the NIC receives without loss, so that legitimate packets are not dropped before even reaching the validator. The other three links run at full rate. Attack packets fail the MAC check and are dropped.

Results. Figure 7 shows the HTTP request latency. Without attack traffic, the validation pipeline closely follows raw forwarding: at the median, p99, and p99.9, the two no-attack curves stay within each other’s interquartile range (IQR). Forged traffic, the worst-case offered load as it traverses the full validation pipeline, increases p99.9 by at most 0.7 ms ($3.78 \rightarrow 3.85$ ms at $s=90$ B; $3.53 \rightarrow 4.26$ ms at $s=256$ B). In all runs, `wrk2` sustained the offered load: median throughput was $19,925 \pm 25 \sigma$ req/s without attack and $19,925 \pm 45 \sigma$ req/s under attack. Without the validator, the same attack traffic collapsed the served throughput to under 18 req/s at a 17.3 s median latency, confirming that the validator’s filtering is what preserves service under attack.

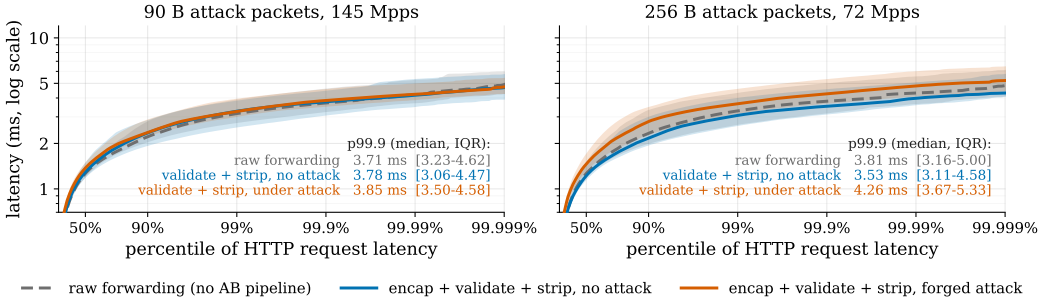


Fig. 7. HTTP request latency through the AB validator at 20 k req/s, for three operating modes and two attack-packet sizes. Lines are per-percentile medians; shaded bands are IQRs over 50 iterations of 30 s each.

8 Discussion

8.1 Mitigating Application-Layer Attacks

While AB is designed to differentiate authenticated from unauthenticated traffic, it also provides a basis for reactive mitigation of application-layer flooding or other application-layer DoS attacks mounted from compromised or malicious accounts. Once the application detects anomalous behavior by an authenticated user, it can block their *ruid*; packets continue to pass the validator for at most the remaining bundle lifetime, but are dropped at the application. To identify the user and block them from retrieving further bundles, the application resolves their *ruid* (Section 4.3) and calls the ABS' revocation endpoint. This changes the economics of application-layer attacks: since the *ruid* in every authenticated packet lets the application rate-limit per account instead of per address, the attack surface is bounded by the number of user accounts an adversary can compromise, rather than the bandwidth or number of bots it can deploy.

8.2 Regulatory Compliance

The regulatory position of AB is summarized in Table 3. In contrast to CDN/WAF-based mitigation, AB does not require third-party TLS termination or expose plaintext traffic to processors; unlike scrubbing centers, it still supports application-layer traffic differentiation. As a result, AB is to the best of our knowledge the only approach that simultaneously provides application-layer traffic differentiation while remaining compatible with banking secrecy and medical confidentiality obligations; a full GDPR analysis is provided in Section D.

8.3 Deployment Considerations

Incremental Deployment. AB does not require changes to network infrastructure, transport protocols, or application servers. Applications register with IdPs and deploy validators; users install a user agent. Individual services can adopt AB independently, and users benefit as soon as they link their accounts. AB's support for multiple IdPs per user-application linkage provides resilience against IdP unavailability, an advantage over capability-based systems that typically rely on a single capability source, and gives users a choice of identity providers, avoiding vendor lock-in.

IPv6 Requirement. The TCP variant of the AB header uses an IPv6 Destination Options extension header, limiting TCP-based AB to IPv6 networks. The UDP/QUIC variant appends the header to the payload, avoiding this constraint. Given the ongoing growth of QUIC adoption, the practical impact of this limitation is likely to diminish.

Deployment Effort. On the client, the user agent consists of a user-space SOCKS5 proxy and a kernel eBPF interceptor that adds the AB header to outgoing packets; it requires Linux and elevated privileges, but no changes to browsers or applications. Validators are DPDK applications on commodity x86 servers with AES-NI and NICs that apply RSS on the IPv6 five-tuple; they sit in front of the protected service and need no changes to the server software. The application adds account linking and distributes keys to its validators, and each IdP adds the ABS (Section 6).

Alternative Validator Implementations. Verifying AB headers on programmable switches is difficult, since switching ASICs lack dedicated AES units; AES can be emulated with lookup tables on Tofino, but only at the cost of multiple recirculations per encryption, reducing effective throughput [10]. Switches can, however, act as a pre-filter in front of software validators, dropping packets without an AB extension header and balancing flows across validators by five-tuple, which preserves per-worker duplicate detection (Section 6.3). Crypto-capable DPUs expose DPDK-capable Arm cores with AES instructions [43], which could enable a lower-overhead deployment: the validator runs on the DPU within the protected server and offloads validation entirely, so the host receives only validated packets and no separate validator host is required.

Alternative Client Implementations. A browser's QUIC stack or a native application could append the AB header directly, eliminating the need for a proxy, kernel hook, or elevated privileges and thereby easing deployment across a wide range of devices, such as mobile platforms. This does require changes to the source code of the QUIC stack, since the header is added in the send path after packet encryption. In practice, much of the QUIC traffic is produced by a limited number of libraries, so maintaining patched forks of these, or upstreaming a send-path hook, would cover most browsers and native applications without patching each one individually. Alternatively, a gateway (e.g., at ISP level) holding a key bundle could tag packets for the hosts behind it, without client software; all per-account measures then apply per gateway, not per user.

8.4 Limitations

Asymmetric Application-Layer DoS. Validators check whether a packet comes from an enrolled user without having access to the packet's plaintext. Asymmetric application-layer DoS attacks [15] are therefore left to application-level detection, which AB supports by attributing requests to accounts and enabling revocation (Section 8.1).

No Protection for Unauthenticated Users. AB provides differentiated service only for authenticated users. Unauthenticated traffic, including first-time visitors without linked accounts, receives no priority during attacks. This is inherent to authentication-based differentiation.

IdP Trust and Availability. AB relies on IdPs as trusted entities. A compromised IdP could issue unauthorized key bundles or fail to honor revocation requests, so that attack traffic passes validation. This trust assumption is mitigated by the use of established IdPs (governmental eID platforms, major SSO providers) whose compromise would have consequences beyond AB. If abnormal authenticated traffic is traced to an IdP, the application can instruct its validators to reject all packets carrying that IdP's identifier, invalidating every bundle it issued; this is an operational configuration change requiring no protocol modification. If an IdP becomes unavailable, users linked only to it lose authenticated service and receive the same service as unauthenticated users. Users linked to a second, independent IdP remain authenticated.

Floods from Compromised Accounts. An adversary with access to compromised accounts can request key bundles and distribute them among many bots, whose packets pass validation until the accounts are revoked and their bundles expire. Validators can bound this with standard DoS

mechanisms: a fixed-size sketch counts verified, non-replayed packets per *ruid*, and outliers are throttled or blocked until their bundle expires; counting only verified packets prevents off-path adversaries from throttling other users. Our prototype does not yet implement this.

Replay Protection Saturation. Replay suppression relies on a set-associative table of tags with round-robin eviction. A replay is detected only while its tag is still resident, giving a finite memory window set by the table capacity. Operators should size this table for the time window in which a captured packet could plausibly be re-injected.

Header Overhead. The 24-byte AB header adds overhead to every packet. For small packets such as TCP ACKs, this is a non-trivial fraction of the total size. For typical web traffic with larger payloads, the overhead is modest.

9 Conclusion

We present AB, a packet-authentication architecture that lifts federated user authentication from the application layer into per-packet verification at the network layer. Instead of asking a mitigation provider to inspect HTTP(S) requests, AB lets a browser user agent attach short-lived authentication tags derived from IdP-issued credentials. Forward-deployed validators verify these tags using only symmetric-key operations, without terminating TLS, inspecting application payloads, or learning user identities. The protected service retains the information needed to identify and revoke misbehaving accounts.

Our prototype shows that this design is practical. A single validator server, built with commodity hardware, sustains more than 70 million authenticated packet validations per second and reaches line rates of 160 Gbps. Attack-triage experiments show that forged packets and short-window replays are filtered at line rate, and end-to-end experiments show that validation adds only sub-millisecond p99.9 latency to an HTTP workload under a 160 Gbps forged-packet attack. Internet-path measurements show that the IPv6 Destination Options header used by the TCP variant survives on a substantial fraction of measured paths, with survivability depending on the vantage-point set.

AB changes the deployment choices available to regulated web services. Today, such services often face a binary choice: preserve plaintext confidentiality and rely on defenses that operate below the application layer, or obtain request-aware mitigation by delegating TLS termination to a third party. AB adds a third option: application-layer user identification with network-layer verification. This lets operators reserve capacity for enrolled users during attacks while keeping application payloads inside the service's existing TLS trust boundary. AB does not eliminate all DoS risks, but it mitigates application-layer flooding against services whose confidentiality requirements rule out TLS-terminating mitigation providers. As a result, the users who most need these services, enrolled customers, patients, and citizens, retain access during an attack. In doing so, AB shows that application-layer flooding defense does not require third-party plaintext access, and that the identity infrastructure it builds on is already deployed at Internet scale.

Acknowledgments

We gratefully acknowledge support for this project from the Werner Siemens Stiftung (WSS) Centre for Cyber Trust at ETH Zurich, and from the Zurich Information Security and Privacy Center (ZISC).

References

- [1] 2024. German Criminal Code, Violation of private secrets. https://www.gesetze-im-internet.de/englisch_stgb/englisch_stgb.html#p1990

- [2] Albert Gran Alcoz, Martin Strohmeier, Vincent Lenders, and Laurent Vanbever. 2022. Aggregate-based congestion control for pulse-wave DDoS defense. In *Proceedings of the ACM SIGCOMM Conference*. doi:10.1145/3544216.3544263
- [3] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic k -TAA. In *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4116)*, Roberto De Prisco and Moti Yung (Eds.). Springer, 111–125. doi:10.1007/11832072_8
- [4] David Barrera, Laurent Chuat, Adrian Perrig, Raphael M. Reischuk, and Pawel Szalachowski. 2017. The SCION internet architecture. *Commun. ACM* 60 (2017), 56 – 65. <https://api.semanticscholar.org/CorpusID:34883126>
- [5] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Junpei Urakawa. 2016. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society. doi:10.14722/ndss.2016.23132
- [6] Mihir Bellare, Joe Kilian, and Phillip Rogaway. 2000. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. System Sci.* 61, 3 (2000), 362–399. doi:10.1006/jcss.1999.1694
- [7] John Black and Phillip Rogaway. 2002. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In *Advances in Cryptology – EUROCRYPT*. doi:10.1007/3-540-46035-7_25
- [8] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings (Lecture Notes in Computer Science)*, Matthew K. Franklin (Ed.). Springer, 41–55. doi:10.1007/978-3-540-28628-8_3
- [9] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding (Lecture Notes in Computer Science)*, Birgit Pfitzmann (Ed.). Springer, 93–118. doi:10.1007/3-540-44987-6_7
- [10] Xiaoqi Chen. 2020. Implementing AES Encryption on Programmable Switches via Scrambled Lookup Tables. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure (Virtual Event, USA) (SPIN '20)*. Association for Computing Machinery, New York, NY, USA, 8–14. doi:10.1145/3405669.3405819
- [11] Cloudflare. 2023. Cloudflare mitigates record-breaking 71 million request-per-second DDoS attack. <https://blog.cloudflare.com/cloudflare-mitigates-record-breaking-71-million-request-per-second-ddos-attack/>
- [12] Cloudflare. 2026. How DDoS protection works. Cloudflare DDoS Protection documentation. <https://developers.cloudflare.com/ddos-protection/about/how-ddos-protection-works/> Accessed: 2026-10-12.
- [13] Cloudflare. 2026. What is mutual TLS (mTLS)? Cloudflare Learning Center. <https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/> Accessed: 2026-10-12.
- [14] Court of Justice of the European Union. 2020. Data Protection Commissioner v. Facebook Ireland and Maximilian Schrems (Schrems II), Case C-311/18. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:62018CJ0311>
- [15] Henri Maxime Demoulin, Isaac Pedisich, Nikos Vasilakis, Vincent Liu, Boon Thau Loo, and Linh Thi Xuan Phan. 2019. Detecting Asymmetric Application-layer Denial-of-Service Attacks In-Flight with FineLame. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*. 693–708. <https://www.usenix.org/conference/atc19/presentation/demoulin>
- [16] Wesley Eddy. 2007. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987. doi:10.17487/RFC4987
- [17] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A Scriptable High-Speed Packet Generator. In *Proceedings of ACM Internet Measurement Conference (IMC)*. doi:10.1145/2815675.2815692
- [18] European Data Protection Board. 2024. Guidelines 01/2024 on processing of personal data based on Article 6(1)(f) GDPR. https://www.edpb.europa.eu/our-work-tools/documents/public-consultations/2024/guidelines-12024-processing-personal-data-based_en
- [19] European Parliament and Council of the European Union. 2011. Directive 2011/24/EU of the European Parliament and of the Council of 9 March 2011 on the application of patients' rights in cross-border healthcare. <https://eur-lex.europa.eu/eli/dir/2011/24/oj>
- [20] European Parliament and Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. <https://data.europa.eu/eli/reg/2016/679/oj>
- [21] Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe, and Georg Carle. 2021. The pos Framework: A Methodology and Toolchain for Reproducible Network Experiments. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. doi:10.1145/3485983.3494841
- [22] Shay Gueron. 2012. Intel Advanced Encryption Standard (AES) New Instructions Set. Intel White Paper.
- [23] Dick Hardt. 2012. The OAuth 2.0 Authorization Framework. RFC 6749. doi:10.17487/RFC6749
- [24] Geoff Huston. 2022. IPv6 extension headers revisited. APNIC Blog. <https://blog.apnic.net/2022/10/13/ipv6-extension-headers-revisited/>
- [25] Justin Iurman and Benoit Donnet. 2025. The Razor's Edge: IPv6 Extension Headers Survivability. In *Proceedings of International Conference on Passive and Active Measurement*. doi:10.1007/978-3-031-85960-1_1

- [26] Mattijs Jonker, Alistair King, Johannes Krupp, Christian Rossow, Anna Sperotto, and Alberto Dainotti. 2017. Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. In *Proceedings of the 2017 Internet Measurement Conference, IMC 2017, London, United Kingdom, November 1-3, 2017*, Steve Uhlig and Olaf Maennel (Eds.). ACM, 100–113. doi:10.1145/3131365.3131383
- [27] Stephen Kent. 2005. IP Authentication Header. RFC 4302. doi:10.17487/RFC4302
- [28] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. 2014. Lightweight Source Authentication and Path Validation. In *Proceedings of the ACM SIGCOMM Conference*. 271–282. doi:10.1145/2619239.2626323
- [29] Warren Kumari and Danny R. McPherson. 2009. Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF). RFC 5635. doi:10.17487/RFC5635
- [30] Soo Bum Lee, Min Suk Kang, and Virgil D. Gligor. 2013. CoDef: Collaborative Defense Against Large-Scale Link-Flooding Attacks. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. 417–428. doi:10.1145/2535372.2535398
- [31] Markus Legner, Tobias Klenze, Marc Wyss, Christoph Sprenger, and Adrian Perrig. 2020. EPIC: Every Packet Is Checked in the Data Plane of a Path-Aware Internet. In *Proceedings of the USENIX Security Symposium (USENIX Security)*. 541–558. <https://www.usenix.org/conference/usenixsecurity20/presentation/legner>
- [32] Shihan Lin, Suting Chen, Yunming Xiao, Yanqi Gu, Aleksandar Kuzmanovic, and Xiaowei Yang. 2025. PreAcher: Secure and Practical Password Pre-Authentication by Content Delivery Networks. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. <https://www.usenix.org/conference/nsdi25/presentation/lin-shihan>
- [33] Xin Liu, Ang Li, Xiaowei Yang, and David Wetherall. 2008. Passport: Secure and Adoptable Source Authentication. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. <https://www.usenix.org/conference/nsdi-08/passport-secure-and-adoptable-source-authentication>
- [34] Xin Liu, Xiaowei Yang, and Yong Xia. 2010. NetFence: preventing internet denial of service from inside out. In *Proceedings of the ACM SIGCOMM Conference*. 255–266. doi:10.1145/1851182.1851214
- [35] Zhuotao Liu, Yuan Cao, Min Zhu, and Wei qi Ge. 2019. Umbrella: Enabling ISPs to Offer Readily Deployable and Privacy-Preserving DDoS Prevention Services. *IEEE Transactions on Information Forensics and Security* 14 (2019), 1098–1108. doi:10.1109/TIFS.2018.2870828
- [36] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. 2016. MiddlePolice: Toward Enforcing Destination-Defined Policies in the Middle of the Internet. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (2016). doi:10.1145/2976749.2978306
- [37] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *Proceedings of the USENIX Security Symposium (USENIX Security)*. 3829–3846. <https://www.usenix.org/conference/usenixsecurity21/presentation/liu-zaoxing>
- [38] Christoph Loibl, Susan Hares, Robert Raszuk, Danny R. McPherson, and Martin Bacher. 2020. Dissemination of Flow Specification Rules. RFC 8955. doi:10.17487/RFC8955
- [39] National Institute of Standards and Technology. 2005. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B.
- [40] National Institute of Standards and Technology. 2022. Recommendation for Key Derivation Using Pseudorandom Functions. NIST Special Publication 800-108 Rev. 1.
- [41] David Naylor, Matthew K. Mukerjee, and Peter Steenkiste. 2014. Balancing Accountability and Privacy in the Network. In *Proceedings of the ACM SIGCOMM Conference*. 75–86. doi:10.1145/2619239.2626306
- [42] David Naylor, Kyle Schomp, Matteo Varvello, Ilias Leontiadis, Jeremy Blackburn, Diego R. López, Konstantina Papagianaki, Pablo Rodriguez Rodriguez, and Peter Steenkiste. 2015. Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (London, United Kingdom) (SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 199–212. doi:10.1145/2785956.2787482
- [43] NVIDIA. [n. d.]. NVIDIA BlueField-3 DPU. <https://www.nvidia.com/en-us/networking/products/data-processing-unit/>. Accessed: 2026-10-07.
- [44] The Federal Assembly of the Swiss Confederation. 2024. Federal Act on Banks and Savings Banks. https://www.fedlex.admin.ch/eli/cc/51/117_121_129/de
- [45] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce M. Maggs, and Yih-Chun Hu. 2007. Portcullis: protecting connection setup from denial-of-capability attacks. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. doi:10.1145/1282380.1282413
- [46] Sivaramakrishnan Ramanathan, Jelena Mirkovic, Minlan Yu, and Y. Zhang. 2018. SENS Against Volumetric DDoS Attacks. *Proceedings of the 34th Annual Computer Security Applications Conference* (2018). doi:10.1145/3274694.3274717

- [47] Piotr Rataj. 2025. Botnet defense under EU data protection law. *Computer Law & Security Review* 56 (2025), 106080. doi:10.1016/j.clsr.2024.106080
- [48] Eric Rescorla. 2026. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 9846. doi:10.17487/RFC9846
- [49] Natushiko Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. 2014. OpenID Connect Core 1.0. https://openid.net/specs/openid-connect-core-1_0.html
- [50] Jared M. Smith, Kyle Birkeland, Tyler McDaniel, and Max Schuchard. 2020. Withdrawing the BGP Re-Routing Curtain: Understanding the Security Impact of BGP Poisoning through Real-World Measurements. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society. doi:10.14722/ndss.2020.24240
- [51] Jared M. Smith and Max Schuchard. 2018. Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing. In *IEEE Symposium on Security and Privacy (SP)*. 599–617. doi:10.1109/SP.2018.00032
- [52] David Taylor, Tom Wu, Nikos Mavrogiannopoulos, and Trevor Perrin. 2007. Using the Secure Remote Password (SRP) Protocol for TLS Authentication. RFC 5054. doi:10.17487/RFC5054
- [53] Gil Tene. [n. d.]. wrk2: A constant throughput, correct latency recording variant of wrk. <https://github.com/giltene/wrk2>.
- [54] The Linux Foundation. 2026. Data Plane Development Kit (DPDK). <https://www.dpdk.org/>.
- [55] Joe Touch, Allison Mankin, and Ronald P. Bonica. 2010. The TCP Authentication Option. RFC 5925. doi:10.17487/RFC5925
- [56] Muoi Tran, Min Suk Kang, Hsu-Chun Hsiao, Wei-Hsuan Chiang, Shu-Po Tung, and Yu-Su Wang. 2019. On the Feasibility of Rerouting-Based DDoS Defenses. In *IEEE Symposium on Security and Privacy (SP)*. 1169–1184. doi:10.1109/SP.2019.00055
- [57] Karl Wüst, Giacomo Giuliani, Markus Legner, Jean-Pierre Smith, Marc Wyss, Jules Bachmann, Juan A. García-Pardo, and Adrian Perrig. 2025. Hummingbird: Fast, Flexible, and Fair Inter-Domain Bandwidth Reservations. In *Proceedings of the ACM SIGCOMM Conference*. 1070–1089. doi:10.1145/3718958.3750495
- [58] Marc Wyss, Giacomo Giuliani, Markus Legner, and Adrian Perrig. 2022. DoCile: Taming Denial-of-Capability Attacks in Inter-Domain Communications. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. 1–10. doi:10.1109/IWQoS54832.2022.9812889
- [59] A. Yaar, A. Perrig, and D. Song. 2004. SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. 130–143. doi:10.1109/SECPRI.2004.1301320
- [60] Xiaowei Yang, David Wetherall, and Thomas Anderson. 2008. TVA: A DoS-Limiting Network Architecture. *IEEE/ACM Transactions on Networking* 16, 6 (2008), 1267–1280. doi:10.1109/TNET.2007.914506
- [61] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qi Li, Mingwei Xu, and Jianping Wu. 2020. Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches. In *NDSS*. The Internet Society. doi:10.14722/ndss.2020.24007
- [62] Shengbao Zheng and Xiaowei Yang. 2019. DynaShield: Reducing the Cost of DDoS Defense using Cloud Services. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*. <https://www.usenix.org/conference/hotcloud19/presentation/zheng>

A Research Reproducibility

To support reproducibility, we make all artifacts required to reproduce our results available. The release consists of two artifact packages. The first package contains the source code of our high-performance DPDK-based implementation of AB and the MoonGen scripts used to generate AB's packet floods. The package can be found at <https://github.com/netsec-ethz/authenticated-browsing>. The second package contains the experiment setup and evaluation artifacts and can be found at <https://github.com/netsec-ethz/authenticated-browsing-eval>. Our experiments were conducted using *pos*, a framework for creating reproducible artifacts [21]. This package includes the *pos* scripts, raw results, and the plotting code used to generate the figures in this paper. The scripts also cover the configuration of the HTTP load generator and the nginx web server used in our evaluation.

B Ethical Considerations

This work does not raise any ethical issues. The research did not involve human subjects, personally identifiable information, or otherwise sensitive user data. All experiments that generate high traffic volumes were conducted in an isolated testbed. No vulnerabilities affecting real-world systems were discovered or disclosed as part of this work. As such, no ethical review or responsible disclosure process was required.

C Generative AI Usage

The authors used generative AI tools while preparing this paper and remain fully responsible for the final submission, including its text, figures, code, experimental data, and citations.

Tools Used and How. We used GitHub Copilot and Claude Code to assist with implementing the validator and load generator. Copilot provided inline code completions and documentation support, while Claude Code was used to explain code, debug implementation issues, and troubleshoot setup issues such as dependency mismatches. In addition, Claude Code was used to monitor long-running benchmark runs for anomalies that occurred while experiments were in progress.

For the writing of this paper, we used Claude (Opus 4.6-5.5, Fable 5-5.1) to assist with structuring portions of the text and with sentence-level revisions aimed at improving readability. The model's output was treated as suggestions, and subject to human review. Claude Opus 5.5 was also used during the preparation of the camera ready version to condense sections of the paper.

Validation. All AI-assisted contributions were reviewed by a human author before inclusion. Code suggestions and proposed fixes were inspected and tested. Any issues flagged during benchmark monitoring were diagnosed by the authors, and any resulting changes were reviewed like other code changes. Writing suggestions were edited and incorporated by the authors, who are responsible for the final wording, technical claims, and citations.

D Legal Analysis

This appendix provides a detailed analysis of AB's compliance properties under the European Union's General Data Protection Regulation (GDPR), which represents one of the strictest data protection regimes worldwide. Compliance with the GDPR generally ensures compatibility with other major frameworks such as the UK Data Protection Act 2018, and the Swiss Federal Act on Data Protection.

D.1 Personal Data and Role Assignment under AB

Within the AB architecture, several categories of personal data are processed: the rotating user identifier (*ruid*), authentication tags (MAC), logs of authentication events, and traffic metadata (IP addresses, timestamps). The relevant actors assume distinct roles under the GDPR:

- **Client (controller):** The protected service determines the purposes and means of data processing for DoS mitigation. It decides which IdPs to enroll with, which users to link, and which filtering policies to apply.
- **IdP (controller):** The IdP independently manages user accounts, performs identity verification, and issues cryptographic key material. It determines the means of its own processing.
- **Validator (processor):** The validator acts on the client's instructions, performing packet filtering according to defined policies. It processes only the data necessary for authentication verification and does not determine the purposes of processing.

This role assignment is significant: because the validator is a processor rather than a controller, the legal requirements for the client-validator relationship are less onerous. Under GDPR Article 28, a data processing agreement between the client and the validator operator suffices, and no joint controllership arrangement or separate legal basis for the validator's processing is required. This means that organizations can integrate third-party validators using a standard controller-processor contract, instead of needing to negotiate specific joint controllership terms and purpose-specific legal bases for each deployment.

Compared to traditional DoS mitigation mechanisms that terminate TLS and may reuse traffic data for their own analytics or optimization purposes, AB includes a validator role that is confined to a narrow, clearly defined processor role, which reduces contractual complexity and the scope of compliance assessments.

More broadly, AB clearly separates the roles of client, IDP, and validators, and limits each to the data necessary for its function. This avoids the "all-in-one" controller role that is often assumed by conventional DoS services. This modular role allocation reduces legal ambiguity and concentrates identification power only at the client, which simplifies governance and accountability under GDPR.

D.2 Legal Basis for Processing

Organizations deploying AB can rely on multiple legal bases for the processing of personal data:

- (1) **Legitimate interest** (GDPR Article 6(1)(f)): The European Commission and the European Data Protection Board (EDPB) have recognized that ensuring the network and information security of IT systems constitutes a legitimate interest [18]. Recital 49 of the GDPR explicitly mentions "preventing unauthorised access to electronic communications networks" as an example [47].
- (2) **Legal obligation** (GDPR Article 6(1)(c)): Organizations in critical sectors are subject to legal obligations that require strong information security measures, including under the NIS2 Directive (Article 21(1)), the ePrivacy Directive (Article 4(1)), and sector-specific regulations such as DORA for financial services.
- (3) **Consent** (GDPR Article 6(1)(a)): AB's opt-in architecture, where users must actively link their accounts before receiving authenticated service, can serve as a basis for consent, particularly since a reasonable alternative (standard unauthenticated access) remains available.

Although a legal basis for each purpose must be decided before processing, organizations can rely on different bases for different aspects of the AB infrastructure. In practice, AB's core operations of network and information security fit naturally under a confirmed ground for legitimate interest and, for some entities, under legal obligation [47]. Where appropriate, the opt-in model can also support a consent basis, but consent is not required for the core AB infrastructure and is usually

not preferred, as the ability to revoke consent at any given point makes it a less stable basis. This alignment with a well-established legitimate interest ground makes deployment easier to justify to regulators than architectures that depend on user consent or weakly documented fallback bases.

D.3 Optimized Privacy-by-Design Compliance

GDPR Article 25 requires controllers to implement “appropriate technical and organizational measures” for data protection “by design and by default.” AB’s architecture embodies this principle through several design choices:

End-to-end encryption maintained: Unlike TLS-terminating proxies, AB does not decrypt user traffic at any intermediate point. The validator authenticates packets based on a cryptographic header without accessing payload content.

Data minimization: The AB header contains only the minimum data necessary for authentication: a 1-byte IdP identifier, a 2-byte expiration timestamp, a 7-byte *ruid*, and an 8-byte authentication tag. No user-identifying information is transmitted in plaintext.

Pseudonymization with rotation: The *ruid* is derived via a PRF keyed by K_a^{id} , yielding an output indistinguishable from a uniformly random 56-bit string to any party not holding the key, and thus constituting pseudonymous data for all other parties. The 5-minute bundle lifetime ensures that *ruids* rotate frequently, limiting the potential for cross-session tracking.

Distributed processing: Authentication verification (validator), identity management (IdP), and user deanonymization (client) are separated across distinct entities. No single entity other than the client can both verify packet authenticity and identify the user.

Purpose limitation: Validators are technically constrained, not merely contractually obligated, to perform only authentication verification. They lack the cryptographic keys (K_a^{id}) required to resolve *ruids* to users, and thus cannot deanonymize users even if they attempted to.

Together, these properties result in a markedly stronger “privacy by design” posture, and make AB easier to defend under GDPR Article 25 than conventional DoS mitigation mechanisms that routinely decrypt, inspect, or store plaintext traffic. A strong privacy by design framework also allows for more adaptability to changes in data protection regulations, lowering compliance costs in the long run.

D.4 Risk Minimization in Cross-Border Data Transfer

Cross-border data transfers present particular challenges for regulated organizations, especially following the *Schrems II* judgment [14] by the Court of Justice of the European Union (CJEU), which invalidated the EU-US Privacy Shield and imposed the stringent conditions that hold for transfers to countries without adequate data protection. More recently, the European Commission has adopted the EU-US Data Privacy Framework (DPF) as a new adequacy decision for certified U.S. organizations, which simplifies transfers to participating providers. However, adequacy does not remove the need to minimize the exported data or assess residual risks, and the AB framework remains a legally safer option compared to mechanisms that terminate TLS and share plaintext data. Consider an EU-based financial institution using validators deployed by a US-based DoS mitigation provider:

- The validator processes only AB header metadata: pseudorandom *ruids*, which it cannot resolve to users (Section D.3), authentication tags, timestamps, and IdP identifiers. It does not access payload content.
- Traffic metadata (IP addresses) is processed transiently for packet forwarding and authentication; no persistent logging of user-identifying data is required at the validator.

Under these conditions, the data transferred to a non-EU validator is limited to pseudonymized metadata that cannot be linked to identifiable individuals without the client’s identity-resolution

key K_a^{id} . A transfer impact assessment under the *Schrems II* framework would therefore indicate substantially lower risk than transferring plaintext traffic to a third-party proxy.

For organizations requiring stronger assurances, AB's distributed architecture allows validators to be deployed within the EU while using IdPs from other jurisdictions, or vice versa, providing flexibility in jurisdictional placement.

D.5 Compatibility with High-Sensitivity Sectors

Beyond general data protection law, many high-stakes sectors, such as finance, healthcare, and e-government, are subject to sector-specific secrecy and confidentiality requirements that constrain the use of third-party infrastructure. AB's separation of identity, metadata, and content makes it adaptable to the requirements of these regimes, enabling organizations to deploy external DoS mitigation without exposing protected information.

Financial Services. Banking secrecy laws generally contain wide restrictions on how financial institutions can disclose client information to third parties. AB naturally addresses these constraints because neither validators nor IdPs access the content of user communications. The client retains exclusive control over user identity extraction.

Healthcare. Medical confidentiality statutes, including the EU's Cross-Border Healthcare Directive (2011/24/EU) and national implementations, restrict the processing of health-related data. Since AB validators process only pseudonymized authentication metadata and do not access clinical data within encrypted traffic, healthcare organizations can deploy third-party validators without raising patient confidentiality concerns.

E-Government. Government services handling citizen data face both data protection requirements and data sovereignty expectations. AB's distributed architecture allows governments to operate their own IdPs (or use existing governmental eID platforms) while deploying validators in trusted jurisdictions, keeping control over identity data while using distributed DoS mitigation.

D.6 Regulatory Comparison of DoS Mitigation Approaches

The main text argues that AB occupies a different regulatory position from conventional DoS mitigation mechanisms. Table 3 summarizes the operational features that drive this distinction.

Table 3. Regulatory comparison of DoS mitigation approaches.

Criterion	CDN/WAF	Scrubbing	Authenticated Browsing
TLS termination by third party	Yes	No [†]	No
Plaintext access by third party	Yes	No [†]	No
Personal data at processor	Full traffic	Metadata	Pseudonymized
Cross-border content transfer	Yes	No	No
Banking secrecy compatible	No	Partially	Yes
Medical confid. compatible	No	Partially	Yes
App-layer attack differentiation	Yes	No	Yes

[†]Scrubbing centers cannot differentiate app-layer attacks without terminating TLS.

Received 5 June 2026; accepted 10 September 2026