

SESAME: A Layered Security Framework for SCTE 130 ESAM with Sub-Millisecond Overhead

Bo Kelleher

CITE THIS ARTICLE

Kelleher, Bo; 2026. SESAME: A Layered Security Framework for SCTE 130 ESAM with Sub-Millisecond Overhead. SET INTERNATIONAL JOURNAL OF BROADCAST ENGINEERING. ISSN Print: 2446-9246 ISSN Online: 2446-9432, doi: 10.18580/setijbe.2026.6 web link: <https://dx.doi.org/10.18580/setijbe.2026.6>



COPYRIGHT This work is made available under a Creative Commons Attribution-NonCommercial 4.0 International License. Reproduction in whole or in part is permitted provided the source is acknowledged.

SESAME: A Layered Security Framework for SCTE 130 ESAM with Sub-Millisecond Overhead

Bo Kelleher, *Senior Solutions Architect, Techex Ltd.*

Abstract—The SCTE 130 suite defines the architecture for digital program insertion in broadcast and streaming, but its real-time Event Signaling and Management (ESAM) interface has no standardized security mechanism. As ad insertion infrastructure migrates to multi-tenant cloud and hybrid environments, the absence of authentication, integrity verification, and channel-scoped authorization on ESAM becomes a measurable liability. This paper introduces SESAME (Secure ESAM Authentication and Message Encryption), a layered security framework proposed as SCTE 130-9. SESAME defines three additive tiers, namely HMAC-SHA256 authentication and integrity, channel-scoped authorization, and AES-256-GCM payload encryption, conveyed through standard HTTP headers without modifying ESAM XML schemas. An open-source Rust reference implementation sustains 243,000 requests per second at eight threads with sub-millisecond per-request overhead, well within broadcast latency budgets.

Index Terms—broadcast security, SCTE 130, ESAM, POIS, digital program insertion, ad insertion, message authentication, HMAC-SHA256, AES-GCM, cloud broadcast

I. INTRODUCTION AND MOTIVATION

DIGITAL program insertion is the technical foundation of modern advertising-supported broadcast and streaming. The SCTE 130 family of standards defines a comprehensive architecture for it: SCTE 35 conveys cueing messages within transport streams [1], while ESAM (Event Signaling and Management, formalized in ANSI/SCTE 250 and used throughout SCTE 130-2, 130-5, 130-7, and 130-8 [2]–[5]) provides the real-time HTTP interface over which Signal Acquisition Systems (SAS), Placement Opportunity Information Services (POIS), and Ad Decision Servers (ADS) coordinate splice decisions. These interfaces control the routing of billions of dollars in advertising inventory worldwide.

Despite this commercial significance, the SCTE 130 suite is silent on security. The ESAM interface has no standardized mechanism to authenticate request originators, verify message integrity in transit, restrict operations to authorized channels, or protect commercially sensitive payload content from passive observation. This silence was acceptable when ad insertion infrastructure operated within physically isolated broadcast facilities. It is no longer acceptable. ESAM endpoints today routinely span availability zones, virtual private clouds, and even cloud providers; they are reachable across multi-tenant infrastructure shared with neighbouring services; and they are

a documented target of an ad-fraud ecosystem that generates billions of fraudulent ad requests annually.

This paper introduces SESAME, a layered security framework designed to close this gap, formally proposed to the SCTE Engineering Committee as SCTE 130-9 and specified, implemented, and benchmarked here.

II. THREAT MODEL

Table I sets out six concrete attack vectors against unprotected ESAM deployments, each grounded in observable deployment architectures, with the SESAME tier that mitigates it. Two shape the design that follows: tampering by an intermediary such as a misconfigured load balancer or compromised reverse proxy, and cross-channel interference exploiting the absence of any binding between requester identity and permitted channels.

III. DESIGN CONSTRAINTS AND ALTERNATIVES

Any security mechanism for ESAM must respect five non-negotiable constraints inherited from broadcast signal processing. End-to-end ESAM processing budgets are typically under 100 ms; the security layer must add no more than 1 ms to request validation. Existing unauthenticated SAS devices must continue to function during incremental migration. Each ESAM request must remain stateless, with no session establishment and no multi-round-trip handshakes on the request path. Operational management must be tractable for broadcast engineers, not security specialists. And the mechanism must operate over standard HTTP without changes to existing ESAM XML schemas.

Mutual TLS is the obvious alternative, and the decisive objections are architectural rather than performance-related. TLS protects a transport that, in the deployments SESAME targets, does not reach the POIS: it is customarily terminated at an edge proxy or load balancer, leaving the inner hop under separate trust. Section II names exactly that intermediary as the tampering vector, and mutual TLS, authenticating each hop independently, cannot detect modification by an intermediary that is itself a legitimate peer, a practice both common and damaging in measurement [13]. A SESAME signature is computed by the originating SAS and verified by the terminating POIS, so it spans every intermediary between them. TLS further authenticates an endpoint without constraining what it may do, whereas cross-channel interference is an authorization failure: Tier 2 binds a credential to its channels

B. Kelleher is with Techex Ltd. E-mail: bo.kelleher@techex.tv.

The reference implementation described in Section V is available as open source at <https://github.com/bokelleher/rust-pois>.

TABLE I
ESAM THREATS AND SESAME MITIGATIONS.

Threat	Attack Vector	SESAME Mitigation
Signal Injection	Unauthorized ESAM requests from rogue systems	Tier 1: HMAC authentication rejects unsigned requests
Signal Tampering	Man-in-the-middle modification of XML payloads	Tier 1: HMAC integrity verification detects modifications
Cross-Channel	Requests targeting unauthorized channel IDs	Tier 2: Channel-scoped credentials prevent cross-tenant access
Reconnaissance	Passive observation of ESAM traffic	Tier 3: AES-256-GCM payload encryption
Replay	Retransmission of captured valid requests	Tier 1: Timestamp + nonce replay protection
DoS Flooding	Volume-based ESAM request flood	Tier 1: Early rejection of unauthenticated requests

and signs the declared scope, and response signing binds a conditioning decision to the request that elicited it, neither of which a tunnel expresses. Client certificates also have no practical provisioning path on legacy encoders and cannot be enforced per channel, so mutual TLS demands a coordinated rollout across every peer at once, the flag day the tier model exists to avoid. We do not rest the case on latency: under TLS 1.3 [11] session resumption amortises handshake cost on persistent connections toward zero, so the objection is confined to connection establishment, redistributed connections, and cold starts, where SESAME's stateless construction is invariant to connection topology.

Signing a canonical serialisation of an HTTP request with HMAC-SHA256 is not a novel construction, and SESAME does not claim it as one. RFC 9421 [12] standardises the idea, supports HMAC-SHA256, and is motivated by the same intermediary-transformation case, so Tier 1 is accurately described as a constrained profile of an established mechanism. The contribution is the profile. RFC 9421 is deliberately general, and two conformant implementations may make mutually incompatible choices, which is correct for a framework and a liability for a multi-vendor broadcast specification; SESAME fixes the covered components, the algorithm, and the replay discipline, publishes golden conformance vectors as the contract, and adds channel-scoped authorization, correlation-bound response signing, per-channel tier policy, and payload encryption, none of which RFC 9421 addresses. A future revision could reasonably be expressed as an RFC 9421 profile, and we record that as an open question for the SCTE Engineering Committee. AWS Signature Version 4 [14] applies the same discipline at scale but is bound to the AWS credential, region, and service model, scopes key derivation rather than resource authorization, and does not authenticate responses. JWT bearer tokens do not couple identity to message content, and custom binary protocols violate schema stability. SESAME is engineered to satisfy all five constraints simultaneously.

IV. SESAME PROTOCOL

SESAME defines three additive security tiers. Operators adopt the tier appropriate to their threat model and operational maturity, and a per-channel security policy on the server enables tier requirements to be raised gradually.

A. Tier 1 — Authentication and Integrity

Every ESAM request carries five HTTP headers: protocol version, key identifier, ISO-8601 UTC timestamp, 128-

bit nonce, and HMAC-SHA256 signature. The signature is computed over a canonical string consisting of HTTP method, request path, timestamp, nonce, and the SHA-256 hash of the request body, joined with line-feed separators [6]. This construction covers both metadata and payload without forcing the server to buffer the full body before validation. Replay protection combines a sliding timestamp window (default 300 seconds) with a nonce cache, optionally implemented as a probabilistic data structure for high-throughput deployments. Keys are referenced by a `key-id` to enable rotation without service interruption, with both the old and new key valid simultaneously during a configurable overlap.

The canonical string is defined formally in ABNF [8] as:

$$\begin{aligned} \text{canonical} = & \text{method} \parallel \text{LF} \parallel \text{path} \parallel \text{LF} \parallel \text{timestamp} \\ & \parallel \text{LF} \parallel \text{nonce} \parallel \text{LF} \parallel \text{body-hash} \\ & [\parallel \text{LF} \parallel \text{scope}] \end{aligned} \quad (1)$$

where *path* is the request path component, whose syntax is defined by [9], *body-hash* is the lowercase hexadecimal SHA-256 of the (possibly-encrypted) request body, *nonce* is a 128-bit value drawn from a CSPRNG meeting the requirements of [7], and *scope* is the value of the `X-SESAME-SCOPE` header, present only when Tier 2 is active.

To authenticate the POIS's conditioning decision — the highest-value target identified in Section II — responses are signed with the same HMAC-SHA256 construction over a *response canonical string* that substitutes a fixed `RESPONSE` token and the answered acquisition signal identifier for the method and path, binding each response to the request it answers. A conformant client verifies the response signature, timestamp, and correlation, and rejects any response that fails validation.

B. Tier 2 — Authorization

Tier 2 binds credentials to specific channel scopes. An additional `X-SESAME-SCOPE` header declares the intended channel and is included in the signature computation, preventing replay of a signed request against a different channel. The server enforces a key-id-to-scope mapping at validation time, returning HTTP 403 with a SESAME-specific error code on mismatch.

C. Tier 3 — Encryption

Tier 3 wraps the ESAM XML body in AES-256-GCM authenticated encryption [10]. Encryption metadata — encryption key identifier and 96-bit IV — travels in additional

headers. A fresh IV is generated by a CSPRNG for every encryption and is never reused with a given key, as required by [10]; in particular a request and its response use independent IVs. The SESAME header set is bound as additional authenticated data, so the headers cannot be altered without invalidating the GCM tag, which is appended to the ciphertext. Encryption keys are managed independently from signing keys, allowing different rotation cadences for integrity and confidentiality material.

D. Replay Protection and Time Discipline

The 300 s acceptance window is often misread as a replay exposure window. It is not: a request replayed within it presents a (key-id, nonce) pair already recorded and is refused by the nonce cache, while one replayed after it fails the freshness check, so no interval exists in which replay succeeds. The window instead bounds cache residency, and hence memory, to the product of window and request rate. It must exceed the worst-case clock offset between signing SAS and verifying POIS, a weaker assumption in broadcast than in general web infrastructure: facilities carrying uncompressed video already distribute precision time, whether PTP under IEEE 1588 in ST 2110 plants or GPS-disciplined references, and NTP-disciplined hosts typically hold within tens of milliseconds, four orders of magnitude inside the window. Implementations should expose the window as configuration and alarm on the timestamp-rejection rate, a direct indicator of time-discipline failure. The bound is the cache itself: a restart discarding it reopens replay for the remainder of the window, and nodes that do not share a cache allow a request accepted by one to be replayed against another, so strict cross-node replay protection requires backing the cache seam with a shared store.

The full protocol — including header reference, error codes, and worked end-to-end examples for all three tiers — is specified in the companion SCTE 130-9 draft.

V. REFERENCE IMPLEMENTATION

A reference implementation has been built as Axum middleware for the open-source Rust POIS project at <https://github.com/bokelleher/rust-pois>. The middleware intercepts ESAM requests before the signal-processing handler, performing in order header extraction, replay-window timestamp validation, key-id lookup, HMAC-SHA256 verification, scope authorization, and Tier 3 decryption. Failures short-circuit the pipeline with appropriate HTTP status codes before any ESAM logic executes. After the signal-processing handler produces a conditioning decision, the same layer signs the response — and, when Tier 3 is active, encrypts it under a freshly generated IV — before it is returned, so the framework is bidirectional rather than request-only.

The implementation is approximately 2,000 lines of Rust and is, to the author's knowledge, the only open-source ESAM/POIS server with a security framework built in.

VI. PERFORMANCE EVALUATION

Two experiments are reported, since per-operation cost and behaviour under load answer different questions. Per-operation

TABLE II
PER-REQUEST SESAME OVERHEAD ON AWS `c6i.xlarge` (1 KB PAYLOAD).

Operation	Median (µs)	P99 (µs)
Tier 1 (HMAC verify)	1.37	1.39
Tier 2 (+ scope check)	1.43	1.46
Tier 3 (+ AES-GCM decrypt)	3.31	3.43

TABLE III
SUSTAINED TIER 1 THROUGHPUT WITH THE REPLAY CACHE IN PATH (XEON E5-2680 v4, 1 KB PAYLOAD).

Threads	Throughput (req/s)	p50 (µs)	p99 (µs)
1	40,111	23.4	39.7
2	80,222	21.2	41.0
4	143,899	23.9	45.4
8	242,823	24.1	49.1

figures use the Criterion harness (rustc 1.96.0, `--release`, 1 s warm-up, 3 s measurement), where p50 is the median estimate and p99 the 99th percentile of per-sample iteration means, a dispersion indicator rather than a production service level; these runs are single-threaded and perform no replay-cache maintenance. Sustained-load figures come from a harness that pre-signs requests so the timed region is verification only and gives each a distinct nonce, so every request inserts into the replay cache, the worst case for growth; there p50 and p99 are true per-request percentiles. Sample sizes are 60,000 requests for cache occupancy, 20,000 per thread for concurrency, and 40,000 per arm for rejection, over synthetic bodies of 1,024, 4,096, and 16,384 bytes, with a 300 s window.

On an AWS `c6i.xlarge` instance (Intel Xeon Platinum 8375C, 4 vCPU, with AES-NI and SHA extensions) at a 1 KB payload, per-operation overhead is as shown in Table II. The combined inbound path costs 3.31 µs at 1 KB and 27 µs at 16 KB, and the encrypt-and-sign response path stays under 0.03 ms at 16 KB, inside the 1 ms budget of Section III. The tier increments differ in kind: Tier 2 adds a constant-time scope comparison of 0.06 µs independent of payload, whereas Tier 3 adds a full AES-256-GCM pass proportional to payload, which is why only that increment grows with size, reaching 13 µs at 16 KB. A third profile, a QEMU virtual CPU with neither AES-NI nor SHA extensions, holds worst-case combined p99 at 0.219 ms, a conservative floor for hardware without crypto acceleration.

Micro-benchmarks miss the behaviour of a server whose replay cache is shared mutable state on the request path. Table III reports sustained Tier 1 throughput on a third profile, a 16-core Intel Xeon E5-2680 v4 with AES-NI but without SHA extensions, the pessimistic case for body hashing. Throughput scales close to linearly to eight threads with flat median latency, and per-request cost is invariant to cache occupancy from 5,000 to 60,000 live entries. Payload sensitivity at four threads follows body hashing: 143,674 req/s at 1 KB, 61,970 at 4 KB, and 28,901 at 16 KB with a p99 of 258 µs, so even the largest payload under load stays four times inside the budget.

That invariance was not free. Sweeping expired entries on every lookup, as the cache first did, makes per-request cost proportional to occupancy and hence to request rate: median

latency rose from 23.4 to 177.4 μ s as the cache filled, and with the sweep under the cache mutex throughput *fell* from 15,143 to 2,434 req/s from one thread to eight, a four-thread p99 of 6.55 ms and six times outside the budget. Sweeping at most once per wall-clock second makes this $O(1)$ amortised and yields Table III; an entry lingering past expiry can only reject, never falsely accept. SESAME's security properties rest on its cryptography, its latency properties on the replay cache. The same harness measured the rejection path behind the denial-of-service argument of Section II: invalid signatures were refused at 82,759 req/s against 55,718 req/s accepted, a modest saving of roughly 4 μ s since both paths hash the body before the signature can be tested. The substantive benefit is that rejection precedes all ESAM parsing and signal-conditioning work, a cost these measurements do not attribute.

These results are bounded in four ways. All three profiles are single-socket Intel x86-64; no ARM64 platform was measured, and the tiers depend on acceleration whose availability differs by architecture. All measurements are single-node and in-process, excluding network transport, TLS termination, and the shared replay store a scaled deployment requires, whose coordination cost is unmeasured and could dominate. Payloads are synthetic rather than production ESAM XML, excluding parsing cost. And the harness saturates one process with pre-signed requests, reporting a capacity ceiling, not realistic arrival patterns.

VII. ADOPTION AND STANDARDIZATION PATH

SESAME is engineered for incremental deployment. The recommended migration runs the POIS in monitoring-only mode, enrolls clients per channel in priority order, moves the policy from optional to required, and finally enables Tier 3 for commercially sensitive channels. No phase requires a coordinated cutover.

We propose SESAME for inclusion in the SCTE 130 family as SCTE 130-9, modeled after SCTE 130-7 (Message Transport) as a cross-cutting specification applying to any ESAM HTTP interface regardless of service role; incorporation as a normative annex to a future SCTE 130-7 revision is an alternative path. The current document contains normative requirements for the SESAME Server role; ADS-side normative requirements and mutual authentication are identified as future work.

A. Key Management and Distribution

SESAME does not define a key distribution protocol. This is a scoping decision rather than an oversight, though it does bound what the framework guarantees. The reason is the argument of Section III: mandating distribution at Tier 1 would impose exactly the coordinated infrastructure rollout that makes mutual TLS unadoptable, and a framework whose adoption barrier is a shared secret per peer can be deployed channel by channel where one requiring a certificate authority cannot. The draft nonetheless addresses key management in three respects. It fixes the lifecycle: keys are referenced indirectly by `key-id` so rotation proceeds without interruption, old and new keys remain valid across a configurable overlap,

and signing and encryption keys rotate independently. It states the properties any conformant distribution mechanism must supply: confidentiality and authenticity in transit, keys unique per peer, unique key identifiers, a revocation path, and key material at least as strong as the primitive it feeds. And it observes that enterprise secret managers already satisfy them, as deployments of the reference implementation do. Per-peer uniqueness deserves emphasis, because violating it degrades the framework silently: a deployment issuing one shared secret to every SAS retains Tier 1 integrity against outsiders but forfeits Tier 2 entirely, since a scope any peer can assert binds nothing. Formal distribution is accordingly the principal item of future standardization, with ACME-style enrolment and bootstrap from an operator's existing identity infrastructure as the candidate directions. Until one is standardized, SESAME's guarantees are conditional on correct out-of-band provisioning.

VIII. CONCLUSION

The ESAM interface is critical infrastructure operating without standardized security at exactly the moment its deployment surface is expanding into shared cloud environments. SESAME closes this gap with a layered framework that respects broadcast latency constraints, preserves backward compatibility, and is supported by an open-source reference implementation. We present it to the SET community in the expectation that Latin American operators and vendors, many at the leading edge of cloud broadcast and FAST-channel adoption, will be among its most direct beneficiaries, and we welcome their participation in the standardization process.

ACKNOWLEDGMENT

The author thanks the colleagues and reviewers at Techex Ltd. and within the SCTE Digital Video Subcommittee for early feedback on the SESAME proposal.

REFERENCES

- [1] ANSI/SCTE 35, *Digital Program Insertion Cueing Message for Cable*, SCTE.
- [2] ANSI/SCTE 130-2 2020, *DPI: Advertising Systems Interfaces Part 2*, SCTE.
- [3] ANSI/SCTE 130-5 2020, *DPI: Advertising Systems Interfaces Part 5*, SCTE.
- [4] ANSI/SCTE 130-7 2020, *DPI: Advertising Systems Interfaces Part 7*, SCTE.
- [5] ANSI/SCTE 130-8 2020, *DPI: Advertising Systems Interfaces Part 8*, SCTE.
- [6] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," IETF RFC 2104, Feb. 1997.
- [7] D. Eastlake, J. Schiller, and S. Crocker, "Randomness Requirements for Security," IETF RFC 4086, Jun. 2005.
- [8] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," IETF RFC 5234, Jan. 2008.
- [9] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," IETF RFC 3986, Jan. 2005.
- [10] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-38D, Nov. 2007.
- [11] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, IETF, Aug. 2018.
- [12] A. Backman, J. Richer, and M. Sporny, *HTTP Message Signatures*, RFC 9421, IETF, Feb. 2024.
- [13] Z. Durumeric *et al.*, "The security impact of HTTPS interception," in *Proc. NDSS*, San Diego, CA, USA, 2017.
- [14] Amazon Web Services, *Signing AWS API Requests: AWS Signature Version 4*, AWS IAM User Guide, 2024.



Bo Kelleher is a Senior Solutions Architect at Techex Ltd. and the developer of the open-source rust-pois project. He has extensive experience in broadcast television infrastructure, SCTE-35 signal processing, and systems-level Rust development. Prior to Techex he held engineering roles at Zixi and Akamai, focused on IP video transport and edge delivery.