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 — 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 demonstrates sub-millisecond per-request overhead on commodity hardware, well within broadcast latency budgets. We present the threat model, protocol design, performance evaluation, and proposed path to standardization within the SCTE 130 family.**

***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 — Secure ESAM Authentication and Message Encryption — a layered security framework designed to close this gap. SESAME has been formally proposed to the SCTE Engineering Committee as SCTE 130-9 and is specified, implemented, and benchmarked here.

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.

II. Threat Model

We identify six concrete attack vectors against unprotected ESAM deployments.

Signal injection allows an unauthorized system on the same network to fabricate ESAM requests and trigger ad breaks in live content. *Signal tampering* permits an intermediary — a misconfigured load balancer, a compromised reverse proxy — to modify acquisition signal IDs, UTC timestamps, or SCTE-35 binary payloads without detection. *Cross-channel interference* exploits the absence of cryptographic binding between requester identity and the channels they are permitted to address, allowing a regional affiliate's encoder to apply transformation rules to a national feed. *Signal reconnaissance* extracts competitive intelligence — break timing, placement metadata, segmentation descriptors — by passive observation of unencrypted ESAM traffic. *Replay attacks* retransmit captured valid requests within the POIS acceptance window, producing duplicate ad insertion. *Application-layer denial of service* floods the POIS with valid-looking ESAM XML that conventional network DoS mitigation cannot distinguish from legitimate traffic.

Each vector is grounded in observable deployment architectures. Table I summarizes the threats together with their SESAME mitigations.

III. Design Constraints

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 — no session establishment, no multi-round-trip handshakes on the request path. Operational management must be tractable for broadcast

TABLE I
SUMMARY OF ESAM THREATS AND SESAME MITIGATIONS.

Threat	Attack Vector	Impact	SESAME Mitigation
Signal Injection	Unauthorized ESAM requests from rogue systems	Unauthorized ad inventory, viewer disruption	Tier 1: HMAC authentication rejects unsigned requests
Signal Tampering	Man-in-the-middle modification of XML payloads	Misdirected ads, timing errors	Tier 1: HMAC integrity verification detects modifications
Cross-Channel	Requests targeting unauthorized channel IDs	Wrong rules applied to wrong inventory	Tier 2: Channel-scoped credentials prevent cross-tenant access
Reconnaissance	Passive observation of ESAM traffic	Competitive intelligence leakage	Tier 3: AES-256-GCM payload encryption
Replay	Retransmission of captured valid requests	Duplicate ad breaks	Tier 1: Timestamp + nonce replay protection
DoS Flooding	Volume-based ESAM request flood	Degraded signal processing	Tier 1: Early rejection of unauthenticated requests

engineers, not security specialists. And the mechanism must operate over standard HTTP without changes to existing ESAM XML schemas.

These constraints rule out several apparent alternatives. Mutual TLS is plausible for encryption but introduces handshake costs that compound across the signal chain. JWT-based bearer tokens lack message-integrity coupling without additional canonicalization. Custom binary protocols violate the schema-stability requirement. 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:

$$\begin{aligned} \textit{canonical} = \textit{method} \,\|\, \mathrm{LF} \,\|\, \textit{path} \,\|\, \mathrm{LF} \,\|\, \textit{timestamp} \\ \|\, \mathrm{LF} \,\|\, \textit{nonce} \,\|\, \mathrm{LF} \,\|\, \textit{body-hash} \\ [\,\|\, \mathrm{LF} \,\|\, \textit{scope}\,] \end{aligned} \tag{1}$$

where *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.

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

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

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. A per-channel security policy supports backward compatibility, allowing channels to accept unauthenticated requests during migration while logging warnings to support adoption monitoring.

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

Benchmarks on an AWS `c6i.xlarge` instance (Intel Xeon Platinum 8375C, 4 vCPU, with AES-NI and SHA extensions) over a 1 KB ESAM payload demonstrate the per-request overhead shown in Table II. All three tiers operate far within the 1 ms latency budget established in Section III; even the combined encrypt-and-sign response path stays under 0.03 ms at 16 KB payloads — well under one percent of the end-to-end ESAM processing budget. The same benchmark on a virtual machine without hardware crypto acceleration remains under 0.22 ms, supporting the deploy-anywhere thesis.

The full paper expands these results with methodology, payload-size sensitivity, and comparative measurements against alternative designs — notably mutual TLS handshake cost amortized across signal-chain hops, and JWT-based bearer-token validation for equivalent integrity coverage.

VII. ADOPTION AND STANDARDIZATION PATH

SESAME is engineered for incremental deployment. The recommended migration begins with the POIS in monitoring-only mode (logging without rejecting), proceeds to per-channel enrollment of clients in priority order, transitions the security policy from optional to required, and finally enables Tier 3 encryption for channels carrying commercially sensitive content. 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. An alternative path — incorporation as a normative annex to a future SCTE 130-7 revision — is also discussed. The current document contains normative requirements for the SESAME Server role; ADS-side normative requirements, mutual authentication, and formal key-distribution protocols are identified as future work.

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 practical, layered framework that respects broadcast latency constraints, preserves backward compatibility, and is supported by an open-source reference implementation. We present this work to the SET community in the expectation that Latin American operators and vendors — many of whom are at the leading edge of cloud broadcast and FAST-channel adoption — will be among the most direct beneficiaries of the framework, 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*, Society of Cable Telecommunications Engineers.
[2] ANSI/SCTE 130-2 2020, *Digital Program Insertion: Advertising Systems Interfaces Part 2 — Core Data Elements*, Society of Cable Telecommunications Engineers, 2020.
[3] ANSI/SCTE 130-5 2020, *Digital Program Insertion: Advertising Systems Interfaces Part 5 — Placement Opportunity Information Service*, Society of Cable Telecommunications Engineers, 2020.
[4] ANSI/SCTE 130-7 2020, *Digital Program Insertion: Advertising Systems Interfaces Part 7 — Message Transport*, Society of Cable Telecommunications Engineers, 2020.
[5] ANSI/SCTE 130-8 2020, *Digital Program Insertion: Advertising Systems Interfaces Part 8 — General Information Service*, Society of Cable Telecommunications Engineers, 2020.
[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.