Redacted Report Template: Disclosure Control and Release Audit

A public exemplar for sanitized release workflows

Daniel Ari Friedman
Active Inference Institute
daniel@activeinference.institute
ORCID: 0000-0001-6232-9096
DOI: 10.5281/zenodo.21298890

2026-07-10

Contents

1 Abstract

This exemplar demonstrates a complete disclosure-control pipeline for sanitized public release reports. The methodology combines classification-ceiling enforcement, source-protection validation, mosaic-risk scoring, and TPM-backed sealed sidecars across a sixteen-variant visual proof matrix. Four redaction styles—blackout, whiteout, grayout, and blur—are rendered across four PDF backgrounds—white, gray, black, and blur—yielding sixteen base proof PDFs. Each receives nine steganographic security methods including SHA-256/SHA-512 hash manifests, diagonal watermark overlays, footer provenance stamps, invisible text, QR and Code128 barcodes, PDF Info and XMP metadata, and embedded manifest attachments. Optional Kmyth TPM sealing wraps each hash manifest and steganography PDF in a `.ski` sidecar sealed against the TPM2-TSS storage hierarchy, bound to PCR selections and policy or-values. The release gate requires three reviewer roles—originator, classification reviewer, and release authority—each providing a non-empty rationale. A source-safe redaction ledger records SHA-256 hashes of each redacted span without exposing source text, and a segment hash manifest compares source and public SHA-256 digests for reproducible audit. The comprehensive release packet combines sanitized text, audit findings, ledger, hashes, review gate status, and paragraph-level audit tables into a single JSON-ready export. This exemplar confirms that visual presentation choices remain orthogonal to the release gate: the same source-safe decisions drive every output variant.

2 Introduction

Disclosure control—the process of sanitizing classified or sensitive information before public release—requires multiple layers of validation to ensure that no source identities, operational details, time-place selectors, or controlled dissemination markers leak into the public record. This exemplar implements a reproducible pipeline that combines text-level disclosure control with visual redaction proofing, steganographic provenance embedding, and hardware-backed TPM sealing.

The pipeline operates on invented fixture data: synthetic segments with classification levels ranging from UNCLASSIFIED through TOP_SECRET//SCI, synthetic redaction decisions with bounded reasons, and synthetic reviewer records. No real source material is used. The exemplar demonstrates the full release-review workflow: classification-ceiling enforcement, redaction-span validation, source-control coverage, mosaic-risk scoring, residual-marker detection, review-gate evaluation, source-safe ledger generation, and TPM-sealed sidecar production.

2.1 Scope and Safety Boundaries

This exemplar is limited to lawful redaction, declassification support, public-records release review, taxonomy normalization, source-safe ledgers, reviewer approval gates, sanitized packet export, and source-protection auditing. It does not provide targeting, collection, evasion, or surveillance operational guidance. All committed examples are invented fixtures.

2.2 Contributions

1. A classification taxonomy with SCI alias support and configurable public ceilings.
2. A release-audit engine that validates redaction spans, checks orphan decisions, enforces source-control coverage, and scores mosaic risk.
3. A source-safe redaction ledger that records SHA-256 hashes of redacted spans without exposing source text.
4. A visual proof matrix that enumerates four redaction styles across four PDF backgrounds, producing sixteen variant PDFs with identical source-safe decisions.
5. A steganography layer that post-processes each base PDF with nine security methods, producing provenance-enhanced companion PDFs with hash manifests.
6. Optional Kmyth TPM sealing that wraps each hash manifest and steganography PDF in `.ski` sidecars sealed against the TPM2-TSS storage hierarchy.
7. A comprehensive release packet that combines sanitized text, audit findings, ledger, hashes, review gate status, and paragraph audit tables.

3 Architecture

The pipeline architecture consists of two layers. Layer one performs text-level disclosure control: each segment is audited against a public classification ceiling, redaction spans are validated for non-overlap, orphan decisions are flagged, and source-control coverage is enforced. Layer two applies visual redaction treatments, steganographic provenance overlays, and optional Kmyth TPM sidecar sealing.

3.1 Layer One: Disclosure Control

The disclosure-control engine operates on `RedactionSegment` objects, each carrying an identifier, classification level, text content, and optional source controls. Redaction decisions are `RedactionDecision` records with bounded reasons: `source_identity`, `operational_detail`, `time_place_selector`, `legal_privilege`, and `privacy`. The audit engine validates that:

- Redaction spans are non-overlapping and within text bounds.
- Each segment above the public ceiling has at least one redaction decision.
- Each source control has a corresponding `source_identity` redaction.
- No orphan decisions reference missing segments.
- Residual markers (email, IP, coordinates, NOFORN, HUMINT, SIGINT) are absent from sanitized text.
- The mosaic risk score—residual markers normalized by segment and pattern count—does not exceed the policy threshold.

3.2 Layer Two: Visual Proof and Steganography

Visual proofing is parameterized separately from the release-audit text path. Four redaction styles—blackout (solid black fill), whiteout (white fill with gray text), grayout (mid-gray fill), and blur (offset-rendered token)—are rendered across four PDF backgrounds—white, gray, black, and blur (subdued text with blur effect). The 4×4 matrix yields sixteen variant PDFs, each receiving the same source-safe redaction decisions.

The steganography layer post-processes each base PDF with nine security methods:

Method	Purpose
SHA-256/SHA-512 hash manifest	Cryptographic integrity verification
Diagonal watermark overlay	Visible provenance stamp
Footer provenance overlay	Page-level release metadata
First-page invisible text	Hidden identification marker
QR payload barcode	Machine-readable provenance
Code128 page barcode	Per-page tracking barcode
PDF Info metadata	Document-level metadata injection
XMP metadata	Standards-compliant metadata embedding
Embedded stego manifest attachment	Self-contained provenance archive

3.3 Layer Three: Kmyth TPM Sealing

Kmyth TPM sealing wraps each hash manifest and steganography PDF in a `.ski` sidecar sealed against the TPM2-TSS storage hierarchy. The sealed objects are bound to PCR selections and policy or-values, ensuring that unsealing requires the same platform configuration.

On macOS, which lacks a hardware TPM, a software TPM emulator (swtpm) and an mssim-to-swtpm protocol proxy bridge the TPM2-TSS mssim TCTI to swtpm's native socket protocol. The proxy:

- **Control channel**: intercepts MS simulator platform commands (POWER_ON=1, NV_ON=11, TPM_SESSION_END=20) and returns success, since swtpm's `--flags startup-clear` handles TPM initialization.
- **Data channel**: strips the 9-byte mssim wire header (4B cmd_type + 1B locality + 4B tpm_size), forwards raw TPM commands to swtpm, and wraps responses back in mssim format.

The kmyth-seal binary was patched to call `Tss2_Sys_FlushContext` for the storage key handle before freeing TPM2 resources. Without this patch, consecutive kmyth-seal invocations exhaust the swtpm transient object slots (typically three), causing "out of memory for object contexts" errors on the second seal.

4 Results

4.1 Fixture Release Packet

The fixture release packet contains fourteen segments spanning four classification levels: UNCLASSIFIED (ten segments), CUI (one segment), SECRET (two segments), and TOP_SECRET (one segment). Three segments carry source controls (HUMINT, SIGINT, IMINT). Fifteen redaction decisions are applied across four segments, using all five bounded reasons: `source_identity`, `operational_detail`, `time_place_selector`, `legal_privilege`, and `privacy`.

The audit produces:

- **Releasable**: true (no error-level findings after redaction).
- **Release safety score**: weighted by error count, warning count, and mosaic risk.
- **Redaction coverage**: 1.0 (all sensitive segments have at least one decision).
- **Mosaic risk score**: residual markers normalized by segment and pattern count.
- **Findings**: warning-level findings for residual markers in sanitized text.

4.2 Source-Safe Redaction Ledger

The redaction ledger records each decision with:

- A decision ID derived from the SHA-256 of the segment ID, span, reason, and replacement.
- The segment ID, start, end, span length, reason, and replacement.
- A `valid_span` flag indicating whether the span falls within the segment text.
- A `source_span_sha256` hash of the redacted text—present only for valid spans, absent for invalid or orphan decisions.

The ledger never exposes source text. It provides reproducible audit evidence without disclosure risk.

4.3 Segment Hash Manifest

The hash manifest records, for each segment:

- `source_sha256`: SHA-256 of the original segment text.
- `public_sha256`: SHA-256 of the redacted segment text.

This allows downstream verification that the public release matches the audited version without comparing source text directly.

4.4 Review Gate

The release gate requires three reviewer roles: originator, classification reviewer, and release authority. Each reviewer provides a non-empty rationale. The fixture reviews all approve, yielding:

- **Approved**: true.
- **Approval count**: 3.
- **Required roles present**: classification_reviewer, originator, release_authority.
- **Findings**: none.

The `final_release_recommended` flag is true only when the packet is releasable, the review gate is approved, and no blocking warnings remain.

4.5 Visual Proof Matrix

The development proof matrix produces sixteen base PDFs (four redaction styles × four backgrounds) and, when steganography is enabled, sixteen companion steganography PDFs with hash manifests. Each variant records:

- Base PDF filename, byte size, and SHA-256.
- Steganography PDF filename, byte size, and SHA-256.
- Hash manifest filename and SHA-256.
- Kmyth sidecar count and filenames (when Kmyth is available).

4.6 Kmyth TPM Sidecar Production

When Kmyth tools are runnable and a TPM backend is available, each variant produces two `.ski` sidecars:

1. `{variant_id}.hashes.json.ski` — the hash manifest sealed against the TPM storage hierarchy.
2. `{variant_id}_steganography.pdf.ski` — the steganography PDF sealed against the TPM storage hierarchy.

The `.ski` files are ASCII-armored and contain PCR selections, policy or-values, the storage key public area, and the sealed data object. Unsealing requires the same TPM and platform configuration.

In the verified run, all sixteen variants produced both sidecars, yielding thirty-two `.ski` files total. The kmyth-seal binary's FlushContext patch ensured that consecutive seal invocations did not exhaust the swtpm transient object slots.

4.7 Residual Risk Detection

The residual-risk detector scans sanitized text for common public-release leaks:

Pattern	Regex
Email address	`\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b`
IPv4 address	`\b(?:\d{1,3}\.){3}\d{1,3}\b`
Coordinate pair	`\b-?\d{1,2}\.\d{3,},\s*-?\d{1,3}\.\d{3,}\b`
Controlled dissemination	`\b(?:NOFORN|ORCON|REL\s+TO)\b`
Collection discipline	`\b(?:HUMINT|SIGINT|IMINT|MASINT|OSINT)\b`
Compartment marker	`\b(?:SCI|TS_SCI|TOP\s+SECRET)\b`
Sensitive markers	`HUMINT`, `SIGINT`, `source`, `selector`, `location`, `2026-`

Each detected marker generates a warning finding. The mosaic risk score aggregates residual markers across all segments.

5 Discussion

The exemplar demonstrates that disclosure control can be decomposed into orthogonal concerns: text-level audit, visual presentation, steganographic provenance, and hardware-backed sealing. Each concern is independently configurable and verifiable.

The visual proof matrix confirms that blackout, whiteout, grayout, and blur treatments produce equivalent source-safe outputs—only the visual token differs. The steganography layer adds provenance without altering the redaction decisions. Kmyth TPM sealing adds a hardware binding that ensures sealed sidecars can only be unsealed on the same platform configuration.

The mssim-to-swtpm protocol proxy is a necessary bridge on macOS, where no hardware TPM exists. The proxy handles three protocol mismatches: (1) platform commands use different command numbers, (2) the data channel wraps TPM commands in a 9-byte mssim header, and (3) swtpm's transient object slots are limited and must be flushed between seal invocations.

The FlushContext patch to kmyth-seal is a critical fix for batch sealing workflows. Without it, the second kmyth-seal invocation fails with "out of memory for object contexts" because the storage key from the first invocation remains loaded in the TPM's transient object table.

5.1 Limitations

- The fixture data is invented; real release packets may require additional classification taxonomies and review-role policies.
- The swtpm software TPM does not provide hardware-level tamper resistance; production deployments should use a hardware TPM.
- The mssim proxy adds latency to each TPM command; batch sealing of thirty-two sidecars takes approximately thirty seconds.
- PDF password encryption is optional and uses AES-256; the password is not stored in the variant matrix.

References