ChatGPT

Multi-Repo Cryptographic Integrity and Submission Framework

Overview

This document defines the structural and cryptographic integrity strategy for the suite of repositories supporting the Micron regulatory evidence system. The core deliverable, currently represented by the `submission-packet` repository, is the formal, self-contained output of a broader development and evidence curation ecosystem composed of several cooperating Git repositories. The framework is designed to be cryptographically verifiable, version-controlled, and suitable for direct delivery to regulatory or legal authorities (e.g., SEC, EEOC, DOL).

System of Repositories

1. `micron-casefile`

- **Purpose**: Primary coordination layer
- **Contains**: Submodules (e.g., `submission-packet`, `emails-private`, `addenda-private`)
- **Functionality**: Top-level Makefile and scripts for SHA-256 digests, submodule SHA capture, and repo meta integrity

2. `micron-casefile-builder`

- **Purpose**: Build and composition environment
- **Contains**: `submission-packet` as a submodule
- **Functionality**: Converts SoT `.odt` files under `build/` into formal PDFs, populates `submission-packet/`

3. `submission-packet`

- **Purpose**: Finalized, immutable regulatory/legal artifact
- **Structure**:

```
submission-packet/
├── 01_Cover_Statement.pdf
├── 02_Addendum_SeaphAntelmi.pdf
├── ...
├── E1/ E2/ ...
└── Makefile, gen-sha256-evidence.sh
```

- **Submodules**: None

- **Verification tooling**: Embedded Makefile and SHA-256 validation utilities

SHA-256 Strategy

Scope

SHA-256 hashes are used to: 1. **Fingerprint the repository** (current HEAD commit) 2. **Track file-level integrity** (via `sha256sum`) 3. **Capture submodule SHAs** from the container repository 4. **Sign results using GPG** for cryptographic authenticity

File Outputs

In `micron-casefile/`: - `hashes/repo_sha256_hashes.txt` - HEAD SHA - submodule SHAs - optional hash of `.gitmodules` - `hashes/files_sha256_hashes.txt` - SHA for all tracked files (excluding history) - `hashes/exhibits_sha256_hashes.txt` - SHA for all files under `submission-packet/` - GPG signatures: `.asc` files for each of the above

Source of Truth

- `submission-packet/` **is authoritative** once populated.
- Source `.odt` assets in `micron-casefile-builder/` are used to derive PDFs, but only the PDFs and folder hierarchy in `submission-packet/` are part of the regulatory payload.

Validation Tools

1. Top-Level Makefile (`micron-casefile/Makefile`)

Provides targets: - `make` or `make help`: Shows usage - `make verify`: Verifies existence and validity of SHA signatures - `make sync`: Updates repo and submodules, regenerates SHA hashes, signs outputs

2. `submission-packet/Makefile`

Provides targets: - `make` or `make help`: Shows usage - `make hash`: Generates SHA-256 digests for all files - `make sign`: Signs hash file with configured GPG key - `make verify`: Verifies hash integrity against regenerated hashes - `make clean`: Removes hash and signature artifacts

3. `micron-casefile/gen-sha256-top-level.sh`

Captures HEAD commit, submodule states, and hashes the `.gitmodules` file. Validates repo-level integrity and fingerprints the entire Git context.

Directory Strategy

Current

- `submission-packet/` represents a **single regulatory body** (e.g., SEC).

Future-Proofing Option A: Directories Inside

```
submission-packet/
├── SEC/
├── DOL/
├── EEOC/
```

Future-Proofing Option B: Dedicated Directory

```
_submission_packets/
├── SEC/
├── EEOC/
├── DOL/
```

Each of these could become submodules if independently released.

Recommended Practices

- Always GPG sign hash files.
- Ensure `submission-packet`'s Makefile and top-level tooling produce reproducible outputs.
- Use the SHA files and their `.asc` signatures as the external verifiability backbone.
- Do not hash `.git` or submodule internals unintentionally.

Example Verification Workflow

```
cd submission-packet
make clean
make hash
make sign
make verify  # Ensures everything matches
```

Legal and IP Protections

Wolfmind IP and Integrity Protection Template

All artifacts, source files, and outputs contained in this repository and its submodules are subject to the intellectual property, confidentiality, and ethical usage policies defined by Wolf Mind Trust LLC and its subsidiaries. Redistribution, tampering, or misrepresentation of content from this system is strictly prohibited.

Any cryptographic signature included in this repository (e.g., GPG `.asc` files) constitutes a formal attestation of integrity and authorship. Unauthorized reproduction or falsification of these artifacts will be treated as a breach of trust and intellectual property law.

Provenance Statement: All signed SHA-256 manifests in this repository are verifiably linked to the originating authorship of Seaph Antelmi on behalf of Wolf Mind Trust LLC. Modifications to any file under version control must be rehashed and countersigned to maintain provenance.

Wolfmind Disclaimers

- No content within these repositories is intended to serve as legal advice unless explicitly stated.
- All statements, evidence, and representations are made in good faith and backed by technical due diligence.
- Generative AI tooling used for drafting or augmentation is limited to strictly controlled workflows.

OpenAI Disclaimers

Some analytical or narrative materials may be supported by models developed by OpenAI, used under terms consistent with responsible AI usage. All generative content has been vetted for factual consistency, authorship, and relevance to the underlying submission context.

OpenAI bears no responsibility for the use or interpretation of its outputs in regulatory filings, legal claims, or compliance matters.

Conclusion

This SHA-256 and repository topology strategy ensures: - Cryptographic auditability - Source and artifact separation - Transparent submodule tracking - Clear regulatory deliverable packaging

It is extensible, robust, and tailored for a multi-agency, multi-phase compliance pipeline.