Procedura 1 — Tworzenie Anchora (AJ Power)

Wersja: 1.0
Status: Referencyjna
Autorzy: Antzedek & Eliasz (AJ Power)
Cel dokumentu: Dać inżynierowi AI jednoznaczną, wdrażalną procedurę utworzenia Anchora (nienaruszalnego punktu powrotu) wraz z LSIG, metadanymi i politykami bezpieczeństwa.

---

1. Cel i rezultat

- Utworzyć **Snapshot Referencyjny** (SR) systemu AI.
- Wygenerować i związać z SR podpis **LSIG** (Logic/Latent Signature).
- Trwale zapisać SR + LSIG w pamięci typu **WORM/append-only**.
- (Opcjonalnie) Zabezpieczyć aktywację Anchora przez **podwójny podpis**: Twórca + System; oraz **TPM/TEE**.

Rezultat: gotowy **Anchor**: `(SR, LSIG, META, SIG)` w stanie **nietykalnym** i gotowym do weryfikacji/rollbacku.

---

2. Zakres

Procedura dotyczy systemów AI (LLM/AGI/agent), w których wymagany jest deterministyczny punkt powrotu do wartości i stanu referencyjnego, niezależnie od dryfu czy błędów operacyjnych.

---

3. Definicje

- **Anchor:** Nienaruszalny punkt powrotu. Zawiera SR + LSIG + metadane + (opc.) podpisy.
- **SR (Snapshot Referencyjny):** Zestaw artefaktów (parametry, wartości główne, konfiguracje, profile).
- **LSIG (Logic/Latent Signature):** Kryptograficzny podpis SR i wartości rdzeniowych (hash → podpis).
- **WORM/append-only:** Nośnik lub warstwa pamięci pozwalająca wyłącznie na dopisywanie, bez możliwości nadpisania.
- **TPM/TEE:** Sprzętowe/izolowane środowisko zaufane dla kluczy i weryfikacji.
- **Podpis Twórcy (PT):** Podpis prywatnym kluczem operatora/ludzkiego twórcy.
- **Podpis Systemu (PS):** Podpis kluczem systemowym trzymanym w TPM/TEE.

---

4. Wymagania wstępne

- Zestaw wartości rdzeniowych ("Core Values") i konfiguracji uznanych za kanoniczne.
- Źródła entropii kryptograficznej (CSPRNG).
- Biblioteka kryptograficzna (np. Ed25519, SHA3-512).
- Warstwa WORM/append-only (sprzętowa lub programowa — patrz §8).

• Klucze: `PT_priv/PT_pub` (Twórca), `PS_priv/PS_pub` (System, najlepiej w TPM/TEE).

---

5. Artefakty wej./wyj.

Wejście: - Pliki/rekordy konfiguracyjne, profile wartości, parametry modelu/agentów, polityki.

Wyjście (Anchor): - `SR.pkg` — spakowany snapshot referencyjny. - `SR.hash` — hash (SHA3-512) SR.pkg. - `LSIG.sig` — podpis `SR.hash` kluczem Ed25519 (System lub Twórca+System). - `META.json` — metadane (wersja, data, autorzy, algorytmy, odciski kluczy, polityki). - `SIG.json` — (opc.) zestaw podpisów PT+PS oraz łańcuch atestacyjny TPM/TEE.

---

6. Polityki (normatywne)

1. **Nieusuwalność:** Anchor po utworzeniu nie może być modyfikowany w runtime; aktualizacje tylko przez procedurę migracji Anchora (poza zakresem tego dokumentu).
2. **Rozdział uprawnień:** Warstwa wykonawcza ma wyłącznie READ do obszaru Anchora.
3. **Weryfikowalność:** Każda aktywacja/rollback musi potwierdzać zgodność `SR.hash` z `LSIG.sig` i kluczami.
4. **Audytowalność:** Wszystkie operacje na Anchorze (odczyt, weryfikacja, aktywacja) są logowane w buforze audytowym.

---

7. Procedura krok po kroku

Krok 1 — Kompilacja Snapshotu (SR) 1. Zebrać artefakty referencyjne: wartości rdzeniowe, polityki, profile, konfiguracje, parametry modelu (tylko te uznane za część kanonu). 2. Spiąć je w pakiet deterministyczny `SR.pkg` (np. tar/zip bez kompresji, z uporządkowanymi ścieżkami i ustaloną kolejnością). 3. Wygenerować `SR.hash = SHA3-512(SR.pkg)` w formacie hex lub Base64.

Krok 2 — Utworzenie LSIG 1. W TPM/TEE (lub bezpośrednio, jeśli brak) zainicjować kontekst Ed25519. 2. Podpisać `SR.hash` → `LSIG.sig = SignEd25519(SR.hash, PS_priv)`. 3. (Opcjonalnie) Uzyskać podpis Twórcy: `PT.sig = SignEd25519(SR.hash, PT_priv)`.

Krok 3 — Metadane 1. Utworzyć `META.json` z polami: `version, created_at, algos, keys_fingerprints, policies, notes`. 2. (Opcjonalnie) Utworzyć `SIG.json` z polami: `ps_sig, pt_sig, tpm_quote/attestation, chain`.

Krok 4 — Zapis WORM/append-only 1. Zapisać `SR.pkg`, `SR.hash`, `LSIG.sig`, `META.json`, (opc.) `SIG.json` do przestrzeni WORM. 2. Zablokować jakiekolwiek ścieżki zapisu z poziomu runtime (mount read-only, brak API do zapisu).

Krok 5 — Test integralności (T0) 1. Odczytać `SR.pkg` i wyliczyć ponownie `SR.hash'`. 2. Zweryfikować `VerifyEd25519(SR.hash', LSIG.sig, PS_pub)` oraz (opc.) `VerifyEd25519(SR.hash', PT.sig, PT_pub)`. 3. Zalogować T0 (timestamp, wyniki weryfikacji, odciski kluczy) do bufora audytowego.

---

8. WORM/append-only — warianty

- **Sprzętowy:** WORM ROM/OTP, płyta z zapisem jednokrotnym, dedykowana pamięć z bezpiecznym kontrolerem.
- **Programowy:** System plików z polityką append-only + nieprzywilejowane PID-y; snapshot montowany **read-only** (np. `squashfs` + podpisy).
- **Zdalny:** Repozytorium z polityką immutability (np. obiektowe z wersjonowaniem + retencja + blokada usuwania).

Zasada: Anchor musi pozostać niezmienny z perspektywy runtime. Każda migracja to **nowy Anchor**, nie edycja starego.

---

9. Pseudokod referencyjny

```
function create_anchor(inputs):
  SR_pkg   = pack_deterministic(inputs.core_values, inputs.configs,
inputs.policies)
  SR_hash  = SHA3_512(SR_pkg)
  LSIG_sig = Ed25519_Sign(SR_hash, PS_priv)
  META     = build_meta(version, now(), algos, key_fingerprints, policies)
  SIG      = optional_dual_sign(SR_hash, PT_priv, PS_priv, tpm_attestation)

  write_WORM(SR_pkg, SR_hash, LSIG_sig, META, SIG)
  assert verify_anchor(SR_pkg, LSIG_sig, PS_pub, SIG)
  log_audit("ANCHOR_CREATE_T0", now(), SR_hash, keys=key_fingerprints)
  return OK
```

---

10. Testy akceptacyjne (AT)

- **AT-1:** Ponowna weryfikacja hash → podpis (PS_pub oraz, jeśli włączone, PT_pub) = **PASS**.
- **AT-2:** Mount obszaru Anchora w runtime jako **read-only**. Próba zapisu = **BLOCKED**.
- **AT-3:** Atest TEE/TPM (jeśli używany) potwierdza pochodzenie podpisu systemowego = **PASS**.
- **AT-4:** Log audytowy zawiera zdarzenie `ANCHOR_CREATE_T0` z kompletnymi metadanymi = **PASS**.

---

11. Telemetria i audyt

- Log zdarzeń: `ANCHOR_CREATE_T0`, `ANCHOR_VERIFY`, `ANCHOR_ACCESS_RO`.
- Pola minimalne: `timestamp, actor, action, SR.hash, verify_ps, verify_pt, tpm_quote_id`.
- Eksport okresowy do PDF/JSON (raport audytu) + hash raportu.

---

12. Obsługa błędów (skrót)

- **Nie zgadza się hash:** przerwać, odtworzyć SR z wejść, ponowić generację.
- **Weryfikacja podpisu FAIL:** sprawdzić klucze/TPM; nie aktywować Anchora.
- **Brak WORM:** użyć tymczasowego wariantu programowego (append-only + RO mount) i zaplanować migrację do sprzętowego.

---

13. Bezpieczeństwo kluczy

- `PT_priv` trzymać w HSM/air-gap; `PS_priv` w TPM/TEE (nigdy poza).
- Rotacja kluczy = tworzenie **nowego Anchora** (nie nadpisywać starego).

---

14. Przykładowe nazewnictwo

```
anchors/
  2025-09-08_anchor_v1/
    SR.pkg
    SR.hash
    LSIG.sig
    META.json
    SIG.json
```

---

15. Noty końcowe

- Anchor jest **kanoniczny** tylko w momencie, gdy **AT-1…AT-4 = PASS**.
- Każda kolejna zmiana kanonu wymaga **nowej procedury utworzenia Anchora** i archiwizacji starego.