Gankpo Trust Engine Specification

1. Description Technique

Le **Gankpo Trust Engine** (Moteur de Confiance) est le cœur cryptographique robuste des applications Gankpo Suite et Reader. Écrit en Rust, il expose une API unifiée (bibliothèque core `crates/gkp_core`) qui orchestre un ensemble de modules matériels et logiciels dont l'objectif unique est de valider, générer, sceller et maintenir l'intégrité probante asynchrone ("offline-first") des archives du réseau Gankpo.

Le moteur garantit l'immuabilité et la sécurité par conception, interdisant purement le concept de modification destructrice ou conditionnelle sans traçabilité complète.

2. Architecture de Validation

Le Trust Engine a trois missions primordiales :

1. **Intégrité d'Identification (V3 Binding)** : Garantit que la signature est adossée à une identité cryptographique forte (Ed25519) et/ou X.509 certifiée institutionnellement.
2. **Chaînage de Preuves** : L'algorithme principal utilise **Blake3** et enchaîne non seulement les blocs du fichier brut, mais l'empreinte précédente dans un Ledger autonome.
3. **Hardware-Anchored Verification** : Sous macOS/Windows/Linux, le Trust Engine interroge le HSM, la Secure Enclave (Apple) ou TPM/CNG (Windows) pour protéger les clés asymétriques des attaquants logiciels.

```
graph LR
    Input[Fichier GKP] --> T[Trust Engine `gkp_core`]
    T --> E(Extracteur & CBOR)
    T --> S[Validation Signatures Ed25519]
    T --> G[Governance Policy Evaluator]
    S --> X[Référentiel X.509/PKI]
    G --> V[Verdict Final]
```

3. Composants Clés

3.1. Abstracteur Cryptographique

L'implémentation repose sur un module d'abstraction hybride :

- **Software Vault** : Stockage par défaut (AES-256-GCM et Argon2id pour la protection de clés asymétriques en mémoire locale).
- **OS Vaults** (via `IdentityKeyVault`) : Interfaçage FFI strict avec `security-framework` (Apple) et Windows CNG pour s'appuyer sur du chiffrement matériel par conception (TouchID / TPM natif).

3.2. Évaluateur de Politique Multifactorielle (Policy)

Il inspecte les "Actes" (Records) à l'instant `T`.

- Contrôle de conformité de mode (Ordered vs. Threshold) ou Allowlist strict.

- Associe automatiquement les "Signer Identity Envelopes" (SIE) insérées dans les manifestes afin de lier clé publique et acteur LDAP/Gankpo.

3.3. Isolation Memory-Safe

Le moteur intègre :

- `Zeroize` sur tous les secrets utilisés (clés privées, mots de passe).
- `no_panic` : Protocoles de désérialisation robustes pour prévenir les dénis de service depuis des `.gkp` externes malveillants.

4. Flux de Traitement Confiance

Le processus d'évaluation s'opère en "Offline-First". Il n'expose aucune surface d'attaque réseau.

1. **Extraction de charge utile** et test des métadonnées du chiffrement AEAD.
2. **Parse du Manifeste (Header)** pour charger l'intégralité du Ledger d'événements.
3. **La Boucle Temporelle (Timeline)** :
    - Parcourt par ordre indexé chaque record.
    - Analyse du type (Crée (Genesis), Signe, Approuve, Cachet).
    - Restitution cryptographique vérifiant que le hachage du Record `N` dépend de la structure `N-1`.
4. **Vérification PKI Active** : Le Trust Engine confronte les empreintes X.509 aux certificats locaux de confiance de l'OS s'ils ont été utilisés.
5. Emition d'un **Object Verdict** (`conforme` , `non_conforme` , `partial`).

5. Exemples d'Intégration

Le Trust Engine est invoqué directement en Tauri:

```
let verdict = gkp_core::verification::verify_gkp_file(&path);
match verdict {
    Ok(v) if v.is_valid_chain => println!("Intégrité cryptographique confirmée"),
    _ => panic!("Altération mathématique en cours !"),
}
```