

Des logs IA aux dossiers de décision vérifiables

Capture opposable, continuité de population et vérification hostile dans une frontière agentique déclarée

VERIFIED $\neq$ VRAI

Résumé

Les systèmes d'IA ne se contentent plus de produire du texte. Ils appellent des modèles, des outils et des API, délèguent du travail à d'autres agents, déplacent des fichiers entre plusieurs états applicatifs et peuvent déclencher des actions ayant des conséquences juridiques, financières ou opérationnelles.

L'observabilité classique reste nécessaire pour exploiter ces systèmes. Mais elle ne répond pas à une question d'audit plus exigeante : un examinateur peut-il détecter que l'historique qui lui est présenté a été modifié, tronqué, sélectivement raccourci après admission, ou laissé dans un état non résolu ?

KOREV AI a développé et testé une réponse bornée à cette question dans KOREV Evidence / PRISM-Oracle. L'implémentation enregistre une admission avant l'exécution, engage des événements ordonnés et les empreintes de leurs payloads dans un journal durable, exige des acquittements d'un témoin doté d'une clé distincte sur des frontières déclarées, puis exporte des dossiers de preuve signés qu'un vérificateur classe en VERIFIED, INCOMPLETE ou FAILED.

La revendication centrale n'est pas que le système prouve la vérité. Elle est plus étroite et testable : lorsqu'une admission a franchi une frontière déclarée et instrumentée puis a été reconnue par le témoin configuré, cette admission doit continuer à être représentée et doit se réconcilier avec un état terminal avant que la population puisse être considérée comme complète.

Ce travail dépasse le seul protocole cryptographique. Des adaptateurs runtime couvrent des frontières sélectionnées autour des modèles, outils, sorties UI et HTTP. La transition applicative Conversation → Dossier a en outre été durcie contre la substitution de fichiers acceptés, les courses d'admission concurrentes, les exécutions incertaines, les acquittements tardifs, les contenus actifs historiques et les uploads orphelins. Ces corrections applicatives ont fait l'objet d'une relecture hostile dans la PR #30 et ont été intégrées aux contrôles CI bloquants.

Le résultat doit donc être compris comme une **capture vérifiable à l'intérieur de frontières déclarées**. Il ne s'agit ni d'une certification, ni d'une preuve de conformité réglementaire, ni d'une preuve de vérité factuelle, ni d'une preuve d'approbation humaine, ni d'une affirmation d'exhaustivité globale de toutes les actions possibles en production.

1. Le problème : un historique peut être cohérent et pourtant incomplet

Un log classique répond à la question : « qu'est-ce que le système a enregistré ? »

Un audit pose une question plus forte : « qu'est-ce qui aurait dû apparaître dans l'historique, et la partie qui le produit peut-elle supprimer silencieusement une partie de cet historique ? »

Des signatures au niveau de chaque enregistrement ne suffisent pas à résoudre ce problème. Un opérateur peut supprimer une fin de chaîne gênante, reconstruire un historique cohérent, puis le signer de nouveau avec une clé qu'il contrôle.

Une chaîne de hachage permet de détecter une modification relativement à une chaîne connue. Mais, à elle seule, elle ne permet pas à un tiers de savoir si la chaîne a été raccourcie avant de lui être présentée.

Trois propriétés doivent donc être distinguées :

1. **Intégrité des enregistrements** — les octets présentés plus tard sont bien ceux qui ont été engagés.
2. **Continuité de l'historique** — l'ordre et les liens entre événements n'ont pas été silencieusement réécrits ou tronqués par rapport à une référence retenue.
3. **Complétude de population après admission** — toute admission reconnue par le témoin reste représentée et réconciliée avant que la population ne puisse être acceptée comme complète.

La troisième propriété est au cœur de ce travail.

La question de recherche est la suivante :

Un opérateur qui contrôle son runtime et son journal local peut-il fournir à un examinateur des éléments permettant de détecter l'altération ciblée, la troncature, l'omission après admission et l'incomplétude, sans imposer à cet examinateur de faire confiance à la base locale du producteur ?

Pour le chemin déclaré et instrumenté, sous le modèle de menace et les limites décrits ici, l'implémentation et les expériences soutiennent une réponse conditionnelle : **oui**.

2. Ce que KOREV a réellement construit

Le travail ne repose pas sur une primitive cryptographique isolée. Il repose sur une chaîne de contrôles qui doivent rester cohérents entre eux.

2.1 Admission durable avant exécution

En mode de capture obligatoire, certaines admissions et opérations sont persistées avant toute exécution. La transaction locale stocke l'identité, les métadonnées d'entrée et les engagements de payload avant que le runtime ne poursuive.

Un crash entre l'admission et la fin de l'exécution laisse donc une admission non résolue plutôt qu'une action invisible.

2.2 Journal de décision ordonné

Chaque événement transporte notamment :

- un identifiant de protocole et de scope ;
- un numéro de séquence ;
- l'empreinte de l'événement précédent ;
- des métadonnées d'état et de résultat ;
- l'empreinte d'un payload conservé séparément.

L'API locale est append-only par contrat, mais KOREV ne décrit **pas** la base SQLite sous-jacente comme physiquement immuable. Un producteur capable de réécrire son stockage local fait explicitement partie du modèle de menace.

2.3 Témoin à clé distincte

Un processus témoin détient une clé de signature différente et un état persistant propre. Il n'accepte qu'une extension exacte de l'historique qu'il reconnaît déjà et retourne un checkpoint signé liant le scope, la séquence, la tête courante, le temps et le challenge.

Dans les expériences rapportées, le témoin est séparé en processus et en clé, mais reste sous la même administration locale. L'architecture permet une administration séparée ; les expériences ne revendiquent pas encore une indépendance opérationnelle.

2.4 Fraîcheur et références retenues

Un vérificateur n'accepte pas un ancien checkpoint signé uniquement parce que sa signature est valide. Il vérifie la fraîcheur, le challenge et l'état courant du témoin.

Un bundle historiquement valide peut donc échouer face à un état de témoin plus récent.

2.5 Réconciliation plutôt que disparition

Une admission peut se terminer en succès, refus, erreur, annulation, état superseded ou autre état terminal explicite.

Une admission interrompue reste visible comme non résolue tant qu'elle n'a pas été réconciliée.

Le protocole distingue donc « rien ne s'est passé » de « une opération admise n'a pas atteint un état terminal connu ».

2.6 Capture à la frontière de libération

Le runtime étend le même principe à certaines sorties. Résultats de modèles/providers, résultats et erreurs d'outils, sorties visibles par l'utilisateur et certains octets de réponse HTTP peuvent être persistés et acquittés avant libération.

Cela prouve ce que la frontière instrumentée a proposé de libérer. Cela ne prouve pas la réception finale par un client aval.

2.7 Export signé et surface de vérification tierce

Le bundle de preuve exporté combine la transcription ordonnée des événements avec un Decision Evidence Pack dont la signature du producteur lie l'inventaire des états terminaux et les empreintes de payloads. Le checkpoint du témoin lie séparément la tête d'historique reconnue.

Les clés de confiance et le scope attendu sont fournis séparément au vérificateur au lieu d'être acceptés depuis le bundle lui-même.

Le vérificateur renvoie l'un de trois états techniques de vérification :

- **`VERIFIED`** — inventaire, octets et état courant du témoin se réconcilient, et aucune exécution requise n'est non résolue ;
- **`INCOMPLETE`** — la fraîcheur ou la réconciliation est absente, ou une opération admise reste ouverte/inconnue ;
- **`FAILED`** — échec de signature, scope, séquence, hachage, fraîcheur ou inventaire.

Ces statuts sont des états de vérification technique, et non des labels réglementaires.

3. La propriété revendiquée

La propriété centrale est :

Pour la frontière runtime configurée et instrumentée, toute admission acquittée par le témoin configuré doit rester représentée dans l'historique témoin et se réconcilier avec un état terminal avant que la population exportée puisse être classée `VERIFIED`.

Cette propriété ne signifie pas :

- que toute requête réelle a atteint la frontière ;
- que tout effet distant a été capturé ;
- que toute source était vraie ;
- qu'un champ de revue humaine correspond forcément à une revue humaine réelle ;
- que toutes les voies internes du produit sont instrumentées ;
- que le résultat est juridiquement correct ;
- que la décision est sémantiquement correcte ;
- que le système est certifié ;
- que la production est exhaustivement couverte.

L'intérêt de cette propriété est d'être falsifiable : si une admission reconnue par le témoin disparaît, reste ouverte, est substituée ou ne correspond plus à l'état retenu par le témoin, le vérificateur ne doit pas retourner silencieusement **VERIFIED**.

4. Modèle de menace

Le producteur peut contrôler et réécrire sa base locale et peut re-signer ses propres bundles.

Le témoin possède une clé et un état distincts. Le vérificateur reçoit les matériaux de confiance attendus en dehors du bundle généré par le producteur.

Le modèle traite explicitement :

- la réécriture de l'historique local ;
- la troncature ;
- l'omission d'éléments admis ;
- le rejeu d'un ancien reçu ;
- les checkpoints obsolètes ou futurs ;
- les mauvaises clés de témoin ;
- la re-signature côté producteur après réécriture locale ;
- la perte de l'acquittement final ;
- l'interruption de processus ;
- les écritures concurrentes ;
- les preuves obligatoires manquantes ;
- la substitution d'inventaire.

Restent hors périmètre ou seulement partiellement couverts :

- les requêtes n'ayant jamais atteint une frontière instrumentée ;
- le contournement volontaire des wrappers/adaptateurs ;
- les chemins futurs non instrumentés ;
- la compromission conjointe producteur+témoin sans référence externe retenue ;
- la vérité sémantique ;
- l'horodatage certifié ;
- la non-équivocation publique multi-auditeurs ;
- une rotation/révocation complète des clés ;
- la capture réseau exhaustive ;
- les effets distants exactly-once.

5. Cycle de vie du protocole

5.1 Admission

L'admission est écrite transactionnellement avant l'exécution. SQLite est configuré avec `synchronous=FULL` sur ce chemin.

En mode obligatoire, un acquittement du témoin est requis avant de poursuivre.

5.2 Exécution

Le travail modèle/provider/outil n'a lieu qu'après franchissement du gate de capture configuré.

Un système distant peut néanmoins produire un effet secondaire entre l'intention et le résultat. Le protocole ne revendique donc pas une sémantique exactly-once.

5.3 État terminal

Le résultat, l'erreur, l'annulation ou tout autre état terminal est enregistré.

L'absence d'état terminal reste visible comme une chaîne incomplète au lieu d'être interprétée comme un succès ou un néant.

5.4 Libération

Certaines sorties sont persistées puis acquittées avant d'être libérées sur les frontières déclarées.

Une panne du stockage ou du témoin en mode obligatoire bloque l'exécution ou la libération ultérieure de la chaîne concernée.

5.5 Export

L'exporteur construit un dossier de preuve depuis le journal et les payloads retenus, signe le pack et produit un rapport de vérification.

5.6 Vérification

Le vérificateur contrôle successivement :

4. le scope attendu et les clés de confiance fournies extérieurement ;
5. la signature du témoin et le challenge ;
6. la fraîcheur ;
7. l'alignement entre la tête du témoin et la transcription présentée ;
8. la continuité des séquences et des previous hashes ;
9. les octets des payloads face à leurs empreintes ;
10. la réconciliation admission → état terminal ;
11. l'inventaire du dossier et la signature du producteur.

Un dossier peut donc être bien formé tout en étant **INCOMPLETE** ou **FAILED** relativement à l'état courant du témoin.

6. Preuves expérimentales

L'étude composant utilise un workflow synthétique de recommandation et le véritable chemin de finalisation de sortie critique du dépôt. Les appels LLM externes sont volontairement évités pour maintenir les scénarios hostiles déterministes.

Deux lots enregistrés contiennent 10 puis 100 recommandations. Chaque lot exécute 11 scénarios prédéterminés.

Sur 22 observations enregistrées, les classifications observées correspondent aux classifications attendues définies par le runner.

La suite v2 ciblée contient 67 tests : 24 hérités du travail initial et 43 ajoutés pour la capture durable et le comportement du témoin.

Les cas testés comprennent notamment : redémarrage du témoin, écritures concurrentes, panne de stockage, mort brutale du processus, perte d'acquiescement, preuve obligatoire manquante, réécriture locale entièrement re-hashée, checkpoints obsolètes/futurs, mauvaises clés témoin, ancien reçu sous nouveau challenge et ancien bundle confronté à un état témoin plus récent.

6.1 Scénarios hostiles

Scénario	Résultat attendu	Propriété exercée
checkpoint live intact	VERIFIED	la population témoin courante se réconcilie
annulation réconciliée	VERIFIED	l'interruption est fermée explicitement
témoin absent	INCOMPLETE	la fraîcheur/continuité ne peut être établie
admission interrompue	INCOMPLETE	une admission existe sans état terminal
checkpoint obsolète	FAILED	un ancien reçu n'est pas silencieusement accepté
ancien reçu sous nouveau challenge	FAILED	un reçu lié à un challenge ne peut satisfaire un nouveau challenge
historique tronqué	FAILED	la tête témoin et l'historique présenté divergent
admission réécrite	FAILED	l'engagement événement ne correspond plus
refus omis	FAILED	la réconciliation de population échoue
signature export invalide	FAILED	l'authentification du bundle producteur échoue
ancien bundle vs témoin plus récent	FAILED	validité historique ≠ validité courante

Ces résultats ne démontrent que la détection des attaques spécifiées dans la frontière testée.

7. Du protocole à la frontière applicative : Conversation → Dossier

Un protocole vérifiable reste fragile si l'application peut fournir de mauvais octets au chemin protégé.

La PR #30 a donc traité la transition Conversation → Dossier comme une partie de la frontière de preuve, et non comme une simple question d'interface.

L'audit initial avait identifié neuf constats. Le rapport de correction a ensuite consigné onze items DEF trouvés pendant la relecture hostile ; tous ont été corrigés dans le périmètre concerné.

Les changements comprennent notamment :

- staging hors du contexte actif jusqu'à acceptation de l'admission ;
- manifeste ordonné de fichiers acceptés avec empreintes SHA-256 ;
- nouvelle vérification des octets avant publication, afin qu'une extension ne puisse pas modifier un fichier après son hachage ;
- admission sous verrou avec traitement déterministe des états busy/drain/concurrence ;
- idempotence liée au JSON serveur et au manifeste ordonné, et non au seul nom/taille de fichier ;
- états explicites `pending`, `claimed` et `accepted` pour le lancement ;
- refus certains et réessayables, tandis qu'une exécution incertaine n'est pas rejouée automatiquement ;
- purge des uploads actifs lors de la suppression et protection contre la recréation d'un ancien contexte ;
- réconciliation des acquittements tardifs avec le brouillon d'origine sans écraser les modifications plus récentes de l'utilisateur ;
- rendu sûr des historiques, avec HTML échappé et filtrage des liens/images exécutables ;
- conservation du texte et des téléchargements sans inventer de statut d'approbation ou de validation.

Ce travail est important parce qu'un journal cryptographiquement solide ne suffit pas si l'application peut changer silencieusement l'ensemble des fichiers avant admission ou inférer un verdict historique plus fort que les preuves disponibles.

Le rapport de la PR #30 conserve explicitement les limites : aucun parcours de production authentifié, backend et interface à déployer ensemble, anciens reçus sans empreinte à examiner explicitement, et suppression des uploads actifs ne valant pas purge automatique des sauvegardes externes ou mémoires indépendantes.

8. Registre CI et validation

8.1 Gel protocole / publication

La PR #29 a été fusionnée dans `main` au commit :

```
f7cfe7ede525d5ba1b01f8f44ea8b2b54fe759eb
```

Avant fusion, son dernier head de release candidate :

```
763be256836afd04156bc9308b79dd1a93c42c83
```

a terminé avec succès les quatre workflows principaux :

- Evidence outputs and execution : **SUCCESS** ;
- Runtime capture : **SUCCESS** ;
- Main Gate : **SUCCESS** ;
- workflow général tests : **SUCCESS**.

Le nettoyage final a supprimé du chemin normal une dépendance transitive PyMuPDF/AGPL en empaquetant le shim ReportLab existant de KOREV comme distribution locale `markdown-pdf==1.5`, avec des gates fail-closed empêchant la réapparition de `fitz` / `pymupdf`.

La suite de parité PyMuPDF reste conservée comme couverture historique optionnelle de migration, au lieu de faire semblant qu'une dépendance volontairement retirée doit rester présente dans l'environnement de production.

8.2 Durcissement applicatif post-gel

La PR #30 a été fusionnée dans `main` au commit :

```
13524a13bfaald312c84fa7dcc588ede742b6388
```

Sur son head final :

```
7ac2d9f622eabe05b0ced9dbe1b651a10b40207c
```

les contrôles bloquants rapportés étaient verts, notamment Core Tests/sécurité, gates bloquants, Conversation & Dossiers, rendu/publication, WebUI et capture runtime.

La PR rapporte :

- 364 tests cœur dans le job Core bloquant ;
- 476 tests de sécurité, avec 8 skips attendus ;
- 5 973 tests unitaires hermétiques réussis, 46 skips ;
- 61 tests étendus E2E/intégration/infra/property réussis ;
- 121 tests UI ciblés réussis en validation locale ;
- 17 régressions d'admission ciblées réussies ;
- 2 parcours Chromium desktop/mobile ;
- couverture de la vue/sortie à 97,93 % lignes, 92,12 % branches et 95,12 % fonctions.

La même PR documente aussi que le job optionnel Extended Tests (Warning) comporte encore 22 échecs parmi 5 322 réussites, principalement liés à OCR et à d'anciennes assertions de rapport, et que le full-report rencontre un timeout asyncio également observé sur `main`.

Ces résultats non bloquants ne sont ni cachés ni présentés comme une suite produit globalement parfaite.

9. Le contre-exemple qui définit la frontière

La suite conserve volontairement une source sémantiquement fausse.

Si les octets admis à l'origine contiennent une assertion fausse et que ces octets restent exactement ceux engagés, la couche de preuve peut tout de même classer le dossier `VERIFIED`.

C'est volontaire.

L'intégrité répond à la question « la preuve a-t-elle changé ? ». Elle ne répond pas à « la preuve était-elle vraie ? ».

De même, la présence d'un artefact `human_review`, `sources` ou `consensus` prouve qu'un rôle documentaire requis contient des octets liés à l'intégrité.

Elle ne prouve pas que la source est autoritative, que le consensus est raisonnable, ou qu'un humain a réellement réalisé la revue.

La classification ne constitue donc pas un contrat de suffisance métier.

10. Frontière runtime et frontière applicative

L'implémentation actuelle instrumente des frontières déclarées incluant :

- admissions d'agents de haut niveau et exécution de tâches ;
- tentatives modèles, streams et embeddings ;
- certains providers directs ;
- arguments, résultats et erreurs d'outils ;
- sorties visibles côté UI ;
- certains octets de requêtes/réponses HTTP ;
- ensembles de fichiers acceptés lors du passage Conversation → Dossier.

Restent notamment hors périmètre :

- les requêtes n'ayant jamais atteint une frontière instrumentée ;
- le comportement brut interne shell/SSH/browser au-delà des frontières outils capturées ;
- l'état distant non renvoyé à travers une frontière capturée ;
- les futurs adaptateurs contournant la capture ;
- la réception effective d'octets HTTP proposés au client ;
- le raisonnement sémantique interne d'un modèle ;
- les machines externes non instrumentées.

La formulation correcte reste donc **capture sur des frontières déclarées et instrumentées**, pas « capture de tout ce que fait le système ».

11. Relation avec les trust planes centrés gateway

Traefik Labs décrit publiquement un Sovereign Trust Plane centré gateway couvrant le trafic modèles, MCP et API, avec engagement d'empreintes de logs gateway/access dans un journal Merkle append-only, checkpoints signés et témoins.

KOREV n'a ni benchmarké ni testé hostilement l'implémentation Traefik et ne revendique aucune supériorité.

Les périmètres techniques diffèrent suffisamment pour qu'il soit utile de les distinguer :

Dimension	Travail KOREV décrit ici	Description publique Traefik
Point d'observation principal	runtime agentique et frontières applicatives déclarées	gateway sur trafic modèle/MCP/API
Primitive de continuité	chaîne de hachage ordonnée + checkpoints reconnus par témoin	journal Merkle append-only + checkpoints/témoin
Propriété d'audit centrale	réconciliation de population après admission reconnue	intégrité/inclusion des enregistrements gateway
État applicatif	admission/état terminal + passage Conversation→Dossier	décision/policy/identité au niveau gateway
Payload de preuve	octets retenus + empreintes + pack signé	logs d'origine gardés par le client ; empreintes engagées
Modèle de statut	VERIFIED / INCOMPLETE / FAILED	vérification checkpoint/witness

Ces approches ne sont pas présentées comme équivalentes.

La preuve gateway concerne les frontières de communication visibles du gateway. Le travail KOREV présenté ici se concentre sur l'admission vérifiable, la réconciliation de population à l'intérieur d'un runtime déclaré et la frontière applicative qui détermine quels octets entrent réellement dans le dossier protégé.

12. Pourquoi un témoin retenu séparément change la propriété

Un producteur qui contrôle à la fois un log et sa clé de signature peut reconstruire un historique plus court mais cohérent, puis le signer.

Un témoin à clé distincte change la question : on ne demande plus seulement « cet historique est-il cohérent en lui-même ? », mais « cet historique prolonge-t-il bien l'état déjà retenu ailleurs ? »

Cela ne supprime pas toute confiance.

Si producteur et témoin sont compromis ensemble avant qu'une référence externe soit conservée, le protocole ne peut pas reconstruire une vérité perdue.

C'est pourquoi la feuille de route comprend notamment rétention externe des checkpoints, administration séparée et mécanismes de non-équivocation multi-auditeurs.

13. Vérifier sans publier tout le dépôt privé

Le dépôt de développement reste privé.

Le modèle de publication est conçu pour que la propriété puisse néanmoins être contestée sans exposer le code propriétaire non pertinent.

Le pack public de reproduction prévu doit contenir :

- le manuscrit et le registre des claims ;
- le protocole et le modèle de menace ;
- un vérificateur minimal suffisant pour les fixtures publiées ;
- les fixtures synthétiques et verdicts attendus ;
- les clés publiques de vérification ;
- les observations brutes et leurs empreintes ;
- les versions exactes de dépendances ;
- les commandes de reproduction ;
- les références immuables du source privé ;
- un manifeste SHA-256 du pack public.

Il doit exclure les clés privées, secrets, données clients/employés, bases de production, prompts propriétaires non nécessaires et détails d'infrastructure interne.

Tant qu'un tiers externe n'a pas réellement exécuté les matériaux, la formulation correcte reste « matériaux de reproduction fournis », et non « reproduit indépendamment ».

14. Contexte réglementaire et d'ingénierie

Le règlement européen sur l'IA établit des obligations de journalisation automatique et de traçabilité pour les systèmes à haut risque ainsi que, dans certaines situations, des obligations de conservation des logs.

Ces exigences rendent les historiques fiables opérationnellement importants, mais elles ne prescrivent pas ce protocole et ne font pas d'un résultat KOREV réussi une preuve de conformité réglementaire.

Sigstore/Rekor et SLSA fournissent des références d'ingénierie établies pour la provenance vérifiable, la vérification et la séparation des racines de confiance dans la supply chain logicielle.

L'annonce du Sovereign Trust Plane de Traefik en septembre 2026 illustre que la preuve autour des agents IA est également traitée au niveau infrastructure, et non plus seulement comme une question de logs applicatifs.

15. Ce que ce travail démontre

Sous réserve de la frontière déclarée et du modèle de menace, l'implémentation et les tests enregistrés soutiennent les affirmations suivantes :

1. les admissions reconnues par le témoin doivent se réconcilier avant qu'une population soit classée VERIFIED ;
2. les scénarios testés de modification, troncature, rejeu et omission sont détectés ;
3. les admissions interrompues restent représentables plutôt que de disparaître ;
4. une validité historique n'est pas silencieusement assimilée à une fraîcheur courante ;
5. les matériaux de confiance peuvent être fournis séparément du bundle producteur ;

6. certaines sorties peuvent être capturées avant libération sur des frontières runtime déclarées ;
 7. la transition Conversation→Dossier lie désormais plus étroitement les octets de fichiers acceptés et l'état d'admission au dossier protégé ;
 8. un examen externe peut être soutenu sans rendre public tout le dépôt privé, à condition de publier les matériaux de reproduction convenus.
-

16. Ce que ce travail ne démontre pas

Il ne démontre **pas** :

- une certification ou une conformité juridique ;
 - la vérité factuelle ou la justesse d'un modèle ;
 - l'adéquation métier d'une décision ;
 - l'efficacité d'une revue humaine ;
 - une complétude globale hors frontières déclarées ;
 - une indépendance opérationnelle du témoin dans l'expérience rapportée ;
 - une protection contre la compromission conjointe producteur+témoin sans référence externe conservée ;
 - des effets distants exactly-once ;
 - une qualification de latence/débit à l'échelle production ;
 - un déploiement de production authentifié des corrections Conversation→Dossier de la PR #30 ;
 - une reproduction tierce indépendante ;
 - une supériorité sur Traefik, Sigstore, SLSA ou un autre produit.
-

17. Travaux ouverts

Les étapes suivantes les plus fortes sont :

1. déployer le témoin sous administration et garde de clés séparées ;
 2. ajouter une rétention externe des checkpoints et de meilleures garanties de non-équivocation ;
 3. publier le pack de reproduction expurgé et obtenir une reproduction externe ;
 4. qualifier latence, débit, croissance stockage et comportements de panne en mode required-capture ;
 5. exercer des pipelines proches de la production avec cartographie explicite des frontières de capture ;
 6. formaliser la rotation et la révocation des clés ;
 7. qualifier chiffrement, sauvegarde, rétention et suppression des magasins de preuve ;
 8. exécuter des tests staging/production authentifiés pour Conversation→Dossier avec backend et UI déployés ensemble.
-

18. Conclusion

La distinction utile n'est pas « avoir des logs ou ne pas en avoir ».

Elle oppose **un historique affirmé par le système examiné à des éléments de preuve dont les propriétés annoncées peuvent être contestées par un examinateur**.

Le travail KOREV ajoute une propriété précise à cette discussion : la continuité de population après admission.

Si une admission a franchi une frontière déclarée et instrumentée puis a été reconnue par le témoin configuré, le système doit continuer à en rendre compte. Un historique ultérieur ne peut pas être qualifié de complet simplement parce que les enregistrements qu'il contient encore sont individuellement bien signés.

Cette revendication est volontairement plus étroite que la vérité, l'exhaustivité globale ou la conformité. Elle est en revanche plus testable.

Le durcissement applicatif réalisé après le gel du protocole prolonge exactement la même logique au moment où les fichiers et l'état réels d'un utilisateur passent de Conversation à Dossier : **la qualité de la preuve commence avant la cryptographie, au moment où le système décide ce qui a réellement été admis.**

Pour des systèmes d'IA participant à des décisions à conséquences, c'est à ce niveau que la responsabilité doit commencer.

Références

1. Règlement (UE) 2024/1689, articles 12 et 19, EUR-Lex : <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>
2. Sigstore, « Security Model — Rekor Security Model » : <https://docs.sigstore.dev/about/security/>
3. Sigstore, « Rekor » : <https://docs.sigstore.dev/logging/overview/>
4. SLSA v1.2, « Requirements » : <https://slsa.dev/spec/v1.2/requirements>
5. SLSA v1.2, « Build: Verifying artifacts » : <https://slsa.dev/spec/v1.2/verifying-artifacts>
6. Traefik Labs, « Traefik Labs Introduces the Sovereign Trust Plane, Bringing Verifiable Evidence to AI Agent Governance », 15 septembre 2026 : <https://traefik.io/press/traefik-labs-introduces-the-sovereign-trust-plane-bringing-verifiable-evidence-to-ai-agent-governance>
7. Emile Vauge, Traefik Labs, « Agentic Telemetry Is Not an Audit Trail: Why We Built the Sovereign Trust Plane », 14 septembre 2026 : <https://traefik.io/blog/why-we-built-the-sovereign-trust-plane>

Contrôle de publication

Gel protocole/publication : f7cfe7ede525d5ba1b01f8f44ea8b2b54fe759eb. Référence d'implémentation durcie après PR #30 : 13524a13bfaa1d312c84fa7dcc588ede742b6388.

Toute communication publique doit rester alignée avec CLAIMS_LEDGER.md. Un nouveau commit d'implémentation ne doit pas remplacer silencieusement ces références sans réexécuter les contrôles de publication annoncés.