INSTITUT DE L'ÉTAT-CONTRAT

GUIDE TECHNIQUE — INFRASTRUCTURE BLOCKCHAIN

Ethereum + Linea

Version 6 — Architecture complète 5 modules (révision technique)

Frédérick Gérald
contact@institutetatcontrat.fr · www.institutetatcontrat.fr · Octobre 2026

Introduction — Vue d'ensemble de l'architecture

Ce guide décrit l'architecture blockchain de l'État-Contrat, telle que définie dans le livre et le Manifeste de Frédérick Gérald. Il couvre les cinq modules dans leur ordre logique : du plus central (le budget) au plus systémique (le vote citoyen).

PRINCIPE FONDATEUR : « La confiance ne peut plus reposer uniquement sur la morale des individus. Elle doit s'appuyer sur des mécanismes techniques rendant les décisions traçables et vérifiables. » — Le Manifeste de l'État-Contrat

L'architecture repose sur deux réseaux complémentaires :

- Ethereum L1 — ancrage permanent des preuves critiques : registre public, pratiquement impossible à falsifier et très résistant à la censure
- Linea L2 (réseau de seconde couche d'Ethereum, de type zk-rollup) — exécution quotidienne des flux, à un coût par transaction très inférieur à celui de L1 ; chaque lot de transactions est validé sur Ethereum L1 par une preuve de validité (ZK-proof)

À SAVOIR : la blockchain garantit que ce qui est enregistré ne peut plus être modifié en silence. Elle ne garantit pas que ce qui est saisi est exact. La qualité des données d'entrée (le « problème de l'oracle ») reste une responsabilité institutionnelle, traitée module par module.

Les 5 modules dans leur ordre logique

Module	Nom	Poids	Objet	Priorité
A	Le Trésor Programmable	45%	Chaque euro public traçable de l'engagement à la réalisation	1 — Capital
B	Le Permis de Gouverner	15%	Certification psychologique et cognitive de ceux qui gouvernent	2 — Fondamental
C	Les Contrats de Mandat	15%	Engagements chiffrés, jalons, sanctions automatiques	3 — Structurant
D	Les Registres d'Intégrité	20%	Anti-corruption, lobbying, patrimoine, conflits d'intérêts	4 — Systémique
E	Le Vote Citoyen	5%	Vote anonyme vérifiable via ZK-SNARKs sur les sujets structurants	5 — Complémentaire

LOGIQUE NARRATIVE : On commence par ce qui est central (le budget), puis on vérifie qui le gère (le Permis), puis on encadre ses promesses (le Mandat), puis on surveille ses réseaux (l'Intégrité), et enfin on fait valider les grands choix par les citoyens (le Vote).

MODULE A — Le Trésor Programmable (45%)

Le budget est le seul endroit où le réel résiste. On peut promettre n'importe quoi, voter n'importe quelle loi — mais on ne peut pas dépenser indéfiniment ce qu'on ne produit pas. C'est pourquoi le Trésor Programmable est le cœur de toute l'architecture.

Principe

Chaque euro public devient traçable de la loi de finances jusqu'au bénéficiaire final. L'attribution, le montant, la règle qui l'autorise, la preuve d'exécution — tout est enregistré sur la blockchain de manière immuable et publiquement vérifiable. Les données personnelles des bénéficiaires restent hors chaîne : seule leur empreinte cryptographique est publiée.

Flux technique

```
LOI DE FINANCES
       ↓
SMART CONTRACT DE BUDGET
  (crédits ouverts par mission, règles d'attribution encodées)
       ↓
PAIEMENT CONDITIONNEL
  (refusé automatiquement si les critères ou les crédits ne sont pas réunis)
       ↓
ENREGISTREMENT SUR LINEA L2
  (empreinte, montant, bénéficiaire pseudonymisé, horodatage)
       ↓
ANCRAGE SUR ETHEREUM L1
  (preuve permanente, non modifiable)
       ↓
VÉRIFICATION PUBLIQUE
  (tout citoyen peut contrôler via Lineascan / Etherscan)
```

Données enregistrées sur la blockchain

- Empreinte (hash) du document budgétaire source (loi de finances, décret)
- Montant alloué par mission (Santé, Éducation, Défense, etc.)
- Critères d'attribution encodés dans le contrat
- Engagement cryptographique du bénéficiaire (empreinte « salée ») — son identité reste dans le système comptable de l'État
- Horodatage de chaque étape du flux
- Empreinte du justificatif d'exécution (document stocké sur IPFS)

Mesures concrètes

Périmètre	Objectif	Délai d'audit
80–95% des paiements publics	Comptes budgétaires programmables	0–7 jours (vs 12–24 mois aujourd'hui)
100% des marchés publics	Critères, scoring, attribution, paiement traçables	Temps réel
100% des subventions	De l'émission au bénéficiaire final	Temps réel
Exceptions strictes	Défense, urgences (cadre légal renforcé)	Audit différé sécurisé

Pourquoi la blockchain est indispensable ici

- Registre append-only : aucune retouche comptable discrète possible
- Preuve opposable : empreinte, transaction et règle du contrat vérifiables publiquement
- L'entité contrôlée ne maîtrise plus les données sur lesquelles porte le contrôle
- Passage du contrôle a posteriori (après coup) au contrôle ex ante (en temps réel)

CE QUE CE MODULE CHANGE : La fraude prospère sur 4 facteurs — opacité des flux, complexité procédurale, possibilité de réécriture, absence de sanction automatique. Le Trésor Programmable s'attaque aux 4 simultanément.

Exemple de Smart Contract — Trésor Programmable

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {AccessControl} from "@openzeppelin/contracts/access/AccessControl.sol";

/// @notice Exemple pédagogique — non audité, ne pas déployer en l'état.
contract TresorEtatContrat is AccessControl {
    bytes32 public constant LOI_DE_FINANCES = keccak256("LOI_DE_FINANCES"); // ouvre les crédits
    bytes32 public constant COMPTABLE = keccak256("COMPTABLE");             // exécute les paiements

    struct Paiement {
        bytes32 missionId;       // ex : keccak256("SANTE")
        uint256 montant;         // en centimes d'euro
        bytes32 beneficiaire;    // engagement salé : aucune donnée personnelle en clair
        bytes32 justificatif;    // empreinte du justificatif stocké sur IPFS
        uint64  horodatage;
        address validateur;
    }

    mapping(bytes32 => uint256) public creditsRestants; // plafond par mission
    Paiement[] private _paiements;                      // append-only : aucune fonction de modification

    event CreditsOuverts(bytes32 indexed missionId, uint256 montant, bytes32 texteSource);
    event PaiementEnregistre(uint256 indexed id, bytes32 indexed missionId, uint256 montant,
                             bytes32 justificatif, address indexed validateur);

    constructor(address admin) {
        _grantRole(DEFAULT_ADMIN_ROLE, admin);
    }

    /// Ouverture des crédits votés (texteSource = empreinte de la loi de finances ou du décret)
    function ouvrirCredits(bytes32 missionId, uint256 montant, bytes32 texteSource)
        external onlyRole(LOI_DE_FINANCES)
    {
        creditsRestants[missionId] += montant;
        emit CreditsOuverts(missionId, montant, texteSource);
    }

    /// Paiement conditionnel : refusé si la mission n'a plus de crédits
    function enregistrerPaiement(bytes32 missionId, uint256 montant, bytes32 beneficiaire, bytes32 justificatif)
        external onlyRole(COMPTABLE) returns (uint256 id)
    {
        require(montant > 0, "Montant nul");
        require(creditsRestants[missionId] >= montant, "Credits insuffisants");
        creditsRestants[missionId] -= montant;

        id = _paiements.length;
        _paiements.push(Paiement(missionId, montant, beneficiaire, justificatif,
                                 uint64(block.timestamp), msg.sender));
        emit PaiementEnregistre(id, missionId, montant, justificatif, msg.sender);
    }

    // Lecture publique — tout citoyen peut appeler sans restriction
    function paiement(uint256 id) external view returns (Paiement memory) {
        return _paiements[id];
    }

    function nombrePaiements() external view returns (uint256) {
        return _paiements.length;
    }
}
```

Ce que l'exemple montre : seuls les comptables habilités peuvent enregistrer un paiement ; un paiement est refusé si la mission n'a plus de crédits ; aucune fonction ne permet de modifier ou de supprimer un paiement. Les textes (nom de mission, document) sont remplacés par leurs empreintes, ce qui réduit fortement les frais de transaction.

MODULE B — Le Permis de Gouverner (15%)

C'est la contribution la plus originale et la plus disruptive du modèle. Il n'existe nulle part dans les démocraties occidentales actuelles. Principe : on exige un permis pour conduire une voiture, une certification pour exercer la médecine — mais aucune vérification n'est exigée pour gérer le budget des retraites ou appuyer sur le bouton nucléaire.

« La stabilité d'un système repose sur l'équilibre de ceux qui le dirigent. »

Principe fondateur

Gérer le destin de millions de personnes est une fonction critique. Elle nécessite une intégrité mentale vérifiée. Le Permis de Gouverner est une condition préalable à toute prise de fonction à haute responsabilité — non pas un jugement idéologique, mais une condition de fonctionnement.

Les 3 composantes du Permis de Gouverner

Composante 1 — Tests Psychopathologiques

Objectif : écarter les profils présentant des traits incompatibles avec l'exercice responsable du pouvoir.

- Détection des profils narcissiques pathologiques — recherche du pouvoir pour le pouvoir
- Détection des profils paranoïaques — suspicion systématique, décisions imprévisibles
- Détection des profils prédateurs — exploitation d'autrui, absence d'empathie
- Évaluation de la tolérance au stress et à la pression décisionnelle
- Capacité à intégrer des avis contradictoires sans déstabilisation

Ces tests sont réalisés par des psychologues cliniciens indépendants, sous protocole certifié, sans que l'évaluateur connaisse l'identité politique du candidat.

Composante 2 — Intégrité Cognitive

Objectif : s'assurer que le jugement n'est pas altéré par des dépendances ou des états cliniques.

- Tests toxicologiques — absence de dépendances altérant le jugement
- Bilan neuropsychologique de base — capacités décisionnelles, mémoire de travail
- Absence de pathologies non traitées affectant le raisonnement
- Renouvellement périodique obligatoire en cours de mandat

Composante 3 — Certification et Renouvellement

- Obligatoire avant toute prise de fonction à haute responsabilité
- Gradué selon le niveau : élu local → parlementaire → ministre → chef d'État
- Renouvelé à mi-mandat et en cas de suspicion clinique signalée
- Seul le résultat (apte / non apte) est archivé sur la blockchain — jamais le détail médical

Le défi technique : confidentialité médicale + preuve publique

C'est ici qu'interviennent les Zero-Knowledge Proofs (ZKP). Ce mécanisme cryptographique permet de prouver publiquement qu'une attestation d'aptitude a été délivrée par un évaluateur accrédité et que le seuil défini par le protocole est atteint, SANS jamais révéler les données médicales brutes.

Précision importante : la preuve garantit l'authenticité de l'attestation, pas la validité scientifique du test. Celle-ci relève du protocole d'évaluation et de son contrôle indépendant.

```
PROCESSUS TECHNIQUE DU PERMIS DE GOUVERNER

ÉTAPE 1 — TEST réalisé par psychologue indépendant accrédité
          (anonymat du candidat garanti côté évaluateur)
        ↓
ÉTAPE 2 — RÉSULTAT BRUT stocké chiffré hors-chaîne
          (secret médical — jamais sur la blockchain)
        ↓
ÉTAPE 3 — ZK-PROOF généré
          Preuve cryptographique que :
          • une attestation signée par un évaluateur accrédité existe
          • le seuil d'aptitude est atteint
          SANS révéler le contenu du résultat
        ↓
ÉTAPE 4 — PREUVE vérifiée et STATUT enregistré sur LINEA L2
          Horodatage immuable, rattaché à l'identifiant public de l'élu
        ↓
ÉTAPE 5 — ANCRAGE sur ETHEREUM L1
          Preuve permanente, non modifiable
        ↓
ÉTAPE 6 — VÉRIFICATION PUBLIQUE
          Tout citoyen peut contrôler :
          « Ce candidat possède un Permis valide »
          SANS accéder aux données médicales
```

IDENTITÉ : un élu est une personne publique. Le statut de son Permis est donc rattaché à son identité publique — sinon aucun citoyen ne pourrait vérifier qu'un candidat donné en possède un. Ce qui reste secret, ce sont les données médicales et le détail des évaluations.

Condition préalable au Module C — Contrat de Mandat

Dans l'architecture de l'État-Contrat, le Permis de Gouverner est une condition sine qua non. Un élu ne peut pas signer de Contrat de Mandat (Module C) tant que son Permis n'est pas valide et enregistré sur la chaîne.

```
CANDIDAT À UNE FONCTION DE HAUTE RESPONSABILITÉ
                  ↓
     PASSAGE DES TESTS (Module B)
     • Psychopathologie
     • Intégrité cognitive
     • Toxicologie
                  ↓
     ┌──────────────────────────────┐
     │ RÉSULTAT ?                   │
     │ APTE  → ZK-Proof généré      │
     │         Enregistré sur Linea L2 │
     │         Ancré sur Ethereum L1   │
     │         PERMIS VALIDE ✓      │
     │ INAPTE → Inéligibilité selon │
     │          protocole défini    │
     │          (recours possible)  │
     └──────────────────────────────┘
                  ↓
     SIGNATURE DU CONTRAT DE MANDAT (Module C)
     (impossible sans Permis valide)
```

Exemple de Smart Contract — Permis de Gouverner

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {AccessControl} from "@openzeppelin/contracts/access/AccessControl.sol";

/// @notice Exemple pédagogique — non audité, ne pas déployer en l'état.
contract PermisDeGouvernerRegistry is AccessControl {
    bytes32 public constant CERTIFICATEUR = keccak256("CERTIFICATEUR"); // organisme de certification
    bytes32 public constant JURIDICTION = keccak256("JURIDICTION");     // tranche les contestations
```

```
    enum Niveau { LOCAL, PARLEMENTAIRE, MINISTRE, CHEF_ETAT }
    enum Statut { INEXISTANT, VALIDE, SUSPENDU, REVOQUE }

    struct Permis {
        Niveau  niveau;
        Statut  statut;
        bytes32 attestation;     // empreinte de l'attestation signée (aucune donnée médicale)
        uint64  dateEmission;
        uint64  dateExpiration;  // renouvellement obligatoire
        address certificateur;
    }

    // Clé = identifiant public de l'élu (un élu est une personne publique :
    // c'est le résultat qui est public, jamais les données médicales)
    mapping(bytes32 => Permis) public permis;

    event PermisDelivre(bytes32 indexed elu, Niveau niveau, uint64 dateExpiration, bytes32 attestation);
    event PermisSuspendu(bytes32 indexed elu, bytes32 motif);
    event PermisRetabli(bytes32 indexed elu, bytes32 decision);
    event PermisRevoque(bytes32 indexed elu, bytes32 decision);

    constructor(address admin) {
        _grantRole(DEFAULT_ADMIN_ROLE, admin);
    }

    function delivrer(bytes32 elu, Niveau niveau, uint64 dureeValidite, bytes32 attestation)
        external onlyRole(CERTIFICATEUR)
    {
        require(dureeValidite > 0, "Duree nulle");
        Statut actuel = permis[elu].statut;
        require(actuel != Statut.SUSPENDU && actuel != Statut.REVOQUE, "Permis suspendu ou revoque");
        uint64 expiration = uint64(block.timestamp) + dureeValidite;
        permis[elu] = Permis(niveau, Statut.VALIDE, attestation, uint64(block.timestamp), expiration, msg.sender);
        emit PermisDelivre(elu, niveau, expiration, attestation);
    }

    /// Suspension conservatoire (ex. suspicion clinique signalée)
    function suspendre(bytes32 elu, bytes32 motif) external onlyRole(CERTIFICATEUR) {
        require(permis[elu].statut == Statut.VALIDE, "Permis non valide");
        permis[elu].statut = Statut.SUSPENDU;
        emit PermisSuspendu(elu, motif);
    }

    /// Décision après procédure contradictoire (decision = empreinte de la décision publiée)
    function retablir(bytes32 elu, bytes32 decision) external onlyRole(JURIDICTION) {
        require(permis[elu].statut == Statut.SUSPENDU, "Permis non suspendu");
        permis[elu].statut = Statut.VALIDE;
        emit PermisRetabli(elu, decision);
    }

    function revoquer(bytes32 elu, bytes32 decision) external onlyRole(JURIDICTION) {
        require(permis[elu].statut != Statut.INEXISTANT, "Permis inexistant");
        permis[elu].statut = Statut.REVOQUE;
        emit PermisRevoque(elu, decision);
    }

    // Vérification publique — tout citoyen peut appeler
    function estApte(bytes32 elu, Niveau niveau) public view returns (bool) {
        Permis memory p = permis[elu];
        return p.statut == Statut.VALIDE
            && uint8(p.niveau) >= uint8(niveau)
            && block.timestamp < p.dateExpiration;
    }

    // Appelé par le Module C avant toute signature de mandat
    function exigerPermisValide(bytes32 elu, Niveau niveau) external view {
        require(estApte(elu, niveau), "Permis de Gouverner invalide ou expire");
    }
}
```

Ce que l'exemple montre : seul un organisme certificateur habilité délivre ou suspend un Permis ; un Permis suspendu ne peut pas être redélivré en contournant la suspension ; le rétablissement ou la révocation relèvent d'une juridiction, après procédure contradictoire. Aucune donnée médicale n'est stockée : seulement l'empreinte de l'attestation.

Gouvernance du Module B

Acteur	Rôle	Indépendance garantie par
Collège de psychologues accrédités	Réalise les évaluations	Rotation aléatoire, anonymat du candidat
Organisme de certification	Valide et génère le ZK-proof	Séparation stricte de l'État
Smart contract blockchain	Enregistre et vérifie la validité	Code open source, audits de sécurité publiés
Tout citoyen	Vérifie qu'un élu a un permis valide	Interface publique sans identification
Tribunal constitutionnel	Tranche les contestations	Procédure définie dans le socle contractuel

POINT CLÉ : Le Permis de Gouverner ne définit pas quelle politique mener. Il garantit simplement que celui qui gouverne est mentalement apte à le faire de manière responsable. C'est une condition de fonctionnement, pas un jugement idéologique.

POINTS JURIDIQUES À VALIDER

- Les résultats d'évaluation sont des données de santé : leur traitement relève du régime le plus protecteur du RGPD (article 9) et du secret médical.
- Conditionner l'éligibilité à un test touche au principe d'égale admissibilité aux fonctions publiques : la mise en œuvre pourrait nécessiter une révision constitutionnelle.
- Tout refus, suspension ou révocation doit ouvrir un recours effectif devant une juridiction indépendante.

MODULE C — Les Contrats de Mandat (15%)

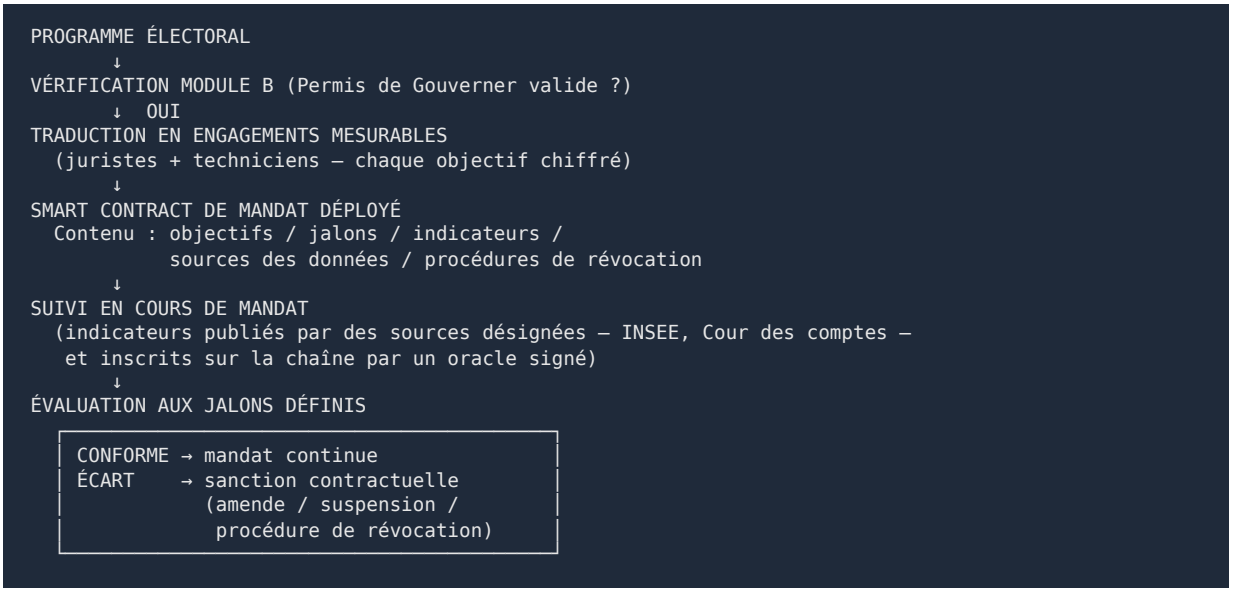
CONDITION PRÉALABLE : Le Permis de Gouverner (Module B) doit être valide et enregistré sur la chaîne AVANT toute signature d'un Contrat de Mandat. Le contrat ne peut pas être déployé sans cette vérification automatique.

Principe

Les engagements d'un élu cessent d'être des promesses. Ils deviennent des contrats exécutables — avec objectifs chiffrés, jalons mesurables, et sanctions automatiques en cas de non-respect.

L'élu ne reçoit plus un chèque en blanc : il signe un cahier des charges. Ce qui est décidé engage juridiquement et techniquement dans le temps.

Flux technique



Le point critique : l'oracle

Un smart contract ne lit pas le monde réel : il ne connaît que les données qu'on lui transmet. Les indicateurs (chômage, déficit, délais) doivent donc être inscrits sur la chaîne par une source désignée à l'avance dans le contrat, qui signe chaque donnée publiée. Le choix de cette source et sa méthode de calcul font partie intégrante du Contrat de Mandat.

Contenu du Contrat de Mandat

Élément	Description	Exemple concret
Objectifs chiffrés	Résultats mesurables attendus	Réduire le chômage de X% en 3 ans
Jalons temporels	Points de contrôle intermédiaires	Rapport à 6 mois, 1 an, 2 ans
Indicateurs	Méthode de calcul et source définies à l'avance	Source INSEE, méthode publiée
Caution de mandat	6 à 24 mois de rémunération selon niveau	Exposée contractuellement dès la prise de poste

Élément	Description	Exemple concret
Procédure de révocation	Conditions et seuils déclenchant la révocation	Écart > X% pendant Y mois consécutifs

Ce que ce module change

- **Le débat politique ne porte plus sur qui a raison — mais sur qui respecte ses engagements**
- L'alternance cesse d'être destructrice : la trajectoire contractuelle survit aux changements d'équipe
- La révocation cesse d'être exceptionnelle — elle devient un outil normal de régulation
- Le charisme et la rhétorique ne suffisent plus à couvrir l'échec

POINTS JURIDIQUES À VALIDER

- Une sanction peut être déclenchée automatiquement, mais elle doit rester contestable : le contrat ouvre la procédure, une juridiction indépendante la confirme (principe du contradictoire).
- La caution de mandat et les amendes prélevées sur la rémunération d'un élu nécessitent une base légale.
- Le droit français actuel interdit le mandat impératif pour les parlementaires (article 27 de la Constitution) : le Contrat de Mandat suppose donc une évolution du cadre constitutionnel.

MODULE D — Les Registres d'Intégrité (20%)

Ce module crée un registre public et immuable de tout ce qui, aujourd'hui, reste opaque : les conflits d'intérêts, le lobbying, le patrimoine des élus, leurs agendas décisionnels et les nominations sensibles.

Domaines couverts

Domaine	Obligation	Délai
Lobbying	100% des rendez-vous d'influence déclarés	Sous 48 heures
Conflits d'intérêts	Participations et intérêts déclarés	Avant prise de fonction + mises à jour
Patrimoine	Déclaration en début et fin de mandat	J+30 de la prise de poste
Agendas décisionnels	Rendez-vous liés à des décisions publiques	Dans les 7 jours
Nominations sensibles	Critères et processus de sélection	Avant annonce officielle

Pourquoi la blockchain est indispensable ici

La blockchain ne peut pas obliger quelqu'un à déclarer : l'omission se détecte par les contrôles et les recoupements. En revanche, elle rend impossible la modification ultérieure d'une déclaration. Horodatage + immutabilité = disparition de la manipulation silencieuse.

- Une déclaration déposée ne peut plus être rétroactivement modifiée
- Toute correction prend la forme d'une nouvelle déclaration, visible et datée
- Les informations confidentielles (montants précis, tiers protégés) peuvent être prouvées sans être publiées, via ZK-proofs

Sanctions automatiques

Manquement	Sanction contractuelle automatique
Non-déclaration d'un rendez-vous d'influence	Amende paramétrée + mention publique
Déclaration tardive (> délai contractuel)	Pénalité progressive selon retard
Conflit d'intérêts non déclaré	Suspension + procédure de révocation
Fausse déclaration prouvée	Inéligibilité et procédure pénale

POINT CLÉ : La sanction n'est pas politique, elle est contractuelle et automatique. Elle ne dépend pas d'une volonté politique de punir — elle s'applique mécaniquement dès que le manquement est constaté et enregistré sur la chaîne.

POINTS JURIDIQUES À VALIDER

- Le retard d'une déclaration se constate automatiquement ; une omission ou une fausse déclaration doit d'abord être établie par une autorité de contrôle, qui l'inscrit sur la chaîne.
- Les données personnelles (patrimoine, rendez-vous) restent hors chaîne, avec leur seule empreinte sur la blockchain : une donnée inscrite sur la chaîne ne peut plus être effacée, ce qui est incompatible avec le droit à l'effacement du RGPD.
- La Haute Autorité pour la transparence de la vie publique (HATVP) publie déjà certaines déclarations : le registre peut s'adosser à ce dispositif.

MODULE E — Le Vote Citoyen via ZK-SNARKs (5%)

Zone 2 — Utile

Dans l'État-Contrat, le vote n'est pas le cœur du système — le cœur c'est le budget, l'engagement et la preuve. Mais le vote citoyen sur les sujets structurants est l'outil de validation démocratique finale. Il doit être à la fois anonyme et auditable. Les ZK-SNARKs résolvent cette contradiction apparente.

Principe

Le citoyen prouve qu'il a le droit de voter — son éligibilité — sans révéler qui il est. Sa voix est comptabilisée sur Linea L2, et la preuve globale du scrutin est ancrée sur Ethereum L1. L'auditabilité est totale, et l'anonymat repose sur la cryptographie plutôt que sur la confiance.

C'est l'inverse exact du vote actuel : aujourd'hui l'anonymat est déclaratif (on vous fait confiance). Avec les ZK-SNARKs, l'anonymat est garanti par la cryptographie — aucun acteur, pas même l'État, ne peut relier un vote à un votant à partir des données publiées.

Pourquoi ZK-SNARKs et pas une simple blockchain ?

- Une blockchain classique enregistre les transactions publiquement — ce qui permettrait de tracer qui a voté quoi
- ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) prouve la validité d'un vote SANS révéler l'identité du votant
- Le résultat : un registre de votes vérifiable par tous, sans qu'aucun vote individuel ne soit rattachable à une personne

Flux technique complet

```
ÉTAPE 0 — INSCRIPTION (une seule fois)
  L'autorité électorale vérifie l'identité du citoyen et ajoute
  son « engagement d'identité » (empreinte de son secret) à la liste
  électorale cryptographique (arbre de Merkle).
  Elle sait qui est inscrit — jamais qui vote quoi.
                ↓
ÉTAPE 1 — LE CITOYEN SOUHAITE VOTER
                ↓
ÉTAPE 2 — GÉNÉRATION DU ZK-SNARK (côté client)
  Un circuit cryptographique génère une preuve :
  'Je figure sur la liste, je n'ai pas encore voté
   à ce scrutin, et je choisis l'option X'
  La preuve ne contient aucune donnée personnelle
                ↓
ÉTAPE 3 — VOTE + ZK-PROOF transmis via un relais sur LINEA L2
  • Le relais paie les frais : l'adresse du votant n'apparaît pas
  • Le ZK-proof garantit la validité du vote sans l'identifier
  • Un nullifier unique empêche le double vote
    (sans révéler l'identité du votant)
                ↓
ÉTAPE 4 — COMPTABILISATION SUR LINEA L2
  • Les votes valides sont agrégés
  • Chaque vote est vérifié mathématiquement
  • Le dépouillement est automatique et public
                ↓
ÉTAPE 5 — RÉSULTAT + PREUVE GLOBALE ancrés sur ETHEREUM L1
  • Résultat définitif, non modifiable
  • Audit complet possible par n'importe qui
  • Aucun individu ne peut être identifié
```

Option avancée (MACI) : le vote est en plus chiffré jusqu'au dépouillement. Cela empêche les résultats intermédiaires et permet de changer son vote en secret, ce qui rend l'achat de voix et la pression inutiles.

Les 3 garanties cryptographiques

Garantie	Mécanisme	Ce que cela signifie concrètement
Anonymat	ZK-SNARK + relais	Aucune donnée publiée ne permet de relier un vote à un votant
Unicité du vote	Nullifier	Chaque citoyen éligible ne peut voter qu'une seule fois par scrutin
Intégrité du résultat	Ancrage L1	Le résultat final est immuable et vérifiable par tout citoyen

Limites à connaître

- Coercition et achat de voix : un vote à distance peut être surveillé. MACI y répond en permettant de modifier son vote en secret.
- Sécurité du terminal : un téléphone ou un ordinateur compromis peut fausser le vote avant même la génération de la preuve.
- Cérémonie initiale : les ZK-SNARKs de type Groth16 nécessitent une « cérémonie de mise en place de confiance ». Semaphore s'appuie sur une cérémonie publique déjà réalisée.

Le Nullifier — empêcher le double vote sans identifier le votant

C'est l'un des mécanismes les plus élégants de l'architecture. Un nullifier est une empreinte unique dérivée du secret du votant et de l'identifiant du scrutin — mais qui ne révèle ni l'un ni l'autre.

```
COMMENT FONCTIONNE LE NULLIFIER

nullifier = Hash(secret_votant, id_scrutin)

→ Avant le vote : le contrat vérifie que ce nullifier
  n'a jamais été utilisé pour ce scrutin
→ Après le vote : le nullifier est enregistré
→ Une tentative de revoter produit le même nullifier
  → rejeté automatiquement
→ Pour un autre scrutin, le nullifier est différent :
  les votes d'une même personne ne sont pas reliables entre eux

Résultat : double vote impossible, identité invisible.
```

Usages dans l'État-Contrat

Le vote citoyen n'est pas consulté en permanence — il intervient sur des sujets structurants, après cadrage institutionnel (chapitre 10.7 du livre).

- Validation des grandes orientations de civilisation
- Choix des priorités économiques et sociales structurantes
- Arbitrages budgétaires majeurs rendus lisibles et chiffrés
- Politiques environnementales et territoriales importantes
- Révision du socle contractuel de l'État-Contrat

RÈGLE FONDAMENTALE : Le vote ne porte jamais sur des slogans ou des récits. Chaque sujet soumis au vote est d'abord qualifié par un organe constitutionnel indépendant, chiffré, assorti de ses conséquences mesurables, et présenté en options claires. Le citoyen arbitre entre des réalités, pas entre des promesses.

Exemple de Smart Contract — Vote ZK-SNARK

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {AccessControl} from "@openzeppelin/contracts/access/AccessControl.sol";

/// Vérificateur Groth16 généré à partir du circuit (ex. circuit Semaphore).
/// Signaux publics : [racine de la liste électorale, nullifier, option, scrutin]
interface IVerificateurVote {
    function verifierPreuve(uint256[2] calldata a, uint256[2][2] calldata b,
                            uint256[2] calldata c, uint256[4] calldata signaux)
        external view returns (bool);
}

/// @notice Exemple pédagogique — non audité, ne pas déployer en l'état.
contract VoteCitoyenZK is AccessControl {
    bytes32 public constant ORGANE_CONSTITUTIONNEL = keccak256("ORGANE_CONSTITUTIONNEL");

    struct Scrutin {
        bytes32   sujet;         // empreinte du dossier chiffré soumis au vote (IPFS)
        uint64    ouverture;
        uint64    fermeture;
        uint256   racineListe;   // racine de Merkle des électeurs inscrits
        uint256[] resultats;     // un compteur par option
    }

    IVerificateurVote public immutable verificateur;
    Scrutin[] private _scrutins;
    mapping(uint256 => mapping(uint256 => bool)) public nullifierUtilise; // scrutin => nullifier

    event ScrutinCree(uint256 indexed scrutinId, bytes32 sujet, uint8 nbOptions);
    event VoteEnregistre(uint256 indexed scrutinId, uint256 option, uint256 nullifier);

    constructor(address admin, IVerificateurVote _verificateur) {
        _grantRole(DEFAULT_ADMIN_ROLE, admin);
        verificateur = _verificateur;
    }
```

```
    function creerScrutin(bytes32 sujet, uint64 ouverture, uint64 fermeture,
                          uint256 racineListe, uint8 nbOptions)
        external onlyRole(ORGANE_CONSTITUTIONNEL) returns (uint256 id)
    {
        require(nbOptions >= 2, "Au moins 2 options");
        require(fermeture > ouverture, "Dates invalides");
        id = _scrutins.length;
        Scrutin storage s = _scrutins.push();
        s.sujet = sujet;
        s.ouverture = ouverture;
        s.fermeture = fermeture;
        s.racineListe = racineListe;
        s.resultats = new uint256[](nbOptions);
        emit ScrutinCree(id, sujet, nbOptions);
    }

    /// Transaction envoyée par un relais : l'adresse qui paie les frais n'est pas celle du votant
    function voter(uint256 scrutinId, uint256 option, uint256 nullifier,
                   uint256[2] calldata a, uint256[2][2] calldata b, uint256[2] calldata c)
        external
    {
        Scrutin storage s = _scrutins[scrutinId];
        require(block.timestamp >= s.ouverture, "Pas encore ouvert");
        require(block.timestamp <= s.fermeture, "Scrutin clos");
        require(option < s.resultats.length, "Option inconnue");
        require(!nullifierUtilise[scrutinId][nullifier], "Deja vote");

        // La preuve établit, sans révéler l'identité, que le votant figure sur la liste,
        // que le nullifier dérive de son secret et de ce scrutin, et qu'il a choisi cette option.
        uint256[4] memory signaux = [s.racineListe, nullifier, option, scrutinId];
        require(verificateur.verifierPreuve(a, b, c, signaux), "Preuve ZK invalide");

        nullifierUtilise[scrutinId][nullifier] = true;
        s.resultats[option] += 1;
        emit VoteEnregistre(scrutinId, option, nullifier);
    }

    // Résultat public — consultable par tout citoyen
    function resultats(uint256 scrutinId) external view returns (uint256[] memory) {
        return _scrutins[scrutinId].resultats;
    }
}
```

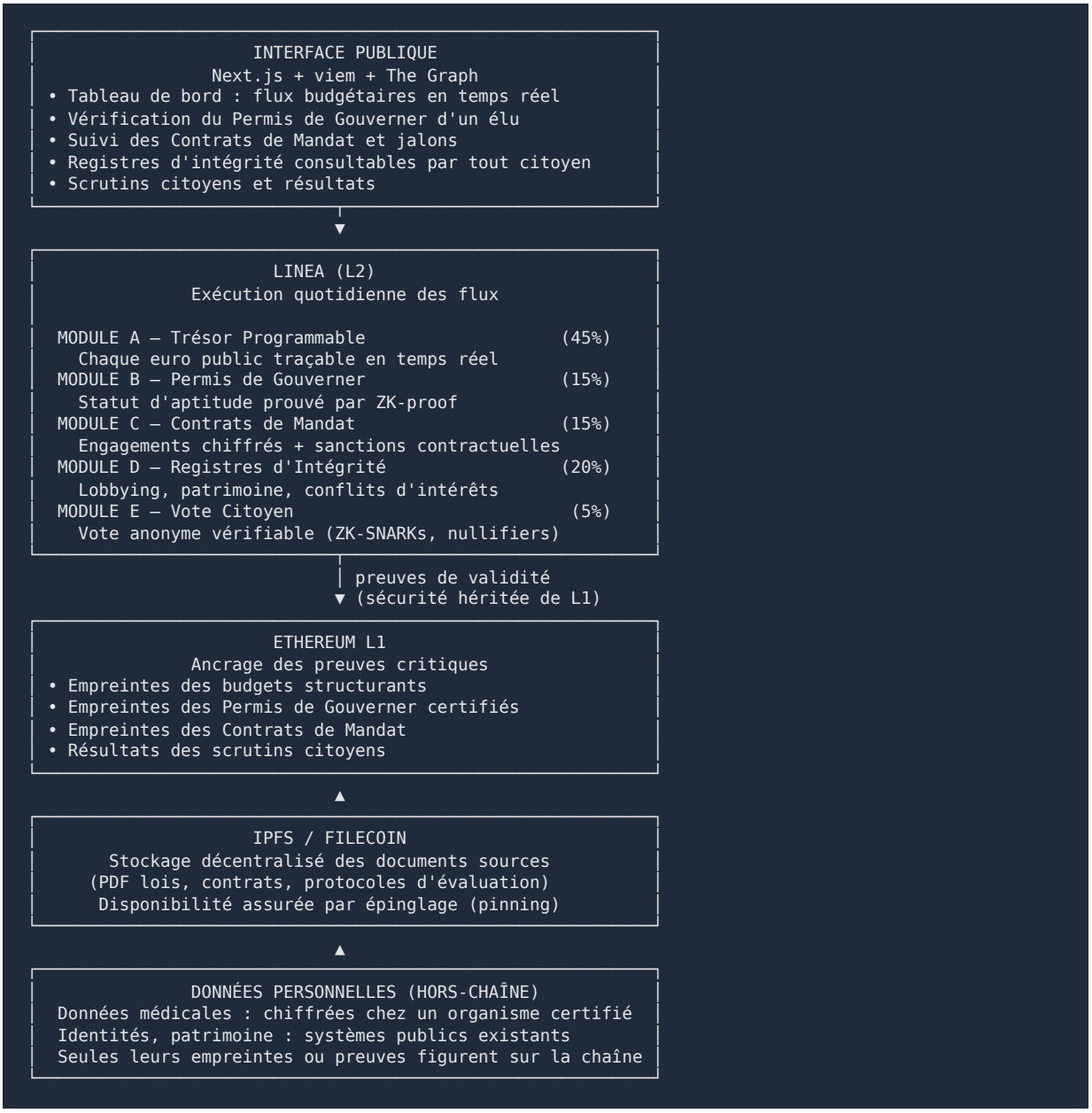
Ce que l'exemple montre : seul l'organe constitutionnel crée un scrutin ; la preuve est liée à la liste électorale, au scrutin et à l'option choisie, ce qui empêche un intermédiaire de modifier le vote ; chaque nullifier ne sert qu'une fois par scrutin ; une option inexistante est refusée. Le choix exprimé est public, mais il n'est rattaché à aucune identité.

Outils spécifiques au Module E

Outil	Usage	Lien
Semaphore	Protocole ZK d'identité anonyme : appartenance à un groupe + nullifiers	docs.semaphore.pse.dev
MACI	Vote chiffré et anti-collusion pour les scrutins sensibles	maci.pse.dev
Circom	Création des circuits de vote	docs.circom.io
SnarkJS	Génération et vérification des ZK-SNARKs	github.com/iden3/snarkjs

SEMAPHORE est particulièrement recommandé : c'est un protocole open source développé par l'équipe PSE (Privacy & Scaling Explorations), soutenue par l'Ethereum Foundation, conçu pour le vote et la signalisation anonymes. Il implémente nativement les nullifiers et les ZK-proofs d'appartenance à un groupe. Pour les scrutins les plus sensibles, MACI y ajoute le vote chiffré et la résistance à la collusion.

Architecture Système Complète — Les 5 Modules



Outils Techniques — Liste Complète

A. Environnement de développement

Outil	Usage	Lien
Node.js v22+	Runtime JavaScript (Hardhat 3 exige la v22.13 ou plus)	nodejs.org
VS Code	Éditeur de code	code.visualstudio.com
Git	Versioning	git-scm.com
npm / pnpm	Gestionnaire de paquets	npm inclus avec Node.js · pnpm.io

B. Smart Contracts

Outil	Usage	Lien
Hardhat 3	Framework : développement, tests, déploiement	hardhat.org
Foundry	Alternative rapide (tests)	getfoundry.sh
Solidity v0.8.x	Langage des smart contracts	docs.soliditylang.org
OpenZeppelin Contracts	Bibliothèque de contrats audités (contrôle d'accès, etc.)	openzeppelin.com

Installation Hardhat :

```
mkdir etat-contrat && cd etat-contrat
npx hardhat --init
```

C. ZK-Proofs — Spécifiques au Module B (Permis de Gouverner)

Outil	Usage	Lien
Circom	Création de circuits ZK	docs.circom.io
SnarkJS	Génération et vérification ZK-proofs	github.com/iden3/snarkjs
Noir (Aztec)	Langage ZK plus accessible	noir-lang.org
Privado ID (ex-Polygon ID)	Identité auto-souveraine avec ZK-proofs, attestations vérifiables	privado.id

Les ZK-proofs sont la technologie clé du Module B : ils permettent de prouver publiquement l'aptitude d'un élu SANS jamais révéler ses données médicales. C'est la solution la plus adaptée au secret médical.

D. Connexion aux réseaux

Outil	Usage	Lien
Infura	Nœud Ethereum/Linea en API	infura.io
Alchemy	Alternative à Infura	alchemy.com

Outil	Usage	Lien
Linea RPC public	Accès direct au réseau Linea (développement)	rpc.linea.build

Configuration Linea dans Hardhat :

```
linea_mainnet: { url: 'https://rpc.linea.build',         chainId: 59144 }
linea_sepolia: { url: 'https://rpc.sepolia.linea.build', chainId: 59141 }
```

E. Wallets et sécurité des clés

Outil	Usage	Lien
MetaMask	Wallet de développement et tests	metamask.io
Ledger Hardware Wallet	Sécurisation des clés en production	ledger.com
Safe (multisignature)	Administration des contrats : aucune décision ne dépend d'une seule clé	safe.global
dotenv	Variables d'environnement sécurisées	npm install dotenv

RÈGLE ABSOLUE : Ne jamais mettre une clé privée dans le code source. Toujours utiliser .env + .gitignore. En production, les rôles d'administration des contrats sont détenus par des coffres multisignatures, jamais par une clé unique.

F. Tests et audit de sécurité

Outil	Usage	Lien
Hardhat Test	Tests unitaires et d'intégration	hardhat.org
Slither	Analyse statique de sécurité	github.com/crytic/slither
Mythril	Détection de vulnérabilités	github.com/Consensys/mythril
OpenZeppelin Monitor	Surveillance en production (open source, successeur de Defender)	openzeppelin.com

OpenZeppelin Defender a été retiré le 1er juillet 2026 ; ses fonctions sont reprises par les outils open source Monitor et Relayer.

G. Interface publique — Tableau de bord citoyen

Outil	Usage	Lien
Next.js	Framework React	nextjs.org
viem / ethers.js	Connexion JavaScript ↔ blockchain	viem.sh · docs.ethers.org
wagmi	Hooks React pour blockchain	wagmi.sh
The Graph	Indexation des données blockchain	thegraph.com
Vercel	Hébergement web	vercel.com

H. Explorateurs publics

Outil	Usage	Lien
Etherscan	Explorateur Ethereum L1 — audit citoyen	etherscan.io
Lineascan	Explorateur Linea L2 — audit citoyen	lineascan.build

I. Stockage décentralisé des documents

Outil	Usage	Lien
IPFS	Stockage décentralisé (PDF, protocoles), adressé par empreinte	ipfs.tech
Pinata	Épinglage IPFS — assure la disponibilité des fichiers	pinata.cloud
Filecoin	Stockage avec engagement contractuel de durée	filecoin.io

Environnement de Test et Planning

Testnets gratuits — commencer sans risque

Réseau	Usage	Faucet ETH gratuit
Sepolia (Ethereum testnet)	Test Layer 1	sepoliafaucet.com
Linea Sepolia (Linea testnet)	Test Layer 2	Faucet MetaMask (docs.metamask.io) ou pont Sepolia → Linea

Planning de mise en œuvre

Phase	Durée	Contenu	Coût
0 — Préparation	1-2 sem.	Installation, testnets, Hardhat, MetaMask	0 €
1 — Module A prototype	4-6 sem.	Trésor Programmable + interface de lecture basique	0 € (testnet)
2 — Modules B, C, D prototypes	8-10 sem.	Permis + ZK-proofs + Contrats + Registres	0 € (testnet)
3 — Module E prototype	4-6 sem.	Vote Semaphore, relais, interface de vote	0 € (testnet)
4 — Audit sécurité	6-10 sem.	Audit externe des contrats et des circuits ZK + corrections	30 000 – 150 000 € (ordre de grandeur)
5 — Déploiement	2-3 sem.	Linea Mainnet + ancrage Ethereum L1	100 – 500 €
6 — Exploitation	Continu	API publique, The Graph, documentation	20 – 100 €/mois

À LIRE AVANT DE PLANIFIER : les coûts indiqués sont les coûts techniques, hors rémunération de l'équipe. Ce planning couvre des prototypes sur réseau de test. Un déploiement réel suppose en amont le cadre juridique des Modules B, C et D, et une expérimentation locale encadrée.

Équipe et Ressources

Compétences nécessaires

Profil	Rôle	Module(s)
Développeur Solidity	Smart contracts principaux	A, B, C, D, E
Expert ZK-proofs (Circom/Noir)	Architecture du Permis de Gouverner	B
Développeur Full-Stack Next.js	Interface publique et tableau de bord citoyen	Tous
Expert sécurité blockchain	Audit des contrats avant déploiement	Tous
Juriste constitutionnaliste	Traduction juridique des engagements	B, C, D
Juriste protection des données (RGPD)	Conformité des données personnelles et de santé	Tous
Psychologue clinicien	Définition des protocoles d'évaluation	B
Expert ZK vote (Semaphore/MACI)	Architecture du vote anonyme vérifiable	E
DevOps	Infrastructure, déploiement, monitoring	Tous

Ressources documentaires

Ressource	URL
Ethereum Developer Docs	ethereum.org/developers
Linea Developer Docs	docs.linea.build
Hardhat Documentation	hardhat.org/docs
Solidity Documentation	docs.soliditylang.org
OpenZeppelin	docs.openzeppelin.com
Circom (ZK-proofs)	docs.circom.io
SnarkJS	github.com/iden3/snarkjs
Noir Language	noir-lang.org/docs
The Graph	thegraph.com/docs
IPFS	docs.ipfs.tech
Semaphore (vote ZK)	docs.semaphore.pse.dev
MACI (anti-collusion)	maci.pse.dev

Institut de l'État-Contrat

contact@institutetatcontrat.fr · www.institutetatcontrat.fr

« La paix n'est pas un idéal : c'est un effet secondaire d'un cadre clair. »