FORUM FINTECH ACPR-AMF

Rapport du Groupe de travail sur la certification des smart contracts

Résumé

Un groupe de travail a été établi en janvier 2024 par l'ACPR et l'AMF, sous l'égide du Forum Fintech ACPR-AMF, afin d'explorer la question de la certification des *smart contracts*, dans l'éventualité d'une règlementation à venir de la DeFi. Au cours de l'année 2024, ses travaux ont porté successivement sur les standards possibles de certification, les modalités d'audit en vue d'une certification et la façon dont une règlementation pourrait intégrer de tels mécanismes de certification.

Ce rapport reflète les travaux et les discussions du groupe de travail, et ne constitue ni une proposition d'encadrement règlementaire, ni une position officielle de l'ACPR, de l'AMF, ou d'une autre autorité ayant participé à ce groupe de travail.

Standards

Le groupe de travail s'est attaché à dégager des principes de haut niveau, applicables à différents environnements d'exécution. Ces principes pourraient être les suivants :

Principes de sécurité :

1. Le code informatique du *smart contract* est conforme aux bonnes pratiques de conception logicielle, et permet en particulier d'éviter les défaillances déjà documentées.
2. Le code informatique du *smart contract* permet d'atténuer les vulnérabilités découlant de l'environnement d'exécution du *smart contract*.
3. La compilation du code informatique du *smart contract* est déterministe et reproductible.
4. Le code informatique du *smart contract* est publié, afin d'assurer la transparence du service rendu et de renforcer la confiance des utilisateurs.
5. Chaque entité dispose uniquement des autorisations strictement nécessaires (principe du moindre privilège).
6. Les tâches et privilèges servant des processus métiers distincts sont répartis entre plusieurs personnes distinctes (principe de séparation des tâches).
7. L'ensemble du cycle de vie du *smart contract* est sécurisé, en particulier les changements subséquents.

Principes de gouvernance :

8. Les processus de prise et de mise en œuvre des décisions concernant le *smart contract* (ou le protocole) sont décrits clairement et avec précision, mis à jour sans délai et publiés.
9. Les dispositifs de gouvernance prévoient des règles préservant les intérêts des clients et utilisateurs en cas de modification du *smart contract* (information, préavis, possibilité de sortie, etc.).
10. Les dispositifs de gouvernance comportent des mécanismes d'urgence permettant de réagir rapidement à une attaque ou à une vulnérabilité. Des garde-fous à ces mesures exceptionnelles sont prévus (cas d'activation, actions possibles), afin de limiter les effets d'une rupture d'activité et d'éviter les risques de prise de contrôle par les entités chargées de protéger temporairement le protocole ou le *smart contract*.

Principes de conformité du service :

11. Un *smart contract* est accompagné d'une documentation décrivant avec exactitude les services rendus, le niveau de qualité attendu, les risques liés à l'utilisation, les responsabilités des différentes parties prenantes, ainsi que les mécanismes de gouvernance.
12. Un *smart contract* génère pour l'utilisateur un montant raisonnable de frais relativement au service rendu. À cet effet, l'architecture du code est optimisée, en tenant compte des exigences découlant des objectifs de sécurité.

Par ailleurs, la forme et les caractéristiques de la certification devraient prendre en compte les *possibilités d'évolution du service*, selon les principes suivants :

13. La certification est obtenue pour un état donné du *smart contract*, des connaissances informatiques, ainsi que de son environnement, et pour une durée limitée. À l'issue de cette période, elle devrait être renouvelée.
14. En cas de modification du *smart contract*, la certification devrait être renouvelée. Des dispositifs de proportionnalité, définis *ex ante* en fonction de la nature des changements opérés, peuvent cependant être envisagés.
15. Un certificat est obtenu à l'issue du processus de certification : il n'est pas falsifiable, disparaît automatiquement quand la certification n'est plus valide et est facilement consultable et compréhensible par les utilisateurs ; à cet effet, le certificat est notamment accessible par la couche des interfaces utilisateurs.

Audit

Les audits actuels de *smart contracts*, inspirés des démarches classiques de l'audit informatique (définition du périmètre et des méthodes, revue des protocoles, formulation de recommandations), peuvent servir de base à un processus de certification des *smart contracts*, pour peu que soit ajoutée une étape supplémentaire, celle de la décision d'attribution de la certification. À cet égard, trois schémas sont envisageables, bien que ne fournissant pas le même niveau de garantie ou la même capacité de traitement : (i) la certification est délivrée par une autorité publique, sur la base d'un audit effectué par un organisme agréé ; (ii) la certification est directement délivrée par l'auditeur ; (iii) le fournisseur du *smart contract* procède à une auto-certification et ne fait l'objet que d'une surveillance ex-post.

L'éventail des méthodes d'audit (méthodes d'analyse statiques, manuelle ou automatique, méthodes dynamiques – tests unitaires ou *fuzzing* –, méthodes de vérification formelle) apparaît suffisamment large et l'état de l'art, quoi que très évolutif, suffisamment avancé pour envisager un mécanisme de certification robuste. Les méthodes automatisées sont généralement indiquées pour la validation des principes de sécurité et pour certaines d'entre elles, pour certains principes de conformité. Il semble en revanche probable que l'examen de la gouvernance, notamment lorsque celle-ci est partiellement *off-chain*, nécessite une revue exclusivement manuelle.

À l'issue de l'audit, la certification devrait être accordée pour une durée limitée, avec deux options possibles identifiées par le groupe de travail : une durée de validité fixe, par exemple 3 ans comme dans d'autres dispositifs de certification, ou une durée dépendant de la criticité ou de la complexité du *smart contract*. Dans tous les cas, un changement « majeur » du *smart contract* devrait entrainer une nouvelle certification : afin d'objectiver les cas nécessitant une nouvelle certification, l'audit initial pourrait être chargé de définir le périmètre des composants critiques dont la modification ferait tomber la certification.

Le groupe de travail a également évoqué la question des compétences disponibles pour effectuer les audits de certification. Il apparaît qu'à ce jour, l'industrie du *smart contract* est confrontée à une pénurie de talents pour mener à bien ses démarches d'audit. Il est toutefois probable qu'un dispositif de certification généralisé serait de nature à créer un marché de l'audit des *smart contracts* plus profond et plus mûr.

Réglementation

Une réglementation prévoyant la certification des *smart contracts* devrait avoir pour objectifs principaux la protection de la clientèle et des investisseurs ainsi que l'établissement et le maintien de la confiance dans la DeFi.

Cette réglementation ne s'appliquerait pas aux *smart contracts* utilisés (i) à des fins non financières, (par exemple pour fournir des NFT), ou (ii) dans le cadre d'activités déjà réglementées (par exemple par le règlement MiCA pour des *stablecoins*). Pour être efficace, cette réglementation devrait s'appliquer au minimum à l'échelle de l'Union européenne (UE).

La certification s'apprécierait au niveau du service rendu et donc du protocole (l'agencement de *smart contracts* permettant de rendre le service). Les trois volets du standard (sécurité, gouvernance, conformité) s'appliqueraient au protocole, tandis que la certification de chaque *smart contract* du protocole serait accordée sur le seul volet sécurité. Un registre des certifications permettrait de rendre accessible à tous l'information sur les protocoles certifiés.

Des mesures de proportionnalités pourraient être prévues pour moduler le niveau des exigences (allégées, standard, ou élevées) en fonction du risque de chaque protocole. Les critères pour évaluer ce risque pourraient être, selon les cas : la *Total Value Locked* (TVL), le volume des transactions, le nombre de transactions ou le nombre d'utilisateurs.

Des trois schémas de réglementation envisageables : (i) la certification optionnelle, (ii) la certification obligatoire pour tous les *smart contracts*, (iii) la certification obligatoire avec des mesures de proportionnalité, c'est ce dernier qui pourrait apporter le plus de garantie aux utilisateurs et aux investisseurs sans freiner indûment l'innovation. Une telle certification obligatoire supposerait d'interdire à l'ensemble des intermédiaires opérant dans l'UE d'interagir avec des *smart contracts* non certifiés.

Parmi les trois grandes modalités possibles d'organisation de la certification, la certification *ex ante* délivrée par une autorité publique agréant pour ce faire un ensemble d'évaluateurs tiers présente des avantages en théorie. Toutefois, si le nombre de certification à délivrer était important, la certification *ex ante* accordée directement par des évaluateurs agréés paraîtrait plus réaliste.

Le coût de la certification, variable en fonction du niveau de complexité des protocoles, devrait être prioritairement supporté par le producteur du *smart contract*, qui est théoriquement l'entité qui a le plus intérêt à demander la certification.

Le cadre réglementaire envisagé devrait donner aux autorités publiques des pouvoirs adaptés à la surveillance des « produits » mis sur le marché. Les autorités publiques devraient en particulier pouvoir élaborer ou valider les grands principes du standard, de la sélection des évaluateurs et de la décision de certification.

Introduction – contexte

À la suite des travaux de l'ACPR[1] et de l'AMF[2] sur la finance décentralisée ou désintermédiée (DeFi), les deux Autorités ont proposé de créer un groupe de travail sous l'égide du Forum Fintech ACPR-AMF afin d'explorer la question de la certification des *smart contracts*, comme élément possible d'un dispositif réglementaire futur de la DeFi.

Ce groupe de travail, co-animé par l'ACPR et l'AMF, est composé de membres représentants du secteur des crypto-actifs, d'experts techniques; ainsi que d'autorités concernées. Au cours de l'année 2024, il s'est réuni à onze reprises et a examiné les trois thématiques suivantes :

1. Les **standards de certification** existants ou à créer pour envisager une certification des *smart contracts*.
2. Les **méthodes d'audit** existantes, et leur possible utilisation dans le cadre d'une certification, ainsi que les organismes ou les moyens susceptibles de les mettre en œuvre.
3. La façon dont une **réglementation** pourrait intégrer un mécanisme de certification, en précisant les objectifs, le champ, les mécanismes de proportionnalité envisageables dans cette hypothèse, ainsi que les questions de mise en œuvre pratique.

Le présent **rapport** rend compte de l'ensemble de ces travaux. Au moment où la Commission européenne s'interroge sur les suites à donner au règlement MICA, ce rapport a vocation à éclairer le débat public sur la finance décentralisée et plus généralement sur le fonctionnement de services rendus[3] par les *smart contracts* sur la *blockchain* et comparables à des services financiers. Ce rapport ne constitue pas une position de l'ACPR ou de l'AMF.

Issu des travaux d'un groupe à la représentation très diversifiée, il fera l'objet d'une **consultation** largement ouverte afin de compléter l'analyse menée dans ces pages (*cf.* questionnaire de consultation en annexe).

---

[1] « Finance « décentralisée » ou « désintermédiée » (DeFi) : quelle réponse réglementaire ? » (document de réflexion : https://acpr.banque-france.fr/finance-decentralisee-ou-desintermediee-quelle-reponse-reglementaire ; synthèse de la consultation publique : https://acpr.banque-france.fr/finance-decentralisee-ou-desintermediee-defi-quelle-reponse-reglementaire)

[2] « Papier de discussion de l'AMF sur la Finance décentralisée (DeFi) » (https://www.amf-france.org/fr/actualites-publications/communiques/communiques-de-lamf/lamf-publie-un-papier-de-discussion-sur-la-finance-decentralisee-defi)

[3] Lorsqu'ils renvoient aux *smart contracts*, certains termes utilisés dans ce document, tels que « service » ou « produit », sont employés dans leur sens courant et ne doivent pas être interprétés comme une qualification juridique au sens des règlementations existantes. Ils permettent de caractériser simplement certaines activités observées sur la blockchain, et d'établir des analogies utiles à la réflexion développée dans le document.

Table des matières

Résumé 1

Introduction – contexte 4

PARTIE 1 : STANDARDS 7

1. Le contenu d'un standard de certification des *smart contracts* 7

1.1. Considérations générales 7
1.1.1. Établir un standard général ou un standard par *blockchain* ? 7
1.1.2. Quelle inspiration tirer de l'existant ? 7
1.1.3. La transparence du code des *smart contracts* suffit-elle ? 8
1.2. Sécurité 9
1.2.1. La sécurité informatique des *smart contracts* : de quoi parle-t-on ? 9
1.2.2. Faut-il prendre en compte les composantes externes avec lesquelles un *smart contract* interagit ? 10
1.2.3. Quels sont les grands principes de sécurité qui devraient figurer dans un standard ? 10
1.2.4. Au regard de ces exigences, existe-t-il aujourd'hui des standards satisfaisants ? 12
1.2.5. L'intérêt de disposer d'un historique des défaillances constatées sur les *smart contracts* . 13
1.3. Gouvernance 13
1.3.1. Les différents modes de gouvernance des *smart contracts* 14
1.3.2. Les objectifs de la gouvernance 15
1.3.3. Assurer la transparence quant au mode de gouvernance du *smart contract* 15
1.3.4. Assurer aux clients que le service rendu ne sera pas modifié sans préavis 16
1.3.5. Pouvoir réagir rapidement aux risques 16
1.4. Conformité du service 17
1.4.1. Vérifier que le *smart contract* rend le service attendu 17
1.4.2. Vérifier que le service est rendu efficacement 18
1.4.3. Vérifier le respect de la conformité réglementaire 18

2. Gestion de la mutabilité des *smart contracts* 20

2.1. Le cas des protocoles dits « immuables » 20
2.2. La mutabilité des *smart contracts* en pratique 20
2.3. Tout changement apporté à un *smart contract* doit-il rendre la certification caduque ? 21
2.4. Comment formaliser techniquement le certificat obtenu à l'issue du processus de certification ? 22

3. Qui peut et doit établir des standards pour la certification ? 23

PARTIE 2 : AUDIT 25

1. Comment auditer un *smart contract* ? 25

1.1. La démarche générale d'audit 25
1.1.1. L'audit de *smart contract* tel qu'il se pratique aujourd'hui 25
1.1.2. L'audit dans une démarche de certification 28
1.2. Les techniques d'audit du code informatique d'un *smart contract* 28
1.2.1. Les méthodes d'analyse statique 28
1.2.2. Les méthodes d'analyse dynamique 29
1.2.3. La vérification formelle 30
1.3. Quelles méthodes pour quels objectifs ? 32
1.4. Sur quel environnement tester et auditer les *smart contracts* ? 33

2. Quand auditer ou ré-auditer un *smart contract* ? 34

2.1. La certification initiale soulève-t-elle des difficultés particulières ? 34
2.2. Quelle doit être la durée de validité de la certification ? 35

2.3. Quels sont les critères devant déclencher un nouvel audit ? 35
3. Qui peut réaliser l'audit d'un *smart contract* ? 36
3.1. Le marché de l'audit du *smart contract* aujourd'hui 36
3.2. La certification des *smart contracts* en pratique 38
3.2.1. Une certification *ex ante* délivrée par une autorité publique, agréant pour ce faire un ensemble d'évaluateurs tiers 38
3.2.2. Une certification *ex ante* délivrée par des évaluateurs tiers 39
3.2.3. Une surveillance *ex post* plutôt qu'une certification *ex ante* 40
PARTIE 3 : PISTES REGLEMENTAIRES 41
1. Les bases réglementaires pour une certification des *smart contracts* 41
1.1. L'objectif de la réglementation 41
1.2. Le champ d'application 42
1.2.1. Le champ d'application matériel 42
1.2.2. Le champ d'application géographique 42
1.2.3. Que certifier précisément ? 43
1.3. Les critères de proportionnalité 46
1.4. Les schémas de réglementation possibles autour du mécanisme de certification 48
1.4.1. La certification optionnelle 49
1.4.2. La certification obligatoire 50
1.4.3. La certification obligatoire avec des mesures de proportionnalité 50
2. Le mécanisme de certification 51
2.1. La publication de la certification 51
2.2. Qui délivrerait la certification ? 52
2.3. Qui assumerait le coût de la certification ? 55
2.3.1. Le coût de la certification 55
2.3.2. Qui prendrait en charge le coût de la certification ? 55
2.4. Le rôle des pouvoirs publics 55
Annexe – Questionnaire de consultation (février 2025) 57
Annexe – Glossaire 59
Annexe – Membres et participants au groupe de travail 63

PARTIE 1 : STANDARDS

1. Le contenu d'un standard de certification des *smart contracts*

1.1. Considérations générales

1.1.1. Établir un standard général ou un standard par *blockchain* ?

Certains participants au groupe de travail (GT) estiment qu'en raison des différences technologiques entre *blockchains*, il est difficile d'adopter une approche générale, c'est-à-dire qui puisse être valable quelle que soit la technologie sous-jacente. Ainsi, des règles de sécurité valant pour des *smart contracts* déployés sur une *blockchain* ne seraient pas nécessairement ou aisément transposables sur une autre[4].

Cette question pose plus généralement celle de la prise en compte de **l'environnement d'exécution** du *smart contract*. Ainsi des différences d'architecture technique entre les *blockchains* de niveau 1 (*Layer 1* ou *L1*) et les couches de niveau 2 (*Layer 2* ou *L2*)[5] peuvent induire des modalités d'exécution différentes d'un *smart contract*.

À rebours, d'autres participants estiment qu'il est possible de formuler des principes suffisamment abstraits, transcendant les différences technologiques. Ainsi, quelle que soit la technologie *blockchain* employée, un *smart contract* constitue toujours un logiciel gérant un état et permettant aux utilisateurs d'une *blockchain* d'interagir avec cet état.

En combinant ces deux approches, il semble donc **qu'une approche hiérarchisée pourrait être adoptée, avec des principes de haut niveau pouvant s'appliquer à toutes les *blockchains* et leurs interfaces** (ou en tout cas aux principales d'entre elles), **puis des recommandations plus précises dépendant de la technologie ou famille technologique employée**[6].

1.1.2. Quelle inspiration tirer de l'existant ?

Selon les participants du GT, il n'existe pas aujourd'hui de standard faisant autorité dans l'écosystème. Il existe cependant des *check-lists* destinées aux développeurs, regroupant des préconisations pour l'écriture de code[7]. Il existe également certaines bases de données historiques des défaillances constatées dans l'industrie du *smart contract*[8], qui ne sont cependant pas toujours exhaustives ni

---

[4] Malgré l'éclatement du paysage des blockchains, il convient de noter la prévalence actuelle des blockchains basées sur Ethereum (*Ethereum Virtual Machine*, *EVM*).

[5] Par exemple le standard de taille du code compilé (*bytecode*) sur les L2 de type zkEVM comparativement à la blockchain correspondante L1.

[6] En lien, certains participants estiment que la méthode de consensus de la blockchain (*Proof of Work*, *Proof of Stake*, *Proof of Authority* etc.) devrait être un élément à prendre en compte dans le cadre de la certification des *smart contracts*, certaines attaques lui étant spécifiques (exemple : la manipulation des oracles TWAP d'Uniswap s'appuyant sur le Proof of Stake).

[7] On mentionnera notamment les deux exemples suivants : GitHub - nascentxyz/simple-security-toolkit: A collection of practical security-focused guides and checklists for *smart contract* development ; GitHub - transmissions11/solcurity: Opinionated security and code quality standard for Solidity *smart contracts*.

[8] Par exemple : https://rekt.news/leaderboard/ ; https://defillama.com/hacks ; https://www.notion.so/0e85e02c5ed34df3855ea9f3ca40f53b?pvs=21

tenues à jour, du fait de la diversité et du nombre de *blockchains* et protocoles, mais qui n'en constituent pas moins une ressource très utile pour la recherche en sécurité.

Par ailleurs, certaines briques essentielles pour l'écriture des *smart contracts* proviennent de sources externes. On peut citer des **standards d'interopérabilité**, comme le standard ERC-20, qui définit un ensemble de fonctions techniques visant à construire des jetons fongibles sur les *blockchains* compatibles avec Ethereum[9]. Les membres du GT indiquent aussi que certaines bibliothèques ainsi que des ensembles de lignes de code, largement utilisés par les développeurs, sont tenus à jour par des organisations[10].

Il faut ici bien différencier deux types de standards :

- les standards d'interopérabilité, qui permettent d'assurer la communication, l'interconnexion ou l'échange entre des systèmes (y compris des formats de données) ;
- les **standards de fonctionnement**, destinés à fixer des niveaux minimums, et qui incluent donc nécessairement une dimension normative quant à la qualité attendue.

Un standard d'interopérabilité ne fait pas nécessairement un standard de fonctionnement. De même, des morceaux de code « standards » (habituels) ne font pas un standard de sécurité. Ainsi, le fait que certaines bibliothèques[11] soient très largement utilisées dans l'univers des *smart contracts* peut certes permettre de réduire les risques d'erreur dans l'écriture du code, mais une telle configuration **augmente aussi les conséquences en cas de vulnérabilité**. Ce point rejoint la question plus large de l'*open source* : l'utilisation de bibliothèques de code ou de programmes en accès ouvert améliore généralement l'efficacité et l'interopérabilité, mais parfois au détriment de la sécurité[12].

1.1.3. La transparence du code des *smart contracts* suffit-elle ?

D'après les participants du GT, l'autorégulation dans l'industrie du *smart contract* repose aujourd'hui essentiellement sur la visibilité du code informatique. De fait, le **principe de sécurité par transparence** permet à des communautés de développeurs de repérer et de signaler les vulnérabilités. Pour autant, la transparence du code ne permet pas toujours de repérer à temps les défauts d'un programme informatique[13]. La publication du code peut même avoir pour inconvénient de permettre à des attaquants de découvrir des vulnérabilités[14]. Enfin, la possibilité d'examiner le code informatique ne garantit pas la compréhension des mécanismes financiers à l'œuvre dans un *smart contract*. **Pour toutes ces raisons, un mécanisme de certification apparaît nécessaire.**

---

[9] Ce standard s'étend en fait au-delà des protocoles non compatibles avec Ethereum, comme dans les blockchains Tron, Tezos ou Solana, qui s'appuient sur des standards équivalents ou très proches d'ERC-20.

[10] Par exemple la société OpenZeppelin a développé une suite de contrats réutilisables, comme des *smart contracts* spécialisés sur la gouvernance.

[11] Ou d'autres outils, comme des modèles de *smart contracts*, selon un modèle dit « *factory* ».

[12] Certains participants soulignent l'importance de cet enjeu de sécurité dans un contexte où l'open source et les blockchains publiques ont les faveurs de nombreux acteurs du web3.

[13] Comme l'a montré la faille majeure *Log4Shell* sur le logiciel open source *log4j.*

[14] En contrepartie, l'expertise de ces attaquants peut aussi être mise à profit en les employant pour découvrir les vulnérabilités d'un protocole (*white hats*).

1.2. Sécurité

1.2.1. La sécurité informatique des *smart contracts* : de quoi parle-t-on ?

La sécurité informatique vise à empêcher l'utilisation non autorisée, le mauvais usage, la modification non consentie ou le détournement d'un système. Elle se décline classiquement en quatre objectifs dits « CAID »[15] :

- **Confidentialité** : seules les personnes autorisées doivent avoir accès aux informations qui leur sont destinées ; tout accès indésirable doit être empêché.
- **Authenticité** : les utilisateurs doivent prouver leur identité via un élément secret.
- **Intégrité** : les données ne doivent pas être altérées (involontairement ou de façon malveillante).
- **Disponibilité** : l'accès aux ressources du système doit être permanent et sans faille sur les plages prévues.

Dans l'univers de la DeFi, **ces propriétés dépendent au premier chef de l'infrastructure *blockchain***. De fait, la raison d'être des *blockchains* est de garantir l'intégrité des données ; elles assurent généralement aussi la disponibilité, même si elles peuvent connaître des épisodes de congestion[16]. Cependant, la plupart des *blockchains* sur lesquelles se déroulent les activités de la DeFi sont publiques et régies par le pseudonymat : quiconque est ainsi autorisé à interagir avec l'infrastructure, et il n'y a pas vraiment de démarche d'authentification des participants.

Il faut toutefois noter que le « système informatique » auquel accède un utilisateur de la DeFi est schématiquement **composé de trois couches empilées les unes sur les autres** : l'infrastructure *blockchain*, les *smart contracts* fonctionnant sur celle-ci, et la couche des applications utilisateurs (*wallets*, interfaces d'accès). Chacune de ces trois couches, ainsi que les interactions entre elles, contribuent donc à la sécurité de l'ensemble ; les *smart contracts*, situés « au milieu », peuvent donc être vulnérables à des attaques visant l'infrastructure *blockchain* ou la couche d'interface. Quoi qu'importants, certains des aspects de la sécurité informatique de l'ensemble – qui peuvent d'ailleurs impliquer des mesures de sécurisation « classiques » – dépassent cependant le cadre des travaux du GT.

Au niveau de la seule couche des *smart contracts*, la sécurité informatique peut comprendre une dimension de **disponibilité** : sur des *blockchains* fonctionnant en continu, il pourrait ainsi être attendu que les *smart contracts* fassent de même, et le standard de certification pourrait requérir cette fonctionnalité avec des exigences ad hoc[17]. De même la dimension d'**intégrité**, si elle peut s'appuyer sur les caractéristiques des *blockchains* sous-jacentes, peut impliquer des exigences particulières au niveau des *smart contracts* qui reposent sur des variables d'état, des paramètres d'appel et des données externes fournies par des oracles.

Enfin, certains participants estiment que les avancées technologiques sur les preuves à divulgation nulle de connaissance (*zero-knowledge proofs ou ZKP*) pourraient contribuer à renforcer l'authenticité

---

[15] Voir : https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA

[16] Ces questions de congestion expliquent le développement des solutions de layer 2 sur la plupart des blockchains (*cf.* encadré 2). Plus généralement, il faut noter que la question du « passage à l'échelle », c'est-à-dire la capacité à traiter davantage de transactions sans perte d'efficacité, constitue aussi un enjeu de résilience opérationnelle (cas d'une attaque par déni de service, par exemple).

[17] *Cf.* également *infra* les réflexions sur un plan de continuité de l'activité ; celui-ci pourrait notamment intégrer un mécanisme de notification au niveau de l'interface utilisateur en cas d'indisponibilité.

en permettant de prouver l'identité des utilisateurs, tout en respectant la confidentialité de données personnelles.

Dans tous les cas, l'essentiel de la sécurité au niveau des *smart contracts* consiste à s'assurer de **l'absence de faille logicielle**, afin d'empêcher un éventuel attaquant de porter atteinte à leur fonctionnement.

1.2.2. Faut-il prendre en compte les composantes externes avec lesquelles un *smart contract* interagit ?

La sécurité d'un *smart contract* peut aussi être compromise par les éléments externes avec lesquels il interagit : on a déjà évoqué les **couches d'infrastructure et d'interface**, de même que la question des **layer 2** (voir aussi à ce sujet l'encadré 2) ; on peut également mentionner le **compilateur**[18] de code (qui peut introduire des vulnérabilités à des attaques dans certains cas) ou encore les ***smart contracts* composites**[19] ; il faut encore y ajouter un élément important : les **oracles**[20]. Certains aspects liés aux oracles se situent en dehors du périmètre des travaux du GT : ainsi, un oracle envoyant de fausses informations – à la suite d'une attaque ou d'un dysfonctionnement – pourra déclencher à tort l'exécution de certaines opérations, sans que la fiabilité logicielle du *smart contract* soit en cause. En revanche, d'autres éléments concernent plus directement le GT : par exemple, si la manière dont est construit le code informatique d'un *smart contract* le rend particulièrement vulnérable à un arrêt ou un mauvais fonctionnement de l'oracle.

Un participant mentionne également les **wallets**, qui pourraient faire l'objet de bonnes pratiques de sécurité et de protection des utilisateurs[21] ; toutefois, la sécurisation du wallet et des clefs privées doit rester de la responsabilité du fournisseur du wallet et des utilisateurs.
Enfin, un participant indique que certains protocoles permettent à l'utilisateur de choisir certaines composantes externes, comme l'oracle fournissant les données.

1.2.3. Quels sont les grands principes de sécurité qui devraient figurer dans un standard ?

Outre la prise en compte de l'environnement d'exécution et des composantes externes avec lesquelles un *smart contract* interagit (voir 1.2.2), un certain nombre de principes généraux peuvent être énoncés :

---

[18] Le code source d'un smart contract nécessite d'être compilé, afin d'être traduit en un langage intermédiaire « bytecode » qui est le format de déploiement sur la blockchain ; suivant les architectures ce bytecode peut aussi être compilé à son tour en langage machine lors de l'exécution du smart contract sur la blockchain. Les outils compilateurs, notamment s'ils ne sont pas mis à jour, peuvent introduire des vulnérabilités dans le code machine produit, qui peuvent être mises à profit par des attaquants. Le standard EthTrust d'EEA recense ainsi 21 exigences dans ce domaine, faisant écho à plusieurs bulletins d'alerte de sécurité.

[19] *Smart contracts* interagissant avec des smart contract tiers, qui peuvent être considérés comme des composants externes.

[20] La problématique de la manipulation d'oracles est identifiée comme l'une des principales menaces dans l'univers blockchain (source : Chainalysis, *Crypto crime report 2023*).

[21] Par exemple : des exigences de vérification de signature cryptographiques robustes pour l'authentification ; l'implémentation de limites les montants de transactions ; des fonctions/solutions de recouvrement de clés perdues/volées ; des alertes en cas de comportement suspect du smart contract.

- **Principe de séparation des tâches** : selon ce principe, par exemple, la personne déployant et opérant un *smart contract* ne peut être la même que celle qui conçoit et développe ce *smart contract*.
- **Principe du moindre privilège** : ce principe stipule que les entités (utilisateurs etc.) doivent avoir uniquement les autorisations nécessaires pour accomplir leur tâche, afin de limiter la surface d'attaque.
- **Cycle de vie sécurisé** : adopter un cycle d'évolution sécurisé, qui pourrait comprendre de façon exhaustive les phases suivantes : la conception sécurisée (incluant les spécifications), le développement sécurisé, les tests de sécurité, l'audit initial[22], le déploiement, le monitoring (surveillance[23] et audits de suivi réguliers, ainsi que la gestion des incidents[24]).
- Plus généralement, il peut être souligné que la **qualité d'écriture du code** ou le recours à des **bonnes pratiques de conception logicielle** (*design pattern*) sont de nature à limiter le risque de créer involontairement des failles de sécurité. Sur Ethereum, l'utilisation de composants « *package* » peut faciliter le recours à ces bonnes pratiques logicielles. Dans tous les cas, le recours à des standards ou des référentiels sur la sécurité doit s'accompagner d'une consultation permanente des sources de données actualisées du domaine (forums de cyber-sécurité dédiés, réseaux sociaux de référence etc.)
- **Principe de transparence du code** : dans la mesure du possible, le code informatique doit être publié afin d'instaurer une confiance dans le code déployé et favoriser une détection pro-active des vulnérabilités
- Plus spécifiquement, la **compilation du code d'un *smart contract* est déterministe,** en vue d'assurer la reproduction d'un code binaire déployé à partir de son code source. Ce principe doit permettre d'assurer qu'un code binaire déployé a été compilé à partir d'un code source de confiance, et qu'il n'a ainsi pas été altéré par une injonction de code malicieuse.

PRINCIPES DE SECURITE

Proposition 1 : le code informatique du *smart contract* est conforme aux bonnes pratiques de conception logicielle, et permet en particulier de prévenir les défaillances déjà documentées.

Proposition 2 : le code informatique du *smart contract* permet d'atténuer les vulnérabilités découlant de l'environnement d'exécution du *smart contract*.

Proposition 3 : la compilation du code informatique du smart contract est déterministe et reproductible.

Proposition 4 : le code informatique du smart contract est publié, afin d'assurer la transparence du service rendu et renforcer la confiance des utilisateurs.

Proposition 5 : chaque entité dispose uniquement des autorisations strictement nécessaires (principe du moindre privilège).

[22] Qui pourrait par exemple comprendre un contrôle de l'intégrité du smart contract, afin de s'assurer que celui-ci fonctionne comme prévu, et que les données qu'il manipule restent intactes (intègres).

[23] Un participant mentionne à ce titre l'émergence de protocoles avancés sur le marché, capables d'identifier des schémas de transactions malveillantes ou de permettre aux développeurs de spécifier des schémas indésirables. Dans ce cas, un smart contract pourrait bloquer uniquement les transactions correspondant à ces critères.

[24] Des participants ont également mentionné le Plan de continuité d'activité (PCA), qui s'accompagne usuellement du Plan de reprise d'activité (PRA) ; une défaillance dans un smart contract critique pourrait ainsi déclencher une bascule vers un autre smart contract (ou package) dans un contexte *on-chain*, ou bien vers un processus ad hoc exécuté dans un contexte *off-chain*. Un participant pointe néanmoins les risques de vulnérabilité additionnels que peuvent poser la mise en œuvre d'un mécanisme de secours, qui doivent donc être pris en compte et qui rejoignent notamment les considérations liées à la gouvernance (*cf.* section dédiée).

Proposition 6 : les tâches et privilèges servant des processus métiers distincts sont répartis entre plusieurs personnes distinctes (principe de séparation des tâches).

Proposition 7 : l'ensemble du cycle de vie du *smart contract* est sécurisé, en particulier les changements subséquents.

1.2.4. Au regard de ces exigences, existe-t-il aujourd'hui des standards satisfaisants ?

Le standard EthTrust de l'Ethereum Enterprise Alliance (EEA) constitue selon certains participants une bonne base de standard pour le développement de *smart contracts* sécurisés et la vérification de la sécurité des *smart contracts* (*cf.* encadré 1). Il a en effet été conçu en vue de permettre une certification de sécurité au regard des vulnérabilités connues sur les *smart contracts* écrits en Solidity (autrement dit pour l'Ethereum Virtual Machine ou EVM).
Pour un autre participant, ce standard de sécurité n'est toutefois pas suffisant : sa lecture doit être complétée en pratique (lors d'un audit par exemple) par d'autres sources de référence, comme les notifications ou les forums d'échange de la communauté, qui permettent de diffuser et partager des informations actualisées sur les dernières vulnérabilités identifiées.

Enfin un participant souligne l'intérêt, dans ce standard, de la distinction en plusieurs niveaux de sécurité de complexité croissante, et suggère l'idée d'y adjoindre un niveau supplémentaire couvrant la bonne gestion des fonctions de haut niveau[25].
D'autres standards plus généraux, mais abordant également la sécurité des *smart contracts*, ont été mentionnés par plusieurs participants, comme le standard ERC 3643 sur la tokénisation des actifs[26].

Encadré 1 : Le standard de sécurité EthTrust de l'Enterprise Ethereum Alliance (EEA)

Ce standard, dont la dernière version a été publiée en décembre 2023 (v2), est un **référentiel d'exigences de sécurité élaboré en vue d'une démarche de certification**, afin d'attester qu'un *smart contract* écrit en langage Solidity n'est pas vulnérable à une liste d'attaques ou de défaillances connues, et fonctionne comme attendu. Il combine analyse statique et analyse dynamique. Sa consultation par les développeurs est également encouragée en amont du déploiement du *smart contract*.

Les exigences couvrent, à un niveau plus ou moins détaillé, différentes composantes, telles que : les mécanismes de mise à jour et de déploiement des *smart contracts*, les oracles et les interactions externes, les mécanismes de signature, la consommation du gaz de la *blockchain*, les mécanismes de manipulation de type *Maximum extractable value* (MEV), ou encore les compilateurs.

Ce standard définit trois niveaux de sécurité, suivant un niveau croissant de complexité des exigences : « S » rassemble les exigences suivant des schémas bien connus et pouvant être testées automatiquement par des outils ; « M » requiert une analyse plus stricte avec l'intervention d'un auditeur humain ; « Q » implique une analyse de la logique métier et une vérification de la conformité du code vis-à-vis de sa documentation (autrement dit : que le *smart contract* effectue les tâches attendues). Pour chaque niveau, les exigences sont complétées par une série de bonnes pratiques, optionnelles mais facilitant l'atteinte du niveau. Sur cette base, le standard prévoit une certification par niveau de sécurité (un *smart contract* peut être certifié en sécurité au niveau S, M ou Q) ; ces niveaux sont hiérarchisés (S < M < Q) de sorte qu'une certification à un niveau donné

[25] Par exemple la gestion d'un coffre-fort (*vault*), d'un transfert ou bien encore d'une gouvernance *on-chain*.
[26] Cette norme propose notamment de vérifier l'identité et l'éligibilité des détenteurs de tokens via des fournisseurs d'identité décentralisée.

implique, outre la conformité avec toutes les exigences de ce niveau, la conformité avec toutes les exigences de ce niveau ainsi qu'avec celles des niveaux inférieurs.

L'Enterprise Ethereum Alliance, auteur de ce standard, se définit comme une organisation de représentants de l'industrie visant à promouvoir l'adoption en entreprise de la technologie de la *blockchain* Ethereum en tant que standard ouvert. EEA élabore des spécifications, tests et programmes de certification. L'organisation comprend actuellement 64 membres issus de diverses industries (dont *blockchain* et web3, banque, logiciel, audit).

EEA a élaboré à ce jour six spécifications actives et gère cinq groupes de travail ; des orientations ont récemment été publiées sur l'évaluation et l'atténuation des risques dans la DeFi[27].

1.2.5. L'intérêt de disposer d'un historique des défaillances constatées sur les *smart contracts*

Plusieurs raisons plaident pour l'existence d'une base de données retraçant l'historique des défaillances constatées sur les *smart contracts* :

- Évaluation des risques : pour les auditeurs et assureurs, un historique des défaillances permet de mieux évaluer les risques liés au *smart contract*.
- Amélioration continue de la sécurité : en gardant une trace des failles existantes, les développeurs peuvent analyser les schémas récurrents et renforcer la sécurité des futurs *smart contracts*.
- Audit : un historique des défaillances fournit des pistes d'audit, et permet de voir quelles actions ont été entreprises pour remédier à la vulnérabilité.

Un participant propose de s'inspirer du modèle des *CVE* (*Common Vulnerabilities and Exposures*) de l'industrie logicielle traditionnelle.

Encadré 2 : Les « layer 2 »

En raison des besoins croissants de passage à l'échelle (« *scalability* »), d'interopérabilité et de maîtrise des coûts des transactions, les opérations de la DeFi s'inscrivent de plus en plus dans les environnements layer 2 des *blockchains*. Cette transition peut impliquer des spécificités technologiques.

Ainsi, dans l'écosystème Tezos, la logique de programmation repose sur une architecture dite de noyaux, qui se substitue aux *smart contracts*. La question du périmètre de la certification, et donc des techniques de certification, peut donc se poser au-delà du périmètre des *smart contracts* en layer 1.

Dans ce contexte, les discussions dans le GT ont mis en évidence qu'une démarche de certification des *smart contracts* ne pouvait se limiter au seul examen du code : le contexte d'exécution doit également être pris en compte. Ce qui met en exergue une question résiduelle : des composantes de l'architecture en layer 2, même écrites sous des formes autres que le *smart contract*, devraient-elles intégrer le périmètre de certification ? Certains participants y sont favorables, en considérant un seuil minimum en matière de volume d'activité.

1.3. Gouvernance

[27] Voir notamment la section 5 sur l'atténuation des risques (et en particulier la section 5.1.2 *Mitigating software risks*)

La gouvernance est l'instance – ou l'ensemble des instances – qui prend les décisions relatives au fonctionnement et à la modification d'un *smart contract*. Elle est parfois clairement formalisée – avec un organe de gouvernance identifié, des règles de décision bien établies etc. – tandis que certains *smart contracts* sont gouvernés *de facto* par les possesseurs des clés d'administration (avec le risque induit lié à la conservation de ces clefs[28]), mais sans que ce point ne fasse l'objet d'une communication claire aux utilisateurs.

1.3.1. Les différents modes de gouvernance des *smart contracts*

Abstraction faite du cas particulier des *smart contracts* réellement « immuables », l'écosystème se caractérise par une grande variété des modes de gouvernance des *smart contracts* et un degré variable dans leur formalisation. Cette variabilité s'illustre dans les deux grands processus d'administration d'un *smart contract* : la prise de décision et la mise en œuvre d'une décision.

S'agissant tout d'abord de la prise de décision, deux modalités se dégagent de l'observation de l'écosystème :

i. **La prise de décision centralisée** : un individu ou une entité – ou des groupes d'individus ou d'entités agissant en concertation – décident seuls des modifications à apporter à un *smart contract* ou à un protocole[29]. Le cas échéant, la prise de décision peut être précédée de consultations ou de sondages au sein de la communauté des parties prenantes du protocole.

ii. **La prise de décision par des votes communautaires**, généralement organisés au sein du protocole lui-même. Le degré de décentralisation dépend alors du type de pondération des votes (généralement ceux-ci sont pondérés par le nombre de jetons de gouvernance détenus par chaque adresse sur la *blockchain*) et, le cas échéant, de la répartition des jetons de gouvernance au sein de la communauté des détenteurs. Ainsi, si les jetons de gouvernance sont majoritairement concentrés dans les mains d'un individu ou d'un groupe d'individus liés entre eux, le vote ne diffère guère de la prise de décision centralisée.

Une fois la décision prise, la manière de la mettre en œuvre peut également différer. Schématiquement, deux grands pôles existent aujourd'hui dans l'écosystème, entre lesquels peuvent exister une variété de situations :

i. **Mise en œuvre centralisée** : un développeur ou un groupe de développeurs est chargé de mettre en œuvre la modification décidée.

ii. **Mise en œuvre automatisée** : un *smart contract* exécute directement la modification (changement de paramètre, changement de code). Ce mécanisme ne peut évidemment être utilisé que pour mettre en œuvre des changements déjà écrits sous forme de code. Cette modalité fait écho au principe de « minimisation de la gouvernance » (cf. aussi la section sur la mutabilité des *smart contracts*).

Ces différentes modalités apparaissent finalement indépendantes de l'existence éventuelle d'une DAO (*decentralised autonomous organisation*). De fait, une DAO peut tout aussi bien constituer un outil au

[28] Typiquement le vol ou la perte de ces clefs ; au-delà de la prévention usuelle via les dispositifs de signatures multiples (*multisig*), d'autres solutions existent (procédures strictes de gestion des clefs, solutions institutionnelles centralisées, initiatives privées ou *self custody*, mécanismes d'urgence ou de récupération des clefs) ; celles-ci présentent toutefois des degrés de maturité disparates.

[29] Un protocole consiste en un empilement de *smart contracts*. Souvent, les décisions de modification se font au niveau d'un protocole entier, et impliquent donc un ensemble de *smart contracts*. Les règles établies au niveau élémentaire d'un smart contract donné restent toutefois valables au niveau supérieur du protocole.

service d'une gouvernance largement décentralisée qu'un paravent à une administration en réalité très centralisée[30].

1.3.2. Les objectifs de la gouvernance

Dans le secteur financier traditionnel, la gouvernance répond à **deux objectifs principaux : la maîtrise des risques** – qui inclut la capacité de réagir en cas de difficulté –, **et le respect des engagements et des règles de protection des clients**. Ces deux objectifs devraient valoir également pour des services « financiers » distribués sous une forme décentralisée, même si les modalités pour les atteindre peuvent différer. **Un troisième objectif devrait être ajouté**, afin de prendre en compte le caractère décentralisé de la fourniture du service, qui peut être à l'origine d'une certaine opacité : **les utilisateurs doivent pouvoir être informés de la manière dont la gouvernance est organisée**. Ce troisième objectif a d'autant plus de sens que, dans l'univers de la DeFi, les utilisateurs d'un protocole peuvent également en être les « actionnaires », par le biais des jetons de gouvernance qu'ils reçoivent pour chaque utilisation du service (et qu'ils peuvent également acquérir sur le marché) : faire la transparence sur le fonctionnement du service leur permet donc aussi, le cas échéant, de participer à une prise de décision plus éclairée.

1.3.3. Assurer la transparence quant au mode de gouvernance du *smart contract*

Pour l'utilisateur ou le client d'un protocole, il est important de pouvoir comprendre comment s'organise la gouvernance. De ce point de vue, **la publication d'informations sur les modalités techniques de la prise de décision devrait certainement constituer une obligation pour chaque protocole**, et devrait donc figurer dans un standard de certification (par exemple : existe-t-il un quorum pour voter ?). En outre, la production de **contenus pédagogiques**, afin de favoriser la compréhension sur la manière dont les décisions sont prises, devrait être encouragée, même si elle ne rentre pas dans le périmètre de la certification du *smart contract*.

Dans le cas où les décisions se prennent par des **votes communautaires**, **il faut cependant bien distinguer les informations concernant les procédures formelles du vote de celles concernant le poids de chaque détenteur de jetons dans les votes**. Sur ce dernier point, la transparence peut difficilement être atteinte sur des *blockchains* régies par le pseudonymat : dès lors qu'une même personne peut détenir plusieurs adresses sur la *blockchain*, sans que le lien puisse nécessairement être fait entre ces adresses, **il est illusoire de prétendre faire la transparence sur la répartition des jetons de gouvernance, et donc la pondération des votes**[31]. D'éventuelles avancées en la matière pourraient provenir du développement de **solutions d'identité numérique**, susceptibles de concilier les exigences d'identification (recueil et vérification) et de respect de la vie privée[32].

ASSURER LA TRANSPARENCE QUANT AU MODE DE GOUVERNANCE DU *SMART CONTRACT*

Proposition 8 : les processus de prise et de mise en œuvre des décisions concernant le *smart contract* (ou le protocole) sont décrits clairement et avec précision, mis à jour sans délai et publiés.

[30] Certaines spécificités des DAO, par exemple dans la gestion des droits de vote ou la gestion des conflits, pourraient néanmoins faire l'objet de recommandations dédiées dans un standard.

[31] Le développement des « votes délégués » sur un certain nombre de protocoles DeFi complexifie encore l'identification du poids de chaque acteur dans le vote.

[32] Sur ce point, voir la section 2.-4. de la Synthèse de la consultation publique sur le document de réflexion de l'ACPR.

1.3.4. Assurer aux clients que le service rendu ne sera pas modifié sans préavis

Dans une relation contractuelle classique, les conditions générales d'utilisation d'un service ne peuvent pas être modifiées du jour au lendemain : les clients doivent être prévenus à l'avance. La même règle peut s'appliquer aux services « financiers » fournis sous forme décentralisée ; sans préjudice de la possibilité pour un protocole de compléter son offre en ajoutant de nouveaux services additionnels à un service existant. Cela se traduit par un **certain nombre de principes**, qui pourraient être déclinés opérationnellement dans un standard de certification :

- Toute modification du fonctionnement d'un protocole devrait faire l'objet d'une **communication claire** à toutes les parties prenantes.
- Plus encore, les utilisateurs devraient être **informés de la manière dont un protocole peut être modifié, et des risques associés** – avec, éventuellement, un certain degré de standardisation des informations affichées (ainsi, un protocole administré de manière centralisée pourrait-il devoir afficher une plaquette harmonisée au sujet des risques qu'une telle situation présente pour les différentes parties prenantes).
- En outre, avant toute mise en œuvre d'un changement dans le fonctionnement, un **délai suffisant devrait être prévu**, permettant aux utilisateurs ou aux clients qui sont en désaccord avec les orientations prises de récupérer leurs fonds[33].
- Dans le cas du déploiement d'une nouvelle version d'un *smart contract* ou d'un protocole, les utilisateurs ou clients devraient être **clairement informés des conséquences qui en découlent** (notamment par la publication d'une documentation suffisamment claire), et ne devraient pas faire l'objet d'une migration contrainte vers la nouvelle version (comme dans une relation contractuelle classique).
- Enfin, pour assurer un bon niveau d'information quant aux risques encourus, les utilisateurs d'un *smart contract* préalablement certifié devraient être avertis des **conséquences qu'un changement entraîneraient sur le statut de certification** (le certificat du *smart contract* doit-il être révoqué et une nouvelle certification doit-elle être obtenue ? Pourquoi ? Dans quel délai prévisionnel ? etc.).

ASSURER AUX CLIENTS QUE LE SERVICE RENDU NE SERA PAS MODIFIE SANS PREAVIS

Proposition 9 : les dispositifs de gouvernance prévoient des règles préservant les intérêts des clients et utilisateurs en cas de modification du *smart contract* (information, préavis, possibilité de sortie, etc.).

1.3.5. Pouvoir réagir rapidement aux risques

Cet objectif va au-delà de la seule question de la sécurité informatique (traitée dans la section 1.2), pour recouvrir l'ensemble des risques opérationnels de la finance : risques de marché, de contrepartie, de liquidité etc. En effet, en cas d'attaque directe sur un *smart contract*, ou sur une composante avec laquelle il interagit et à laquelle il est vulnérable, **des dispositifs doivent permettre aux organes de gouvernance de réagir rapidement**. On peut ici évoquer des mécanismes dérogatoires aux règles habituelles de gouvernance des *smart contracts*, le cas échéant via des dispositifs centralisés[34]. Ces dispositifs peuvent néanmoins apparaître en contradiction avec l'objectif de stabilité abordé précédemment. En réalité, ces deux objectifs peuvent se combiner, dès lors que les mécanismes

[33] Des garde-fous permettant d'annuler une mise à jour problématique pourraient également être mis en place.
[34] Par exemple : clés d'administrateur, plan de continuité d'activité.

d'urgence font l'objet d'une communication en amont aux utilisateurs, et que des garde-fous appropriés existent pour définir précisément les situations requérant leur activation. **Un standard pour la certification des *smart contracts* devrait donc tout à la fois préciser les contours de ces dispositifs d'urgence et des garde-fous à mettre en place.**

Certains membres du GT ont par ailleurs souligné l'intérêt de prévoir des leviers d'action en cas de défaillance du *smart contract*, notamment une fonction de mise en pause du *smart contract*. Celle-ci permettrait notamment de corriger d'éventuels bogues. Le développement d'une telle fonction impliquerait néanmoins de prendre en considération les dépendances existantes entre les *smart contracts*, afin d'éviter que le dispositif ne mette également en pause, par inadvertance, d'autres applications.

POUVOIR REAGIR RAPIDEMENT AUX RISQUES

Proposition 10 : les dispositifs de gouvernance comportent des mécanismes d'urgence permettant de réagir rapidement à une attaque ou à une vulnérabilité. Des garde-fous à ces mesures exceptionnelles sont prévus (cas d'activation, actions possibles), afin de limiter les risques de prise de contrôle par les entités chargées par les mécanismes d'urgence de protéger temporairement le protocole ou le *smart contract*.

1.4. Conformité du service

Même lorsque le code informatique d'un *smart contract* est visible par tous – ce qui n'est pas systématique –, les utilisateurs non experts sont généralement dans l'incapacité d'en déduire le fonctionnement réel du programme. De même, les utilisateurs ne sont généralement pas capables d'évaluer l'adéquation entre les documents décrivant le fonctionnement d'un *smart contract* et la réalité. L'un des objectifs prioritaires d'une démarche de certification consiste donc à **vérifier que le *smart contract* rend bien le service attendu, et que ce service est rendu efficacement** (s'agissant par exemple des frais de gaz sur la *blockchain* : un *smart contract* inefficace, donc coûteux, peut faire perdre tout intérêt économique à une transaction). En outre, si la DeFi devenait davantage encadrée à l'avenir, les *smart contracts* pourraient **directement embarquer dans leur code un certain nombre d'obligations réglementaires**[35], en complément des bonnes pratiques qui pourraient déjà être mises en œuvre sur base volontaire[36] ; la certification pourrait alors vérifier le respect de ces exigences.

1.4.1. Vérifier que le *smart contract* rend le service attendu

Ici, plusieurs exigences pourraient être formalisées dans un standard de certification. D'abord, **l'exigence d'une documentation** accompagnant un *smart contract* : celle-ci doit au minimum décrire les services rendus, les responsabilités des parties prenantes, les risques liés à l'utilisation, ainsi que les mécanismes de gouvernance (*cf.* point précédent)[37]. En outre, **le fonctionnement réel du *smart***

[35] Rafael Auer (2022), *Embedded Supervision: How to Build Regulation into Decentralised Finance*, CESifo Working Paper Series 9771.

[36] Par exemple le *white-listing* des adresses des détenteurs autorisés.

[37] Dans l'industrie du web3 des documents d'information (*white papers*) sont couramment employés pour décrire un produit ou un service, avec toutefois un degré d'exigence variable. Le règlement MiCA (portant sur la fourniture de services dans un cadre centralisé) fixe ainsi des obligations précises de publication des informations relatives à l'émission de jetons et à la responsabilité des acteurs. Dans cette perspective, un participant du GT estime qu'il serait techniquement envisageable d'exiger la publication d'informations dans une section

***contract* doit correspondre avec cette documentation**. C'est le respect de ces deux points qui doit être vérifié dans la démarche de certification.

VERIFIER QUE LE SMART CONTRACT REND LE SERVICE ATTENDU

Proposition 11 : un *smart contract* est accompagné d'une documentation décrivant avec exactitude les services rendus, le niveau de qualité attendu, les risques liés à l'utilisation, les responsabilités des différentes parties prenantes, ainsi que les mécanismes de gouvernance.

1.4.2. Vérifier que le service est rendu efficacement

Un standard de certification devrait définir la notion d' « efficacité » du *smart contract*. En particulier, l'utilisation des ressources informatiques du registre partagé induit des frais (gaz), même si les mécanismes précis peuvent différer d'une *blockchain* à l'autre. Un *smart contract* efficace est donc celui qui, pour une action donnée, entraîne pour l'utilisateur un **montant raisonnable de frais**, au regard des ressources de l'utilisateur mais aussi afin de préserver l'intérêt économique de la transaction. Ce montant raisonnable serait à définir dans le standard ; il pourrait par exemple s'agir d'une déviation maximale par rapport à la moyenne constatée pour rendre un service donné. L'efficacité d'un *smart contract* tenant notamment à l'architecture de son code informatique, des recommandations visant à **optimiser le code** pourraient être intégrées dans un standard de certification. D'autres éléments de performance – par exemple : délai ou niveau de service rendu –, pourraient également être pris en compte[38].

VERIFIER QUE LE SERVICE EST RENDU EFFICACEMENT

Proposition 12 : un *smart contract* génère pour l'utilisateur un montant raisonnable de frais relativement au service rendu. À cet effet, l'architecture du code est optimisée, en tenant compte des exigences découlant des objectifs de sécurité.

1.4.3. Vérifier le respect de la conformité réglementaire

À ce jour, la loi n'impose pas d'intégrer des composantes réglementaires dans le code ou l'opération d'un *smart contract*. Toutefois les risques liés aux *smart contracts* font l'objet de réflexions depuis plusieurs années dans les enceintes réglementaires (voir l'encadré 3 ci-après). Dans ce contexte, les innovations liées à l'identité numérique sur *blockchain* (voir également la section 1.3. sur la gouvernance) pourraient favoriser une meilleure connaissance au regard des acteurs interagissant dans cet environnement et renforcer des exigences réglementaires dans le domaine[39]. D'autres exemples peuvent être imaginés, par exemple l'interdiction d'utiliser un crypto-actif donné dans un type de *smart contract*[40].

---

commentaire au début du code du smart contract, avec un lien permettant de diriger vers la documentation complète.

[38] Dans l'industrie informatique traditionnelle, ces performances font généralement l'objet d'une formalisation contractuelle (*Service Level Agreement*, SLA).

[39] Au-delà des considérations de lutte contre le blanchiment de capitaux et le financement du terrorisme, une connaissance plus précise des acteurs interagissant avec les *smart contracts* pourrait également introduire une notion de profil d'utilisateur, ce qui permettrait de modérer l'accès ou l'utilisation à certains services plus spécialisés de la DeFi, concourant ainsi à renforcer la protection des clients.

[40] Le document de réflexion de l'ACPR proposait par exemple d'interdire aux *smart contracts* d'utiliser comme actifs de règlement des jetons qui ne seraient pas des stablecoins au sens du règlement européen MiCA.

S'agissant de la faisabilité technique de l'intégration d'éléments de conformité réglementaire, il ressort des débats au sein du GT que **la couche des *smart contracts* constitue, d'un point de vue technique, la meilleure manière d'imposer une obligation absolue**. L'autre possibilité principale consiste à agir au niveau de l'interface avec l'utilisateur. Or, d'une part, il est impossible d'empêcher la création d'interfaces dans des pays non coopératifs, interfaces qui peuvent en retour donner accès aux protocoles DeFi à des individus ou à des entités sous sanction. D'autre part, il est toujours possible pour des utilisateurs motivés et chevronnés d'interagir directement avec un *smart contract* sans passer par la couche d'interface. Au contraire, les participants du GT indiquent que des **logiques d'exclusion au niveau des *smart contracts* sont techniquement réalisables**, par exemple en stockant les adresses interdites dans l'état d'un *smart contract* tiers (qui serait interrogé par le premier), ou encore en s'appuyant, dans le futur, sur la fourniture d'une identité numérique. **L'existence de ces mécanismes d'exclusion pourrait donc être vérifiée lors d'une démarche de certification**.

Des participants du GT indiquent toutefois que de tels mécanismes d'interdiction sont contraires au principe des *blockchains* sans permission, et risquent donc d'être mal accueillis par l'écosystème DeFi. D'autres participants estiment à l'inverse que cela pourrait conférer une forme de choix aux utilisateurs, entre ceux qui privilégient les services sans permission en acceptant le risque qu'ils comportent, et ceux qui préfèrent utiliser des services intégrant des briques de conformité réglementaire.

Encadré 3 : Les recommandations du GAFI et les orientations de l'ABE

Depuis plusieurs années, le Groupe d'action financière (GAFI), organisme mondial de surveillance du blanchiment de capitaux et du financement du terrorisme (BC-FT), mentionne l'interaction avec les smart contrats comme un facteur de risque à prendre en compte pour identifier, évaluer et déterminer comment atténuer au mieux les risques de blanchiment de capitaux et de financement du terrorisme associés à la fourniture de services sur actifs numériques.

De même, l'Autorité bancaire européenne (ABE) considère que l'interaction avec des plateformes d'échange décentralisées fait courir un risque accru de BC-FT en raison de l'absence de cadre juridique applicable à ces plateformes et de l'absence d'identification et de vérification d'identité des utilisateurs.

En effet, les *smart contracts* peuvent être accessibles depuis des adresses auto-hébergées ou des couches applicatives contrôlées par des personnes situées dans des pays à risque.

Dans ce contexte, l'intégration de normes de conformité au sein même de certains *smart contracts* pourrait faciliter leur utilisation par les entités assujetties aux règles de lutte contre le blanchiment de capitaux et le financement du terrorisme (LCB-FT), qu'il s'agisse de prestataires de services en crypto-actifs (PSCA au sens du règlement MiCA) ou d'autres institutions financières. En effet, à l'heure actuelle, ces derniers sont souvent dans l'impossibilité d'interagir avec des protocoles de DeFi du fait de leurs obligations de LCB-FT et en matière de gel des avoirs.

De telles normes pourraient inclure, par exemple, la mise à jour régulière des adresses avec lesquelles un *smart contract* n'est pas autorisé à interagir, ou encore l'intégration d'éléments de connaissance clientèle de type ZKP (certaines fonctionnalités de *smart contracts* pouvant être réservées à des utilisateurs soumis à une vérification d'identité renforcée).

***

2. Gestion de la mutabilité des *smart contracts*

Une fois déployés, les *smart contracts* sont en principe immuables dans la *blockchain* (seul l'état du *smart contract* peut être modifié). **En pratique, toutefois, il est possible de faire évoluer leur comportement ou de les remplacer par d'autres**. Ceci peut passer par des **mécanismes d'appel** (recours à des *proxies*), ou en utilisant des ***smart contracts* paramétrables** (les paramètres permettant de modifier l'état du *smart contract*, le code restant formellement inchangé). **La mutabilité des *smart contracts* constitue évidemment un enjeu au regard de tout processus de certification**, qui consiste à valider un type de fonctionnement à un moment donné. Il faut toutefois noter que ce problème est loin d'être spécifique à la DeFi : toute base de code auditée peut, d'une part, contenir des paramètres de configuration et tend, d'autre part, à faire l'objet de mises à jour régulières, risquant de rendre une éventuelle certification obsolète.

L'enjeu de la mutabilité est fortement lié à la question de la gouvernance. Ainsi, la section 1.2 contient déjà un certain nombre de recommandations quant à la manière de mettre en œuvre les changements, en particulier dans les garanties données aux utilisateurs. Cette section aborde les autres questions liées à la mutabilité.

2.1. Le cas des protocoles dits « immuables »

Certains protocoles sont dits « immuables »[41]. Cette assertion doit cependant être relativisée : en réalité, rares sont les *smart contracts* dont le comportement est impossible à modifier. **Les protocoles immuables** (incluant leurs *smart contracts* sous-jacents) **sont en fait des protocoles dans lesquels la gouvernance ne peut mettre en œuvre que des changements limités** (principe de « minimisation de la gouvernance »), par exemple la modification de certains paramètres, éventuellement à l'intérieur de fourchettes prédéfinies. Dans ce cadre, des changements fondamentaux de code nécessitent de passer à une autre version du protocole, c'est-à-dire formellement à un autre ensemble de *smart contracts*. **Mais même au sein d'un périmètre limité, ces protocoles dits « immuables » peuvent donc en réalité évoluer** ; de ce fait, ils ne constituent pas un cas radicalement différent pour l'analyse de la mutabilité des *smart contracts*.

2.2. La mutabilité des *smart contracts* en pratique

Pour certains protocoles, toute évolution significative du code nécessite de passer à une nouvelle version du ou des *smart contracts* concernés. De nouveaux *smart contracts* sont alors déployés, que les utilisateurs ont l'obligation ou le choix de rejoindre, selon les cas (*cf. supra*, sur la gouvernance, pour les principes formulés en la matière).

Au contraire, **certaines solutions permettent de faire évoluer un *smart contract* sans avoir à le remplacer** : de ce fait, il est possible de transformer certaines caractéristiques du programme sans rien modifier de la manière dont les utilisateurs accèdent au service, ce qui peut constituer un avantage important. De multiples raisons peuvent présider à ces modifications, par exemple : changer la valeur d'un paramètre (le niveau des frais par exemple), créer de nouveaux produits (généralement dans le cadre d'une modification à l'échelle d'un protocole), effectuer des mises à jour de sécurité, ou encore améliorer l'efficacité d'un service.

[41] Par exemple : Uniswap, Liquity.

Dans ce cas précis les concepteurs de *smart contracts* ont recours à **deux grandes modalités** pour effectuer de telles modifications :

i. Ils peuvent tout d'abord recourir à des ***smart contracts* paramétrables**, c'est-à-dire que certains mécanismes du fonctionnement global (et pas seulement la valeur de certains paramètres, comme le niveau des frais) sont définis au sein d'un composant externe, qui peut être aisément modifié.

ii. Ils peuvent alternativement recourir à une **architecture imbriquée** (ou « en cascade »), où un premier *smart contract*, dit « d'appel » (*proxy*), pointe vers un autre *smart contract,* dit « d'exécution », qui contient réellement les instructions (le nombre de couches imbriquées peut en pratique être plus élevé). Il suffit alors de modifier l'adresse vers laquelle le *smart contract* appelant pointe, vers un nouveau *smart contract* d'exécution, pour en modifier une partie des fonctionnalités.

Dans un cas comme dans l'autre, le risque est de voir un individu malveillant – ou simplement non mandaté – disposant de privilèges effectuer une manœuvre permettant de **détourner le fonctionnement d'un *smart contract*** : soit en modifiant la composante appelée via le paramètre ; soit en remplaçant le *smart contract* appelé par un *smart contract* malhonnête : l'architecture emboîtée rend alors la manœuvre plus difficile à détecter pour les différentes parties prenantes[42].

Par ailleurs, au-delà des accès à privilège, les mécanismes de déploiement d'un *smart contract* (premier déploiement ou déploiements subséquents) peuvent également receler des risques, qu'il conviendrait de couvrir par des exigences de sécurité *ad hoc*.

2.3. Tout changement apporté à un *smart contract* doit-il rendre la certification caduque ?

Certains participants au GT avancent une solution pour garantir la confiance dans le processus de certification : **tout changement de code apporté à un *smart contract* – ou à un composant appelé par celui-ci – devrait rendre caduque la certification précédemment obtenue**. Cette proposition présente l'avantage de la simplicité ; tandis que le document de réflexion ACPR avançait plutôt l'idée que seul un changement « substantiel » de code devait entraîner le besoin de re-certifier un *smart contract* – avec la difficulté de définir ce que recouvrirait concrètement cette notion[43]. En outre, si la certification intégrait à l'avenir des dimensions de conformité réglementaire (*cf. supra*), même une légère modification pourrait remettre en cause la validité du dispositif dans un domaine sensible. Enfin, un participant note qu'une modification mineure ne demanderait pas à un certificateur de réexaminer tout le dispositif ; la nouvelle certification pourrait donc être obtenue rapidement et pour un coût réduit.

---

[42] Il est à noter que ce risque serait minimisé en intégrant aussi la phase de déploiement dans le cycle de vie sécurisé du smart contract (*cf. supra*), et ce pour un smart contract proxy comme pour un smart contract paramétrable.

[43] Pour un participant, des changements « non substantiels » (par exemple : optimisation ou reformulation de code, correction de bogues mineurs sans risque présumé, ajout de fonctionnalités très circonscrites) peuvent parfois être définis avant le déploiement ; toutefois, cela ne permet pas nécessairement de définir les changements « substantiels ». Un autre participant propose de définir un seuil de « substantialité » en combinant des métriques quantitatives (comme un pourcentage du code modifié) avec une analyse d'impact (criticité des modifications, dont l'énumération pourrait prendre en compte : changement de la logique métier ; correction d'une vulnérabilité critique ; évolution de l'interaction avec les utilisateurs ou d'autres contracts ; modification dans les dépendances avec d'autres composants tiers ou librairies).

D'autres participants proposent plutôt de tenir compte de la nature des fonctionnalités modifiées au sein du *smart contract*[44]. Selon ce qui est modifié, en effet, l'impact sur le service peut être assez différent. **Il s'agirait alors de déterminer quels sont les domaines où les modifications sont les plus critiques**[45]**.** Un participant donne l'exemple d'une modification concernant la valeur des jetons : ici, le moindre changement devrait obliger à une nouvelle certification. Un autre participant indique que ce type de mécanisme est utilisé dans la gestion des modifications informatiques du domaine ferroviaire : le degré de criticité de chaque fonction est défini *ex ante*, afin de déterminer la réaction appropriée en cas de changement. Un participant indique qu'il serait en tout cas possible, d'un point de vue technique, de lier l'identification des fonctions modifiées et la gestion automatique de la certification : lorsque l'une des fonctions sur laquelle la certification a été attribuée est modifiée, celle-ci est automatiquement retirée (*cf. infra* la discussion sur le certificat technique obtenu à l'issue du processus). D'autres participants au GT pointent cependant un problème similaire à celui de la proposition du document de réflexion ACPR : dans le domaine financier, une définition *ex ante* de l'ensemble des niveaux de risque pourrait s'avérer complexe à réaliser en pratique.

RENOUVELLEMENT DE LA CERTIFICATION

Proposition 13 : la certification est obtenue pour un état donné du *smart contract*, des connaissances informatiques, ainsi que de son environnement, et pour une durée limitée. À l'issue de cette période, elle devrait être renouvelée.

Proposition 14 : en cas de modification du *smart contract*, la certification devrait être renouvelée. Des dispositifs de proportionnalité, définis *ex ante* en fonction de la nature des changements opérés, peuvent cependant être envisagés.

2.4. Comment formaliser techniquement le certificat obtenu à l'issue du processus de certification ?

Les participants au GT s'accordent sur la nécessité que la certification obtenue soit facilement visible pour les utilisateurs d'un *smart contract*. Plusieurs solutions techniques apparaissent réalisables pour parvenir à cet objectif ; elles peuvent aussi comporter des risques[46].

i. **Le recours à des *smart contracts*** : les participants conviennent qu'il ne faut pas que le *smart contract* sous revue porte son propre certificat, au risque de déboucher sur de possibles formes d'auto-certification. En revanche, le certificat peut être **référencé** par un second *smart contract* : il suffit alors d'interroger ce dernier pour s'assurer de son existence. Ce dispositif peut en outre facilement intégrer une dimension temporelle (pour fixer une durée maximale de validité du certificat), mais aussi se combiner avec une règle de mutabilité, car le certificat peut être automatiquement retiré si le code du *smart contract* sous revue a été modifié.

ii. **Le recours à des NFT**, représentant le certificat technique, dont le *smart contract* sous revue serait alors propriétaire. Cette solution aurait des avantages techniques similaires à la solution

[44] À l'échelle d'un protocole, tout changement implique généralement de modifier plusieurs *smart contracts* de concert.

[45] Pour autant, il convient de garder à l'esprit que les différentes fonctions d'un smart contract ne sont généralement pas isolées les unes des autres : une fonction critique pourrait ainsi être impactée par un changement intervenu sur une fonction non critique qui n'aurait pas fait l'objet d'une vérification approfondie (à travers des variables d'état partagées, ou via un déni de service affectant l'ensemble du smart contract, par exemple).

[46] Un participant évoque ainsi les vulnérabilités potentielles d'un registre, le danger de duplication et d'usurpation par des entités malveillantes, ou encore les risques liés à la superposition des certificats.

précédente et pourrait bénéficier de l'émission par une autorité reconnue agissant en tant que tiers de confiance.

iii. **Le recours à un registre**, disponible sur la *blockchain* et maintenu par l'organisme émettant les certifications : ce registre fournirait la preuve qu'un *smart contract* donné a bien été certifié. Cette solution est similaire à la i. à la différence qu'elle s'appuie spécifiquement sur un organisme de confiance.

Quelle que soit la solution technique retenue pour « porter » techniquement le certificat, les participants s'accordent sur **la nécessité de mobiliser également la couche des interfaces** (applications de bureau, applications mobiles, wallets…) **pour propager l'information et la rendre facilement accessible aux utilisateurs**. Ainsi, de même que les navigateurs internet vérifient en permanence tous les certificats des sites (le certificat SSL du protocole HTTPS par exemple), les interfaces pourraient afficher la présence ou l'absence d'un certificat. Un participant indique qu'un fournisseur de wallet[47] a déjà introduit ce type de fonctionnalité, qui commence à se diffuser sur ce type d'interface.

Le code vérifié du *smart contract* devrait aussi être publié et authentifié à l'issue de la phase de déploiement et pourrait dès lors accompagner le certificat technique.

FORME TECHNIQUE DU CERTIFICAT OBTENU À L'ISSUE DU PROCESSUS DE CERTIFICATION

Proposition 15 : un certificat est obtenu à l'issue du processus de certification : il n'est pas falsifiable, disparaît automatiquement quand la certification n'est plus valide et est facilement consultable et compréhensible par les utilisateurs ; à cet effet, il est notamment accessible par la couche des interfaces utilisateurs.

***

3. Qui peut et doit établir des standards pour la certification ?

Deux grands modèles prévalent historiquement pour l'établissement de standards : une initiative des acteurs de marché eux-mêmes, ou une démarche initiée par les autorités publiques.

Dans le premier cas de figure, qui est le plus courant d'après un participant, les standards adoptés tendent à être proches des réalités de marché, ce qui favorise leur acceptabilité par l'industrie et facilite leur mise en œuvre. Les autorités publiques peuvent également jouer un rôle, en encourageant l'adoption des standards. Cela suppose toutefois que les acteurs privés parviennent à s'accorder sur des normes communes. En outre, dans le cas de standards destinés en premier lieu à garantir la protection de la clientèle, cela suppose également que le niveau d'exigence fixé dans les standards soit suffisamment élevé pour atteindre cet objectif.

Dans un second schéma, les autorités publiques fixent elles-mêmes les standards. Ceci a normalement l'avantage d'assurer que les standards choisis répondent à des objectifs d'intérêt général, définis par la règlementation en vigueur, et permet de trancher les désaccords entre acteurs ou segments du marché. En pratique, les standards fixés par les autorités publiques font généralement l'objet de concertations avec les acteurs de marché, afin d'assurer leur viabilité.

[47] Par exemple *Rabby Wallet*

Un participant estime par ailleurs nécessaire d'intégrer à la réflexion des considérations sur l'industrie émettrice des certificats : par exemple son degré de concentration. Un autre participant souligne la nécessité que l'émission du certificat soit effectuée par un acteur tiers indépendant, séparé du *smart contract* ou de ses producteurs.

PARTIE 2 : AUDIT

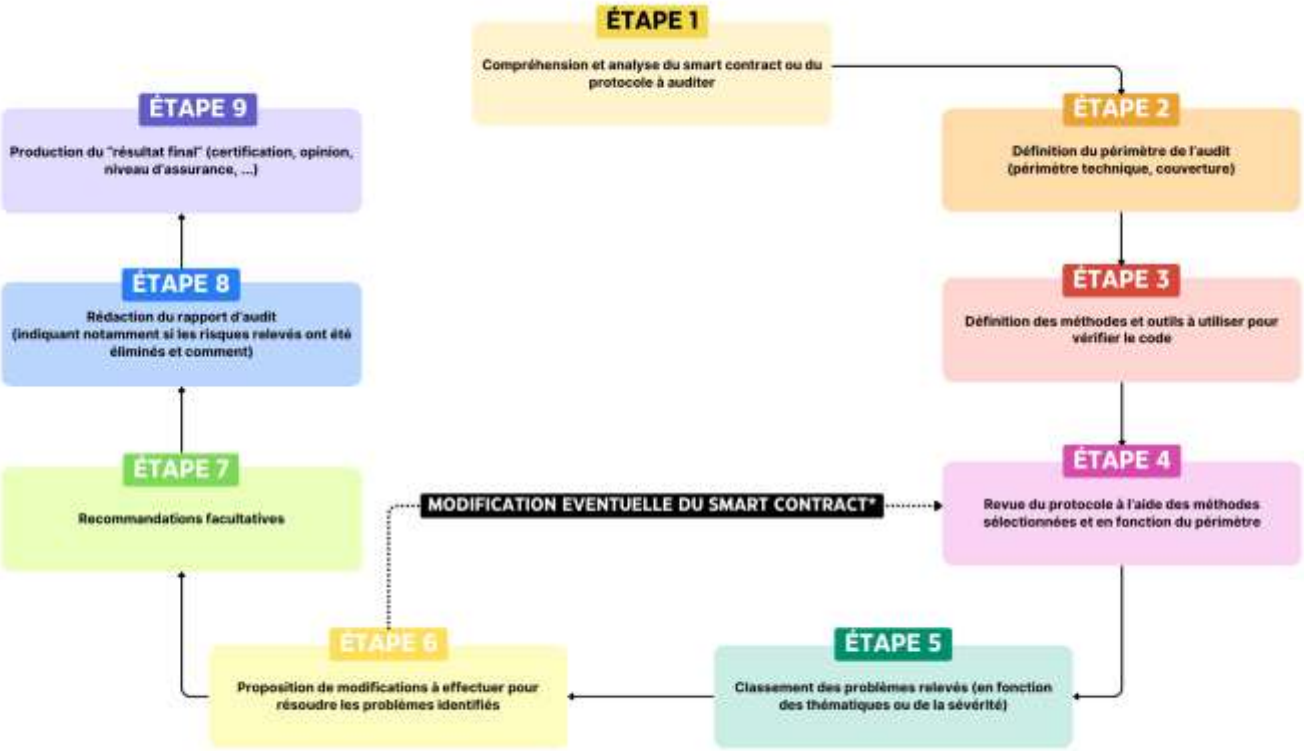
Le terme d'audit s'entend ici comme un audit de certification : il s'agit d'examiner si un *smart contract* respecte les exigences définies dans un standard (*cf.* la première partie de ce document, consacrée aux standards). Les travaux du GT ont toutefois montré que cette démarche peut et doit s'inspirer des techniques d'audit existantes, en général et dans le domaine spécialisé des *smart contracts*.

1. Comment auditer un *smart contract* ?

1.1. La démarche générale d'audit

1.1.1. L'audit de *smart contract* tel qu'il se pratique aujourd'hui

Tel qu'il est actuellement pratiqué, l'audit des *smart contracts* est **proche de l'audit informatique classique**. De manière schématique, la démarche d'audit peut être décomposée en **9 étapes** (*cf.* schéma ci-dessous).



** la démarche s'applique au smart contract ou aux smart contracts qui constituent le protocole DeFi audité*

Étape 1 : L'auditeur réalise une première analyse du *smart contract* ou du protocole à auditer, sur la base du mandat reçu. Cette phase consiste en un premier examen du code informatique du *smart contract*, permettant à l'auditeur d'accéder à une compréhension de premier niveau du fonctionnement du service. Elle permet aussi de lister les dépendances à des éléments externes (bibliothèques *open source*, appel à d'autres *smart contracts* etc.). Cette phase permet plus généralement à l'auditeur de comprendre le contexte de l'audit (organisation, enjeux, objectifs).

Étape 2 : L'auditeur définit le périmètre technique à auditer, ainsi que le niveau de couverture, c'est-à-dire les types de risques qui seront examinés et le niveau d'assurance recherché lors de l'audit. Il définit également les points de contrôle de son audit.

Le **périmètre technique** comprend au minimum le code informatique du *smart contract*, mais aussi le code de déploiement sur une *blockchain* donnée (code permettant la mise en production du *smart contract*). L'audit peut également porter sur les vulnérabilités pouvant découler de l'interaction avec d'autres systèmes (*smart contracts* appelés, oracles, *blockchain*, *layers 2*, interfaces d'accès pour les utilisateurs).

S'agissant de la **couverture**, l'audit comprend toujours un examen de la sécurité du code informatique, c'est-à-dire la détection des dysfonctionnements involontaires ainsi que des vulnérabilités à des utilisations abusives et à des attaques informatiques. L'audit peut en outre comprendre un examen des risques financiers liés aux services proposés (erreurs dans les calculs arithmétiques régissant le fonctionnement du service, mécanismes pouvant favoriser la manipulation de marché, risques de liquidité, de contrepartie etc.), un examen de la conformité de la documentation et des règles de communication avec les utilisateurs, ainsi qu'un examen des risques liés à la gouvernance.

Étape 3 : L'auditeur définit les méthodes et les outils à utiliser pour vérifier le code (ce point est développé dans la section 1.2 ci-après). Il identifie également les compétences nécessaires à la réalisation de l'audit. À l'issue de cette étape, qui conclut la phase de pré-audit, l'auditeur a ainsi élaboré un programme d'audit définissant l'ensemble des points et objectifs de contrôle associés à chaque processus ou composant du périmètre à auditer. Il précise également les éventuelles exclusions de processus ou composants.

Étape 4 : L'auditeur effectue la revue du *smart contract* à l'aide des méthodes sélectionnées, et sur le périmètre qui a été défini.

Étape 5 : L'auditeur classe les vulnérabilités détectées[48], soit en en dressant une typologie thématique[49], soit en les classant en fonction de leur risque[50] afin, le cas échéant, de prioriser les travaux d'atténuation (*cf.* étape 7).

Étape 6 : L'auditeur propose des modifications pour résoudre les vulnérabilités et dysfonctionnements identifiés à l'étape 4, éventuellement en prenant en compte le degré de sévérité (*cf.* étape 5). Le cas échéant, des **modifications sont effectivement apportées** par l'audité[51]. **Le *smart contract* modifié fait alors l'objet d'un nouvel examen centré sur les éléments modifiés** (retour à une étape 4 simplifiée). Le cas échéant, plusieurs itérations entre les étapes 4 et 6 peuvent être nécessaires pour parvenir à un résultat satisfaisant.[52]

---

[48] Cette étape n'est pas réalisée par tous les auditeurs.

[49] À titre d'exemple, on pourra consulter la liste produite par l'auditeur Runtime Verification ou la *Common Weakness enumeration* de l'organisation MITRE.

[50] Pour mémoire, la mesure d'un risque est la combinaison d'un impact potentiel et d'une probabilité d'occurrence. L'auditeur Runtime Verification combine ainsi deux échelles pour évaluer le risque : l'impact potentiel de la faille, et la difficulté pour un attaquant d'exploiter celle-ci (mesurant la probabilité d'occurrence).

[51] Certains auditeurs proposent parfois de mettre en œuvre eux-mêmes les recommandations. Certains participants au GT font toutefois remarquer que, pour une meilleure gestion des risques et des responsabilités, il est nettement préférable de conserver une séparation étanche des rôles de l'auditeur et de l'audité.

[52] Si l'audit intègre un volet de sécurité, une bonne pratique est de formaliser un rapport intermédiaire, centré sur la sécurité, décrivant les vulnérabilités et les anomalies décelées, afin que l'audité puisse mettre en œuvre

Étape 7 : L'auditeur peut formuler des recommandations additionnelles, sur des éléments ne présentant pas de risque opérationnel (par exemple : optimisation du code, meilleur traitement des données)[53].

Étape 8 : L'auditeur rédige le rapport d'audit comprenant l'ensemble des constats (vulnérabilités et dysfonctionnements détectés), et contenant le cas échéant un scénario d'exploitation pour chacune de ces failles. Le rapport mentionne également le niveau de risque de chacun des constats et si les risques signalés sont assumés par les responsables du service ; il précise dans ce cas les justifications données. À l'inverse, si, pour réduire certains risques, des modifications ont été entreprises au cours du processus d'audit (*cf.* étape 6), le rapport indique si elles ont ensuite pu être vérifiées par l'auditeur. Le rapport contient enfin les éventuelles recommandations additionnelles (*cf.* étape 7).

Étape 9 : Lorsque des recommandations n'ont pas pu être mises en œuvre avant la fin de l'audit, l'auditeur peut être chargé d'effectuer un suivi de leur mise en œuvre par l'audité, éventuellement à travers un nouvel audit allégé. Dans le cas où les modifications demandées sont complexes, l'auditeur peut être missionné pour résoudre les vulnérabilités ou dysfonctionnements identifiés et de proposer une correction.

Encadré 1 : Les différents niveaux de code et leur audit

Le **code source** du *smart contract* est le code écrit par le développeur, dans un **langage informatique de haut niveau** (c'est-à-dire compréhensible par l'humain). L'un des langages informatiques les plus utilisés pour coder les *smart contracts* est le langage « Solidity », initialement développé pour les *smart contracts* de la *blockchain* Ethereum.

Une fois le code du *smart contract* écrit, il est **compilé**, c'est-à-dire traduit en un langage intermédiaire entre code source et instructions machine. Ce langage se nomme le **bytecode**. Cette compilation est effectuée par un **compilateur**, installé localement ou disponible sur un serveur de compilation. L'un des compilateurs les plus utilisés pour compiler du code écrit en Solidity est Solc.

Le *smart contract* écrit sous forme de bytecode est ensuite **transféré et enregistré sur la *blockchain*, c'est-à-dire déployé** ; une adresse lui est associée. Il est alors accessible par tous les utilisateurs de la *blockchain*. Lorsqu'un utilisateur interagit avec un *smart contract*, son bytecode est **interprété** ou compilé en **langage machine** pour être exécuté. Dans le cas d'Ethereum, cet environnement d'exécution se nomme l'Ethereum Virtual Machine (EVM).

Des **vulnérabilités peuvent apparaître à chaque étape** de la chaîne de développement et de déploiement. Toutefois, la plupart des audits de *smart contracts* se concentrent uniquement sur la première (revue du code source) et la dernière étape (revue de l'exécution du bytecode) de cette chaîne, même si certains auditeurs intègrent des éléments additionnels dans leurs vérifications.

Pour être exhaustif, **l'audit devrait inclure une vérification complète de chacune des étapes du développement au déploiement, c'est-à-dire :**

1. Revue du code source.
2. Examen de la compilation du code du *smart contract* sous revue en bytecode : en examinant les failles potentielles du logiciel de compilation lui-même (mises à jour, optimisation…) ainsi que le résultat de cette compilation (c'est-à-dire une revue du bytecode).

ces recommandations rapidement, qui seront ensuite revues par l'auditeur. Si l'audit n'intègre pas de volet de sécurité (un audit de conformité par exemple) alors un seul rapport définitif est rédigé (*cf.* étape 8).

[53] Il est intéressant de noter que les non-conformités mineures ne font pas obstacle à la délivrance d'une qualification ou d'une certification par l'ANSSI (*cf.* encadré 3). Toutefois, si celles-ci ne sont pas corrigées après un certain délai, elles peuvent être requalifiées en non-conformité majeures et conduire à retirer le visa.

3. Revue du processus de déploiement, c'est-à-dire de l'ensemble des opérations permettant le transfert du code informatique de l'espace de travail du développeur à la *blockchain*. Cela inclut notamment un examen du code de déploiement.
4. Vérification que le bytecode sous revue est bien identique au bytecode déployé sur la *blockchain*.
5. Revue de l'exécution du *smart contract* (c'est-à-dire de l'exécution du bytecode).

1.1.2. L'audit dans une démarche de certification

La démarche générale d'audit des *smart contracts* présentée dans la section précédente paraît pertinente dans le cadre d'un audit de certification. Pour correspondre à la démarche de certification tel que définie dans la partie « standards » de ce document, **l'audit devrait être réalisé dans sa version la plus étendue** : en termes de périmètre technique, en couvrant les interactions du *smart contract* avec les autres systèmes ; en termes de couverture, en ne se limitant pas à la seule sécurité informatique, mais en couvrant également les aspects de « conformité » et de gouvernance (*cf.* section 1.3 ci-dessous).

Pour transformer l'audit général décrit ci-dessus en un audit de certification, **le processus en 9 étapes décrit ci-dessus serait suivi d'une 10e et dernière étape, correspondant à la décision d'attribuer ou non la certification**.

1.2. Les techniques d'audit du code informatique d'un *smart contract*

Comme pour les autres objets logiciels, l'audit d'un *smart contract* repose d'abord sur la définition de **spécifications**. Une spécification est un ensemble de propriétés qui doivent être respectées par le code sous revue. Les spécifications peuvent être de deux ordres. Les **spécifications fonctionnelles** détaillent les fonctions attendues et fixent des exigences quant à la **conception** du logiciel. Les **spécifications non fonctionnelles** concernant la manière dont le logiciel opère (performance, sécurité, évolutivité etc.) et fixent des exigences relatives à **l'architecture** du système[54].

Dans le cadre d'un processus de certification, les spécifications correspondent à l'ensemble des exigences listées dans le standard de certification ; elles sont complémentaires aux spécifications usuelles, fonctionnelles et non-fonctionnelles, qui auront été écrites pour le développement du *smart contract*.

Plusieurs techniques peuvent être mobilisées pour vérifier le respect, par le *smart contract*, des spécifications définies dans le standard de certification.

1.2.1. Les méthodes d'analyse statique

Les **méthodes d'analyse statique** consistent en **l'inspection systématique des lignes de code** pour en identifier les vulnérabilités. **Cet examen** vise notamment à vérifier le respect des **bonnes pratiques d'écriture de code**, afin d'éviter de reproduire des erreurs déjà documentées. L'analyse statique prend

[54] En outre, les spécifications peuvent correspondre à des préconditions à l'exécution du code (éléments nécessaires pour que le code fonctionne), à des post-conditions à l'exécution du code (conditions à respecter par les résultats du code), à des invariants (conditions devant être vérifiées tout au long de l'exécution du code) et à des assertions (conditions devant être vérifiées à certains point spécifiques de l'exécution du code).

deux formes principales : (i) la **revue manuelle** de code par l'auditeur ; (ii) la **revue automatique**[55], qui permet des gains d'efficacité, mais qui peut conduire à perdre une partie de la qualité de la revue d'un auditeur humain spécialisé et expérimenté. En pratique, ces deux techniques sont souvent combinées.

L'analyse manuelle repose sur l'expertise de l'auditeur ; elle a l'avantage de ne pas nécessiter d'infrastructure technologique particulière (« *low tech* »). En revanche, elle peut se révéler longue à mener, et refléter les biais conscients ou inconscients de l'auditeur. **L'analyse automatique**, quant à elle, est plus rapide à conduire ; lorsqu'elle repose sur un outil recensant de manière exhaustive les attaques connues (et alimenté en continu), elle permet de couvrir un large spectre de vulnérabilités. En revanche, l'analyse automatique ne permet pas de détecter les failles encore inconnues, ni en général les failles les plus complexes. Plus généralement, les différentes techniques d'analyse statique **ne permettent pas toujours de déceler les dysfonctionnements et vulnérabilités apparaissant lors de l'exécution du code** (même si elles permettent d'anticiper certains d'entre eux).

Encadré 2 : L'analyse statique assistée par intelligence artificielle (IA)

Un nouveau type d'analyse statique du code connaît actuellement un développement rapide : l'examen par des outils d'**intelligence artificielle générative**, en l'occurrence des *large language models* (LLMs). Le fonctionnement est simple : il suffit de fournir à un outil d'IA le code informatique à analyser, et de demander dans l'invite (*prompt*) si celui-ci peut être considéré comme sûr.

Les outils grand public donnent actuellement des résultats d'une qualité variable, et **qui se dégradent rapidement lorsque la complexité du code augmente**. Les LLM spécialisés dans l'audit de *smart contracts* donnent quant à eux des résultats plus pertinents, mais là aussi leur qualité tend à diminuer lorsque le code devient plus complexe. Ces outils peuvent cependant permettre d'identifier des **zones d'intérêt** dans le code, c'est-à-dire des parties qu'il convient d'examiner plus en détail car elles pourraient être dysfonctionnelles ou propices aux attaques.

Si ces outils ne présentent pour l'heure qu'un intérêt limité pour l'audit de code, la situation pourrait évoluer dans les années à venir, la technologie sous-jacente étant en rapide progrès.

1.2.2. Les méthodes d'analyse dynamique

Les méthodes d'analyse dynamique consistent à observer et analyser le comportement « en action » du code, dans un environnement de test. L'analyse dynamique repose sur l'idée que les bogues ou vulnérabilités apparaîtront plus facilement ou plus rapidement en exécutant le code plutôt qu'en le lisant (analyse statique) – voire que certaines difficultés ne peuvent apparaître qu'en conditions réelles, dans un contexte d'exécution donné (par exemple des problèmes d'allocation de la mémoire). Même si elle peut être envisagée comme un substitut à l'analyse statique, l'analyse dynamique en est plus souvent un complément, constituant généralement en pratique une deuxième phase de l'audit du code.

L'analyse dynamique est en pratique conduite de façon automatisée. **Deux méthodes principales** sont en pratique mobilisées par les auditeurs. La méthode des **tests unitaires**, tout d'abord, consiste à « découper » le code en petites unités, qui sont ensuite exécutées pour s'assurer du bon fonctionnement de chacune[56]. Le découpage permet ainsi une analyse ordonnée des

[55] Pour la revue automatisée, on peut citer les outils *Slither* (développé par la société Trail of Bits en 2018) ou *OpenZeppelin Defender* (développé par la société OpenZeppelin en 2020).

[56] Deux outils fréquemment utilisés peuvent être cités : *Hardhat* (Nomic Foundation) et *Foundry*. Ils ne se limitent d'ailleurs pas aux tests, mais permettent aussi de mettre en œuvre certaines des autres techniques citées ici.

dysfonctionnements. Cette phase est généralement suivie par des tests d'intégration, permettant d'examiner le fonctionnement d'ensemble, une fois les unités assemblées. Des tests de validation permettent ensuite de vérifier la conformité aux spécifications : tests fonctionnels pour les spécifications fonctionnelles, tests de performance pour les spécifications non fonctionnelles. Enfin, des tests spécifiques peuvent être conduits, par exemple pour vérifier la résistance du code à une attaque.

Cette méthodologie de test a ainsi le grand intérêt de vérifier le **comportement réel du code**. En revanche, pour être efficace, elle nécessite une bonne couverture de tests (ce qui peut notamment se traduire par des coûts accrus). En outre, l'analyse dynamique de code nécessite un **environnement de test réaliste**, ce qui n'est pas toujours chose aisée puisque l'environnement doit dans le même temps être sécurisé (*cf.* section 1.4 sur cette question).

Le ***fuzzing*** (ou test à données aléatoires) constitue l'autre grande méthode d'analyse dynamique du code. Elle consiste à générer un **grand nombre d'entrées aléatoires**, et potentiellement malveillantes[57], pour tester de manière approfondie le comportement du code dans un grand nombre de scénarios[58]. L'application est bombardée par ces entrées potentiellement malveillantes, et l'outil en analyse les réponses. Peuvent ainsi être générés :

- des chaînes d'entrée longues, visant à tester la vulnérabilité du débordement de la mémoire tampon ;
- des nombres négatifs et des grands nombres positifs (pour détecter les vulnérabilités de soupassement et surpassement de nombres entiers) ;
- des requêtes de manipulation de données, afin d'identifier une éventuelle vulnérabilité à des injections de code;
- des entrées inattendues et apparemment illogiques dans le contexte d'exécution du code, afin de tester la robustesse des hypothèses retenues implicitement ou explicitement par les développeurs.

Le grand avantage du fuzzing est ainsi de permettre de **détecter des comportements inattendus**, et des portes d'entrée éventuelles à des attaques non encore inventées. Toutefois, il est **difficile de garantir que toutes les entrées pertinentes ont été testées**.

1.2.3. La vérification formelle

Par rapport aux analyses statiques ou dynamiques, la **vérification formelle** affiche un degré d'ambition plus élevé : elle entend **prouver, au sens logico-mathématique, le respect des spécifications** (par exemple, prouver qu'une variable reste toujours dans un intervalle donné). Cette méthode repose en effet sur l'analyse de la sémantique des programmes, c'est-à-dire la description mathématique formelle du sens d'un programme informatique. Son intérêt est de fournir un **niveau maximal d'assurance** quant à la validité d'un code informatique, de même que la démonstration d'un théorème permet d'obtenir une certitude sur la validité d'un énoncé mathématique.

La vérification formelle nécessite d'abord de **formaliser les spécifications** attendues dans un langage informatique dont la sémantique est bien définie (non ambiguë). Puis un outil automatique permet de générer les conditions logiques à respecter pour se conformer aux spécifications, dites aussi

[57] Celles-ci sont généralement issues d'un dictionnaire des vulnérabilités connues.

[58] On citera quatre outils de *fuzzing* : Echidna (développé par Trail of Bits en 2018), Medusa, Manticore et Foundry.

« obligations de preuve ». Enfin, un « prouveur » automatique[59] permet de vérifier mathématiquement que le code sous revue, dont une représentation purement logique et sémantique est élaborée – respecte les spécifications[60].

Il existe deux façons principales de conduire une vérification formelle :

- La **vérification de modèles** consiste en l'exploration exhaustive des états (configurations) possibles du code sous revue, afin de vérifier le respect en toutes circonstances des spécifications. La formalisation logico-mathématique permet de lister les états du système sous une forme symbolique – donc finie –, ce qui permet leur vérification exhaustive.
- La **vérification déductive** utilise des systèmes de preuves pour démontrer mathématiquement et logiquement que les spécifications formelles sont toujours respectées, à la manière d'une démonstration de théorème.

Outre le **degré d'assurance** qu'elles permettent d'obtenir sur le respect par le code informatique des exigences inscrites en spécification, le potentiel des méthodes formelles provient de leur caractère **automatisable**, qui permet en théorie un passage à l'échelle sans limitation.

Toutefois, il faut noter que les méthodes de vérification formelle sont crucialement dépendantes de l'écriture de **spécifications formelles** : si les spécifications ne sont pas écrites correctement, la preuve de leur respect n'a guère de valeur (« *garbage in, garbage out* »)[61]. L'écriture de spécifications est généralement accomplie manuellement par des **experts en méthodes formelles**, ce qui constitue une **compétence rare**. En outre, les temps de calcul nécessaires peuvent être importants. De ce fait, la **vérification formelle est généralement complexe et coûteuse** à mettre en place ; elle est donc fréquemment réservée aux applications les plus critiques[62]. Enfin, la vérification formelle nécessite que le code informatique sous revue soit écrit dans un **langage de programmation compatible**. En effet, certains langages sont plus ambigus et donc peu propices à la vérification formelle[63]. Face à cette difficulté, des systèmes de **traduction vers un code compatible** commencent à voir le jour[64], mais ce processus de traduction peut lui-même être source d'erreurs, et remettre en cause la validité de la vérification formelle[65].

Enfin, la vérification formelle **s'applique mal aux architectures imbriquées de *smart contracts***, où des proxies appellent d'autres *smart contracts* (*cf.* partie « standards »). En effet, les appels entre *smart contracts* tendent à générer une explosion combinatoire empêchant la vérification formelle de s'appliquer. En revanche, la vérification formelle fonctionne sans difficulté sur les architectures

---

[59] Par exemple : Z3, CVC4, Certora Prover.

[60] Si l'écriture des spécifications formelles, au départ de l'audit, nécessite l'intervention d'un spécialiste, la compréhension du résultat final de l'audit ne nécessite pas de connaissances particulières en méthodes formelles ; elle est donc accessible à tout le monde.

[61] Ce problème a été mis en lumière par une faille de sécurité sur la plateforme d'échange Dexter (sur la blockchain Tezos). Le succès de sa vérification formelle avait été annoncé mais les spécifications testées comportaient elles-mêmes des vulnérabilités.

[62] Les méthodes formelles ont jusqu'ici principalement été utilisées pour le développement d'algorithmes dans des domaines critiques, mettant en jeu la sécurité des personnes, comme les transports ou l'industrie nucléaire.

[63] Au-delà du cas des méthodes formelles, il semble, selon les participants du GT, que plus un langage de programmation est « expressif » (c'est-à-dire peu ambigu), plus un audit du code sera en capacité de fournir un degré élevé d'assurance quant à la sécurité du smart contract.

[64] Voir par exemple : Zeinab Nehai, François Bobot, *Deductive Proof of Ethereum Smart contracts Using Why3*, CEA DILS, 2019.

[65] Ces enjeux de « traduction » sont actuellement à l'origine d'importants travaux de R&D, selon des participants au GT, qui indiquent en outre que des générateurs de code pourraient à l'avenir permettre d'écrire du code formellement vérifié dans n'importe quel langage de programmation (et donc aussi sur n'importe quelle blockchain).

paramétrables, qui constituent l'autre manière de gérer la mutabilité dans l'univers des *smart contracts*.

Plus fondamentalement, et nonobstant les remarques énoncées ci-dessus, comparées aux autres méthodes d'analyse, **la vérification formelle** réalisée dans les règles de l'art **fournit une assurance plus élevée que le code ne comporte pas de défaut**, bien qu'elle ne puisse jamais être une assurance absolue.

1.3. Quelles méthodes pour quels objectifs ?

L'audit de certification d'un *smart contract* doit permettre de vérifier le respect d'un certain nombre d'exigences définies dans un standard. Comme cela a été établi (cf. partie « Standards » de ce document), ces exigences peuvent être décomposées en :

(i) **Des principes de sécurité** : bonnes pratiques de conception logicielle, atténuation des vulnérabilités découlant de l'environnement, principe du moindre privilège pour les autorisations d'accès, prise en compte du cycle de vie.

(ii) **Des principes de conformité** : le *smart contract* rend bien le service attendu, et il le rend efficacement ; en outre, le service rendu est conforme à la réglementation[66].

(iii) **Des principes de gouvernance** : description claire des processus de décision, préservation des intérêts des clients en cas de modification, et mécanismes d'urgence permettant de réagir rapidement à des attaques.

L'ensemble des méthodes d'audit décrites précédemment (analyse statique, analyse dynamique, vérification formelle) peuvent être mobilisées en vue d'examiner le respect des **principes de sécurité** d'un *smart contract*. En particulier, elles semblent toutes permettre d'analyser les vulnérabilités d'un *smart contract* à des failles déjà documentées, ou encore de vérifier le respect du principe du moindre privilège. En revanche, s'agissant des vulnérabilités découlant de **l'environnement d'exécution** du *smart contract*, les méthodes de vérification formelle n'apparaissent pas comme les plus indiquées, puisqu'elles s'appliquent mal aux problèmes d'interactions entre objets logiciels.

Les méthodes de vérification formelle apparaissent en revanche tout indiquées pour évaluer le respect des **principes de conformité** du code informatique, et en particulier vérifier que le programme accomplit bien les opérations qui sont attendues de lui. Les techniques d'analyse statique ou dynamiques peuvent également permettre d'évaluer cet objectif, même si elles sont d'abord conçues pour détecter des vulnérabilités. En revanche, les aspects liés à **l'efficacité** du *smart contract* font davantage appel à une évaluation subjective de l'auditeur, et apparaissent donc plus délicats à automatiser entièrement ; c'est d'autant plus vrai qu'un développeur doit parfois arbitrer entre sécurité et efficacité[67]. Il en va de même pour l'évaluation de la **conformité réglementaire** du service fourni.

En revanche, aucune des méthodes automatisées listées ci-dessus ne semble de nature à permettre la vérification du respect de **certains principes de gouvernance**, et en particulier tous les aspects qui ne sont pas fixés dans le code du *smart contract* : ainsi la documentation à l'usage des utilisateurs, les règles relatives aux dispositifs de *multisig*, ou encore les mécanismes d'urgence. La **revue manuelle** paraît la seule manière d'examiner ces éléments périphériques au code, dont l'évaluation comporte

[66] Par exemple des règles relatives à la LCB-FT (*cf.* section 1.4.3 de la partie « standards » de ce document).

[67] Toutefois, certaines pratiques connues pour être particulièrement coûteuses en gaz sans apporter pour autant d'avantage particulier pourraient être identifiées par des outils automatiques.

nécessairement une part de subjectivité. Il faut toutefois noter que **d'autres éléments relatifs à la gouvernance sont, eux, fixés dans le code du *smart contract*** et peuvent être audités par le biais de **méthodes automatiques**. Un point particulier est par ailleurs à noter, lorsque le *smart contract* sous revue est gouverné par une DAO[68] : la gestion de cette dernière repose en effet sur un *smart contract* dédié. L'examen des règles de gouvernance peut donc nécessiter d'auditer le code de ce second *smart contract*[69].

Enfin, il apparaît que les différentes méthodes présentent des coûts et des degrés de complexité différents, et permettent en retour de parvenir à **différents niveaux d'assurance** quant à la qualité du code sous revue. En pratique, toutefois, **un processus d'audit consiste généralement en une combinaison de plusieurs techniques** : ainsi, une revue manuelle de code pourra par exemple être suivie d'une phase de tests unitaires (puis de tests d'intégration etc.), avant éventuellement d'enclencher un examen par fuzzing.

1.4. Sur quel environnement tester et auditer les *smart contracts* ?

Les *smart contracts* ne sont généralement pas déployés directement sur la *blockchain* principale (*mainnet*). Au contraire, des **tests de premier niveau** sont généralement conduits sur une ***blockchain* privée locale**. Des **tests de deuxième niveau** sont ensuite conduits sur une ***blockchain* de test (*testnet*)** reproduisant au maximum le comportement de la *blockchain* de production. Les *testnets* sont des *blockchains* publiques, c'est-à-dire accessibles à tous, comme la *blockchain* principale, ce qui permet à la communauté des développeurs d'éprouver le protocole. En particulier, le test de certains éléments du protocole sous revue nécessite de pouvoir réaliser des transactions. Sur Ethereum, par exemple, les transactions sur les *testnets* sont simulées en recourant à des **ethers (ETH) à valeur réelle nulle (*faucet*)**, qui peuvent être obtenus sur demande. Ces transactions sont souvent **validées via la « preuve d'autorité »**, un mécanisme de consensus différent de celui du mainnet, reposant sur un réseau fermé de validateurs préalablement choisis[70].

Pour conduire à un test satisfaisant du *smart contract* ou du protocole sous revue, **le testnet doit assurer des conditions d'exécution les plus proches possibles du mainnet**[71]. Les participants du GT expriment sur ce sujet des **opinions divergentes**, certains estimant que les testnets ne peuvent, par construction, pas permettre d'auditer de façon réaliste toutes les fonctionnalités des *smart contracts* ni l'ensemble de leurs vulnérabilités potentielles.

De ce fait, des **tests de troisième niveau** sont parfois réalisés sur le mainnet, généralement en limitant l'accès à des utilisateurs participant au test, via des mécanismes de permissionnement positionnés au niveau du *smart contract* ou du protocole.

Dans la perspective d'une **certification**, l'audit pourrait débuter lorsque le *smart contract* ou le protocole a été déployé sur un testnet et vérifié sur cet environnement par les équipes de développement. **L'audit se déroulerait en deux phases.** L'auditeur procèderait d'abord à son propre examen du *smart contract*, sur le testnet, afin d'atteindre un premier niveau d'assurance quant à sa

[68] Sur la définition et le rôle des DAO, voir la partie « standards » de ce document.
[69] Bien qu'il ne soit pas forcément spécifique au smart contract ou au protocole sous revue (une même DAO peut ainsi gérer plusieurs protocoles).
[70] Dans l'univers Ethereum, certains *testnets* fonctionnent cependant avec une validation en « preuve d'enjeu » ouverte à tous les validateurs candidats, comme sur le mainnet.
[71] Une des techniques fréquemment employées pour assurer une proximité maximale consiste à réaliser un « *fork* » local de la blockchain principale (des outils permettent de faciliter cette opération).

conformité. Une fois cette **première phase validée**, le *smart contract* ou le protocole serait déployé sur le mainnet, où l'auditeur procèderait à des **vérifications complémentaires**. Durant cette seconde phase, **l'accès serait restreint** à certains utilisateurs expressément autorisés (comme l'auditeur), afin de limiter les risques[72]. **Le permissionnement ne serait levé que lors de l'obtention de la certification**. Toutefois, ce schéma pose la question des éventuelles contraintes imposées à l'architecture du *smart contract* par la nécessité de mettre en œuvre un mécanisme de permissionnement.

Dans ce contexte d'audit en deux phases, un participant souligne que les délais entre les phases de développement, de tests et de déploiement peuvent être très importants. Les composantes de gouvernance, de sécurité ou de conformité peuvent avoir évolué significativement durant ce laps de temps, ce qui impliquerait un périmètre à auditer en phase 2 substantiellement différent du périmètre audité en phase 1, ce qui serait de nature à remettre en cause l'intégrité de l'audit réalisé. Le participant suggère alors une pré-certification pour la phase 1 suivi d'une certification pour la phase 2 avec un délai maximum d'un mois entre les deux phases.

***

2. Quand auditer ou ré-auditer un *smart contract* ?

Trois situations doivent être distinguées : (i) **l'audit initial** de certification, c'est-à-dire la première fois qu'un *smart contract* ou un protocole est examiné ; (ii) **l'audit de renouvellement après expiration de la certification**, sans que le *smart contract* ait été modifié ; (iii) l'audit de **renouvellement** de la certification **après modification** du *smart contract* ou du protocole. La présente section aborde les questions relatives à chacune de ces trois situations.

2.1. La certification initiale soulève-t-elle des difficultés particulières ?

Le processus de certification se base essentiellement sur le code du *smart contract* ; toutefois, il ne s'y limite pas. C'est le cas en particulier de certains éléments relatifs à la **gouvernance** : si certaines dispositions peuvent être inscrites dans le code du *smart contract* – mécanismes de distribution des tokens, procédures de vote etc. –, d'autres doivent nécessairement se déployer *off-chain*. Ainsi, l'existence d'une documentation à l'usage des utilisateurs, ou encore la nécessité de préserver les intérêts des utilisateurs et des clients en cas de modification du *smart contract* – qui constitue un principe de haut niveau à inscrire dans un futur standard (*cf.* partie « standards » de ce document) – peuvent difficilement être intégrées dans un code informatique.

L'existence d'éléments *off-chain* dans le dispositif sous revue ne constitue pas une difficulté rédhibitoire ; elle demande en revanche un examen manuel (*cf.* section 1.3). De manière générale, il faut remarquer que le processus de certification initial tend à s'effectuer « sur plans », tant pour le code informatique que pour les éléments *off-chain*.

En revanche, une difficulté peut se poser lorsque le dispositif de gouvernance **n'est pas encore (entièrement) organisé au moment de la certification**. Il peut alors être difficile de l'évaluer. Le

[72] L'intégration d'une "*whitelist*" pour restreindre l'accès au smart contract nécessite d'apporter des modifications au code du smart contract. Cela revient donc à auditer un smart contract avec un code légèrement différent de celui du smart contract définitif. Le fait que le code définitif du smart contract ne soit pas identique à celui qui a été audité pourrait introduire un risque, qui n'a pas été à ce stade discuté par le groupe de travail.

problème se pose notamment pour les protocoles dont la gouvernance doit à terme être décentralisée, mais qui sont, au moment de la revue, gouvernés de manière centralisée. Face à cette difficulté, **un principe peut être avancé** : **soit le dispositif futur de gouvernance est décrit de manière univoque, et il peut alors être validé « sur plans »** ; soit il est décrit de manière trop peu précise, et cela fait obstacle à l'octroi de la certification.

Un problème proche est le cas de figure où un dispositif de gouvernance serait certifié, puis modifié pour sa partie *off-chain* (documentation etc.), faisant possiblement obstacle à ce que cette modification soit automatiquement détectée[73], contrairement à celles intervenant sur le code du *smart contract* (*cf.* section 2.3). Ce cas de figure fait partie des raisons pour lesquelles la certification doit nécessairement avoir une durée limitée : **il est essentiel que les *smart contracts* et protocoles fassent l'objet d'un réexamen régulier**, et ce notamment afin de **vérifier que les engagements pris quant à la gouvernance ont effectivement été tenus**.

2.2. Quelle doit être la durée de validité de la certification ?

Les participants du GT s'étaient préalablement accordés sur un principe fondamental : **la certification ne doit être accordée que pour une durée finie**[74], à l'issue de laquelle elle est automatiquement retirée (cf. la partie « standards »). À défaut, du fait d'un environnement en constante évolution, elle risque en effet de **devenir obsolète**, même sans changement de l'objet logiciel. En outre, un renouvellement périodique de la certification permet de détecter et corriger d'éventuelles dérives qui seraient apparues dans le fonctionnement d'un protocole. Mais à combien de temps fixer la durée de vie de la certification ?

La durée de **3 ans** est fréquemment citée par les participants du GT. À titre de comparaison, la qualification (*cf.* encadré 3) d'une solution informatique par l'Agence nationale de la sécurité des systèmes d'information (ANSSI) est attribuée pour une durée de 2 à 3 ans, et ne fait pas l'objet d'un renouvellement automatique[75]. Un participant suggère de retenir une durée de 3 ans, mais d'effectuer une **première revue (surveillance) du *smart contract* au bout de 2 ans**, afin d'évaluer sa résistance, notamment aux nouveaux types d'attaques. En cas d'alerte, cela permettrait notamment aux gestionnaires du *smart contract* de lancer les travaux d'adaptation – dont il ne faut pas négliger la durée – en vue de repasser une nouvelle certification au bout de 3 ans.

D'autres participants du GT suggèrent de **faire varier la durée de validité en fonction de la criticité du *smart contract***, notamment déterminée par l'importance des montants en valeur manipulés. Ainsi, plus le niveau de risque est élevé, plus les audits devront être fréquents, et plus courte doit donc être la durée de validité de la certification. Dans le même esprit, **un *smart contract* plus complexe devrait aussi nécessiter des audits plus réguliers**, ce qui pourrait justifier une moindre durée de vie de la certification.

2.3. Quels sont les critères devant déclencher un nouvel audit ?

[73] Il existe toutefois des outils de détection des changements de contenu sur les URL ; ceux-ci pourraient permettre d'identifier certaines modifications des éléments de gouvernance off-chain.
[74] Ce principe figurait déjà dans le document de réflexion de l'ACPR « *Finance décentralisée ou désintermédiée : quelle réponse réglementaire ?* » (Avril 2023).
[75] Par ailleurs, le règlement européen sur la sécurité (règlement (UE) 2019/881) retient généralement la durée de 5 ans.

Si la certification est de toute façon retirée au bout d'une certaine durée, elle peut également l'être avant cette borne en cas de changement du *smart contract*. Dans la partie « standards » du document (section 2.3), le GT affirmait le **principe d'un renouvellement de la certification en cas de changement**, accompagné le cas échéant de **dispositifs de proportionnalité** pour prendre en compte le cas de changements mineurs ou non significatifs.

En effet, **la difficulté inhérente à la notion de changement « majeur » ou « mineur » tient à sa subjectivité**. Or le fournisseur d'un *smart contract* peut avoir intérêt à qualifier de mineure toute modification, afin d'éviter d'avoir à renouveler la certification obtenue.

Lorsqu'elle attribue une certification à un produit, l'ANSSI demande le renouvellement de la certification en cas de changement majeur, en distinguant les modifications de fonctionnalités ou de l'architecture des simples corrections techniques. Toutefois, l'ANSSI traite avec des éditeurs de logiciels identifiés, établis sous la forme d'entreprise, et pour lesquels les **enjeux de réputation** sont fondamentaux. De ce fait, ces entités ont de **fortes incitations à collaborer** de bonne foi avec l'autorité. Dans l'univers de la DeFi, toutefois, les *smart contracts* peuvent être conçus par des groupes de développeurs impossibles à identifier. **Pour la certification des *smart contracts*, il faut donc fixer des règles plus strictes que dans l'univers de la sécurité informatique**.

A l'inverse, il serait absurde et contre-productif qu'un fournisseur de *smart contracts* ou de protocole soit tenu de demander une nouvelle certification trop souvent. **Une solution pourrait alors être de définir le niveau de criticité des différents éléments du *smart contract* ou du protocole lors de l'audit initial**. **Ainsi, le niveau de criticité ne serait pas déterminé par le fournisseur lui-même**[76]**, mais par une tierce partie** à qui reviendrait la décision finale. Un **audit de suivi** ne serait alors nécessaire que si l'un des éléments critiques est modifié.

En outre, afin de prendre en compte le risque qu'un fournisseur ne modifie un *smart contract* certifié sans en avertir le responsable de la certification (en recourant à une architecture imbriquée[77]), **la certification devrait automatiquement disparaître dès lors que l'une des fonctions critiques du *smart contract* a été modifiée** (et la publication du certificat technique serait automatiquement interrompue). Les participants du GT avaient indiqué, lors de la séquence consacrée aux standards, que cela était techniquement possible car : (i) la *blockchain* permet d'identifier toute modification du code informatique d'un *smart contract*[78], (ii) et la certification peut être informatiquement reliée à certaines des fonctions critiques du *smart contract*[79]. Par ailleurs la certification devrait mentionner la version du code audité.

***

3. Qui peut réaliser l'audit d'un *smart contract* ?

3.1. Le marché de l'audit du *smart contract* aujourd'hui

[76] Même si l'audité pourrait lui aussi concourir à l'exercice de par la connaissance de son produit.
[77] *Cf.* section 2.2 de la partie « standards » de ce document pour un rappel sur les architectures imbriquées.
[78] Certaines blockchains ne publient toutefois pas le code informatique des *smart contracts* qui y sont déployés (exemple : Solana), ce qui ne permet pas une telle identification automatique.
[79] *Cf.* section 2.4 de la partie « standards » du document.

Dans l'univers de la DeFi, la revue des *smart contracts* est parfois réalisée de manière classique, par des **entreprises d'audit spécialisées**. Celles-ci sont souvent engagées par les DAO qui gèrent les protocoles. Toutefois, d'après les participants du GT, les entreprises spécialisées dans l'univers de la *blockchain* ne disposent pas toujours de toutes les compétences et procédures permettant d'effectuer un audit de qualité. De fait, la discipline est jeune et **encore peu structurée**. Surtout, l'industrie du *smart contract* est actuellement confrontée à une **pénurie de talents**. Pour autant, selon les participants du GT, les porteurs de projets DeFi préfèrent souvent un audit de qualité incertaine que pas d'audit du tout. Ce constat souligne d'ailleurs, en creux, l'intérêt que les porteurs de projets pourraient trouver dans une démarche de certification.

Une autre partie de la revue de *smart contracts* tend à s'effectuer de manière moins classique, par des ***bug bounties***, c'est-à-dire l'organisation de concours en ligne dans lesquels sont récompensés les participants qui parviennent à trouver et corriger des vulnérabilités dans le code informatique d'un *smart contract*. Ces programmes **présentent des avantages** aux yeux des participants au GT. Ils incitent d'abord les développeurs à **rendre visible le code informatique** des *smart contracts* et à préciser les dépendances avec les autres objets logiciels. Ils permettent, en outre, d'**impliquer une communauté** de personnes intéressées, et ainsi de bénéficier d'une **expertise collective**. Enfin, les participants indiquent que **certains des individus les plus compétents** dans le domaine des *smart contracts* exercent plus volontiers leurs talents dans l'univers des bug bounties que dans le domaine de l'audit « classique ».

Cette pratique pose cependant un **certain nombre de difficultés dans l'hypothèse d'un développement et d'une institutionnalisation de la DeFi**. Tout d'abord, des entreprises pourraient avoir du mal à faire reconnaître la validité de ces examens par d'éventuels financeurs (banques, potentiels détenteurs de tokens de gouvernance etc.) ou d'autres parties prenantes (assureurs etc.). Ensuite, les récompenses liées aux bug bounties sont généralement versées en crypto-actifs sur la *blockchain*, à des adresses qui ne font pas toujours l'objet d'une vérification[80], **ce qui induit le risque de versement à des individus ou entités qui pourraient avoir partie liée avec le blanchiment de capitaux ou le financement du terrorisme**. Enfin, il arrive parfois que les participants à un bug bounty **s'abstiennent de révéler certaines vulnérabilités qu'ils ont identifiées**, dans le but d'utiliser ces informations pour attaquer le protocole. Enfin, les bug bounties se concentrent en général sur la seule question de la sécurité informatique, sans se préoccuper d'autres aspects, comme les règles de gouvernance.

Le bug bounty ne semble donc pas constituer une solution suffisante ni pérenne pour l'audit des *smart contracts* et des protocoles de DeFi, mais plutôt une **voie complémentaire**. Ainsi, les auditeurs pourraient, pour leurs travaux, **se référer aux résultats des bug bounties**. Cela permettrait d'intégrer explicitement le bug bounty dans les efforts d'ensemble visant à rendre les *smart contracts* plus sûrs, sans pour autant formaliser cette pratique au sein d'un cadre institutionnel[81].

S'agissant des audits classiques, il apparaît finalement que **la constitution d'un dispositif de certification largement répandu serait de nature à créer un marché de l'audit de *smart contract* plus profond et plus mûr**. Les possibilités de développement d'un tel marché paraissent d'autant plus crédibles que l'audit de *smart contract* présente de nombreuses similarités avec l'audit informatique

[80] Un participant indique cependant que la majorité des programmes de *bug bounties* dans le secteur de la blockchain recourent aujourd'hui à des dispositifs de KYC (« *Know your customer* ») afin de limiter ces risques.

[81] Il est à noter qu'il existe également un modèle mixte entre *bug bounty* et audit classique, appelé *competitive audit* (« concours d'audit »). Un smart contract est mis à disposition de différentes entités en concurrence. En fin de période d'audit, chaque entité partage son rapport. Les différents rapports sont évalués par un jury, qui attribue les récompenses.

classique (*cf.* section 1.1). En bénéficiant de formations professionnelles spécialisées dans le domaine de la *blockchain*, une partie des spécialistes de l'audit informatique pourraient ainsi venir répondre à la demande croissante d'audit dans le domaine des *smart contracts*, qu'un dispositif de certification pourrait accélérer.

3.2. La certification des *smart contracts* en pratique

Afin de répartir les rôles dans un dispositif de certification, **trois grands schémas d'organisation** apparaissent possibles, dépendant du caractère obligatoire ou non de la certification avant toute mise sur le marché d'un protocole, et de l'identité de l'entité chargée de délivrer la certification.

3.2.1. Une certification *ex ante* délivrée par une autorité publique, agréant pour ce faire un ensemble d'évaluateurs tiers

Dans ce premier schéma, la certification est obligatoire avant toute mise sur le marché d'un *smart contract* ou d'un protocole, et une autorité publique[82] est chargée de la délivrer. Du fait du grand nombre de protocoles et de *smart contracts* à certifier, l'autorité agrée des organismes évaluateurs (généralement privés), qui sont chargés de réaliser l'audit proprement dit. C'est par exemple le schéma retenu par l'ANSSI pour la délivrance de ses visas de sécurité (qualification et certification, *cf.* encadré 3).

L'agrément des évaluateurs constitue une garantie cruciale de la qualité du dispositif. L'autorité peut ainsi vérifier que les évaluateurs disposent des compétences de l'organisation et de l'indépendance nécessaires à l'accomplissement de leurs missions ; en outre, l'autorité publique exerce une **surveillance continue** des évaluateurs, et peut leur adresser des mises en garde, voire leur retirer l'agrément en cas de défaillances graves ou répétées.

Dans ce schéma, les évaluateurs procèdent au test des *smart contracts* ou des protocoles, mais le cahier des charges de ces tests – c'est-à-dire le standard de certification (ou a minima les principes de ce standard) – est déterminé par l'autorité publique. À l'issue de leur examen, les évaluateurs rédigent un **rapport d'audit** exposant les vulnérabilités détectées, leur résolution éventuelle, mais aussi un certain nombre d'anomalies mineures n'empêchant pas selon eux la certification. **Toutefois, c'est bien l'autorité publique qui décide souverainement d'octroyer ou non la certification, sur le rapport de l'évaluateur**.

L'avantage de cette organisation est d'abord de placer le dispositif de certification **sous le contrôle d'une autorité publique, gage de sa qualité, tout en démultipliant les capacités d'action de celle-ci** via un réseau d'évaluateurs privés agréés et surveillés. Les fonctions d'instruction (audit du système) sont séparées de celle d'octroi de la certification, ce qui permet aux auditeurs privés de se concentrer uniquement sur les aspects techniques de leur mission. En outre, ce système conduit indirectement à la **création et à la structuration d'un marché de l'audit de certification**, avec les avantages associés : outre la création d'emplois sur le territoire national ou européen, **le fait que les auditeurs soient en concurrence les incite à maintenir des prix compétitifs, et donc à limiter les coûts du processus pour les audités**. Enfin, il s'agit d'un système **déjà éprouvé par la pratique**. Toutefois, ce système présente une **certaine complexité**, ce qui peut se traduire par un **allongement des délais** nécessaires à l'obtention d'une certification.

[82] Nouvelle autorité publique spécialisée dans cette mission, ou autorité publique existante exerçant des missions dans des domaines connexes (audit informatique, surveillance du secteur financier etc.).

Encadré 3 : Les visas de sécurité de l'ANSSI

Les visas de sécurité ont pour objectif de permettre aux citoyens et aux entreprises **d'identifier facilement les solutions jugées les plus fiables par l'ANSSI**. Les visas de sécurité prennent deux formes : la qualification et la certification. Gages de sécurité pour les utilisateurs, ils constituent aussi un **atout concurrentiel** pour les fournisseurs de produits et de services de sécurité.

La **qualification** est la recommandation par l'État français de **produits ou services** de cyber-sécurité éprouvés et approuvés par l'ANSSI. Elle atteste de leur **conformité** aux exigences règlementaires, techniques et de sécurité promues par l'ANSSI. La recommandation de l'ANSSI permet aux fournisseurs de solutions d'accéder à certains **marchés réglementés** français et européens, car l'administration et certains opérateurs (opérateurs d'importance vitale etc.) sont tenus de recourir à des solutions qualifiées pour certaines de leurs missions.

La qualification est valable pour une **durée comprise entre 2 et 3 ans** ; elle ne fait pas l'objet d'un renouvellement automatique (le fournisseur doit faire une demande). Par ailleurs, le fournisseur est **tenu d'informer l'ANSSI de tout changement important** concernant le commanditaire ou le fournisseur de la solution.

La qualification repose :

- sur l'évaluation de la **performance** et de la **robustesse** (c'est-à-dire la capacité à résister aux attaques) pour les produits, et sur l'évaluation de la **compétence du prestataire** pour les services ;
- ainsi que sur l'évaluation de la **confiance**, c'est-à-dire la capacité du fournisseur à respecter sur le long terme un ensemble d'engagements pris auprès de l'ANSSI (pour les produits : confidentialité et protection des données de l'utilisateur, correction des failles et vulnérabilités etc. ; pour les services : maintien des compétences, etc.).

La **certification** ne concerne, quant à elle, que des **produits** et atteste de leur niveau de robustesse. La certification d'une solution permet à son fournisseur d'accéder à de nombreux marchés de cyber-sécurité en France et à l'international. La certification est attribuée pour une version donnée du produit, mais **sans limite de temps** (tant que le produit demeure inchangé) ; elle est renouvelée en cas de **changement majeur**.

Il existe dans le détail deux niveaux de certification :

- la « certification critères communs » (CCC) constitue la certification de plus haut niveau, reconnue internationalement ; elle se déploie sur une échelle de 7 niveaux d'assurance ;
- la « certification de sécurité de premier niveau » (CSPN), moins longue et moins coûteuse, repose sur des tests en « boîte noire » effectués en temps et délais contraints.

Pour **évaluer** la qualité et la conformité des produits et services candidats à la qualification ou à la certification, l'ANSSI se repose sur des **Centres d'évaluation de la sécurité des technologies de l'information (CESTI)**. Il s'agit de **laboratoires privés, agréés et surveillés par l'ANSSI**, et qui doivent présenter des garanties d'indépendance et d'impartialité vis-à-vis des produits et services qu'ils évaluent. C'est toutefois **l'ANSSI qui reste maître de la décision finale** d'attribuer ou non une qualification ou une certification, sur la base du rapport transmis par les CESTI.

3.2.2. Une certification *ex ante* délivrée par des évaluateurs tiers

Dans ce deuxième schéma, la certification doit toujours être obtenue avant toute mise sur le marché, mais ce sont les évaluateurs tiers qui sont chargés de la délivrer. Ici encore, ceux-ci doivent être préalablement agréés par une autorité publique, qui exerce la surveillance sur leurs pratiques, mais

cette autorité n'intervient pas directement dans la démarche de certification des *smart contracts* ou des protocoles[83].

Cette organisation, a le mérite de **simplifier et, logiquement, d'accélérer la délivrance de la certification, et de limiter drastiquement la charge de travail de l'autorité publique**. En revanche, le regroupement au sein d'une même entité (l'évaluateur-certificateur) des fonctions d'instruction et de décision peut créer des **risques** ; en outre, la **responsabilité** confiée aux évaluateurs dans une telle organisation peut être jugée **excessive**.

3.2.3. Une surveillance *ex post* plutôt qu'une certification *ex ante*

Dans ce troisième schéma, la certification d'un *smart contract* ou d'un protocole par une tierce partie n'est plus un préalable à sa mise sur le marché ou à sa mise en service. Le fournisseur réalise en revanche une **auto-évaluation de la conformité** de son produit par rapport aux exigences fixées par la réglementation[84], et ce avant la mise sur le marché ou la mise en service. Il **procède également à l'enregistrement de son produit dans une base de données** nationale ou européenne, ce qui assure l'information en continu de l'autorité publique chargée de la surveillance des *smart contracts* et des protocoles de DeFi. Grâce à ces informations, à son évaluation générale des risques, mais aussi aux alertes éventuelles remontant des utilisateurs, **l'autorité publique exerce la surveillance en continu des produits et services**. Pour ce faire, elle peut notamment requérir **l'accès au code** du *smart contract* ou du protocole, ainsi que **l'accès à l'auto-évaluation initiale**. Comme dans le premier schéma, l'autorité peut recourir à des **évaluateurs privés** pour l'assister dans ses missions de surveillance.

Ce schéma d'organisation est celui qui a été retenu pour la surveillance des systèmes d'intelligence artificielle (IA) à « haut risque » par le **règlement européen sur l'IA** récemment adopté (« *AI Act* »). Il permet d'éviter de recourir à une certification externe, et donc les phénomènes de **congestion** que celle-ci tend à générer. En se passant d'une certification proprement dite, les coûts peuvent également être réduits pour le fournisseur de *smart contract*, même s'il doit assumer les coûts internes liés à la nécessité de réaliser une auto-évaluation de conformité (en particulier, si le fournisseur fait pour cela appel à des prestataires externes, le coût peut être *in fine* équivalent).

Une telle organisation aurait le mérite de **reconnaître les efforts entrepris par l'industrie** pour sécuriser les *smart contracts*, et en particulier **le développement des diverses pratiques d'audit ou de vérification, qui pourraient ainsi être encadrées et standardisées**. Elle présente toutefois un **défaut majeur** : du fait du contrôle *ex post* qui est exercé, **des produits défectueux ou dangereux peuvent être mis sur le marché ou mis en service**, et ce sans que les utilisateurs puissent en avoir connaissance. La surveillance et les risques de sanction doivent normalement désinciter de tels comportements de la part des fournisseurs ; toutefois, dans l'univers de la DeFi où les fournisseurs peuvent être anonymes, **ce dispositif devrait, pour être efficace**, **s'accompagner de conditions supplémentaires** (par exemple exiger des fournisseurs qu'ils constituent des entreprises ou des associations sur le territoire national ou européen, dotées de représentants juridiquement responsables etc.).

[83] Ce type de dispositif est classique dans les règlementations européennes relatives à la sécurité des produits, comme la sécurité des jouets, des ascenseurs ou des équipements de protection individuelle.

[84] Comme dans les autres schémas d'organisation, la réglementation peut faire référence à des standards de qualité.

PARTIE 3 : PISTES REGLEMENTAIRES

Cette partie constitue une synthèse des réflexions du groupe de travail sur ce que pourraient être les grandes lignes d'une réglementation prévoyant la certification des *smart contracts* de la DeFi. Ces pistes ont pour objet de nourrir la réflexion des institutions européennes quant à la possibilité de légiférer dans ce domaine ; elles ne constituent pas une prescription de réglementation.

1. Les bases réglementaires pour une certification des *smart contracts*

1.1. L'objectif de la réglementation

L'objectif d'un cadre réglementaire permet de définir son champ d'application et le niveau d'exigence des dispositions qu'ils contient. Ainsi, par exemple, l'objectif du règlement MiCA (*Market in Crypto-Asset*) est de créer un cadre harmonisé favorisant le développement des activités basées sur l'utilisation des DLT et des crypto-actifs. Cet objectif se traduit par des règles encadrant l'offre et la fourniture de services sur crypto-actifs. Ce cadre prévoit notamment des règles destinées à protéger les détenteurs, investisseurs ou utilisateurs : règles prudentielles, agrément, obligation de transparence[85].

En ce qui concerne la DeFi, un éventuel mécanisme de certification des *smart contracts* permettrait de poursuivre plusieurs objectifs, par exemple, permettre le développement ordonné de la DeFi, assurer la protection des clients, limiter la transmission des risques dans le système financier, promouvoir la confiance dans la DeFi en la sécurisant, réduire l'arbitrage réglementaire entre juridictions, etc. Les travaux du GT soulignent l'importance de retenir comme **objectifs principaux la protection de la clientèle – qui doit ici s'entendre comme les utilisateurs de la DeFi – ainsi que l'établissement ou le maintien de la confiance**. Ce dernier objectif s'entend habituellement dans le domaine financier comme l'assurance pour l'utilisateur que le service effectivement fourni correspond bien à celui qui est annoncé, mais aussi qu'il bénéficie de garanties de bon fonctionnement du service (c'est-à-dire que même en cas de problème technique ou d'attaque informatique, des mesures existent pour empêcher ou, a minima, limiter les conséquences de ces évènements). Le maintien de la confiance suppose également que la DeFi soit le plus hermétiques possible aux risques de fraude, de BC-FT et plus généralement à toute délinquance financière. **La protection de la clientèle et le renforcement de la confiance constituent deux prérequis indispensables au développement ordonné de la DeFi.**

Dans cette perspective, **la certification devrait permettre d'empêcher, ou au moins de limiter les possibilités pour la clientèle d'être exposée et de souscrire des services jugés dangereux** (la dangerosité étant envisagée plus largement que sous l'angle de la seule sécurité informatique). C'est pourquoi les grands principes du standard définis dans la première partie des travaux du GT ne se limitent pas à la sécurité mais concernent aussi la gouvernance et la conformité du service. En ce sens,

[85] Le règlement MiCA exclut de son champ d'application les services sur crypto-actifs qui sont fournis de manière entièrement décentralisée sans aucun intermédiaire (considérant 22).Le règlement MiCA complète également la politique de lutte contre le blanchiment de capitaux et de financement du terrorisme (LCB-FT) de l'Union européenne, car il définit les prestataires de services sur crypto-actifs dont l'assujettissement aux règles européennes de LCB-FT est prévu par le règlement TFR (*Transferts de fonds*), entré en vigueur le même jour.

les réflexions du groupe de travail vont plus loin que les pratiques actuelles de l'industrie en matière d'audit des *smart contracts*.

1.2. Le champ d'application

1.2.1. Le champ d'application matériel

Le champ d'application d'une réglementation prévoyant la certification des *smart contracts* doit logiquement découler de cet objectif.

Ainsi, **les *smart contracts* utilisés à des fins non financières**, par exemple pour fournir des NFT à usage non financier ou des services de chaine logistique en vue d'assurer la traçabilité des produits et d'accélérer les transactions ou pour faciliter le transfert de propriété immobilière **n'auraient pas vocation à entrer dans le champ de la certification envisagée ici**.

De plus, l'utilisation de *smart contracts* dans le cadre d'activités déjà réglementées – par exemple par le règlement MiCA pour des *stablecoins*, ou par la directive MiFID II pour des instruments financiers, **pourrait également être exclue du champ d'application de la certification.** En effet, dans ce cas, il existe déjà un cadre réglementaire protecteur, mis en œuvre par des institutions financières autorisées et contrôlées par des autorités de supervision. Cependant on peut s'interroger sur le bien-fondé de faire peser une exigence de certification sur certaines classes d'acteurs seulement, d'autant plus que la frontière des activités décentralisées est ténue, compte tenu de l'intermédiation de services en DeFi. Dans ce contexte, **le GT souligne qu'une large part de ses travaux pourrait, sinon s'appliquer directement, du moins inspirer les acteurs utilisant des *smart contracts* pour fournir des services déjà réglementés.** Cela donnerait en effet aux utilisateurs des garanties de qualité et de sécurité sur ces services.

Le dispositif de certification envisagé peut-il s'appliquer aux oracles ?

Le problème de la manipulation d'oracles est identifié comme l'une des principales menaces dans l'univers DeFi, ce qui pourrait justifier leur certification. Pour autant, les oracles présentent généralement des particularités (traitement d'informations *off-chain*) qui empêchent d'appliquer tel quel le dispositif de certification discuté ici. Un dispositif de certification des oracles pourrait cependant être développé par ailleurs ; il présenterait par exemple un intérêt pour les oracles fournissant des informations d'identité (*cf.* encadré suivant).

1.2.2. Le champ d'application géographique

Pour être efficace, la réglementation relative à la certification devrait s'appliquer à une échelle large, c'est-à-dire au minimum à l'échelle de l'Union européenne[86]. Elle concernerait alors **l'ensemble des *smart contracts* éligibles** (c'est-à-dire inclus dans le champ d'application matériel défini dans la section 1.2.1 ci-dessus) **utilisés au sein de l'UE**, quel que soit l'origine géographique présumée de leur élaboration[87]. Un tel périmètre géographique permettrait i) **d'assurer la sécurité** des utilisateurs et

[86] Une application au niveau mondial constituerait la solution idéale. Une réglementation européenne pourrait constituer une première étape pour promouvoir une telle réglementation.

[87] Cette notion apparaît de toute façon délicate à manier, s'agissant d'objets logiciels dont les responsables ne sont pas toujours identifiables et dont la réalisation est parfois assurée par des équipes de développeurs situées dans des juridictions différentes (voire inconnues).

investisseurs européens, quel que soit le rattachement géographique présumé du *smart contract*, et ii) de **garantir des conditions de concurrence équitables** entre les fournisseurs, quelle que soit leur localisation géographique[88].

Il apparaît impossible d'empêcher des producteurs localisés dans une juridiction tierce de développer et de déployer un *smart contract* non certifié sur une *blockchain* publique. Il est en revanche possible d'interdire à **l'ensemble des intermédiaires** – fournisseurs **centralisés** ou **interfaces** assurant l'accès des utilisateurs aux services DeFi (*front-ends* etc.) – **opérant dans l'UE** d'interagir avec des *smart contracts* non certifiés. Cela reviendrait à en interdire l'accès à la très grande majorité des utilisateurs[89]. De ce point de vue, la création d'un statut de « producteur de *smart contract* » n'apparaît pas nécessaire pour interdire en pratique l'utilisation de *smart contracts* non certifiés dans l'UE.

Peut-on mettre en œuvre un filtrage par localisation géographique au niveau du smart contract et pourquoi?

Définir la localisation géographique du *smart contract* n'a pas forcément de sens théorique, et en tout état de cause ne constitue pas une notion utilisable d'un point de vue pratique. En revanche, il est possible d'identifier la localisation géographique des utilisateurs. Le filtrage des clients par localisation géographique constitue ainsi une technique permettant aux développeurs ou gestionnaires de *smart contracts* de respecter des réglementations locales. Ainsi, un *smart contract* non certifié dans l'UE pourra interdire l'utilisation de son service aux clients européens.

Techniquement mettre en œuvre un tel filtrage revient à intégrer un contrôle *ad hoc* de la localisation géographique des utilisateurs dans le code du *smart contract*. L'information quant à la localisation géographique de l'utilisateur étant de nature *off-chain* (c'est-à-dire que les données relatives à la localisation ne sont pas stockées sur la *blockchain*), elle devrait être fournie par un oracle. En outre, l'oracle ne devrait pas seulement s'appuyer sur les techniques habituelles d'identification par adresse IP, car celles-ci peuvent être facilement contournées (par exemple au moyen de VPN). Pour aboutir à un filtrage efficace, il semble que l'oracle devrait nécessairement s'appuyer sur des techniques d'identité décentralisée : l'oracle agit alors comme un tiers de confiance, qui est en mesure de générer *on-chain* (c'est-à-dire sur la *blockchain*, via un NFT) une information d'identité qui emporte une information de résidence.

1.2.3. Que certifier précisément ?

Pour atteindre l'objectif de la certification, il faut prendre en compte le service offert aux utilisateurs. En général, **la fourniture d'un service de DeFi repose sur un « protocole »**, c'est-à-dire l'assemblage d'un certain nombre de *smart contracts*[90]. Certains protocoles peuvent être composés d'un grand nombre de *smart contracts*, selon l'architecture retenue pour leur développement.

En **première approche**, la certification du protocole audité pourrait découler de la **certification de tous les *smart contracts*** le constituant, ainsi que des interactions entre eux.

[88] Il est en revanche possible que l'imposition d'exigences élevées dans l'UE puisse nuire à la capacité des fournisseurs européens de *smart contracts* à « exporter » leurs produits ailleurs dans le monde ; dans ce cas des mécanismes technique de débrayage pourraient être envisagés pour que les fonctionnalités de ces mêmes *smart contracts* puissent être opérées sans application des standards européens.

[89] Sur ce point, voir ACPR, document de réflexion DeFi, page 30

[90] On pourrait définir un protocole DeFi comme un ensemble de *smart contracts* liés entre eux par un lien fonctionnel et contrôlés par un même acteur ou groupe d'acteurs, concourant à la fourniture d'un service « financier » ou d'un ensemble de services « financiers » complémentaires.

La difficulté d'une telle approche, du moins au départ, réside dans son **coût global** : le stock de *smart contracts* à certifier est très large. Elle aurait toutefois comme intérêt de permettre la **réutilisation** d'un *smart contract* déjà certifié par d'autres projets. Cette approche pourrait notamment conduire à certifier des **bibliothèques de *smart contracts* en open source**, qui seraient ensuite facilement réutilisables par les développeurs de projets[91].

Cependant, l'application de l'ensemble des exigences du standard de certification à tous les *smart contracts* d'un protocole pose d'autres problèmes. En effet, si les exigences de sécurité doivent concerner tous les *smart contracts* (la défaillance d'une partie peut compromettre l'ensemble), les aspects liés à la gouvernance et à la conformité du service s'apprécient plus naturellement au niveau du service rendu, c'est-à-dire au niveau du protocole. Aussi la règle suivante pourrait-elle être avancée : **les protocoles certifiés devraient respecter l'ensemble des dimensions du standard (sécurité, gouvernance, conformité) tandis que les *smart contracts* qui les composent n'auraient à respecter que les éléments relatifs à la sécurité**.

Pour autant, un problème demeure : les protocoles ne sont pas aisément identifiables en tant que tels. En particulier, ils n'ont pas d'adresse en propre sur la *blockchain*, et ne constituent finalement qu'une construction logique résultant de l'interaction d'un certain nombre de *smart contracts*. Dans ce contexte, comment un utilisateur peut-il être certain qu'il interagit avec un protocole certifié ? Ce problème se pose de façon accrue en raison de la composabilité de la DeFi : un *smart contract* donné peut tout à fait être appelé par différents protocoles.

Dans ce cadre, une solution pourrait être **d'identifier les *smart contracts* caractérisant spécifiquement la fourniture du service**. Un service DeFi ne peut être rendu en accédant à n'importe quel *smart contract* : celui-ci doit pouvoir être appelé. En pratique, l'utilisateur peut accéder aux services DeFi de 3 façons : (i) par une interface utilisateur (un *front-end*) connectée au *wallet* de l'utilisateur[92], (ii) par l'explorateur de blockchain qui permet d'interagir avec un *smart contract* ou (iii) en exécutant un script permettant d'appeler un *smart contract*. L'utilisateur peut ainsi interagir avec le ou les *smart contracts* qui enclenchent la suite d'actions qui constitue le service. Dès lors, ces ***smart contracts* dits d'entrée**[93] permettent d'identifier de façon non équivoque un protocole[94]. Ainsi, à travers l'adresse unique d'un *smart contract* d'entrée, toute personne pourrait vérifier si elle interagit avec un protocole certifié (voir schémas 1 et 2 ci-dessous).

---

[91] Ainsi, au principe de la composabilité des *smart contracts* et des protocoles répondrait un principe de composabilité de la certification. De tels mécanismes de réutilisation seraient d'ailleurs encouragés par la démarche de certification, puisqu'un auditeur n'aurait pas à réexaminer la validité du code d'un smart contract déjà certifié. En outre, le mécanisme de certification deviendrait progressivement de plus en plus efficace, à mesure que le stock total de *smart contracts* certifiés augmenterait.

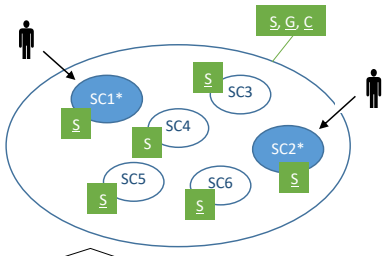
[92] La cinématique est la suivante : l'utilisateur interagit avec le *front-end* (interface utilisateur) ; le *front-end* appelle le *smart contract* à travers un protocole de communication (RPC) ; l'utilisateur signe l'appel et les transactions en résultant à travers son *wallet*. La cinématique d'appel via les explorateurs de blockchain est identique.

[93] Cette notion de *smart contract* point d'entrée est proche mais pas strictement équivalente à la notion de *smart contract proxy* introduite dans la partie Standard. Un *smart contract proxy* dont l'objet sera d'aiguiller vers les *smart contracts* actualisés d'un protocole pourra ainsi être considéré comme un point d'entrée du protocole. Toutefois, tous les protocoles ne s'appuient pas sur des *smart contracts proxies*, aussi convient-il d'introduire la notion générique de *smart contract* d'entrée.

[94] On s'intéresse ici aux *smart contracts* constituant des points d'entrée **publics** vers un protocole ; il existe en effet d'autres *smart contracts* dont l'accès est **permissionné**, utilisés pour des appels restreints par d'autres *smart contracts*. Ces derniers ne sont pas considérés comme permettant d'identifier de façon non équivoque un protocole donné.

En pratique, lors de l'audit de certification d'un protocole, les deux premières étapes – l'analyse du protocole et la définition du périmètre (voir la section 1.1.1 de la partie 2) – permettraient l'identification de ces *smart contracts* d'entrée. En partant de ces *smart contracts* d'entrée et en recensant le chaînage des appels entre *smart contracts*, l'auditeur pourrait alors délimiter le périmètre pratique du protocole à certifier.

Schéma 1 : Cas nominal



REGISTRE DES PROTOCOLES CERTIFIÉS	
Protocole P	• Adresse des points d'entrée (@SC1*, @SC2*) • Adresse des autres smart contracts composant le protocole (@SC3, @SC4, @SC5, @SC6) • Autres informations du protocole (nom, livre blanc, audits, version ou empreinte du code du protocole, adresse du front end web : « url web3 », …)
(autres protocoles…)	

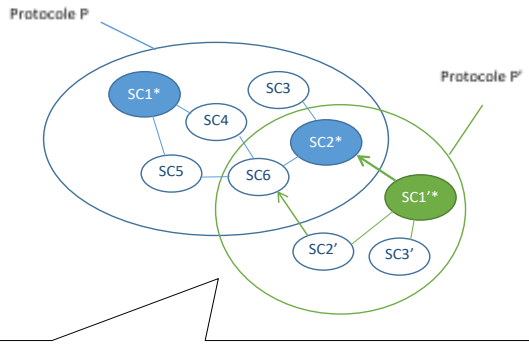
Le Protocole P est composé de 6 smart contracts, dont 2 sont des points d'entrées (« SC1* » et « SC2* ») pour appeler des fonctions du protocole. **Chacun** des 6 **smart contracts** seront évalués par l'auditeur suivant le standard **Sécurité** de certification ; l'auditeur analysera aussi les smart contracts **pertinents** du protocole en vue d'évaluer le **protocole** suivant les standards de **Gouvernance** et de **Conformité** de certification.

Il reviendra à l'auditeur, lors de l'analyse technique du protocole et de la définition du périmètre, de valider l'dentification des smart contracts considérés comme point d'entrée du protocole.

Un **registre** recense tous **les protocoles certifiés** en vue d'une part de constituer une **liste blanche** pour les utilisateurs et les acteurs de l'écosystème, et d'autre part pour énumérer les informations le caractérisant **dont le périmètre des smart contracts incluant les points d'entrée**.

Ici le protocole P est identifié par l'adresse des smart contracts servant de points d'entrée pour le protocole.

Schéma 2 : Cas d'un recouvrement entre protocoles



REGISTRE DES PROTOCOLES CERTIFIÉS	
Protocole P	• @SC1*, @SC2* • @SC3, @SC4, @SC5, @SC6 • url
Protocole P'	• @SC1'* • @SC2', @SC3', @SC6 • url'
(autres protocoles…)	

Le Protocole P' est constitué d'un smart contract point d'entrée (« SC1'* ») ainsi que de deux autres smart contracts (« SC2' » et « SC3' »). Pour rendre son service ce protocole s'appuie également sur deux smart contracts du protocole P, dont un point d'entrée du protocole P (« SC2* » et « SC6 »).

Le registre recense les protocoles P et P' certifiés. Les protocoles P et P' sont identifiés par l'adresse de leurs smart contracts point d'entrée (respectivement SC1*, SC2* pour le protocole P et SC1'* pour le protocole P').

Les smart contracts SC2* et SC6 utilisés par les deux protocoles apparaissent sous ces deux entrées de protocole dans le registre. Mais le smart contract SC2* reste identifié comme un point d'entrée caractéristique du protocole P.

Un utilisateur non expert (ou un service utilitaire ad hoc) pourrait s'appuyer sur l'information de l'adresse du frontal web pour vérifier qu'il utilise un protocole certifié (ici « url » et « url' »).

En conclusion, le schéma suivant pourrait s'appliquer :

- **La certification d'un protocole comprendrait les trois volets du standard (sécurité, gouvernance, conformité), tandis que la certification de chaque *smart contract* le composant ne s'effectuerait que sur le volet sécurité.**
- **Les *smart contracts* d'entrée porteraient l'identification de la certification**
- **La certification d'un *smart contract* « isolé » ou « autonome » devrait bien répondre aux trois volets du standard.**

Peut-on alléger la certification en se dispensant d'examiner certains smart contracts ?

Pour alléger le travail de certification, le GT a évoqué différentes façons de limiter l'audit aux *smart contracts* **portant véritablement les risques**. Une première approche consisterait à distinguer les ***smart contracts* « financiers »** – ceux effectuant les opérations touchant directement aux fonds en crypto-actifs – des ***smart contracts* « techniques »**, assurant le reste des opérations. Si elle apparaît logique à première vue, cette distinction ne tient pas compte des phénomènes de propagation possibles des failles entre les opérations considérées comme techniques et les opérations sur les fonds. Les travaux du GT ont ainsi montré qu'un défaut sur un *smart contract* chargé d'une opération basique (par exemple : effectuer une soustraction) peut compromettre la sécurité de l'ensemble du protocole. Pour les mêmes raisons, une seconde approche s'appuyant sur la distinction entre ***smart contracts* « critiques »** et « **non critiques** » pour le fonctionnement d'un protocole a été écartée. La notion de criticité (des fonctions) demeure en revanche opérante pour la gestion de la mutabilité des *smart contracts* (voir à ce sujet la section 2.3 de la partie « audit » de ce document).

1.3. Les critères de proportionnalité

Les services fournis par les *smart contracts* utilisés dans la DeFi peuvent présenter un risque plus ou moins grand pour l'utilisateur. **Il paraît alors justifié de proportionner le cadre de la certification à ce niveau de risque.** Ce mécanisme doit reposer sur des critères objectifs, vérifiables et transparents permettant de moduler le niveau d'exigence de la certification. La proportionnalité peut avoir pour objet d'alléger les règles pesant sur les protocoles peu risqués ; en sens inverse, elle peut conduire à imposer des exigences plus fortes aux protocoles les plus importants, notamment à ceux qui ont ou acquièrent une dimension « systémique » pour l'écosystème de la DeFi, en raison des répercussions que leur défaillance peut avoir sur d'autres protocoles (notamment en termes de liquidité).

Les discussions du GT ont principalement porté sur **les indicateurs les plus utilisés dans la DeFi afin d'évaluer leur pertinence comme critères pour évaluer le risque qu'un *smart contract* peut faire peser sur la clientèle** ou sur l' « écosystème » de la DeFi **: la *Total Value Locked* (TVL), le volume des transactions, le nombre de transactions et le nombre d'utilisateurs.**

La TVL est l'indicateur le plus utilisé pour comparer la taille et la popularité des différents protocoles DeFi. La TVL mesure l'ampleur des actifs bloqués dans le protocole, par exemple pour servir de collatéral ou fournir de la liquidité. Elle peut donc donner une indication de la confiance que les investisseurs ont de ce protocole mais aussi du niveau de risque de contrepartie, c'est-à-dire de perte des actifs pour les investisseurs. En effet, plus la valeur globale des actifs traités dans le protocole est élevée, plus la TVL est importante, plus les conséquences par exemple d'une attaque visant les actifs est potentiellement importante. Au contraire, s'il y a peu de TVL, il devrait en principe y avoir moins de risque.

Pourtant, la TVL présente des limites : (i) elle peut donner une impression trompeuse de hausse ou de baisse car sa valeur dépend du prix des actifs bloqués, dont le nombre peut rester stable mais la valeur fluctuer, (ii) elle ne représente pas avec précision la valeur totale verrouillée du fait de la comptabilisation multiple de certaines activités[95], et du fait qu'il n'existe pas de méthode standardisée pour la calculer.

De plus, la TVL ne reflète pas parfaitement la taille relative de toutes les catégories de protocoles DeFi. En premier lieu, certaines activités de la DeFi impliquent naturellement un verrouillage d'actifs plus important : c'est le cas du *staking* ou du prêt (*crypto-lending*) par exemple. D'autres activités

[95] Sur ce point, voir ACPR, Document de réflexion DeFi, page 10.

nécessiteraient de verrouiller moins d'actifs alors même que le volume d'activité – et le risque associé - pourrait être important : c'est le cas des plateformes d'échange (« DEX ») par exemple. En second lieu, les efforts des fournisseurs de protocole pour optimiser la performance des protocoles pourraient avoir tendance à faire diminuer le montant d'actifs verrouillés relativement au volume d'activité[96].

Des indicateurs possibles pour compléter la TVL

La TVL ne permet pas de mesurer certains risques liés aux flux d'actifs, comme les risques de défaillance de protocoles systémiques, ou encore le risque LCB-FT. La TVL ne permet pas non plus d'identifier tous les risques liés au stock des actifs mobilisés, comme le risque de liquidité. Pour mieux évaluer ce dernier risque, il serait intéressant de vérifier que le protocole a suffisamment de fournisseurs de liquidité ; un participant suggère ainsi de substituer, dans le cas des DEX ou du *crypto-lending*, un critère « LP » (*Liquidity provider* ou fournisseur de liquidité), qui serait plus représentatif que l'indicateur de TVL, compte tenu qu'une liquidité fournie n'est pas forcément verrouillée dans un protocole[97].

Un autre participant mentionne aussi le cas des autorisations de dépenses qui peuvent être accordées par des détenteurs de *wallets* à des protocoles : dans ce cas un investisseur peut être exposé à un risque sur les actifs détenus sur son *wallet*, alors même que ses actifs ne sont pas comptabilisés dans la TVL ou les actifs mobilisés du protocole. Un indicateur complémentaire pourrait donc ajouter les actifs mobilisables au volume des actifs mobilisés directement par le protocole, lorsque cela est pertinent.

Le volume des transactions est également souvent utilisé pour mesurer l'importance d'un protocole. Il peut être utilisé de façon alternative ou complémentaire à la TVL pour évaluer l'importance d'un protocole et le risque qu'il fait courir.

Les critères liés au nombre de transactions et au nombre d'utilisateurs peuvent également donner une approximation du niveau de risque, via le nombre d'utilisateurs lésés ou de transactions compromises en cas d'attaque ou de dysfonctionnement. Toutefois il faut tenir compte du fait que le nombre des transactions est parfois manipulé pour faire croître artificiellement la valeur d'un crypto-actif. Le nombre des utilisateurs, quant à lui, ne reflète pas toujours le nombre de personnes utilisant activement un protocole[98]. Aussi le nombre de portefeuilles utilisateur réellement actifs pourrait-il constituer un critère alternatif intéressant.

Des critères de centralité pour mesurer le caractère systémique d'un protocole – c'est-à-dire un protocole utilisé par un très grand nombre d'utilisateurs, indépendamment de sa TVL – pourraient aussi être envisagés, en s'inspirant de techniques de graphes ou de connectivité.

De manière générale, les valeurs précises permettant de fixer des seuils devraient faire l'objet d'une étude approfondie. **Ces valeurs devraient en outre pouvoir être régulièrement actualisées, afin de prendre en considération l'évolution du marché et des risques.** Enfin, des mesures de transition

[96] On peut notamment citer le cas des prêts-emprunts, qui sont généralement devenus plus efficients dans leur gestion de la liquidité.

[97] Comme pour tout critère, un critère « LP » devrait être considéré au regard de l'activité servie car la complexité des *smart contracts* d'un protocole de DEX n'est la même que la complexité des *smart contracts* d'un protocole de *crypto lending*, ce qui impacte directement les risques portés par ces activités.

[98] La pertinence de cet indicateur pourrait toutefois être relevée dans un contexte de recours à des mécanismes d'identité décentralisée (voir l'encadré *supra* « Peut-on mettre en œuvre un filtrage par localisation géographique au niveau du smart contract et pourquoi ? »)

pourraient être envisagées, pour éviter à des protocoles se situant à la lisière d'un seuil de passer d'une catégorie à une autre trop fréquemment (voir à ce sujet la discussion en section 1.4.3).).

Chaque type d'activité DeFi (prêt-emprunt, *staking* etc.) présentant des caractéristiques spécifiques, **il pourrait être justifié d'utiliser des critères et des seuils propres à chacune de ces activités. Il peut aussi s'avérer nécessaire d'utiliser plusieurs critères de façon complémentaire pour évaluer le risque d'un *smart contract*. En outre, il conviendrait généralement de retenir la moyenne ou la médiane des valeurs sur une certaine période, afin de neutraliser l'effet de variations soudaines qui ne seraient pas représentatives.** Le dépassement des seuils devrait par ailleurs faire l'objet d'une surveillance (voir *infra* 2.4). En cas de dépassement d'un seuil, un mécanisme de notification automatique par le *smart contract* aux autorités de surveillance pourrait également être envisagé.

Un participant a toutefois souligné l'importance d'une **démarche pragmatique** afin de **ne pas élaborer des règles de proportionnalité qui seraient trop complexes ou trop coûteuses à mettre en œuvre**.

Les mesures de proportionnalité dans le règlement MiCA (Market in crypto-asset)

En matière de proportionnalité, MiCA prévoit des règles pour diminuer ou augmenter les exigences en fonction du risque identifié pour les utilisateurs ou pour la stabilité du système financier :

- Exemption : lorsque certains critères indiquant par exemple qu'une émission de *stablecoins* est réservée à un nombre limité de personnes ou aux seuls investisseurs qualifiés, ou encore qu'elle ne porte que sur des montants limités, des exigences allégées sont prévues.
- Agrément : les exigences prudentielles pour fournir des services sur crypto-actifs ont été définies en fonction du niveau de sensibilité du service. Ainsi, alors que la plupart des services nécessitent de disposer de 50 000 à 150 000 EUR de capital minimum, le statut d'émetteur de monnaie électronique permettant d'émettre des jetons de monnaie électronique requiert 350 000 EUR de capital minimum.
- *Stablecoins* significatifs : si des *stablecoins* dépassent des seuils prévus par le règlement (exemple : nombre de détenteurs, montant de la réserve, etc.), l'émission est qualifiée de significative par l'Autorité bancaire européenne (ABE). Dans ce cas, l'émission passe sous supervision (complète ou partielle) de l'ABE et des exigences prudentielles supplémentaires sont appliquées (en matière de liquidité et de fonds propres en particulier).

1.4. Les schémas de réglementation possibles autour du mécanisme de certification

L'éventuelle introduction d'un cadre réglementaire prévoyant la certification des *smart contracts* conduirait forcément à dépasser les pratiques actuelles du marché, qui s'apparentent à une forme d'autorégulation dans laquelle des standards – couvrant essentiellement les aspects relatifs à la sécurité – sont développés par une partie de la communauté, et dans laquelle les audits de *smart contracts* sont effectués sur une base volontaire et non harmonisée.

Dès lors, les éléments relatifs à l'objectif, au champ d'application et aux critères de proportionnalité devraient être utilisés pour déterminer le contenu des trois schémas de réglementation qui semblent envisageables : la mise en place (i) d'une certification optionnelle, ou (ii) d'une certification obligatoire pour tous les *smart contracts* ou (iii) une certification obligatoire avec des mesures de proportionnalité.

Dans ces trois schémas de réglementation possibles, il y aurait néanmoins des principes communs : un standard de certification à respecter, des procédures d'audit et de certification approuvées par une autorité publique, et une information publique sur les certifications accordées.

1.4.1. La certification optionnelle

Dans ce schéma, la certification des *smart contracts* ne serait pas obligatoire mais elle pourrait être demandée par le développeur du *smart contract* ou éventuellement une tierce personne. Elle reposerait néanmoins sur un cadre réglementaire prévoyant le respect d'un standard et une procédure de certification ; toutefois, la **certification délivrée ici, puisque non obligatoire, aurait la nature d'un label qui donnerait un gage de crédibilité.** La mise en place de mesures d'incitation à la certification serait primordiale. En principe, la possibilité de communiquer sur ce label auprès des potentiels utilisateurs et investisseurs devrait constituer la principale mesure d'incitation. Toutefois, d'autres mesures incitatives ont été mentionnées par les participants, comme par exemple de faciliter l'accès aux services fournis par les institutions financières agréées (compte en banque, assurance, etc.)[99].

Un tel dispositif aurait l'avantage de permettre aux acteurs les plus modestes de pouvoir débuter leur activité sans certification, et de ne rechercher la certification que dans un second temps, au cours de leur phase de croissance. Les acteurs plus importants ou plus ambitieux se tourneraient, quant à eux, directement vers la certification.

Néanmoins, il faut souligner que ce dispositif ne permettrait pas de limiter au mieux des risques pour les utilisateurs et investisseurs, dans la mesure où ceux-ci pourraient interagir avec des *smart contracts* dont il n'a pas été vérifié qu'ils sont sûrs, qu'ils sont correctement gouvernés et qu'ils rendent correctement le service attendu (autrement dit les trois axes du standard pour la certification). De plus, les exemples récents de mise en place de régimes optionnels d'autorisation, en particulier dans le domaine des actifs numériques – visa optionnel pour les ICO, agrément optionnel pour les PSAN, tous deux prévus par la loi Pacte – ont montré les limites possibles de ce type de dispositif : malgré des intérêts exprimés pour ce type de régime, trop peu d'acteurs ont en pratique[100] demandé à obtenir un visa de l'AMF pour émettre des actifs numériques et peu de PSAN ont souhaité requérir l'agrément optionnel, en sus de l'enregistrement obligatoire.

Certification optionnelle	
Avantages	**Inconvénients**
Souplesse pour les acteurs (choix de recourir ou non à la certification). Permet une approche progressive (il n'y a pas d'échéance pour être conforme, la certification étant facultative).	Moindre recours à une démarche optionnelle si les parties prenantes ne perçoivent pas clairement un rapport bénéfice/coût favorable. Peut retarder une entrée sur le marché comparativement à des acteurs ne recourant pas à la certification.
La certification optionnelle peut être utilisée comme un mécanisme de transition avant une certification obligatoire.	Ne permet pas d'atteindre l'objectif de protection des utilisateurs (pour les *smart contracts* non certifiés)

[99] Par exemple, la certification pourrait permettre aux *smart contracts* d'être utilisés par des institutions financières qui, en raison des règles de LCB-FT auxquelles elles sont soumises, seraient contraintes de refuser les interactions avec des *smart contracts* non certifiés (si le standard de conformité intégrait une composante de conformité réglementaire), indépendamment du schéma de certification retenu (optionnel ou obligatoire).

[100] Dans le cas de l'agrément PSAN, les représentants de l'industrie ont évoqué d'une part la difficulté de répondre concomitamment aux exigences de l'enregistrement PSAN et d'autre part la difficulté de répondre aux exigences de cyber-sécurité (mise en œuvre d'un audit cyber ou disponibilité d'une offre d'assurance alternative).

1.4.2. La certification obligatoire

Dans ce schéma, la certification serait obligatoire pour tous les *smart contracts* entrant dans le champ d'application du cadre réglementaire. Dans ce cas, tous les *smart contracts* devant être certifiés, il ne serait pas possible en principe de fournir des services reposant sur des *smart contracts* non certifiés.

Ce type de certification obligatoire présente l'avantage de limiter au maximum les risques concernant les utilisateurs et investisseurs. Toutefois, **un tel schéma de réglementation représente une charge importante à supporter, en particulier pour des acteurs de taille modeste** ou pour des projets naissants, et pourrait donc impacter l'essor de l'écosystème innovant autour des *smart contracts*. Aussi, il pourrait être envisagé d'accorder un délai entre la mise en œuvre sur le marché d'un *smart contract* et l'obtention nécessaire de la certification, ce qui permettrait notamment de s'assurer de la pertinence du *smart contract* sur le marché, et donc de sa viabilité, avant d'engager des ressources pour assurer sa certification.

Certification obligatoire	
Avantages	**Inconvénients**
Remplit l'objectif de la réglementation	Charge non négligeable pour les acteurs de taille modeste ou les nouveaux projets.
Renforce la confiance des utilisateurs et des investisseurs dans l'écosystème (tous les *smart contracts* sont par défaut présumés certifiés).	Délai de certification à prévoir avant l'ouverture du service aux utilisateurs.
Meilleure lisibilité de l'offre et processus de gestion unique par les intermédiaires (tous les *smart contracts* sont présumés par défaut certifiés).	Charge de conformité accrue comparativement à des juridictions n'appliquant pas de certification obligatoire.
Économies d'échelle via la composabilité des *smart contracts* certifiés.	

1.4.3. La certification obligatoire avec des mesures de proportionnalité

Dans ce schéma de réglementation, la certification serait obligatoire mais des mesures de proportionnalité seraient appliquées afin d'adapter les exigences au niveau de risque de chaque protocole.

Les critères de proportionnalité discutés dans la section précédente (comme la TVL, le nombre de transactions, ou le nombre de clients) pourraient dès lors être utilisés pour (i) limiter le champ d'application, en excluant certains *smart contracts* considérés comme peu risqués, (ii) moduler le niveau des exigences attendues, par exemple en trois catégories : allégé, standard, élevé.

S'agissant du champ d'application, il pourrait ainsi être décidé que les projets les plus petits seraient dispensés de la certification obligatoire. Dans cette hypothèse, il pourrait être prévu de permettre les demandes de certification sur base volontaire (cela pourrait alors s'apparenter à un label, voir *supra* 1.4.1 sur la certification optionnelle).

Mais l'utilisation la plus naturelle des dispositifs de proportionnalité est de permettre d'appliquer, selon les cas, des mesures allégées ou bien des exigences supplémentaires destinées à limiter les risques. Une certification allégée pourrait par exemple consister à ne respecter qu'une partie du standard. En plus des projets peu risqués, une telle mesure pourrait aussi s'appliquer aux

renouvellements périodiques de certification (cf. partie 2 sur l'audit, section 2.3). Au contraire, si un *smart contract* apparaît plus risqué, il pourrait être décidé d'élever le niveau des exigences. Cela pourrait se traduire par des exigences supplémentaires dans le standard, mais aussi par un audit plus fréquent, ou bien plus poussé, avec par exemple l'obligation d'utiliser une plus grande variété de méthodes (cf. partie 2 sur l'audit, section 1.2).

Un tel dispositif aurait l'avantage de tenir compte du niveau de développement de chaque projet. Il permettrait ainsi aux activités de petite taille de démarrer sans trop de contraintes, et de ne répondre aux exigences d'une pleine certification qu'une fois qu'ils auraient atteint une taille critique. En revanche, un tel dispositif n'assurerait pas pleinement la protection des utilisateurs individuels : s'il garantirait la protection du plus grand nombre (puisque les protocoles comptant de nombreux utilisateurs ou traitant des volumes importants de crypto-actifs seraient obligatoirement certifiés), il n'empêcherait pas un individu donné d'interagir avec un protocole qui pourrait s'avérer dangereux.

Certification obligatoire avec mesures de proportionnalité	
Avantages	**Inconvénients**
Protection du plus grand nombre d'utilisateurs	Ne garantit pas à un utilisateur donné qu'il n'interagit pas avec un protocole dangereux.
Cadre proportionné et adapté au risque	Suivi dans le temps des conditions conduisant à l'application d'un régime de proportionnalité, et le cas échéant gestion de la transition vers le dispositif standard.
Facilite le démarrage des activités de taille modeste	*Délai de certification à prévoir avant l'ouverture du service aux utilisateurs.*
	Charge de conformité accrue comparativement à des juridictions n'appliquant pas de certification obligatoire.
Renforcement de la maturité de l'industrie des smart contracts et économies d'échelle par la composabilité des smart contracts certifiés.	

*(en italique les avantages et inconvénients présents également pour une certification obligatoire **sans** mesures de proportionnalité, cf. tableau précédent)*

***

2. Le mécanisme de certification

2.1. La publication de la certification

Le registre des certifications[101] devra garantir l'accessibilité de l'information. Cela recouvre en particulier deux fonctions : (i) rendre publique la liste des *smart contracts* de confiance (autrement dit certifiés) pour les utilisateurs et investisseurs, ainsi que pour des institutions financières ou des autorités publiques éventuellement amenées à interagir avec ces *smart contracts* ; **(ii) permettre**

[101] Le choix est fait ici de privilégier l'utilisation d'un registre mais, pour rappel, d'autres possibilités concernant la publication de la certification sont mentionnées dans la partie 1 sur le standard dans la section 2.4.

d'identifier les *smart contracts* déjà certifiés, et pouvant donc être réutilisés pour le développement de nouveaux services.

Au-delà de son caractère informatif, le registre participerait ainsi aux mesures d'incitation à la certification. Il pourrait d'ailleurs être prévu que seuls les *smart contracts* certifiés et publiés sur le registre soient rendus accessibles au public[102].

Le registre pourrait en outre être tenu sous une forme décentralisée, ce qui revêtirait un intérêt particulier dans le cas d'une multiplicité de certificateurs et faciliterait aussi son intégration au sein de l'écosystème décentralisé des *smart contracts*. Cependant, il parait difficile d'envisager un tel système sans mécanisme de permission, permettant aux entités chargées de délivrer la certification de disposer d'un privilège d'écriture.

Afin de bénéficier d'un niveau de confiance très élevé, le registre pourrait se conformer aux exigences d'un registre électronique qualifié au sens de l'Article 45 terdecies du règlement (UE) n° 910/2014 (dit eIDAS). Le registre bénéficierait alors d'une présomption légale quant au classement chronologique séquentiel unique et précis des certifications, ainsi qu'à leur intégrité[103].

Les informations figurant dans ce registre devraient enfin permettre de savoir si un *smart contract* a fait l'objet d'un régime allégé de certification, ou au contraire de l'application de mesures supplémentaires. La date d'octroi de la certification, et l'éventuel délai prévisionnel avant son renouvellement, devraient figurer dans le registre. Il pourrait également être envisagé que le registre rende public une partie du rapport d'audit ayant permis d'octroyer la certification, ou d'autres informations techniques utiles comme la version du *smart contract* audité[104]. Ici, un équilibre serait cependant à trouver entre l'intérêt de communiquer des informations utiles à la réutilisation d'un *smart contract* certifié, et le risque de divulguer des éléments pouvant faciliter l'attaque du *smart contract*.

2.2. Qui délivrerait la certification ?

Comme indiqué dans la partie 2 sur l'audit, un cadre réglementaire prévoyant une certification des *smart contracts* devrait s'appuyer sur les principes suivants :

- Une autorité est chargée d'établir la liste des auditeurs compétents pour évaluer les *smart contracts*.
- Les auditeurs respectent un cahier des charges permettant de garantir leurs compétences, la pertinence de l'audit et leur indépendance.
- La procédure d'audit comprend une phase d'échange menant à la correction éventuelle des problèmes identifiés.
- Le rapport d'audit est un des éléments permettant au certificateur d'accorder ou non la certification.

[102] Cette action de publication pourrait être réalisée par les entités déjà autorisées par les autorités compétentes (comme les prestataires de services sur crypto-actifs).

[103] Par ailleurs, s'appuyer sur un prestataire de service de confiance qualifié selon la réglementation eIDAS répondrait aux exigences de sécurité les plus élevées tout en harmonisant la responsabilité et la charge de la preuve sur l'enregistrement des certifications, comme prévu par l'Article 13 de cette réglementation.

[104] En vue de s'assurer que la version courante du *smart contract* déployé sur la blockchain est bien en phase avec la version du *smart contract* qui a obtenu la certification.

En outre, différentes organisations institutionnelles ont été envisagées, chacune étant associée à un certain nombre d'avantages et d'inconvénients (cf. section 3.2 de la partie « audit ») :

	Avantages	Inconvénients
Certification *ex ante* délivrée par une autorité publique agréant pour ce faire un ensemble d'évaluateurs tiers	Contrôle de l'autorité publique, gage de qualité. Indépendance des auditeurs. Création et structuration d'un marché de l'audit de certification. Système éprouvé par la pratique (ex. : ANSSI en France).	Certaine complexité du processus, pouvant se traduire par des délais allongés ou des coûts plus importants. Expertise et ressources nécessaires de l'autorité publique compétente.
Certification *ex ante* délivrée par des évaluateurs tiers	Processus plus simple, qui pourrait favoriser des délais et des coûts moindres.	Confusion des rôles d'auditeur et de certificateur, induisant des risques pour l'indépendance de l'analyse de l'auditeur.
Surveillance *ex post*	Reconnaît et tire parti des efforts de l'industrie pour sécuriser les *smart contracts*.	Des produits défectueux ou dangereux peuvent être mis sur le marché ou mis en service au détriment des utilisateurs. Volume important de *smart contracts* à surveiller sans que l'autorité de supervision puisse capitaliser sur une connaissance a priori des *smart contracts*. Risque de moindre réactivité ou de collaboration limitée des parties prenantes aux requêtes de l'autorité de supervision (le *smart contract* étant déjà en opération). Nécessite des conditions supplémentaires pour fonctionner (fournisseurs constitués en entreprises et localisés sur le territoire européen, juridiquement responsables etc.).

Ces différentes modalités d'organisation font apparaître **deux difficultés structurelles** de tout processus de certification des produits : i) les **ressources limitées des autorités publiques**, qui ne leur permettent que difficilement de procéder elles-mêmes à l'audit des produits – dans l'hypothèse où elles en ont les **compétences techniques** (ce qui, dans le cas des nouvelles technologies telles que la *blockchain* et les *smart contracts* reste un enjeu) ; ii) les enjeux **d'indépendance et les possibles conflits d'intérêts des entités privées** chargées d'examiner la conformité des produits.

Sur ce dernier aspect, le GT a par exemple envisagé qu'une entité privée – association ou fondation, disposant d'un niveau suffisant d'expertise –, puisse être impliquée dans la certification, soit pour définir la liste des auditeurs, soit pour délivrer la certification, voire les deux. Ce schéma permet de donner la compétence de certification à une entité qui dispose de l'expertise technique nécessaire. Mais il pose des questions aiguës en terme d'indépendance, et même de financement d'une telle

entité. À l'heure actuelle, le marché de l'audit des *smart contracts* apparaît **insuffisamment structuré** pour envisager une telle organisation.

Ainsi, des trois grandes modalités d'organisation envisagées, **la première apparaît en théorie comme la plus satisfaisante**. Il s'agit d'une organisation à **deux étages**, où l'autorité publique délivre la certification, mais **délègue à des acteurs tiers, préalablement agréés**, la charge d'auditer les protocoles et *smart contracts*. **Un tel système permet ainsi de tirer parti des ressources et des compétences des acteurs privés de l'audit, tout en contrôlant leur action**, d'une part en les agréant au préalable (agrément qui peut être remis en question en cas de manquements constatés), d'autre part en ne leur permettant pas d'accorder eux-mêmes la certification, ce qui répond aux enjeux d'indépendance de la fonction d'audit.

Ce système, déjà éprouvé par la pratique, aurait également l'avantage de favoriser la **structuration d'un marché de l'audit de certification**. Outre la création d'emplois sur le territoire européen, **la concurrence entre auditeurs les inciterait à conserver un haut niveau de compétences et à pratiquer des prix compétitifs, ce qui limiterait les coûts du processus pour les audités**. Cette solution qui présente beaucoup d'avantages, pourrait néanmoins difficilement s'appliquer si le nombre de *smart contracts* à certifier était important. **Dans cette hypothèse, la balance entre les avantages et les inconvénients mentionnées dans le tableau ci-dessus pourrait conduire à considérer comme plus réaliste une certification *ex ante* délivrée par des évaluateurs tiers.**

Exemple d'une procédure d'autorisation d'un produit dans le domaine de l'accès aux données de paiement, impliquant un audit et la décision d'une autorité

En 2015, la deuxième directive européenne sur les services de paiement (DSP2) a défini deux nouveaux services : un service d'initiation de paiement et un service d'information sur les comptes. Pour fournir ces services, des prestataires de services de paiement (PSP) spécialisés et autorisés par l'ACPR (PSP tiers) accèdent aux informations des comptes de paiement. La DSP2 oblige les banques à développer des interfaces d'accès permettant de collecter des données de paiement ou encore de transmettre des ordres de paiement de façon sécurisée et conforme au mandat donné par les utilisateurs. Pour se conformer à cette obligation les PSP peuvent développer plusieurs types d'interface. L'une d'entre elles, appelée « interface dédiée sans mécanisme de secours en cas d'indisponibilité », nécessite une approbation de l'ACPR.

Sur la base d'un dossier de demande d'exemption de mécanisme de secours, l'ACPR évalue si l'interface dédiée remplit les exigences définies par la DSP2, en vue de garantir la continuité de service. Le dossier comprend notamment des indicateurs de performance, ainsi qu'un rapport sur les aspects liés à la sécurité, élaboré par un CESTI (un Centre d'Évaluation de la Sécurité des Technologies de l'Information, c'est-à-dire un prestataire de service indépendant agréé par l'ANSSI). Ce rapport d'audit de la sécurité de l'API mesure sa conformité avec le référentiel de sécurité élaboré par la Banque de France, et comprend notamment des tests de montée en charge. Les exigences couvrent la gouvernance et l'organisation des développements suivant les bonnes pratiques, l'évaluation des risques et les contre-mesures associées, l'authentification du PSP tiers, la sécurisation des flux selon les normes en vigueur, la traçabilité et l'interopérabilité. Ce référentiel a fait l'objet d'une consultation des acteurs de la Place.

Sur la base de ce rapport et des échanges avec le demandeur, la Banque de France remet un avis sur la sécurité de l'API à l'ACPR. Si les conditions sont réunies, l'ACPR accorde l'exemption. Cette décision fait l'objet d'une publication au registre officiel de l'autorité.

2.3. Qui assumerait le coût de la certification ?

2.3.1. Le coût de la certification

Le coût de la certification dépend au premier chef de la complexité et du volume du ou des *smart contracts* à certifier. Comme vu précédemment, certains protocoles reposent sur une architecture complexe, faite d'appels entre *smart contracts*, ce qui a nécessairement pour effet de complexifier leur audit. D'autres facteurs peuvent jouer sur le coût de la certification, et notamment des facteurs réglementaires : **ainsi, pour des *smart contracts* jugés particulièrement risqués, si des exigences supplémentaires étaient appliquées au titre des mesures de proportionnalité** (voir supra, section 1.4.3), **cela ferait augmenter le coût de la certification.** À l'inverse, si certaines briques de *smart contracts* déjà certifiés étaient réutilisées dans le développement d'un nouveau protocole (voir supra, section 2.1), le coût de la certification pourrait diminuer. **Des exigences allégées au titre de la proportionnalité pourrait également faire baisser le coût de la certification pour les *smart contracts* éligibles. De la même manière, le cas d'un renouvellement périodique de la certification devrait en principe justifier un coût relativement limité.**

Il est à noter que pour les protocoles les plus importants, l'audit est généralement complété par des *bug bounties* (concours auprès de la communauté, dont l'objet est d'identifier les potentielles failles du code, voir à ce sujet la partie 2 sur l'audit), dont les récompenses peuvent atteindre des sommes importantes (parfois plus d'un million d'euros).

2.3.2. Qui prendrait en charge le coût de la certification ?

C'est en principe le producteur du *smart contract* qui devrait supporter le coût de la certification. Celui-ci a en effet un intérêt naturel à la certification de son produit, pour qu'il puisse être utilisé largement. La publication de la liste des *smart contracts* certifiés sur un registre public pour faciliter la communication auprès des utilisateurs et des investisseurs, ou d'autres incitations – voir les mesures évoquées plus haut – pourraient d'ailleurs encourager la certification.

De plus, **des fournisseurs centralisés de services sur crypto-actifs** (plateforme de négociation, par exemple) peuvent avoir un intérêt à proposer à leurs clients un service reposant sur un protocole décentralisé ; dans ce cas, ils **peuvent avoir intérêt à demander eux-mêmes la certification** du protocole en question, et donc à en supporter le coût. De même, des communautés de développeurs ou des acteurs de l'industrie logicielle peuvent avoir intérêt à faire certifier des bibliothèques de *smart contracts* – par exemple en open source – de manière à ce que ces logiciels puissent facilement être réutilisés. Dans ce dernier cas, le financement d'une telle démarche pourrait reposer sur des pratiques s'assimilant à du financement participatif dans le cadre d'une communauté de développeurs s'associant à cette fin, ou encore reposer sur un système de paiement de droits d'utilisation.

2.4. Le rôle des pouvoirs publics

Un cadre réglementaire prévoyant la certification des *smart contracts* utilisés dans la DeFi, devrait donner aux pouvoirs publics des moyens de faire appliquer la réglementation. Les pouvoirs confiés à ou aux autorités chargées de faire respecter cette réglementation devraient prendre en considération certaines caractéristiques de cette activité et en particulier le fait qu'il n'y aurait pas de statut pour les développeurs de *smart contract*, mais une certification du produit développé par ces derniers.

Dans les grandes lignes, ces pouvoirs pourraient comprendre les éléments suivants :

- **Élaborer les principes, en collaboration avec le marché, du standard à respecter (cf. partie 1 de ce document) ;**
- **Définir la liste des auditeurs, ainsi que le cahier des charges à respecter ; contrôler régulièrement la qualité de leur activité d'audit ;**
- **Délivrer la certification, ou s'assurer que la procédure de certification est conforme au cadre réglementaire, si la certification est délivrée par une autre entité ;**
- Tenir le registre des certifications ou s'assurer que seuls les certificateurs ont accès à ce registre en écriture ;
- Surveiller le bon fonctionnement du marché : par exemple s'assurer que les intermédiaires agréés n'interagissent qu'avec des *smart contracts* certifiés ; examiner les dépassements de seuil (si des mesures de proportionnalité sont appliquées), ou s'assurer que le ou les certificateurs surveillent ces dépassements ; collecter et traiter les remontées d'information, notamment les réclamations des clients ;
- Lorsque nécessaire, émettre des alertes sur certains services défaillants, voire procéder à la révocation d'une certification, et le cas échéant imposer des sanctions.

Annexe – Questionnaire de consultation (février 2025)

Le présent questionnaire accompagne le rapport du groupe de travail du Forum fintech ACPR-AMF sur la certification des *smart contracts*.

Il a pour objectif de recueillir l'avis des acteurs du secteur financier et du secteur des crypto-actifs, ainsi que celui de toute autre partie concernée (chercheurs, prestataires, autorités de contrôle, etc.) sur les questions évoquées dans ce rapport. Les répondants sont invités à illustrer leurs commentaires par des cas concrets et à préciser les références des éventuels travaux

Les réponses sont à envoyer aux adresses fintech-innovation@acpr.banque-france.fr et innovation@amf-france.org à l'échéance du **10 mars 2025.**

À l'issue de la consultation, une synthèse sera publiée.

Partie 1 du document : standards

Q 1 : avez-vous des commentaires sur les principes de **sécurité** énoncés par le document ?

Q 2 : avez-vous des commentaires sur les principes de **gouvernance** énoncés par le document ?

Q 3 : avez-vous des commentaires sur les principes de **conformité du service** énoncés par le document ?

Q 4 : souhaitez-vous commenter d'**autres aspects** développés dans la **partie 1** du document ?

Partie 2 du document : audit

Q 5 : avez-vous des commentaires à formuler ou des compléments à apporter sur les **méthodes d'audit** exposées en partie 2-1 ?

Q 6 : avez-vous une opinion sur la **durée de validité** d'une certification ?

Q 7 : souhaitez-vous commenter d'**autres aspects** développés dans la **partie 2** du document ?

Partie 3 du document : pistes règlementaires

Q 8 : avez-vous des commentaires ou remarques sur les développements relatifs aux **bases règlementaires** (objectifs, champ d'application, critères de proportionnalité, différents schémas règlementaires possibles)?

Q 9 : quelle est votre opinion sur la discussion développée dans le III-1.2.3 tendant à concilier la certification des **protocoles** et celle des ***smart contracts*** sous-jacents ?

Q 10 : souhaitez-vous commenter d'**autres aspects** développés dans la **partie 3** du document ?

Annexe – Glossaire

Analyse dynamique : dans le cadre de l'audit d'un smart contract, les méthodes d'analyse dynamique consistent à observer et analyser le comportement en cours d'exécution du code, dans un environnement de test, en vue de déceler plus aisément les vulnérabilités ou bien d'en déceler des vulnérabilités spécifiques. L'analyse dynamique est complémentaire à l'analyse statique.

Analyse statique : dans le cadre de l'audit d'un smart contract, les méthodes d'analyse statique consistent en l'inspection systématique des lignes de code pour en identifier les vulnérabilités. L'analyse statique prend deux formes principales, souvent combinées en pratique : la revue manuelle de code par l'auditeur et la revue automatique.

API (*application programming interface* ou interface de programmation d'application) : interface logicielle permettant de « connecter » un logiciel ou un service à un autre logiciel ou service, afin d'échanger des données et des fonctionnalités.

Blockchain ou *distributed ledger technology (DLT)* : s'entend dans la réglementation financière française comme un dispositif d'enregistrement électronique partagé. Il s'agit d'un registre électronique qui conserve les données des transactions et qui est partagé et synchronisé entre un ensemble de nœuds du réseau d'utilisateurs, fonctionnant au moyen d'un mécanisme de consensus. Les conditions d'accès au réseau et d'utilisation du registre (les « permissions ») déterminent si cette blockchain est publique, c'est-à-dire ouverte à tous, ou privée, c'est-à-dire réservée à certains utilisateurs.

Bug bounties : concours en ligne visant à récompenser les participants parvenant à déceler et corriger des vulnérabilités dans le code informatique d'un smart contract faisant l'objet du concours.

Bytecode : traduction technique intermédiaire du code informatique d'un smart contract, non lisible par un humain, en vue d'une exécution des fonctions du smart contract dans l'environnement de la blockchain (suivant les environnements blockchain le *bytecode* sera interprété ou bien compilé en langage machine).

Certificat ou Certificat technique : résultat technique d'un processus de certification dont l'exposition permet de montrer, notamment aux utilisateurs, qu'un smart contract donné ou un ensemble de smart contracts définissant un protocole DeFi a bien été certifié ainsi que le statut et les caractéristiques de cette certification.

Crypto-actif : une représentation numérique d'une valeur ou d'un droit qui peut être transférée et stockée électroniquement au moyen d'une blockchain. Certains crypto-actifs sont désignés par le terme « jeton » (*token*).

Clé : paramètre utilisé en entrée d'une opération cryptographique (chiffrement, déchiffrement, scellement, signature numérique, vérification de signature). Une clé de chiffrement peut être symétrique (la même clé sert à chiffrer et à déchiffrer) ou asymétrique : c'est ce second cas qui est utilisé dans les blockchains. On recourt alors à deux clés différentes : la clé publique est utilisée pour le chiffrement, tandis que la clé privée, qui permet le déchiffrement, est gardée secrète.

Code source (ou informatique) d'un smart contract : ensemble des fonctions décrivant le comportement d'un smart contract, écrites dans un langage de haut niveau et dans un format de type

texte. Le code est rédigé par un développeur et lisible par un humain (un auditeur par exemple) ; il doit être traduit ou compilé dans un langage technique (non lisible par des humains) en vue d'être déployé puis exécuté sur la blockchain.

Compilation : processus technique assurant la traduction du code source d'un smart contract en langage intermédiaire (« *bytecode* ») compréhensible par l'environnement d'exécution de la blockchain.

Couche des interfaces (Interface layer) : ensemble des ressources logicielles permettant à un utilisateur d'interagir avec les smart contracts des protocoles DeFi, qui constituent la couche applicative ; il s'agit typiquement d'applications décentralisées avec une interface web, mobile ou de bureau, ou bien encore de wallets. La couche des interfaces est empilée au-dessus de la couche applicative, elle-même empilée au-dessus de la couche d'infrastructure (la blockchain).

Cryptographie : discipline s'attachant à protéger des messages, en assurant leur confidentialité, leur authenticité et leur intégrité, en s'aidant souvent de *clés*. Elle vise à rendre les informations inintelligibles à d'autres personnes que leur destinataire.

Crypto-lending : activité consistant pour un utilisateur (prêteur) à transférer un certain montant valorisé en crypto-actifs ou en fonds à un autre utilisateur (emprunteur), en l'échange d'un engagement de l'emprunteur à restituer au prêteur un montant valorisé équivalent en crypto-actifs ou en fonds, ainsi que des éventuels intérêts, à une date future ou lors de la survenue d'un événement convenu.

ERC-20 : norme technique pour l'émission et la mise en œuvre de jetons sur la blockchain Ethereum.

EVM (*Ethereum Virtual Machine*) ou Machine virtuelle Ethereum : environnement virtuel décentralisé conçu pour exécuter des smart contracts et gérer l'état de la blockchain Ethereum ; l'EVM exécute le code de manière cohérente et sécurisée sur tous les nœuds Ethereum. En pratique chaque nœud exécute l'EVM, qui exécute à son tour les smart contracts, utilisant du « gaz » pour mesurer l'effort de calcul requis pour les opérations, en vue de garantir une allocation efficace des ressources et la sécurité du réseau. L'EVM est caractérisée par quatre fonctions clefs : la gestion de l'état de la blockchain Ethereum ; l'exécution des smart contracts ; le stockage des données ; le calcul des frais de gaz.

Fuzzing (test à données aléatoires) : méthode d'analyse dynamique du code. Elle consiste à générer un grand nombre d'entrées aléatoires, et potentiellement malveillantes, pour tester de manière approfondie le comportement du code dans un grand nombre de scénarios.

Gas ou Gaz : unité permettant de mesurer la puissance de calcul nécessaire à l'exécution des opérations dans le cadre d'une interaction avec la blockchain. La perception des frais de gaz permet de récompenser les validateurs de transactions ainsi que de concourir à la sécurisation du réseau de la blockchain.

KYC (*Know your customer*) ou connaissance du client : nom donné au processus permettant de vérifier l'identité des clients d'une entreprise. Le terme est également utilisé pour faire référence à la réglementation bancaire qui régit ces activités. Les processus *KYC* sont utilisés pour s'assurer de l'identité et de la probité des clients, et ont notamment pour but de prévenir l'usurpation d'identité, la fraude fiscale, la corruption, le blanchiment d'argent et le financement du terrorisme.

Liquidity Provider (LP) ou Fournisseur de liquidités : un fournisseur de liquidité crypto est une personne ou entité apportant de la liquidité sur une plateforme crypto en y verrouillant les crypto-actifs qu'elle détient en vue d'obtenir des commissions de transaction (*trading fees*) en retour.

Mutabilité d'un smart contract : caractéristique d'un smart contract à pouvoir évoluer dans son comportement ou bien à être substitué (notamment à travers un « proxy ») par une nouvelle version du smart contract, bien que tout smart contract déployé sur une blockchain soit immuable.

NFT (*Non-fungible token*) ou jeton non fongible : jeton émis sur la blockchain dont les caractéristiques de non fongibilité et d'unicité ne permet pas l'interchangeabilité ou la substituabilité avec un autre jeton (du même émetteur ou de tout autre émetteur).

Off-chain : environnement technique en dehors de la blockchain ; contraire de *on-chain*.

Open source ou code source ouvert : méthode d'ingénierie logicielle qui consiste à développer et maintenir un logiciel, ou des composants logiciels, suivant un mode de collaboration ouvert, et de laisser en libre accès le code source produit en vue de l'utiliser, l'analyser, le modifier et le redistribuer.

Oracle : entité transportant des informations du monde physique vers des *smart contracts*. Il fait le lien entre le monde physique et une blockchain, et permet aux *smart contracts* de ne pas être limités aux informations internes à la blockchain.

Organisation autonome décentralisée (DAO) : composante usuelle (mais pas systématique) des protocoles *DeFi*, visant à en organiser la gouvernance ; elle est définie habituellement par la communauté des détenteurs de jetons de gouvernance, les *smart contracts* qui régissent ses règles de fonctionnement et les actifs qu'elle contrôle (*treasury* du protocole).

Protocole DeFi : ensemble de smart contracts interagissant suivant un lien fonctionnel et contrôlés par un même acteur ou groupe d'acteurs, qui concourt à la fourniture d'un service « financier » ou d'un ensemble de services « financiers » complémentaires.

Smart contract : protocole informatique qui facilite, vérifie et exécute des transactions sur blockchains. Ces programmes informatiques ne sont pas « intelligents », dans le sens où ils ne modifient pas leur comportement au fil du temps, mais se bornent au contraire à exécuter un code lorsque sont remplies des conditions prédéfinies. Les *smart contracts* ne constituent pas toujours non plus des contrats, au sens juridique du terme.

Smart contract composite : Smart contracts interagissant avec des smart contract tiers, qui peuvent être considérés comme des composants externes.

Smart contract proxy : smart contract intermédiaire permettant de rediriger les utilisateurs vers la nouvelle version actualisée d'un smart contract déployé ; cette technique permet de pallier à l'immuabilité technique d'un smart contract déployé sur la blockchain afin de le substituer par une nouvelle version du smart contract comportant une évolution de son comportement ou la correction de bugs.

Solutions de *Layer 2* : solution de passage à l'échelle des blockchains (alternatives aux solutions dites de *layer 1*), dont le principe est de traiter une partie des transactions hors chaîne, en n'enregistrant que le minimum d'informations dans la chaîne principale (considérée comme le *layer 1*).

Staking ou *Crypto-staking* : processus d'immobilisation de crypto-actifs en vue de concourir aux opérations techniques des mécanismes de consensus pour les blockchains de type preuve d'enjeu (*Proof-of-Stake*, *PoS*) ; ces opérations permettent l'octroi de privilèges pour valider des blocs sur la blockchain, ce qui génère au final des récompenses pour l'opérateur du processus.

Testnet ou Blockchain de test : environnement de test public sur blockchain permettant de déployer et tester un smart contract avant le déploiement et l'ouverture du service sur la blockchain principale de production (« mainnet »).

TVL (Total value locked) : indicateur couramment utilisé dans la DeFi pour comparer la taille et la popularité des différents protocoles. Il mesure l'ampleur des actifs conservés pour le compte de tiers dans le protocole.

Vérification formelle (ou méthodes formelles) : méthode de vérification du code informatique, complémentaire aux méthodes d'analyse statique ou dynamique. Elle vise à prouver, en analysant la sémantique des programmes et en s'appuyant sur des règles mathématiques et logiques, la conformité du code analysé au regard de ses spécifications, rédigées dans un langage formel non ambigu.

Wallet (portefeuille) : interface contenant une clé publique pour recevoir des crypto-actifs, et une clé privée pour y accéder. Les crypto-actifs ne sont pas stockés sur le *wallet* (ils demeurent toujours sur la blockchain) ; contrairement à ce que son nom laisse entendre, le *wallet* constitue donc davantage un porte-clés qu'un portefeuille. Le *wallet* peut être hébergé (*custodial*), c'est-à-dire qu'un tiers détient la clé privée et donc *in fine* le contrôle sur les crypto-actifs. Avec un *wallet* non hébergé (*non-custodial*), au contraire, l'utilisateur exerce directement le contrôle sur ses fonds. Enfin, certains portefeuilles sont logiciels et connectés à internet (*hot wallets*), ce qui les rend plus faciles d'utilisation, tandis que d'autres sont des portefeuilles matériels, c'est-à-dire des dispositifs physiques hors ligne (*cold wallets*), ce qui est de nature à réduire les possibilités d'attaque.

ZKP (Zero-knowledge proof) ou preuve à divulgation nulle de connaissance : protocole cryptographique permettant à un acteur de démontrer la véracité d'une situation sans avoir à divulguer les informations associées. Son objectif est de préserver la confidentialité des données personnelles des utilisateurs et un cas d'usage type est de prouver l'identité d'un utilisateur sans avoir à la révéler.

Annexe – Membres et participants au groupe de travail

- Autorité de contrôle prudentiel et de résolution (ACPR) :
  - Olivier Fliche
  - Laurent Camus
  - Julien Uri
  - Mathieu Vileyn
  - Hatim Tapory
  - Henriette Richard
  - Alexis Huz

- Autorité des marchés financiers (AMF) :
  - Charles Moussy
  - Olivier Brochand
  - Victor Faugeras
  - Théo Leguebe

- Banque de France :
  - Frédéric Faure
  - Béranger Butruille
  - David Zhang
  - Tran Huynh

- Agence nationale de la sécurité des systèmes d'information (ANSSI) :
  - Raphael Kenigsberg
  - Giacomo Martinelli
  - Elyes Lehtihet

- DG Trésor :
  - Rodolphe Baroukh
  - David Sabban

- Association pour le développement des actifs numériques (ADAN) :
  - Faustine Fleuret
  - Mélodie Ambroise
  - Alizée Van Den Schrieck
  - *Comité DeFi :*
    - Marc Zeller (AAVE Chan initiative)
    - Alexis Masseron (Atlendis Labs)
    - Merlin Egalite (Morpho Labs)
  - *Comité juridique :*
    - Victor Charpiat
  - *Comité PSAN :*
    - Owen Simonin (Meria)
    - Thibaut Boutrou (Meria)

- BNP Paribas :
    - Julien Clausse
    - Erik Lenoir
    - Mickael Da Rocha

- Caisse des Dépôts et Consignations :
    - Bastien Voituriez

- CEA List :
    - Alvaro Garcia Perez

- Crypto4All :
    - Christophe Ozcan

- Deloitte :
    - Marie-Line Ricard
    - Jérémy Stevance
    - Tantely Andriamalala

- Dowsers :
    - Patrick Azzopardi
    - Hugo de sa Pereira Pinto

- Exaion groupe EDF :
    - Gilles Deleuze

- Fédération française des professionnels de la blockchain (FFPB) :
    - Rémy Ozcan

- Fever Tokens :
    - Zakaryae Boudi

- Functori :
    - Çağdaş Bozman

- Gide Loyrette Nouel
    - Matthieu Lucchesi

- INRIA :
    - Daniel Augot

- Institut Louis Bachelier :
    - Louis Bertucci

- KPMG :
    - Guy Posbic

- Kramer Levin Naftalis & Frankel :
    - Hubert de Vauplane
    - Victor Charpiat

- Mazars :
    - Baptiste Pella

- Nomadic Labs :
    - Cédric Roche

- Société Générale Forge :
    - Annabelle Bernal
    - Sylvain Prigent

- Université Paris Saclay :
    - Sylvain Conchon