Elrond

Une chaîne de blocs publique hautement scalable, grâce au partage d'états adaptatif et à un mécanisme sécurisé de consensus par preuve d'enjeu.

[Livre Blanc technique — release 2 — revision 1]
Chapitres mis à jour : [5 - 13]
le 19 juin 2019 - L'équipe Elrond
https ://www.elrond.com/

Traduction&relecture de la version française :
Pascal Landeau (@poussette), Yan Delcroix (@kryptchioyaya), Jean-Noël Schilling (@Jnxmas), Clément A. (@MyRayke), Guillaume Gemot (@guillaumerd)

***Résumé*—L'avènement de la chaîne de blocs publique sécurisée avec Bitcoin puis Ethereum, a suscité un intérêt notable et l'afflux de capitaux, posant les prémisses d'une vague mondiale d'innovation ouverte à tous. Malgré de grandes promesses, la création d'une chaîne de blocs décentralisée, sécurisée et évolutive s'est révélée être une tâche ardue.**

Ce document est consacré à Elrond, une nouvelle architecture qui transcende l'état de l'art en introduisant un véritable mécanisme de partage d'états à travers un partitionnement horizontal (*State Sharding*) clé de voute d'une scalabilité opérationnelle, éliminant les gaspillages d'énergie et de calcul tout en garantissant une équité distribuée grâce à un consensus par preuve d'enjeu sécurisée (*Secure Proof of Stake - SPoS*). En mettant l'accent sur la sécurité, le réseau d'Elrond est construit pour résister aux problèmes de sécurité connus comme l'attaque Sybil, l'attaque "Rien à perdre" et d'autres. Dans un écosystème qui prône l'interconnectivité, notre solution pour implémenter les contrats intelligents (*Smart Contracts*) offre un moteur compatible avec EVM afin d'inscrire l'interopérabilité dans son ADN.

Les simulations préliminaires et les résultats du testnet montrent qu'Elrond dépasse le débit moyen de Visa en l'améliorant d'un facteur supérieur à 3, ou, d'un facteur 1000 par rapport aux approches viables existantes, tout en réduisant considérablement les coûts d'amorçage et de stockage pour assurer la durabilité à long terme.

I Introduction

1 Aspects généraux

Les plates-formes de crypto-monnaie et de contrats intelligents (*Smart Contracts*) telles que Bitcoin et Ethereum ont suscité un intérêt considérable et sont devenues des solutions prometteuses pour les paiements électroniques, les applications décentralisées et les réserves numériques potentielles de valeur. Toutefois, en comparaison des indicateurs clés de leurs homologues centralisés [1], l'état actuel des choses suggère que les versions actuelles des Chaînes de blocs publiques présentent de sérieuses limitations, notamment en ce qui concerne la scalabilité, ce qui fait obstacle à leur adoption par le grand public et en retarde son utilisation. En fait, il s'est avéré extrêmement difficile de gérer les limites techniques actuelles imposées par les compromis du paradigme du trilemme des Chaînes de blocs[2]. Plusieurs solutions ont été proposées, mais peu d'entre elles ont donné des résultats significatifs et viables. Ainsi, pour résoudre le problème de la scalabilité, il a fallu repenser complètement les infrastructures de la chaîne de blocs publique.

2 Définir les défis

Plusieurs défis doivent être correctement relevés dans le processus de création d'une solution innovante de chaîne de blocs publique conçue pour pouvoir passer à l'échelle :

- **Décentralisation complète** - Élimination du besoin de tout tiers de confiance, supprimant ainsi tout point de défaillance unique ;
- **Sécurité robuste** – Permettant des transactions sécurisées et empêchant toute attaque basée sur des vecteurs connus ;
- **Haute scalabilité** - Permettre au réseau d'atteindre une performance en TPS (Transactions par seconde) au moins égale à son homologue centralisé ;
- **Efficacité** - Exécution de tous les services réseaux avec des exigences énergétiques et informatiques minimales ;
- **Amorçage et amélioration du stockage** - Assurant un coût compétitif pour le stockage et la synchronisation des données ;
- I**nteropérabilité entre chaînes** - renforcée nativement à la conception, permettant une communication illimitée avec les services externes.

À partir des défis ci-dessus, nous avons créé Elrond en repensant complètement l'infrastructure de chaîne de blocs publique, spécialement conçue pour être sécurisée, efficace et interopérable. La principale contribution d'Elrond repose sur 2 piliers fondamentaux de construction :

1) **Un véritable mécanisme de partage d'états** *(State Sharding)* : partitionner efficacement la chaîne de blocs et partager l'état des comptes en plusieurs fragments, gérés en parallèle par différents valideurs participants ;

2) **Un mécanisme de consensus sécurisé par preuve d'enjeu** : une variante améliorée de la preuve d'enjeu *(Proof of Stake - PoS*) qui garantit la sécurité à long terme et l'équité distribuée, tout en éliminant le besoin d'algorithmes PoW énergivores.

3 partage d'états adaptatif

Elrond propose une solution au problème ci-dessous, mais, au préalable, quelques notions doivent être définies : utilisateurs et nœuds. Les utilisateurs sont des acteurs externes et peuvent être identifiés par une adresse de compte unique ; les nœuds sont des ordinateurs/terminaux du réseau Elrond qui exécutent notre protocole. Des notions telles que les utilisateurs, les nœuds et les adresses seront décrites plus en détail dans la section II.1 Entités. Elrond résout ce défi en :

1) Divisant l'espace d'adressage des comptes dans les fragments, en utilisant un arbre binaire qui peut être construit avec, comme seule exigence, la connaissance exacte du nombre de fragments à une époque donnée. Cette méthode permet de réduire la latence accumulée et d'améliorer l'efficacité du réseau de deux manières. Premièrement, grâce au modèle ainsi conçu, la division de l'espace d'adressage des comptes est prédéterminée par la hiérarchie. Ainsi, il n'y a pas de surcoût lors du partitionnement, ce qui signifie que lorsqu'un fragment se divise en deux, chacun d'entre eux ne conserve que la moitié de l'espace d'adressage précédent en plus de l'état associé. Ensuite, la latence est réduite grâce au mécanisme de redondance d'états, car la fusion est préparée en conservant l'état au sein des nœuds frères.
2) Introduisant une technique d'équilibrage des nœuds dans chaque fragment, afin d'obtenir un équilibre global de l'architecture. Cette technique assure une charge de travail équilibrée et une récompense pour chaque nœud du réseau.
3) Concevant un mécanisme intégré pour l'acheminement automatique des transactions dans les fragments correspondants, ce qui réduit considérablement la latence. L'algorithme de routage est décrit au chapitre IV.4 Approche d'Elrond sur le partitionnement horizontal.
4) Afin d'obtenir des améliorations considérables sans pénaliser l'initialisation au démarrage et le stockage, Elrond utilise un mécanisme d'élagage des fragments. Cela assure la durabilité de notre architecture, même avec un débit de dizaines de milliers de transactions par seconde (TPS).

4 Preuve d'enjeu sécurisée (SPoS ou Secure Proof of Stake)

Nous introduisons un mécanisme qui sécurise le consensus à preuve d'enjeu (Proof of Stake – PoS), qui va au-delà de l'idée d'Algorand [3] d'un mécanisme de sélection aléatoire, en se différenciant par les aspects suivants :

1) Elrond introduit une amélioration qui réduit la latence en permettant à chaque nœud dans le fragment de déterminer les membres du groupe de consensus (promoteurs de bloc et valideurs) dès le début du tour. Cela est possible parce que le facteur de répartition aléatoire r est stocké dans chaque bloc et est créé par le promoteur de bloc en utilisant une signature BLS [4] sur le r précédent.
2) Le promoteur du bloc est le valideur dans le groupe de consensus dont le résultat du hachage de la clé publique et du facteur de répartition aléatoire est le plus petit. Contrairement à l'approche d'Algorand [3], où la sélection aléatoire des participants peut prendre jusqu'à 12 secondes, avec Elrond, le temps nécessaire à la sélection aléatoire du groupe de consensus est considérablement réduit (estimé à moins de 100 ms), hors latences réseau. En effet, il n'y a pas d'exigence de communication pour ce processus de sélection aléatoire, ce qui permet à Elrond de disposer d'un groupe nouvellement et aléatoirement sélectionné qui aboutira à la validation d'un nouveau bloc dans le registre à chaque tour. Cette amélioration implique un compromis qui repose sur le principe qu'un adversaire ne peut pas s'adapter plus vite que la chronologie du cycle et peut choisir de ne pas proposer le bloc. Une autre amélioration de la sécurité de la source aléatoire serait l'utilisation de fonctions retard vérifiables (FRV) afin d'empêcher toute possibilité de falsification de la source aléatoire jusqu'à ce qu'il soit trop tard. Actuellement, la recherche sur les FRV est toujours en cours - il n'existe que quelques implémentations FRV fonctionnelles (et mal testées).
3) En plus du facteur d'enjeu généralement utilisé dans les architectures de PoS comme unique élément de décision, Elrond affine son mécanisme de consensus en ajoutant un facteur de pondération supplémentaire appelé notation. La probabilité que le nœud soit sélectionné dans le groupe de consensus prend en considération à la fois l'enjeu et la notation. La notation d'un promoteur de bloc est recalculée à la fin de chaque époque, sauf dans les cas où des coupes devraient se produire, où la dégradation réelle de la notation est faite instantanément, ajoutant une autre couche de sécurité en promouvant la méritocratie.
4) Un système modifié de signatures multiple du BLS [5] avec 2 cycles de communication est utilisé par le groupe de consensus pour la signature des blocs
5) Elrond envisage une vérification formelle pour les implémentations critiques du protocole (par exemple, le mécanisme de consensus SPoS) afin de valider l'exactitude de nos algorithmes.

II Aperçu de l'architecture

1 Entités

Il y a deux entités principales dans Elrond : les utilisateurs et les nœuds. Les utilisateurs détiennent tous un nombre (fini) de paires de clés (Pk/sk) publiques / privées (par exemple dans une ou plusieurs applications de portefeuille). Ils utilisent le réseau Elrond pour déployer des transactions

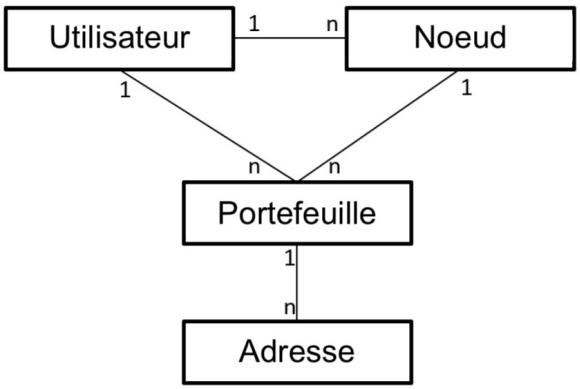

FIGURE 1: Relations entre les entités Elrond

signées, afin de transférer de la valeur, ou pour exécuter des contrats intelligents (*Smarts Contracts*). Les utilisateurs sont identifiés par l'une de leurs adresses de compte (dérivées de la clé publique). Les nœuds sont représentés par les terminaux dont est constitué le réseau Elrond, et peuvent être engagés passivement ou activement dans le traitement des tâches. Les valideurs éligibles sont des participants actifs dans le réseau Elrond. Plus précisément, ils sont chargés de l'application du consensus, de l'ajout de blocs, du maintien du statut, en échange de quoi, ils perçoivent des récompenses. Chaque valideur éligible est identifié de manière unique par une clé publique dérivée à la fois de l'adresse où est déposé l'enjeu, et aussi de l'identifiant du nœud.

De plus, le réseau est divisé en unités plus petites appelées fragments (*Shards*). Un valideur éligible est attribué à un fragment sur la base d'un algorithme qui maintient les nœuds uniformément répartis à travers les fragments, en fonction du niveau dans l'arbre. Chaque fragement contient un groupe de consensus choisi au hasard. Quiconque propose un bloc est responsable de l'agrégation des transactions dans un nouveau bloc. Les valideurs sont chargés soit de rejeter, soit d'approuver le bloc proposé, de sorte à le valider et à l'inscrire définitivement dans la chaîne de blocs.

2 Jeton intrinsèque

La façon dont Elrond autorise l'accès pour l'utilisation de son réseau se fait par le biais de jetons d'utilité intrinsèque appelés Elronds, en abrégé ERD. Tous les coûts liés au traitement des transactions, à l'exécution des contrats intelligents (*Smart Contracts*) et aux récompenses pour les diverses contributions au réseau seront payés en ERD. Les références aux frais, aux paiements ou aux soldes sont supposées être en ERD.

3 Modèle de menace

Elrond s'appuie sur un modèle contradictoire byzantin, dans lequel au moins $\frac{2}{3}n+1$ des nœuds éligibles d'un fragment sont intègres. Le protocole autorise l'existence d'adversaires qui disposent d'un enjeu ou d'une bonne côte, qui puissent retarder ou envoyer des messages conflictuels, compromettre d'autres nœuds, comporter des bogues ou être de connivence entre eux, mais tant que $\frac{2}{3}n+1$ des valideurs éligibles dans un fragment sont intègres/non compromis, le protocole peut aboutir à un consensus. Le protocole présuppose que les adversaires sont hautement adaptables, sans pour autant pouvoir s'adapter plus rapidement que la durée d'un tour. La puissance de calcul d'un adversaire est limitée, c'est pourquoi les hypothèses cryptographiques consenties par le niveau de sécurité des primitives choisies sont prises de manière ferme à l'intérieur de la classe de complexité des problèmes pouvant être résolus par une machine de Turing en temps polynomial. Le réseau des nœuds intègres est supposé former un graphe bien connecté et la propagation de leurs messages se fait dans un temps borné Δ.

Prévention des vecteurs d'attaque :

1) **Attaques Sybil :** atténuées par le verrouillage de l'enjeu au moment de rejoindre le réseau. Ainsi, la génération des nouvelles identités présente un coût égal à celui de l'enjeu minimal ;
2) **Attaque « Rien à perdre » :** (*"Nothing at stake"*) supprimée de par le besoin en signatures multiples, en plus de celle du promoteur, et de par la réduction de l'enjeu. Comparée à la récompense par bloc, l'enjeu verrouillé découragera de tels comportements ;
3) **Attaques longue distance :** atténuées par notre mécanisme d'élagage, et par l'utilisation d'un groupe de consensus sélectionné au hasard à chaque tour (et pas uniquement d'un seul promoteur) ainsi que par le verrouillage de l'enjeu. De plus, notre algorithme de consensus pBFT garantit l'irrévocabilité ;
4) **Les attaques DDoS :** le groupe de consensus est échantillonné au hasard à chaque tour (quelques secondes) ; le faible délai rendant le DDoS presque impossible.

Les autres vecteurs d'attaque que nous avons pris en considération sont : attaque par prise de contrôle des fragments, censure des transactions, doubles dépenses, attaques par corruption, etc.

4 Chronologie

Dans le réseau d'Elrond, la chronologie est divisée en époques et en tours. Les époques ont une durée constante, fixée à un jour (valeur qui pourra évoluer au gré de l'architecture), à l'issue de laquelle la réorganisation et l'élagage des fragments sont déclenchés. Les époques sont ensuite divisées en tours, d'une durée fixe. Un nouveau groupe de consensus est sélectionné au hasard par fragment à chaque tour, qui peut valider au maximum un bloc dans le registre du fragment.

Les nouveaux valideurs peuvent rejoindre le réseau en verrouillant leur enjeu, comme présenté au chapitre V.2 - Preuve d'enjeu sécurisée. Ils sont ajoutés au pool de nœuds non attribués de l'époque courante e, sont inscrits sur la liste d'attente d'un fragment au début de l'époque $e+1$, mais ne peuvent devenir des valideurs éligibles pour participer au consensus et être récompensés, uniquement à l'époque $e+2$ suivante.

Les aspects relatifs à la chronologie sont détaillés dans la section IX.1 Division de la chronologie.

III Travaux connexes

Elrond a été conçu et inspiré par les idées d'Ethereum [6], Omniledger [7], Zilliqa [8], Algorand [4] et ChainSpace [9]. Notre architecture va au-delà de l'état de l'art et peut être considérée comme un enrichissement des modèles existants, améliorant les performances tout en se concentrant sur un meilleur équilibre entre sécurité, scalabilité et décentralisation.

1 Ethereum

Une grande partie du succès d'Ethereum [6] peut être attribuée à l'introduction de sa couche d'applications décentralisées par le biais d'EVM [10], Solidity [11] et Web3j [12]. Si les Dapps ont été l'une des caractéristiques essentielles d'Ethereum, la scalabilité s'est avérée une limitation criante. Des recherches considérables ont été menées pour résoudre ce problème, mais les résultats demeurent insignifiants jusqu'à présent. Pour autant, quelques améliorations prometteuses ont été proposées : Casper [13] prépare une mise à jour qui remplacera le consensus actuel concernant la preuve de travail (Proof of Work) par une preuve d'enjeu (Proof of Stake), tandis que les chaînes latérales basées sur Plasma et le partage (des états ou des transactions) devraient être disponibles dans un avenir proche, ce qui atténuera au moins partiellement le problème de scalabilité d'Ethereum [14].

Comparé à Ethereum, Elrond élimine les gaspillages énergétiques et informatiques des algorithmes de Preuves de Travail (PoW) en mettant en œuvre un consensus par preuve d'Enjeu Sécurisée (SPoS) tout en utilisant le parallélisme du traitement des transactions par fragments (*Shards*).

2 Omniledger

Omniledger [7] propose un nouveau registre distribué scalable horizontalement qui préserve la sécurité à long terme dans le cadre d'une exploitation ouverte à tous. Il garantit la sécurité et l'exactitude en utilisant un protocole public à caractère aléatoire résistant aux déviances pour choisir les grands fragments statistiquement représentatifs qui traitent les transactions. Pour valider des transactions à travers les fragments de manière atomique, Omniledger présente Atomix, un protocole performant de validation interfragments.

Le concept est un protocole de "verrouillage/déverrouillage" à deux phases, piloté par le client, qui garantit que les nœuds peuvent, soit engager complètement une transaction sur des fragments, soit obtenir des "preuves de rejet" pour interrompre et déverrouiller l'état affecté par des transactions partiellement achevées. Omniledger optimise également les performances grâce au traitement parallèle des transactions intrafragments, l'élagage du registre par le biais de blocs d'états signés collectivement et à la validation à faible latence "faire confiance mais vérifier" (trust-but-verify) pour les transactions de faible valeur. Le consensus utilisé dans Omniledger est une variante de BFT, appelée ByzCoinX, qui augmente les performances et la robustesse contre les attaques de déni de service.

Par rapport à Omniledger, Elrond propose une approche adaptative du partage d'états, une sélection aléatoire plus rapide du groupe de consensus et une sécurité améliorée en remplaçant l'ensemble des valideurs après chaque tour (quelques secondes) et non après chaque époque (1 jour).

3 Zilliqa

Zilliqa [8] est la première architecture de partage des transactions permettant au réseau mineur de traiter les transactions en parallèle et d'atteindre un débit élevé en divisant le réseau mineur en fragments. Plus précisément, sa conception permet un taux de transaction plus élevé à mesure que de nouveaux nœuds rejoignent le réseau. L'essentiel est de veiller à ce que les fragments traitent des transactions différentes, sans chevauchement et donc sans double dépense. Zilliqa utilise le pBFT [15] pour le consensus et la preuve de travail (*PoW*) pour établir les identités et prévenir les attaques de type Sybil.

Par rapport à Zilliqa, Elrond repousse les limites du partage en utilisant, non seulement le partage de transactions, mais aussi le partage d'états. Elrond élimine complètement le mécanisme de la preuve de travail (PoW) et utilise la preuve d'enjeu Sécurisée (*SPoS*) pour le consensus. Les deux architectures construisent leur propre moteur de contrat intelligent, mais Elrond vise non seulement la conformité EVM, de sorte que le contrat intelligent (*Smart Contract*) écrit pour Ethereum fonctionne de manière transparente sur notre Machine Virtuelle, mais vise également à réaliser l'interopérabilité entre les chaînes de blocs.

4 Algorand

Algorand [3] propose un registre public qui conserve la commodité et l'efficacité des systèmes centralisés, sans les inefficacités et les faiblesses des mises en œuvre décentralisées actuelles. Le dirigeant et l'ensemble des vérificateurs sont choisis au hasard, en fonction de leur signature appliquée à la valeur de la quantité du dernier bloc. Les sélectionnés sont à l'abri des manipulations et restent imprédictibles jusqu'au dernier moment. Le consensus repose sur un nouvel accord byzantin de transmission de messages qui permet à la communauté et au protocole d'évoluer sans bifurquer.

Par rapport à Algorand, Elrond n'a pas de chaîne de blocs unique, mais augmente le débit des transactions grâce aux fragments. Elrond améliore également l'idée de sélection aléatoire d'Algorand en réduisant le temps de sélection du groupe de consensus de plus de 12 secondes à moins d'une seconde, mais présume que les adversaires ne peuvent pas s'adapter au cours d'un tour.

5 Chainspace

Chainspace [9] est une plateforme de registres distribués pour un traitement haute-intégrité et transparent des transactions. Il utilise des contrats intelligents (Smart Contracts), agnostiques sur le plan linguistique et respectueux de la vie privée, pour l'extensibilité. L'architecture en fragments (Shards) garantit un débit de traitement des transactions linéairement scalable à l'aide de S-BAC, un nouveau un

protocole de validation atomique distribué qui garantit la cohérence et offre une grande vérifiabilité. Des fonctions de protection de la vie privée sont mises en œuvre par des techniques modernes de connaissance zéro (*Zero Knowledge*), tandis que le consensus est assuré par la BFT.

Par rapport à Chainspace, où le TPS diminue avec chaque nœud ajouté dans un fragment, l'approche d'Elrond n'est pas influencée par le nombre de nœuds dans celle-ci, car le groupe de consensus a une taille fixe. Un point fort de Chainspace est l'approche des contrats intelligents agnostiques au langage, tandis qu'Elrond se concentre sur la construction d'une couche d'abstraction pour la conformité EVM. Les deux projets utilisent des approches différentes pour les fragments d'états afin d'améliorer les performances. Cependant, Elrond va plus loin en anticipant le problème de la taille des Chaînes de blocs dans les architectures à haut débit et utilise un mécanisme d'élagage efficace. En outre, Elrond présente une plus grande résistance aux changements soudains de la population de nœuds et à la prise de contrôle malveillante des fragments en introduisant la redondance des fragments, une nouvelle fonctionnalité pour les chaînes de blocs fragmentées.

IV Scalabilité grâce au partage d'états adaptatif

1 Le partitionnement horizontal, ou, pourquoi partitionner la chaîne de blocs en fragments ?

A l'origine, le partitionnement horizontal était utilisé dans les bases de données en tant que méthode de distribution des données entre plusieurs machines. Cette technique de passage à l'échelle peut être appliquée à la chaîne de blocs pour partager les états et le traitement des transactions, de sorte à ce que chaque nœud ne traite qu'une fraction de toutes les transactions en parallèle avec les autres nœuds. Tant qu'il y a un nombre de nœuds suffisant, et qui vérifient chaque transaction afin que le système maintienne une fiabilité et une sécurité élevées, dans ces conditions, le découpage d'une chaîne de blocs donnée, en fragments lui permettra de traiter de nombreuses transactions en parallèle, et donc d'améliorer considérablement le débit des transactions ainsi que l'efficacité des traitements. La promesse de ce partage est de pouvoir augmenter le débit au fur et à mesure que le réseau de valideurs s'étend, c'est une propriété designée sous le nom de scalabilité horizontale.

2 Différents types de partitionnement horizontal

Une introduction complète et détaillée [16] met l'accent sur les trois principaux types de partage à travers un partitionnement horizontal : partage de réseaux, partage de transactions, et partage d'états. Le partage de réseaux gère la façon dont les nœuds sont regroupés en fragments et peut être utilisé pour optimiser la communication, car la propagation des messages à l'intérieur d'un fragment peut se faire beaucoup plus rapidement que la propagation à l'ensemble du réseau. C'est le premier défi dans chaque approche de partage et le mécanisme qui fait correspondre les nœuds aux fragments doit prendre en considération les attaques possibles d'un attaquant qui prend le contrôle d'un fragment spécifique. Le partage de transactions gère la manière dont les transactions sont mises en correspondance avec les fragments où elles seront traitées. Dans un système basé sur des comptes, les transactions pourraient être attribuées à des fragments en fonction de l'adresse de l'expéditeur. L'approche la plus difficile est celle du partage d'états.

Contrairement aux mécanismes de découpage en fragments décrits précédemment, où tous les nœuds stockent l'état complet, dans les Chaînes de blocs découpées horizontalement en fragments d'états, chaque fragment ne maintient qu'une partie de l'état. Chaque transaction traitant des comptes qui se trouvent dans des fragments différents, devrait échanger des messages et mettre à jour les états dans des fragments différents. Afin d'accroître la résilience aux attaques malveillantes, les nœuds des fragments doivent être re-mélangés de temps en temps. Cependant, le déplacement des nœuds entre les fragments introduit des coûts de synchronisation, qui correspond au temps nécessaire pour que les nœuds nouvellement ajoutés puissent télécharger le dernier état. Ainsi, il est impératif que seul un sous-ensemble de tous les nœuds soit redistribué à chaque époque, pour éviter les temps d'arrêt pendant le processus de synchronisation.

3 Tendances autour du partitionnement horizontal

Certaines propositions de partitionnement horizontal tentent de mettre en œuvre uniquement le partage de transactions [8] ou uniquement du partage d'états [17], ce qui augmente le débit de la transaction, soit en forçant chaque nœud à stocker beaucoup de données d'états, soit en étant un superordinateur [2]. Pourtant, plus récemment, au moins une revendication a été faite quant à la réussite d'un partage qui portait à la fois sur les transactions et les états, et ce, sans compromettre la puissance de stockage ou de traitement [13].

Mais le partitionnement horizontal pose de nouveaux défis, comme l'attaque par prise de contrôle d'un fragment unique, la communication entre les fragments, la disponibilité des données et la nécessité d'une couche d'abstraction qui masque les fragments. Toutefois, sous réserve que les problèmes susmentionnés soient correctement traités, le partage d'états apporte des améliorations globales considérables : le débit des transactions augmente de manière significative en raison du traitement parallèle des transactions et les frais de transaction seront considérablement réduits. Deux principaux critères largement considérés comme des obstacles se transformant en autant d'avantages et d'incitations à l'adoption généralisée de la technologie de chaîne de blocs.

4 Approche d'Elrond sur le partitionnement horizontal

Pour remédier à la complexité de la combinaison du partage de transactions avec le partage d'états, l'approche d'Elrond a été conçue avec les objectifs suivants en ligne de mire :

1) **Scalabilité sans affecter la disponibilité** : Augmenter ou diminuer le nombre de fragments devrait affecter de manière négligeable un petit nombre des nœuds qui sont

à proximité, sans provoquer de temps d'arrêt, ou de les minimiser tout en actualisant les états ;

2) **Envoi et traçabilité instantanée** : La découverte du fragment de destination d'une transaction doit être déterministe, triviale à calculer, éliminant le besoin de communication entre plusieurs tours ;
3) **Efficacité et adaptabilité** : Les fragments doivent être aussi équilibrés que possible à chaque instant.

4.1 Description de la méthode

Pour calculer un nombre optimal de fragments (*Shards*) N_{sh} dans l'époque e_{i+1} ($N_{sh,i+1}$), nous avons défini un coefficient de seuil pour le nombre de transactions dans un bloc, θ_{TX}. La variable $optN$ représente le nombre optimal de nœuds dans un fragment, ϵ_{sh} est un nombre positif et représente la fourchette de variation du nombre de nœuds d'un fragment. $totalN_i$ est le nombre total de nœuds (valideurs éligibles, nœuds dans les listes d'attente et nouveaux nœuds dans le pool de nœuds) sur toutes les fragments de l'époque e_i, tandis que $N_{TXB,i}$ est le nombre moyen de transactions dans un bloc sur tous les fragments de l'époque e_i. $N_{sh,i0}$ sera considéré comme valant 1.

Le nombre total de fragments $N_{sh,i+1}$ changera si le nombre de nœuds $totalN_i$ du réseau change et si l'utilisation de la chaîne de blocs en a besoin : si le nombre de nœuds augmente au-delà d'un seuil $nSplit$ d'une époque à l'autre et que le nombre moyen de transactions par bloc est supérieur au seuil de transactions par bloc $N_{TXB,i} > \theta_{TX}$ ou si le nombre de nœuds diminue en dessous d'un seuil $nMerge$ comme indiqué dans la fonction $ComputeShardsN$.

```
1: function ComputeShardsN(totalN_{i+1}, N_{sh,i})
2:     nSplit ← (N_{sh,i} + 1) * (optN + ε_{sh})
3:     nMerge ← (N_{sh,i} − 1) * a
4:     N_{sh,i+1} ← N_{sh,i}
5:     if (totalN_{i+1} > nSplit and N_{TXB,i} > θ_{TX}) then
6:         N_{sh,i+1} ← totalN_{i+1}/(optN + ε_{sh})
7:     else if totalN_{i+1} < nMerge then
8:         N_{sh,i+1} ← totalN_{i+1}/(optN)
9:     return N_{sh,i+1}
```

D'une époque à l'autre, il y a une probabilité que le nombre de fragments actifs change. Si cet aspect influence le nombre de fragments, n'importe qui peut calculer les deux masques $m1$ et $m2$, utilisés dans la distribution des transactions.

```
1: function ComputeM1andM2(N_{sh})
2:     n ← math.ceil(log_2 N_{sh})
3:     m_1 ← (1 << n) − 1
4:     m_2 ← (1 << (n − 1)) − 1
5:     return m_1, m_2
```

L'objectif principal étant d'augmenter le débit au-delà des milliers de transactions par seconde et de diminuer la communication entre les fragments, Elrond propose un mécanisme de répartition qui détermine automatiquement les fragments impliqués dans la transaction en cours et achemine la transaction en conséquence. L'expéditeur prendra en considération

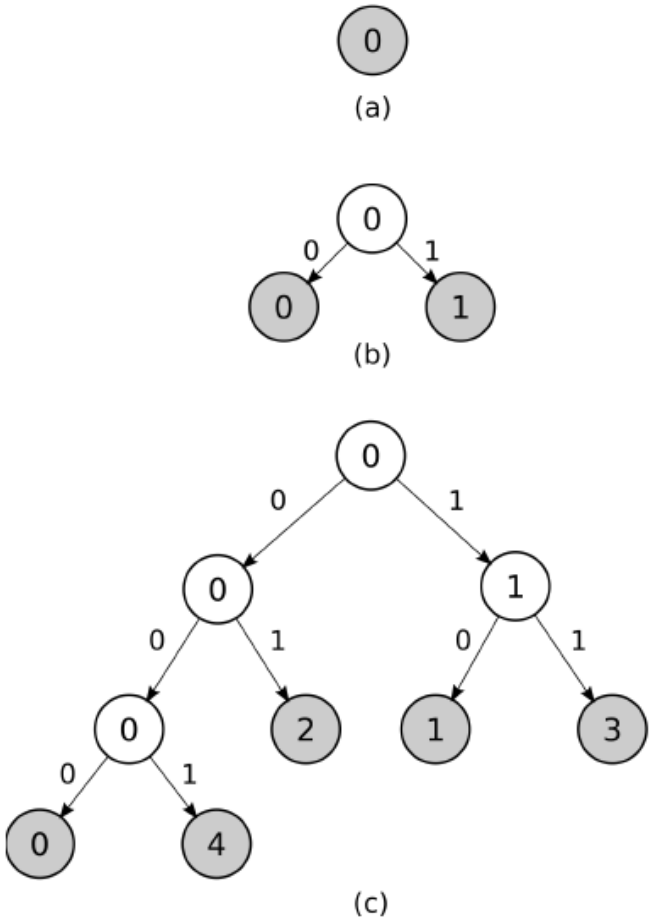

FIGURE 2: Exemple d'une structure arborescente de fragment

l'adresse de compte ($addr$) de l'expéditeur/récepteur de la transaction. Le résultat est le numéro de fragment ($shard$) auquel la transaction sera envoyée.

```
1: function ComputeShard(N_{sh}, addr, m_1, m_2)
2:     shard ← (addr and m_1)
3:     if shard > N_{sh} then
4:         shard ← (addr and m_2)
5:     return shard
```

L'ensemble du système de partage est basé sur un arbre binaire qui distribue les adresses des comptes, favorise la scalabilité et traite les transitions d'états. Une représentation de l'arbre est visible sur la figure 2.

L'arborescence présentée n'est qu'une représentation logique de l'espace d'adresse du compte utilisé pour une mise en correspondance déterministe ; par exemple, l'attribution des fragments, le calcul des nœuds frères, etc. Les feuilles de l'arbre binaire représentent les fragments avec leur numéro ID. En partant de la racine (nœud/fragment 0), s'il n'y a qu'un seule fragment/feuille (a), toutes les adresses de compte sont associées à celui-ci et toutes les transactions seront exécutées là. Plus loin, si la formule de N_{sh} impose la nécessité de 2 fragments (b), l'espace d'adressage sera divisé en parties égales, en fonction des derniers bits de l'adresse.

Parfois, l'arbre peut aussi se déséquilibrer (c) si N_{sh} n'est pas une puissance de 2. Ce cas n'affecte que les feuilles du 6ème niveau. La structure redeviendra équilibrée lorsque le nombre de fragments atteindra de nouveau une puissance de 2.

Lorsque l'arbre binaire n'est plus équilibré, ceci implique que les fragments situés au niveau le plus bas occupent la moitié de l'espace d'adressage des nœuds d'un fragment située au niveau du dessus, on peut donc affirmer que les nœuds actifs attribués à ces fragments percevront des revenus plus faibles - les récompenses de bloc ne sont pas affectées. Toutefois, ce problème est résolu en distribuant un tiers de chaque nœud de fragment de façon aléatoire à travers chaque époque (détaillée dans la section Division de la chronologie) et présentant une répartition équilibrée des nœuds selon le niveau de l'arbre.

En regardant l'arbre, à partir de n'importe quelle feuille et en progressant le long des branches en direction de la racine, l'encodage des branches correspond aux n derniers bits des adresses de compte qui feront traiter, par cette feuille/fragment, leurs transactions d'origine associées. En allant dans l'autre sens, de la racine vers la feuille, l'information est liée au développement de la structure, fragments frères, du fragment parent d'où ils se séparent. À l'aide de cette hiérarchie, le fragment qui se divise lorsque N_{sh} augmente ou les fragments qui fusionnent lorsque N_{sh} diminue peuvent être facilement calculés. L'ensemble du mécanisme de partage d'états bénéficie de cette structure en gardant toujours l'adresse et l'état associé dans le même fragment.

Connaissant N_{sh}, n'importe quel nœud peut suivre le processus de redistribution sans avoir besoin de communication. L'allocation des ID pour les nouveaux fragments est incrémentale et la réduction du nombre de fragments implique que les fragments les plus élevés seront supprimés. Par exemple, lorsque vous allez de N_{sh} à $N_{sh} - 1$, deux fragments seront fusionnés, le fragment à supprimer est le fragment numéroté le plus élevé ($sh_{merge} = N_{sh} - 1$). Trouver le numéro de fragment avec lequel sh_{merge} sera fusionné est trivial. Selon la structure de l'arbre, le fragment résultant aura le numéro du frère :

```
1: function ComputeSibling(sh_merge, n)
2:     sibling ← (sh_merge xor (1 << (n − 1)))
3:     return sibling
```

Pour la redondance des fragments, la traçabilité des transitions d'états et le passage à l'échelle rapide, il est important de déterminer le frère et le parent d'un fragment générique avec le nombre p :

***4.2** Redondance des fragments*

Sur la chaîne de blocs, le partage d'états est exposé aux échecs de fragments lorsqu'il y a un nombre insuffisant de nœuds en ligne dans un fragment ou que la distribution est concentrée géographiquement. Dans le cas peu probable où un fragment échoue (soit le fragment ne peut pas être contacté - tous les nœuds sont hors-lignes ou un consensus ne peut pas être atteint - plus d'un des nœuds ne répond pas), il y a un risque élevé que l'architecture entière ne repose que sur des nœuds super-pleins [2], qui téléchargent entièrement chaque

```
1: function ComputeParentSiblings(n, p, N_sh)
2:     mask_1 ← 1 << (n − 1)
3:     mask_2 ← 1 << (n − 2)
4:     sibling ← (p xor mask_1)
5:     parent ← min(p, sibling)
6:     if sibling ≥ N_sh then
7:         sibling ← (p xor mask_2)
8:         sibling_2 ← (sibling xor mask_1)
9:         parent ← min(p, sibling)
10:        if sibling_2 ≥ N_sh then                ▷ sibling est un fragment
11:            return parent, sibling, NULL
12:        else
13:                                  ▷ sibling est un sousarbre avec
14:                                  ▷ fragment (sibling, sibling_2)
15:            return parent, sibling, sibling_2
16:    else                                        ▷ sibling est un fragment
17:        return parent, sibling, NULL
```

bloc de chaque fragment, vérifiant l'intégralité. Comme le montre la figure 3, notre protocole dispose d'un mécanisme de protection qui introduit un compromis dans la structure de maintien de l'état en imposant aux fragment du dernier niveau de l'arbre de maintenir également l'état de leurs frères. Ce mécanisme réduit la communication et élimine l'amorçage lorsque des fragments fusionnent puisqu'ils ont déjà les données.

***4.3** Changement de contexte*

Pour préserver la sécurité dans les Chaînes de blocs publiques appliquant du partitionnement horizontal (*Sharding*), le changement de contexte devient crucial [7]. Cela se rapporte à la réaffectation des nœuds actifs entre les fragments sur un intervalle de temps fixe selon certains critères aléatoires. Dans l'approche d'Elrond, le changement de contexte améliore la sécurité, mais augmente également la complexité nécessaire pour maintenir la cohérence entre plusieurs états. La transition d'états laisse la plus grande empreinte sur les performances puisque le mouvement des nœuds actifs nécessite de resynchroniser l'état, la chaîne de blocs et les transactions en même temps que les nœuds éligibles dans le nouveau fragment. Au début de chaque époque, afin de maintenir le caractère temps réel, seul moins d'un de ces nœuds sera uniformément redistribué entre les fragments. Ce mécanisme est très efficace contre la formation de groupes malveillants

***4.4** Chaîne de notarisation (Meta)*

Toutes les opérations relatives au réseau et aux données globales (nœud rejoignant le réseau, nœud quittant le réseau, calcul des listes de valideurs éligibles, affectation des nœuds aux listes d'attente des fragments, accord par consensus sur un bloc dans un fragment spécifique, les contestations des blocs non valides) seront notariées dans la métachaîne. Le consensus de la métachaîne est géré par un fragment différent qui communique avec tous les autres fragments et facilite les opérations entre fragments. À chaque tour de chaque époque, la métachaîne reçoit les en-têtes de bloc des autres fragments et, si nécessaire, les preuves des contestations des blocs invalides. Ces informations seront regroupées en blocs

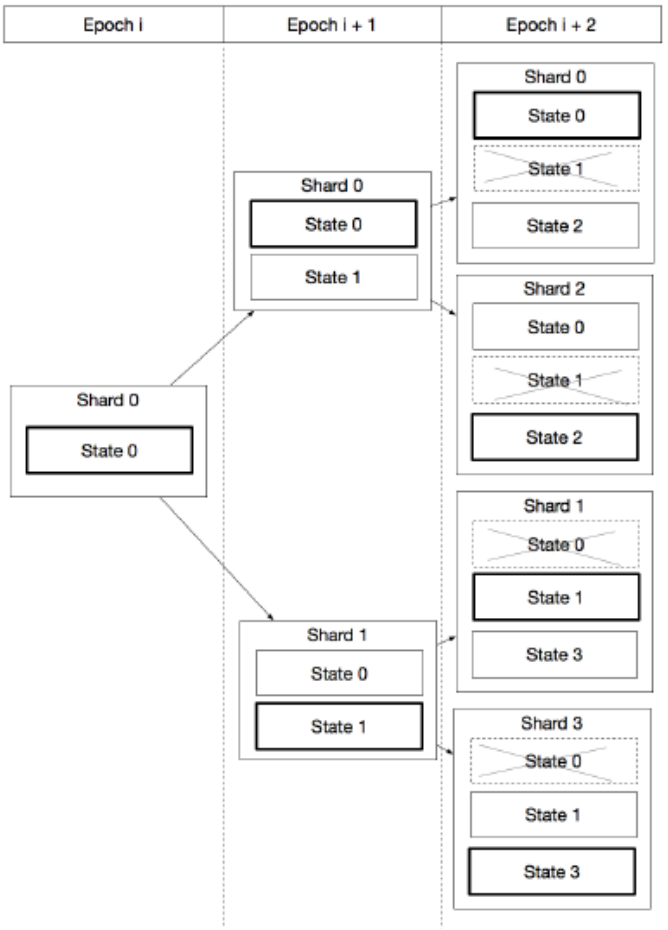

FIGURE 3: Redondance des fragments à travers les époques

sur la métachaîne sur lesquels un consensus doit être exécuté. Une fois les blocs validés dans le groupe de consensus, les fragments peuvent demander des informations sur les blocs, les miniblocs (voir chapitre VII), les valideurs éligibles, les nœuds des listes d'attente, etc. afin de traiter en toute sécurité les transactions entre fragments. De plus amples informations sur l'exécution des opérations croisées entre fragments, la communication entre les fragment et la métachaîne seront présentées au chapitre VII Traitement des transactions interfragments.

V Consensus par preuve d'enjeu sécurisée

1 Analyse des consensus

Le premier algorithme de consensus de chaîne de blocs basé sur la preuve de travail (*Proof of Work = PoW*), est utilisé dans Bitcoin, Ethereum et d'autres plates-formes de Chaînes de blocs. Dans la preuve de travail, chaque nœud est nécessaire pour résoudre un puzzle mathématique (difficile à calculer mais facile à vérifier). Et le premier nœud qui termine le puzzle recueille la récompense [18]. Les mécanismes de preuve de travail permettent d'éviter les doubles dépenses, les attaques DDoS et Sybil au prix d'une consommation d'énergie élevée.

La preuve d'enjeu (*PoS = Proof of Stake*) est un nouveau mécanisme de consensus et plus efficace proposé comme une alternative à la consommation intensive d'énergie et de puissance de calcul dans les mécanismes de consensus de preuve de travail (*PoW*). La PoS peut être trouvée dans de nombreuses nouvelles architectures comme Cardano [19] et Algorand[3] et peut-être dans la prochaine version d'Ethereum. Dans la Preuve d'Enjeu, le nœud qui propose le bloc suivant est sélectionné par une combinaison d'enjeux (richesse), de tirage aléatoire et/ou d'ancienneté. Cela atténue le problème de l'énergie de la preuve de travail mais pose également deux problématiques importantes sur la table : l'attaque "Rien à perdre" (*"Nothing at stake"*) et un risque de centralisation plus élevé.

La preuve de mémoire (*Proof of Meme*), telle qu'elle est envisagée dans Constellation [20], est un algorithme basé sur la participation historique du nœud au réseau. Son comportement est stocké dans une matrice de pondération dans la chaîne de blocs et prend en charge les changements au fil du temps. Il permet également aux nouveaux nœuds de gagner la confiance des utilisateurs en se forgeant une réputation. Le principal inconvénient des attaques Sybil est atténué par l'algorithme NetFlow.

La preuve d'autorité déléguée (*Delegated Proof of Stake - DPoS*), que l'on trouve dans Bitshares [21], Steemit [22] et EOS [23], est un hybride entre la preuve d'autorité et la preuve d'enjeu, dans lequel les quelques nœuds responsables du déploiement de nouveaux blocs sont élus par les parties prenantes. Bien que présentant un débit élevé, le modèle est sensible aux problèmes sociaux spécifiques à l'homme tels que la corruption. De plus, un petit nombre de délégués rend le système vulnérable aux attaques DDoS et à la centralisation.

2 Preuve d'enjeu sécurisée (SPoS = Secure Proof of Stake)

L'approche d'Elrond en matière de consensus consiste à combiner une sélection aléatoire des valideurs, une éligibilité à travers l'enjeu et une notation, avec une dimension optimale pour le groupe de consensus. L'algorithme est décrit dans les étapes ci-dessous :

1) Chaque nœud n_i est défini comme un tuple de clé publique (Pk), de la notation (par défaut 0) et de l'enjeu verrouillé. Si n_i souhaite participer au consensus, il doit d'abord s'enregistrer par le biais d'un contrat intelligent, en envoyant une transaction qui contient un montant égal à l'enjeu minimal requis et d'autres informations (Pk_s, une clé publique dérivée de Pk et $nodeid$ qui sera utilisé pour le processus de signature afin de ne pas utiliser une adresse réelle de portefeuille).
2) Le nœud n_i rejoint le pool de nœuds et attend l'affectation des fragments à la fin de l'époque e en cours. Le mécanisme d'affectation des fragments crée un nouvel ensemble de nœuds contenant tous les nœuds qui se sont joints à l'époque e et tous les nœuds qui doivent être re-mélangés (moins de $\frac{1}{3}$ de chaque fragment). Tous les nœuds de cet ensemble seront réaffectés aux listes d'attente de fragments. W_j représente la liste d'attente de fragments de j et N_{sh} représente le nombre de fragments. Un nœud possède également une clé secrète sk qui, par nature, ne doit pas être rendue publique.

$$n_i = (Pk_i, rating_i, stake_i)$$

$$n_i \in W_j, 0 \leq j < N_{sh}$$

3) À la fin de l'époque à laquelle il s'est joint, le nœud sera déplacé vers la liste des nœuds éligibles (E_j) d'un fragment j, où e est l'époque courante.

$$n_i \in W_{j,e-1} \rightarrow n_i \notin W_{j,e}, n_i \in E_{j,e}$$

4) Chaque nœud de la liste E_j peut être sélectionné dans le cadre d'un groupe de consensus dimensionné de manière optimale (en termes de sécurité et de communication), par une fonction déterministe, basée sur le caractère aléatoire de la source ajoutée au bloc précédent, le tour r et un ensemble de paramètres de variation. Le nombre aléatoire, connu de tous les nœuds du fragment par le biais d'un protocole de bavardage, ne peut être prédit avant que le bloc ne soit effectivement signé par le groupe de consensus précédent. Cette propriété en fait une bonne source aléatoire et empêche les attaques malveillantes hautement adaptatives. Nous définissons une fonction de sélection pour renvoyer l'ensemble des nœuds choisis (groupe de consensus) N_{chosen}, le premier étant le promoteur du bloc, qui prend les paramètres suivants : E, r et sig_{r-1} - la signature du bloc précédent.

$$N_{chosen} = f(E, r, sig_{r-1}), where N_{chosen} \subset E$$

5) Le bloc sera créé par le promoteur de bloc et les valideurs le cosigneront sur la base du protocole "practical Byzantine Fault Tolerance" (PbFt).

6) Si, pour une raison quelconque, le promoteur de bloc n'a pas créé de bloc pendant le créneau horaire qui lui était attribué (malveillant, hors ligne, etc.), le tour r sera utilisé en conjugaison avec la source aléatoire du dernier bloc pour sélectionner un nouveau groupe de consensus.

Si le promoteur du bloc courant agit de manière malveillante, les autres membres du groupe émettront un retour négatif pour dégrader son classement, diminuant ou même annulant les chances que ce nœud particulier soit à nouveau sélectionné. La fonction de rétroaction pour le promoteur de bloc (n_i) dans le tour numéro r, avec le paramètre $ratingModifier \in Z$ est calculée comme suit :

$$feedbackfunction = ff(n_i, notationModificateur, r)$$

Lorsque $RatingModifier < 0$, une coupure se produit de sorte que le nœud n_i perde sa mise.

Le protocole de consensus reste sûr face aux attaques DDoS dans la mesure où il peut s'appuyer sur un nombre élevé de valideurs possibles existants dans la liste E (des centaines de nœuds) et dans la mesure où il n'y a aucun moyen de prédire l'ordre des valideurs avant qu'ils ne soient sélectionnés.

Pour réduire le coût additionnel de communication qui accompagne un nombre accru de fragments, un consensus sera exécuté sur un bloc composite. Ce bloc composite est formé par :

— **Le bloc registre :** le bloc à ajouter dans le registre du fragment, comportant toutes les transactions intrafragments ainsi que toutes les transactions interfragments pour lesquelles une preuve de confirmation a été reçue ;
— **Mini-blocs multiples :** chacun d'eux détenant des transactions interfragments pour un fragment différent ;

Le consensus ne sera exécuté qu'une seule fois, sur le bloc composite contenant les transactions à la fois intrafragments et interfragments. Une fois le consensus obtenu, l'en-tête de bloc de chaque fragment est envoyé à la métachaîne pour notarisation

VI La couche cryptographique

1 Analyse des signatures

Les signatures numériques sont des primitives cryptographiques utilisées pour assurer la sécurité de l'information en offrant plusieurs propriétés comme l'authentification du message, l'intégrité des données et la non-répudiation [24].

La plupart des schémas utilisés pour les plates-formes de chaînes de blocs existantes reposent sur le problème du logarithme discret (LD) : fonction d'exponentiation unidirectionnelle $y \rightarrow \alpha^y mod\ p$. Il est scientifiquement prouvé que le calcul du logarithme discret avec base est difficile [25]. La cryptographie à courbe elliptique (ECC) utilise un groupe cyclique de points au lieu d'un groupe cyclique d'entiers. Ce système réduit l'effort de calcul, de sorte que pour des longueurs de clé de seulement 160 à 256 bits, l'ECC offre le même niveau de sécurité que RSA, Elgamal, DSA que d'autres fournissent pour des longueurs de clé de 1024 à 3072 bits (voir tableau 1 [24]).

La raison pour laquelle l'ECC offre un niveau de sécurité similaire pour des longueurs de paramètres beaucoup plus petites est que les attaques existantes sur des groupes de courbes elliptiques sont plus faibles que les attaques LD entières existantes, la complexité de ces algorithmes nécessitant une moyenne $\sqrt{p}$ des étapes de résolution. Cela signifie qu'une courbe elliptique utilisant un p premier de 256 bits offre en moyenne un niveau de sécurité de 2^{128} étapes nécessaires pour le briser [24].

Ethereum et Bitcoin utilisent tous deux la cryptographie sur les courbes, avec l'algorithme de signature ECDSA. La sécurité de l'algorithme est très dépendante du générateur de nombres aléatoires, car si le générateur ne produit pas un nombre différent à chaque requête, la clé privée peut être perdue [26].

Un autre système de signature numérique est l'EdDSA, une variante de Schnorr basée sur des courbes Edwards qui permettent un calcul rapide [27]. Contrairement à l'ECDSA, il est prouvé qu'il est non malléable, ce qui signifie qu'à partir d'une simple signature, il est impossible de trouver un autre ensemble de paramètres qui définisse le même point sur la courbe elliptique [28], [29]. En outre, l'EdDSA n'a pas besoin d'un générateur de nombres aléatoires car il utilise un nonce, calculé à partir du hachage de la clé privée et du message, ce qui permet d'éviter le vecteur d'attaque d'un générateur de nombres aléatoires brisé qui peut révéler la clé privée.

Les variantes de signature de Schnorr attirent de plus en plus l'attention [8], [30] en raison de leur capacité multi-signature native et de leur sécurité prouvée dans le modèle

de l'oracle aléatoire [31]. Un système multi-signature est une combinaison d'un algorithme de signature et de vérification, où plusieurs signataires, chacun avec ses propres clés privées et publiques, peuvent signer le même message, produisant une seule signature [32], [33] . Cette signature peut ensuite être vérifiée par un vérificateur qui a accès au message et aux clés publiques des signataires. Une méthode sous-optimale consisterait à demander à chaque nœud de calculer sa propre signature et de concaténer ensuite tous les résultats en une seule chaîne. Cependant, une telle approche est irréalisable car la taille de la chaîne générée augmente avec le nombre de signataires. Une solution pratique consisterait à agréger les résultats en une seule signature de taille fixe, indépendamment du nombre de participants. De tels systèmes ont été proposés à plusieurs reprises, la plupart d'entre eux étant susceptibles de faire l'objet d'attaques de type "rogue-key" (annulation). Une solution à ce problème serait d'introduire une étape où chaque signataire doit prouver la possession de la clé privée associée à sa clé publique [34].

Bellare et Neven [35] (BN) ont proposé un système sécurisé à signatures multiples sans preuve de possession, dans le modèle de clé publique en clair, sous l'hypothèse du logarithme discret [31]. Les participants valident d'abord leur part de R_i en propageant son hachage à tous les autres signataires afin qu'ils ne puissent pas calculer une fonction de celui-ci. Chaque signataire calcule une contestation (*Challenge*) différente pour sa signature partielle. Toutefois, ce système sacrifie l'agrégation de la clé publique. Dans ce cas, la vérification de la signature agrégée nécessite la clé publique de chaque signataire.

Un article récent de Gregory Maxwell et Andrew Poelstra [30] propose un autre schéma multi-signature dans le modèle de clé publique simple [36], sous l'hypothèse d'un "logarithme plus discret" (OMDL). Cette approche améliore le schéma précédent [35] en réduisant les cycles de communication de 3 à 2, réintroduisant l'agrégation de clés avec un coût de complexité plus élevée.

BLS [4] est un autre schéma de signature intéressant, issu de l'appariement de Weil, qui qui fait reposer sa sécurité sur l'hypothèse de la différence de calcul de Hellman sur certaines courbes elliptiques et génère des signatures courtes. Il possède plusieurs caractéristiques utiles comme la vérification par lots, l'agrégation de signatures, l'agrégation de clés publiques, ce qui fait du BLS un bon candidat pour les mécanismes de seuil et de signatures multiples.

Famille d'algo	**Système crypto**	**Niveau de sécurité (bit)**			
		80	**128**	**192**	**256**
Produit de facteurs premiers	RSA	1024	3072	7680	15360
Logarithme discret	DH, DSA, Elgamal	1024	3072	7680	15360
Courbes elliptiques	ECDH, ECDSA	160	256	384	512
Clé symétriques	AES, 3DES	80	128	192	256

TABLE 1: Longueurs (en bits) des algorithmes à clé publique pour différents niveaux de sécurité

Dan Boneh, Manu Drijvers et Gregory Neven ont récemment proposé un système multi-signature du BLS [5], utilisant des idées issues des travaux précédents [35], [30] pour fournir au système des défenses contre les attaques de clés malhonnêtes. Le système permet une vérification efficace avec seulement deux paires nécessaires pour vérifier une signature multiple et sans aucune preuve de la connaissance de la clé secrète (fonctionne dans le modèle de clé publique). Un autre avantage est que la multi-signature peut être créée en seulement deux cycles de communication.

Pour des raisons de traçabilité et de sécurité, un consensus basé sur un ensemble réduit de valideurs exige la clé publique de chaque signataire. Dans ce contexte, notre analyse conclut que le schéma multi-signature le plus approprié pour la signature en bloc à Elrond est le BLS multi-signature [5], qui est globalement plus rapide que les autres options en raison de seulement deux cycles de communication.

2 La signature de bloc dans Elrond

Pour la signature de blocs, Elrond utilise une cryptographie sur courbes basée sur le schéma multi-signature BLS appliqué au groupe bilinéaire $bn256$, qui met en œuvre l'appariement $\ll$ Optimal Ate $\gg$ sur une courbe Barreto Naehrig 256 bits. L'appariement bilinéaire est défini comme suit :

$$e : g_0 \times g_1 \rightarrow g_t \tag{1}$$

où g_0, g_1 et g_t sont des courbes elliptiques d'ordre premier p définies par $bn256$, et e est une carte bilinéaire (c'est-à-dire une fonction d'appariement). Soit G_0 et G_1 sont les générateurs de g_0 et g_1. Soit également H_0 une fonction de hachage qui produit des points sur la courbe g_0 :

$$H_0 : \mathcal{M} \rightarrow g_0 \tag{2}$$

où $\mathcal{M}$ est l'ensemble de tous les messages binaires possibles, quelle que soit leur longueur. Le système de signature utilisé par Elrond utilise également une seconde fonction de hachage, dont les paramètres sont connus de tous les signataires :

$$H_1 : \mathcal{M} \rightarrow Z_0 \tag{3}$$

Chaque signataire i a sa propre paire de clés privées/publiques (sk_i, Pk_i), où sk_i est choisi aléatoirement dans Zp. Pour chaque paire de clés, la propriété $Pk_i = sk_i.G_1$ est retenue.

Soit $L = Pk_1, Pk_2, ..., Pk_n$ l'ensemble des clés publiques de tous les signataires possibles au cours d'un tour spécifique qui, dans le cas d'Elrond, est l'ensemble des clés publiques de tous les nœuds du groupe de consensus. Les deux étapes du processus de signature en bloc sont présentées ci-dessous : la signature et la vérification.

2.1 Signature en pratique - Tour 1

Le dirigeant du groupe de consensus crée un bloc avec les transactions, puis signe et diffuse ce bloc aux membres du groupe de consensus.

***2.2** Signature en pratique - Tour 2*

Chaque membre du groupe de consensus (y compris le dirigeant) qui reçoit le bloc doit le valider, et s'il le reconnaît valide, il le signe avec le BLS et envoie ensuite la signature au dirigeant :

$$Sig_i = sk_i * H_0(m) \tag{4}$$

Où Sig_i est un point de g_0.

***2.3** Signature en pratique - Tour 3*

Le dirigeant du groupe attend de recevoir les signatures pendant une période déterminée. S'il ne reçoit pas au moins $\frac{2}{3} \cdot n + 1$ signatures dans ce délai, le cycle de consensus est avorté. Mais si le dirigeant reçoit $\frac{2}{3} \cdot n + 1$ ou plus de signatures valables, il les utilise pour générer la signature agrégée :

$$SigAgg = \sum_i H_1(Pk_i) \cdot Sig_i \cdot B[i] \tag{5}$$

Où $SigAgg$ est un point de g_0.

Le dirigeant ajoute ensuite la signature agrégée au bloc avec les signataires sélectionnés dans la bitmap B, où un 1 indique que le signataire correspondant dans la liste L a vu sa signature ajoutée à la signature agrégée $SigAgg$.

***2.4** Vérification pratique*

Compte tenu de la liste des clés publiques L, de la bitmap des signataires B, de la signature agrégée $SigAgg$ et d'un message m (bloc), le vérificateur calcule la clé publique agrégée :

$$PKAgg = \sum_i H_1(Pk_i) \cdot Pk_i \cdot B_i \tag{6}$$

Le résultat, $PkAgg$, est un point sur g_1. La vérification finale est

$$e(G_1, SigAgg) == e(PkAgg, H_0(m)) \tag{7}$$

où e est la fonction d'appariement.

VII Traitement des transactions interfragments

Pour un exemple approfondi de la manière dont les opérations interfragments sont exécutées et pour illustrer comment la communication entre les fragments et la métachaîne se déroule, nous simplifierons l'ensemble du processus pour ne mettre en jeu que deux fragments adossés à la métachaîne. En supposant qu'un utilisateur génère une transaction à partir de son portefeuille, qui a une adresse dans le fragment 0 et souhaite envoyer des ERD à un autre utilisateur qui a un portefeuille avec une adresse dans le fragment 1, les étapes décrites dans la figure 4 sont nécessaires pour traiter la transaction interfragments.

Comme mentionné dans le chapitre V Consensus par preuve d'enjeu sécurisée, la structure des blocs est représentée par un en-tête de bloc qui rassemble des informations sur le bloc (nombre ad-hoc *(nonce)* du bloc, tour, promoteur, horodatage des valideurs, etc.), et une liste de miniblocs de chaque fragment à l'intérieur de laquelle sont contenues les transactions en question. Chaque minibloc contient toutes les transactions qui ont soit l'émetteur dans le fragment actuel et le récepteur dans un autre fragment, soit l'émetteur dans un fragment différent et la destination dans le fragment actuel. Dans notre cas, pour un bloc dans le fragment 0, il y aura normalement 3 miniblocs :

- **minibloc 0 :** contenant les transactions intrafragments pour le fragment 0
- **minibloc 1 :** contenant les transactions interfragments avec l'expéditeur dans le fragment 0 et la destination dans le fragment 1
- **minibloc 2 :**contenant les transactions interfragments avec l'expéditeur dans le fragment 1 et la destination dans le fragment 0. Ces transactions ont déjà été traitées dans le fragment de l'expéditeur 1 et seront également finalisées après le traitement dans le fragment courant.

Il n'y a pas de limite sur le nombre de miniblocs avec le même expéditeur et le même récepteur dans un bloc. Ce qui signifie que plusieurs miniblocs avec le même expéditeur et le même récepteur peuvent apparaître dans le même bloc.

1 Traitements

Actuellement, l'unité atomique de traitement interfragments est le minibloc : soit toutes les transactions du minibloc sont traitées en même temps, soit aucune, et l'exécution du minibloc sera retentée au tour suivant.

Notre stratégie de transactions interfragments utilise un modèle asynchrone. La validation et le traitement sont effectués d'abord dans le fragment de l'expéditeur, puis dans le fragment du destinataire. Les transactions sont d'abord expédiées dans le fragment de l'expéditeur, car il peut valider entièrement toute transaction initiée à partir du compte dans ce fragment - principalement le solde courant. Ensuite, dans le fragment du récepteur, les nœuds n'ont besoin que de la preuve d'exécution offerte par la métachaîne, font la vérification de la signature et vérifient l'absence d'attaque par rediffusion et enfin mettent à jour le solde pour le récepteur, en ajoutant le montant de la transaction.

Le fragment 0 traite à la fois les transactions intrafragments dans le minibloc 0 et un ensemble de transactions interfragments qui ont des adresses du fragment 1 comme récepteur dans le minibloc 1. L'en-tête du bloc et les miniblocs sont envoyés à la métachaîne. La métachaîne notifie le bloc du fragment 0, en créant un nouveau bloc de métachaîne (metabloc) qui contient les informations suivantes sur chaque minibloc : ID du fragment émetteur, ID du fragment récepteur, hachage du minibloc.

Le fragment 1 récupère le hachage du minibloc 1 dans le métabloc, demande le minibloc du fragment 0, analyse la liste des transactions, requête les transactions manquantes (le cas échéant), exécute le même minibloc 1 dans le fragment 1 et envoie au bloc résultant de la métachaîne. Après la notarisation, l'ensemble des transactions interfragments peut être considéré comme finalisé.

Le diagramme suivant montre le nombre de tours nécessaires pour qu'une transaction soit finalisée. Les tours sont comptabilisés entre la première inclusion dans un minibloc et la notarisation du dernier minibloc.

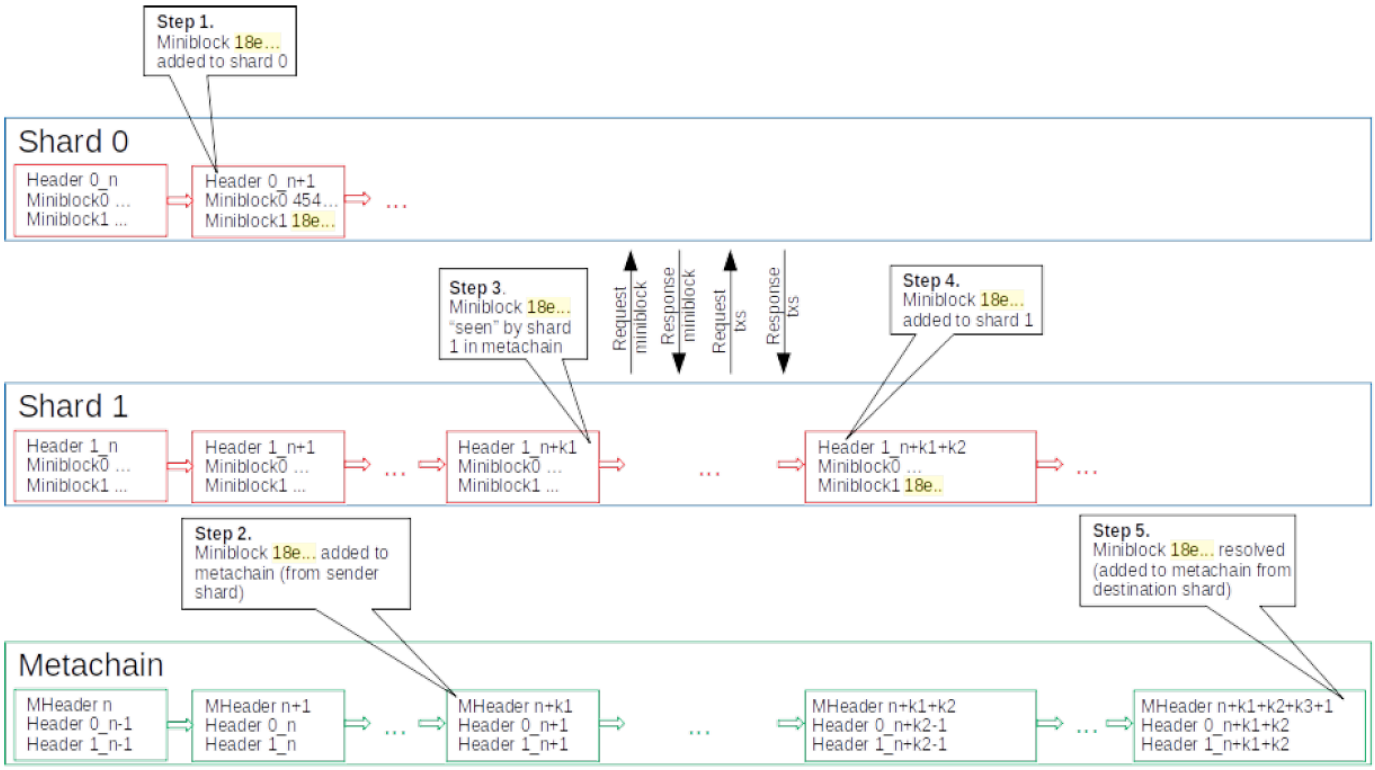

FIGURE 4: Traitement des transactions interfragments (*Shards*)

VIII Contrats intelligents (*Smart Contracts*)

L'exécution de contrats intelligents est un élément clé pour toutes les futures architectures de chaînes de blocs. La plupart des solutions existantes évitent d'expliquer correctement les dépendances entre les transactions et les données. Ce contexte conduit aux deux scénarios suivants :

1) Lorsqu'il n'y a pas de corrélation directe entre les transactions des contrats intelligents, comme le montre la figure 5, toute architecture peut utiliser un ordonnancement qui ne garantit pas l'ordre. Cela signifie qu'il n'y a pas de contraintes supplémentaires ni sur le moment, ni sur le lieu (fragment) où un contrat intelligent est exécuté.

2) Le second scénario fait référence au parallélisme induit par les transactions qui impliquent des contrats intelligents corrélés [37]. Ce cas, illustré à la figure 6, ajoute une pression supplémentaire sur les performances et augmente considérablement la complexité. Fondamentalement, il doit y avoir un mécanisme permettant de s'assurer que les contrats sont exécutés dans le bon ordre et au bon endroit (fragment). Pour couvrir cet aspect, le protocole Elrond propose une solution qui assigne et déplace le contrat intelligent vers le fragment même qui maintient ses dépendances statiques. De cette façon, la plupart, voire la totalité des appels aux contrats intelligents auront des dépendances réduites au même fragment et aucun verrouillage/déverrouillage interfragments ne sera nécessaire.

Elrond se concentre sur la mise en œuvre de la machine virtuelle Elrond, un moteur conforme à la norme EVM.

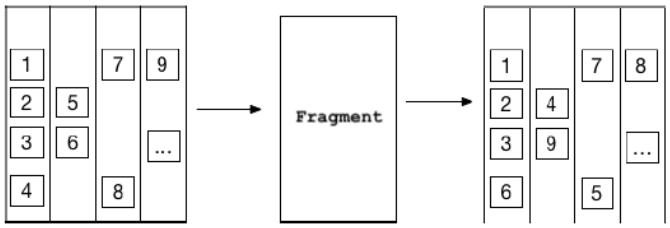

FIGURE 5: Traitement indépendant des transactions dans de contrats intelligents (Smart Contracts) simples qui peuvent être exécutés sans garantir l'ordre

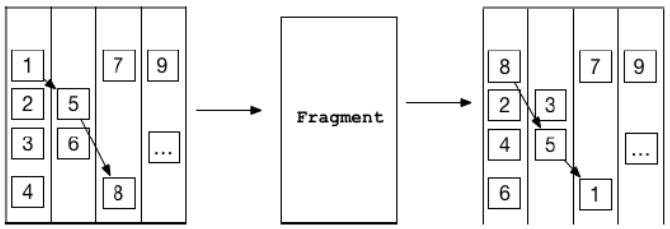

FIGURE 6: Mécanisme pour les contrats intelligents (Smart Contracts) corrélés qui peuvent uniquement être exécutés séquentiellement

En vue de l'adoption, la conformité à EVM est extrêmement importante, en raison du grand nombre de contrats intelligents construits sur la plateforme d'Ethereum.

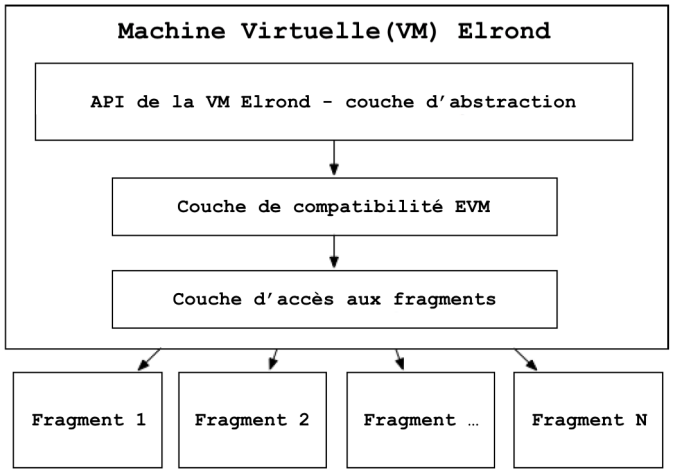

FIGURE 7: Couche d'abstraction pour les contrats intelligents

L'implémentation de la machine virtuelle Elrond masquera l'architecture sous-jacente isolant les développeurs de contrats intelligents des systèmes internes assurant ainsi une couche d'abstraction appropriée, comme illustré à la figure 7.

Dans Elrond, l'interopérabilité entre chaînes peut être mise en œuvre en utilisant un mécanisme d'adaptation au niveau de la machine virtuelle, comme le propose Cosmos [38]. Cette approche nécessite des adaptateurs spécialisés et un moyen de communication externe entre les adaptateurs de contrats intelligents pour chaque chaîne qui interopérera avec Elrond. L'échange de valeur sera opéré à l'aide de certains contrats intelligents spécialisés agissant en tant que dépositaires, aptes à assurer la garde des jetons natifs de la chaîne ainsi adaptée et d'émettre des jetons natifs Elrond.

1 infrastructure de machines virtuelles

Elrond construit son infrastructure de Machines Virtuelles au-dessus du Framework K, qui est un framework sémantique exécutable permettant de définir des langages de programmation, de calculs, ainsi que des systèmes de types ou des outils d'analyse formelle [39].

Le plus grand avantage de l'utilisation du Framework K c'est qu'avec celui-çi, les langages des contrats intelligents peuvent être définis de manière non-ambiguë, ce qui élimine le risque de comportements non spécifiés et de bogues difficiles à détecter.

Le Framework K est exécutable, en ce sens que les spécifications sémantiques des langages peuvent être utilisé directement comme des interpréteurs de langages opérationnels. Plus précisément, on peut soit exécuter des programmes en fonction des spécifications en utilisant directement l'implémentation cœur du Framework K, soit générer un interpréteur dans plusieurs langages de programmation différents. Ces derniers sont également appelés "backends". Pour des raisons de rapidité d'exécution et de facilité d'interopérabilité, Elrond utilise son propre backend du Framework K, qui est construit sur mesure.

2 Langages de contrats intelligents

Un grand avantage du framework K est que l'on peut générer un interpréteur pour n'importe quelle langue définie en K, sans avoir besoin de programmation supplémentaire. Cela signifie également que les interpréteurs produits de cette manière sont "corrects par construction"

Il existe déjà plusieurs langages de contrats intelligents spécifiés dans le Framework K, ou dont les spécifications sont en cours d'élaboration. Le réseau Elrond prendra en charge trois langages de bas niveau : IELE VM, KEVM et WASM.

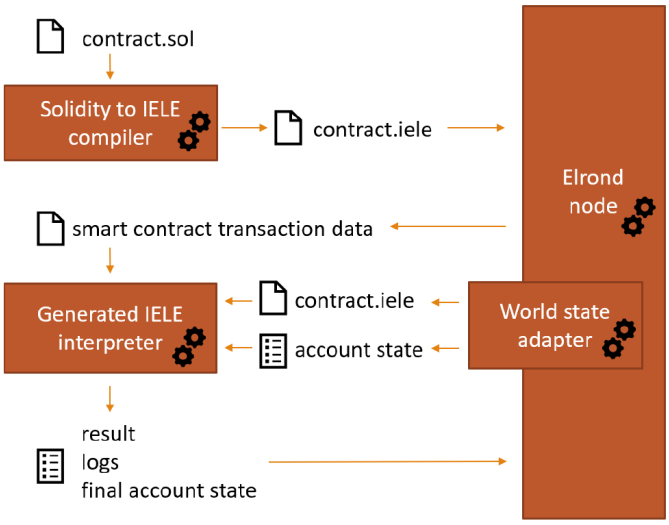

FIGURE 8: Exécution de la Machine Virtuelle (VM) Elrond

— IELE VM est un langage de niveau intermédiaire, dans le style de LLVM, mais adapté à la chaîne de blocs. Il a été construit directement en K, aucune autre spécification ou implémentation de celui-ci n'existe en dehors du framework K [40]. Son but est d'être humainement lisible, rapide, et d'être capable de surmonter certaines limitations de l'EVM. Elrond utilise une version légèrement modifiée d'IELE - la plupart des changements sont liés à la gestion des adresses des comptes. Les développeurs de contrats intelligents peuvent programmer directement en IELE, mais la plupart choisiront de coder en Solidity et d'utiliser ensuite un compilateur Solidity vers IELE, comme on peut le voir sur la figure 8.
— KEVM est une version de la machine virtuelle Ethereum (EVM), écrite en K [41]. Certaines vulnérabilités de l'EVM sont corrigées dans la version en K, et les fonctionnalités vulnérables sont entièrement écartées.
— Le Web Assembly (WASM) est un format d'instruction binaire, destiné aux machines virtuelles à mémoire de pile, qui peut être utilisé pour exécuter des contrats intelligents. Une infrastructure WASM permet aux développeurs d'écrire des contrats intelligents en C/C++, Rust, C#, et autres. Avoir une spécification linguistique et générer l'interpréteur n'est que la moitié du défi. L'autre moitié consiste à intégrer l'interprèteur généré au réseau Elrond. Nous avons construit une interface commune de VM, qui nous permet de connecter n'im-

porte quelle VM à un nœud Elrond, comme le montre la figure 9. Chaque VM dispose alors d'un adaptateur qui met en œuvre cette interface. Chaque contrat est enregistré comme bytecode de la VM pour laquelle il a été compilé et fonctionne sur sa VM correspondante.

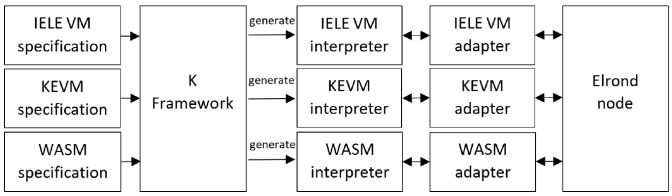

FIGURE 9: Exécution de la Machine Virtuelle (VM) Elrond

3 Assistance à la modélisation et à la vérification formelle

Comme les langages des contrats intelligents sont formellement définis dans le framework K, il est possible de procéder à une vérification formelle des contrats intelligents rédigés dans ces langages. Pour ce faire, il est nécessaire de modéliser formellement leurs exigences, ce qui peut également être fait en utilisant le framework K [42].

4 Contrats intelligents sur une architecture partitionnée horizontalement

Les contrats intelligents sur les architectures en fragments n'en sont encore qu'à leurs débuts et posent de sérieux problèmes. Des protocoles comme Atomix [7] ou S-BAC [9] représentent un point de départ. Les dépendances des contrats intelligents dynamiques ne peuvent pas être résolues en déplaçant les contrats intelligents dans le même fragment, car au moment du déploiement, toutes les dépendances ne peuvent pas être calculées.

Solutions en cours d'investigation sur la question :

1) Un mécanisme de verrouillage qui permet l'exécution atomique de contrats intelligents à partir de différents fragments, garantit que les contrats intelligents concernés seront, ou bien tous exécutés en même temps, ou alors aucun d'entre eux. Cela nécessite de multiples messages d'interaction et une synchronisation entre les consensus des différents fragments. [9]
2) La proposition de transfert de contrats interfragments pour Ethereum 2.0 permettrait de transférer le code et les données de ce contrat intelligent dans le fragment de l'appelant au moment de l'exécution. L'exécution atomique n'est pas nécessaire, mais le mécanisme de verrouillage est obligatoire sur le contrat intelligent déplacé, ce qui bloquerait l'exécution du contrat intelligent pour d'autres transactions. Le mécanisme de verrouillage est plus simple, mais il doit transférer l'ensemble de l'état interne du contrat intelligent. [43]

Suivant le modèle d'Ethereum, Elrond présente les types de transaction suivants :

1) Construction et déploiement de contrats intelligents : l'adresse du destinataire des transactions est vide et le champ de données contient le code du contrat intelligent sous forme de tableau d'octets ;
2) Méthode d'invocation du contrat intelligent : la transaction contient une adresse de destinataire non vide et cette adresse dispose d'un code associé ;
3) Transactions de paiement : la transaction contient un destinataire non vide et cette adresse ne dispose pas de code associé.

L'approche d'Elrond à ce problème est d'utiliser le modèle d'exécution asynchrone interfragments pour le cas de contrats intelligents. L'utilisateur crée une transaction d'exécution de contrat intelligent. Si le contrat intelligent n'est pas dans le fragment courant, la transaction est traitée comme une opération de paiement, la valeur de la transaction est soustraite du compte émetteur et elle est ajoutée au bloc où réside le fragment émetteur, dans un minibloc avec le fragment de destination où se trouve le compte récepteur. La transaction est notariée par la métachaîne, puis traitée par le fragment de destination. Dans le fragment de destination, la transaction est traitée comme une méthode d'invocation de contrat intelligent, car l'adresse du destinataire est un contrat intelligent qui existe dans ce fragment. Pour l'appel de contrat intelligent, un compte temporaire est créé, en tant qu'image du compte de l'expéditeur, avec le solde de la valeur de la transaction puis le contrat intelligent est appelé. Après l'exécution, le contrat intelligent peut renvoyer des résultats qui affectent un certain nombre de comptes de différents fragments.

Tous les résultats, qui affectent des comptes à l'intérieur d'un même fragment, sont exécutés au cours du même tour. Pour les comptes qui ne sont pas dans le fragment où le contrat intelligent a été exécuté, des transactions appelées "Résultats de contrats intelligents " (RCI) seront créées, sauvegardant le résultat de l'exécution du contrat intelligent pour chacun de ces comptes. Des miniblocs RCI sont créés pour chaque fragment de destination. Ces miniblocs sont notariés de la même manière que les transactions entre fragments par la métachaîne, puis traités par les fragments respectifs, où se trouvent les comptes. Si un contrat intelligent appelle dynamiquement un autre contrat intelligent à partir d'un autre fragment, cet appel est enregistré comme résultat intermédiaire et traité de la même manière que pour les comptes.

La solution se déroule en plusieurs étapes et la finalisation d'un appel de contrat intelligent entre fragments nécessitera au moins 5 tours, mais elle ne requiert pas de verrouillage ni de propagation d'états à travers les fragments.

IX Amorçage et stockage

1 Division de la chronologie

Les systèmes à preuve d'enjeu ont tendance à diviser la chronologie en époques et chaque époque en tours plus petits [19]. La chronologie et la terminologie peuvent varier d'une architecture à l'autre, mais la plupart d'entre elles utilisent une approche similaire.

1.1 Les époques

Dans le protocole Elrond, chaque époque a une durée fixe, initialement fixée à 24 heures (valeur susceptible d'évoluer

après quelques étapes de confirmation du testnet). Pendant cette période, la configuration des fragments reste inchangée. Le système s'adapte aux exigences de scalabilité entre les époques en modifiant le nombre de fragments. Pour éviter toute collusion, après une époque, la configuration de chaque fragment doit être modifiée. Un remaniement de tous les nœuds entre les fragments permettrait d'obtenir le niveau de sécurité le plus élevé, mais il affecterait le caractère temps-réel du système en introduisant une latence supplémentaire due à l'amorçage. C'est pourquoi, à la fin de chaque époque, moins de $\frac{1}{3}$ des valideurs éligibles appartenant à un fragment sera redistribué de manière non déterministe et uniforme sur les listes d'attente des autres fragments.

Ce n'est qu'avant le début d'une nouvelle époque que la distribution des valideurs aux fragments peut être déterminée, sans communication supplémentaire, comme le montre la figure 10. Le processus de remaniement pour mélanger les nœuds se déroule en plusieurs étapes :

1) Les nouveaux nœuds enregistrés à l'époque courante e_i atterrissent dans le pool de nœuds non attribués jusqu'à la fin de l'époque actuelle ;
2) Moins de $\frac{1}{3}$ des nœuds de chaque fragment sont sélectionnés aléatoirement pour être mélangés et sont ajoutés au groupe de nœuds attribué ;
3) Le nouveau nombre de fragments $N_{sh,i+1}$ est calculé sur la base du nombre de nœuds du réseau k_i et de l'utilisation du réseau ;
4) Les nœuds qui se trouvaient auparavant dans toutes les listes d'attente de fragments, qui sont actuellement synchronisés, sont ajoutés aux listes de valideurs éligibles ;
5) Les nouveaux nœuds ajoutés à partir du pool de nœuds non attribués sont répartis de manière aléatoire et uniforme sur toutes les listes d'attente de fragments pendant l'époque e_{i+1} ;
6) Les nœuds mélangés du pool de nœuds attribués sont redistribués avec des ratios plus élevés aux listes d'attente des fragments qui devront se séparer à la prochaîne époque e_{i+2}.

1.2 Les tours

Chaque tour a une durée fixe de 5 secondes (durée susceptible d'évoluer après plusieurs étapes de confirmation de testnet). A travers chaque tour, un nouveau bloc peut être produit dans chaque fragment par un ensemble de valideurs de bloc choisis aléatoirement (dont un promoteur un bloc). D'un tour à l'autre, l'ensemble est modifié en utilisant la liste des nœuds éligibles, telle que détaillée au chapitre IV

Comme décrit précédemment, la reconfiguration des fragments au sein des époques et la sélection arbitraire des valideurs au cours des cycles décourage la création de coalitions malhonnêtes, diminue la possibilité de DDoS et d'attaques de corruption tout en maintenant la décentralisation et un débit de transactions élevé.

2 Elagage

Un débit élevé conduira à un registre distribué qui croîtra rapidement en taille et augmentera le coût d'amorçage (temps+stockage), comme souligné dans la section XI XI.1.

Ce coût peut être compensé par l'utilisation d'algorithmes d'élagage efficaces, qui peuvent résumer l'état complet de la chaîne de blocs dans une structure plus condensée. Le mécanisme d'élagage est similaire aux points de contrôle stables de la pBFT [15] et compresse l'ensemble de l'état du registre.

Le protocole Elrond utilise un algorithme d'élagage performant [7] détaillé ci-dessous. Considérons que e est l'époque actuelle et que a est le fragment actuel :

1) les nœuds de fragment gardent une trace des soldes des comptes de e dans un arbre Merkle [44] ;
2) à la fin de chaque époque, le promoteur de bloc crée un bloc d'états $sb(a, e)$, qui stocke le hachage de la racine de l'arbre Merkle dans l'en-tête du bloc et les soldes dans le corps du bloc ;
3) les valideurs vérifient et exécutent le consensus sur $sb(a, e)$;
4) si un consensus est obtenu, le promoteur du bloc stockera sb(a, e) dans le registre du fragment, ce qui en fera le bloc de genèse pour l'époque $e + 1$;
5) à la fin de l'époque $e + 1$, les nœuds feront tomber le corps de sb(a, e) et tous les blocs précédents $sb(a, e)$.

Grâce à ce mécanisme, l'amorçage des nouveaux nœuds devrait être très performant. En fait, ils ne partent que du dernier bloc d'états valide et ne calculent que les blocs suivants au lieu de son historique complet.

X Évaluation de la sécurité

1 Source de nombres aléatoires

Elrond utilise des nombres aléatoires dans son fonctionnement, par exemple pour l'échantillonnage aléatoire des promoteurs de blocs et des valideurs des groupes de consensus et pour le mélange des nœuds entre les fragments à la fin d'une époque. Étant donné que ces fonctionnalités concourent aux garanties de sécurité d'Elrond, il est donc important d'utiliser des nombres aléatoires démontrés infalsifiables et imprédictibles. En plus de ces fonctionnalités, la génération de nombres aléatoires doit également être efficace afin qu'elle puisse être utilisée dans l'architecture d'une chaîne de blocs scalable et à haut débit.

Ces propriétés peuvent être trouvées dans certains schémas cryptographiques asymétriques, comme le schéma de signature BLS. Une propriété importante de BLS est que l'utilisation de la même clé privée pour signer le même message produit toujours les mêmes résultats. Ce résultat est similaire à celui obtenu avec l'ECDSA et sa génération déterministe k et il est dû au fait que le système n'utilise aucun paramètre aléatoire :

$$sig = sk \cdot H(m) \tag{8}$$

où H est une fonction de hashage qui hashe en points de la courbe utilisée et sk est la clé privée.

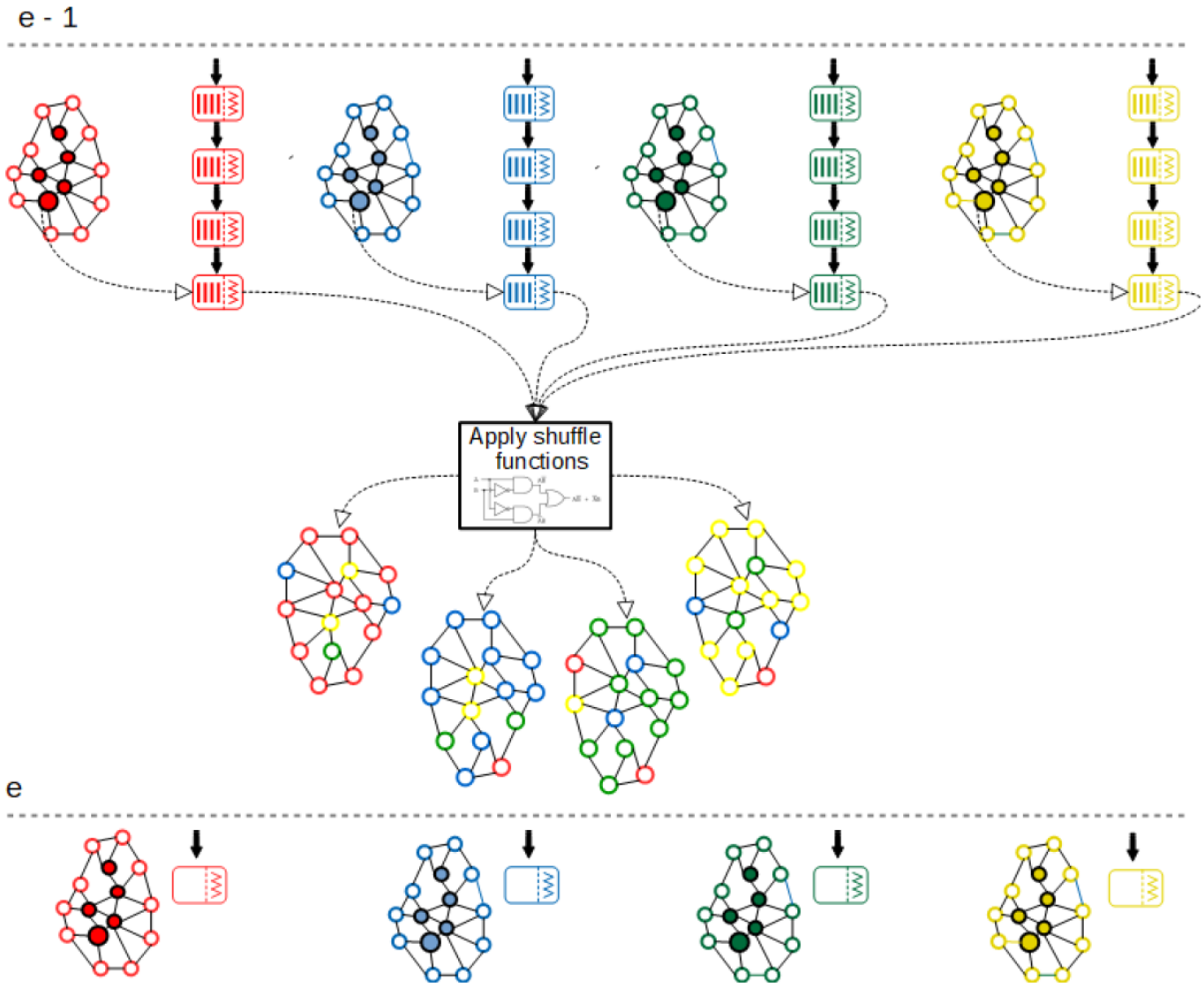

FIGURE 10: Remaniement des nœuds à la fin de chaque époque

2 Création de nombres aléatoires pour Elrond

Un nombre aléatoire est créé dans chaque tour, et ajouté par le promoteur de bloc à chaque bloc de la chaîne de blocs. Cela garantit que les nombres aléatoires sont impredictibles, car chaque nombre aléatoire est la signature d'un promoteur de bloc différent par rapport à la source aléatoire précédente. La création de nombres aléatoires est décrite ci-dessous dans le cadre d'un cycle de consensus :

1) Le nouveau groupe de consensus est sélectionné à l'aide de la source aléatoire de l'en-tête de bloc précédent. Le groupe de consensus est formé par un promoteur de bloc et des valideurs.
2) Le promoteur du bloc signe la source aléatoire précédente avec le BLS, ajoute la signature à l'en-tête du bloc proposé comme nouvelle source aléatoire, puis diffuse ce bloc au groupe de consensus.
3) Chaque membre du groupe de consensus valide la source aléatoire dans le cadre de la validation du bloc, et envoie sa signature de bloc au promoteur de bloc.
4) Le promoteur de bloc agrège les signatures de bloc des valideurs et diffuse le bloc avec la signature de bloc agrégée et la nouvelle source aléatoire à l'ensemble du fragment.

L'évolution aléatoire de la source dans chaque cycle peut être considérée comme une chaîne de blocs non falsifiable et vérifiable, où chaque nouveau nombre aléatoire peut être lié et vérifié par rapport au nombre aléatoire précédent.

3 Schéma de finalisation du bloc "K"

Le bloc signé au tour n est définitif, si et seulement si les blocs $n+1, n+2, ..., n+k$ sont signés. En outre, un bloc final ne peut pas être annulé. La métachaîne ne notarie que les blocs finaux afin de garantir qu'une bifurcation dans un fragment n'affecte pas les autres. Celles-ci ne prennent en considération que les blocs finaux de la métachaîne, afin de ne pas être affectés si elle se ramifie. La finalité et l'exactitude

sont vérifiées lors de la création et de la validation des blocs. Le paramètre k choisi est 1, ce qui garantit des bifurcations d'une longueur maximale de 2 blocs. La probabilité qu'une super majorité malveillante ($> \frac{2}{3} \cdot n + 1$) soit sélectionnée dans le fragment pour le même tour dans le même consensus est de 10^{-9}, même si 33% des nœuds du fragment sont malveillants. Dans ce cas, ils peuvent proposer un bloc et le signer - appelons-le *bloc m* -, mais il ne sera pas notarié par la métachaîne. La métachaîne ne notarie le *bloc m* que si le *bloc m+1* est construit par-dessus. Afin de créer le *bloc m+1*, le groupe de consensus suivant doit être d'accord avec le *bloc m*. Seul un groupe malveillant sera d'accord avec le *bloc m*, donc le groupe suivant doit à nouveau avoir une super majorité malveillante. Comme la séquence aléatoire pour la sélection du groupe ne peut être modifiée, la probabilité de sélectionner un autre groupe super majoritaire malveillant est de 10^{-9} ($5,38 \cdot 10^{-10}$, pour être exact). La probabilité de signer deux blocs malveillants consécutifs est égale à la sélection de deux sous-groupes comportant au moins ($> \frac{2}{3} \cdot n+1$) membres du groupe malveillant en conséquence. La probabilité pour cela est de 10^{-18}. En outre, les groupes sélectionnés en conséquence doivent être de connivence, sinon les blocs ne seront pas signés.

4 Le défi du pêcheur

Lorsqu'un bloc invalide est proposé par une majorité malveillante, la racine de l'état du fragment est altérée avec un résultat invalide (après avoir inclus des modifications invalides à l'arbre à états). En fournissant la preuve combinée de merkle pour un certain nombre de comptes, un nœud honnête pourrait soulever une contestation avec une preuve. Les nœuds honnêtes fourniront le bloc de transactions, l'arbre de merkle réduit précédent avec tous les comptes affectés avant d'appliquer le bloc contesté et les états de contrats intelligents, ce qui permet de démonter la transaction/état invalide. Si une contestation avec la preuve n'est pas fournie dans le délai imparti, le bloc est considéré comme valide. Le coût d'une contestation non valable correspond à la totalité de l'enjeu du nœud qui a soulevé la contestation.

La métachaîne détecte l'incohérence, soit une transaction non valide, soit une racine d'état non valide, grâce aux contestations et aux preuves présentés. Elle permet de remonter à la source de l'incohérence et de réduire le groupe de consensus. Dans le même temps, le contestataire peut être récompensé par une partie du montant réduit. Un autre problème se pose lorsqu'un groupe malveillant cache le bloc invalide à d'autres nœuds - non malveillants. Cependant, en rendant obligatoire, dans le cadre du consensus actuel, la propagation du bloc produit aux fragments frères et aux nœuds observateurs, les données ne peuvent plus être cachées. La surcharge de communication est encore réduite en n'envoyant que le minibloc intrafragment aux fragments frères. Les mini-blocs transversaux sont toujours envoyés sur différents sujets accessibles par les nœuds intéressés. En fin de compte, les défis peuvent être relevés par plusieurs nœuds honnêtes. La mise en place de sujets P2P constitue une autre mesure de sécurité. La communication d'un fragment vers la métachaîne se fait par un ensemble défini de sujets / canaux, qui peuvent être écoutés par n'importe quel valideur honnête - la métachaîne n'acceptera aucun autre message provenant d'autres canaux. Cette solution introduit un certain retard dans la métachaîne uniquement en cas de partitionnement, qui sont très peu nombreux et très improbables car, s'ils sont détectés (forte probabilité d'être détectés), les fragments risquent de perdre tout leur enjeu.

5 Mélange des fragments

Après chaque époque, moins de $\frac{1}{3} \cdot n$ des nœuds de chaque fragments sont redistribués de manière uniforme et non déterministe sur les autres fragments, afin d'éviter toute collusion. Cette méthode ajoute un surcoût d'amorçage pour les nœuds qui ont été redistribués, mais n'affecte pas le caractère temps réel car les nœuds remaniés ne participent pas au consensus de l'époque à laquelle ils ont été redistribués. Le mécanisme d'élagage diminuera cette fois-ci jusqu'à un montant réalisable, comme expliqué dans la section IX.2.

6 Sélection du groupe de consensus

Après chaque tour, un nouvel ensemble de valideurs est sélectionné en utilisant la graine (*seed*) aléatoire du dernier bloc validé, le tour en cours et la liste des nœuds éligibles. En cas de désynchronisation du réseau due à des retards dans la propagation des messages, le protocole dispose d'un mécanisme de récupération, et prend en considération à la fois le cycle r et la graine aléatoire du dernier bloc validé afin de sélectionner de nouveaux groupes de consensus à chaque cycle. Cela évite les bifurcations et permet la synchronisation sur le dernier bloc. La petite fenêtre de temps (temps d'un tour) dans laquelle les valideurs est connu, minimise les vecteurs d'attaque.

7 Notation des nœuds

Outre l'enjeu, la notation du valideur éligible influence les chances d'être sélectionné dans le cadre du groupe de consensus. Si le promoteur de bloc est honnête et que son bloc s'engage dans la chaîne de blocs, sa note sera augmentée, sinon, sa note sera diminuée. De cette façon, chaque valideur possible est incité à être honnête, à utiliser la version la plus récente du logiciel client, à augmenter sa disponibilité de service et à assurer ainsi que le réseau fonctionne comme prévu.

8 La redondance des fragments

Les nœuds qui ont été distribués en fragments frères au niveau le plus bas de l'arbre (voir section IV.4) gardent la trace des données de la chaîne de blocs et de l'état de l'application de chacun. En introduisant le concept de redondance des fragments, lorsque le nombre de nœuds dans le réseau diminue, certains des fragments frères devront être fusionnés. Les nœuds ciblés lanceront instantanément le processus de fusion des fragments.

XI Comprendre les vrais problèmes

1 Centralisé vs Décentralisé

La chaîne de blocs a été initialement instanciée comme alternative au système financier centralisé des systèmes [45]. Même si la liberté et l'anonymat des architectures distribuées demeurent un avantage incontesté, les performances doivent être analysées à l'échelle mondiale dans un environnement réel.

La mesure la plus pertinente de la performance est le nombre de transactions par seconde (TPS), comme le montre le tableau 2. Une comparaison des transactions par seconde entre les systèmes centralisés traditionnels et les nouvelles architectures décentralisées, dont la fiabilité et l'efficacité ont été validées à grande échelle, reflète une réalité objective mais troublante [46], [47], [48], [49].

La scalabilité des architectures des Chaînes de blocs est un problème critique mais toujours non résolu. Prenons, par exemple, l'exemple de la détermination des implications en matière de stockage des données et d'amorçage des architectures de chaîne de blocs actuelles qui se mettraient soudainement à fonctionner au même niveau de débit que Visa. En procédant à de tels exercices, l'ampleur des multiples problèmes secondaires devient évidente (voir la figure 11)

XII Le modèle de performance des blockchains

Le processus de conception d'architectures distribuées sur la chaîne de blocs est confronté à plusieurs défis, l'un des plus difficiles étant peut-être la nécessité de maintenir le fonctionnement dans des conditions de charge qui varient selon le contexte. Les principaux éléments qui conditionnent la charge sur les performances sont :

— la complexité
— la taille du système
— le volume des transactions

1 Complexité

Le premier facteur qui limite les performances du système est le protocole de consensus. Un protocole plus compliqué implique un point critique plus important. Dans les architectures à consensus par preuve de travail (*PoW*), une grande dégradation des performances est induite par la complexité du minage qui vise à maintenir le système décentralisé et la résilience des ASICs [50]. Pour surmonter ce problème, le consensus par preuve d'enjeu (*Proof of Stake - PoS*) apporte un compromis, simplifie la gestion du réseau en concentrant la puissance de calcul sur un sous-ensemble du réseau, mais apporte plus de complexité au mécanisme de contrôle.

2 Taille du système

L'augmentation du nombre de nœuds dans les architectures éprouvées existantes entraîne une grave dégradation des performances et induit un coût de calcul plus élevé qui doit être payé. Le partitionnement horizontal semble être une bonne approche, mais la taille des fragments joue un rôle majeur. Les petits fragments sont agiles, mais plus susceptibles d'être affectés par des groupes malveillants. Les gros fragments sont plus sûrs, mais leur reconfiguration affecte les caractéristiques temps-réel du système.

3 Volume de transaction

Plus pertinent que les autres, le dernier point sur la liste concerne la performance du traitement des transactions. Afin de mesurer correctement l'impact de ce critère, elle doit être analysée en tenant compte des deux points de vue suivants :

— Débit des transactions C1 - combien de transactions un système peut traiter par unité de temps, dit TPS, en sortie du système [51] ;
— la finalité de la transaction C2 - à quelle vitesse une transaction particulière est traitée, en se référant à l'intervalle entre son lancement et sa finalisation - depuis l'entrée vers la sortie.

C1. Le débit des transactions dans les architectures à chaîne unique est très faible et peut être augmenté en utilisant des solutions de contournement telles que la chaîne latérale (sidechain) [52]. Dans une architecture en fragments comme la nôtre, le débit des transactions est influencé par le nombre de fragments, les capacités de calcul des valideurs/promoteurs de blocs et l'infrastructure d'échanges de messages [8]. En général, comme le montre la figure 12, cela va bien au grand public, mais malgré l'importance de la métrique, cela ne donne qu'une vue incomplète.

C2. Finalité des transactions - Un aspect plus délicat qui souligne que même si le système est capable de traiter 1000 transactions par seconde, il peut prendre un certain temps pour traiter une transaction particulière. Outre les capacités de calcul des valideurs/promoteurs de bloc et l'infrastructure de messagerie, la finalité de la transaction est principalement affectée par l'algorithme de répartition (lorsque la décision est prise) et le protocole de routage (où la transaction doit être exécutée). La plupart des architectures de pointe existantes renoncent à mentionner cet aspect, mais du point de vue de l'utilisateur, c'est extrêmement important. La figure 13 montre le temps total nécessaire à l'exécution d'une transaction donnée, en prenant en compte le temps entre le début et la fin.

Avec Elrond, le mécanisme de répartition (détaillé dans la section V) permet un meilleur délai de finalisation en acheminant les transactions directement sur le bon fragment, ce qui atténue les retards globaux.

XIII Conclusion

1 Performance

Les tests de performance et simulations, présentés à la figure 14, témoignent de l'efficacité de la solution en tant que registre distribué. Au fur et à mesure que de que de nouveaux nœuds rejoignent le réseau notre méthode de partitionnement horizontal affiche une augmentation linéaire du débit. Le modèle de consensus choisi implique une communication à travers plusieurs tours, le résultat est donc fortement tributaire de la qualité du réseau (vitesse, latence, disponibilité). Les

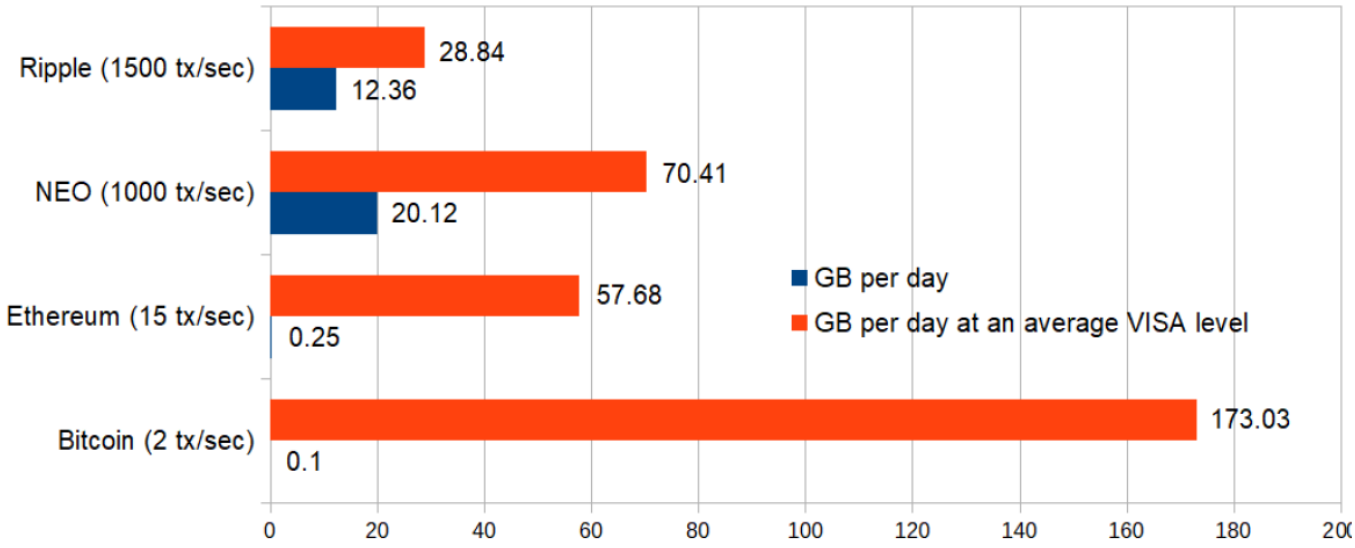

FIGURE 11: Estimation du stockage - Architectures distribuées validées fonctionnant à un débit (TPS) moyen correspondant à celui de VISA

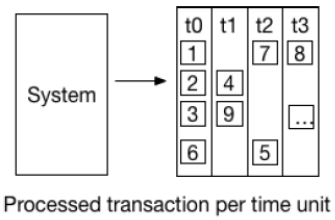

FIGURE 12: Débit des transactions

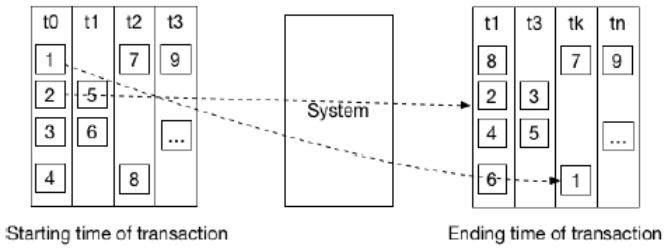

FIGURE 13: Finalité des transactions

Architecture	Type	Dispersion	TPS (average)	TPS (max limit)
VISA	Virtualisation distribuée	Centralisé	3500	55000
Paypal	Virtualisation distribuée	Centralisé	200	450
Ripple	Chaine de blocs privée	Avec autorisation	1500	55000
NEO	Chaine de blocs privée	Mixte	1000	10000
Ethereum	Chaine de blocs publique	Décentralisée	15	25
Bitcoin	Chaine de blocs publique	Décentralisée	2	7

TABLE 2: Comparaison des débits entres systèmes centralisés et décentralisés

simulations utilisant notre testnet en utilisant les moyennes de vitesse du réseau mondial, à son maximum limite théorique, suggèrent qu'Elrond dépasse le niveau moyen de VISA avec seulement 2 fragments, et atteint un niveau proche du pic de VISA avec 16 fragments.

2 Recherche en cours et à venir

Notre équipe réévalue et améliore en permanence la conception d'Elrond, dans l'objectif d'en faire l'architecture blockchain publique la plus attractive ; résoudre les problèmes de scalabilité par le partage d'états adaptatif, tout en maintenant la sécurité et l'efficacité énergétique grâce à un consensus par preuve d'enjeu. Quelques-unes de nos prochaines pistes d'amélioration incluent :

1) **Renforcement de l'apprentissage :** nous visons à accroître l'efficacité du processus de partitionnement horizontal en allouant les clients dans le même fragment pour réduire le coût global ;
2) **Supervision par IA :** créer un superviseur en IA qui détecte les modèles de comportement malveillant ; on ne sait pas encore comment cette fonctionnalité peut être intégrée dans le protocole sans perturber la décentralisation ;
3) **La fiabilité comme facteur de consensus :** le protocole existant fait la pesée entre l'enjeu et la notation, mais nous prévoyons d'ajouter la fiabilité, comme une mesure qui devrait être calculée de manière distribuée après

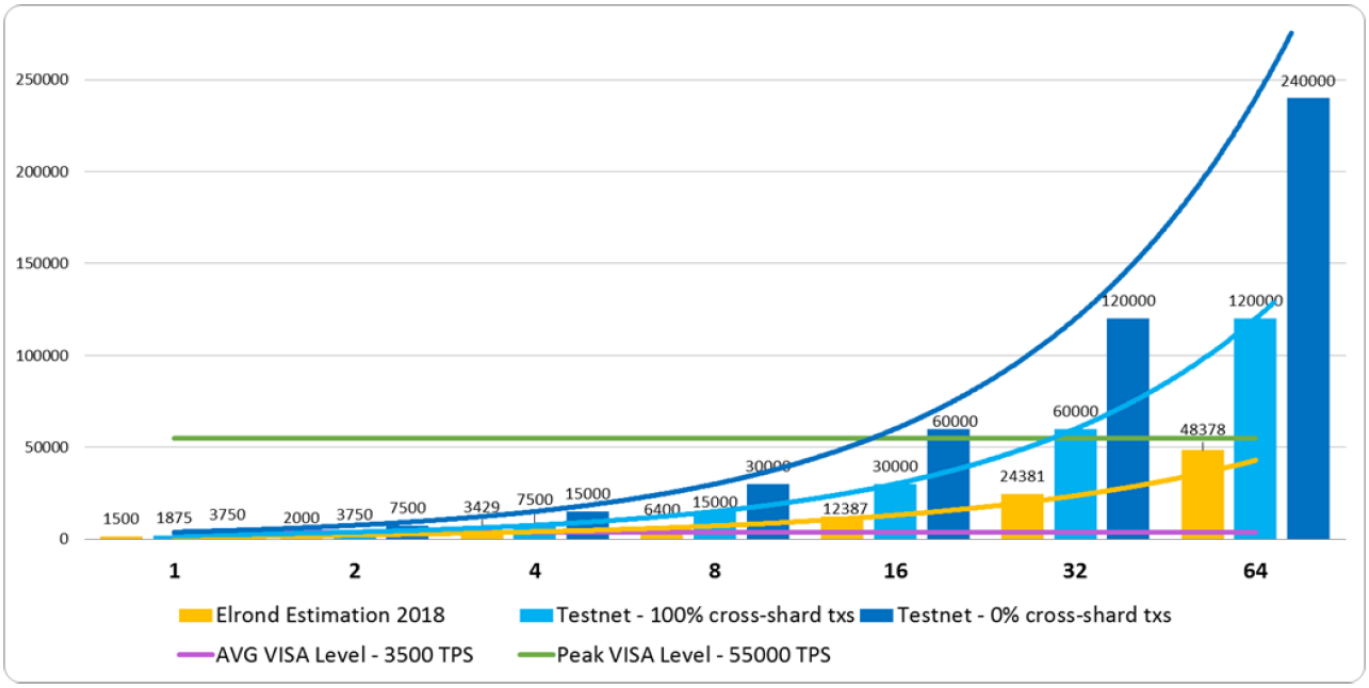

FIGURE 14: Débit du réseau mesuré en transactions par seconde avec une vitesse globale du réseau de 8 Mo/s

l'application d'un protocole de consensus sur des blocs soumis provenant de la période précédente très proche ;

4) **Interopérabilité interchaînes** : mettre en œuvre et contribuer aux normes comme celles initiées par la "Decentralized Identity Foundation" [53] ou la "Blockchain Interoperability Alliance" [54] ;
5) **Transactions préservant la vie privée** : utiliser le principe "zéro connaissance" (Zero-Knowledge) L'argument succinct et non interactif de la connaissance [55] pour protéger l'identité des participants et d'offrir des capacités d'audit tout en préservant la vie privée.

3 Conclusions générales

Elrond est la première chaîne de blocs publique hautement scalable qui utilise le nouvel algorithme de preuve d'enjeu sécurisée (*Secure Proof of Stake*) dans une véritable architecture de partage d'états et qui atteint un débit équivalent à celui de VISA et avec des temps de confirmation de l'ordre de quelques secondes. Le système d'Elrond met en oeuvre une nouvelle approche de partage d'états adaptatif en améliorant la solution d'Omniledger visant à accroître la sécurité et le débit, tout en réduisant considérablement les temps de latence grâce à un mécanisme intégré qui assure l'acheminement automatique des transactions et la redondance des états. Les coûts d'amorçage et de stockage sont également considérablement réduits par rapport à d'autres approches grâce à une technique d'élagage. L'algorithme de consensus "*Secure Proof of Stake - SPoS*" récemment introduit assure une équité distribuée et améliore l'idée d'Algorand de la sélection aléatoire, réduisant ainsi le temps nécessaire à la sélection du groupe de consensus de 12 secondes à 100 ms. Notre méthode pour combiner le partage d'états avec l'algorithme très performant de consensus par preuve d'enjeu a montré des résultats prometteurs dans nos premières estimations, validés par les derniers résultats de notre testnet.

Références

[1] G. Hileman and M. Rauchs, "2017 Global Cryptocurrency Benchmarking Study," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2965436, Apr. 2017. [Online]. Available : https://papers.ssrn.com/abstract=2965436

[2] "The Ethereum Wiki - Sharding FAQ," 2018, original-date : 2014-02-14T23 :05 :17Z. [Online]. Available : https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand : Scaling Byzantine Agreements for Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA : ACM, 2017, pp. 51–68. [Online]. Available : http://doi.acm.org/10.1145/3132747.3132757

[4] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology – ASIACRYPT '01, LNCS*. Springer, 2001, pp. 514–532.

[5] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Advances in Cryptology – ASIACRYPT 2018*, ser. Lecture Notes in Computer Science, vol. 11273. Springer, 2018, pp. 435–464.

[6] V. Buterin, "Ethereum : A Next-Generation Smart Contract and Decentralized Application Platform," 2013. [Online]. Available : https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf

[7] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger : A Secure, Scale-Out, Decentralized Ledger via Sharding," Tech. Rep. 406, 2017. [Online]. Available : https://eprint.iacr.org/2017/406

[8] "The ZILLIQA Technical Whitepaper," 2017. [Online]. Available : https://docs.zilliqa.com/whitepaper.pdf

[9] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace : A Sharded Smart Contracts Platform," *arXiv :1708.03778 [cs]*, Aug. 2017, arXiv : 1708.03778. [Online]. Available : http://arxiv.org/abs/1708.03778

[10] G. Wood, "Ethereum : A Secure Decentralised Generalised Transaction Ledger," 2017. [Online]. Available : https://ethereum.github.io/yellowpaper/paper.pdf

[11] "Solidity — Solidity 0.4.21 documentation." [Online]. Available : https://solidity.readthedocs.io/en/v0.4.21/

[12] "web3j," 2018. [Online]. Available : https://github.com/web3j

[13] "Casper," 2018. [Online]. Available : http://ethresear.ch/c/casper

[14] "The State of Ethereum Scaling, March 2018 – Highlights from EthCC on Plasma Cash, Minimum Viable Plasma, and More… – Medium," 2018. [Online]. Available : https://medium.com/loom-network/the-state-of-ethereum-scaling-march-2018-74ac08198a36

[15] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. Berkeley, CA, USA : USENIX Association, 1999, pp. 173–186. [Online]. Available : http://dl.acm.org/citation.cfm?id=296806.296824

[16] Y. Jia, "Op Ed : The Many Faces of Sharding for Blockchain Scalability," 2018. [Online]. Available : https://bitcoinmagazine.com/articles/op-ed-many-faces-sharding-blockchain-scalability/

[17] "Using Merklix tree to shard block validation | Deadalnix's den," 2016. [Online]. Available : https://www.deadalnix.me/2016/11/06/using-merklix-tree-to-shard-block-validation/

[18] S. Nakamoto, "Bitcoin : A Peer-to-Peer Electronic Cash System," p. 9, 2008.

[19] "Why we are building Cardano - Introduction." [Online]. Available : https://whycardano.com/

[20] "Constellation - a blockchain microservice operating system - White Paper," 2017, original-date : 2018-01-05T20 :42 :05Z. [Online]. Available : https://github.com/Constellation-Labs/Whitepaper

[21] "Bitshares - Delegated Proof-of-Stake Consensus," 2014. [Online]. Available : https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[22] dantheman, "DPOS Consensus Algorithm - The Missing White Paper," May 2017. [Online]. Available : https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[23] "EOS.IO Technical White Paper v2," 2018, original-date : 2017-06-06T07 :55 :17Z. [Online]. Available : https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[24] C. Paar and J. Pelzl, *Understanding Cryptography : A Textbook for Students and Practitioners*. Berlin Heidelberg : Springer-Verlag, 2010. [Online]. Available : //www.springer.com/gp/book/9783642041006

[25] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, Jan. 1991. [Online]. Available : https://link.springer.com/article/10.1007/BF00196725

[26] K. Michaelis, C. Meyer, and J. Schwenk, "Randomly Failed ! The State of Randomness in Current Java Implementations," in *Topics in Cryptology – CT-RSA 2013*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2013, pp. 129–144. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-642-36095-4_9

[27] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted Edwards Curves," in *Progress in Cryptology – AFRICACRYPT 2008*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jun. 2008, pp. 389–405. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-540-68164-9_26

[28] A. Poelstra, "Schnorr Signatures are Non-Malleable in the Random Oracle Model," 2014. [Online]. Available : https://download.wpsoftware.net/bitcoin/wizardry/schnorr-mall.pdf

[29] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," *arXiv :1403.6676 [cs]*, vol. 8713, pp. 313–326, 2014, arXiv : 1403.6676. [Online]. Available : http://arxiv.org/abs/1403.6676

[30] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr Multi-Signatures with Applications to Bitcoin," Tech. Rep. 068, 2018. [Online]. Available : https://eprint.iacr.org/2018/068

[31] Y. Seurin, "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model," in *Advances in Cryptology – EUROCRYPT 2012*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2012, pp. 554–571. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-642-29011-4_33

[32] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," 1983.

[33] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup Multisignatures : Extended Abstract," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, ser. CCS '01. New York, NY, USA : ACM, 2001, pp. 245–254. [Online]. Available : http://doi.acm.org/10.1145/501983.502017

[34] T. Ristenpart and S. Yilek, "The Power of Proofs-of-Possession : Securing Multiparty Signatures against Rogue-Key Attacks," in *Advances in Cryptology - EUROCRYPT 2007*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 2007, pp. 228–245. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-540-72540-4_13

[35] M. Bellare and G. Neven, "Multi-signatures in the Plain public-Key Model and a General Forking Lemma," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA : ACM, 2006, pp. 390–399. [Online]. Available : http://doi.acm.org/10.1145/1180405.1180453

[36] D.-P. Le, A. Bonnecaze, and A. Gabillon, "Multisignatures as Secure as the Diffie-Hellman Problem in the Plain Public-Key Model," in *Pairing-Based Cryptography – Pairing 2009*, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 2009, pp. 35–51. [Online]. Available : https://link.springer.com/chapter/10.1007/978-3-642-03298-1_3

[37] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding Concurrency to Smart Contracts," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC '17. New York, NY, USA : ACM, 2017, pp. 303–312. [Online]. Available : http://doi.acm.org/10.1145/3087801.3087835

[38] J. Kwon and E. Buchman, "Cosmos Network - Internet of Blockchains," 2017. [Online]. Available : https://cosmos.network/whitepaper

[39] G. Roșu and T. F. Șerbănuță, "An overview of the k semantic framework," *The Journal of Logic and Algebraic Programming*, vol. 79, no. 6, pp. 397–434, 2010.

[40] T. Kasampalis, D. Guth, B. Moore, T. Serbanuta, V. Serbanuta, D. Filaretti, G. Rosu, and R. Johnson, "Iele : An intermediate-level blockchain language designed and implemented using formal semantics," Tech. Rep., 2018.

[41] E. Hildenbrandt, M. Saxena, X. Zhu, N. Rodrigues, P. Daian, D. Guth, and G. Rosu, "Kevm : A complete semantics of the ethereum virtual machine," Tech. Rep., 2017.

[42] "How Formal Verification of Smart Contracts Works | RV Blog." [Online]. Available : https://runtimeverification.com/blog/how-formal-verification-of-smart-contracts-works/

[43] "Cross-shard contract yanking." [Online]. Available : https://ethresear.ch/t/cross-shard-contract-yanking/1450

[44] R. C. Merkle, "A Certified Digital Signature," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, ser. Lecture Notes in Computer Science. Springer, New York, NY, Aug. 1989, pp. 218–238. [Online]. Available : https://link.springer.com/chapter/10.1007/0-387-34805-0_21

[45] A. Veysov and M. Stolbov, "Financial System Classification : From Conventional Dichotomy to a More Modern View," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2114842, Jul. 2012. [Online]. Available : https://papers.ssrn.com/abstract=2114842

[46] "XRP - The Digital Asset for Payments." [Online]. Available : https://ripple.com/xrp/

[47] "Visa - Annual Report 2017," 2018. [Online]. Available : https://s1.q4cdn.com/050606653/files/doc_financials/annual/2017/Visa-2017-Annual-Report.pdf

[48] "PayPal Reports Fourth Quarter and Full Year 2017 Results (NASDAQ :PYPL)," 2018. [Online]. Available : https://investor.paypal-corp.com/releasedetail.cfm?releaseid=1055924

[49] M. Schwarz, "Crypto Transaction Speeds 2018 - All the Major Cryptocurrencies," 2018. [Online]. Available : https://www.abitgreedy.com/transaction-speed/

[50] "The Ethereum Wiki - Mining," 2018, original-date : 2014-02-14T23 :05 :17Z. [Online]. Available : https://github.com/ethereum/wiki/wiki/Mininghttps://github.com/ethereum/wiki

[51] "Transaction throughput." [Online]. Available : https://docs.oracle.com/cd/E17276_01/html/programmer_reference/transapp_throughput.html

[52] W. Martino, M. Quaintance, and S. Popejoy, "Chainweb : A Proof-of-Work Parallel-Chain Architecture for Massive Throughput," 2018. [Online]. Available : http://kadena.io/docs/chainweb-v15.pd

[53] "DIF - Decentralized Identity Foundation." [Online]. Available : http://identity.foundation/

[54] H. I. World, "Blockchain Interoperability Alliance : ICON x Aion x Wanchain," Dec. 2017. [Online]. Available : https://medium.com/helloiconworld/blockchain-interoperability-alliance-icon-x-aion-x-wanchain-8aeaafb3ebdd

[55] S. Goldwasser, S. Micali, and C. Rackoff, “The Knowledge Complexity of Interactive Proof-systems,” in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, ser. STOC ’85. New York, NY, USA : ACM, 1985, pp. 291–304. [Online]. Available : http://doi.acm.org/10.1145/22145.22178