Réseau Axelar :
Connecter les applications aux écosystèmes de blockchains

Draft 1.0

Janvier 2021

Traduction de https://static1.squarespace.com/static/5f7679246f2afb311bbb67dd/t/602d3503ad35b40d1bdc209c/1613575428020/axelar_whitepaper_v1.pdf

Résumé

De nombreux écosystèmes de blockchains émergent, qui offrent des caractéristiques uniques et distinctes attrayantes pour les utilisateurs et les développeurs d'applications. Cependant, la communication entre les écosystèmes est très rare et fragmentée. Pour permettre aux applications de communiquer avec fluidité à travers les écosystèmes de blockchains, nous proposons Axelar. La pile Axelar fournit un réseau décentralisé, des protocoles, des outils et des API qui permettent une communication simple entre les chaînes. La suite de protocoles Axelar se compose de protocoles de routage et de transferts transfrontaliers. Un réseau ouvert décentralisé de validateurs alimente le réseau ; tout le monde peut y adhérer, l'utiliser et y participer. Le consensus byzantin, la cryptographie et les mécanismes d'incitation sont conçus pour répondre à des exigences élevées de sécurité et d'animation, uniques pour les requêtes inter-chaînes.

1 Introduction

Les systèmes de blockchains gagnent rapidement en popularité et attirent de nouveaux cas d'utilisation pour la tokenisation des actifs, la finance décentralisée et d'autres applications distribuées. Plusieurs grandes plateformes telles que Ethereum, Monero, EOS, Cardano, Terra, Cosmos, Avalanche, Algorand, Near, Celo et Polkadot offrent chacune leurs caractéristiques et leurs environnements de développement qui les rendent attrayantes pour diverses applications, divers cas d'utilisation et divers utilisateurs finaux [5, 11, 4, 21, 20, 23, 24, 19, 6, 14, 25]. Cependant, les fonctionnalités utiles de chaque nouvelle plateforme sont actuellement offertes à moins de 1% des utilisateurs de l'écosystème, à savoir les détenteurs du jeton natif sur cette plateforme. Pouvons-nous permettre aux développeurs de plateformes de connecter facilement leurs blockchains à d'autres écosystèmes ? Pouvons-nous permettre aux créateurs d'applications de construire sur la plateforme la mieux adaptée à leurs besoins tout en communiquant avec de multiples écosystèmes de blockchains ? Pouvons-nous permettre aux utilisateurs d'interagir avec n'importe quelle application sur n'importe quelle blockchain directement depuis leur portefeuille, leur *wallet* ?

Pour relier les écosystèmes de la chaîne de production et permettre aux applications de communiquer avec fluidité entre elles, nous proposons le réseau Axelar. Les validateurs exécutent collectivement un protocole de consensus byzantin et exécutent les protocoles facilitant les requêtes inter-chaînes. Tout le monde peut rejoindre le réseau, y participer et l'utiliser. Le réseau sous-jacent est optimisé pour des

exigences élevées de sécurité et de vitalité, uniques pour les requêtes inter-chaînes. Le réseau Axelar comprend également une suite de protocoles et des API. Les protocoles de base sont les suivants :

- Cross-Chain Gateway Protocol (CGP). Ce protocole est analogue au Border Gateway Protocol sur Internet. Ce protocole est utilisé pour connecter plusieurs écosystèmes de blockchains autonomes et est responsable du routage à travers eux. Les blockchains n'ont pas besoin de « parler un langage spécial », les développeurs de leurs plateformes n'ont pas besoin d'apporter de modifications spécifiques à leurs chaînes et leurs chaînes peuvent être facilement connectées au réseau mondial.
- Cross-Chain Transfer Protocol (CTP). Ce protocole est analogue aux protocoles de niveau application FTP et HTTP de l'Internet. Il s'agit d'une pile de protocoles de niveau application au-dessus des protocoles de routage (tels que CGP et autres technologies de routage). Les développeurs d'applications peuvent connecter leurs dapps sur n'importe quelle chaîne pour effectuer des requêtes inter-chaînes. Les utilisateurs peuvent utiliser le protocole CTP pour interagir avec des applications sur n'importe quelle chaîne en utilisant de simples appels API analogues aux requêtes HTTP GET/POST. Les développeurs peuvent verrouiller, déverrouiller et transférer des ressources entre deux adresses quelconques sur n'importe quelle plateforme de chaîne de bloc, exécuter des déclencheurs d'applications inter-chaînes (par exemple, une dapp sur la chaîne A peut mettre à jour son état si une autre application sur la chaîne B satisfait à certains critères de recherche (taux d'intérêt $>$ X)), et effectuer des requêtes génériques entre applications sur les chaînes (un smart contract sur la chaîne A peut faire un appel pour mettre à jour l'état d'un smart contract sur la chaîne B). Ce protocole permet la composabilité des programmes à travers les écosystèmes des blockchains.

Le réseau Axelar offre les avantages suivants :

- *Pour les développeurs de plates-formes blockchain :* Possibilité de connecter facilement leurs blockchains à tous les autres écosystèmes. Seul un compte « à seuil » (*threshold account*) doit être créé sur la chaîne pour se connecter au réseau.
- *Pour les auteurs de dapps :* Les créateurs d'applications peuvent héberger leurs dapps n'importe où, verrouiller, déverrouiller, transférer des ressources et communiquer avec les applications de n'importe quelle autre chaîne via l'API CTP.
- *Pour les utilisateurs :* Les utilisateurs peuvent interagir avec toutes les applications de l'écosystème directement depuis leur wallet.

Une plate-forme pour les développeurs. Enfin, le réseau Axelar est une plateforme pour les développeurs ainsi qu'une communauté mondiale. Son modèle de gouvernance est ouvert à tous. Les développeurs peuvent proposer de nouveaux points d'intégration, de nouveaux routages et de nouveaux protocoles au niveau des applications, et les utilisateurs peuvent décider de les adopter en votant sur les propositions et, si elles sont approuvées, les validateurs adopteront les changements.

1.1 Solutions d'interopérabilité existantes

Les tentatives précédentes pour résoudre le problème de l'interopérabilité entre blockchains appartiennent à l'une des quatre catégories suivantes : échanges centralisés, écosystèmes interopérables, actifs enveloppés (*wrapped*) et ponts de tokens. Nous résumons brièvement ces approches ci-dessous.

Systèmes centralisés. Aujourd'hui, les systèmes centralisés sont les seules solutions véritablement évolutives pour répondre aux besoins d'interopérabilité de l'écosystème. Ils peuvent lister n'importe quel actif ou embarquer n'importe quelle plateforme relativement facilement. Cependant, les systèmes centralisés sont connus pour avoir divers problèmes de sécurité et leur qualité n'est pas suffisante pour propulser le système financier décentralisé émergent qui nécessite une sécurité solide, de la transparence et une gouvernance ouverte. Ils ne peuvent pas à eux seuls alimenter les applications décentralisées au fur et à mesure de leur développement.

Les centres d'interopérabilité. Des projets tels que Cosmos, Polkadot et Avalanche traitent de l'interopérabilité entre les *sidechains* natives de leurs écosystèmes en utilisant des protocoles de communication inter-chaîne personnalisés [23, 25, 24]. Par exemple, on peut faire tourner une sidechain (une zone de Cosmos) qui peut communiquer avec le Cosmos Hub. La sidechain doit être basée sur le consensus de Tendermint et doit parler le protocole nativement compris par le Cosmos Hub. Les connexions avec les autres blockchains et écosystèmes qui parlent des langages différents sont laissées à des technologies externes.

Ponts par paires. Les actifs enveloppés (par exemple, les bitcoins enveloppés) tentent de combler le manque d'interopérabilité entre les chaînes dans l'écosystème. Un exemple est le tBTC [9], qui est un protocole personnalisé dans lequel une combinaison ingénieuse de smart contracts et de garanties est utilisée pour sécuriser les transferts. Ces solutions nécessitent des efforts d'ingénierie considérables ; pour chaque paire de chaînes, les développeurs doivent établir un nouveau smart contract sur la chaîne de destination qui analyse les preuves d'état de la chaîne d'origine (comme chaque sidechain pourrait, en principe, analyser l'état des autres chaînes). Seuls quelques ponts ont été déployés en utilisant cette approche. Ces approches ne sont pas adaptées lorsque l'une des blockchains sous-jacentes souhaite améliorer ses règles de consensus ou son format de transaction. En effet, tous les smart contracts qui dépendent de l'état de ces chaînes doivent alors être mis à niveau. Il faut également mettre en place des validateurs et leur demander de bloquer différents actifs afin de surdimensionner tout transfert d'actifs, ce qui limite l'efficacité économique de ces transferts.

Nous avons également vu quelques autres ponts à usage unique créés par des développeurs de plateformes qui réécrivent la logique de transition d'état dans des smart contracts pour établir des ponts avec d'autres écosystèmes [1, 7]. Ils souffrent de multiples problèmes d'extensibilité, ne permettent pas à l'écosystème de s'adapter de manière uniforme et introduisent des dépendances

supplémentaires pour les applications. Par exemple, si une plateforme change, tous les smart contracts de toutes les passerelles devront être mis à niveau. Cela finira par mettre l'écosystème dans une impasse où personne ne pourra l'améliorer. Enfin, si un pont à usage unique relie les plateformes A et B, et qu'un second pont à usage unique relie B et C, cela ne signifie pas que les applications sur A pourront parler aux applications sur C. Il faudra peut-être créer un autre pont à usage unique ou recâbler la logique applicative.

D'autres tentatives pour aborder l'interopérabilité comprennent les oracles fédérés (par exemple, Ren [8]), et les blockchains interopérables spécifiques aux applications [10].

En résumé, les solutions existantes en matière d'interopérabilité nécessitent un travail d'ingénierie important de la part des développeurs de plateformes et des créateurs d'applications qui doivent comprendre les différents protocoles de communication pour communiquer entre chaque paire d'écosystèmes. Ainsi, l'interopérabilité est pratiquement inexistante dans l'espace actuel des chaînes de production. En fin de compte, les développeurs de plateformes veulent se concentrer sur la création de plateformes et les optimiser pour leurs cas d'utilisation et être en mesure de se connecter facilement à d'autres blockchains. Et les développeurs d'applications veulent écrire des dapps sur les meilleures platesformes pour leurs besoins tout en continuant à tirer parti des utilisateurs, des liquidités et à communiquer avec d'autres dapps sur d'autres chaînes.

2 La quête d'une communication inter-chaînes extensible

Au cœur de la communication inter-chaînes, il faut que les réseaux hétérogènes trouvent la possibilité de communiquer en utilisant le même langage. C'est pourquoi nous expliquons la suite de protocoles Axelar, décrivons ses propriétés de haut niveau et expliquons comment ces propriétés s'adressent au cœur de la communication interchaînes évolutive.

1 *Intégration « Plug-and-play ».* Les développeurs de plates-formes blockchains ne devraient pas être obligés d'effectuer des travaux d'ingénierie ou d'intégration lourds pour parler un « langage spécial » afin de prendre en charge la chaîne croisée. Le protocole inter-chaînes devrait pouvoir se connecter de manière fluide à toute blockchain existante ou nouvelle. De nouveaux actifs devraient être ajoutés avec un minimum d'efforts.

2 *Routage inter-chaînes.* Des fonctions telles que la découverte des adresses, des routes et des réseaux sont au cœur de l'Internet et sont facilitées par BGP et les autres protocoles de routage. De même, pour faciliter la communication entre les écosystèmes de blockchains, il faut supporter la découverte des adresses, des applications et du routage entre eux.

3 *Support des mises à niveau.* Si l'un des écosystèmes de la chaîne de production change, cela ne devrait pas affecter l'interopérabilité des autres chaînes. Le système doit reconnaître les mises à jour, et un effort minimal doit être requis pour les prendre en charge (c'est-à-dire qu'aucune

« logique de transition d'état » ne doit être réécrite, et les applications ne doivent pas être cassées de ce fait).

4 *Un langage uniforme pour les applications.* Les applications ont besoin d'un protocole simple pour verrouiller, déverrouiller, transférer et communiquer entres applications, quelle que soit la chaîne sur laquelle elles se trouvent. Ce protocole doit être indépendant des chaînes et prendre en charge les appels simples, comme les protocoles HTTP/HTTPS qui permettent aux utilisateurs et aux navigateurs de communiquer avec n'importe quel serveur web. À mesure que de plus en plus de réseaux et d'actifs rejoignent les protocoles de routage de bas niveau, les applications devraient pouvoir les utiliser pour communiquer sans réécrire leurs socles logiciels.

Ensuite, nous résumons les exigences de sécurité auxquelles ces protocoles doivent se conformer.

1 *Confiance décentralisée.* Le réseau et les protocoles doivent être décentralisés, ouverts et permettre à chacun de participer équitablement.

2 *Sécurité élevée.* Le système doit satisfaire à des garanties de sécurité élevées. Le système doit préserver la sécurité des actifs et de l'état pendant que le réseau inter-chaînes le traite.

3 *Grande vitalité.* Le système doit satisfaire à des garanties de vitalité pour pouvoir supporter des applications tirant parti de ses caractéristiques inter-chaînes. Il est facile de satisfaire un sous-ensemble de ces propriétés. Par exemple, on peut créer un compte fédéré *multisig* avec ses amis et verrouiller/déverrouiller les actifs sur les chaînes correspondantes. De tels systèmes sont intrinsèquement vulnérables aux attaques de collusion et de censure, et les validateurs ne sont pas suffisamment incités à les protéger. La création d'un réseau décentralisé et d'une suite de protocoles où chacun peut participer tout en étant correctement incité permet une communication de manière fluide entre les chaînes, mais il s'agit d'un problème difficile qui nécessite une combinaison complexe de protocoles de consensus, de cryptographie et de conception de systèmes.

3 Réseau Axelar

Le réseau Axelar fournit une solution uniforme de communication inter-chaînes qui répond aux besoins des développeurs de plateformes - aucun travail d'intégration n'est requis de leur part - et des créateurs d'applications - un protocole et une API simples pour accéder aux liquidités globales et pour communiquer avec l'ensemble de l'écosystème.

Le réseau Axelar se compose d'un réseau décentralisé qui relie des écosystèmes de blockchains parlant différents langages et d'une suite de protocoles avec des API au sommet, ce qui permet aux applications d'effectuer facilement des requêtes inter-chaînes. Le réseau relie des blockchains autonomes existantes telles que Bitcoin, Stellar, Terra, Algorand, et des centres d'interopérabilité tels que des solutions comme Cosmos, Avalanche, Ethereum et Polkadot. Notre mission est de permettre aux développeurs d'applications d'écrire plus facilement ces applications en utilisant un

protocole universel et une API sans avoir à déployer leurs protocoles propriétaires inter-chaînes en dessous ou à réécrire des applications au fur et à mesure que de nouveaux ponts sont développés. Pour ce faire, nous avons conçu une suite de protocoles qui comprend le Cross-Chain Gateway Protocol (voir section 6) et le Cross-Chain Transfer Protocol (voir section 7).

Les protocoles décentralisés sous-jacents constituent un élément essentiel du réseau. Les validateurs assurent collectivement la maintenance du réseau Axelar et font fonctionner les nœuds qui sécurisent la blockchain Axelar. Ils sont élus par les utilisateurs dans le cadre d'un processus de délégation. Les validateurs reçoivent un pouvoir de vote proportionnel à l'enjeu qui leur est délégué. Les validateurs parviennent à un consensus sur l'état des multiples blockchains auxquelles la plateforme est connectée. La blockchain est responsable de la maintenance et de l'exécution des protocoles de routage et de transfert entre les chaînes. Les règles de gouvernance permettent aux participants au réseau de prendre des décisions concernant les protocoles, comme par exemple les blockchains à relier et les actifs à prendre en charge.

La blockchain Axelar suit un modèle de preuve d'enjeu déléguée (DPoS) similaire à celui de Cosmos Hub. Les utilisateurs élisent des validateurs qui doivent bloquer leur enjeu pour participer au consensus et maintenir un service de haute qualité. Le modèle DPoS permet le maintien d'un large ensemble de validateurs décentralisés et de solides incitations pour garantir que les validateurs sont responsables du maintien des ponts et des parts des systèmes à seuils cryptographiques. Dans le cadre du consensus, les validateurs utilisent des logiciels clients légers d'autres blockchains, ce qui leur permet de vérifier l'état de ces dernières. Les validateurs signalent ces états à la blockchain Axelar, et une fois qu'ils sont suffisamment nombreux à le faire, l'état des chaînes Bitcoin, Ethereum et autres est enregistré sur Axelar.

Par la suite, la couche de base d'Axelar est au courant de l'état des blockchains externes à tout moment, créant ainsi les « ponts entrants » à partir d'autres blockchains. Les validateurs gèrent collectivement des *comptes de signatures à seuil sur les* autres blockchains (par exemple, 80 % des validateurs doivent approuver et cosigner toute transaction en dehors de ces chaînes), ce qui leur permet de verrouiller et de déverrouiller des actifs et des états dans les chaînes et d'afficher des états sur d'autres blockchains, les « ponts sortants ». Dans l'ensemble, on peut considérer le réseau Axelar comme un *oracle décentralisé de lecture/écriture inter-chaîne.*

Le reste du document décrit les préliminaires et les éléments constitutifs du réseau (section 4), certains détails techniques du réseau (section 5), le protocole de passerelle inter-chaînes (section 6) et le protocole de transfert inter-chaînes (section 7).

4 Préliminaires

4.1 Notation et hypothèses

Soit V^r l'ensemble des validateurs Axelar au tour R. Chaque validateur a un *poids*, un nombre dans $(0, 1)$ indiquant le pouvoir de vote de ce validateur particulier. La somme des poids de tous les validateurs est égale à 1. Un validateur est *correct* s'il exécute un nœud qui est conforme aux règles du protocole Axelar. Pour finaliser des blocs, ou pour signer des requêtes inter-chaînes, Axelar a besoin de validateurs corrects d'un poids total $> F$. Nous appelons le paramètre $F \in [0{,}5, 1]$ le *seuil du protocole*.

Axelar peut être basé sur une blockchain *à finalité instantanée en preuve d'enjeu déléguée.* Les validateurs exécutent le *consensus Byzantine Fault Tolerant (BFT)* à chaque tour i pour finaliser le ième bloc. Une fois que le ième bloc est finalisé, un nouveau consensus BFT est exécuté pour finaliser le bloc $i + 1$, et ainsi de suite. Les validateurs sont élus par délégation des parties prenantes. Un utilisateur ayant un certain enjeu peut choisir d'exécuter un nœud de validateur, ou de déléguer son pouvoir de vote (enjeu) à un validateur existant, qui vote alors en son nom. Le jeu de validateurs peut être mis à jour, les validateurs rejoignent ou quittent le jeu, et les utilisateurs délèguent ou retirent leur pouvoir de vote.

Les diverses blockchains fonctionnent selon des hypothèses réseau différentes. La *communication synchrone* signifie qu'il existe une limite supérieure fixe Δ sur le temps que prennent les messages pour être délivrés, où Δ est connu et peut être intégré dans le protocole. La *communication asynchrone* signifie que les messages peuvent prendre un temps quelconque pour être délivrés, et il est connu que les protocoles BFT ne peuvent pas être implémentés pour des réseaux asynchrones même en présence d'un seul validateur malveillant. Un compromis réaliste entre la synchronie et l'asynchronie est l'hypothèse d'une *communication partiellement synchrone*. Le réseau peut être complètement asynchrone jusqu'à un certain temps de stabilisation global inconnu (GST), mais après le GST, la communication devient synchrone avec une limite supérieure connue Δ [17].

Les blockchains typiques fonctionnent selon l'hypothèse de $> F$ validateurs corrects. Pour les réseaux synchrones, $F = 1/2$ est généralement fixé, mais pour l'hypothèse plus faible d'un réseau partiellement synchrone, $F = 2/3$. Bitcoin, ses *forks* et la version actuelle de la preuve de travail d'Ethereum ne fonctionnent qu'en supposant la synchronie. D'autres, comme Algorand et Cosmos, ne requièrent qu'une synchronisation partielle. Lors de la connexion de chaînes par Axelar, la connexion fonctionne en supposant les hypothèses réseau les plus fortes parmi ces chaînes, ce qui est la synchronie dans le cas de la connexion de Bitcoin et Cosmos, par exemple. La blockchain Axelar elle-même fonctionne dans un cadre partiellement synchrone et nécessite donc $F = 2/3$, mais il est possible d'améliorer le seuil requis en supposant que les autres blockchains existantes sont sûres et en tirant parti de leur sécurité.

4.2 Préliminaires cryptographiques

Signatures numériques. Un *schéma de signature numérique* est un ensemble d'algorithmes (*Keygen, Sign, Verify*). *Keygen* produit une paire de clés (*PK, SK*). Seul le propriétaire de *SK* peut signer des messages, mais n'importe qui peut vérifier les signatures grâce à la clé publique *PK*. La plupart des blockchains utilisent aujourd'hui l'un des schémas de signature standard tels que ECDSA, Ed25519, ou quelques unes de leurs variantes [2, 3].

Signatures à seuils. Un *système de signature à seuil* permet à un groupe de n parties de diviser une clé secrète pour un système de signature de telle sorte que tout sous-ensemble de $t + 1$ ou plusieurs parties puissent collaborer pour produire une signature, mais qu'aucun sous-ensemble de t ou moins de parties ne puisse produire une signature ou même apprendre des informations sur la clé secrète. Les signatures produites par les protocoles à seuil pour l'ECDSA et l'EdDSA sont identiques aux signatures produites par les algorithmes classiques.

Un système de signature à seuil remplace les algorithmes *Keygen* et *Sign* pour un système de signature ordinaire par des protocoles distribués à *n parties T.Keygen*, *T.Sign*. Ces protocoles nécessitent généralement un canal de diffusion public et des canaux privés par paire entre les parties, et ils impliquent généralement plusieurs cycles de communication. Après avoir réussi le *T.Keygen,* chaque utilisateur détient une part s_i *d'*une clé secrète SK et de la clé publique correspondante PK. Le protocole *T.Sign* permet à ces parties de produire une signature pour un message donné qui est valable sous la clé publique PK. Cette signature peut être vérifiée par toute personne utilisant l'algorithme *Vérifier* du système de signature d'origine.

4.3 Propriétés des signatures à seuils

Il existe plusieurs propriétés pour un système à seuil particulièrement souhaitables pour les réseaux décentralisés :

La sécurité contre une majorité malhonnête. Certains systèmes à seuil ont comme restriction qu'ils ne sont sûrs que lorsqu'une majorité des n parties est honnête. Ainsi, le paramètre de seuil t doit être inférieur à $n/2$ [15]. Cette restriction s'accompagne généralement du fait que 2t + 1 parties honnêtes sont nécessaires pour signer, même si seules $t + 1$ parties corrompues peuvent s'entendre pour récupérer la clé secrète. Les systèmes qui ne souffrent pas de cette restriction sont censés être *à l'abri d'une majorité malhonnête.*

Comme nous le verrons plus loin dans la section 5.2, les plateformes inter-chaînes doivent maximiser la sécurité de leurs réseaux et être capables de tolérer le plus grand nombre possible de parties corrompues. Il est donc nécessaire de mettre en place des systèmes capables de tolérer une majorité malhonnête.

Pré-signatures, signature en ligne non interactive. Afin de réduire la charge de communication pour les parties signataires d'un message, plusieurs protocoles récents ont identifié une partie importante du travail de signature qui peut être effectuée « hors ligne », avant que le message

à signer ne soit connu [18, 13]. Le résultat de cette phase hors ligne est appelé une *pré-signature*. La production de pré-signatures est considérée comme un protocole distinct *T.Presign,* distinct de T.*Keygen* et *T.Sign*. Les résultats du protocole de pré-signature doivent rester privés par les parties jusqu'à ce qu'elles les utilisent lors de la phase de signature. Plus tard, lorsque le message à signer est connu, il ne reste plus qu'un petit travail supplémentaire « en ligne » à effectuer dans *T.Sign* afin de compléter la signature.

La phase de *T.Sign* en ligne ne nécessite aucune communication entre les parties. Chaque partie fait simplement un calcul local sur le message et la présignature, puis annonce sa part s_i *de* la signature. (Une fois rendue publique, ces parts de signature $s_1,\ldots,s_{t+1}$ sont facilement combinées par quiconque pour révéler la signature réelle s.) Cette propriété est appelée *signature en ligne non interactive*.

Robustesse. Les systèmes à seuil garantissent uniquement qu'un sous-ensemble de parties malveillantes ne peut pas signer de messages ou apprendre la clé secrète. Cette garantie n'exclut toutefois pas la possibilité que des acteurs malveillants puissent empêcher tout le monde de produire des clés ou des signatures. Dans certains systèmes, le comportement malveillant d'une seule partie peut entraîner l'interruption de *T.Keygen* ou de *T.Sign* sans résultat utile. Le seul recours est de redémarrer le protocole, éventuellement avec d'autres parties.

Au contraire, pour les réseaux décentralisés, nous voulons que *T.Keygen* et *T.Sign* réussissent si au moins $t + 1$ des parties sont honnêtes, même si certaines parties malveillantes envoient des messages malformés ou laissent tomber des messages dans les protocoles. Cette propriété est appelée *robustesse*.

Attribution de fautes. La capacité à identifier les mauvais acteurs dans le *T.Keygen* ou le *T.Sign* est appelée *attribution de faute*. Sans attribution de faute, il est difficile d'exclure ou de punir de manière fiable les mauvais acteurs, auquel cas les coûts imposés par les mauvais acteurs doivent être supportés par tout le monde. Cette propriété est également importante pour les réseaux décentralisés où les comportements malveillants devraient être identifiables et économiquement désincités par des règles de punition.

Sécurité dans les configurations concurrentes. Le système de signature doit être sécurisé dans un environnement simultané, où plusieurs instances du *keygen* et des algorithmes de signature peuvent être utilisées en parallèle. (Drijvers et al. [16], par exemple, ont montré une attaque contre les schémas multi-signatures de Schnorr dans ces configurations). Il existe des versions de systèmes ECDSA et Schnorr qui satisfont à ces propriétés [13, 22].

ECDSA et EdDSA sont de loin les systèmes de signature les plus largement déployés dans le domaine de la blockchain. En tant que tels, les versions à seuil de ces deux systèmes ont fait l'objet d'une récente reprise de la recherche et du développement. Les lecteurs intéressés par l'état de l'art peuvent se référer à [22, 13, 18] et à un document d'enquête récent [12].

5 Réseau Axelar

5.1 Conception d'un réseau transversal ouvert

Les ponts que le réseau Axelar entretient sont adossés à des comptes à seuil de sorte que (presque) tous les validateurs doivent collectivement autoriser toute demande inter-chaîne. La conception d'un réseau auquel tout le monde peut participer pour sécuriser ces ponts nécessite de répondre aux exigences techniques suivantes :

- *Adhésion ouverte.* Tout utilisateur doit pouvoir devenir un validateur (en suivant les règles du réseau).
- *Mise à jour de l'adhésion.* Lorsqu'un validateur quitte le système honnêtement, sa clé doit être révoquée de manière appropriée.
- *Incitations et réductions.* Les validateurs malveillants doivent être identifiables et leurs actions doivent être identifiées et traitées par le protocole.
- *Consensus.* Les systèmes à seuils sont définis comme des protocoles autonomes. Pour propager les messages entre les nœuds, nous avons besoin de canaux privés de diffusion et en point à point. De plus, les validateurs doivent s'accorder sur le dernier état de chaque invocation des systèmes à seuil, car ils ont souvent plusieurs itérations d'interactions.
- *Gestion des clés.* Tout comme les validateurs ordinaires de n'importe quel système de points de vente doivent protéger leurs clés avec soin, les validateurs d'Axelar doivent protéger leurs parts de seuil. Les clés doivent être renouvelées, scindées entre les parties en ligne et hors ligne, etc.

Axelar commence avec un modèle de *Delegated Proof of Stake* (preuve d'enjeu déléguée), où la communauté élit un ensemble de validateurs pour diriger le consensus. Notez que les modèles à seuil standard traitent chaque acteur de manière identique et n'ont aucune notion de « poids » dans le consensus. Le réseau doit donc les adapter pour prendre en compte le poids des validateurs. Une approche simple consiste à attribuer plusieurs parts de seuil à des validateurs plus importants. Les trois fonctions de base que les validateurs remplissent collectivement sont décrites ci-dessous.

- *Génération de clefs à seuil. Les* algorithmes de génération de clés à seuil existants pour les schémas de signature de blockchains standard (ECDSA, Ed25519) sont des protocoles interactifs entre plusieurs participants (voir section 4). Une transaction spéciale sur le réseau Axelar ordonne aux validateurs de commencer l'exécution de ce protocole avec état. Chaque validateur exécute un processus de démon de seuil qui est responsable de la conservation sécurisée de l'état secret. Pour chaque phase du protocole :

  1 Un validateur conserve l'état du protocole dans sa mémoire locale.

  2 Il appelle le démon des secrets pour générer les messages selon la description du protocole pour les autres validateurs.

  3 Il propage les messages soit par diffusion générale, soit par les canaux privés à d'autres validateurs.

4 Chaque validateur exécute des fonctions de transition d'état pour mettre à jour son état, passer à la phase suivante du protocole et répéter les étapes ci-dessus.

A la fin du protocole, une clé publique à seuil est générée sur la chaîne Axelar, et elle peut être affichée à l'utilisateur (par exemple, pour les dépôts) ou à l'application qui a généré la demande initiale.

- *Signature à seuil.* Les demandes de signature sur le réseau Axelar sont traitées de la même manière que les demandes de génération de clés. Elles sont invoquées, par exemple, lorsqu'un utilisateur souhaite retirer un actif de l'une des chaînes. Ce sont des protocoles interactifs, et la transition d'état entre les tours est déclenchée en fonction des messages propagés à travers la vue des blockchains Axelar et la mémoire locale de chaque validateur.

- *Traitement des changements d'adhésion des validateurs.* L'ensemble des validateurs doit faire l'objet d'une rotation périodique pour permettre à de nouvelles parties prenantes de le rejoindre. Lors de la mise à jour d'un ensemble de validateurs, nous devons mettre à jour la clé à seuil à partager avec le nouvel ensemble. Ainsi, si nous permettions à n'importe qui d'adhérer à tout moment, nous devrions mettre à jour la clé à seuil très fréquemment. Pour éviter cela, nous effectuons une rotation des validateurs tous les T blocs. Dans les intervalles de T blocs, l'ensemble V^R et la clé à seuil sont fixes. À chaque tour qui est un multiple entier du paramètre T , nous mettons à jour le jeu de validateurs comme suit :
  1 À chaque tour R, l'état Axelar garde la trace de l'ensemble de validateurs actuel V^R. $V^{R+1} = V^R$ sauf si $R + 1$ est un multiple de T .
  2 Pendant les tours $((i - 1)T, iT)$, les utilisateurs déposent des messages d'engagement et de désengagement.
  3 À la fin du cycle iT, ces messages sont appliqués à V^{iT-1} pour obtenir V^{iT} .

- *Génération et signature de clés à seuil en présence de validateurs tournants.* La blockchain Axelar peut émettre une demande de nouvelle clé ou de signature à seuil au tour R. Le processus de signature prend plus d'un tour et nous ne voulons pas ralentir le consensus, c'est pourquoi nous demandons que la signature soit produite avant le début du tour $R + 10$. En particulier, les validateurs ne commencent le tour $R + 10$ qu'après avoir vu un certificat pour le tour $R + 9$ et une signature pour chaque demande de clé/signature émise au tour R. Le résultat de toutes les demandes du tour R doit être inclus dans le bloc $R + 11$. En d'autres termes, une proposition de bloc R qui ne contient pas les résultats d'un tour $R - 11$ est considérée comme non valable, et les validateurs ne votent pas dessus. Pour garantir que tous les messages à seuil sont signés avant la mise à jour d'un jeu de validateurs, Axelar n'émet pas de demande à seuil au cours d'un cycle égal à -1, -2, ... -9 modulo T .

5.2 Sécurité des réseaux

La sécurité des systèmes de blockchains repose sur divers protocoles cryptographiques et de théorie des jeux, ainsi que sur la décentralisation du réseau. Par exemple, dans les blockchains avec preuve d'enjeu, sans les incitations appropriées, les validateurs peuvent s'entendre et réécrire l'historique, volant ainsi les fonds d'autres utilisateurs au cours du processus. Dans les réseaux en preuve de

travail, sans décentralisation suffisante, il est assez facile de créer de longs *forks* et des doubles dépenses, comme l'ont prouvé les multiples attaques contre Bitcoin Gold et Ethereum Classic.

La plupart des recherches sur la sécurité des blockchains ont porté sur les chaînes souveraines. Mais une fois que les chaînes interopèrent, il faut envisager de nouveaux vecteurs d'attaque. Par exemple, supposons qu'Ethereum parle à une petite blockchain X par un pont direct contrôlé par deux smart contracts, l'un sur Ethereum et l'autre sur X. Outre les problèmes d'ingénierie que nous avons résumés en section 1.1, il faut décider de ce qui se passe lorsque les hypothèses de confiance de X sont violées. Dans ce cas, si de l'ETH s'est déplacé vers X, les validateurs de X peuvent s'entendre pour forger un historique de X où ils détiennent tout l'ETH, afficher les preuves de consensus forgées sur Ethereum et voler l'ETH. La situation est encore pire lorsque X est relié à plusieurs autres chaînes par des ponts directs, où si X *forke*, les effets se propagent à travers chaque pont. L'établissement de directives de gouvernance de la relance pour chaque pont par paire est une tâche écrasante pour tout projet individuel.

Le réseau Axelar répond aux problèmes de sécurité en utilisant les mécanismes suivants :

- *Sécurité maximale.* Axelar fixe le seuil de sécurité à 90 %, ce qui signifie que presque tous les validateurs devront s'entendre pour retirer les fonds bloqués par son réseau ou falsifier les preuves d'état[1]. En pratique, il a été observé que les validateurs en PoS ont un temps de disponibilité très élevé (près de 100%), à condition qu'ils soient correctement incités. Par conséquent, le réseau Axelar produira des blocs même en dépit de ce seuil élevé. Cependant, dans le rare cas où quelque chose tourne mal et que le réseau cale, le réseau a besoin de mécanismes de repli robustes pour redémarrer le système décrit ci-après.
- *Décentralisation maximale.* Comme le réseau utilise des systèmes de signature à seuil, le nombre de validateurs peut être aussi important que possible. Le réseau n'est pas limité par le nombre de validateurs que nous pouvons prendre en charge, les limites de transaction ou les frais qui découleraient de l'utilisation, par exemple, de signatures multiples sur différentes chaînes où la complexité (et les frais) augmentent de manière linéaire avec le nombre de validateurs[2].
- *Mécanismes de repli robustes.* La première question qui doit être abordée dans un réseau avec des seuils de sécurité élevés comme ci-dessus est ce qui se passe lorsque le réseau lui-même cale. Supposons que le réseau Axelar lui-même s'arrête. Pouvons-nous avoir un mécanisme de repli qui permettrait aux utilisateurs de récupérer leurs fonds ? Pour remédier à un éventuel blocage du réseau Axelar lui-même, chaque compte de pont à seuil sur une blockchain X que les validateurs Axelar contrôlent collectivement possède une « clé de déverrouillage d'urgence ». Cette clé peut être partagée à travers des milliers de parties et peut même être une clé spécifique pour la blockchain X qui est partagée à travers la communauté de cette chaîne. Par

1 Le paramètre final qui sera choisi pour le déploiement du réseau pourra être ajusté.

2 Pour certaines blockchains, les multi-signatures offrent une alternative raisonnable lorsque les gaz sont petits et que les formats de message supportés sont appropriés. Mais elles ne sont pas adaptées à deux des plus grandes plateformes comme Bitcoin et Ethereum.

conséquent, si le réseau Axelar est bloqué, cette clé servira de solution de repli et permettra la récupération des actifs (voir ci-dessous pour plus de détails).

- *Décentralisation maximale des mécanismes de repli.* Ce mécanisme de repli comprend un *ensemble de recouvrement* d'utilisateurs, auquel n'importe qui peut participer sans aucun coût. Ces utilisateurs n'ont pas besoin d'être en ligne, d'exécuter des nœuds ou de se coordonner entre eux. Ils ne sont « appelés en service » que si le réseau Axelar est bloqué et ne peut pas se rétablir. La sécurité du réseau est renforcée par un seuil très élevé sur l'ensemble de validateurs primaire et un ensemble de recouvrement secondaire décentralisé au maximum.
- *Gouvernance partagée.* Un protocole commun régit le réseau Axelar. Collectivement, les utilisateurs peuvent voter sur la chaîne qui doit être soutenue par son réseau. Le réseau allouera également une réserve de fonds pouvant être utilisés pour rembourser les utilisateurs en cas d'urgence imprévue, contrôlée également par les protocoles de gouvernance.

Divers mécanismes de sécurité sont examinés ci-dessous.

Mécanismes de repli. Lorsque Axelar stagne en raison du seuil élevé, une « clé de déverrouillage d'urgence » prend le contrôle du réseau. Il existe de multiples façons d'instancier cette clé de déverrouillage, et certaines chaînes/applications peuvent choisir d'utiliser une variante différente de « l'ensemble de recouvrement », ou de se désengager complètement[3]:

- *Option a.* Partager la clé entre les fondations des projets blockchain et les personnes de bonne réputation dans la communauté.
- *Option b.* Partager les informations entre les parties élues par le biais du mécanisme de PoS délégué.
- *Option c.* Pour les comptes gérant des actifs et des informations pour la chaîne/application X, partager une clé personnalisée entre les détenteurs d'enjeu/validateurs de X. En supposant que X ait mis en place des mécanismes de gouvernance, les mêmes mécanismes peuvent être appliqués pour déterminer une ligne de conduite si Axelar stagne.

Maintenant, connaissant l'identité des utilisateurs de la récupération et leurs clés publiques, un simple protocole génère des partages de la clé de récupération que personne ne connaît. De plus, les utilisateurs de l'ensemble de recouvrement n'ont pas besoin d'être en ligne avant d'être appelés à récupérer via les mécanismes de gouvernance. En suivant les protocoles standards de génération de clés distribuées, chaque validateur Axelar partage une valeur aléatoire. La clé secrète de recouvrement est générée en additionnant ces valeurs. Au lieu d'effectuer les additions en clair, toutes les parts sont chiffrées par les clés publiques des utilisateurs de la récupération et ensuite additionnées de manière homomorphe (cela suppose un chiffrement homomorphe additif et une couche supplémentaire de connaissance nulle, qui sont tous deux facilement accessibles). Le résultat de ce protocole est une clef publique de recouvrement (*recovery public key*, *RPK)* et potentiellement des milliers de chiffrements (par les clés publiques des utilisateurs de recouvrement) des parts de la clé secrète correspondante $Enc_i(s_i)$ qui sont distribués à leurs propriétaires (par exemple, affichés sur la

3 Le déploiement final sur le réseau Axelar sera déterminé à une date ultérieure.

chaîne). Les contrats des ponts Axelar comprennent une option permettant de recouvrer les fonds en utilisant le RPK sous certaines conditions. Enfin, il est également possible de mettre à jour cette clé de recouvrement et même de changer l'ensemble des utilisateurs détenant ses parts sans exiger de travail de la part des actionnaires participants.

Si la chaîne X qui est connectée à Axelar s'arrête, il existe plusieurs options :

- Imposer des limites à la valeur en USD des actifs qui peuvent être transférés depuis ou vers X au cours d'une même journée. Ainsi, une chaîne malveillante X ne peut voler qu'une petite fraction de tous les actifs qui lui sont reliés avant que les validateurs d'Axelar ne le détectent, et les mécanismes de gouvernance des items suivants entrent en jeu.
- Le module de gouvernance Axelar peut être utilisé pour voter sur ce qui se passe dans ces situations. Par exemple, en cas de bug bénin et si la communauté redémarre X, la gouvernance Axelar peut décider de reprendre la connexion là où elle s'est arrêtée.
- Si de l'ETH avait été déplacé vers X, une clé de récupération Ethereum spécifique peut déterminer ce qu'il advient des actifs ETH.

6 Cross-Chain Gateway Protocol (CGP)

Dans cette section, nous expliquons le protocole de passerelle inter-chaînes et les mécanismes de routage avec deux exemples de base communs aux besoins de nombreuses applications :

Synchronisation de l'état (section 6.2). Envoyer les informations sur l'état d'une blockchain source S dans l'état d'une blockchain destination D.

(Par exemple, envoyer un en-tête de bloc Bitcoin dans la chaîne de bloc Ethereum).

Transfert d'actifs (section 6.3). Transférer un bien numérique de S à D et inversement.

(Par exemple, transférer des bitcoins de la blockchain Bitcoin vers la blockchain Ethereum, puis revenir à la blockchain Bitcoin).

Pour simplifier, nous supposons que la chaîne D *a* au moins un support minimal pour les smart contracts, mais que la chaîne S peut être n'importe quelle chaîne de bloc.

6.1 Comptes sur d'autres chaînes

Pour relier les différentes chaînes, des comptes à seuil sont créés sur chaque chaîne qui contrôlent le flux de valeur et d'informations entre elles. Pour la chaîne *Chain*, on note le compte par $Chain_{Axelar}$.

Compte Bitcoin. Pour les Bitcoins et autres chaînes de contrats non intelligents, les validateurs Axelar créent une clé ECDSA à seuil conformément à la section 5.1. Cette clé contrôle le compte ECDSA sur Bitcoin, et est l'adresse de destination où les utilisateurs envoient des dépôts. Des clés à

seuil spécifiques peuvent être créées à la demande de l'utilisateur. La clé peut être mise à jour périodiquement, et la dernière clé et les clés personnalisées peuvent être trouvées en interrogeant un nœud Axelar.

Compte de pont à seuil sur les chaînes avec des smart contracts. On note la chaîne SC. Les validateurs créent une clé seuil ECDSA ou ED25519 selon la section 5.1, en fonction du type de clé que la chaîne supporte. Nous notons cette clé PK_{Axelar}, lorsqu'il n'y a pas d'ambiguïté quant à la chaîne à laquelle nous nous référons. Cette clé contrôle un compte de smart contract sur SC, désigné par SC_{Axelar}, et toute application sur SC peut interroger SC_{Axelar} pour connaître l'adresse PK de cette clé. Ainsi, toute application sur SC peut reconnaître les messages signés par SK_{Axelar}. Le protocole doit également tenir compte des valeurs en rotation de PK_{Axelar}. Cela se passe comme suit :

1 Initialisation du SC_{Axelar} sur SC. Il stocke PK_{Axelar} comme une partie de son état, qui est initialisé comme sa valeur de genèse sur Axelar. SC_{Axelar} comprend également des règles pour la mise à jour de la PK.

2 Pour mettre à jour PK_{Axelar}, une transaction du format (*update, PKnew*) doit être soumise avec une signature du SK_{Axelar} actuel. Ensuite, le contrat définit $PK_{Axelar} = PK_{new}$.

3 Chaque fois que les validateurs mettent à jour la clé à seuil pour SC de PK^i à PK^{i+1}, Axelar demande que les validateurs utilisent SK^i pour signer (*mise à jour,* PK^{i+1}). Par la suite, cette signature est envoyée à SC_{Axelar} qui met à jour PK_{Axelar}.

6.2 Synchronisation des États

Soit q_S une question quelconque sur l'état de la chaîne S. Voici quelques exemples de ces questions :

- *« A quel tour de bloc, le cas échéant, une transaction tx est-elle apparue ? »*
- *« Quelle est la valeur d'un certain champ de données ? »*
- *« Quel est le hachage de racine Merkle de l'état entier de S au bloc 314159 ? »*

Soit a_S la bonne réponse à q_S et nous supposons qu'un utilisateur final ou une application demande que a_S soit posté à la chaîne D. Le réseau Axelar répond comme suit :

1 L'utilisateur envoie une requête q_S sur l'un des comptes du pont (qui sont ensuite récupérés par les validateurs) ou directement sur la blockchain Axelar.

2 Dans le cadre du consensus Axelar, chaque validateur doit exécuter un logiciel de nœud pour les chaînes S, D. Les validateurs Axelar interrogent l'API de leur logiciel de nœud de chaîne S *pour la* réponse a_S et rapportent la réponse à la chaîne Axelar.

3 Une fois que $> F$ validateurs pondérés rapportent la même réponse au tour R, Axelar demande aux validateurs de signer a_S.

4 En utilisant la cryptographie à seuil, les validateurs signent a_S. La signature est incluse dans le bloc $R + 11$.

5 N'importe qui peut prendre la valeur signée a_S du bloc $R + 11$ et l'envoyer à D.

6 La demande a été traitée. Toute demande sur D peut maintenant prendre la valeur signée a_S, interroger D_{Axelar} pour la dernière PK_{Axelar}, et vérifier que la signature de a_S correspond à PK_{Axelar}. Les validateurs affichent également a_S sur le compte du pont de la chaîne D, que les demandes peuvent récupérer.

6.3 Transfert d'actifs entre chaînes

Le réseau permet des transferts inter-chaînes d'actifs numériques en étendant le flux de synchronisation d'état de la section 6.2.

Une quantité suffisante de tokens adossés S est émise et contrôlée par D_{Axelar} lors de son initialisation. Supposons qu'un utilisateur demande d'échanger une quantité x de jetons sur la chaîne source S contre une quantité x *de* jetons S rattachés à la chaîne destination D, à déposer à une adresse Dw_D *de son* choix. Nous présentons le flux de travail général complet, qui prend en charge des chaînes source S quelconques, même les chaînes telles que Bitcoin qui ne supportent pas les smart contracts :

1 L'utilisateur (ou une application agissant en son nom) envoie une demande de transfert (x, w_D) sur le compte de pont à seuil qui est ensuite acheminé vers le réseau Axelar.

2 Les validateurs d'Axelar utilisent la cryptographie à seuil pour créer collectivement une nouvelle adresse de dépôt d_S pour S. Ils envoient d_S à la blockchain Axelar.

3 L'utilisateur (ou une application agissant en son nom) prend connaissance de d_S en surveillant la blockchain Axelar. L'utilisateur envoie une quantité x de S-tokens pour prendre en compte d_S via une transaction S ordinaire tx_S en utilisant son logiciel habituel pour la chaîne S.

(En raison de la propriété de seuil de la d_S, les jetons ne peuvent pas être dépensés à partir de la d_S à moins qu'un nombre seuil de validateurs ne se coordonnent pour le faire).

4 tx_S est postée sur Axelar. Les validateurs interrogent l'API de leur logiciel de nœud de chaîne S pour connaître l'existence de tx_S et, si la réponse est « vrai », envoient la réponse à la chaîne Axelar.

5 Une fois que $> F$ validateurs pondérés déclarent « vrai » pour tx_S au tour R, Axelar demande aux validateurs de signer une transaction a_D qui envoie un montant de x jetons S rattachés de $\mathrm{D}_{\mathrm{Axelar}}$ à w_D.

6 En utilisant la cryptographie à seuil, les validateurs signent a_D. La signature est incluse dans le bloc $R + 11$.

7 Tout le monde peut prendre la valeur signée a_D du bloc $R + 11$ et l'envoyer à D.

8 La demande a été traitée et une fois qu'a_D est affiché sur D, le transfert est traité.

Supposons maintenant qu'un utilisateur demande à racheter une quantité x' de jetons S enveloppés de la chaîne D à la chaîne S, pour les déposer à une adresse Sw_S de son choix. Le déroulement des opérations est le suivant :

1 L'utilisateur lance une demande de transfert (x', w_S) en déposant la quantité x' de jetons S enveloppés dans la chaîne D via une transaction D ordinaire en utilisant son logiciel habituel pour la chaîne D.

2 (x', wS) est publié sur Axelar. Les validateurs interrogent l'API de le logiciel de leur nœud de chaîne D pour connaître l'existence de (x', w_S) et, si la réponse est « vrai », rapportent la réponse à la chaîne Axelar.

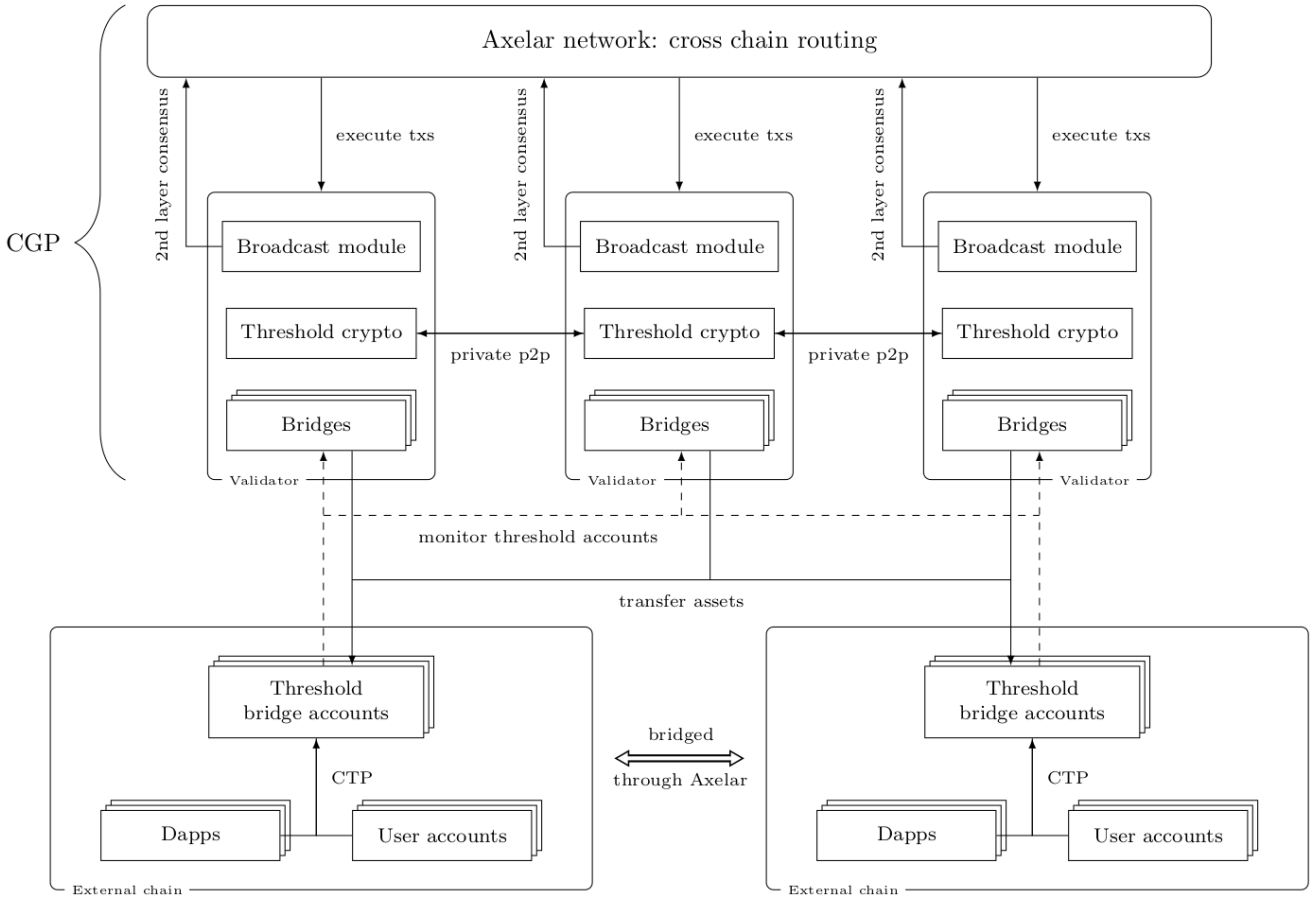

Figure 1 : Schéma des composants

3 Une fois que $> F$ validateurs pondérés déclarent « vrai » pour (x', w_S) au tour R, Axelar demande aux validateurs de signer une transaction a_S qui envoie un montant x' de tokens S de S_{axelar} à w_S.

4 En utilisant la cryptographie à seuil, les validateurs signent a_S. La signature est incluse dans le bloc $R + 11$.

5 Tout le monde peut prendre la valeur signée a_S du bloc $R + 11$ et le poster à S.

6 La demande a été traitée et une fois que l'a_S *est* affiché sur S, le transfert est traité.

Les demandes supplémentaires prises en charge par la couche de routage CGP comprennent le verrouillage, le déverrouillage ou le transfert d'actifs entre les chaînes.

Flux de transactions atomiques inter-chaînes. Selon le type de requête inter-chaînes, Axelar essaie de s'assurer que les transactions correspondantes sont exécutées sur plusieurs chaînes ou sur aucune. Pour ce faire, chaque requête peut se trouver dans l'un des états suivants dans la blockchain Axelar : *(initialized, pending, completed, timed out)*. Si un *timeout* est déclenché, la demande renvoie un code d'erreur. Certains événements de *timeout* déclenchent également un événement de *remboursement :* par exemple, si un actif d'une chaîne doit être transféré dans un actif d'une autre chaîne, si la chaîne réceptrice n'a pas traité la transaction, l'actif est remboursé à l'utilisateur d'origine.

7 Cross-Chain Transfer Protocol (CTP)

Le CTP est un protocole au niveau de l'application qui permet aux applications d'exploiter facilement les fonctionnalités inter-chaînes. Nous expliquons l'intégration en nous concentrant sur les caractéristiques de transfert d'actifs (par exemple, utilisées dans DeFi). Ces applications se composent généralement de trois éléments principaux : une interface graphique en frontal, des smart contracts sur une chaîne et un nœud intermédiaire qui enregistre les transactions entre le frontal et les smart contracts. Les frontaux interagissent avec le wallet de l'utilisateur pour accepter des dépôts, traiter des retraits, etc. Les applications peuvent tirer parti des fonctionnalités inter-chaîne en appelant des requêtes CTP analogues aux méthodes HTTP/HTTPS GET/POST. Ces requêtes sont ensuite récupérées par la couche CGP pour être exécutées et les résultats sont renvoyés aux utilisateurs.

- *Requêtes CTP*. Les développeurs d'applications peuvent héberger leurs applications sur n'importe quelle chaîne et intégrer leurs smart contracts avec des comptes de pont à seuil pour exécuter des requêtes CTP.
- *Comptes de pont à seuil*. Supposons qu'un développeur d'applications implémente ses contrats sur la chaîne A. Il référencera alors des contrats de pont à seuil pour obtenir un soutien inter-chaîne. Ce contrat permet aux applications de :
  - Enregistrer une blockchain avec laquelle elle souhaite communiquer.
  - Enregistrer les actifs sur cette blockchain qu'elle souhaite exploiter.
  - Effectuer des opérations sur les actifs, comme accepter des dépôts, traiter des retraits et d'autres fonctions (semblables, par exemple, aux appels de contrat ERC-20).

Supposons qu'une application DeFi importante, MapleSwap, réside nativement dans la chaîne A s'enregistre avec un compte de pont à seuil. Les validateurs Axelar gèrent collectivement le contrat lui-même sur la chaîne correspondante. Supposons qu'un utilisateur souhaite effectuer un dépôt

dans une paire de trading entre les actifs X et Y qui résident respectivement sur les deux chaînes. Ensuite, lorsqu'un utilisateur soumet une demande correspondante, elle est acheminée via le compte de pont à seuil vers le réseau Axelar pour être traitée. À partir de là, les étapes suivantes sont effectuées :

1 Le réseau Axelar comprend que cette application s'est enregistrée pour le support inter-chaînes entre les actifs. Elle génère la clé de dépôt en exploitant la cryptographie à seuil et le consensus pour l'utilisateur sur les chaînes A et B correspondantes.

2 Les clés publiques associées sont renvoyées à l'application et affichées à l'utilisateur qui peut utiliser son wallet habituel pour effectuer des dépôts. La clé secrète correspondante est partagée entre tous les validateurs Axelar.

3 Lorsque les dépôts sont confirmés, Axelar met à jour son répertoire inter-chaînes pour enregistrer que l'utilisateur des chaînes correspondantes a déposé ces actifs.

4 Les validateurs Axelar exécutent des protocoles multipartites pour générer une signature à seuil qui permet de mettre à jour le compte de pont à seuil sur la chaîne A où réside l'application.

5 La requête CTP est ensuite renvoyée aux smart contracts de l'application DeFi, qui peut mettre à jour son état, actualiser ses formules de rendement, les taux de change, ou exécuter d'autres conditions liées à l'état des applications.

Tout au long de ce processus, le réseau Axelar, à un haut niveau, agit comme un oracle de lecture/écriture inter-chaînes décentralisé, CGP est la couche de routage entre les chaînes, et CTP est le protocole d'application.

Demandes supplémentaires de la chaîne croisée. Le CTP prend en charge les demandes croisées plus générales entre les applications à travers des blockchains telles que :

- Effectuer des services de nommage de clé publique (*Public Key Name Services*, PKNS). Il s'agit d'un annuaire universel permettant de faire correspondre des clés publiques à des numéros de téléphone ou à des pseudonymes Twitter (quelques projets, comme Celo, proposent ces fonctionnalités sur leurs plateformes).
- Déclencheurs d'applications inter-chaînes. Une application sur la chaîne A peut mettre à jour son état si une autre application sur la chaîne B répond à un critère de recherche (taux d'intérêt $< X$).
- Composabilité des smart contracts. Le smart contract sur la chaîne A peut mettre à jour son état en fonction de l'état des contrats sur la chaîne B, ou déclencher une action pour mettre à jour un smart contract sur la chaîne B.

Ces demandes peuvent être traitées à un haut niveau puisque, collectivement, les protocoles CTP, CGP et le réseau Axelar peuvent transmettre et écrire des informations d'état arbitrairement vérifiables à travers des blockchains.

8 Résumé

Au cours des prochaines années, des applications et des actifs importants seront développés en plus des multiples écosystèmes de blockchains. Le réseau Axelar peut être utilisé pour intégrer ces blockchains dans une couche de communication inter-chaînes uniforme. Cette couche fournit des protocoles de routage et de niveau applicatif qui répondent à la fois aux exigences des constructeurs de plateformes et des développeurs d'applications. Les développeurs d'applications peuvent s'appuyer sur les meilleures plates-formes pour leurs besoins et tirer parti d'un protocole et d'une API simples pour accéder aux liquidités inter-chaînes globales, aux utilisateurs et pour communiquer avec d'autres chaînes.

Références

1 Althea peggy. https://github.com/cosmos/peggy. [Cité en p. 2.]

2 Utilisation déterministe de l'algorithme de signature numérique (dsa) et de l'algorithme de signature numérique à courbe elliptique (ecdsa). https://tools.ietf.org/html/rfc6979. [Cité en p. 5.]

3 Edwards-curve digital signature algorithm (eddsa). https://tools.ietf.org/html/rfc8032. [Cité en p. 5.]

4 Eos.io technical white paper v2. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. [Cité en p. 1.]

5 Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf. [Cité en p. 1.]

6 The near white paper. https://near.org/papers/the-official-near-white-paper/. [Cité en p. 1.]

7 Rainbow bridge. https://github.com/near/rainbow-bridge. [Cité en p. 2.]

8 Ren: A privacy preserving virtual machine powering zero-knowledge financial applications. https://whitepaper.io/document/419/ren-litepaper. [Cité en p. 3.]

9 tbtc: A decentralized redeemable btc-backed erc-20 token. https://docs.keep.network/tbtc/index.pdf. [Cité en p. 2.]

10 Thorchain: A decentralized liquidity network. https://thorchain.org/. [Cité en p. 3.]

11 Kurt M. Alonso. Zero to monero. https://www.getmonero.org/library/Zero-to-Monero-1-0-0. pdf. [Cité en p. 1.]

12 Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. A survey of ecdsa threshold signing. Cryptology ePrint Archive, Report 2020/1390, 2020. https://eprint.iacr.org/2020/1390. [Cité en p. 6.]

13 Ran Canetti, Nikolaos Makriyannis et Udi Peled. Uc non-interactive, proactive, threshold ecdsa. Cryptology ePrint Archive, Report 2020/492, 2020. https://eprint.iacr.org/2020/492. [Cité en p. 6.]

14 cLabs Whitepapers. https://celo.org/papers. [Cité en p. 1.]

15 Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergård. Fast threshold ECDSA with honest majority. In SCN, volume12238 of Lecture Notes in Computer Science, pages 382–400. Springer, 2020. [Cité en p. 6.]

16 Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In IEEE Symposium on Security and Privacy, pages 1084–1101. IEEE, 2019. [Cité en p. 6.]

17 Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. https://groups.csail.mit.edu/tds/papers/Lynch/jacm88.pdf. [Cité en p. 5.]

18 Rosario Gennaro and Steven Goldfeder. One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive, Report 2020/540, 2020. https://eprint.iacr.org/2020/540. [Cité en p. 6.]

19 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Proceedings of the 26th Symposium on Operating Systems Principles, 2017. https://dl.acm.org/doi/pdf/10.1145/3132747.3132757. [Cité en p. 1.]

20 Evan Kereiakes, Do Kwon, Marco Di Maggio, and Nicholas Platias. Terra money: Stability and adoption. https://terra.money/Terra_White_paper.pdf. [Cité en p. 1.]

21 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. https://eprint.iacr.org/2016/889.pdf. [Cité en p. 1.]

22 Chelsea Komlo and Ian Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852, 2020. https://eprint.iacr.org/2020/852. [Cité en p. 6.]

23 Jae Kwon and Ethan Buchman. Cosmos: A network of distributed ledgers. https://cosmos.network/resources/whitepaper. [Cité en pp. 1 et 2.]

24 Avalanche Team. Avalanche platform. https://www.avalabs.org/whitepapers. [Cité en pp. 1 et 2.]

25 Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. https://polkadot.network/PolkaDotPaper.pdf. [Cité en pp. 1 et 2.]