Echange de jetons inter-blockchains

VOLPE Dorian, ROTONDO Eloïse, TESTUD Romain,
DE CAMPOU Louis, JOLY Amaury
Encadrants : TRAVERS Corentin, LABOUREL Arnaud

Aix-Marseille Université, M2 Fiabilité et sécurité informatique

4 avril 2023

Remerciements

Merci à M. TRAVERS Corentin et M. LABOUREL Arnaud pour la proposition de ce sujet et son encadrement.

Résumé

Ces dernières années, un grand nombre de blockchains ont vu le jour, accompagnées d'un lot de problématiques. Parmi les questions les plus importantes figure celle des échanges entre ces blockchains. En effet le marché des cryptomonnaies augmente fortement et malheureusement toutes les blockchains ne sont pas forcément compatibles entre elles, ce qui crée une forte pression sur le marché actuel. Dans cet ouvrage, vous trouverez un résumé exhaustif des recherches menées sur les échanges cross-chain. Que ce soit en termes de protocoles ou bien d'implémentation plus techniques, nous avons essayé de regrouper le plus grand nombre d'exemples. Nous commencerons par les protocoles d'échanges centralisés qui sont les plus répandus grâce à leur facilité d'utilisation et leur rapidité. Mais ces protocoles sont souvent la source d'attaques ou bien même de failles, car le marché des échanges inter-blockchain est souvent rejoint par des entités n'ayant pas de connaissance en fiabilité. Ensuite nous continuerons par les protocoles d'échanges décentralisés qui représentent une bonne alternative aux échanges centralisés, car ils offrent une meilleure sécurité au détriment d'une complexité d'utilisation accrue. Vous allez donc pouvoir comprendre quels sont les avantages et les inconvénients de ces deux méthodes d'échanges de jetons inter-blockchain.

Table des matières

1 Introduction

Les échanges financiers sur Internet reposent presque exclusivement via les institutions financières, qui agissent comme tiers de confiance pour le traitement des paiements électroniques. En 2008, le *whitepaper* Bitcoin [38] a partagé une solution permettant à deux parties d'échanger de la monnaie électronique. La particularité de cette solution est la suppression de ce modèle de confiance par l'ajout d'une preuve cryptographique. Cela permet de s'émanciper de la centralisation exercée par ces institutions et de tendre vers la décentralisation.

La structure de donnée sous-jacente est la *blockchain*, une base de données distribuée constituée d'une chaîne de blocs liés et sécurisés par des hachés cryptographiques. Une des propriétés d'une fonction de hachage cryptographique est qu'une modification de l'entrée modifie la sortie, le haché. D'où le fait qu'une *blockchain* est considérée comme immuable , chaque bloc (sauf le premier) est lié au bloc précédent car il contient le haché de ce dernier. Toute tentative de modifier un bloc antérieur affecterait tous les blocs suivants, créant ainsi une incohérence dans la chaîne.

Dans un système centralisé, toutes les transactions sont enregistrées dans une base de données unique gérée par une entité centrale telle qu'une banque. Cette entité est responsable de l'intégrité des données et les usagers lui font confiance que toute modification involontaire ou malveillante sera détectée. Dans le cas de la *blockchain*, elle est distribuée, un réseau de noeuds connectés travaillent ensemble pour valider et enregistrer les blocs. Un noeud est un périphérique connecté à un réseau pair à pair qui stocke une copie de la *blockchain* et participe à la validation et à la propagation des blocs. En raison de sa nature distribuée, les noeuds peuvent valider et ajouter un bloc sans avoir recours à un tiers de confiance, ce qui accrût son caractère décentralisé.

Depuis, le concept de *blockchain* a grandi en popularité et il existe aujourd'hui un grand nombre de *blockchains*. Une problématique est apparue : les utilisateurs ont voulu échanger des jetons provenant de *blockchains* différentes mais ces mêmes *blockchains* ne supportaient pas les mêmes protocoles. D'où l'implémentation de protocoles d'échanges de jetons inter-*blockchain* qui a rajouté un maillon supplémentaire à sécuriser, ce dernier étant un vecteur d'attaque privilégié. L'objectif de ce rapport est de dresser un état de l'art de ces protocoles en deux temps : les protocoles d'échanges sur les plateformes centralisées puis les protocoles d'échanges décentralisés.

Glossaire

actif Dans le contexte de la *blockchain*, un actif ou jeton peut être matériel (une maison, une voiture, de l'argent, un terrain) ou immatériel (propriété intellectuelle, brevets, droits d'auteur, marque). Tout ce qui a de la valeur est traçable et échangeable sur un réseau de blockchain. 6–12, 14–16, 18, 20, 23

Bitcoin Le Bitcoin est considéré comme la première preuve de concept de la blockchain. C'est un système décentralisé de paiement et d'échange de valeur basé sur cette technologie. . 7, 13, 17, 18, 20, 24, 25

blockchain Une *blockchain* est une base de données distribuée avec une liste (c'est-à-dire une chaîne) d'enregistrements (c'est-à-dire des blocs) liés et sécurisés par des empreintes numériques (c'est-à-dire des hachages crypto). 3, 6–25, 27

bridge Un *blockchain bridge* également appelé *cross-chain bridge* est un protocole reliant deux *blockchains* entre elles de manière unilatérale ou bilatérale dans une optique d'interopérabilité. . 12, 14

CEX Centralized EXchange. 6

cross-chain Les échanges cross-chain sont des échanges entre plusieurs blockchains. Un participant utilise ses actifs dans une *blockchain* pour échanger les actifs d'autres personnes dans différentes *blockchains*. 7, 13, 18, 25

dApp abréviation de "application décentralisée". C'est une application qui fonctionne sur une blockchain ou tout autre registre décentralisé public et qui est conçu pour être autonome et transparent. 14, 15, 17

DEX Decentralized EXchange. 20

ECSDA Proof of Authority. 13

Ethereum Ethereum est un protocole d'échanges décentralisés permettant la création par les utilisateurs de contrats intelligents. Il fourni des transactions beaucoup plus rapide que Bitcoin ce qui lui donne un intérêt particulier pour la finance décentralisée.. 7, 10, 12, 13, 17–21

fonction de hachage cryptographique Une fonction de hachage cryptographique est une primitive cryptographique qui transforme un message de taille arbitraire en un message de taille fixe, appelé un haché. Une fonction de hachage cryptographique robuste doit être rapide à calculer et difficile à inverser, il doit être facile pour une fonction de hachage f de calculer une image $f(x)$ à partir de x mais il doit être difficile de calculer une pré-image $f^{-1}(y)$ à partir de y. Cette fonction doit aussi être déterministe et résistante aux collisions, deux messages différents ne doivent pas produire le même haché.. 3

HTLC Hashed Time Locked Contract. 21

IOU I Owe You. 7

noeud Un noeud d'une *blockchain* est un ordinateur connecté au réseau de cette dernière. . 7

noeud léger Un noeud léger est un logiciel permettant de connecter les noeuds des *blockchains* entre elles. . 8

Nomad Nomad est un protocole de communication inter-chaines qui permet aux utilisateurs de transférer des actifs numériques en toute sécurité entre différentes blockchains.. 12, 14–16

off chain On déclare qu'une blockchain est *off chain* lorsqu'elle est une surcouche de la blockchain principale (on parle de blockchain de niveau supérieur). La blockchain fille n'est pas vouée à être fusionner avec la blockchain mère mais juste à stocker temporairement les transactions avant de les envoyer sur la blockchain principale.. 2, 23, 24

PoA Proof of Authority. 13

PoS Proof of Stake. 13

PoW Proof of Work. 13

relay Les relays sont des applications décentralisées qui permettent la transmission d'informations de manière unilatéral entre deux blockchains distinctes . 17

sidechain Une sidechain est une blockchain secondaire qui fonctionne en parallèle d'une blockchain principale. Elles permettent de réaliser des opérations en marge de la chaîne principale, apportant ainsi plus de scalabilité et de fonctionnalités. . 19, 20

smart contract Un smart contract est une application décentralisée qui exécute automatiquement des instructions prédéfinies lorsqu'il est déployé sur une *blockchain*.. 8, 9, 12, 13, 17, 21, 24

Solana Solana est une blockchain avec des fonctionnalités de contrats intelligents qui vise à augmenter le débit au-delà de ce qui est couramment réalisé par les blockchains populaires tout en maintenant des coûts bas.. 12, 14

VAA Verified Action Approval. 13

vérificateur Un vérificateur est une entité connectée en tant que noeud au réseau de la *blockchain*. Ce dernier agit comme autorité de confiance, vérifiant les transactions sur cette dernière.. 7–9, 11, 12

Wormhole Wormhole est un protocole de communication inter-chaines basé sur les bridges et utilisant de la vérification par un réseau de guardien.. 2, 8, 12–14, 16

échange atomique Les atomic swaps (échanges atomiques) sont des transactions entre deux parties qui permettent l'échange sur deux blockchains différentes sans avoir besoin d'un tiers de confiance. Ils sont considérés comme une méthode sûre et rapide pour échanger des cryptomonnaies. . 22, 23

2 Systèmes Centralisés

2.1 Les Plate-formes d'échanges centralisés

2.1.1 Définition

Nous avons commencé nos recherches en nous intéressant en premier lieu aux moyens d'échanges les plus répandus. Cela nous a mené vers les plate-formes d'échanges centralisé ou CEX. Ce sont des plate-formes, pouvant prendre la forme d'applications web, qui permettent aux utilisateurs d'acheter, de vendre ou d'échanger des actifs numériques contre d'autres actifs numériques ou en monnaies fiduciaires. Ces plate-formes peuvent opérer sur des *blockchains* publiques ou être dédiées à une utilisation en interne. Elles sont dites centralisées car elles sont gérées par une entreprise ou une organisation hiérarchisée qui contrôle les transactions et les fonds des utilisateurs. Ces plate-formes sont donc considérées comme des tiers de confiance et agissent en tant qu'intermédiaires entre les acheteurs et les vendeurs en assurant la sécurité, la liquidité et la rapidité des transactions. C'est la solution la plus utilisée dans le secteur des actifs numériques, elles offrent très souvent une certaine variété de services tels que le prêt et/ou le *stacking*[1]. Elles proposent également un large éventail de cryptomonnaies disponibles.

2.1.2 Inconvénients et risques

Nous avons pu tout de même relever certains inconvénients et certains risques pour les utilisateurs liés à l'utilisation de ces plate-formes. Tout d'abords, les utilisateurs doivent confier leurs fonds et leurs données à un tiers en qui ils doivent avoir confiance. Cela peut exposer les utilisateurs à de la fraude, du vol ou encore du piratage si les plate-formes présentent des failles de sécurité.
Ensuite, ces plate-formes peuvent être victimes de pannes ou de saturation du réseau pouvant entraîner des retards, des pertes de transactions ou encore du déni de service bloquant ainsi l'accès aux actifs des utilisateurs. Finalement, ces plate-formes sont soumises à la réglementation et à la surveillance des autorités financières, limitant leur accessibilité dans certains pays ou régions. Ce point signifie aussi que les actifs de l'utilisateurs sont traçables par les autorités.

2.1.3 Fonctionnement

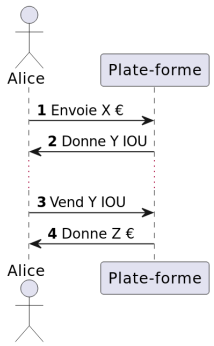


FIGURE 1 – Modélisation d'un échange

Ces plate-formes fonctionnent sur le principe de l'*order book method* (méthode du carnet d'ordre[43]), une modélisation des ordres d'achats et de vente des jetons. Un ordre étant une demande d'un utilisateur visant à réaliser une opération à un prix et une quantité donnée. Cette méthode comprend deux parties : l'offre et la demande. L'offre regroupe les ordres d'achats émis par des utilisateurs sur la plate-forme et la demande, les ordres de vente. Lors d'un dépôt, l'utilisateur s'étant au préalable enregistré sur la plate-forme,

1. Stacking : Action de verrouiller des jetons en vue de recevoir des récompenses [46]

il va déposer les fonds souhaités dans un porte monnaie. La plate-forme va ensuite crée un IOU[2] ce dernier sera échangé contre le crypto-actif souhaité lors d'un échange ou d'une vente.

Dans le cadre des échanges inter-blockchains, les plate-formes d'échanges utilisent des *bridges* reliant les différentes *blockchains*. Ces protocoles seront explicités dans la partie suivante. Cependant, nous n'avons pas pu trouver de plus amples explications quant aux fonctionnements des plate-formes, notamment les protocoles précis utilisés lors des échanges. Les documentations disponibles pour les plateformes d'échanges étant à destination des utilisateurs finaux.

2.2 Les Blockchain Bridges

2.2.1 Fonctionnement

Comme son nom l'indique, un *blockchain bridge* également appelé *cross-chain bridge* est un protocole reliant deux *blockchains* entre elles de manière unilatérale ou bilatérale dans une optique d'interopérabilité.

Dans la but de comprendre la popularité des *bridges* en tant que protocole d'interopéralité, il faut en premier lieu s'intéresser au marché de la cryptomonnaie. Actuellement Bitcoin domine en représentant 40,5% de ce dernier, suivie ensuite par Ethereum avec 19,5%. Cela laisse donc 40% du marché formé de nombreuses crytomonnaies plus petites et plus indépendantes. C'est donc naturellement, qu'une forte demande de possibilité d'échanges entre les *blockchains* ait vu le jour de la part des utilisateurs ayant plusieurs cryptomonnaies[40].

Il existe trois différentes manières de déplacer les actifs en tant que *bridge*. Tout d'abord, le mécanisme de *Lock and Mint* signifiant Verrouiller et Frapper, les actifs se trouvant sur la chaîne de départ sont verrouillés sur celle-ci pour être ensuite créés sur la chaîne destinataire. Un autre mode d'échange est celui du *Burnt and Mint*, ce dernier est très similaire à celui déjà présenté, la seule différence étant que les actifs sont directement effacés plutôt que verrouillés. Pour finir, les échanges atomiques entre chaînes (Atomic Swaps) permettent un échange direct en pair-à-pair d'actifs entre la chaîne d'origine et la chaîne de destinataire.[15]

2.2.2 Mécanisme de vérification

Comme évoqué précédemment, deux *blockchains* ne peuvent pas communiquer directement entre elles, par conséquent lors de l'utilisation d'un *bridge* les deux chaînes ne se connaissent pas et ont seulement connaissance des évènements se produisant sur leur chaîne respective. Il est donc nécessaire d'établir une relation de confiance entre les deux chaînes pour qu'elles puissent accepter de communiquer. Pour cela, les *bridges* emploient un mécanisme utilisant des vérificateurs. Un vérificateur est une entité connectée en tant que noeud au réseau de la *blockchain*. Ce dernier agit comme autorité de confiance, vérifiant les transactions sur cette dernière. Un noeud d'une *blockchain* est un ordinateur connecté au réseau de cette dernière. Un *client* est un logiciel permettant de transformer un ordinateur en noeud. [18]

Il existe un grand nombre de *bridges*, chacun avec leurs propres spécificités mais ils peuvent généralement être séparés en deux catégories les *Trusted blockchain Bridge* et les *Trustless blockchain Bridge*.

Les *Trusted Bridges* sont vérifiés de manière externe car ils utilisent un ensemble de vérificateurs tiers pour transmettre des données entre les chaînes. Ils ont pour avantage leur rapidité, leur moindre coût et la facilité d'échange avec tous les types de données acceptés par les *blockchains*. Cependant les vérificateurs externes sont moins fiables que ceux de la chaîne.[16]

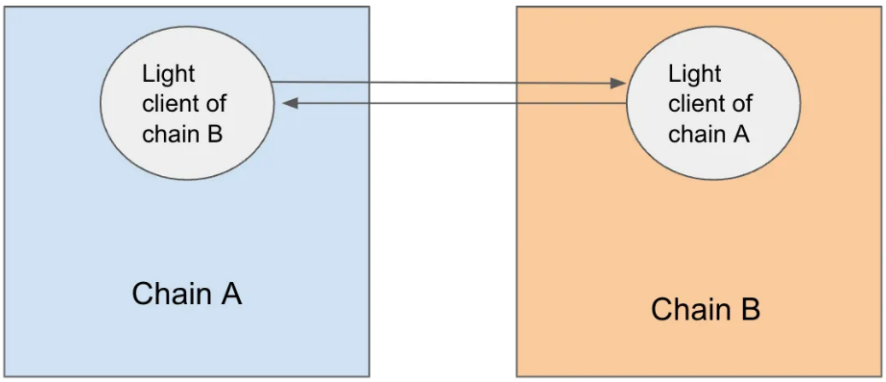
Les *Trustless Bridges* sont désignés comme *trustless* car ils dépendent des chaînes dont ils font l'intermédiaire pour transférer des données ou des actifs. Par conséquent leur niveau de fiabilité est égal à celui des

2. I Owe You, c'est la dette de la plate-forme envers l'utilisateur permettant de bloquer la valeur de la monnaie déposée par l'utilisateur[6]

blockchains et il n'est pas nécessaire de faire confiance à un ensemble de vérificateurs tiers (contrairement aux autres *bridges*). Pour cette raison, ils sont reconnus comme étant plus fiables que les *trusted bridges*.[16]

Les *bridges* peuvent également être distingués en fonction de leur type de vérification. Les plus connus sont la vérification native, externe et locale.[5]

La vérification native commence par l'utilisation d'un noeud léger. [13] Un noeud léger aussi connu sous le nom de client léger est un logiciel permettant de connecter les noeuds des *blockchains* entre elles. Il est codé comme un *smart contract* puis est employé de la chaîne de l'expéditeur vers la machine virtuelle de la chaîne destinataire. Si les vérificateurs de données de la chaîne de l'expéditeur agissent de manière correcte alors le noeud léger est vu comme véridique par la chaîne réceptionnant les actifs ou données et peut être utilisé de manière bilatérale. Un avantage de cette solution est qu'elle est reconnue comme étant celle reposant le moins sur la confiance parmi celles existantes car les chaînes ne se fient qu'à leurs propres vérificateurs pour effectuer le *bridge*. Un autre bénéfice de ce mécanisme est le fait qu'il n'utilise pas de vérificateurs tiers entre les deux *blockchains* et donc la sécuté du réseau dépend des *blockchains* elles-même (ce qui est avantageux car elles sont robutes et préparées aux attaques comme la chaîne d'Ethereum par exemple). Un désavantage de cette méthode est que le noeud léger doit être adapté aux consensus des chaînes auquelles il est attaché ce qui le rend inutilisable avec des chaînes différentes. Le noeud léger nécessite également de la maintenance en cas de changement des règles consensus (utilisées pour valider les transactions). Un autre inconvénient découlant du fait que le noeud léger est programmé de manière spécifique est que ce dernier n'est donc pas réutilisable.



Source : `https://docs.nomad.xyz/the-nomad-protocol/verification-mechanisms/native-verification`

FIGURE 2 – Mécanisme utilisant un noeud léger.

La vérification externe consiste en un ensemble de vérificateurs n'appartenant pas aux *blockchains* relayant les données entre les deux extrémités du *bridge*. Pour se faire, un certains nombre de vérificateurs doivent signer un message provenant de la chaîne d'envoi pour que le chaîne destinataire le reconnaisse comme valide. Par exemple, pour le *bridge* Wormhole 13 vérificateurs sur 19 doivent avoir signé[12]. Ce concept est une primitive cryptographique (algorithme cryptographique de bas niveau servant de base à un système de sécurité informatique) nommée le système de signature à seuil (désignée par TSS pour *Threshold Signature Scheme*)[1]. Contrairement à la vérifications native, les *bridges* vérifiés de manière externe sont faciles à développer, peuvent être réutilisés sans problèmes et leur maintenance coûte peu. Le désavantage conséquent de cette méthode est que la sécurité dépend des vérificateurs tiers du pont ce qui peut fragiliser le système car ils sont généralement moins sécurisés que ceux des *blockchains*.

La vérification externe consiste en un ensemble de vérificateurs n'appartenant pas aux *blockchains* relayant les données entre les deux extrémités du *bridge*. Pour se faire, un certains nombre de vérificateurs

doivent signer un message provenant de la chaîne d'envoi pour que le chaîne destinataire le reconnaisse comme valide. Par exemple, pour le *bridge* Wormhole 13 vérificateurs sur 19 doivent avoir signé[12]. Ce concept est une primitive cryptographique (algorithme cryptographique de bas niveau servant de base à un système de sécurité informatique) nommée le système de signature à seuil (désignée par TSS pour *Threshold Signature Scheme*)[1]. Contrairement à la vérifications native, les *bridges* vérifiés de manière externe sont faciles à développer, peuvent être réutilisés sans problèmes et leur maintenance coûte peu. Le désavantage conséquent de cette méthode est que le bon fonctionnement du système dépend des vérificateurs tiers ce qui peut le fragiliser car ils sont généralement moins fiables que ceux des *blockchains*.

Il est intéressant de noter que les *blockchains* ont également leur propre ensemble de vérificateurs sous la forme de vérificateur de données. Ces derniers sont utilisés lors de la vérification locale. Lors de la vérification locale, les chaînes se vérifient entre elles en utilisant un vérificateur en tant que représentant.

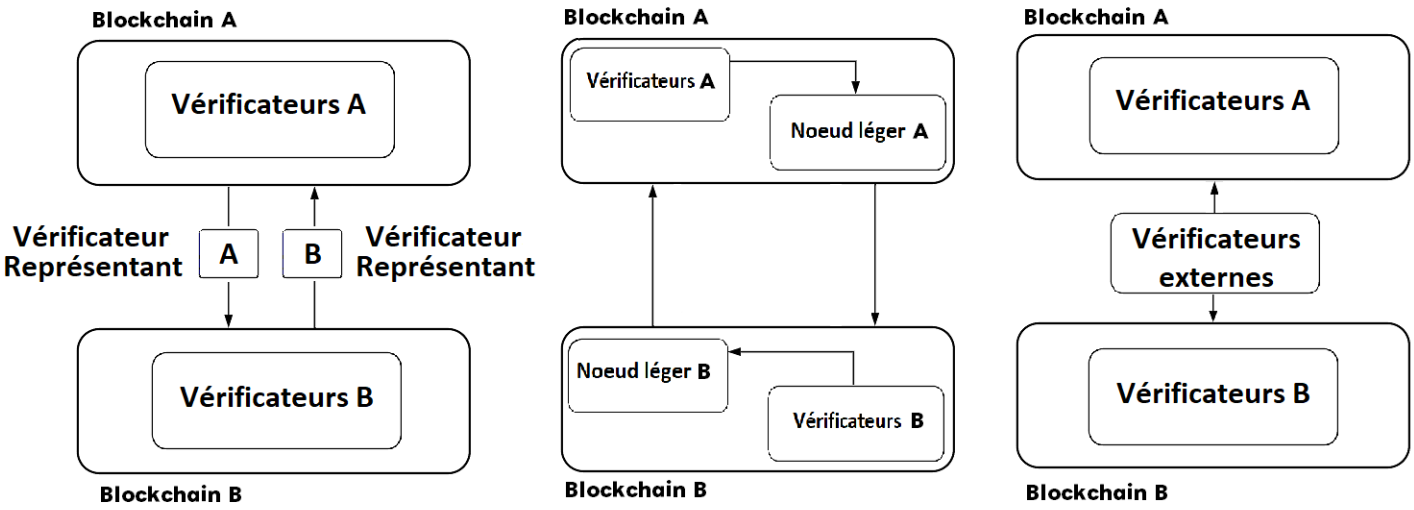


FIGURE 3 – Résumé des types de vérification. (locale, native et externe)

2.2.3 Les risques liés aux Bridges

La popularité des *blockchain Bridges* pour les échanges centralisés ne cesse d'augmenter au fils du temps mais il est important de comprendre que comme tout outil, ces derniers ne sont pas sans risques.

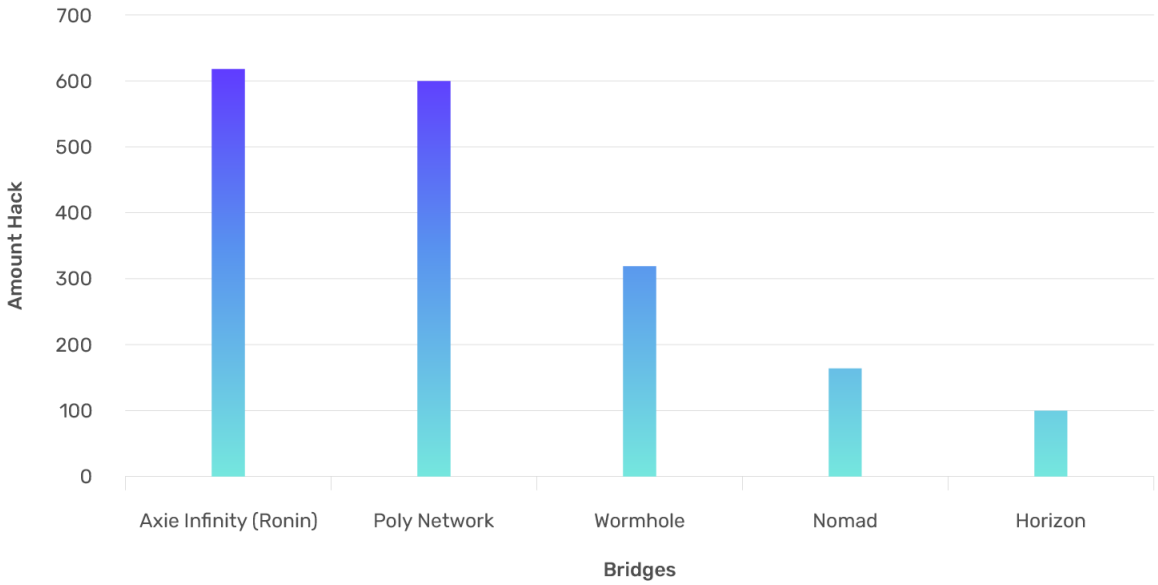
Les *bridges trustless* utilisent des *smart contracts* lors du processus d'échange dans le but de le rendre autonome afin de ne pas utiliser une entité centrale entre les deux blockchains. Cependant ces *bridges* sont néanmoins centralisés car ils utilisent les vérificateurs pour obtenir un consensus lors des transactions. Un *smart contract* étant un script écrit par un développeur, il est possible que certaines erreurs puissent s'être glissées dans le code par inadvertance ou bien qu'il existe des failles dans le programme permettant aux attaquants de le détourner pour un profit personnel. Pour minimiser ce type de risques, il est recommandé d'effectuer des audits sur les *bridges*.

Une faiblesse spécifique des *bridges trusted* repose sur le fait que les utilisateurs doivent léguer le contrôle de leurs actifs et faire confiance aux vérificateurs externes aux blockchains. Sauf que dans certains cas, ces derniers peuvent coopérer pour tromper les utilisateurs en récupérant leurs actifs puis en disparaissant comme dans les *rug pull*[17]. Ce modèle d'escroquerie peut être scindé en deux catégories : les *hard rug pull* et les *soft rug pull*[29]. Le premier cas est basé sur un piège présent dans le code d'un *smart contract* empêchant

les utilisateurs d'utiliser ou revendre les actifs frappés, seul le fraudeur en a le droit. Il peut donc en toute tranquillité revendre les actifs et récupérer l'argent. En revanche, pour les *soft rug pull*, les utilisateurs ne sont pas coincés avec des actifs verrouillés mais les fraudeurs utilisent des techniques psychologiques. En effet, les escrocs rendent attirant leur projet pour que les clients investissent et hésitent à se retirer par peur de perdre leur investissent (souvent de taille conséquente) puis les créateurs de la fraude disparaissent avec leurs actifs.

Comme vu dans la section présentant les différentes méthodes d'échange des *bridges*, ces derniers frappent les actifs désirés sur la chaîne destinataire. Certains attaquants peuvent profiter de ce mécanisme de frappe pour effectuer ce qu'on appelle une *Infinite Mint Attack*.[10] Cette attaque peut se résumer à un *hacker* générant un nombre élevé d'actifs en utilisant une faille d'un *bridge* sans verrouiller ou brûler d'actifs sur sa *blockchain*. Suite à cela, l'individu réintroduit ces actifs sur le marché ce qui fait violemment baisser leur coût ce qui engendre un risque financier systémique.

Les *blockchain Bridges* sont devenus un outil indispensable des échanges centralisés très rapidement, mais il ne faut pas oublier que ces protocoles sont relativement récents. Créés par de petites blockchains comme Syscoin et NEAR Protocol dans le but de rentre leur chaîne interopérable avec les applications décentralisées d'Ethereum, les premiers bridges datent de 2020[35]. Par conséquent, nous ne connaissons pas encore le comportement des *bridges* lorsqu'ils font face à des scénarios sortants de la norme comme des attaques réseaux, un retour en arrière sur les transactions d'une blockchain (souvent désigné par le terme *rollback*) ou bien pendant une congestion du réseau. Ces zones d'incertitudes peuvent donc être une source de risques.



Source : https://www.treehouse.finance/insights/blockchain-and-interoperability-globalization-3-0

FIGURE 4 – Pertes en millions de dollars des bridges les plus connus.

2.3 Le trilemme de l'interopérabilité

Malgré l'existence de plus d'une centaine de *bridges* différents, les développeurs et les utilisateurs voulant utiliser un *bridge* doivent faire des concessions lors de leur choix vis-à-vis des trois notions de *trustless*,

d'extensibilité (*extensible*) et de généralisation (*generalizable*).

Le mot trustless peut être traduit par «sans confiance». Si un *bridge* est caractérisé comme *trustless*, cela signifie que celui-ci possède un niveau équivalent à celui d'une ou des chaînes sous-jacentes, il n'est donc pas nécessaire de faire confiance à une entité externe aux *blockchains*. La notion d'extensibilité signifie que le *bridge* est compatible avec un grand nombre de chaînes. Un *bridge* respecte la notion de généralisation s'il est capable d'échanger n'importe quel type de données acceptées par les deux chaînes.

Pour illustrer ces termes, il est possible de les appliquer aux types de vérification appartenant aux *bridges*. La vérification locale respecte les notions d'extensibilité et de *trustless* puisque qu'elle est applicable sur tous les *bridges* peu importe les chaînes reliées et le niveau de fiabilité dépend de la chaîne la plus faible. La vérification native ne respecte pas la notion d'extensibilité car le *bridge* n'est pas réutilisable. Néanmoins elle respecte la notion de généralisation parce qu'elle est codée de manière spécifique aux *blockchains* reliées au *bridge*. Le critère basé sur la notion *trustless* est également rempli étant donné que le niveau de fiabilité dépend des vérificateurs des chaînes. La vérification externe ne respecte pas la notion de *trustless* cependant elle est fortement extensible et générale vis-à-vis des données. [40]

Suite au paragraphe précédent, il est possible de constater que les *bridges* interopérables ne respectent que deux des trois notions énoncées. Ce problème est connu sous le nom de trilemme de l'interopérabilité.

2.3.1 Une solution optimiste

Une solution proposée pour résoudre ce trilemme est un *optimistic bridge*(*bridge* optimiste) nommé vis-à-vis de sa vérification portant le même nom[4]. En effet, contrairement aux *bridges* vérifiés de manière native, locale ou externe, la vérification optimiste dépend de l'utilisation d'une latence lors de la confirmation du transfert des actifs entre les *blockchains*. Cela priorise donc la sécurité au détriment de la vivacité, puisque les transactions sont par conséquent plus lentes mais sécurisées car le *bridge* est *trustless*.

Voici plus en détails le déroulement du processus de vérification optimiste d'un *bridge*. Ceci commence par l'envoi d'une demande de transaction de la part d'un utilisateur ou d'une application décentralisée vers la *blockchain* native par le biais d'une fonction contrat. Cette demande est ensuite acceptée par un validateur (une entité équivalente à un vérificateur, la seule différence étant leur rôle) puis inscrite sur un *block* de la chaîne.[11] Ensuite un vérificateur a pour rôle de vérifier la transaction. Pour cela, le vérificateur signe le haché des données envoyées précédemment. Suite à cela, n'importe quel système de relais peut lire le haché signé sur la chaîne originelle et l'inscrire sur une ou plusieurs chaînes destinataire. Les validateurs de la chaîne destinataire valide et l'inscrivent sur leur chaîne. Cette action déclenche alors une latence de trente minutes pendant laquelle un observateur peut signaler et prouver une fraude effectuée par la chaîne native ce qui déconnectera la communication avec la chaîne destinataire.

Deux scénarios sont alors possibles. Premier cas, si aucun observateur ne se manifeste, les données de la transaction sont finalisées puis traitées par la chaîne destinataire. Les validateurs sont récompensés pour leur travail avec une partie des frais de transaction. [9] Deuxième cas, un observateur prouve une fraude pendant les trente minutes accordées. Le vérificateur ayant fraudé est pénalisé par la perte de sa récompense (qui sera obtenu par l'observateur) et son exclusion du réseau.[19]

2.3.2 Possibles faiblesses de l'optimisme et leurs solutions

Le bon fonctionnement des *bridges* optimistes dépend des chaînes auquelles il est rattaché donc tant que ces dernières sont protégées correctement la seule conséquence d'une faille du *bridge* est l'arrêt système plutôt qu'une perte de fonds comme cela peut être le cas avec les autres types de *bridge*.

Les deux acteurs principaux ayant les moyens de nuire au bon fonctionnement du *bridge* ainsi que de la transaction est l'agent vérificateur ainsi que l'observateur car ces deux rôles ont de l'influence sur cette dernière.

Le premier cas impliquant le vérificateur se nommant *Updater Fraud* (fraude du vérificateur) fut déjà mentionné lors de la présentation du fonctionnement du *bridge* optimiste. Ce dernier repose sur le fait que toute transaction doit passer par le vérificateur et que par conséquent toute fraude est originaire de ce dernier. Sinon si cela venait d'un autre participant, le vérificateur n'aurait alors pas accepté la transaction. C'est pourquoi, lors de l'intervention d'un observateur prouvant une fraude, le vérificateur est sanctionné par le retrait sur son solde d'un montant équivalent à la récompense promise et par son exclusion du réseau de la *blockchain*.

La seconde faiblesse liée aux vérificateurs est un *Updater DoS* ou déni de services de la part du vérificateur. En effet, il est possible que le processus soit interrompu si un vérificateur arrête de signer empêchant l'échange inter-chaînes de se produire. Une solution a été implémentée pour palier à cela comme la mise en place d'un système de substitution avec la présence de plusieurs vérificateurs sur une même chaîne afin de pouvoir prendre le relai en cas de manque de réponse de la part de celui étant rattaché au transfert. Pour éviter que ce scénario se produise fréquemment le vérificateur ayant manqué son tour lors de la signature (que cela soit accidentel ou voulu) est pénalisé de la même manière que le cas précédent.

Maintenant que les possibles obstacles au bon fonctionnement du *bridge* liés aux vérificateurs ont été mis en lumière, il est également possible que l'observateur ait un comportement malveillant. Effectivement, malgré l'absence de tromperie (puisque le vérificateur remplit son rôle), l'observateur peut abuser du mécanisme de déclaration de fraude pour impacter le bon déroulement du procédé.

La faculté de l'observateur à pouvoir couper la connexion s'il conteste la transaction lui permet d'effectuer un déni de service appelé *Watcher DoS*. C'est pourquoi il lui ait possible de fermer définitivement la connexion d'une transaction si ce dernier continue sans cesse de couper le processus sans raison valable. Heureusement, la fermeture ne concerne que la connexion et n'impacte en aucun cas le système du *bridge*. Cependant cette attaque semble irrationnelle en terme de ressources et de temps car l'observateur effectuant le déni de service ne gagne rien financièrement contrairement au processus habituel. En effet, si un observateur prouve une fraude correctement, ce dernier peut récupérer la récompense du vérificateur. Mais ici puisqu'aucune fraude n'est prouvée les données se trouvant sur la chaîne d'origine sont conservées et sécurisés. Cela cause seulement une perte de temps pour l'utilisateur ou l'application décentralisée voulant effectuer l'échange d'une *blockchain* à une autre.

Une réponse à ce problème actuellement mise en œuvre par le *bridge* de Nomad est la présence d'un groupe restreint d'observateurs autorisés à contester, de cette manière il est facile de connaître les observateurs malveillants. Chaque observateur possède une clé permettant de signer une attestation confirmant la présence d'une fraude dans la transaction, chaque *bridge* stocke un ensemble contenant les adresses des attestations appartenant aux observateurs autorisés. Si l'attestation reçue par le *bridge* est présente dans l'ensemble alors la connexion est rompue[14]. Sur le long terme, une proposition consistant à la mise en place de frais si l'on souhaite contester est en train d'être étudiée. Le montant doit répondre à deux contraintes : ce dernier doit être assez haut pour dissuader les observateurs malhonnêtes mais assez bas pour que ceux ayant réellement l'envie de prouver de manière valide une fraude existante puissent le faire. Dans la continuité de cette solution, il serait également possible de récupérer la signature de la déconnexion générée par l'observateur sur la chaîne originale et de le pénaliser en lui prélevant dans son solde le montant de la récompense et en le bannissant du réseau de la chaîne qu'il surveille.[4].

2.4 Wormhole

En 2017, une cryptomonnaie adossée à la *blockchain* Solana a émergée avec des caractéristiques similaires à Ethereum : *blockchain* publique, *smart contracts*.
Solana est devenue de facto une *blockchain* concurrente à Ethereum et est aujourd'hui la onzième *blockchain* en terme de capitalisation selon l'aggrégateur de marché Coinmarketcap.
Un besoin d'échanger des actifs entre les *blockchains* Ethereum et Solana est apparu, d'où l'introduction en 2020 de la première version de Wormhole. Initialement, Wormhole v1 a été concu comme un *bridge* entre

Ethereum et Solana. Depuis, Wormhole s'est développé au-delà de Solana avec le lancement d'une deuxième version en 2021 en tant que protocole générique de passage de messages.
À l'écriture de ce rapport, 22 [25] *blockchains* sont compatibles avec Wormhole dont : BNBChain, Ethereum, Moonbeam, Polygon, Solana...
Le protocole émet un message à partir d'une *blockchain* source qui est validé par un réseau de gardiens.
Le message est ensuite envoyé à la *blockchain* cible pour être traité.

2.4.1 VAA (*Verified action approval*)

Lorsqu'un *smart contract* envoie un message *cross-chain* comme un verrouillage de jetons sur une *blockchain* source et une demande de frappe de jetons sur une *blockchain* cible, celui-ci interargit avec un *core contract* [22]. Un *core contract* est déployé sur toutes les *blockchains* compatibles avec le protocole Wormhole. Tout *core contract* est observé par le réseau de gardiens. Un message Wormhole est émis grâce à la fonction *publishMessage()* prenant en entrée le *payload*. La sortie de cette fonction est un *sequence number*, un numéro d'index unique pour le message. Combiné à l'adresse du contrat de l'émetteur et à l'identifiant de la chaîne de l'émetteur, le message correspondant peut être récupéré auprès d'un nœud du réseau de gardiens.
Un message Wormhole est vérifié grâce à la fonction *parseAndVerifyVAA()* prenant en entrée le message. Selon la validité de l'entrée, la fonction retourne en sortie le *payload* ou une exception.

VAA [27] est la primitive de messagerie de base de Wormhole. Un VAA contient une en-tête ainsi qu'un *body*. L'en-tête contient l'index des gardiens ayant signés le message et la collection des signatures. L'en-tête permet au *core contract* de vérifier l'authenticité du VAA. Quant au *body*, il contient des informations comme le numéro d'identification de la chaîne Wormhole du contrat émetteur, l'adresse du contrat émetteur, le *sequence number* et le *payload*.
5 *payloads* peuvent être utilisés dont *Transfer* et *AssetMeta*, attestant les méta-données du jeton.
Le *payload AssetMeta* est obligatoire avant un premier transfert. En effet, le *payload Transfer* n'informe pas la chaîne B des meta-données du jeton verrouillé. En l'absence de connaissance de ces informations, il n'est pas possible pour la *blockchain* B de frapper la quantité correcte de jetons.
Si l'on souhaite ensuite transférer des jetons depuis une *blockchain* A vers une *blockchain* B, il faut verrouiller les jetons sur A et les frapper sur B. D'où l'utilisation du *payload Transfer* contenant des informations comme la quantité de jetons transférés, l'adresse de la chaîne d'origine et de destination, le numéro d'identification de la chaîne d'origine et de destination.. Une preuve doit être fournie que les jetons sur A sont verrouillés avant que la frappe puisse avoir lieu sur B. La signature des gardiens sur le VAA correspondant est la preuve apportée à la *blockchain* B que le verrouillage a bien été effectué et que la frappe de jetons sur B est légitime.

2.4.2 Gardiens

Un gardien [23] est une autorité de confiance qui a comme rôle de valider (par une signature) le *payload* contenu dans un VAA. Comme évoqué précédemment, le réseau de gardiens observe tous les messages *crosschain* via la surveillance des *core contracts*. Le réseau de gardiens est composé de 19 gardiens à parts égales sans chef (*leaderless*). Il est conçu pour servir d'oracle à Wormhole et est l'élement le plus critique de l'écosystème. Si une majorité de deux tiers ou plus des gardiens signent le même VAA, alors le consensus est atteint : le VAA est automatiquement considéré valide par tous les contrats Wormhole sur toutes les *blockchains* et le *payload* est actionné. Chaque gardien utilise un algorithme de signature à courbe elliptique (ECSDA). Plus précisément, chaque gardien se réfère à «secp256k1» comme paramètres de la courbe elliptique, aussi utilisé par les *blockchains* Bitcoin et Ethereum.
Le modèle de consensus utilisé est une PoA avec un système de *multisignature* M/N [24], c'est à dire que M clefs parmi N sont nécessaires pour signer un VAA. Ce modèle permet un traitement rapide des transactions et une dispense de participation monétaire, par rapport à PoW ou PoS. Cependant, il présente également des désavantages : le système est par *design* centralisé et dépend d'un petit groupe de nœuds pouvant créer un point de défaillance unique par l'utilisation commune d'une fonction vulnérable. Il est questionnable de restaurer des tiers de confiance dans le cadre d'un système devenu populaire grâce à l'absence de tels autorités. Wormhole justifie la décentralisation de leur système [23] par la présence de plusieurs parties (et non

d'un seul) dans le contrôle du réseau. Selon notre analyse, la décentralisation résulte de l'absence d'un ou plusieurs tier(s) de confiance lorsque deux parties souhaitent réaliser une transaction.

2.4.3 Relais

Un relai [26] est un processus qui délivre un VAA vers une destination. Les relais ne sont ni de confiance, ni privilégiés, ils écoutent directement le réseau de gardiens via l'intermédiaire d'un processus espion. Ces relais ne peuvent pas compromettre l'intégrité d'un VAA car une altération serait détectée lors du processus de vérification des signatures. Cependant, il n'est pas assuré qu'un relai transmette un VAA à destination, d'où une perte de disponibilité. Il est conseillé d'héberger soi-même ces relais pour supporter son application.

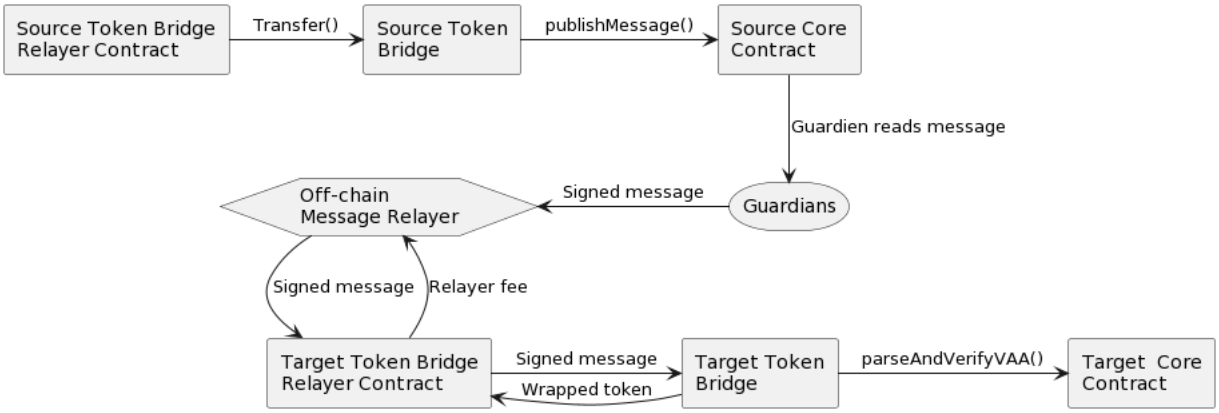


FIGURE 5 – Architecture Wormhole [21]

2.5 Analyse d'attaques

2.5.1 Mise en contexte

Les *blockchains* et leurs protocoles d'échanges ne sont pas exemptes d'attaques informatiques ou bien de défaillances. Ces attaques peuvent cibler des portefeuilles (attaques sur des *hot wallets*[3]) ou encore des *bridges*. Ce sont ces dernières qui nous ont intéressées dans le cadre de ce projet de recherche sur les échanges inter-blockchains. Les bridges, comme explicité dans la partie dédiée du rapport, sont des protocoles permettant la circulation de données entre deux *blockchains* différentes.
Nous avons, au cours de nos recherches, trouvés de nombreux cas d'attaques sur des protocoles d'échanges inter-blockchains. De manière à illustrer les types d'attaques possibles et les points critiques de ces systèmes nous allons décrire deux attaques parmi les plus importantes : Wormhole et Nomad.

2.5.2 Le cas Wormhole

Nous vous avons présenté le protocole *Wormhole* dans la partie précédente. Le 2 Février 2022, une attaque exploite une erreur d'implémentation dans une *dApp* sur la chaîne Solana [50] [55]. Pour se faire l'attaquant à réussi à contourner la vérification des signatures des gardiens en exploitant une correction de bug ayant été publié sur le code source du projet mais n'étant pas encore effective en production. Ainsi il à réussi à récupérer l'équivalent de 120 000 *ETH* en *whETH* (*Wormhole ETH*). Lors d'un transfert d'actif d'une chaîne à une autre, plusieurs étapes sont réalisées par différentes fonctions. Après la formulation de la transaction, une fonction va se charger de récupérer les signatures des gardiens dans un *SignatureSet*[4], ces dernières sont

3. portefeuille de cryptomonnaies en ligne, à différencier des *Cold Wallets*, des portefeuilles hors lignes
4. Ensemble de signatures de gardiens

ensuite vérifiées. Pour cela, une fonction nommée `verify_signature` va appeler un programme de vérification de Solana permettant l'analyse du *SignatureSet*. L'appel à ce programme se fait de la manière suivante, en utilisant le nom `sysvarinstruction` [52] dans la transaction. Dès lors que les signatures sont validées, un *VAA* peut être émis et transmis vers la *blockchain* souhaitée.
La transaction de l'attaquant étant frauduleuse, il n'aurait donc pas pu obtenir de signatures des gardiens. Pour contourner cette étape de récupération des signatures la transaction de l'attaquant était dotée d'un *SignatureSet* correspondant à une transaction antérieure. Seulement, n'étant pas pour la bonne opération cet ensemble ne peut pas être approuvé par `verify_signature`. C'est ici que l'attaquant à utilisé un défaut d'implémentation pour valider son *SignatureSet*. Comme décrit précédemment, la fonction `verify_signature` appelle un programme pour effectuer la vérification des signatures. Cependant il n'y a pas de vérification faites sur le programme appelé, l'attaquant à pu donc utiliser une adresse différente lui permettant de valider sa transaction. Avec le `SignatureSet` ainsi validé, l'attaquant a pu générer un *VAA* valide et pu déclencher une création d'actif vers son propre compte sans en avoir déposé au préalable. La correction de cette faille était contenue dans la mise à jour évoquée en début de paragraphe[53], permettant la vérification du programme appelé pour la vérification.

2.5.3 Le cas Nomad

Nomad est un protocole d'interopérabilité entre chaînes permettant de passer des actifs entre deux *blockchains* différentes. Pour fonctionner, ce protocole fait appel à des applications décentralisées opérant sur les chaînes du réseau. Une première *dApp* appelée *réplica* est déployée sur les *blockchains* recevant les messages, elle fait office de "boite de réception". Une seconde *dApp* appelée *home* est déployé sur les *blockchains* émettrices de message.
Le 1er août 2022 une attaque exploitant une erreur d'implémentation sur l'application *Réplica* a engendré une perte de 190 millions de dollars en liquidité [41] [42]. Cette attaque s'est déroulée après le déploiement d'une mise à jour, un moyen de contourner la vérification des signatures du message étant apparu. En analysant l'application *Réplica* après la mise à jour, nous pouvons voir que lors d'une initialisation, la racine des messages, appelée `_commitedRoot`, est initialisée à 0, ce signifiant que le message n'a pas encore été validé.

Listing 1 – Fonction *initialize* de *Réplica* contenant une erreur [47]

```
function initialize(
    uint32 _remoteDomain,
    address _updater,
    bytes32 _committedRoot,
    uint256 _optimisticSeconds
) public initializer {
    __NomadBase_initialize(_updater);
    // set storage variables
    entered = 1;
    remoteDomain = _remoteDomain;
    committedRoot = _committedRoot;
    // pre-approve the committed root.
    confirmAt[_committedRoot] = 1;
    _setOptimisticTimeout(_optimisticSeconds);
}
```

Dans les lignes précédentes nous observons cette affectation : `confirmAt[_commitedRoot] = 1`, le rôle de cette ligne est de pré-approuver la racine d'un message. Cette fonction est utilisée pour approuver le premier message lors du déploiement du contrat sur une *blockchain*. Or ici, la valeur de la racine a été initialisée à 0, donc cette racine devient une racine valide pour la fonction de vérification des messages. Seulement comme nous l'avons énoncé précédemment, 0 est la valeur par défaut pour un message n'ayant pas encore été vérifié. Ainsi, lors de l'émission d'un message par la fonction `process`, tout message non vérifié sera envoyé. Cette

erreur d'implémentation a permis à des pirates d'effectuer plusieurs transactions frauduleuses et de retirer l'équivalent de 190 Millions de dollars dans la réserve de liquidité du bridge de Nomad. Le contrat a été corrigé, dans une mise en ligne datant du 3 Septembre 2022, tel que la racine 0 n'est plus pré-approuvée.

Listing 2 – Fonction corrigée de l'application *Réplica* [48]

```
function initialize(
    uint32 _remoteDomain,
    address _updater,
    bytes32 _committedRoot,
    uint256 _optimisticSeconds
) public initializer {
    __NomadBase_initialize(_updater);
    // set storage variables
    entered = 1;
    remoteDomain = _remoteDomain;
    committedRoot = _committedRoot;
    // pre-approve the committed root.
    if (_committedRoot != bytes32(0)) confirmAt[_committedRoot] = 1;
    _setOptimisticTimeout(_optimisticSeconds);
}
```

2.5.4 Contre-mesures et solutions envisageables

Comme nous avons pu le voir, de nombreux cas d'attaques sont observables sur des protocoles d'échanges centralisés. Dans la plupart des cas, elles résultent de problèmes d'implémentations et autres oublis dans les codes sources des protocoles utilisés. Cela peut être expliqué par le fait que la *blockchain* est un domaine qui évolue très vite et chaque innovation technique peut rapporter des parts de marché importantes au premier arrivé. De plus, de nombreux acteurs se spécialisent dans ce domaine sans nécessairement avoir une grande culture de la cybersécurité. Il se peut donc que des erreurs d'implémentations paraissant évidentes ne soient pas relevés lors de la mise en production.
Des moyens de limiter l'apparition de tels événements sont néanmoins possibles. Tout d'abord des standards de sécurité pourraient être mis en place afin de déterminer un socle minimal à atteindre. Des audits et des analyses de sécurité peuvent être mis en place pendant la production ou après la publication des protocoles, en respectant un cycle de développement "classique".

2.6 Les limites du centralisé

Pour conclure sur les échanges centralisés, nous avons pu voir que le fonctionnement des plate-formes est relativement opaque. En effet, les protocoles employés ne sont pas ou peu explicités, nous avons trouvé peu de documentation technique au cours de nos recherches. La majeure partie de la documentation que nous avons pu trouver concernant les plate-formes d'échanges centralisé sont à destination des utilisateurs de ces plate-formes, donc à des personnes n'ayant pas forcément un bagage de connaissance dans les *blockchains*. C'est pour cela que nombreuses de nos sources concernant les plate-formes centralisés sont issues de réseaux sociaux ou d'articles web à destination de lecteurs spécialisés dans le domaine. La solution centralisée au fonctionnement le plus ouvert au public que nous avons trouvé est *Wormhole*, il s'avère que ce protocole à aussi été victime d'une attaque de grande envergure.
Nous pouvons ainsi nous questionner sur la présence de l'intermédiaire de confiance dans l'échange centralisé. En effet, la majeure partie des attaques observés sur des *Bridges* ou plus généralement sur des protocoles d'échanges centralisés, résultent de problèmes d'implémentation dans le code de ces protocoles. Le tiers de confiance est donc un point d'entrée pour les pirates vers les actifs des utilisateurs. Cette menace constante a menée à des recherches visant à soutirer cet intermédiaire des échanges inter-blockchain, donc d'employer des moyens décentralisés pour résoudre cette problématique.

3 Systèmes Décentralisés

3.1 Relay

3.1.1 Définition

Les relays décentralisés sont des applications décentralisés permettant une intéropérabilités entre les *blockchains* [45, 54, 3]. Leur but est de transmettre des informations entre des *blockchains* distinctes (par exemple, Bitcoin et Ethereum). Les relays suivent une partie de l'état de leurs chaînes connectées afin de prouver l'existence de transactions d'une chaîne à l'autre.

3.1.2 BTCRelay

BTCRelay est un *smart contract* qui stocke les en-têtes de blocs Bitcoin sur la *blockchain* Ethereum. [45, 3, 7, 20] BTCRelay utilise ces en-têtes de blocs pour construire une mini-version de la *blockchain* Bitcoin : une méthode utilisée par les portefeuilles légers Bitcoin SPV. [5] BTCRelay est open source, sans confiance et décentralisé. Il permet aux contrats Ethereum de vérifier les transactions Bitcoin sans aucun intermédiaire : en d'autres termes, les utilisateurs peuvent payer avec Bitcoin pour utiliser les dApps Ethereum. Il offre également la possibilité de relayer la transaction Bitcoin à n'importe quel contrat Ethereum et d'inspecter le dernier en-tête de bloc Bitcoin stocké dans le contrat. Ce qui offre une possibilité d'opérabilité unidirectionnelle de Bitcoin vers Ethereum.

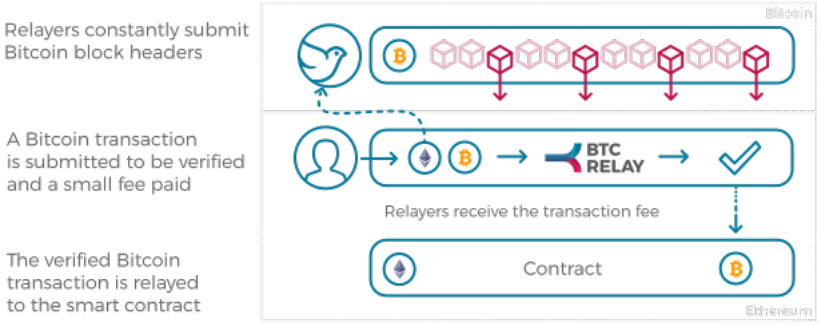


FIGURE 6 – BTCRelay

5. Bitcoin SPV signifie Simplified Payment Verification et c'est un moyen pour Bitcoin de se développer et de se propager en fonctionnant sur des petits appareils, comme les téléphones portables et les ordinateurs portables.

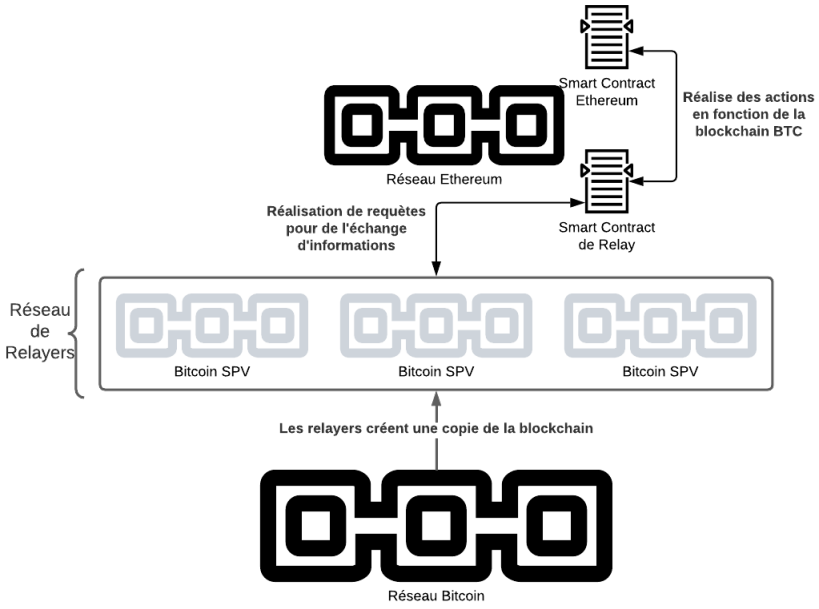


FIGURE 7 – BTCRelay

3.1.3 tBTC

Un exemple d'usage de BTCrelay pour de l'echange cross-chain est le projet tBTC, qui permet aux utilisateurs d'échanger des bitcoins contre des tokens ERC-20 représentant des bitcoins sur la *blockchain* Ethereum [31, 34]. tBTC utilise un contrat intelligent appelé Deposit qui interagit avec un ensemble de signataires qui détiennent les bitcoins en garantie. Le contrat Deposit utilise BTCRelay pour vérifier les preuves SPV des transactions Bitcoin et émettre ou brûler des actif tBTC en conséquence. Ainsi, les utilisateurs peuvent profiter des avantages de la liquidité et de la programmabilité d'Ethereum tout en conservant l'exposition au Bitcoin.
Bien que tBTC se présente comme une solution décentralisée et sans confiance pour échanger des Bitcoins contre des tokens ERC-20, il existe certains défis et limites à son fonctionnement. Par exemple, tBTC repose sur un ensemble de signataires qui doivent déposer une garantie pour sécuriser les Bitcoins verrouillés dans le contrat Deposit. Ceci expose les signataires à un risque financier permettant de limitter les risques de malveillance. De plus, tBTC nécessite que les signataires soient en ligne et disponibles pour répondre aux demandes de rachat des utilisateurs dans un délai donné. Si les signataires sont hors ligne ou malhonnêtes, les utilisateurs peuvent perdre l'accès à leurs Bitcoins ou être obligés d'attendre une longue période avant de pouvoir les récupérer. Les signataires sont choisis aléatoirement par un mécanisme appelé *random beacon*, qui pondère la sélection en fonction du montant misé par les signataires potentiels. Cela vise à éviter la collusion ou la censure entre les signataires, mais cela n'exclut pas complètement la possibilité d'attaques sybil[6] ou de corruption.

6. Une attaque sybil est un type d'attaque sur un réseau pair à pair dans laquelle un attaquant crée un grand nombre d'identités fausses et les utilise pour gagner une influence disproportionnée sur le réseau

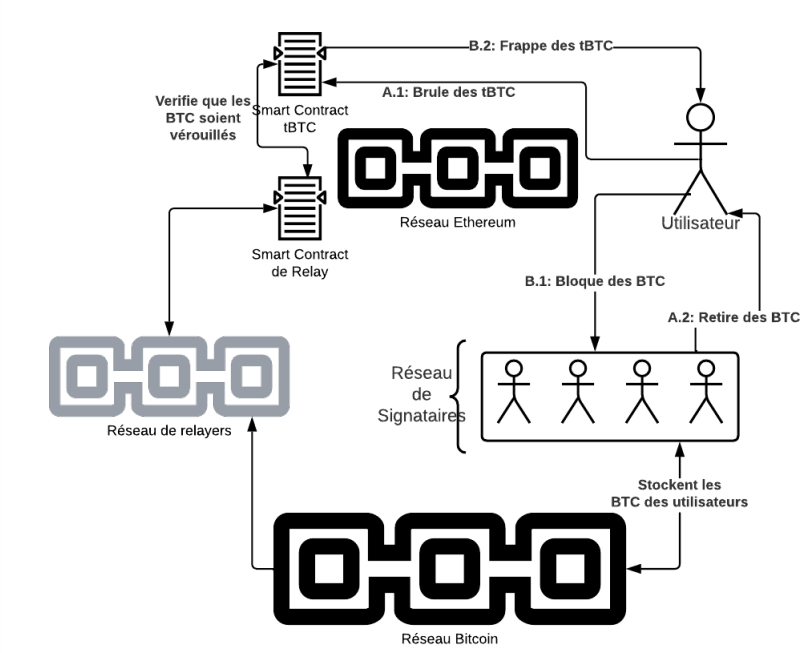


FIGURE 8 – tBTC

3.2 Sidechains

3.2.1 Définition

Les sidechains sont des *blockchains* secondaires qui fonctionnent en parallèle d'une *blockchain* principale [32, 45, 3]. Elles possèdent leurs propres caractéristiques, mais bénéficient de la communauté et de la sécurité inhérente au réseau principal pour les transactions finales qui seront inscrites sur la *blockchain* principale. Les sidechains permettent de réaliser des opérations en marge de la chaîne principale, apportant ainsi plus de scalabilité et de fonctionnalités. Par exemple, certaines sidechains sont compatibles avec l'Ethereum Virtual Machine (EVM) et peuvent donc porter des applications Ethereum.

3.2.2 Zendoo

Zendoo est une plateforme de création de sidechains interopérables avec la *blockchain* Horizen [28, 3]. Elle utilise un protocole de transfert cross-chain vérifiable par zk-SNARK [7], qui permet de garantir la sécurité et la décentralisation des communications entre la chaîne principale et les sidechains. Les sidechains Zendoo peuvent avoir des caractéristiques différentes de la chaîne principale, comme le mécanisme de consensus, le modèle comptable ou la structure des données. Elles peuvent même ne pas être des *blockchains* du tout, tant qu'elles respectent le protocole de transfert cross-chain. Zendoo offre ainsi une grande liberté aux développeurs pour créer des applications sur mesure sans compromettre la scalabilité ou la sécurité du réseau Horizen.
De ce fait, Zendoo facilite l'échange de jetons entre différentes chaînes de blocs, sans passer par des intermédiaires centralisés qui perçoivent des commissions. Les utilisateurs peuvent ainsi bénéficier d'une plus grande liquidité et d'une meilleure efficacité dans leurs transactions cross-chain.

7. zk-SNARK est un acronyme qui signifie « Zero-Knowledge Succinct Non-Interactive Argument of Knowledge ». Il s'agit d'une preuve cryptographique qui permet à une partie, le prouveur, de prouver à une autre partie, le vérificateur, qu'une affirmation sur des informations détenues secrètement est vraie sans révéler les informations elles-mêmes.

3.2.3 Contrainte technique des sidechains

La mise en place de sidechains implique une contrainte technique majeure : la création d'un pont bi-directionnel (*two-way bridge*) entre la chaîne principale et la sidechain. Ce pont permet de transférer des actifs entre les deux chaînes, en respectant un taux de change prédéfini et en garantissant la conservation du nombre total d'actifs. Cependant, ce pont nécessite une coordination entre les deux chaînes, ce qui peut poser des problèmes de sécurité, de performance ou de compatibilité. Par exemple, il est difficile d'utiliser des sidechains avec des *blockchains* comme Ethereum ou Bitcoin, car elles n'ont pas le même algorithme de consensus, le même modèle comptable ou la même structure de données que les sidechains. Il faudrait donc adapter ces *blockchains* pour qu'elles puissent communiquer avec les sidechains, ce qui impliquerait des modifications importantes dans leur protocole.

3.3 Reserves de Liquidités et Echangeurs Décentralisés

3.3.1 Definition

Les réserves de liquidité sont des marchés automatisés qui permettent aux utilisateurs de fournir des liquidités pour les échangeurs décentralisés (DEX) et de remporter une comission à chaque transaction [32, 3, 2]. Les fournisseurs de liquidités déposent des fonds dans une réserve de liquidité et reçoivent des jetons LP [8] en retour. Les jetons LP représentent une part de propriété dans la réserve de liquidité et peuvent être utilisés pour retirer des fonds de la réserve. Les réserves de liquidité sont un concept clé de l'écosystème DeFi. Il permettent la mise en place d'échangeurs décentralisés qui donne la possibilité aux utilisateurs d'échanger des actifs sans avoir besoin d'un intermédiaire centralisé.
A chaque échange réalisé via la réserve, les possesseurs de liquidités recoivents des récompenses qui sont les frais d'échanges des utilisateurs. Les récompenses sont généralement des jetons de gouvernance ou des jetons de protocole. Les réserves de liquidité se régulent ainsi en ajustant les frais de transaction en fonction de l'offre et de la demande. Si la demande pour une réserve de liquidité particulier est élevée, les frais de transaction augmentent pour encourager les fournisseurs de liquidités à déposer plus de fonds dans la réserve. Si la demande est faible, les frais de transaction diminuent pour encourager les utilisateurs à échanger des actifs sur la réserve de liquidité.
Une réserve de liquidité repose sur un smart contract et bénéficie ainsi de la décentralisation et de la sécurité de la blockchain sur laquelle il repose.

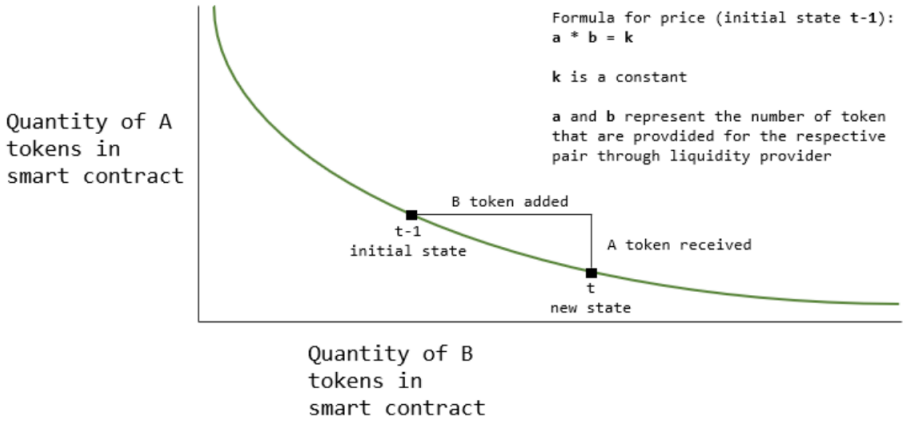
3.3.2 Exemple : PancakeSwap

PancakeSwap est une plateforme d'échange décentralisée (DEX) qui repose sur la blockchain Binance Smart Chain. [2] Elle permet aux utilisateurs d'échanger des cryptomonnaies de manière décentralisée. Le jeton natif de la plateforme PancakeSwap, le CAKE, est utilisé pour la gouvernance du protocole. Ainsi, grâce à lui, vous pouvez voter pour des propositions soumises par la communauté. La sécurité de PancakeSwap est assurée par un ensemble de smart contracts permettant de sécuriser les transactions et les échanges de manière décentralisée. Les réserves de liquidités sont un élément clé de PancakeSwap. Ils permettent aux utilisateurs de fournir des liquidités à la plateforme et de recevoir des récompenses en échange. Les réserves de liquidités sont également utilisés pour déterminer le prix des actifs sur la plateforme.

3.3.3 Limitations

Les réserves de liquidités présentent certaines limites [8]. Tout d'abord, les réserves de liquidités sont vulnérables aux attaques de liquidités. Les attaques de liquidités sont des attaques dans lesquelles un utilisateur manipule le prix d'un actif en ajoutant ou en retirant une grande quantité de liquidités d'une réserve. Cela peut entraîner une baisse significative du prix de l'actif et des pertes pour les fournisseurs de liquidités. De plus, les réserves de liquidités peuvent être affectés par des problèmes de liquidité. Si une réserve de liquidités n'a pas suffisamment de liquidités, il peut être difficile pour les utilisateurs d'acheter ou de vendre des actifs

8. Liquidity Provider Token



Source : An introduction to decentralized finance (defi) [32]

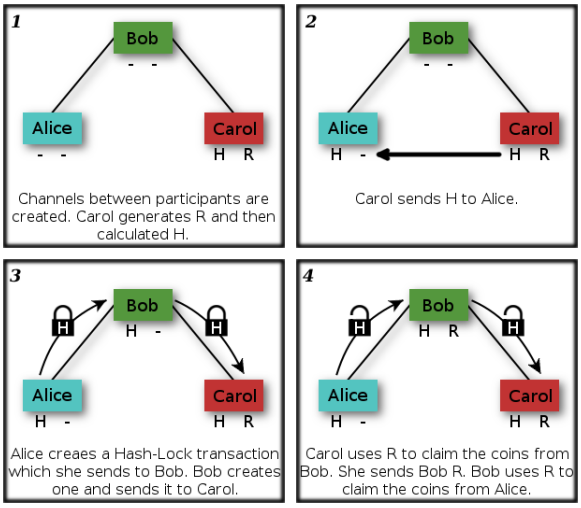
FIGURE 9 – Marché de la reserve de liquidité

sur la plateforme. Enfin, les réserves de liquidités peuvent être affectés par des problèmes de sécurité. Si une réserve de liquidités est compromise, les utilisateurs peuvent perdre leurs fonds.
Un exemple d'attaque sur une réserve de liquidité est la CVE-2021-3006 [51, 49]. La CVE-2021-3006 est une vulnérabilité de sécurité qui a été exploitée en décembre 2020 et janvier 2021. Elle concerne un manquement de controle d'accès dans l'implémentation du *smart contract* pour une réserve de liquidité en lien avec Seal Finance (Seal), un jeton Ethereum. Cette vulnérabilité permet une manipulation des prix ayant permis à l'attaquant de réaliser une plus-value artificiel sur ses jetons.

3.4 Atomic swaps et HTLC

3.4.1 HTLC

Un HTLC est un type de *smart contract* utilisé dans les applications *blockchain* qui réduit le risque de contrepartie en créant une garantie basée sur le temps et une autre sur un verrou cryptographique[39]. Ils reposent sur deux primitives fondamentales : le verrou de hachage et le verrou temporel.
Premièrement le verrou de hachage (ou bien *hashlock*) : Ce dernier va fonctionner comme une assurance qui va couvrir les participants à la transaction, en leur garantissant que chaque modification de l'échange soit réalisée d'un commun accord de toutes les parties. Cela est mis en place grâce à un partage du hachage de la clef privée de la transaction, ce partage va donc impliquer que le canal de paiement mis en place ne puisse être ouvert ou fermé qu'avec l'accord de tous les participants.
Ensuite le verrou temporel (ou bien *timelock*) va laisser un laps de temps prédéterminé afin que chaque côté du canal puisse prendre ses dispositions avant d'effectuer la transaction. Si jamais ce temps est écoulé alors la clause secondaire du contract (elle aussi prédéterminée) sera exécuté. Dans la plupart des cas cette clause secondaire va simplement fermer le canal de paiement et terminer la transaction sans la diffuser sur la *blockchain*.
L'intérêt principal de ce type de contrat est qu'il permet d'effectuer facilement des échanges hors chaines. En effet, vu que les participants s'accordent au préalable sur le montant de la transaction ils peuvent tout à fait réaliser l'échange sur des *blockchains* différentes, sous réserve que ces *blockchains* aient des mécanismes de *smarts contracts*.



Source : `https://blog.scottlogic.com/2016/06/16/bitcoin-redeem-scripts.html`

FIGURE 10 – Utilisation basique d'un HTLC

3.4.2 Atomic swaps

Les *atomics swaps* ou bien échange atomiques sont des échanges effectués entre deux *blockchains*[30]. On les appelle "atomiques" car il respecte une primitive essentielle à leur fonctionnement et à leur principe d'utilisation : l'échange est insécable, c'est-à-dire qu'a son issue, soit il s'est pleinement exécuté, soit l'état antérieur à l'échange est préservé. Cette primitive permet de garantir que les participants ne seront jamais dans des états jugés "inacceptables".
En effet les échanges atomiques partent du postulat que si une partie suit le protocole à la lettre alors, elle ne sera jamais perdante. A contrario les parties ne suivant pas le protocole par malveillance ou par erreur, prennent le risque de perdre leurs mises de départ.

Lors d'un échange atomique les HTLC vont être utilisés de manière centrale car ils vont permettre d'apporter les garanties que nous avons vue dans la partie précédente. Il faut voir les échanges atomiques comme une généralisation de l'utilisation des HTLC. Lors d'un échange atomique les deux *blockchains* mises en relation doivent pouvoir communiquer et échanger sur une interface commune comme un portefeuille multi signature ou un canal d'échange mis en place spécialement pour cette transaction.

Si nous nous penchons sur le déroulement d'un échange atomique à deux parties nous pouvons déduire les étapes suivantes :

1. Les deux parties participent à l'échange, créant un HTLC (sur chaque *blockchain*) dans lequel les fonds sont bloqués.
2. Ils échangent les informations nécessaires pour effectuer une transaction sur la *blockchain* de l'autre partie.
3. Ils vérifient l'exactitude des informations reçues et signent la transaction.
4. Les événements sont envoyés sur les *blockchains* respectives.
5. Les transactions sont confirmés par les *blockchains* respectives et les fonds sont débloqués.

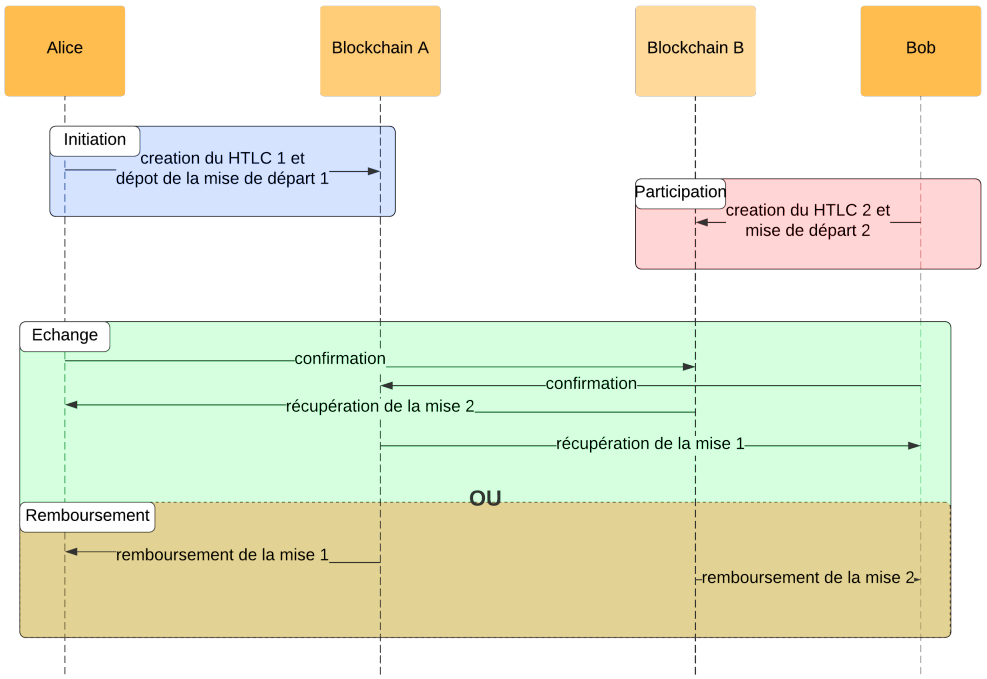
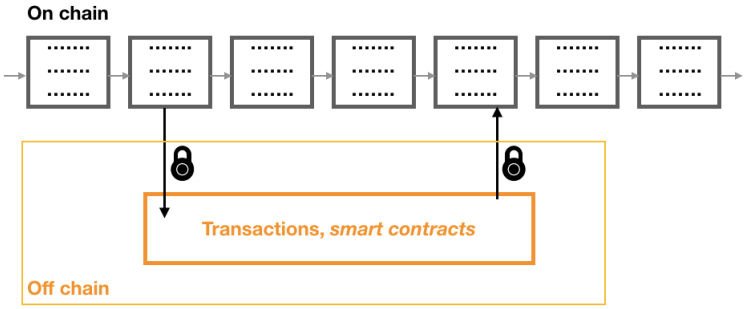


FIGURE 11 – Déroulement simplifié d'un échange atomique

3.5 Les échanges *off chain*

3.5.1 Fonctionnement des échanges *off chain*

Les échanges *off chain* sont des transactions d'actifs dont on déplace la valeur en dehors de la *blockchain* mère. En d'autres termes, il s'agit de la négociation de la valeur d'un actif crypto en dehors de la *blockchain* source. Les transactions *on-chain* sont les échanges d'actifs qui ont lieu sur la *blockchain* elle-même. Ils sont des échanges d'actifs internes au réseau et ont donc leurs propres grands livres, authentification et coûts qui ont lieu parmi la *blockchain*. Les transactions *off chain* sont effectués en dehors du réseau principal de la



Source : https://www.cryptoencyclopedie.com/single-post/\gls{blockchain}-comprendre-distinction-on-chain-\gls{offchain}

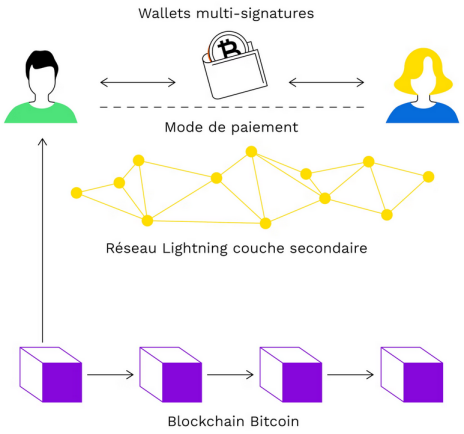
FIGURE 12 – Architecture *off chain*

blockchain, ce qui entraîne souvent un processus moins cher et plus rapide pour l'utilisateur. Tout cela se fait dans un objectif bien précis qui est de garder le maximum de sécurité et de fiabilité de la *blockchain* mère tout en essayant d'améliorer la vitesse d'échange et les frais de transaction. Afin d'illustrer comment

fonctionne les échanges *off chain* nous allons nous concentrer sur une implémentation en particulier : le réseau Lightning.

3.5.2 Le réseau lightning

Le réseau lightning (provenant de l'anglais *Lightning Network* et abrégé par LN) fait partie des processus d'échanges que l'on qualifie d'*off chain* car il se déroule en dehors de leur *blockchains* principale, ce protocole est un cas concret de cette famille. Ce réseau est une couche de protocole de paiement construite au-dessus de la *blockchain* Bitcoin qui vise à accélérer les transactions Bitcoin, à réduire les coûts et à améliorer la mise a l'échelle [44]. Il permet aux utilisateurs de créer des canaux de paiement *peer-to-peer* bidirectionnels pour effectuer des transactions en dehors de la *blockchain* principale (voir Figure 13), permettant des transactions plus rapides, moins chères et plus privées.
Les transactions sur le réseau lightning sont effectuées avec des smart contracts qui permettent aux utilisateurs de transférer des fonds à des tiers sans l'approbation de la *blockchain* principale ou bien d'un tiers de confiance. Les canaux de paiement flash sont créés en verrouillant temporairement des fonds sur une adresse multi-signature (voir Figure 14), qui est ensuite utilisée pour envoyer des transactions à d'autres participants du réseau.
Le réseau lightning utilise un système de routage pour acheminer les paiements entre les participants, en utilisant les canaux de paiement existant pour créer des itinéraires optimaux entre les participants. Les frais d'utilisation de ce réseau sont généralement bien inférieurs aux frais de transaction sur la *blockchain* principale, ce qui en fait une option plus attrayante pour les petites et moyennes transactions.



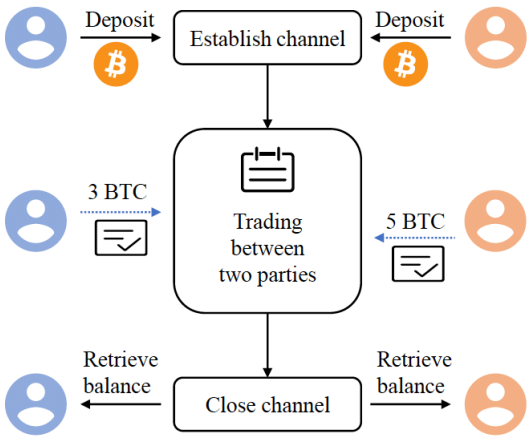
Source : `https://www.bitpanda.com/academy/fr/lecons/quel-est-le-role-du-lightning-network-pour-bitcoin/`

FIGURE 13 – Le réseau Lightning par rapport a la *blockchain* Bitcoin principale

Voici le déroulement classique d'un échange sur le réseau Lightning :
Alice veut échanger avec Bob à travers le réseau Lightning (sur la même ou bien sur des *blockchains* différentes). Alice va donc ouvrir un portefeuille multi-signature[9] avec Bob. Ils vont mettre une "mise de départ" qu'ils vont déposer dans le portefeuille mentionné précédemment (le canal d'échange aura donc la taille des deux mises de départ cumulées). Ensuite ils vont effectuer leurs transactions dans ce portefeuille qui va être mis à jour à chaque transaction.

Il est fondamental de noter que chaque transaction invalide les précédentes (à l'exception de la toute première). Lorsque l'un des participants souhaite terminer l'échange il va publier sur la *blockchain* Bitcoin la dernière transaction effectuée. Ce mécanisme de transaction intermédiaire est mis en place afin d'éviter qu'un

9. Portefeuille qui nécessite plusieurs signatures pour effectuer une transaction



Source : https://issam.ma/jekyll/update/2022/03/01/bitcoin-must-die-5.html

FIGURE 14 – Déroulement d'un échange sur le réseau Lightning

des deux acteurs de l'échange ne puisse "s'échapper" de l'échange, car chacune des parties va signer avec sa clef privée la dernière transaction.
Aussi tous ces échanges se passent à l'intérieur du réseau Lightning, la *blockchain* principale n'a donc aucune idée du nombre exact de transactions intermédiaires effectuées puisque seule une transaction sera envoyée sur la *blockchain* principale. Cela permet d'éviter d'engendrer des frais de transactions inutiles ou bien de congestionner la *blockchain* principale.

Enfin il existe aussi une autre caractéristique intéressante du réseau Lightning : le *channel hopping*. Cette propriété permet de mettre en place une transitivité entre les parties du réseau. Par exemple si Alice a déjà échangé avec Bob et Bob à déjà échangé avec Carol alors Alice pourra échanger avec Carol et vice-versa.

3.5.3 Amélioration du réseau lightning par le MIT

Même si le réseau Lightning permet déjà de faire des échanges *cross-chains*[33], il est de la responsabilité des blockchains de s'adapter au réseau afin de pouvoir échanger sur ce dernier. Le MIT a donc proposé un ouvrage scientifique[37] afin de proposer une interface plus simple à la fois pour les blockchains mises en cause et les utilisateurs souhaitant utiliser le réseau. Leur *Proof of Concept*[36] se base sur un fork du réseau Lightning appelé "lit" leur but est de contrer le problème de mise à l'échelle du réseau lightning grace à leurs améliorations. De même ils cherchent à inclure dans le réseau lightning des blockchains qui n'ont pas de valeurs monétaires (informations, NFT,etc.).
Leur programme rajoute 4 commandes au réseau Lightning mentionné précédemment, ce qui va permettre d'avoir une interface plus simple pour les acteurs des échanges :

— *Price* : Commande qui permet aux utilisateurs de rechercher la valeur marchande actuelle en USD de toute crypto-monnaie disponible sur le site *coinranking.com*.

— *Compare* : Permet de comparer la valeur en USD de deux cryptomonnaies différente et de faire l'équivalent d'une cryptomonnaie à une autre. Cette commande utilise la commande *Price* mentionnée au-dessus.

— *Exchange* : La commande *Exchange* est la première des deux commandes développées pour effectuer des échanges inter-chaînes. Elle permet de spécifier les montants mis en jeu lors de l'échange. Si Alice veut échanger 20 Bitcoin (à travers le canal 1) pour 10 Litecoin (à travers le canal 2) avec Bob, elle entrerait "Exchange 1 20 2 10 ».

— *Respond* : À ce stade, Bob a une minute pour décider si pour accepter la demande d'échange d'Alice. Il peut faire l'une des trois choses suivantes : répondre oui et accepter l'échange, répondre non et le refuser, ou laisser la demande *timeout*.

Malheureusement ce protocole ne supporte pas encore le *channel hopping* malgré le fait qu'il se base sur le réseau lightning. De même il ne supporte pas de sécurité au niveau des valeurs échangées, même si une commande permet de vérifier le prix en USD de l'échange, et cela tout particulièrement pour les chaines non monétaires. Cette amélioration du MIT laisse donc libres les valeurs mises en place lors des échanges.

4 Conclusion

4.1 Centralisé

Les échanges centralisés sont les plus utilisés dans les environnements *blockchain*. Ils offrent en effet des avantages tels qu'une grande accessibilité ainsi qu'un large panel de produits financiers connexes. De plus, le marché de l'échange centralisé dispose d'une grande diversité d'acteurs et de plateformes, ce qui laisse un choix à l'utilisateur et a pour effet de stimuler le marché. Cependant, ils présentent aussi des inconvénients. Premièrement, les acteurs de l'échange centralisé entretiennent une certaine opacité de leurs algorithmes et de leurs protocoles, ce qui est un frein à l'utilisateur final pour comprendre et analyser les transactions réalisées. Ceci l'oblige donc à faire confiance en la plateforme qu'il utilise. De plus, l'utilisation d'un tiers de confiance peut entraîner des coûts supplémentaires pour les utilisateurs et a pour effet de rajouter un point critique en terme de sécurité. Enfin, ces plateformes ont des obligations légales concernant la collecte de données personnelles ce qui entraîne une violation de la vie privée des utilisateurs.
En fin de compte, les solutions centralisées ne s'inscrivent pas dans l'idéologie initiale de la *blockchain*, qui est basée sur la décentralisation et la transparence.

4.2 Décentralisé

Les solutions décentralisées offrent une alternative aux échanges centralisés en permettant aux utilisateurs d'échanger directement entre eux sans avoir besoin d'un tiers de confiance. Cela signifie que les utilisateurs ont un contrôle total sur leurs transactions et qu'ils n'ont pas à faire confiance à une plateforme tierce pour effectuer des transactions. Les solutions décentralisées offrent également d'autres avantages. En terme de transparence dans un premier temps. En effet, il est simple pour l'utilisateur de visualiser exactement comment les transactions sont effectuées et il est souvent simple de construire un écosystème communautaire autour d'un projet décentralisé.
Les systèmes décentralisés offrent aussi une plus grande fiabilité de leurs services, puisqu'étant distribués ces systèmes sont plus résilients aux pannes ce qui offre une plus grande garantie dans les transactions.

Cependant, les solutions décentralisées ont également des inconvénients. Tout d'abord, elles peuvent être difficiles à prendre en main pour les utilisateurs qui ne sont pas familiers avec la technologie *blockchain*. De plus, elles peuvent avoir un intérêt économique moins important pour les investisseurs car il est plus difficile de créer un modèle économique autour de ces solutions.

4.3 Générale

Pour conclure le sujet, nous constatons un certain flou entre ce qui est considéré comme centralisé et décentralisé. Durant nos recherches, nous avons pris comme référentiel une certaine définition de ce que nous considérons comme centralisé ou décentralisé. Il nous parait important de souligner que l'usage de ces termes présente un argument *marketting* important, et beaucoup d'acteurs remanient la définition de décentralisé pour correspondre avec leurs produits.

TABLE 1 – Tableau Récapitulatif

		Décentralisation	Accessibilité
Centralisé	Plateformes d'échanges	− − −	+ + +
	Blockchains Bridges	−	++
Décentralisé	Relay	+	−−
	Sidechains	++	−−
	Reserves de Liquidités	+ + +	+ + +
	HTLC	+ + +	− − −
	Off-chain	++	+

Références

[1] Binance Academy. *Explications sur les Signatures à Seuil*. 2019. URL : https://academy.binance.com/fr/articles/threshold-signatures-explained.

[2] Patrick Augustin, Roy Chen-Zhang et Donghwa Shin. "Yield Farming". In : *Available at SSRN 4063228* (2022).

[3] Rafael Belchior et al. "A Survey on Blockchain Interoperability : Past, Present, and Future Trends". English. In : *ACM computing surveys* 54.8 (2022), p. 1-41.

[4] Arjun Bhuptani. *Optimistic Bridges*. 2022. URL : https://blog.connext.network/optimistic-bridges-fb800dc7b0e0.

[5] Arjun Bhuptani. *The Interoperability Trilemma*. 2021. URL : https://blog.connext.network/the-interoperability-trilemma-657c2cf69f17.

[6] Solomon Brown. *What is an IOU ?* Sept. 2021. URL : https://freewallet.org/blog/what-is-an-iou/.

[7] BTCRelay. *BTCRelay*. http://btcrelay.org/. 2022.

[8] Giulio Caldarelli et Joshua Ellul. "The blockchain oracle problem in decentralized finance—a multivocal approach". In : *Applied Sciences* 11.16 (2021), p. 7572.

[9] Contributor. *What are Crypto Transaction Fees and How they Work*. 2021. URL : https://phemex.com/academy/what-is-bitcoin-transaction-fees.

[10] ChainLink Docs. *Cross-chain bridges and associated risks*. URL : https://docs.chain.link/resources/bridge-risks/#risks.

[11] Nomad Docs. *Background on Verification*. 2022. URL : https://docs.nomad.xyz/the-nomad-protocol/verification-mechanisms/background-on-verification.

[12] Nomad Docs. *External Verification*. 2022. URL : https://docs.nomad.xyz/the-nomad-protocol/verification-mechanisms/external-verification.

[13] Nomad Docs. *Native Verification*. 2022. URL : https://docs.nomad.xyz/the-nomad-protocol/verification-mechanisms/native-verification.

[14] Nomad Docs. *Running a Watcher*. 2022. URL : https://docs.nomad.xyz/developers/node-operators/running-a-watcher.

[15] Ethereum. *BRIDGES*. 2022. URL : https://ethereum.org/en/developers/docs/bridges/#how-do-bridges-work.

[16] Ethereum. *BRIDGES*. 2022. URL : https://ethereum.org/en/developers/docs/bridges/#trade-offs.

[17] Ethereum. *BRIDGES*. 2022. URL : https://ethereum.org/en/developers/docs/bridges/#risk-with-bridges.

[18] Ethereum. *FREQUENTLY ASKED QUESTIONS*. 2022. URL : https://ethereum.org/fr/developers/docs/consensus-mechanisms/pos/faqs/#what-are-nodes-clients-and-validators.

[19] Ethereum. *PROOF-OF-STAKE REWARDS AND PENALTIES*. 2022. URL : https://ethereum.org/fr/developers/docs/consensus-mechanisms/pos/rewards-and-penalties/.

[20] ethereum. *btcrelay*. https://github.com/ethereum/btcrelay. 2022.

[21] Wormhole Foundation. *Wormhole : Architecture*. https://github.com/wormhole-foundation/example-token-bridge-relayer/blob/main/docs/design.png.

[22] Wormhole Foundation. *Wormhole : Core contract*. https://book.wormhole.com/wormhole/3_coreLayerContracts.html.

[23] Wormhole Foundation. *Wormhole : Guardian network*. https://book.wormhole.com/wormhole/5_guardianNetwork.html.

[24] Wormhole Foundation. *Wormhole : Multisignature*. https://docs.chainswap.com/mechanism/chainswap-solana-initiative.

[25] Wormhole Foundation. *Wormhole : Network*. https://wormhole.com/network/.

[26] Wormhole Foundation. *Wormhole : Relayer*. https://book.wormhole.com/wormhole/6_relayers.html.

[27] Wormhole Foundation. *Wormhole : VAA*. https://book.wormhole.com/wormhole/4_vaa.html.

[28] Alberto Garoffolo, Dmytro Kaidalov et Roman Oliynykov. "Zendoo : A zk-SNARK verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains". In : *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2020, p. 1257-1262.

[29] Hacken. *Rug Pull – Top Crypto Scams & How to stay safe from rug pulls*. 2023. url : https://hacken.io/discover/rug-pull-explained/.

[30] Maurice Herlihy. "Atomic cross-chain swaps". In : *Proceedings of the 2018 ACM symposium on principles of distributed computing*. 2018, p. 245-254.

[31] Felix Hildebrandt. "Tokenization and the Symbiosis between Blockchains". In : *Konferenzband zum Scientific Track der Blockchain Autumn School 2020*. Hochschule Mittweida. 2020, p. 14-20.

[32] Johannes Rude Jensen, Victor von Wachter et Omri Ross. "An introduction to decentralized finance (defi)". In : *Complex Systems Informatics and Modeling Quarterly* 26 (2021), p. 46-54.

[33] Lightning Lab. *Connecting Blockchains : Instant Cross-Chain Transactions On Lightning*. https://lightning.engineering/posts/2017-11-16-ln-swap/. Nov. 2017.

[34] Rongjian Lan et al. "Horizon : A gas-efficient, trustless bridge for cross-chain transactions". In : *arXiv preprint arXiv :2101.06000* (2021).

[35] Bitstamp Learn. *What are blockchain bridges ?* 2023. url : https://www.bitstamp.net/learn/blockchain/what-are-blockchain-bridges/.

[36] Jesús Andrés Mathus Garza. *Lightning Network node software*. https://github.com/jmathus/lit/tree/HTLCswaps. 2017.

[37] Jesús Andrés Mathus Garza. "The lightning network cross-chain exchange : a decentralized approach for peer to peer exchange across blockchain". Thèse de doct. Massachusetts Institute of Technology, 2018.

[38] Satoshi Nakamoto. *Bitcoin : A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf. 2008.

[39] Krishnasuri Narayanam et al. "Generalized HTLC for Cross-Chain Swapping of Multiple Assets with Co-Ownerships". In : *arXiv e-prints* (2022), arXiv-2202.

[40] Ngrave. *Blockchain Interoperability : Challenges & Opportunities*. 2023. url : https://www.ngrave.io/en/blog/blockchain-interoperability-challenges-opportunities.

[41] *Nomad Bridge Hack Root Cause Analysis*. Medium. Août 2022. url : https://medium.com/nomad-xyz-blog/nomad-bridge-hack-root-cause-analysis-875ad2e5aacd.

[42] *Nomad Rekt*. Rekt. Août 2022. url : https://rekt.news/nomad-rekt/.

[43] *Order Book*. url : https://academy.binance.com/en/glossary/order-book.

[44] Joseph Poon et Thaddeus Dryja. *The bitcoin lightning network : Scalable off-chain instant payments*. 2016.

[45] Kaihua Qin et Arthur Gervais. "An overview of blockchain scalability, interoperability and sustainability". In : *Hochschule Luzern Imperial College London Liquidity Network* (2018).

[46] *Qu'est-ce que le staking dans la crypto ?* Binance Academy. Sept. 2019. URL : https://academy.binance.com/fr/articles/what-is-staking.

[47] *Replica.sol*. GitHub. Juill. 2022. URL : https://github.com/nomad-xyz/monorepo/blob/2b80ba83755a43e046c65d303packages/contracts-core/contracts/Replica.sol#L103.

[48] *Replica.sol (fixed)*. GitHub. Sept. 2022. URL : https://github.com/nomad-xyz/monorepo/blob/0e02cc1f09d16f809f5d2d8f05abbeea6d1af04e/packages/contracts-core/contracts/Replica.sol#L103.

[49] Block SEC. *Security Incident on Seal Finance*. https://blocksecteam.medium.com/security-incident-on-seal-finance-fa79c27a1c3b. Jan. 2021.

[50] *Solana's Wormhole Hack Post-Mortem Analysis*. Medium. Fév. 2022. URL : https://extropy-io.medium.com/solanas-wormhole-hack-post-mortem-analysis-3b68b9e88e13.

[51] National Institute of STANDARDS et TECHNOLOGY. *CVE-2021-3006*. https://nvd.nist.gov/vuln/detail/CVE-2021-3006. Mars 2021.

[52] *Verify signature*. GitHub. Sept. 2021. URL : https://github.com/wormhole-foundation/wormhole/blob/ca509f2d73c0780e8516ffdfcaf90b38ab6db203/solana/bridge/program/src/api/verify_signature.rs#L101.

[53] *Verify signature (fixed)*. GitHub. Fév. 2022. URL : https://github.com/wormhole-foundation/wormhole/blob/7edbbd3677ee6ca681be8722a607bc576a3912c8/solana/bridge/program/src/api/verify_signature.rs.

[54] Martin WESTERKAMP et Maximilian DIEZ. "Verilay : A verifiable proof of stake chain relay". In : *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2022, p. 1-9.

[55] *Wormhole Rekt*. Rekt. Fév. 2022. URL : https://rekt.news/wormhole-rekt/.