Applications de chaîne de blocs

2022-03-13

# Applications de chaîne de blocs

Appuyons-nous maintenant sur notre compréhension du bitcoin en le considérant comme une *plate-forme d'application*. De nos jours, de nombreuses personnes utilisent le terme "blockchain" ou "chaîne de blocs" pour désigner toute plate-forme d'application qui partage les principes de conception du bitcoin. Le terme est souvent mal utilisé et appliqué à de nombreuses choses qui ne fournissent pas les principales fonctionnalités fournies par la chaîne de blocs de Bitcoin.

Dans ce chapitre, nous examinerons les fonctionnalités offertes par la chaîne de blocs Bitcoin, en tant que plate-forme d'application. Nous considérerons les *primitives* de construction d'applications, qui forment les blocs de construction de toute application chaîne de blocs. Nous examinerons plusieurs applications importantes qui utilisent ces primitives, telles que les canaux de paiement (état) et les canaux de paiement routés (Lightning Network).

## Présentation

Le système Bitcoin a été conçu comme un système de monnaie et de paiement décentralisé. Cependant, la plupart de ses fonctionnalités sont dérivées de constructions de niveau beaucoup plus bas qui peuvent être utilisées pour des applications beaucoup plus larges. Bitcoin n'a pas été construit avec des composants tels que des comptes, des utilisateurs, des soldes et des paiements. Au lieu de cela, il utilise un langage de script transactionnel avec des fonctions cryptographiques de bas niveau, comme nous l'avons vu dans [???](#transactions). Tout comme les concepts de niveau supérieur des comptes, des soldes et des paiements peuvent être dérivés de ces primitives de base, il en va de même pour de nombreuses autres applications complexes. Ainsi, la chaîne de blocs Bitcoin peut devenir une plate-forme applicative offrant des services de confiance aux applications, tels que les contrats intelligents, dépassant de loin l'objectif initial de la monnaie numérique et des paiements.

## Structures de base (les primitifs)

Lorsqu'il fonctionne correctement et sur le long terme, le système Bitcoin offre certaines garanties, qui peuvent servir de structures de base pour créer des applications. Ceux-ci inclus:

Pas de double dépense

La garantie la plus fondamentale de l'algorithme de consensus décentralisé de Bitcoin garantit qu'aucun UTXO ne peut être dépensé deux fois.

Immuabilité

Une fois qu'une transaction est enregistrée dans la chaîne de blocs et que suffisamment de travail a été ajouté avec les blocs suivants, les données de la transaction deviennent immuables. L'immuabilité est garantie par l'énergie, car la réécriture de la chaîne de blocs nécessite une dépense d'énergie pour produire une preuve de travail. L'énergie requise et donc le degré d'immuabilité augmentent avec la quantité de travail engagé au-dessus du bloc contenant une transaction.

Neutralité

Le réseau Bitcoin décentralisé propage des transactions valides quelle que soit l'origine ou le contenu de ces transactions. Cela signifie que n'importe qui peut créer une transaction valide avec des frais suffisants et être sûr qu'il sera en mesure de transmettre cette transaction et de l'inclure dans la chaîne de blocs à tout moment.

Horodatage sécurisé

Les règles de consensus rejettent tout bloc dont l'horodatage est trop éloigné dans le passé ou dans le futur. Cela garantit que les horodatages sur les blocs sont fiables. L'horodatage sur un bloc implique une garantie non dépensée avant pour les entrées de toutes les transactions incluses.

Autorisation

Les signatures numériques, validées dans un réseau décentralisé, offrent des garanties d'autorisation. Les scripts qui contiennent une exigence de signature numérique ne peuvent être exécutés sans l'autorisation du détenteur de la clé privée impliquée dans le script.

Auditabilité

Toutes les transactions sont publiques et peuvent être auditées. Toutes les transactions et tous les blocs sont liés dans une chaîne ininterrompue jusqu’au bloc de genèse.

Comptabilité

Dans toute transaction (à l'exception de la transaction coinbase), la valeur des entrées est égale à la valeur des sorties plus les frais. Il n'est pas possible de créer ou de détruire de la valeur bitcoin dans une transaction. Les sorties ne peuvent pas dépasser les entrées.

Non-expiration

Une transaction valide n'expire pas. S'il est valable aujourd'hui, il le sera dans un avenir proche, tant que les intrants ne seront pas dépensés et que les règles de consensus ne changeront pas.

Intégrité

Une transaction bitcoin signée avec SIGHASH\_ALL ou des parties d'une transaction signée par un autre type SIGHASH ne peuvent pas être modifiées sans invalider la signature, invalidant ainsi la transaction elle-même.

Atomicité des transactions

Les transactions Bitcoin sont atomiques. Ils sont valides et confirmés (minés) ou non. Les transactions partielles ne peuvent pas être extraites et il n'y a pas d'état intermédiaire pour une transaction. À tout moment, une transaction est minée ou non.

Unités de valeur discrètes (indivisibles)

Les sorties de transaction sont des unités de valeur discrètes et indivisibles. Elles peuvent être dépensés ou non dépensés, en totalité. Elles ne peuvent pas être divisés ou partiellement dépensés.

Quorum de contrôle

les contraintes de multisignature dans les scripts imposent un quorum d'autorisation, prédéfini dans le schéma de multisignature. L'exigence M-de-N est appliquée par les règles de consensus.

Verrou temporel

Toute clause de script contenant un verrou temporel relatif ou absolu ne peut être exécutée qu'après que son âge dépasse le temps spécifié.

Réplication

Le stockage décentralisé de la chaîne de blocs garantit que lorsqu'une transaction est extraite, après des confirmations suffisantes, elle est répliquée sur le réseau et devient durable et résistante aux coupures de courant, aux pertes de données, etc.

Protection contre la falsification

Une transaction ne peut dépenser que des sorties existantes et validées. Il n'est pas possible de créer ou de contrefaire de la valeur.

Cohérence

En l'absence de partitions de mineurs, les blocs enregistrés dans la chaîne de blocs sont sujets à une réorganisation ou à un désaccord avec une probabilité décroissante de manière exponentielle, en fonction de la profondeur à laquelle ils sont enregistrés. Une fois profondément enregistrés, le calcul et l'énergie nécessaires pour changer rendent le changement pratiquement irréalisable.

Enregistrement de l'état externe

Une transaction peut valider une valeur de données, via OP\_RETURN, représentant une transition d'état dans une machine à états externe.

Émission prévisible

Moins de 21 millions de bitcoins seront émis, à un rythme prévisible.

La liste des blocs de construction (ou structures de base) n'est pas complète et d'autres sont ajoutées à chaque nouvelle fonctionnalité introduite dans le bitcoin.

## Applications à partir des structures de base

Les structures de base proposés par bitcoin sont des éléments d'une plateforme de confiance qui peuvent être utilisés pour composer des applications. Voici quelques exemples d'applications qui existent aujourd'hui et les blocs de construction qu'elles utilisent :

Preuve d'existence (notaire numérique)

Immuabilité + Horodatage + Durabilité. Une empreinte numérique peut être validée avec une transaction dans la chaîne de blocs, prouvant qu'un document existait (horodatage) au moment où il a été enregistré. L'empreinte digitale ne pourra pas être modifiée ex-post-facto (Immutabilité) et la preuve sera conservée de façon permanente (Durabilité).

Kickstarter (Lighthouse)

Cohérence + Atomicité + Intégrité. Si vous signez une entrée et la sortie (intégrité) d'une transaction de collecte de fonds, d'autres peuvent contribuer à la collecte de fonds mais elle ne peut pas être dépensée (atomicité) tant que l'objectif (valeur de sortie) n'est pas financé (cohérence).

Canaux de paiement

Quorum de contrôle + Verrou temporel + Pas de double dépense + Non-expiration + Résistance à la censure + Autorisation. Un multisig 2-of-2 (Quorum) avec un verrou temporel (Verrou temporel) utilisé comme transaction de "règlement" d'un canal de paiement peut être détenu (Non-expiration) et dépensé à tout moment (Résistance à la censure) par l'une ou l'autre des parties (Autorisation). Les deux parties peuvent alors créer des transactions d'engagement qui doublent (Pas de double dépense) le règlement sur un verrou temporel plus court (Verrou temporel).

## Counterparty

Conterparty est une couche de protocole construite au-dessus de bitcoin. Le protocole Counterparty offre la possibilité de créer et d'échanger des actifs virtuels et des jetons. De plus, Counterparty propose un échange décentralisé d'actifs. Conterparty met également en œuvre des contrats intelligents, basés sur la machine virtuelle Ethereum (EVM).

Conterparty intègre des métadonnées dans les transactions bitcoin, en utilisant l'opcode OP\_RETURN ou des adresses multisignatures 1 sur N qui encodent les métadonnées à la place des clés publiques. En utilisant ces mécanismes, Counterparty implémente une couche de protocole encodée dans les transactions bitcoin. La couche de protocole supplémentaire peut être interprétée par des applications compatibles avec Conterparty, telles que les portefeuilles et les explorateurs de chaînes de blocs, ou toute application construite à l'aide des bibliothèques de Conterparty.

Conterparty peut être utilisée comme plate-forme pour d'autres applications et services, à leur tour. Par exemple, Tokenly est une plate-forme construite sur Counterparty qui permet aux créateurs de contenu, aux artistes et aux entreprises d'émettre des jetons qui expriment la propriété numérique et peuvent être utilisés pour louer, accéder, échanger ou acheter du contenu, des produits et des services. D'autres applications tirant parti de Counterparty incluent des jeux (Spells of Genesis) et des projets de grille informatique (Folding Coin).

Plus de détails sur Counterparty peuvent être trouvés sur <https://counterparty.io>. Le projet à source libre est disponible sur <https://github.com/CounterpartyXCP>.

## Canaux de paiement et canaux d'état

*Les canaux de paiement* sont un mécanisme sans confiance pour échanger des bitcoins transactions entre deux parties, en dehors de la chaîne de blocs Bitcoin. Ces transactions, qui seraient valides si elles étaient réglées sur la chaîne de blocs Bitcoin, sont plutôt détenues hors chaîne, agissant comme des *billets à ordre* pour un éventuel règlement par lots. Étant donné que les transactions ne sont pas réglées, elles peuvent être échangées sans la latence de règlement habituelle, ce qui permet un débit de transaction extrêmement élevé, une latence faible (inférieure à la milliseconde) et une granularité fine (niveau satoshi).

En fait, le terme *canal* est une métaphore. Les canaux d'état sont des constructions virtuelles représentées par l'échange d'état entre deux parties, en dehors de la chaîne de blocs. Il n'y a pas de "canaux" en soi et le mécanisme de transport de données sous-jacent n'est pas le canal. Nous utilisons le terme canal pour représenter la relation et l'état partagé entre deux parties, en dehors de la chaîne de blocs.

Pour expliquer davantage ce concept, pensez à un flux TCP. Du point de vue des protocoles de niveau supérieur, il s'agit d'une "interface de connexion" connectant deux applications sur Internet. Mais si vous regardez le trafic réseau, un flux TCP n'est qu'un canal virtuel sur des paquets IP. Chaque extrémité du flux TCP séquence et assemble les paquets IP pour créer l'illusion d'un flux d'octets. En dessous, ce sont tous les paquets déconnectés. De même, un canal de paiement n'est qu'une série de transactions. S'ils sont correctement séquencés et connectés, ils créent des obligations remboursables auxquelles vous pouvez faire confiance même si vous ne faites pas confiance à l'autre côté du canal.

Dans cette section, nous examinerons différentes formes de canaux de paiement. Premièrement, nous examinerons les mécanismes utilisés pour construire un canal de paiement unidirectionnel (unidirectionnel) pour un service de micropaiement mesuré, tel que le streaming vidéo. Ensuite, nous développerons ce mécanisme et introduirons des canaux de paiement bidirectionnels. Enfin, nous verrons comment des canaux bidirectionnels peuvent être connectés de bout en bout pour former des canaux multisauts dans un réseau routé, d'abord proposé sous le nom de *Lightning Network*.

Les canaux de paiement font partie du concept plus large d'un *canal d'état*, qui représente une altération d'état hors chaîne, sécurisée par un éventuel règlement dans une chaîne de blocs. Un canal de paiement est un canal d'état où l'état modifié est le solde d'une monnaie virtuelle.

### Canaux d'état, concepts de base et terminologie

Un canal d'état est établi entre deux parties, par le biais d'une transaction qui verrouille un état partagé sur la chaîne de blocs. C'est ce qu'on appelle la *transaction de financement* ou *transaction d'ancrage*. Cette transaction unique doit être transmise au réseau et exploitée pour établir le canal. Dans l'exemple d'un canal de paiement, l'état bloqué est le solde initial (en devise) du canal.

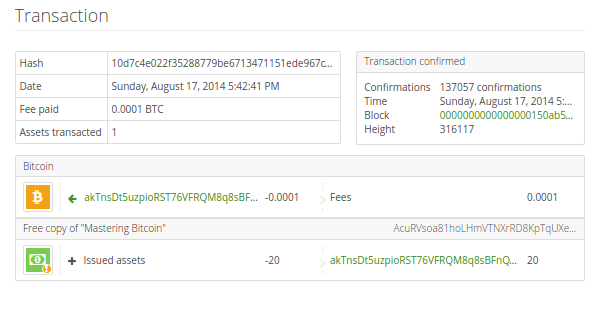
Les deux parties échangent alors des transactions signées, appelées *transactions d'engagement*, qui modifient l'état initial. Ces transactions sont des transactions valides dans la mesure où elles *pourraient* être soumises pour règlement par l'une ou l'autre des parties, mais sont plutôt détenues hors chaîne par chaque partie en attendant la fermeture du canal. Les mises à jour d'état peuvent être créées aussi rapidement que chaque partie peut créer, signer et transmettre une transaction à l'autre partie. En pratique, cela signifie que des milliers de transactions par seconde peuvent être échangées.

Lors de l'échange de transactions d'engagement, les deux parties invalident également les états précédents, de sorte que la transaction d'engagement la plus récente est toujours la seule qui puisse être remboursée. Cela empêche l'une ou l'autre des parties de tricher en fermant unilatéralement le canal avec un état antérieur expiré qui lui est plus favorable que l'état actuel. Nous examinerons les différents mécanismes qui peuvent être utilisés pour invalider l'état antérieur dans la suite de ce chapitre.

Enfin, le canal peut être fermé soit de manière coopérative, en soumettant une *transaction de règlement* finale à la chaîne de blocs, soit unilatéralement, par l'une ou l'autre des parties soumettant la dernière transaction d'engagement à la chaîne de blocs. Une option de fermeture unilatérale est nécessaire au cas où l'une des parties se déconnecte de manière inattendue. La transaction de règlement représente l'état final du canal et est réglée sur la chaîne de blocs.

Pendant toute la durée de vie de la chaîne, seules deux transactions doivent être soumises pour minage sur la chaîne de blocs: les transactions de financement et de règlement. Entre ces deux états, les deux parties peuvent échanger un nombre illimité de transactions d'engagement qui ne sont jamais vues par personne d'autre, ni soumises à la chaîne de blocs.

[figure\_title](#payment_channel) illustre un canal de paiement entre Bob et Alice, montrant les transactions de financement, d'engagement et de règlement.

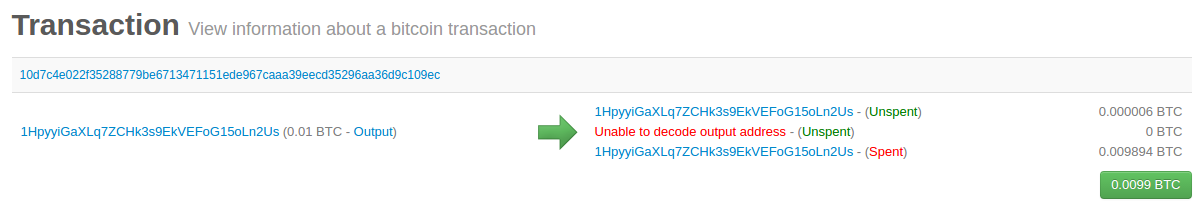


Un canal de paiement entre Bob et Alice, montrant les transactions de financement, d'engagement et de règlement

### Exemple de canal de paiement simple

Pour expliquer les canaux d'état, nous commençons par un exemple très simple. Nous démontrons un canal à sens unique, ce qui signifie que la valeur ne circule que dans une seule direction. Nous partirons également de l'hypothèse naïve que personne n'essaie de tricher, pour garder les choses simples. Une fois que nous aurons expliqué l'idée de base du canal, nous examinerons ce qu'il faut pour le rendre sans confiance afin qu'aucune des parties ne puisse tricher, même si elle essaie de le faire.

Pour cet exemple, nous supposerons deux participants : Emma et Fabian. Fabian propose un service de diffusion vidéo en continu facturé à la seconde via un canal de micropaiement. Fabian facture 0,01 millibit (0,00001 BTC) par seconde de vidéo, ce qui équivaut à 36 millibits (0,036 BTC) par heure de vidéo. Emma est une utilisatrice qui achète ce service de diffusion vidéo en continu auprès de Fabian. [figure\_title](#emma_fabian_streaming_video) montre Emma achetant le service de diffusion vidéo en continu de Fabian en utilisant un canal de paiement.



Emma achète la vidéo en continu de Fabian avec un canal de paiement, payant pour chaque seconde de vidéo

Dans cet exemple, Fabian et Emma utilisent un logiciel spécial qui gère à la fois le canal de paiement et la diffusion vidéo en continu. Emma exécute le logiciel dans son navigateur, Fabian l'exécute sur un serveur. Le logiciel inclut les fonctionnalités de base du portefeuille bitcoin et peut créer et signer des transactions bitcoin. Le concept et le terme "canal de paiement" sont complètement cachés aux utilisateurs. Ce qu'ils voient, c'est une vidéo payée à la seconde.

Pour mettre en place le canal de paiement, Emma et Fabian établissent une adresse multisignature 2 sur 2, chacun d'eux détenant l'une des clés. Du point de vue d'Emma, le logiciel de son navigateur présente un code QR avec une adresse P2SH (commençant par "3") et lui demande de soumettre un "dépôt" pour jusqu'à 1 heure de vidéo. L'adresse est alors financée par Emma. La transaction d'Emma, payant à l'adresse multisignature, est la transaction de financement ou d'ancrage pour le canal de paiement.

Pour cet exemple, disons qu'Emma finance la chaîne avec 36 millibits (0,036 BTC). Cela permettra à Emma de consommer *jusqu'à* 1 heure de diffusion vidéo en continu. La transaction de financement dans ce cas fixe le montant maximum qui peut être transmis dans ce canal, en définissant la *capacité du canal*.

La transaction de financement consomme une ou plusieurs entrées du portefeuille d'Emma, approvisionnant les fonds. Il crée une sortie d'une valeur de 36 millibits payée à l'adresse multisignature 2 sur 2 contrôlée conjointement entre Emma et Fabian. Il peut avoir des sorties supplémentaires pour revenir au portefeuille d'Emma.

Une fois la transaction de financement confirmée, Emma peut commencer à diffuser la vidéo. Le logiciel d'Emma crée et signe une transaction d'engagement qui modifie le solde du canal pour créditer 0,01 millibit à l'adresse de Fabian et rembourser 35,99 millibits à Emma. La transaction signée par Emma consomme la sortie de 36 millibits créée par la transaction de financement et crée deux sorties : une pour son remboursement, l'autre pour le paiement de Fabian. La transaction n'est que partiellement signée - elle nécessite deux signatures (2 sur 2), mais n'a que la signature d'Emma. Lorsque le serveur de Fabian reçoit cette transaction, il ajoute la deuxième signature (pour l'entrée 2 sur 2) et la renvoie à Emma avec 1 seconde de vidéo. Désormais, les deux parties ont une transaction d'engagement entièrement signée que l'une ou l'autre peut racheter, représentant le solde à jour correct du canal. Aucune des parties ne diffuse cette transaction sur le réseau.

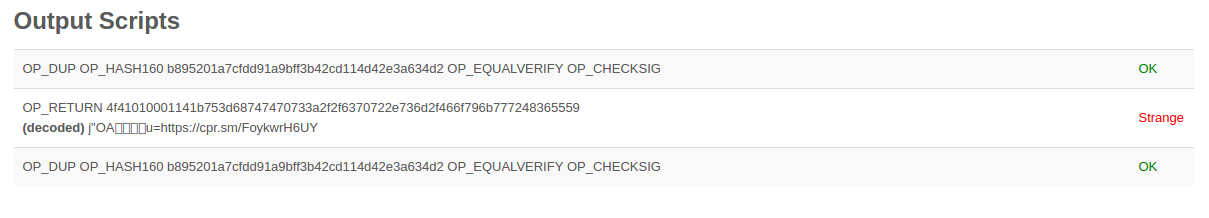
Au tour suivant, le logiciel d'Emma crée et signe une autre transaction d'engagement (engagement #2) qui consomme la *même* sortie 2 sur 2 de la transaction de financement. La seconde transaction d'engagement alloue une sortie de 0,02 millibits à l'adresse de Fabian et une sortie de 35,98 millibits à l'adresse d'Emma. Cette nouvelle transaction est le paiement de deux secondes cumulées de vidéo. Le logiciel de Fabian signe et renvoie la deuxième transaction d'engagement, ainsi qu'une autre seconde de vidéo.

De cette manière, le logiciel d'Emma continue d'envoyer des transactions d'engagement au serveur de Fabian en échange de diffusion vidéo en continu. Le solde de la chaîne s'accumule progressivement en faveur de Fabian, car Emma consomme plus de secondes de vidéo. Disons qu'Emma regarde 600 secondes (10 minutes) de vidéo, créant et signant 600 transactions d'engagement. La dernière transaction d'engagement (#600) aura deux sorties, divisant le solde de la chaîne, 6 millibits pour Fabian et 30 millibits pour Emma.

Enfin, Emma sélectionne "Stop" pour arrêter le streaming vidéo. Fabian ou Emma peuvent maintenant transmettre la transaction d'état finale pour règlement. Cette dernière transaction est la *transaction de règlement* et paie Fabian pour toute la vidéo consommée par Emma, remboursant le reste de la transaction de financement à Emma.

[figure\_title](#video_payment_channel) montre le canal entre Emma et Fabian et les transactions d'engagement qui mettent à jour le solde du canal.

Au final, seules deux transactions sont enregistrées sur la chaîne de bocs: la transaction de financement qui établit le canal et une transaction de règlement qui répartit correctement le solde final entre les deux participants.



Le canal de paiement d'Emma avec Fabian, montrant les transactions d'engagement qui mettent à jour le solde du canal

### Créer des canaux sans confiance

Le canal que nous venons de décrire fonctionne, mais seulement si les deux parties coopèrent, sans échec ni tentative de tricherie. Examinons certains des scénarios qui cassent ce canal et voyons ce qui est nécessaire pour les réparer :

* Une fois la transaction de financement effectuée, Emma a besoin de la signature de Fabian pour récupérer l'argent. Si Fabian disparaît, les fonds d'Emma sont enfermés dans un 2 sur 2 et effectivement perdus. Ce canal, tel que construit, entraîne une perte de fonds si l'une des parties se déconnecte avant qu'il y ait au moins une transaction d'engagement signée par les deux parties.
* Pendant que le canal est en cours d'exécution, Emma peut prendre n'importe laquelle des transactions d'engagement que Fabian a contresignées et en transmettre une à la chaîne de blocs. Pourquoi payer 600 secondes de vidéo, si elle peut transmettre la transaction d'engagement #1 et ne payer que 1 seconde de vidéo ? La chaîne échoue car Emma peut tricher en diffusant un engagement préalable qui est en sa faveur.

Ces deux problèmes peuvent être résolus avec des verrous temporels. Voyons comment nous pourrions utiliser des verrous temporels au niveau des transactions (nLocktime).

Emma ne peut pas risquer de financer un multisig 2 sur 2 à moins qu'elle n'ait un remboursement garanti. Pour résoudre ce problème, Emma construit les transactions de financement et de remboursement en même temps. Elle signe la transaction de financement mais ne la transmet à personne. Emma transmet uniquement la transaction de remboursement à Fabian et obtient sa signature.

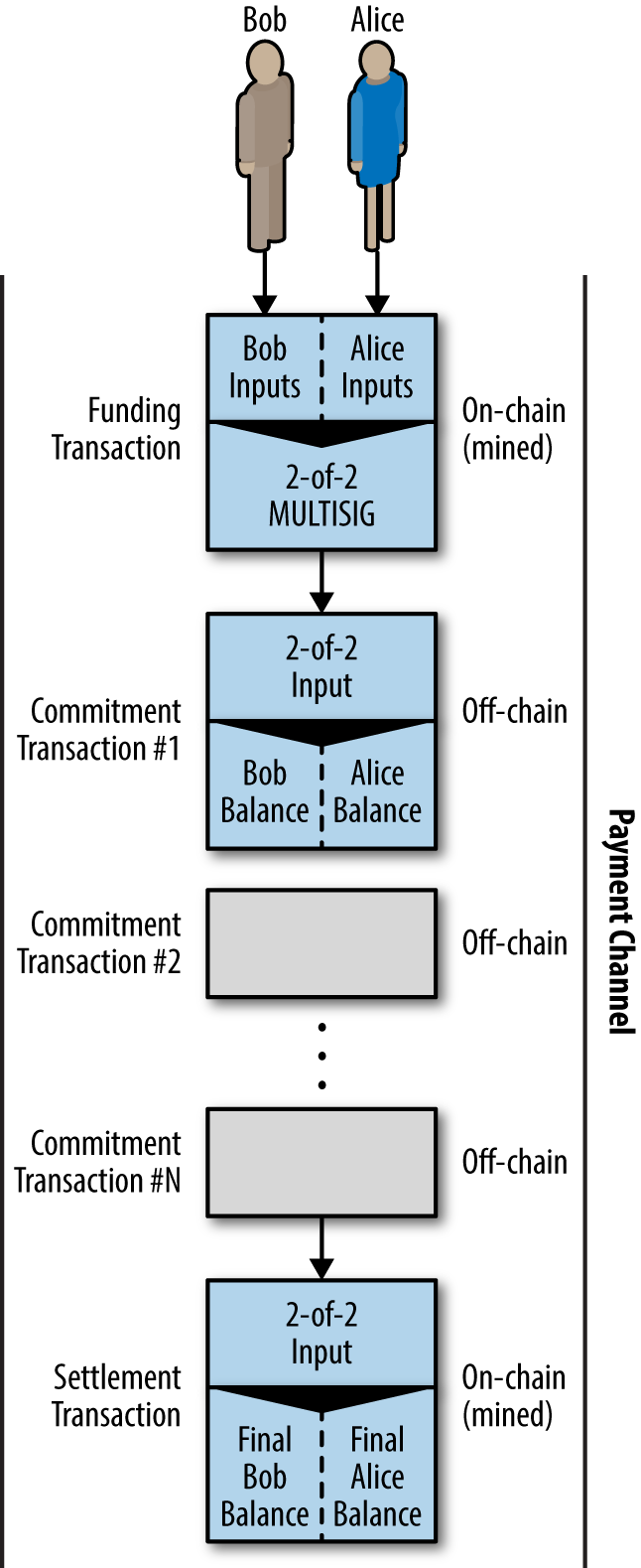
La transaction de remboursement agit comme la première transaction d'engagement et son verrou temporel établit la limite supérieure pour la durée de vie du canal. Dans ce cas, Emma pourrait définir le nLocktime sur 30 jours ou 4320 blocs dans le futur. Toutes les transactions d'engagement ultérieures doivent avoir un verrou temporel plus court, afin qu'elles puissent être remboursées avant la transaction de remboursement.

Maintenant qu'Emma a une transaction de remboursement entièrement signée, elle peut transmettre en toute confiance la transaction de financement signée en sachant qu'elle pourra éventuellement, après l'expiration du délai, racheter la transaction de remboursement même si Fabian disparaît.

Chaque transaction d'engagement que les parties échangent pendant la durée de vie du canal sera verrouillée dans le futur. Mais le délai sera légèrement plus court pour chaque engagement afin que l'engagement le plus récent puisse être remboursé avant l'engagement précédent qu'il invalide. En raison de nLockTime, aucune des parties ne peut propager avec succès l'une des transactions d'engagement jusqu'à l'expiration de leur verrouillage temporel. Si tout va bien, ils coopéreront et fermeront le canal gracieusement avec une transaction de règlement, rendant inutile la transmission d'une transaction d'engagement intermédiaire. Sinon, la transaction d'engagement la plus récente peut être propagée pour régler le compte et invalider toutes les transactions d'engagement précédentes.

Par exemple, si la transaction d'engagement #1 est verrouillée dans le temps sur 4320 blocs à l'avenir, alors la transaction d'engagement #2 est verrouillée dans le temps sur 4319 blocs dans le futur. La transaction d'engagement #600 peut être dépensée 600 blocs avant que la transaction d'engagement #1 ne devienne valide.

[figure\_title](#timelocked_commitments) montre chaque transaction d'engagement définissant un verrou temporel plus court, lui permettant d'être dépensé avant que les engagements précédents ne deviennent valides.



Chaque engagement fixe un délai plus court, ce qui lui permet d'être dépensé avant que les engagements précédents ne deviennent valides

Chaque transaction d'engagement ultérieure doit avoir un verrou temporel plus court afin qu'elle puisse être diffusée avant ses prédécesseurs et avant la transaction de remboursement. La possibilité de diffuser un engagement plus tôt garantit qu'il sera en mesure de dépenser la sortie de financement et d'empêcher toute autre transaction d'engagement d'être remboursée en dépensant la sortie. Les garanties offertes par la chaîne de blocs Bitcoin, empêchant les doubles dépenses et appliquant des verrous temporels, permettent effectivement à chaque transaction d'engagement d'invalider ses prédécesseurs.

Les canaux d'état utilisent des verrous temporels pour appliquer des contrats intelligents dans une dimension temporelle. Dans cet exemple, nous avons vu comment la dimension temporelle garantit que la transaction d'engagement la plus récente devient valide avant tout engagement antérieur. Ainsi, la transaction d'engagement la plus récente peut être transmise, dépensant les entrées et invalidant les transactions d'engagement précédentes. L'application de contrats intelligents avec des délais absolus protège contre la tricherie par l'une des parties. Cette implémentation n'a besoin de rien de plus que des verrous temporels absolus au niveau de la transaction (nLocktime). Ensuite, nous verrons comment les verrous temporels au niveau du script, CHECKLOCKTIMEVERIFY et CHECKSEQUENCEVERIFY, peuvent être utilisés pour construire des canaux d'état plus flexibles, utiles et sophistiqués.

La première forme de canal de paiement unidirectionnel a été présentée sous la forme d'un prototype d'application de streaming vidéo en 2015 par une équipe de développeurs argentins.

Les verrous temporels ne sont pas le seul moyen d'invalider les transactions d'engagement antérieures. Dans les sections suivantes, nous verrons comment une clé de révocation peut être utilisée pour obtenir le même résultat. Les verrous temporels sont efficaces mais ils ont deux inconvénients distincts. En établissant un verrou temporel maximal lors de la première ouverture du canal, ils limitent la durée de vie du canal. Pire, ils obligent les implémentations de canaux à trouver un équilibre entre autoriser des canaux à longue durée de vie et obliger l'un des participants à attendre très longtemps un remboursement en cas de fermeture prématurée. Par exemple, si vous autorisez le canal à rester ouvert pendant 30 jours, en réglant le délai de remboursement sur 30 jours, si l'une des parties disparaît immédiatement, l'autre partie doit attendre 30 jours pour un remboursement. Plus le point final est éloigné, plus le remboursement est éloigné.

Le deuxième problème est que puisque chaque transaction d'engagement ultérieure doit décrémenter le verrouillage temporel, il existe une limite explicite sur le nombre de transactions d'engagement qui peuvent être échangées entre les parties. Par exemple, un canal de 30 jours, définissant un verrouillage temporel de 4320 blocs dans le futur, ne peut accepter que 4320 transactions d'engagement intermédiaires avant de devoir être clôturé. Il y a un danger à définir l'intervalle de transaction d'engagement de verrouillage de temps à 1 bloc. En définissant l'intervalle de verrouillage entre les transactions d'engagement sur 1 bloc, un développeur crée une charge très lourde pour les participants au canal qui doivent être vigilants, rester en ligne et regarder, et être prêts à transmettre la bonne transaction d'engagement à tout moment.

Maintenant que nous comprenons comment les verrous temporels peuvent être utilisés pour invalider des engagements antérieurs, nous pouvons voir la différence entre fermer le canal de manière coopérative et le fermer unilatéralement en diffusant une transaction d'engagement. Toutes les transactions d'engagement sont verrouillées dans le temps, par conséquent, la diffusion d'une transaction d'engagement impliquera toujours d'attendre jusqu'à ce que le verrouillage ait expiré. Mais si les deux parties s'entendent sur le solde final et savent qu'elles détiennent toutes les deux des transactions d'engagement qui feront éventuellement de ce solde une réalité, elles peuvent construire une transaction de règlement sans verrou temporel représentant ce même solde. Dans une clôture coopérative, l'une ou l'autre des parties prend la transaction d'engagement la plus récente et construit une transaction de règlement qui est identique à tous points de vue, sauf qu'elle omet le verrou temporel. Les deux parties peuvent signer cette transaction de règlement en sachant qu'il n'y a aucun moyen de tricher et d'obtenir un solde plus favorable. En signant et en transmettant de manière coopérative la transaction de règlement, ils peuvent fermer le canal et racheter leur solde immédiatement. Dans le pire des cas, l'une des parties peut être mesquine, refuser de coopérer et forcer l'autre partie à conclure unilatéralement la transaction d'engagement la plus récente. Mais s'ils le font, ils doivent aussi attendre leurs fonds.

### Engagements révocables asymétriques

Une meilleure façon de gérer les états d'engagement antérieurs est de les révoquer explicitement. Cependant, ce n'est pas facile à réaliser. Une caractéristique clé du bitcoin est qu'une fois qu'une transaction est valide, elle reste valide et n'expire pas. La seule façon d'annuler une transaction est de doubler ses entrées avec une autre transaction avant qu'elle ne soit minée. C'est pourquoi nous avons utilisé des verrous temporels dans l'exemple de canal de paiement simple ci-dessus pour nous assurer que les engagements les plus récents pourraient être dépensés avant que les engagements plus anciens ne soient valides. Cependant, l'enchaînement des engagements dans le temps crée un certain nombre de contraintes qui rendent les canaux de paiement difficiles à utiliser.

Même si une transaction ne peut pas être annulée, elle peut être construite de manière à rendre son utilisation indésirable. Pour ce faire, nous donnons à chaque partie une *clé de révocation* qui peut être utilisée pour punir l'autre partie si elle essaie de tricher. Ce mécanisme de révocation des opérations d'engagement préalable a été initialement proposé dans le cadre du Lightning Network.

Pour expliquer les clés de révocation, nous allons construire un canal de paiement plus complexe entre deux échanges gérés par Hitesh et Irene. Hitesh et Irene gèrent respectivement des échanges de bitcoins en Inde et aux États-Unis. Les clients de l'échange indien d'Hitesh envoient souvent des paiements aux clients de l'échange américain d'Irene et vice versa. Actuellement, ces transactions se produisent sur la chaîne de blocs Bitcoin, mais cela signifie payer des frais et attendre plusieurs blocs pour les confirmations. La mise en place d'un canal de paiement entre les bourses réduira considérablement les coûts et accélérera le flux des transactions.

Hitesh et Irene démarrent le canal en construisant en collaboration une transaction de financement, chacun finançant la chaîne avec 5 bitcoins. Le solde initial est de 5 bitcoins pour Hitesh et de 5 bitcoins pour Irene. La transaction de financement verrouille l'état du canal dans un multisig 2 sur 2, comme dans l'exemple d'un canal simple.

La transaction de financement peut avoir une ou plusieurs entrées de Hitesh (ajoutant jusqu'à 5 bitcoins ou plus) et une ou plusieurs entrées d'Irene (ajoutant jusqu'à 5 bitcoins ou plus). Les entrées doivent légèrement dépasser la capacité du canal afin de couvrir les frais de transaction. La transaction a une sortie qui verrouille les 10 bitcoins au total sur une adresse multisig 2 sur 2 contrôlée à la fois par Hitesh et Irene. La transaction de financement peut également avoir une ou plusieurs sorties rendant la monnaie à Hitesh et Irene si leurs entrées ont dépassé leur contribution de canal prévue. Il s'agit d'une transaction unique avec des entrées offertes et signées par deux parties. Il doit être construit en collaboration et signé par chaque partie avant d'être transmis.

Maintenant, au lieu de créer une transaction d'engagement unique que les deux parties signent, Hitesh et Irene créent deux transactions d'engagement différentes qui sont *asymétriques*.

Hitesh a une transaction d'engagement avec deux sorties. La première sortie paie à Irene les 5 bitcoins qui lui sont dus *immédiatement*. La deuxième sortie paie à Hitesh les 5 bitcoins qui lui sont dus, mais seulement après un verrou temporel de 1000 blocs. Les sorties de transaction ressemblent à ceci :

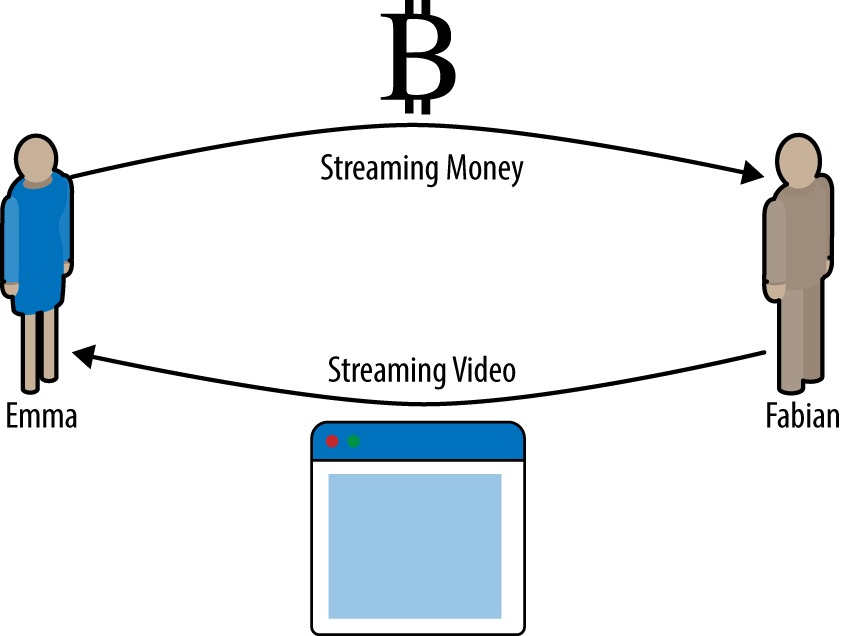
Input: 2-of-2 funding output, signed by Irene  
  
Output 0 <5 bitcoin>:  
 <Irene's Public Key> CHECKSIG  
  
Output 1 <5 bitcoin>:  
 <1000 blocks>  
 CHECKSEQUENCEVERIFY  
 DROP  
 <Hitesh's Public Key> CHECKSIG

Irène a une transaction d'engagement différente avec deux sorties. La première sortie paie à Hitesh les 5 bitcoins qui lui sont dus immédiatement. La deuxième sortie paie à Irene les 5 bitcoins qui lui sont dus mais seulement après un verrou temporel de 1000 blocs. La transaction d'engagement qu'Irene détient (signée par Hitesh) ressemble à ceci :

Entrée : sortie de financement 2 sur 2, signée par Hitesh  
  
Output 0 <5 bitcoin>:  
 <Hitesh's Public Key> CHECKSIG  
  
Output 1 <5 bitcoin>:  
 <1000 blocks>  
 CHECKSEQUENCEVERIFY  
 DROP  
 <Irene's Public Key> CHECKSIG

De cette façon, chaque partie a une transaction d'engagement, dépensant la sortie de financement 2 sur 2. Cette entrée est signée par l'\_autre\_ partie. À tout moment, la partie détenant la transaction peut également signer (remplir le 2 sur 2) et diffuser. Cependant, s'ils diffusent la transaction d'engagement, il paie immédiatement l'autre partie alors qu'elle doit attendre l'expiration d'un verrou temporel. En imposant un délai au remboursement de l'une des sorties, nous désavantageons légèrement chaque partie lorsqu'elle choisit de diffuser unilatéralement une opération d'engagement. Mais un délai ne suffit pas à lui seul à encourager une conduite équitable.

[figure\_title](#asymmetric_commitments) montre deux opérations d'engagement asymétriques, où la sortie payant le titulaire de l'engagement est retardée.



Deux opérations d'engagement asymétriques avec paiement différé pour le titulaire de l'opération

Nous introduisons maintenant le dernier élément de ce schéma : une clé de révocation qui empêche un tricheur de diffuser un engagement expiré. La clé de révocation permet à la partie lésée de sanctionner le tricheur en prélevant tout le solde de la chaîne.

La clé de révocation est composée de deux secrets, chacun généré indépendamment par chaque participant du canal. Il est similaire à un multisig 2 sur 2, mais construit à l'aide de l'arithmétique de la courbe elliptique, de sorte que les deux parties connaissent la clé publique de révocation mais que chaque partie ne connaît que la moitié de la clé secrète de révocation.

À chaque tour, les deux parties révèlent leur moitié du secret de révocation à l'autre partie, donnant ainsi à l'autre partie (qui a maintenant les deux moitiés) les moyens de réclamer la sortie de pénalité si cette transaction révoquée est jamais diffusée.

Chacune des transactions d'engagement a une sortie "retardée". Le script de rachat pour cette sortie permet à une partie de la racheter après 1000 blocs, *ou* à l'autre partie de la racheter si elle a une clé de révocation, pénalisant la transmission d'un engagement révoqué.

Ainsi, lorsque Hitesh crée une transaction d'engagement pour Irene à signer, il rend la deuxième sortie payable à lui-même après 1000 blocs, ou à la clé publique de révocation (dont il ne connaît que la moitié du secret). Hitesh construit cette transaction. Il ne révélera sa moitié du secret de révocation à Irène que lorsqu'il sera prêt à passer à un nouvel état de canal et qu'il voudra révoquer cet engagement.

Le script de la deuxième sortie ressemble à ceci :

Output 0 <5 bitcoin>:  
 <Irene's Public Key> CHECKSIG  
  
Output 1 <5 bitcoin>:  
IF  
# Sortie de pénalité de révocation  
 <Revocation Public Key>  
ELSE  
 <1000 blocks>  
 CHECKSEQUENCEVERIFY  
 DROP  
 <Hitesh's Public Key>  
ENDIF  
CHECKSIG

Irène peut signer cette transaction en toute confiance, car si elle est transmise, elle lui paiera immédiatement ce qui lui est dû. Hitesh détient la transaction, mais sait que s'il la transmet dans une fermeture de canal unilatérale, il devra attendre 1000 blocs pour être payé.

Lorsque le canal passe à l'état suivant, Hitesh doit *révoquer* cette transaction d'engagement avant qu'Irene n'accepte de signer la prochaine transaction d'engagement. Pour cela, il lui suffit d'envoyer sa moitié de *clé de révocation* à Irène. Une fois qu'Irene a les deux moitiés de la clé secrète de révocation pour cet engagement, elle peut signer le prochain engagement en toute confiance. Elle sait que si Hitesh essaie de tricher en publiant l'engagement précédent, elle peut utiliser la clé de révocation pour racheter la sortie retardée de Hitesh. *Si Hitesh triche, Irene obtient les DEUX sorties*. Pendant ce temps, Hitesh n'a que la moitié du secret de révocation pour cette clé publique de révocation et ne peut pas racheter la sortie avant 1000 blocs. Irene pourra racheter la sortie et punir Hitesh avant que les 1000 blocs ne se soient écoulés.

Le protocole de révocation est bilatéral, ce qui signifie qu'à chaque tour, à mesure que l'état du canal est avancé, les deux parties échangent de nouveaux engagements, échangent des secrets de révocation pour les engagements précédents et signent les nouvelles transactions d'engagement de l'autre. Au fur et à mesure qu'ils acceptent un nouvel état, ils rendent l'état antérieur inutilisable, en se donnant mutuellement les secrets de révocation nécessaires pour sanctionner toute tricherie.

Regardons un exemple de la façon dont cela fonctionne. L'un des clients d'Irene souhaite envoyer 2 bitcoins à l'un des clients de Hitesh. Pour transmettre 2 bitcoins à travers le canal, Hitesh et Irene doivent faire avancer l'état du canal pour refléter le nouvel équilibre. Ils s'engageront dans un nouvel état (état numéro 2) où les 10 bitcoins de la chaîne sont divisés, 7 bitcoins pour Hitesh et 3 bitcoins pour Irene. Pour faire progresser l'état du canal, ils créeront chacun de nouvelles transactions d'engagement reflétant le nouveau solde du canal.

Comme auparavant, ces transactions d'engagement sont asymétriques de sorte que la transaction d'engagement que chaque partie détient les oblige à attendre si ils la remboursent. Surtout, avant de signer de nouvelles transactions d'engagement, ils doivent d'abord échanger des clés de révocation pour invalider l'engagement précédent. Dans ce cas particulier, les intérêts de Hitesh sont alignés sur l'état réel de la chaîne et il n'a donc aucune raison de diffuser un état antérieur. Cependant, pour Irene, l'état numéro 1 lui laisse un solde plus élevé que l'état 2. Quand Irene donne à Hitesh la clé de révocation pour sa transaction d'engagement précédente (état numéro 1), elle révoque effectivement sa capacité à tirer profit de la régression du canal vers un précédent. état parce qu'avec la clé de révocation, Hitesh peut racheter sans délai les deux sorties de la transaction d'engagement précédente. Cela signifie que si Irene diffuse l'état antérieur, Hitesh peut exercer son droit de prendre toutes les sorties.

Il est important de noter que la révocation ne se produit pas automatiquement. Alors que Hitesh a la capacité de punir Irene pour avoir triché, il doit surveiller la blockchain avec diligence pour détecter des signes de tricherie. S'il voit une diffusion d'une transaction d'engagement préalable, il dispose de 1000 blocs pour agir et utiliser la clé de révocation pour contrecarrer la tricherie d'Irène et la punir en prenant la totalité du solde, les 10 bitcoins.

Les engagements révocables asymétriques avec blocages temporels relatifs (CSV) sont une bien meilleure façon de mettre en œuvre les canaux de paiement et une innovation très importante dans cette technologie. Avec cette construction, le canal peut rester ouvert indéfiniment et peut avoir des milliards de transactions d'engagement intermédiaires. Dans les implémentations prototypes de Lightning Network, l'état d'engagement est identifié par un index de 48 bits, permettant plus de 281 000 milliards (2,8 x 1014) de transitions d'état dans n'importe quel canal !

### Contrats de verrouillage du temps de hachage (HTLC)

Les canaux de paiement peuvent être encore étendus avec un type spécial de contrat intelligent qui permet aux participants d'engager des fonds dans un secret remboursable, avec un délai d'expiration. Cette fonctionnalité est appelée *contrats de verrouillage du temps de hachage* (*Hash Time Lock Contract*), ou *HTLC*, et est utilisée à la fois dans les canaux de paiement bidirectionnels et routés.

Expliquons d'abord la partie "hachage" du HTLC. Pour créer un HTLC, le destinataire prévu du paiement créera d'abord un R secret. Ils calculent ensuite le hash de ce secret H :

H = Hachage(R)

Cela produit un hachage H qui peut être inclus dans le script de verrouillage d'une sortie. Celui qui connaît le secret peut l'utiliser pour racheter la sortie. Le secret R est également appelé *préimage* pour la fonction de hachage. La préimage est simplement la donnée utilisée comme entrée dans une fonction de hachage.

La deuxième partie d'un HTLC est le composant "verrou temporel". Si le secret n'est pas révélé, le payeur du HTLC peut obtenir un "remboursement" après un certain temps. Ceci est réalisé avec un verrouillage temporel absolu à l'aide de CHECKLOCKTIMEVERIFY.

Le script implémentant un HTLC pourrait ressembler à ceci :

IF  
# Paiement si vous avez le secret R  
 HASH160 <H> EQUALVERIFY  
ELSE  
# Remboursement après expiration du délai.  
 <locktime> CHECKLOCKTIMEVERIFY DROP  
 <Payer Public Key> CHECKSIG  
ENDIF

Toute personne connaissant le secret R, qui, lorsqu'il est haché, est égal à H, peut racheter cette sortie en exerçant la première clause du flux IF.

Si le secret n'est pas révélé et que le HTLC est réclamé, après un certain nombre de blocages, le payeur peut demander un remboursement en utilisant la deuxième clause du flux IF.

Il s'agit d'une implémentation de base d'un HTLC. Ce type de HTLC peut être échangé par *toute personne* qui a le secret R. Un HTLC peut prendre de nombreuses formes différentes avec de légères variations dans le script. Par exemple, l'ajout d'un opérateur CHECKSIG et d'une clé publique dans la première clause limite le rachat du hachage à un destinataire nommé, qui doit également connaître le secret R.

## Canaux de paiement routés (Lightning Network)

Le Lightning Network est un réseau routé proposé de canaux de paiement bidirectionnels connectés de bout en bout. Un tel réseau peut permettre à n'importe quel participant d'acheminer un paiement d'un canal à l'autre sans faire confiance à aucun des intermédiaires. Le Lightning Network était <https://lightning.network/lightning-network-paper.pdf> [décrit pour la première fois par Joseph Poon et Thadeus Dryja en février 2015], s'appuyant sur le concept de canaux de paiement tel que proposé et élaboré par de nombreux autres.

Le "Lightning Network" fait référence à une conception spécifique pour un réseau de canaux de paiement routés, qui a maintenant été mis en œuvre par au moins cinq équipes de sources libres différentes. Les implémentations indépendantes sont coordonnées par un ensemble de normes d'interopérabilité décrites dans le [document *Basics of Lightning Technology (BOLT)*](https://bit.ly/2rBHeoL).

Des implémentations prototypes du Lightning Network ont été publiées par plusieurs équipes.

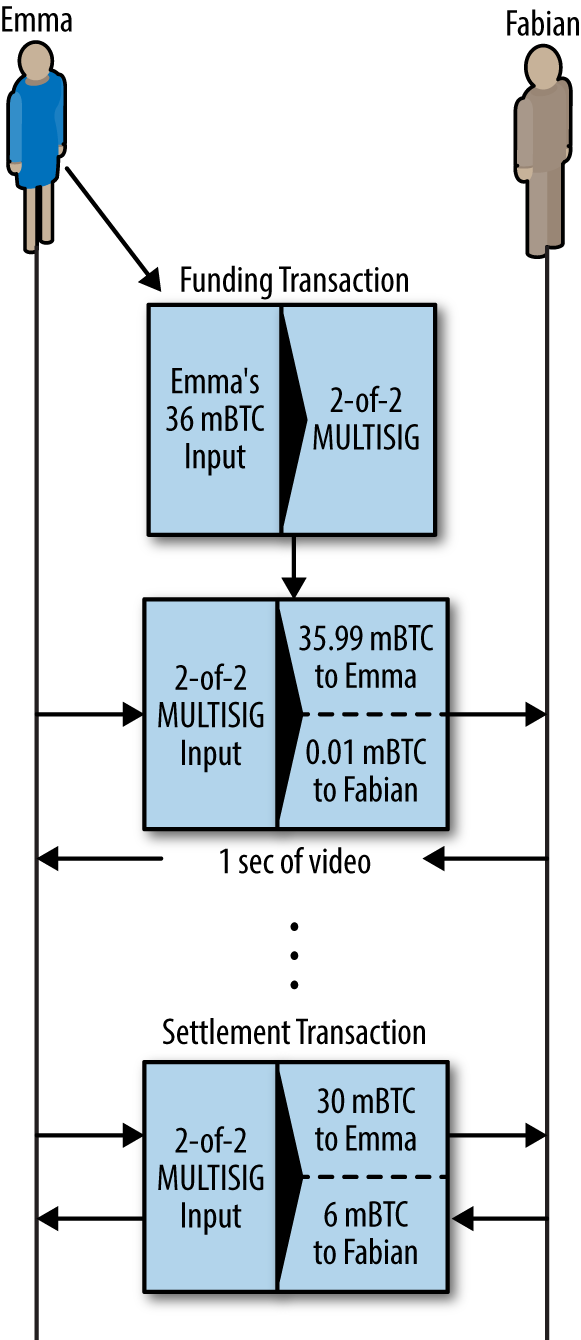
Le Lightning Network est une possibilité de mise en place de canaux de paiement routés. Il existe plusieurs autres conceptions qui visent à atteindre des objectifs similaires, telles que Teechan et Tumblebit.

### Exemple de réseau Lightning de base

Voyons comment cela fonctionne.

Dans cet exemple, nous avons cinq participants : Alice, Bob, Carol, Diana et Eric. Ces cinq participants ont ouvert des canaux de paiement entre eux, par paires. Alice a un canal de paiement avec Bob. Bob est connecté à Carol, Carol à Diana et Diana à Eric. Pour simplifier, supposons que chaque chaîne est financée avec 2 bitcoins par chaque participant, pour une capacité totale de 4 bitcoins dans chaque chaîne.

[figure\_title](#lightning_network_fig) montre cinq participants dans un Lightning Network, connectés par des canaux de paiement bidirectionnels qui peuvent être liés pour effectuer un paiement d'Alice à Eric ([Canaux de paiement routés (Lightning Network)](#lightning_network)).



Une série de canaux de paiement bidirectionnels liés pour former un réseau Lightning qui peut acheminer un paiement d'Alice à Eric

Alice veut payer Eric 1 bitcoin. Cependant, Alice n'est pas connectée à Eric par un canal de paiement. La création d'un canal de paiement nécessite une transaction de financement, qui doit être engagée dans la chaîne de blocs Bitcoin. Alice ne veut pas ouvrir un nouveau canal de paiement et engager davantage de ses fonds. Y a-t-il un moyen de payer Eric, indirectement ?

[formalpara\_title](#ln_payment_process) montre le processus étape par étape d'acheminement d'un paiement d'Alice à Eric, à travers une série d'engagements HTLC sur les canaux de paiement reliant les participants.

**Acheminement des paiements étape par étape via un réseau Lightning**

image ::images/mbc2\_1207.png["Acheminement des paiements étape par étape via un réseau Lightning"]

Alice exécute un nœud Lightning Network (LN) qui assure le suivi de son canal de paiement vers Bob et a la capacité de découvrir des itinéraires entre les canaux de paiement. Le nœud LN d'Alice a également la capacité de se connecter via Internet au nœud LN d'Eric. Le nœud LN d'Eric crée un R secret à l'aide d'un générateur de nombres aléatoires. Le noeud d'Eric ne révèle ce secret à personne. Au lieu de cela, le nœud d'Eric calcule un hachage H du secret R et transmet ce hachage au nœud d'Alice (voir [formalpara\_title](#ln_payment_process) étape 1).

Maintenant, le nœud LN d'Alice construit une route entre le nœud LN d'Alice et le nœud LN d'Eric. L'algorithme de routage utilisé sera examiné plus en détail plus tard, mais pour l'instant supposons que le nœud d'Alice puisse trouver une route efficace.

Le nœud d'Alice construit alors un HTLC, payable au hash H, avec un délai de remboursement de 10 blocs (bloc actuel + 10), pour un montant de 1,003 bitcoin (voir [formalpara\_title](#ln_payment_process) étape 2). Le supplément de 0,003 sera utilisé pour compenser les nœuds intermédiaires pour leur participation à cette voie de paiement. Alice offre ce HTLC à Bob, déduisant 1,003 bitcoin de son solde de canal avec Bob et l'engageant sur le HTLC. Le HTLC a la signification suivante : *"Alice engage 1,003 de son solde de canal à payer à Bob si Bob connaît le secret, ou remboursé sur le solde d'Alice si 10 blocs s'écoulent."* Le solde de canal entre Alice et Bob est maintenant exprimé par des transactions d'engagement avec trois sorties : solde de 2 bitcoins à Bob, solde de 0,997 bitcoin à Alice, 1,003 bitcoin engagé dans le HTLC d'Alice. Le solde d'Alice est réduit du montant engagé dans le HTLC.

Bob s'est maintenant engagé à ce que s'il parvient à obtenir le secret + R + dans les 10 prochains blocs, il puisse réclamer les 1,003 verrouillés par Alice. Avec cet engagement en main, le nœud de Bob construit un HTLC sur son canal de paiement avec Carol. Le HTLC de Bob engage 1,002 bitcoin dans le hachage H pendant 9 blocs, que Carol peut échanger si elle a le secret R (voir [formalpara\_title](#ln_payment_process) étape 3). Bob sait que si Carol peut réclamer son HTLC, elle doit produire R. Si Bob a + R + dans neuf blocs, il peut l'utiliser pour lui réclamer le HTLC d'Alice. Il gagne également 0,001 bitcoin pour avoir engagé le solde de son canal pendant neuf blocs. Si Carol n'est pas en mesure de réclamer son HTLC et qu'il ne peut pas réclamer le HTLC d'Alice, tout revient aux soldes de canal précédents et personne n'est à perte. L'équilibre des canaux entre Bob et Carol est maintenant : 2 pour Carol, 0,998 pour Bob, 1,002 commis par Bob pour le HTLC.

Carol s'est maintenant engagée à ce que si elle obtient R dans les neuf prochains blocs, elle puisse réclamer 1,002 bitcoin verrouillé par Bob. Elle peut désormais s'engager sur HTLC sur sa chaîne avec Diana. Elle engage un HTLC de 1.001 bitcoin au hachage H, pour huit blocs, que Diana peut racheter si elle a le secret R (voir [formalpara\_title](#ln_payment_process) étape 4). Du point de vue de Carol, si cela fonctionne, elle est plus riche de 0,001 bitcoin et si ce n'est pas le cas, elle ne perd rien. Son HTLC à Diana n'est viable que si R est révélé, auquel cas elle peut réclamer le HTLC à Bob. L'équilibre des canaux entre Carol et Diana est désormais : 2 pour Diana, 0,999 pour Carol, 1,001 commis par Carol pour le HTLC.

Enfin, Diana peut proposer un HTLC à Eric, engageant 1 bitcoin pour sept blocs à hacher H (voir [formalpara\_title](#ln_payment_process) étape 5). L'équilibre des canaux entre Diana et Eric est maintenant : 2 à Eric, 1 à Diana, 1 commis par Diana au HTLC.

Cependant, à ce saut dans la route, Eric *a* le secret R. Il peut donc prétendre au HTLC proposé par Diana. Il envoie R à Diana et réclame le 1 bitcoin, l'ajoutant au solde de son canal (voir [formalpara\_title](#ln_payment_process) étape 6). L'équilibre des canaux est maintenant : 1 pour Diana, 3 pour Eric.

Maintenant, Diana a un R secret. Par conséquent, elle peut désormais réclamer le HTLC à Carol. Diana transmet R à Carol et ajoute le bitcoin 1.001 au solde de son canal (voir [formalpara\_title](#ln_payment_process) étape 7). Maintenant, l'équilibre des canaux entre Carol et Diana est : 0,999 pour Carol, 3,001 pour Diana. Diana a "gagné" 0,001 pour sa participation à cette voie de paiement.

En revenant sur le parcours, le secret R permet à chaque participant de réclamer les HTLC restants. Carol réclame 1,002 à Bob, fixant le solde sur leur chaîne à : 0,998 à Bob, 3,002 à Carol (voir [formalpara\_title](#ln_payment_process) étape 8). Enfin, Bob revendique le HTLC d'Alice (voir [formalpara\_title](#ln_payment_process) étape 9). Leur solde de canaux est mis à jour comme suit : 0,997 pour Alice, 3,003 pour Bob.

Alice a payé Eric 1 bitcoin sans ouvrir de chaîne à Eric. Aucune des parties intermédiaires de la voie de paiement n'avait à se faire confiance. Pour l'engagement à court terme de leurs fonds dans le canal, ils peuvent gagner une petite commission, le seul risque étant un léger retard de remboursement si le canal était fermé ou si le paiement acheminé échouait.

### Transport et routage du Lightning Network

Toutes les communications entre les nœuds LN sont cryptées point à point. De plus, les nœuds ont une clé publique à long terme qu'ils utilisent comme identifiant et pour s'authentifier les uns les autres.

Chaque fois qu'un nœud souhaite envoyer un paiement à un autre nœud, il doit d'abord construire un *chemin* à travers le réseau en connectant des canaux de paiement avec une capacité suffisante. Les nœuds annoncent les informations de routage, y compris les canaux qu'ils ont ouverts, la capacité de chaque canal et les frais qu'ils facturent pour acheminer les paiements. Les informations de routage peuvent être partagées de diverses manières et différents protocoles de routage sont susceptibles d'émerger à mesure que la technologie Lightning Network progresse. Certaines implémentations de Lightning Network utilisent le protocole IRC comme mécanisme pratique permettant aux nœuds d'annoncer les informations de routage. Une autre implémentation de la découverte de route utilise un modèle P2P où les nœuds propagent les annonces de canal à leurs pairs, dans un modèle d'"inondation", similaire à la façon dont bitcoin propage les transactions. Les plans futurs incluent une proposition appelée [Flare](https://bit.ly/2r5TACm), qui est un modèle de routage hybride avec des "quartiers" de nœuds locaux et des nœuds de balise à plus longue portée.

Dans notre exemple précédent, le nœud d'Alice utilise l'un de ces mécanismes de découverte de route pour trouver un ou plusieurs chemins reliant son nœud au nœud d'Eric. Une fois que le nœud d'Alice a construit un chemin, elle initialisera ce chemin à travers le réseau, en propageant une série d'instructions cryptées et imbriquées pour connecter chacun des canaux de paiement adjacents.

Il est important de noter que ce chemin n'est connu que du nœud d'Alice. Tous les autres participants de la route de paiement ne voient que les nœuds adjacents. Du point de vue de Carol, cela ressemble à un paiement de Bob à Diana. Carol ne sait pas que Bob transmet en fait un paiement d'Alice. Elle ne sait pas non plus que Diana transmettra un paiement à Eric.

Il s'agit d'une fonctionnalité essentielle du Lightning Network, car elle garantit la confidentialité des paiements et rend très difficile l'application de la surveillance, de la censure ou des listes noires. Mais comment Alice établit-elle ce chemin de paiement, sans rien révéler aux nœuds intermédiaires ?

Le Lightning Network implémente un protocole de routage en oignon basé sur un schéma appelé [Sphinx](https://bit.ly/2q6ZDrP). Ce protocole de routage garantit qu'un expéditeur de paiement peut construire et communiquer un chemin à travers le Lightning Network tel que :

* Les nœuds intermédiaires peuvent vérifier et décrypter leur portion d'informations de route et trouver le saut suivant.
* À part les sauts précédents et suivants, ils ne peuvent pas en savoir plus sur les autres nœuds faisant partie du chemin.
* Ils ne peuvent pas identifier la longueur du chemin de paiement, ni leur propre position dans ce chemin.
* Chaque partie du chemin est cryptée de telle manière qu'un attaquant au niveau du réseau ne peut pas associer les paquets de différentes parties du chemin les uns aux autres.
* Contrairement à Tor (un protocole d'anonymisation routé par couches d’oignon sur Internet), il n'y a pas de "nœuds de sortie" qui peuvent être placés sous surveillance. Les paiements n'ont pas besoin d'être transmis à la chaîne de blocs Bitcoin ; les nœuds mettent simplement à jour les soldes des canaux.

En utilisant ce protocole routé en couches d’oignon, Alice enveloppe chaque élément du chemin dans une couche de cryptage, en commençant par la fin et en remontant. Elle crypte un message à Eric avec la clé publique d'Eric. Ce message est enveloppé dans un message crypté pour Diana, identifiant Eric comme le prochain destinataire. Le message à Diana est enveloppé dans un message crypté avec la clé publique de Carol et identifiant Diana comme le prochain destinataire. Le message à Carol est crypté avec la clé de Bob. Ainsi, Alice a construit cet "oignon" multicouche chiffré de messages. Elle l'envoie à Bob, qui ne peut que déchiffrer et déballer la couche externe. À l'intérieur, Bob trouve un message adressé à Carol qu'il peut transmettre à Carol mais ne peut pas se déchiffrer. En suivant le chemin, les messages sont transférés, décryptés, transférés, etc., jusqu'à Eric. Chaque participant ne connaît que le nœud précédent et suivant dans chaque saut.

Chaque élément du chemin contient des informations sur le HTLC qui doivent être étendues au saut suivant, le montant qui est envoyé, les frais à inclure et l'expiration du temps de verrouillage CLTV (en blocs) du HTLC. Au fur et à mesure que les informations de route se propagent, les nœuds effectuent des engagements HTLC jusqu'au saut suivant.

À ce stade, vous vous demandez peut-être comment il est possible que les nœuds ne connaissent pas la longueur du chemin et leur position dans ce chemin. Après tout, ils reçoivent un message et le transmettent au saut suivant. Ne devient-il pas plus court, leur permettant de déduire la taille du chemin et leur position ? Pour éviter cela, le chemin est toujours fixé à 20 sauts et rempli de données aléatoires. Chaque nœud voit le saut suivant et un message chiffré de longueur fixe à transmettre. Seul le destinataire final voit qu'il n'y a pas de saut suivant. Pour tout le monde, il semble qu'il y ait toujours 20 sauts de plus à faire.

### Avantages du Lightning Network

Un Lightning Network est une technologie de routage de deuxième couche. Il peut être appliqué à toute chaîne de blocs prenant en charge certaines fonctionnalités de base, telles que les transactions multisignatures, les verrous temporels et les contrats intelligents de base.

Si un réseau Lightning est superposé au réseau Bitcoin, le réseau Bitcoin peut bénéficier d'une augmentation significative de la capacité, de la confidentialité, de la granularité et de la vitesse, sans sacrifier les principes de fonctionnement sans confiance sans intermédiaires :

Confidentialité

Les paiements Lightning Network sont beaucoup plus privés que les paiements sur la chaîne de blocs Bitcoin, car ils ne sont pas publics. Bien que les participants à une route puissent voir les paiements se propager sur leurs canaux, ils ne connaissent ni l'expéditeur ni le destinataire.

Fongibilité

Un Lightning Network rend beaucoup plus difficile l'application de la surveillance et des listes noires sur le bitcoin, augmentant la fongibilité de la monnaie.

Vitesse

Les transactions Bitcoin utilisant Lightning Network sont réglées en millisecondes, plutôt qu'en minutes, car les HTLC sont effacés sans valider les transactions dans un bloc.

Granularité

Un réseau Lightning peut permettre des paiements au moins aussi petits que la limite de "poussière" de bitcoin, peut-être même plus petite. Certaines propositions permettent des incréments subsatoshi.

Capacité

Un Lightning Network augmente la capacité du système Bitcoin de plusieurs ordres de grandeur. Il n'y a pas de limite supérieure pratique au nombre de paiements par seconde pouvant être acheminés sur un réseau Lightning, car cela dépend uniquement de la capacité et de la vitesse de chaque nœud.

Fonctionnement sans confiance

Un réseau Lightning utilise des transactions bitcoin entre des nœuds qui fonctionnent comme des pairs sans se faire confiance. Ainsi, un Lightning Network préserve les principes du système Bitcoin, tout en élargissant considérablement ses paramètres de fonctionnement.

Bien sûr, comme mentionné précédemment, le protocole Lightning Network n'est pas le seul moyen de mettre en œuvre des canaux de paiement routés. D'autres systèmes proposés incluent Tumblebit et Teechan. À l'heure actuelle, cependant, le Lightning Network a déjà été déployé sur testnet. Plusieurs équipes différentes ont développé des implémentations concurrentes de LN et travaillent à une norme d'interopérabilité commune (appelée BOLT). Il est probable que Lightning Network sera le premier réseau de canaux de paiement routés à être déployé en production.

===Conclusion

Nous n'avons examiné que quelques-unes des applications émergentes qui peuvent être construites en utilisant la chaîne de blocs Bitcoin comme plate-forme de confiance. Ces applications élargissent la portée du bitcoin au-delà des paiements et au-delà des instruments financiers, pour englober de nombreuses autres applications où la confiance est essentielle. En décentralisant la base de confiance, la chaîne de blocs Bitcoin est une plate-forme qui engendrera de nombreuses applications révolutionnaires dans une grande variété d'industries.