BLOCKCHAIN& PLATFORM CHAIR

Zero-Knowledge : confiance et confidentialité à l'échelle industrielle

Perspectives n°1, septembre 2021

ÉCOLE POLYTECHNIQUE
INSTITUT POLYTECHNIQUE DE PARIS

Introduction

Un des freins principaux au déploiement des blockchains réside dans le fait que les données gérées sur la blockchain sont accessibles publiquement. Cela est inenvisageable dans les domaines de la santé ou le domaine bancaire, par exemple.

Les technologies de *zero-knowledge*[1] ou preuves à divulgation nulle de connaissance peuvent précisément résoudre cette contradiction. Ces technologies peuvent être mises en œuvre soit en déployant une blockchain conçue spécialement pour intégrer du *zero-knowledge*, soit en déployant des briques logicielles *zero-knowledge* sur une blockchain existante capable technologiquement de les intégrer.

Ce concept de *zero-knowledge* est si étonnant et prometteur qu'il a valu à ses inventeurs, Shafi Goldwasser et Silvio Micali, le prix Turing en 2012.

Pour présenter cette technologie contre-intuitive et la mettre en perspective avec les usages et les services, la chaire « **Blockchain and B2B Platforms** » hébergée à l'**École Polytechnique**, et soutenue par **CapGemini**, **NomadicLabs** et la **Caisse des dépôts**, a choisi d'interroger deux doctorants, Sarah Bordage et Youssef El Housni, effectuant leur recherche au laboratoire d'informatique de l'**École polytechnique** dans le cadre de la chaire, ainsi que Anthony Simonet-Boulogne et Gilles Fedak, de la startup **iExec**, qui considèrent le *zero-knowledge* comme une technologie à intégrer dans leur offre.

[1] Nous utiliserons le terme anglophone par souci de concision.

Qu'est ce que le « zero-knowledge » ?

Daniel Augot: Dans le monde des blockchains et des cryptomonnaies, on entend beaucoup parler de *zero-knowledge* comme solution aux problèmes de *privacy*, c'est-à-dire de confidentialité et de respect de la vie privée. En effet, cette technologie peut satisfaire les deux objectifs contradictoires de publicité de l'information (ou d'enregistrement ou de partage de l'information) et, à l'opposé, de confidentialité de cette information.

D'une manière très simplifiée, le principe est le suivant: une « trace » ou « empreinte » minimale (**commitment cryptographique**[2]**, ou mise en gage**) d'une information est enregistrée sur une blockchain, publique ou non (par exemple une transaction). Sur la base de cette trace il est possible de prouver des faits portants sur cette information sans révéler l'information elle-même. Une mise en gage, ou commitment, d'une information consiste à produire un sceau, éventuellement très court, qui fait que seule cette information est certifiée par ce sceau. Ce sceau peut aussi cacher l'information certifiée.

Les applications sont: confidentialité des transactions financières sur une blockchain, gestion de l'identité, preuve de solvabilité, protection des données de santé, etc.

Comment est-ce possible? Cela semble contre-intuitif… Une présentation ludique reposant sur le jeu de Charlie est présentée en encadré.

Où est Charlie?

Le jeu est bien connu: il s'agit de ces livres avec de grandes illustrations très denses et très riches en détails. Charlie s'y cache, facilement reconnaissable, une fois qu'on l'a trouvé. Un enfant (le prouveur) va prouver à son père (le vérificateur) qu'il a trouvé Charlie sans montrer où est Charlie.

Il procède de la manière suivante:

1. il prépare un bristol grand comme neuf fois la page
2. il découpe dans ce bristol la silhouette de Charlie
3. il présente ce bristol face à son père
4. il positionne le livre derrière le bristol de sorte à faire apparaître Charlie dans la découpe
5. le père voit Charlie à travers la découpe mais ne voit pas la position de Charlie dans la page.

Le père est convaincu que son enfant sait où est Charlie, sans apprendre où est Charlie.

[2] Les mots en caractère gras sont définis dans le corps du texte ou dans le glossaire p18

Qu'apporte le zero-knowledge *à la confidentialité des données ?*

Daniel Augot : Généralement lorsqu'on parle de confidentialité des données, on parle de chiffrement, c'est-à-dire d'un algorithme cryptographique qui permet de rendre les données inintelligibles, à l'aide d'une clé de chiffrement. Le seul moyen de retrouver les données est de les déchiffrer, à l'aide de la clé de déchiffrement. Ainsi, le destinataire légitime peut déchiffrer ces données avec cette clé de déchiffrement. C'est une solution tout ou rien, soit on ne sait rien, soit on voit tout, et la possession de la clé de déchiffrement détermine qui a accès aux informations en clair.

En revanche, les preuves *zero-knowledge* permettent de prouver la véracité d'énoncés sur des données qu'on aura gardées cachées par le chiffrement (ou le **hachage** cryptographique ou **mise en gage**). Ces preuves ne révèlent aucune autre information que le fait que ces propriétés ou **énoncés** sont vrais. Par exemple, si on enregistre une empreinte des données d'un passeport, l'utilisateur de ce passeport pourra prouver qu'il est majeur sans révéler son âge ou d'autres informations du passeport. Dans le monde bancaire, les comptes en banque seraient enregistrés et accompagnés de la publication d'une empreinte publique (la racine d'un **arbre de Merkle**. Un **arbre de Merkle** permet de mettre en gage un grand nombre de données avec une empreinte très courte. telle que chaque donnée peut être certifiée individuellement). Sur la base de cette empreinte, la banque peut prouver que tous les comptes ne dépassent pas un certain seuil de déclaration sans révéler l'état des comptes.

Des progrès récents permettent de dépasser le *zero-knowledge* qui portait historiquement sur des **énoncés** abstraits et mathématiques, issus de la théorie des nombres. Maintenant, on peut traiter des **énoncés** de la vie courante, comme ceux décrits précédemment pour l'âge ou pour le seuil de déclaration des comptes en banque. Essentiellement, c'est une sorte de compilation d'un langage de haut niveau vers des équations mathématiques (voir encadré).

Une deuxième avancée importante, grâce à cette flexibilité qui a été ouverte, est de permettre des preuves que des calculs ont été correctement exécutés, sans nécessiter de refaire le calcul, et sans révéler toutes les informations nécessaires à ce calcul. Ces calculs portent sur des données, certaines publiques, d'autres privées et on peut prouver que le résultat est correct sans révéler les données privées. Pour dépasser l'exemple bancaire précédent qui consiste à vérifier un seuil statique, la banque a la possibilité de produire un résultat de calcul plus dynamique, comme la médiane ou la moyenne des comptes sous sa gestion, assortie d'une preuve *zero-knowledge* qui permet de vérifier que ce calcul est juste. En résumé, celui qui a fait un calcul complexe sur des données peut proclamer que ce résultat est correct en produisant la preuve associée, sans toutefois révéler les données dont il est question.

Aspects techniques d'une preuve *zero-knowledge*

D'abord un peu de terminologie : le **prouveur** est la personne qui produit une preuve convaincante (dans l'exemple de Charlie c'est l'enfant), et le **vérificateur** est la personne qui vérifie la preuve (dans l'exemple de Charlie c'est le parent). Le prouveur utilisera un algorithme de preuve pour construire une preuve et le vérificateur un algorithme de vérification pour vérifier une preuve.

Historiquement, les cryptographes ont construit depuis les années quatre-vingt des protocoles *zero-knowledge* bas niveau, qui permettent, entre autres, de prouver qu'un système d'équations algébriques a une solution, qu'un « logarithme discret » est connu, etc. Ces protocoles bas niveau s'expriment en termes mathématiques et informatiques, très éloignés de problèmes concrets de la vie courante.

Mais souvenons-nous qu'en informatique tout traitement d'information se ramène à des manipulations de bits, c'est-à-dire des 0 et des 1. Pour faciliter le travail du programmeur, un programme informatique s'écrit en langage de haut niveau, par exemple rust ou C++, il est ensuite compilé en un langage bas niveau compréhensible par la machine.

Ici, la situation est la même : un **énoncé** naturel demandant une preuve naturelle est compilé en un **énoncé** mathématique. La preuve naturelle est de même compilée en preuve bas niveau, mathématique, qui sera traitée par le protocole *zero-knowledge* cryptographique et mathématique bas niveau. Vérifier en *zero-knowledge* la preuve d'un **énoncé** de haut niveau revient à vérifier en *zero-knowledge* la preuve bas niveau associée.

On dit que le problème mathématique est le ***backend*** et que le langage de haut niveau est le ***frontend***. Cependant, la couche cryptographique basse peut avoir des spécificités selon le système de preuve qui impactent les applications du niveau supérieur. Dans cet entretien, nous considérons les **SNARK**s (*Succinct Non Interactive Argument of Knowledge*) (encadré 3) et **STARK**s (*Scalable Transparent ARguments of Knowledge*) (encadré) qui sont les deux technologies les plus avancées industriellement.

C'est une prouesse scientifique, technologique et d'ingénierie que d'intégrer les protocoles cryptographiques dans une suite logicielle rendant l'expression de problèmes naturels facile au programmeur. Il y a de plus un effort de standardisation de ces systèmes et protocoles, regroupant industriels et académiques.

Quels sont les critères de choix des systèmes de zero-knowledge *selon les applications ?*

Youssef El Housni : Pour une application donnée, on peut traiter cette questions selon deux axes d'analyse, selon les contraintes technologiques de l'environnement dans lequel sera déployée l'application ou selon les fonctionnalités et les besoins de l'application.

Sur le premier axe, pour analyser selon les contraintes technologiques, la question dans le cadre des blockchains et des ***smart contracts*** est de savoir comment et dans quel environnement sera exécuté l'algorithme de vérification de la preuve. Un ***smart contract*** est un programme enregistré de manière irrévocable dans la blockchain considérée, et dont l'exécution est déclenchée automatiquement. Il permet typiquement des transactions plus complexes que de simples transactions financières, et le développement d'outils financiers plus sophistiqués.

Par exemple, pour Ethereum, il est nécessaire que la machine virtuelle d'Ethereum qui exécute les ***smart contracts*** possède de manière précompilée des instructions complexes non standard. Ces instructions complexes non standard, implémentent notamment des opérations de courbes elliptiques avancées nécessaires pour vérifier *on-chain* une preuve *zero-knowledge*. Elles spécifient un contexte mathématique rigide, comme une courbe elliptique spécifique bien définie, qui contraint alors tout le système de preuve et pose une limite technologique. En effet, changer de courbe impliquerait une changement lourd consistant à ajouter de nouvelles instructions précompilées à la machine virtuelle d'Ethereum,

En revanche, pour une nouvelle application ou une nouvelle blockchain sans contexte préalable, la liberté du choix de la courbe elliptique permet d'intégrer les progrès les plus récents issus de la recherche.

Sarah Bordage : Pour les **STARK**s, les prérequis de la plateforme sont relativement légers, il faut une bibliothèque d'algorithmes permettant l'addition et la multiplication de grands nombres et de grands polynômes. Cela n'est pas aussi spécifique et mathématiquement complexe que le choix d'une courbe elliptique pour les **SNARK**s.

Youssef El Housni : Pour le deuxième axe, pour considérer un système *zero-knowledge* selon les fonctionnalités de l'application, il s'agit de caractériser les preuves par rapport à leur taille, au temps de génération des preuves et au temps de vérification des preuves. Dans une application où les preuves sont stockées durablement après vérification, la taille de la preuve est de première importance. Il faut donc que ces preuves soient les plus petites possibles, comme par exemple celles des **SNARK**s. Si l'application nécessite que le prouveur fournisse des preuves rapidement, on va de préférence utiliser des **STARK**s dont l'algorithme de preuve est plus rapide que celui des **SNARK**s.

D'autre part, un algorithme *zero-knowledge*, dont les temps de vérification dépendent de l'**énoncé** et qui ne sont pas constants, poserait des problèmes pour être utilisé dans une application qui vérifiait des énoncés différents de tailles variables. Cela impacterait grandement la performance de l'application qui ne serait plus en temps réel. Les **SNARK**s ont des temps de vérification constants, donc impactent moins, du côté du vérifieur, les performances des applications qui les utilisent.

Par exemple, dans **Zcash**, les preuves de chaque transaction sont stockées sur la blockchain et sont vérifiées par les mineurs : pour qu'elles soient courtes et à vérification rapide, on utilisera des **SNARK**s, qui permettent de contrôler le rythme de production des blocs.

	SNARKs	**STARK**s
Trusted Setup	OUI	NON
Universal Trusted Setup	OUI	Non applicable
Coût du calcul de la preuve	Rapide	Très rapide
Coût de vérification de la preuve	Constant	Logarithmique
Taille de preuve	Constante	Logarithmique

Anthony Simonet-Boulogne : De mon point de vue, ces différents protocoles *zero-knowledge* permettent justement d'ajuster le curseur entre la traçabilité des transactions d'un côté, primordiale dans les applications blockchain, et la confidentialité des données de l'autre. En définitive, ce sont donc effectivement les besoins particuliers de chaque application qui vont guider nos choix entre les différentes solutions.

Le temps de génération des preuves peut par exemple être très contraignant dans certaines situations. Un système de paiement comme **Zcash** a besoin de preuves extrêmement rapides à générer car il faut pouvoir émettre des transactions quasi instantanément, éventuellement sur un terminal léger (*smartphone*). En revanche, je pense qu'il existe de nombreux domaines d'applications, comme par exemple les **oracles** (un acteur qui saisit des informations externes dans la chaîne, qui intrinsèquement ne sont pas vérifiables par les mineurs ou validateurs avec les seules informations de la chaîne, par exemple une météo locale), où la confiance en l'information externe que l'on insère dans la blockchain est si importante, que l'on pourrait accepter des temps de génération de preuves beaucoup plus longs pour certifier cette information externe introduite par un **oracle**.

Youssef El Housni : Il est très important de considérer la problématique de la confiance dans la mise en place des **SNARK**s. En effet, celui qui génère la **chaîne structurée de référence**, les **algorithmes de preuve** et de **vérification** manipule des secrets qui lui permettent de faire des preuves fausses. Il faut donc lui faire confiance pour avoir détruit ces secrets. Dans une application où il n'y a qu'un seul type d'énoncé à prouver, un **SNARK** est pertinent car sa mise en place ne doit être faite qu'une fois. Ainsi pour la cryptomonnaie **Zcash**, où les transactions sont secrètes, il n'y a qu'un seul type de preuve qui sert à vérifier la validité des transactions selon les critères du protocole de **Zcash** : pas de création monétaire, pas de double dépense, légitimé de l'émetteur de la transaction, etc. Une fois ces critères spécifiés, donc le type d'énoncé à vérifier (on parle de **langage**), tout est fixé en dur et une seule mise en place est nécessaire pour le système de preuve associé. La solution des **SNARK**s est appropriée.

En revanche, dans une application de type ***smart contract*** sur Ethereum, qui vérifie on-chain une preuve *zero-knowledge*, dépendante de la logique du ***smart contract***, il faudrait faire une mise en place de la ***chaîne structurée de référence*** pour chaque nouveau ***smart contract*** devant vérifier des preuves *zero-knowledge*. La variabilité des énoncés à vérifier et la complexité de leur mise en place font qu'il n'est pas raisonnable de faire un tel déploiement de ***trusted setup*** pour chaque ***smart contract***. Il faudrait alors envisager des **STARK**s, qui ne nécessitent pas une telle mise en place du cadre initial de confiance.

Les **SNARK**s : Succint Non Interactive Arguments of Knowledge

Youssef El Housni : Mon sujet de thèse porte sur les **SNARK**s. La mise en place des **SNARK**s requiert un tiers de confiance. Ce tiers de confiance génère des données qu'on appelle la **chaîne structurée de référence**, qui est une chaîne d'octets nécessaires aux fonctionnements des algorithmes de preuve et de vérification. Cette **chaîne structurée de référence** est dite **trappée**, cela signifie que des secrets (des « logarithmes discrets ») ont été nécessaires à la mise en place de cette chaîne, qui est en revanche publique. Celui qui connaît ces secrets a le moyen de fabriquer des preuves fausses : c'est une **trappe**. Il faut donc lui faire confiance pour ne pas utiliser ce pouvoir et pour avoir détruit les secrets une fois la chaîne de référence publiée. Cette trappe est familièrement appelée **déchet toxique**. De plus, le résultat de la mise en place est spécifique à l'**énoncé** prouvé : il faut le refaire pour chaque chaque type d'énoncé (langage). Cependant des progrès cryptographiques ont permis de construire des **SNARK**s universels, avec une mise en place valide pour plusieurs **énoncés**, par exemple **PLONK**.

En termes mathématiques, les **SNARK**s reposent sur la théorie des nombres et l'arithmétique : courbes elliptiques sur les corps finis (ces objets permettent une arithmétique rudimentaire, limitée à l'addition), couplages (permettent des opérations plus évoluées, comme la multiplication), ainsi que sur les problèmes algorithmiques et cryptographiques associés. On peut ainsi coder un problème naturel en un problème d'additions et de multiplications portant sur des objets cachés. La **chaîne structurée de référence** est en réalité un certain nombre de points de courbes elliptiques, dont le logarithme discret est connu par celui qui fait la mise en place.

Une fois la mise en place faite pour un énoncé donné, n'importe quel prouveur peut utiliser l'algorithme de preuve et la **chaîne structurée de référence** associée pour produire une preuve d'un énoncé sans révéler ce qui fait que l'énoncé est vrai. N'importe quel **vérificateur** peut utiliser l'algorithme de vérification, la preuve et la **chaîne structurée de référence** pour vérifier que l'énoncé est vrai.

La taille de la preuve des **SNARK**s est très petite (quelques centaines d'octets) et ne dépend pas de la complexité de l'**énoncé** à prouver (cette complexité se retrouve cependant dans la **chaîne structurée de référence** qui croît avec l'**énoncé** à prouver). En conséquence l'algorithme de vérification des **SNARK**s est non seulement très rapide mais il est surtout de coût constant par rapport à la taille de l'**énoncé**.

Les **SNARK**s ont été rendus populaires dans le monde des blockchains par l'irruption de la cryptomonnaie **Zcash**. Dans le protocole de **Zcash**, les utilisateurs ne diffusent pas leurs transactions monétaires en clair mais en diffusent une version opaque dont seul un ***commitment*** sera rendu public assorti d'une preuve *zero-knowledge*. Cette preuve *zero-knowledge* permet aux mineurs de vérifier publiquement que la transaction est correcte, alors même que l'émetteur, le destinataire et les montants sont cachés.

La petite taille des preuves **SNARK**s et la rapidité de leur vérification font qu'elles peuvent être vérifiées et inscrites dans la blockchain de **Zcash** sans impacter les performances des mineurs de la blockchain. La problématique du tiers de confiance lors de la mise en place de la chaîne structurée de référence pose un problème crucial. En effet, celui ou ceux qui connaissent la **trappe** peuvent faire des transactions invalides dont la preuve sera pourtant acceptée comme correcte. Dans le cadre de **Zcash**, c'est toujours le même type d'**énoncé** qui est prouvé, celui défini par les contraintes de validité des transactions. Ce problème ne se pose ici qu'une fois.

Pour remédier à cela, les développeurs ont eu recours à des protocoles distribués complexes dits **MPC** (*secure multiparty computation* - **MPC**. Un protocole **MPC** permet à plusieurs participants de collaborer à un calcul commun, dépendant de chaque secret de chaque participant, sans que les participants révèlent leurs secrets).

Chaque participant contribue au protocole avec son propre secret, ce qui aboutit à la mise en place de la **chaîne structurée de référence** sans qu'aucun d'entre eux ne connaisse le secret global (la **trappe**) associée à cette chaîne structurée de référence

Consensys, mon employeur, développe la librairie « gnark », qui permet de manière autonome de mettre en place la chaîne structurée de référence, l'algorithme de preuve, l'algorithme de vérification, pour des énoncés écrits dans le langage Go. Des couches applicatives supérieures propres aux blockchains peuvent utiliser la librairie gnark.

Pourquoi le zero-knowledge *est-il souvent mentionné comme solution au problème du passage à l'échelle ?*

Youssef El Housni : On parle du *zero-knowledge* comme d'une solution pour assurer la confidentialité des données mais il peut être aussi utilisé pour résoudre le problème de la scalabilité. Précédemment, on parlait des types et des familles de *zero-knowledge* qu'on peut caractériser notamment par la taille de la preuve. Les ***rollups*** sont une application directe des **SNARK**s (ou des **STARK**s) grâce à leurs preuves très courtes (constantes pour les **SNARK**s, logarithmiques en la taille du calcul pour les **STARK**s).

Le principe du ***rollup***, comme son nom l'indique, est de regrouper de nombreuses transactions *off-chain* qui vont être enregistrées en une seule transaction courte sur la blockchain, accompagnée de la preuve que toutes ces transactions sont correctes. On résout le problème de scalabilité en augmentant ainsi la bande passante : une seule transaction courte on-chain encode en réalité des milliers de transactions, avec une taille minuscule de la preuve de la correction de ces transactions. Cela encombre peu la blockchain et les mineurs s'assurent rapidement que toutes ces transactions sont correctes.

Ce n'est donc pas essentiellement la composante *zero-knowledge* qui est importante ici. C'est surtout le fait que la preuve soit de taille constante ou logarithmique qui est intéressant.

Les **STARK**s : scalable transparent arguments of knowledge

Sarah Bordage : Comme les **SNARK**s, les **STARK**s qui constituent mon sujet de thèse, sont des preuves qui accompagnent le résultat d'un calcul pour certifier que ce calcul a été correctement effectué. Les **STARK**s peuvent être *zero-knowledge*, auquel cas le vérificateur ne gagne aucune information sur une entrée secrète du calcul. On parle alors de **ZK-STARK**s.

Au-delà des problèmes de confidentialité et vie privée, les preuves **STARK**s se positionnent comme une solution au problème de scalabilité qui émerge naturellement lorsque l'on passe d'un réseau centralisé à un réseau pair-à-pair, tel que celui d'une blockchain. Le coût de vérification d'une preuve **STARK** est logarithmique en la taille du calcul, et un prouveur peut prendre en charge un gros lot de transactions, produire une unique preuve **STARK** certifiant la validité de toutes les transactions. Ensuite, cette preuve peut être vérifiée on-chain par les mineurs en utilisant très peu de puissance de calcul. Dans le cadre d'une démonstration sur le réseau Ethereum, le système StarkEx a généré une unique preuve **STARK** pour vérifier 300 000 transactions, à un rythme de 3 000 transactions par seconde, avec un coût de 315 gas par transaction.

Les **ZK-STARK**s présentent plusieurs avantages par rapport aux **ZK-SNARK**s. En premier lieu, leur sécurité ne dépend pas de la confiance portée à la bonne exécution d'un algorithme de génération de clés (***trusted setup***) : il n'y a pas de **structured reference string**, ni de **trappe** secrète qui pourrait être exploitée pour forger des preuves valides d'affirmations fausses.

En termes de comparaison, les **SNARK**s fournissent des preuves de tailles constantes, plus courtes que celles des **STARK**s, grâce à la phase de mise en place de la **chaîne structurée de référence** qui encode toute la complexité de l'énoncé associé. Cette phase de mise en place, effectuée pendant l'exécution du ***trusted setup***, est spécifique au programme à vérifier. La contrepartie est que les preuves sont plus longues que celles des **SNARK**s. Elles sont cependant extrêmement courtes : logarithmique en la taille du calcul à prouver.

Si la sécurité de l'application *zero-knowledge* doit résister à l'ordinateur quantique, par exemple pour une sécurité à très long terme, il faut se tourner vers les **STARK**s pour lesquelles il n'y a vraisemblablement pas d'attaque quantique.

Les **STARK**s procurent une grande flexibilité, notamment grâce à la plateforme de développement Cairo, récemment développée par la start-up Starkware. Cairo est le premier langage permettant d'écrire des programmes génériques vérifiables par des preuves **STARK**s. Un programme écrit en Cairo retourne une trace d'exécution qui est envoyée à un prouveur **STARK**. À partir de cette trace, l'algorithme de preuve génère une preuve **STARK** pour certifier la validité du calcul représenté par le programme Cairo. Pour les applications blockchain, l'algorithme de preuve est typiquement exécuté en dehors du réseau blockchain (*off-chain*). En revanche un algorithme de vérification **STARK** peut alors être déployé, sans **trusted setup**, via un ***smart contract*** unique pour toute preuve, capable de valider l'exécution de n'importe quel programme Cairo.

Les **STARK**s sont utilisées en production dans le système StarkEx : un ensemble de ***smart contracts*** déployés sur Ethereum. Il s'agit d'un ***rollup*** qui permet de vérifier *onchain* tout un lot de transactions *offchain*, dont une preuve **STARK** est fournie par le système. Des applications de finance décentralisée (DeFi) comme dYdX (échanges décentralisés), DeversiFi (*trades et swaps*) et Immutable (*blockchain gaming*) utilisent à leur tour la plateforme Starkex.

Comment le prix du gaz d'Ethereum impacte le développement et l'utilisation du zero-knowledge *?*

Youssef El Housni : Rappelons-nous que dans le cas des **SNARK**s, il y a trois algorithmes : un algorithme de mise en place, un algorithme de production de la preuve et un algorithme de vérification de la preuve. Les deux premiers algorithmes de mise en place et de génération de la preuve ne se font pas sur la blockchain (ils utilisent notamment des quantités et des données secrètes). Seul l'algorithme de vérification de preuve est exécuté sur la blockchain par les mineurs ou validateurs. C'est donc ce dernier algorithme qui va impacter le coût de l'utilisation de *zero-knowledge* sur une blockchain. Dans le cas d'une blockchain, comme Ethereum où il faut payer du gaz pour vérifier une preuve, on aimerait que la vérification soit la moins coûteuse possible en gaz. Idéalement, il faudrait que la vérification ait toujours le même coût pour n'importe quel type d'**énoncé** à vérifier, quelle que soit sa complexité. C'est bien le cas des **SNARK**s qui permettent d'avoir un coût de vérification de preuve constant en termes d'opérations, indépendamment de l'**énoncé** prouvé, donc un coût on-chain de vérification des preuves parfaitement maîtrisé.

Sarah Bordage : En revanche, dans le cas des **STARK**s, le coût de la vérification, bien que logarithmique, dépend de la complexité de l'**énoncé** qu'on veut vérifier. Dans le cas d'Ethereum, on a donc un coût on-chain variable suivant l'**énoncé** (la logique du ***smart contract***) et non plus constant, bien que très petit. C'est la contrepartie de la transparence apportée par les **STARK**s qui ne demandent pas de ***trusted setup*** et permettent un déploiement sans nécessiter de confiance.

Cryptomonnaie Zcash : un premier exemple de déploiement à grande échelle

La cryptomonnaie **Zcash**, basée sur le protocole **zerocash**, a contribué à faire connaître le terme de **ZK-SNARK** et la notion de *zero-knowledge*. Cette cryptomonnaie permet de faire des transactions transparentes comme Bitcoin ou Ethereum ou des transactions opaques. Pour les transactions opaques, **Zcash** repose sur la notion de billet, caché par son ***commitment***. Un billet **Zcash** décrit de manière non explicite un montant et une adresse de paiement, en entrée et en sortie. À une adresse de paiement correspond une **clé privée** qui permet de dépenser le billet. Ces informations seront cachées : le montant et l'adresse du destinataire du paiement sont connues seulement de celui qui crée le billet.

À chaque billet est associé son ***commitment***, et un **annulateur** qui ne peut être calculé qu'avec la **clé privée** associée au billet. Il est impossible de faire le lien entre un **annulateur** et un ***commitment*** sans connaître la clé privée correspondante.

Un billet valide consommable à un instant donné est un billet dont le ***commitment*** existe publiquement et dont l'annulateur ne l'est pas. Le système enregistre publiquement les ***commitments*** et les **annulateurs**. Le système est ici considéré comme orthogonal à ce système de paiement *zero-knowledge*. Son rôle sera de vérifier la validité des transactions, qui manipulent les billets, comme indiqué ci-dessous.

Une transaction décrit de manière cachée un billet A en entrée et un billet en sortie B. Plus précisément, elle révèle l'**annulateur** N_A du billet dépensé, et révèle un ***commitment*** C_B du billet en sortie, sans révéler le billet B de sortie lui-même, ni à quel billet dépensé A correspond l'**annulateur**. Plus précisément, une transaction en clair serait formellement correcte si

1. Il existe un ***commitment*** public C_A d'un billet correspondant au billet d'entrée A
2. L'annulateur N_A est celui correspondant au billet d'entrée A du ***commitment*** précédent C_A
3. Le billet en entrée n'a pas déjà été dépensé (pas de double dépense)
4. Les montants en entrée et sortie correspondent (pas de création monétaire)
5. Le ***commitment*** C_B du billet en sortie est correctement formé, correspondant à une adresse de paiement.

Ces informations sont connues de celui qui crée la transaction. Les cinq propriétés ci-dessus sont vérifiables en mode *zero-knowledge* comme suit

1. Il existe un ***commitment*** public passé correspondant au billet d'entrée, mais on garde caché ce ***commitment***
2. L'annulateur révélé correspond au billet d'entrée du ***commitment*** précédent, mais on garde caché ce ***commitment***
3. L'annulateur est nouveau
4. Les montants en entrée et sortie correspondent mais ne sont pas révélés
5. Le ***commitment*** C_B du billet en sortie est correctement formé, correspondant à un billet B, sans révéler le billet de sortie.

Seul le 3. ci-dessus est vérifié publiquement : il est maintenu une liste des **annulateurs** déjà rendus publics. Une transaction révélant un annulateur déjà connu publiquement est considérée comme une double dépense, et sera rejetée.

Le rôle du système (par exemple des mineurs) est de vérifier les preuves *zero-knowledge* certifiant que les points 1. 2. 4. et 5. ci-dessus sont vrais, et maintenir la liste des annulateurs pour vérifier le 3.

On voit que l'**énoncé** à prouver (c'est la **langage** défini par les points 1 2 4 5) est ici défini par le protocole, et qu'il est toujours le même pendant toute l'exécution du protocole. La problématique du ***trusted setup*** ne se pose qu'une fois. Toutefois, si le protocole doit évoluer et que les critères de validité d'une transaction changent, il faut refaire le ***trusted setup***. Cependant, les **SNARK**s sont quand même privilégiés car les preuves sont extrêmement courtes et rapides à vérifier, avec un impact relativement faible sur le système.

Quels sont les autres coûts ou difficultés associés à un système de zero-knowledge ?

Gilles Fedak : Il existe deux coûts principaux pour gérer le *zero-knowledge* : le coût des calculs pour générer les preuves ainsi que celui associé à la mise en place du ***trusted setups***. Ce sont des calculs qui demandent beaucoup de ressources en calcul et en mémoire. Ils ne sont donc pas du tout adaptés à la blockchain. En revanche, du point de vue d'iExec qui est une solution de calcul distribué, il pourrait y avoir ici une opportunité d'utiliser notre infrastructure elle-même pour faire le calcul de preuves ou pour effectuer le ***trusted setup*** de façon distribuée, un peu à la manière de la cérémonie de Tau (voir encadré).

Anthony Simonet-Boulogne : Effectivement, le ***trusted setup*** pour les **SNARK**s est un exemple intéressant car il représente un coût non négligeable à la fois en temps de calcul et en complexité de mise en place. Nous avons réalisé que seule la personne qui a fait le ***trusted setup*** pourra par la suite être réellement persuadée que le **déchet toxique** a été éliminé. Cette personne aura confiance dans **la chaîne structurée de référence** parce qu'elle sait que le ***trusted setup*** a été fait correctement mais cette confiance est difficile à transmettre. En d'autres termes, il n'est pas facile de convaincre les gens en leur demandant de faire confiance à cette **chaîne de référence** et de croire que le ***déchet toxique*** est bien détruit.

Améliorations du *trusted setup* : cérémonie de Tau, cérémonie perpétuelle

Anthony Simonet-Boulogne : De mon point de vue, on a un problème de centralisation avec les **SNARK**s parce que leur mise en place est, d'une certaine manière, centralisée.

Youssef El Housni : C'est exact. Cependant, dans toutes les applications aujourd'hui en production, le trusted setup est fait avec un protocole de *Secure multiparty computation* **(MPC)**. Les protocoles de *Secure multiparty computation* permettent à des partenaires distincts d'obtenir un résultat commun d'un certain calcul, portant sur des données fournies par les participants sans que les participants aient besoin de partager leurs données entre eux. Ainsi, la ***structured reference string*** peut être construite à plusieurs partenaires sans qu'aucun ne sache au final les quantités secrètes qui forment le déchet toxique.

Cela a été fait historiquement pour la crypto-monnaie **Zcash** en 2016 pour la première fois et ensuite en 2017 selon la **cérémonie de Tau**. Depuis, il y a eu d'autres **MPC**s pour générer des **chaînes structurées de référence** pour des énoncés différents (*Tornado.cash ceremony, Aztec Ignition, Celo Plumo, filecoin ceremony*). Ces protocoles garantissent qu'il suffit qu'un seul participant soit honnête et détruise son entrée secrète pour que le **déchet toxique** ne puisse être reconstruit par aucun autre participant, ni même par un groupe se coalisant.

Anthony Simonet-Boulogne: Je voudrais ajouter aussi la notion de cérémonie de Tau perpétuelle, où des **chaînes structurées de référence** sont en permanence fabriquées, chaînées les unes aux autres. Toute personne peut ajouter sa contribution pour la nouvelle **chaîne structurée de référence**, et ainsi avoir confiance. Cela permet d'arriver dans le système et de prendre les clés générées par les ***setups*** précédents ou bien d'ajouter un maillon dans la chaîne avec son propre ***setup***. Si je ne souhaite pas bénéficier de l'entropie de la **chaîne structurée de référence** que les autres ont insérée avant moi, je peux rajouter de l'entropie que j'ai moi-même générée, et ainsi être persuadé que ce que j'ai entre les mains est de confiance. Mais une fois que j'ai fait cette opération très coûteuse, la confiance nouvellement gagnée ne fonctionne que pour moi car je ne peux pas la transmettre facilement à d'autres, sauf sous la forme d'une promesse.

*Une technologie souvent mentionnée quand il s'agit de protection des données est celle des « TEE » (*Trusted Execution Environment*) qui sont des technologies hardware proposées par les constructeurs (Intel, AMD, ARM). Pourquoi et comment iExec utilise ces technologies ?*

Gilles Fedak: Le terme ***Trusted Execution Environments*** (**environnements d'exécution sécurisés**, **TEE**) désigne une technologie matérielle implementée au niveau des processeurs, par exemple Intel avec les enclaves SGX, ARM avec TrustZone ou AMD avec SEV. L'idée est que l'on dispose d'une partie de la mémoire qui est en permanence chiffrée et qui ne peut être déchiffrée que dans l'enclave sécurisée. Ni les utilisateurs ni même le propriétaire de la machine ont accès en clair à ce que cette partie de mémoire protégée contient. Cette technologie est intéressante pour iExec parce qu'elle permet d'assurer le traitement correct et confidentiel sur des données privées par des machines en lesquelles nous n'avons pas confiance (elles sont fournies sur notre place de marché). L'utilisateur chiffre les données avant de les faire circuler sur le réseau et le **TEE** assure que ces données restent toujours confidentielles, même lors d'un traitement après déchiffrement dans le **TEE**. Cette technologie est donc et complémentaire au *zero-knowledge*, dans le sens où grâce à elle on peut s'assurer de la confidentialité des données *off-chain*, là où les *zero-knowledge* permettent de s'assurer de la confidentialité des informations dont la trace est stockée sur la blockchain.

Pour nous, il y a deux intérêts à utiliser la technologie **TEE**. Le premier est qu'elle permet aux utilisateurs de chiffrer leurs données afin qu'elles soient traitées par des programmes approuvés, tout en garantissant leur confidentialité. C'est ce que l'on appelle du chiffrement bout en bout. Le deuxième intérêt est que l'on peut rapporter et enregistrer dans la blockchain le fait que le résultat de l'exécution d'une application n'a pas pu être modifié ou altéré, ni par la personne qui possède la machine sur laquelle l'application est exécutée, ni par le réseau, ni par quiconque. Cela permet donc de reporter sur la blockchain une preuve de bonne exécution, de la bonne réalisation du service pour lequel les gens ont payé.

iExec est une startup technologique française fondée en 2016 par deux anciens chercheurs en informatique. Basée à Lyon, l'entreprise compte aujourd'hui 26 personnes réparties dans les services de R&D et de marketing. iExec développe la première place de marché décentralisée permettant à des particuliers ou des entreprises de monétiser leurs ressources informatiques. Il peut s'agir d'applications telles que des algorithmes d'intelligence artificielle, des jeux de données, ou des ressources de calcul, par exemple des ordinateurs durant leur temps d'inactivité.

Une particularité d'iExec est de s'être financée par une ICO (*Initial Coin Offering*) en avril 2017 qui a permis de lever l'équivalent de 11 M€ en Bitcoin. Une ICO consiste à émettre une nouvelle cryptomonnaie (ici le RLC), et à l'échanger contre des cryptomonnaies à court existant. Le jeton RLC est le moyen d'échange entre les parties prenantes de la place de marché; il est également une pièce maîtresse dans la conception des algorithmes qui régissent la place de marché et qui mettent en oeuvre des principes économiques visant à s'assurer de l'alignement des incitatifs. La mission d'iExec consiste donc à développer les technologies permettant de résoudre la confiance, la gouvernance et la sécurité dans les échanges au sein de la place de marché. Pour ce faire, iExec s'appuie principalement sur la blockchain et sur le calcul confidentiel, dont les preuves "zero-knowledge".

Les premiers utilisateurs d'iExec sont des startups blockchain ainsi que des grands partenaires industriels. Les cas d'usage mettent en oeuvre le partage de données confidentielles, l'utilisation de services dans des plates-formes de *smart-cities* ou encore la gouvernance sur l'accès aux données de mobilité utilisateurs par un opérateur d'autoroute.

Quel est l'intérêt de ces technologies pour iExec ?

Anthony Simonet-Boulogne : Le *zero-knowledge* et les ***rollups*** permettent de trouver un compromis, cette fois entre performance et centralisation. Pour permettre de passer à l'échelle, la plupart des ***sidechains*** reposent sur des protocoles ***Proof-of-Authority*** qui ont pour effet de, plus ou moins, « recentraliser » les consensus en introduisant des points de contrôle. Un protocole ***Proof-of-Authority*** consiste simplement à identifier des autorités qui ont des droits d'administration et de validation, et donc de contrôle. Avec le *zero-knowledge*, nous espérons des performances similaires mais en conservant la décentralisation, composante primordiale pour nous.

Gilles Fedak : iExec est une place de marché décentralisée et propose un protocole pour vérifier que toutes les transactions commerciales entre les parties prenantes se sont bien déroulées au niveau de confiance qu'elles avaient demandé au moment de passer le marché. Ce protocole que nous avons inventé s'appelle la « preuve de contribution » ou PoCo, il est implémenté sous forme de ***smart contracts*** sur la blockchain Ethereum.

Puisque ce protocole s'exécute via des ***smart contracts***, il faut payer aux mineurs d'Ethereum leur exécution à chaque fois qu'il se déclenche. Dans la blockchain Ethereum, il existe un mécanisme de gaz que l'on consomme pour payer l'exécution du ***smart contract***, cela représente un réel coût financier. De plus, l'exécution de ***smart contracts*** est relativement lente. Ainsi, puisque ce protocole se déroule à chaque interaction commerciale, il a un énorme impact sur les performances et donc pour les utilisateurs sur l'utilisabilité, l'expérience utilisateur, et la qualité de service.

Pour résoudre ce problème nous faisons tourner notre propre protocole sur une structure différente de la blockchain principale, plus rapide et moins chère. C'est ce que l'on appelle une ***sidechain***. Grâce à cette approche, PoCo s'exécute maintenant sans aucun coût de fonctionnement pour les utilisateurs car le gaz y est gratuit. De plus, des *bridges* forment des passerelles entre la blockchain principale et notre ***sidechain***. En fin de compte nous sommes effectivement plus rapides et nous n'avons plus de frais de fonctionnement, mais au prix de quelques inconvénients.

L'inconvénient principal est que nous avons un peu recentralisé l'ensemble du processus dans le sens où cette ***sidechain*** est administrée en grande partie par iExec. Les utilisateurs doivent en fin de compte nous accorder confiance pour administrer et sécuriser la ***sidechain***. De plus, les bridges eux-mêmes sont un facteur de centralisation et un point clé de confiance : puisque ce sont eux qui permettent de transmettre un état de la ***sidechain*** vers la chaîne principale, comme par exemple la preuve que des deals ont été passés et que des transactions commerciales donnant lieu à des paiements ont été réalisées. Cet état permet aux transactions sur la ***sidechain*** de se traduire en transferts de valeur sur la chaîne principale.

Ici, en termes de compromis entre performance et centralisation, nous avons mis le curseur sur la performance. On a perdu de la décentralisation mais nous avons gagné sur la performance économique et sur la performance en débit de transactions.

Ce que nous espérons, c'est que les ***rollups*** apportent une solution à ce problème. Une piste serait d'utiliser des ***rollups*** pour faire le lien entre les informations contenues dans la ***sidechain*** et dans la chaîne principale de façon sécurisée et à nouveau décentralisée. Les *zero-knowledge* pourraient servir à exprimer des conditions telles que : « Payez-moi sur la chaîne principale car je mets dans la chaîne principale la preuve que dans la ***sidechain***, une transaction commerciale s'est déroulée et que le service a été bien rendu ».

Daniel Augot : En conclusion de cet entretien, on voit donc que ce sujet du *zero-knowledge*, déjà très industrialisé par certaines startups, est très prometteur dans le monde des blockchains. Il a le potentiel de résoudre des problèmes industriels cruciaux, relatifs à la confidentialité et au passage à l'échelle. La discussion avec iExec a permis de mettre en avant l'intérêt industriel immédiat de ces technologies lié à la confidentialité. Il est fascinant qu'un sujet ancien ouvre maintenant de grandes perspectives mais la recherche est toujours active pour améliorer les performances des systèmes existants et remédier à certains défauts. La discussion que nous venons de faire sera peut-être caduque dans un an, même si la standardisation se profile à l'horizon. Restez à l'écoute !

Annexes

Glossaire

Documents pédagogiques

Références bibliographiques choisies

Bibliothèques informatiques bas niveau

Réalisations industrielles

Performances

Effort de standardisation

Glossaire

Pour illustrer les notions de zero-knowledge, nous utiliserons la notion de nombre composé à deux facteurs, N, dont les facteurs sont P et Q, tels que N = P* Q

- **Langage** : pour une **relation**, c'est l'ensemble des **instances** qui admettent un **témoin** qui satisfait la relation entre l'**instance** et le **témoin**. Exemple : le **langage** de l'« ensemble des nombres composés à deux facteurs premiers »
- **Instance** : des données publiques connues du prouver et du vérifieur. « N »
- **Relation** : lien logique entre une **instance** et un **témoin**. « N = P*Q »
- **Énoncé** : dire qu'une **instance** appartient à un **langage**. Exemple « N est composé avec deux facteurs premiers » : « N » est l'**instance** et le **langage** est celui de nombres composés de deux facteurs.
- **Preuve (ou témoin)** : des données connues seulement du prouveur, qui établissent que l'**instance** appartient au **langage**. Exemple « P et Q » tels que N = P*Q
- **Preuve zero-knowledge non interactive** : Une chaîne d'octets produite par un prouveur, qui peut être vérifiée par le vérifieur. Elle doit être courte, et cacher de l'information. Elle prouvera que N = P*Q sans révéler P et Q.
- ***Backend*** : une implémentation de protocoles cryptographiques et mathématiques bas niveau : courbes elliptiques sur les corps finis, codes correcteurs très longs.
- ***Frontend*** : Un moyen d'exprimer des **énoncés** et des preuves dans un langage informatique agréable, qui seront ensuite compilés en une représentation mathématique bas niveau adaptée au ***backend***.
- **Trappe, trappé** : un algorithme est dit **trappé** s'il existe une quantité secrète, la **trappe**, qui permet de modifier son comportement ou ses propriétés. Par exemple, dans un système de chiffrement à clé publique, la **clé de déchiffrement** est souvent interprétée comme une **trappe** : le chiffrement est impossible à inverser, sauf pour celui qui possède la **trappe**, qui est ici la clé de déchiffrement.
- **Clé de chiffrement, clé de déchiffrement** : un algorithme de chiffrement prend en entrée un message et une clé de chiffrement, et calcule un message chiffré, qui sera donc inintelligible. Un algorithme de déchiffrement prend en entrée un message chiffré et une **clé de déchiffrement** et calcule le message clair correspondant.
- **Hachage cryptographique, *hash*** : un algorithme qui prend en entrée des messages arbitrairement longs, et calcule un condensé (***hash***) très court (typiquement 32 octets). Cet algorithme a la propriété de rendre impossible, étant donné un ***hash***, de trouver le message d'entrée.
- ***Commitment* (mise en gage)** : publication ou partage d'un ***hash*** d'un document, sans révéler le document. Par exemple, le ***hash*** d'un document peut être enregistré dans une blockchain, à la place du document.
- **Ouverture d'un *commitment*** : étant donné un ***commitment*** d'un document ou d'une donnée, le document est révélé et confronté à son ***hash***. On vérifie que le document a bien été mis en gage. Par exemple, le commitment peut être enregistré dans un bloc d'une blockchain. Le possesseur du document peut alors utiliser le bloc d'enregistrement du ***hash*** comme date d'enregistrement et il ne peut pas changer le document un fois celui engagé. C'est une sorte de notarisation à l'aveugle. Ici, les preuves *zero-knowledge* peuvent toutefois prouver la validité d'énoncé sur un documents engagé par son ***hash*** sans le révéler.

- **Arbre de Merkle** : une structure de données agrégative permettant de mettre en gage un lot d'un grand nombre de documents en ayant recours à des hachages successifs. Elle présente deux propriétés intéressantes. Le ***hash*** racine est celui de tous les documents et reste très court. Ensuite il est possible au gestionnaire de l'**arbre de Merkle** de prouver à faible coût qu'un document fait partie du lot, sans révéler les autres documents.

- **Oracle** : Une blockchain ne peut certifier que les données déjà enregistrées dans la blockchain et les opérations faites dessus. Une blockchain en elle-même ne garantit pas la qualité des données externes insérées. Par exemple, si des ***smart contracts*** dépendent d'une micro-météo locale, il faut faire confiance à l'opérateur qui saisira les données de météo pour les enregistrer dans la blockchain. On utilise le terme d'oracle pour désigner un tel opérateur et souligner la notion de confiance dans les données externes.

- ***SideChain*** : il s'agit d'une chaîne de bloc mise en place parallèlement à la chaîne principale pour des raisons de performance et confidentialité, avec des mécanismes de validation et de consensus qui peuvent être différents de ceux de la chaîne principale. Se pose alors le problème de la compatibilité et de la réconciliation des transactions exécutées dans les deux chaînes.

- ***ZK-Rollup*** : un lot de transaction est généré en dehors de la blockchain principale. Ce lot est enregistré dans un **arbre de Merkle**, dont seule la racine est enregistrée dans le chaîne principale. Cette racine est accompagnée d'une preuve *zero-knowledge* courte qui prouve que toutes les transactions du lot sont correctes.

- ***Proof-of-authority*** : c'est simplement une signature électronique avec la **clé privée** d'une entité bien identifiée, dans laquelle le système ou une partie du système met de la confiance et est capable de vérifier les signatures émises par cette entité. Cela permet la validation des transactions dans les blockchains privées ou permissionnées, mais c'est antagonique aux notions de blockchains publiques et de décentralisation.

- **Configuration de confiance (*Trusted setup*)** : Processus par lequel un **ZK-SNARK** est initialisé. Ce processus produit une ***chaîne structurée de référence***.

- **Chaîne structurée de référence** (***Structured reference string***), **déchet toxique** ***(toxic waste)*** : dans le cadre des **ZK-SNARK**s, qui permettent des preuves très courtes et rapides à vérifier, toute la complexité est repoussée dans une chaîne d'octets, qui encode les étapes algébriques devant être réalisées pour vérifier une preuve *zero-knowledge*. Cette chaîne est mise en place à l'initialisation du système, pour un **langage** donné. Cette chaîne, construite selon les principes de la cryptographie à clé publique, se trouve être trappée. Celui qui connaît la trappe peut alors fabriquer des preuves acceptées comme correctes d'énoncés faux d'où l'expression courante « déchet toxique » pour désigner la trappe.

- ***Smart contract*** : un programme informatique enregistré dans la blockchain. Ce programme est exécuté par les validateurs ou mineurs quand des transactions le déclenchent. Un ***smart contract*** permet typiquement de déployer une logique applicative qui lui est propre au-dessus de la blockchain qui le maintient, par exemple l'émission de jetons fongibles ou non, ou de certificats de biens numériques, etc. De plus, un ***smart contract*** permet des achats ou ventes entre les jetons et la monnaie native de la blockchain.

- **Environnements d'exécution sécurisés (*Trusted Environment Execution*)** : technologie matérielle implémentée au niveau des processeurs (SGX d'Intel, TrustZone d'ARM avec TrustZone, SEV d'AMD). Il s'agit d'une partie de la mémoire qui est en permanence chiffrée et qui ne peut être déchiffrée que dans l'enclave sécurisée, selon les garanties du constructeur. De plus, un calcul peut être exécuté sur les données de l'enclave et en sortir le résultat, avec la garantie que le calcul est correct. Cette garantie est attestée par une signature électronique publiquement vérifiable.

Documents pédagogiques

- **Une présentation simple et correcte** : *How to Explain Zero-Knowledge Protocols to Your Children.* Quisquater, Jean-Jacques ; Guillou, Louis C. ; Berson, Thomas A. CRYPTO `89. [online] http://www.cs.wisc.edu/~mkowalcz/628.pdf

- **Compilation d'un programme en un système de polynômes** : https://electriccoin.co/blog/snark-explain5/

- **Présentation pas à pas des mécanismes cryptographiques d'un SNARK** : https://electriccoin.co/blog/snark-explain/

- **Zokrates, un environnement de construction de SNARKs et des smart contracts associés** : https://zokrates.github.io/

- **Déroulement pas à pas d'un STARK sur un exemple :** https://starkware.co/developers-community/stark101-onlinecourse/

Références bibliographiques choisies

- **Invention du *zero-knowledge*** : *The knowledge complexity of interactive proof systems.* Goldwasser, S., Micali, S., Rackoff, C. (1989). *SIAM Journal on Computing, 18 (1)* : 186–208. [online] http://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Proof%20Systems/The_Knowledge_Complexity_Of_Interactive_Proof_Systems.pdf

- ***Seminal paper* sur les preuves à base de courbes elliptiques et de couplages** : *Short Pairing-Based Non-interactive Zero-Knowledge Arguments.* Jens Groth, ASIACRYPT 2010. [online] https://www.iacr.org/archive/asiacrypt2010/6477323/6477323.pdf

- **Introduction du terme SNARK** : *From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again.* Nir Bitansky and Ran Canetti and Alessandro Chiesa and Eran Tromer, ICTS'12. [online] https://eprint.iacr.org/2011/443

- **Premier SNARK efficace** : *Pinocchio : Nearly Practical Verifiable Computation.* Bryan Parno and Craig Gentry and Jon Howell and Mariana Raykova, IEEE Symposium on Security & Privacy 2013. [online] https://eprint.iacr.org/2013/279

- **Introduction des STARKs** : *Scalable, transparent, and post-quantum secure computational integrity.* Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, Michael Riabzev. March 6, 2018 [online] https://eprint.iacr.org/2018/046.pdf. Version simplifiée par les mêmes auteurs : *Scalable Zero Knowledge with no Trusted Setup*, CRYPTO 2019 [sur abonnement]

Bibliothèques informatiques bas niveau de zero-knowledge

- **libsnark** (Scipr-lab.org association d'académiques)
- **gnark** (Consensys)
- **starkware-libs** (Starkware)
- **Bellman** (utilisé dans zcash)

Réalisations industrielles

- **Zcash**
- **StarkEx** : rollup en production
- **Aztec.network** (zk.money, Rollup PLONK)
- **zkSync** (ZK Rollup en production)
- **Filecoin** : système de stockage pair-à-pair contre rémunération

Performances

- ZK-SNARKs for **Zcash** : des circuits à 130000 portes de manière routinière
- **Filecoin** demande un système de preuve pour un circuit à près d'un milliard de portes.
- **Starks** : 3000 transactions par secondes
  https://medium.com/starkware/the-great-reddit-bake-off-2020-c93196bad9ce
- Exemple hors blockchain : systèmes poussés à l'extrême par le **DARPA** (*Defense Advanced Research Projects Agency*). Il s'agit de prouver l'existence d'une faille logicielle dans un programme sans révéler la faille en question. Les preuves peuvent être gigantesques dans ce cas là.

Effort de standardisation

- Zkproof.org

Daniel Augot

Directeur de recherche Inria (Institut national de recherche en informatique et automatique), a obtenu une thèse en informatique en 1993 ainsi que de l'habilitation à diriger des recherches en 2007. Il encadre des doctorants et post-doctorants à l'École polytechnique. Avec Julien Prat, il est responsable de la chaire Blockchain and B2B platform, soutenue par CapGemini, NomadicLabs, et la Caisse des dépôts.

Sarah Bordage

Doctorante depuis 2018 à l'École polytechnique sous la supervision de Daniel Augot. Elle s'intéresse aux constructions de preuves zero-knowledge pour le calcul vérifiable

Youssef El Housni

Ingénieur à ConsenSys, membre de l'équipe «gnark» et doctorant à l'École polytechnique sous la supervision de Daniel Augot et François Morain. Il s'intéresse aux preuves zkSNARKs et aux primitives cryptographique sous-jacentes, en théorie algorithmique des nombres.

Xavier Lavayssière

Chercheur indépendant sur les aspects technologiques et réglementaires des actifs numériques. Il a fondé l'ECAN, https://ecan.fr/. centre de formation sur les technologies blockchains et enseigne à Paris I Panthéon Sorbonne.

Louis Bertucci

Chercheur à l'Institut Louis Bachelier. Il est titulaire d'un doctorat en finance délivré par l'université Paris-Dauphine. Il travaille sur les blockchains depuis 2017.

Noémie Dié

Doctorante CIFRE au département Économie de Télécom Paris (Institut Polytechnique Paris) en partenariat avec Bpifrance Le Lab.

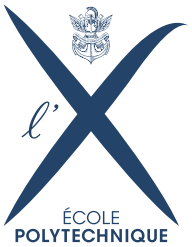
Gilles Fedak

Titulaire d'un doctorat en informatique de l'université Paris Sud. Après un post-doctorat à l'université de Californie à San Diego, il devient chercheur INRIA à l'ENS Lyon. Il est titulaire du prix chinois PIFI. Gilles Fedak est PDG et fondateur de iExec, plateforme de calcul dans le cloud basée sur les blockchains.

Anthony Simonet-Boulogne

Doctorat en Informatique de l'École Normale Supérieure de Lyon en 2015. Il a travaillé sur des problèmes liés au calcul distribué au cloud computing et à la blockhain à Inria, Rutgers University et est Responsable des Projets Scientifiques à iExec depuis 2019.

BLOCKCHAIN& PLATFORM CHAIR



INSTITUT POLYTECHNIQUE DE PARIS

Capgemini

nomadic labs

Directeur de la publication : **Daniel Augot**

Avec l'aimable participation de :

Sarah Bordage — **Gilles Fedak**
Youssef El Housni — **Anthony Simonet-Boulogne**

Et l'aide à la relecture de :

Louis Bertucci — **Xavier Lavayssière**

Collaboration éditoriale : **Noémie Dié**

Graphisme et maquettage : Centre Polymédia de l'école polytechnique
Publication Outlook de la Chaire Blockchain&Platform Polytechnique