Guide d'étude de la semaine 1

Pour les personnes intéressées par les groupes d'étude, nous fournissons quelques ressources que vous pouvez parcourir en groupe. Ce ne sont que des suggestions pour vous aider à démarrer et vous n'êtes pas obligé de les suivre.

Attention: Certains de ces exercices peuvent prendre plus d'une séance! Ne vous sentez pas obligé de les terminer en une seule séance. Il peut être utile de commencer à travailler sur eux séparément, puis de vous réunir lorsque vous êtes tous à un point d'arrêt similaire. Vous pouvez partager ce que vous avez couvert, certains défis que vous avez rencontrés et aider tous ceux qui travaillent encore sur les exercices.

Nous passerons également par ces exemples/tutoriels dans les sessions hebdomadaires. Nous encourageons également les étudiants à partager leur travail là-bas. Soyez gentil avec les membres de votre groupe !

# 1. Début : Atelier Ethereum 101

[Voir un enregistrement du premier exercice ici](https://www.loom.com/share/a02b5dec33d4468e9aa862540561697e)

Le premier consiste en quelques exercices de base montrant à quoi ressemble une clé privée et comment elle fonctionne. Nous allons parcourir ces diapositives, mais il y a des exercices situés au milieu de la présentation qui pourraient vous intéresser, surtout en groupe.

Faites défiler les diapositives à l'aide des touches fléchées jusqu'à ce que vous voyiez "Exercice 1" sur la diapositive, environ vingt diapositives. En bas de la colonne de gauche, cliquez sur le bouton rouge clair "Générer une adresse Ethereum" pour voir les sections "Comptes Ethereum" et "Clé privée" se remplir.

**Veuillez noter : les clés privées générées ici NE SONT PAS sûres et ne doivent pas être utilisées à des fins autres que pédagogiques!**

Cliquez sur le champ "Clé privée" pour faire glisser et déposer la clé privée dans l'emplacement vide "Clé privée". Écrivez un message dans le champ "Message" et cliquez sur "Signer le message"

Dans la colonne de droite, faites glisser le "Texte du message" et la "Signature du message" de la colonne de droite et l'Adresse Ethereum de la colonne de gauche. Remarque : N'utilisez pas la clé privée !

Vérifiez la validité du message en cliquant sur "Vérifier la signature". Essayer de modifier l'une des entrées (texte, signature ou compte) pour voir la vérification échoue. (Veuillez noter que tout se passe localement sur votre machine en utilisant une bibliothèque cryptographique, sans communiquer à un serveur ou un tiers.)

Si vous cliquez sur "Télécharger les données de message", cela générera un fichier .txt que vous pourrez envoyer à d'autres personnes de votre groupe pour tester également leurs messages. Attention : vous devez copier et coller parfaitement le contenu et vous devez ouvrir leur fichier d'une manière qui ne changera pas l'encodage .txt. Si vous l'ouvrez à l'aide de Microsoft Word, par exemple, ou de Google Docs, cela altérera le message et le rendra invalide. C'est un excellent exemple de la façon dont l'identité décentralisée est très puissante mais aussi très fragile : même le plus petit des problèmes casse le système.

Si cela vous intéresse, il y a un autre exercice après cet exercice "Encode / Decode". C'est similaire aux [exercices Anders Blockchain](https://andersbrownworth.com/blockchain/hash), qui montrent comment la modification d'une information dans une chaîne de hachage rend tous les blocs suivants invalides.

Il existe également un éditeur Solidity jouet qui vous permet de déployer du code en un clic si MetaMask est installé avec suffisamment d'éther pour le réseau sur lequel vous vous trouvez!

Nous les passerons en revue lors de la session en direct de mercredi

# 2. Technique : Exercices de hachage Bitcoin

Tout d'abord, [clonez ce référentiel d'exercices du livre blanc Bitcoin](https://github.com/cooganb/bitcoin-whitepaper-exercises)

Ensuite, [suivez les étapes de ce README pour faire le premier exercice, le hachage.](https://github.com/cooganb/bitcoin-whitepaper-exercises/blob/master/hashing/README.md)

Veuillez noter que peu importe l'algorithme de hachage que vous utilisez, tant qu'il est cohérent.

[Un tutoriel plus avancé que vous pouvez suivre est celui-ci en Python](http://karpathy.github.io/2021/06/21/blockchain/). Il s'agit d'une implémentation à partir de zéro de Bitcoin sans aucune dépendance (🤯 🤯 🤯). Pour ceux d'entre vous qui suivent l'apprentissage automatique, c'est d'Andrej Karpathy, qui a écrit des trucs formidables sur les réseaux de neurones). Vous pouvez suivre le Python au fur et à mesure qu'il le construit ou vous pouvez exécuter le bloc-notes Jupyter ici. Ceci est destiné aux utilisateurs Python intermédiaires à avancés. Si vous n'êtes pas familier avec Python, ce n'est pas l'endroit pour apprendre.

# 3. Technique : Atelier sur la création d'un réseau P2P

Jusqu'à présent, ce guide d'étude contient du matériel d'introduction et du matériel sur les primitives cryptographiques. Ce qui nous manque, c'est un didacticiel sur le système distribué / le consensus distribué !

[Voici deux bons tutoriels de WebTorrent. Le premier didacticiel (Atelier P2P)](https://mafintosh.github.io/p2p-workshop/build/01.html) décrit le processus de création d'un serveur jouet sur votre ordinateur local, crée un service de chat, puis développe les processus de mise en réseau pour découvrir les pairs de votre réseau.

[Le deuxième didacticiel (Atelier de partage de fichiers P2P)](https://mafintosh.github.io/p2p-file-sharing-workshop/build/01.html) présente les spécificités techniques des fichiers en streaming, la découverte de services, le hachage de fichiers, la segmentation de fichiers (une fonctionnalité du protocole torrent), puis le partage de fichiers avec plusieurs participants.

Ces tutoriels sont dans Node.js et sont bons car ils décomposent ces projets compliqués en étapes discrètes. Cependant, il est assez avancé et, si vous rencontrez des problèmes, essayez de rejoindre un groupe qui y travaille.