TP projet

Sergey Kirgizov

Polytech Dijon

Tchaî — Chaîne des transactions

Objectif : concevoir un système de transactions électroniques avec une intégrité garantie, accessible par le protocole HTTP.

Mode du travail préférée : binômes.

À la fin de ce document, vous trouverez des liens vers la documentation. Cela vous aidera à effectuer les exercices suivants. Après avoir terminé tous les exercices, vous obtiendrez un système de gestion des transactions ressemblant à celui de la blockchain et Git.

☞ **EXERCICE 1.** Premiers pas de Git [1] :

1. Créer un dépôt Git dans un répertoire existant local (`git init`).
2. Rédiger un fichier `README.md` contenant la description du projet en utilisant un langage de balisage Markdown [2]. Ajouter ce fichier dans le dépôt (`git add`).
3. Faire le premier `git commit`.
4. Consulter le statut du dépôt (`git status`) et la liste des commits (`git log`).

☞ **EXERCICE 2.** Git, un outil de travail collaboratif.

1. Créer un dépôt vide et privé sur BitBucket, GitLab ou GitHub.
   Nommer ce dépôt comme `TCHAI-PRENOM`$_1$`-NOM`$_1$`-PRENOM`$_2$`-NOM`$_2$ en utilisant vos noms et prénoms.
2. Djouter l'URL du dépôt distant (sur BitBucket, GitHub ou GitLab) avec lequel votre dépôt local sera synchronisé (`git remote add origin URL`).
3. Synchroniser (`git push`) votre dépôt local avec le dépôt que vous venez de créer sur le serveur.
4. Ajouter votre binôme en tant que collaborateurs dans le projet sur le serveur.
5. Sur l'ordinateur du binôme :
   (a) Cloner le dépôt sur l'ordinateur du binôme.
   (b) Modifier le fichier `README.md` en ajoutant une nouvelle section **"Auteurs"** contenant vos noms et les adresses mails.
   (c) Enregistrer des modifications dans le dépôt local (`git add`, puis `git commit`)
   (d) Pousser son travail sur le dépôt distant (`git push`)
6. Sur le premier ordinateur :
   (a) Récupérer les modifications de votre binôme, en tirant depuis le dépôt distant (`git pull`)
   (b) Vérifier que tout va bien en consultant le statut du dépôt (`git status`) et la liste des commits (`git log`).

Par défaut, Git vous permet de partager un dépôt privé avec au plus trois personnes, donc le travaille en trinôme peut être problématique. Au cours du projet, vous devez synchroniser régulièrement les dépôts, sans oublier de mettre à jour la description du projet et la documentation dans le fichier `README.md`. Chaque version significative de votre code doit être étiquetée (`git tag`).

INFO : **Pour chaque exercice terminé vous devez avoir au moins un commit correspondant dans le dépôt Git. Le nombre de commits ne doit pas être inférieur au nombre d'exercices résolus.**

ASTUCE : **Vous pouvez également faire l'inverse, c'est-à-dire créer d'abord un dépôt sur le serveur, puis le cloner sur votre ordinateur.**

ASTUCE : **Chaque serveur Git (Bitbucket, GitLab, GitHub, etc) dispose de ses propres mécanismes d'authentification. Vous devrez vous renseigner à ce sujet sur leurs sites web respectifs.**

Tchaî v1

Nous définissons une *transaction* comme étant un tuplet (P_1, P_2, t, a), où a est égal à la somme d'argent transférée de la personne P_1 à la personne P_2 au moment t.

EXERCICE 3. En utilisant Flask [3], réaliser une première version du système "Tchaî". Voici une liste des actions qui doivent être mises à la disposition via un API HTTP (voir TD-1) par votre système "Tchaî" :

(A1) Enregistrer une transaction.

(A2) Afficher une liste de toutes les transactions dans l'ordre chronologique.

(A3) Afficher une liste des transactions dans l'ordre chronologique liées à une personne donnée.

(A4) Afficher le solde du compte de la personne donnée.

INFO : **Vous êtes libre de choisir un autre langage de programmation au lieu du Python, un autre framework web au lieu de Flask, un système préféré de stockage de données (SQLite [8], un fichier texte, ...), etc. Vos choix doivent être documentés et justifié dans le fichier** `README.md`.

EXERCICE 4. Attaquer le système en modifiant directement le fichier de données, en changeant le montant d'une transaction.

INFO : **Toutes les attaques et tous les tests doivent être décrits en détail dans une section adéquate du fichier** `README.md`. **Les scripts permettant leur exécution doivent être disponibles dans le répertoire** `tests` .

Tchaî v2

Nous ajoutons maintenant le hash d'une *transaction* dans son tuplet : (P_1, P_2, t, a, h), où a est égal à la somme d'argent transférée de la personne P_1 à la personne P_2 au moment t et h correspond au hash du tuple (P_1, P_2, t, a).

INFO : **Votre choix de la fonction de hachage doit également être documenté dans le fichier** `README.md`.

EXERCICE 5. Modifier votre programme afin d'intégrer la nouvelle structure des transactions

EXERCICE 6. Ajouter l'action suivante disponible en API HTTP :

(A5) Vérifier l'intégrité des données en recalculant les hashs à partir des données et en les comparant avec les hashs stockés précédemment.

EXERCICE 7. Vérifiez que l'attaque précédente ne fonctionne plus.

EXERCICE 8. Attaquer le système en modifiant directement le fichier de données, en supprimant une transaction. La possibilité de supprimer une transaction peut être très dangereuse, la suppression peut entraîner la double dépense [9]

Tchaî v3

☞ **EXERCICE 9.** Modifier la méthode de calcul de hash. Maintenant la valeur du hash h_{i+1} va dépendre non seulement de la transaction en cours, mais également de la valeur du hash h_i de la transaction précédente.

☞ **EXERCICE 10.** Vérifiez que les attaques précédentes ne fonctionnent plus.

☞ **EXERCICE 11.** Attaquer le système en modifiant directement le fichier de données, en ajoutant, par exemple, une transaction provenant d'une autre personne vers le compte de l'attaquant.

Tchaî cryptographique (v4)

☞ **EXERCICE 12.** Lire le message [4], le papier original de Satoshi Nakamoto [5] et la discussion ultérieure sur la liste de diffusion 'The Cryptography and Cryptography Policy Mailing List".

☞ **EXERCICE 13.** Utiliser la cryptographie asymétrique afin d'assurer l'authenticité de l'expéditeur.

Références

[1] ***Pro Git book***, Scott Chacon et Ben Straub, 2014.
`https://git-scm.com/book/fr/v2`

[2] ***Markdown***, un langage de balisage léger créé par John Gruber et Aaron Swartz en 2004.
`https://daringfireball.net/projects/markdown/`

[3] ***Flask***, un framework léger de développement web en Python, créé initialement par Armin Ronacher comme étant un poisson d'avril, le 1er avril 2010.
`https://palletsprojects.com/p/flask/`

[4] ***Bitcoin P2P e-cash paper***, un message de Satoshi Nakamoto posté sur "The Cryptography and Cryptography Policy Mailing List", 2008.
Archive 1 : `https://www.metzdowd.com/pipermail/cryptography/2008-October/014810.html`
Archive 2 : `https://www.mail-archive.com/cryptography@metzdowd.com/msg09959.html`

[5] ***Bitcoin : A Peer-to-Peer Electronic Cash System***, Satoshi Nakamoto, 2008
`https://web.archive.org/web/20140320135003/https://bitcoin.org/bitcoin.pdf`

[6] ***Python IDE "Thonny"***, créé par Aivar Annamaa
`https://thonny.org`

[7] ***Installation de Git***
`https://git-scm.com/download`

[8] ***SQLite***, une bibliothèque écrite en C, proposant un moteur de base de données relationnelle accessible par le langage SQL. créée par D. Richard Hipp en 2010
`https://sqlite.org`

[9] ***Double dépense***, (en anglais, double-spending), un acte frauduleux dans lequel le même jeton numérique est dépensé plus d'une fois
`https://fr.wikipedia.org/wiki/Double_d%C3%A9pense`