Rapport - Parcours IngénieurDocteur

Sandrine JUILLARD

Mai 2020

Table des matières

Introduction

L'objectif de cette option "Parcours-ingénieur docteur", a été d'acquérir, à travers la lecture d'articles de recherche, une culture scientifique autour des enclaves Software Guard Extensions (SGX) et de la block-chain. Ce rapport s'organise donc d'une part, par la restitution du contenu des articles que j'ai lus. D'autre part d'une partie qui résumera les nouvelles notions que j'ai acquis au cours de ces derniers mois.

1 Software Guard Extensions (SGX)

Software Guard Extension est un kit de développement (sdk) permettant de déployer une enclave software. Une enclave est un environnement d'exécution isolé de la platforme sur lequel on l'a déploie. SGX permet cette isolation en cryptant la totalité des données qui se trouvent à l'interieur, avec une protection au niveau du processeur : les clefs de decriptage sont stokées dans la CPU, et les datas ne peuvent être déchiffrées qu'à l'interieur du processeur. (cf [1]). Ansi, peu importe le niveau de privilège que l'on peux obtenir, seul l'enclave a accès à ses propres données. On utilise une enclave, lorsque l'on veut réaliser des opérations sensibles, sur une platforme qui n'est pas de confiance, c'est à dire une platforme pouvant être curieuse ou infectée par un logiciel malveillant.

1.1 Fonctionnement d'une enclave SGX

Une application utilisant sgx s'organise de la manière suivante. Une partie de l'application est executée hors de l'enclave. C'est la partie "Untrusted" de l'application. L'enclave est crée par l'application. Quand elle aura une opération sensible à effectuer (une opération dont on veut être sur qu'elle ne soit ni observée, ni corrompue ; une opération qui traite des données confidentielles par exemple), elle fera appel à l'enclave qui executera le code qu'elle contient.

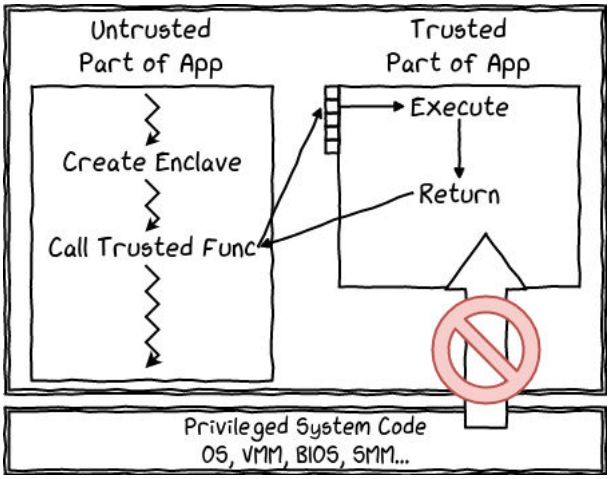

FIGURE 1 – Schema d'utilisation d'une enclave SGX (Source [1])

L'enclave et l'application communiquent par l'intermédiaire de messages : ecall (Enclave Calls) et ocall (Outside Calls). Les ecalls permettent d'appeler une fonction dans l'enclave. Les ocalls permettent à l'enclave de demander l'exécution d'une opération hors de l'enclave. Une des spécificité des enclaves SGX est qu'elles ne peuvent pas réaliser d'appels système. Elle peut néamoins le demander à travers un message Ocall.

1.2 Exemples d'utilisation de SGX

SGX garantit confidentialité, intégrité et attestation aux opérations qui s'exécutent à l'intérieur. Elle permet d'exécuter sans risque d'être corrompu ou observé, des opérations sensibles, même dans un environnement qui peut-être corrompu. Les deux articles ci-dessous mettent à profit cette propriété.

1.2.1 Fideluis

Contexte et problème adressé

Fidelius [3] est une solution Hardware, capable de protéger les utilisateurs du vole de leurs informations des attaques de type "KeyLogger" (cf. 3.3). Par exemple, elle garantie à l'utilisateur, quand il rentre ses coordonnées bancaires sur un site de vente en ligne, que même si son PC est infecté, les informations en provenance de ses périphériques (clavier, écran), ne pourrons pas être collectés.

Rôle de SGX

Dans le "Threat Model" proposé, on considérera que l'OS et le navigateur peuvent potentiellement être infectés. Deux enclave SGX vont communiquer entre elles gâce à des clefs symétriques partagées plus tôt. Ainsi, toutes les informations en provenance des périphériques seront invisibles pour l'OS et le navigateur.(fig.2).

Contribution scientifique

Fidelius fonctionne grâce à deux enclaves : une déployée sur l'OS est chargée de sécuriser certaines des zones d'une page web, identifiées par le logiciel Fidelius. Ces zones sont repérées par les balises d'une page HTML de type `<form>`, `<script>` ou `<input>` ayant un argument signature ainsi que l'argument `"Secure= 'True'"`. L'autre enclave, déployée sur une rapberry pi sert d'intermediaire entre un périphérique (clavier, souris, écran) et le système d'exploitation. Ces deux enclaves communiquent par un canal crypté.

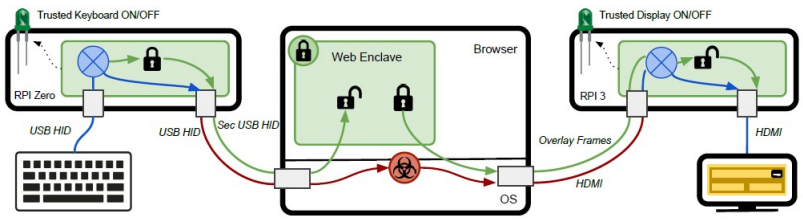

FIGURE 2 – Protocole de Fideius (source [3])

L'enclave web suit le protocole suivant :

Etats

Initi	:	Vérification des certificats d'authtentification de la page web concernée.
Auth	:	Transmission des balises HTML marquées par l'argument secure à l'enclave Vérification des signatures des balises
Ready	:	L'enclave web est opérationnelle. Les deux enclaves commencent à communiquer Les actions des périphériques sont invisibles pour l'OS et le navigateur
End	:	L'utilisateur quitte la page. Fin de la communication entre les enclaves Destruction de l'enclave web
Fail	:	Survient à l'envoi d'un eCall qui ne suit pas le protocole présenté Destruction de l'enclave web

1.2.2 X-Search

Contexte et problème adressé

X-Search[7] est un proxy(cf 3.3) capable de protéger les utilisateurs contre les collecteurs de données, qui stockent et trient les requêtes (recherche web sur Google par exemple) afin d'en déduire des informations personnelles (orientation politique, sexuelle, religion ...). Cette problématique devient d'autant plus importante que de récentes lois aux Etats-Unis ont autorisé les fournisseurs d'accès Internet à collecter et vendre les données de leurs clients sans leur consentement. X-search est un proxy qui mêle les protocoles de recherche anonyme (Unlikability) et fausse requête (Indistinguishability).(cf. 3.1).

Rôle de SGX dans la contribution

Le proxy X-Search doit pouvoir être déployé sur un cloud public qui peut être malveillant. Grâce à SGX, il est possible de déployer X-Search sur ce genre de plateforme sans risquer de corrompre le protocole de sécurité.

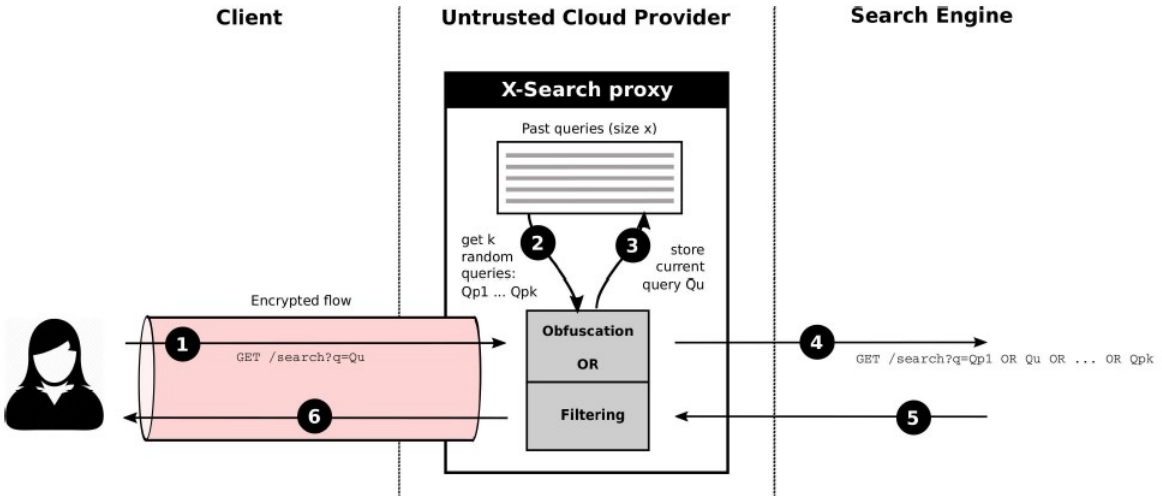

FIGURE 3 – Architecture de la solution proposée (source [7])

Contribution scientifique
La figure 3 détaille le protocole suivit par X-Search.

1. Le client envoie sa requête. Elle est transmise au proxy par un canal de communication cryptée avec le proxy X-search grâce à un broker (processus fesant office d'intermédiaire qui s'exécutera en parallèle du navigateur).
2. Le proxy va être chargé de mélanger cette requête avec les requêtes qu'il possède dans sa "bibliothèque de requêtes". Celle-ci est composée des précédentes requêtes traitées par X-Search.
3. Le proxy enregistrera également la requête (sans conserver d'information sur la source de la requête), pour enrichir sa bibliothèque
4. il enverra la "suite de requêtes", liées par des "OR", sous forme d'une seule requête, au moteur de recherche pour qu'il la traite
5. La réponse de cette requête sera reçue par le proxy, puis filtrée pour en extraire les résultats qui correspond bien à la requête originale du client
6. Puis ces réponse lui seront remis, toujours au travers d'un canal de communication cryté.

L'algorithme qui permet de filtrer les requêtes trie les réponses en considérant que celles-ci seront semblables à la requête ; Il enverra les réponses ayant le plus de mots communs avec la requête. Toutes les opérations citées plutôt qui sont effectuees par le proxy, se font dans l'enclave sur un cloud qui n'est pas de confiance.

1.3 Axes d'améliorations de SGX : HotCalls

Il a été observé que l'utilisation des enclaves SGX pouvaient être optimisées. En effet, les appels eCalls effectuent un changement de contexte (cf. 3.3) pour passer de l'application à l'enclave, ce qui est coûteux en temps. L'article HotCalls s'interrese à un protocole pouvant évité ces changement de contexte, afin d'optimiser les performance de l'enclave.

Contribution scientifique
L'enclave et l'application communiquent par de la mémoire partagée. Lorsque l'un des partis veux transmettre des données, il suivra le protocole décrit figure 4. A noter que les protocoles sont interchangables. Le schéma montre le protocole dans le cas où c'est l'enclave qui veut transmettre des données à l'application.

La spécificité de ce protocole réside dans l'utilisation d'un verrou de type SpinLock. Ce type de verrou est basé sur l'attente active. C'est-à-dire qu'on va vérifier de manière répétée si la condition est vérifiée. Dans le contexte d'un système multiprocessur, l'attente active va réserver un processeur. N'étant plus soumise à l'ordonancement du système, on evite ainsi d'effectuer un changement de contexte.

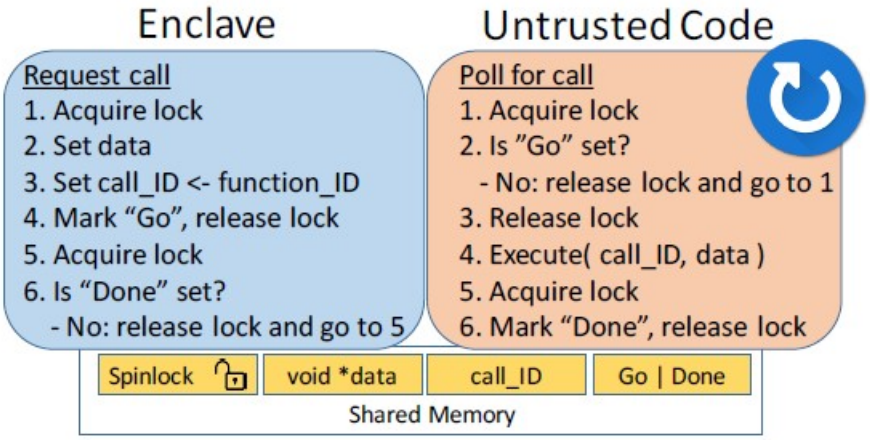

FIGURE 4 – Architecture de HotCalls (source [6])

Cette méthode est couteuse en ressource, mais son utilisation est justifiée par le fait que le changement de contexte est plus long que le temps d'attente moyen en attente active. Avec cette approche, le nombre de cycles pour effectuer un appel ECall par exemple est significativement réduit. On peut observer que HotCalls permet de multiplier le temps de traitement des opérations de 13 à 27 fois par rapport à SGX seul.

2 La block-Chain

2.1 Origine de la block chaine : Horodatage de document

Contexte et problème adressé

Le protocole de la block-chaine est inspiré des protocoles d'Hotodatage de document. L'article "How To time-stamp a document" [8] décrit des protocoles garantissant confidentialité et fiabilité pour dater un document, tout en minimisant l'espace de stockage et la bande passante (quantité de données échangée).

Contribution scientifique

Fonction de Hashage et Signature : Pour résoudre ces problèmes, on va introduire deux outils. D'abord, la fonction de hachage 3.3 qui garantit la confidentialité du client et permet de réduire la bande passante, en ne transmettrant à la TSS qu'une version hashée de son document. Ensuite, pour régler les problèmes de stockage, nous allons utiliser une signature, pour que la TSS n'aie plus à stocker les documents. En ajoutant sa signature, la TSS certifie la provenance du document qui pourra être conservé par le client.

Certificat 'lié' : Nous allons ajouter au modèle précédent le concept de certificat. Lorsque le client va faire une demande d'horodatage, la TSS va générer un certificat à partir de ceux qu'il a créés avant. On appelera "Linking information" la verison hashé de la somme des information suivant : document, la date, l'id du client (fig.5). Un certificat

sera généré à partir des infromation du document à certifié plus les "Linking information" du document précépend. Dans ce modèle la validité des certificats est liée aux certificats précédents.

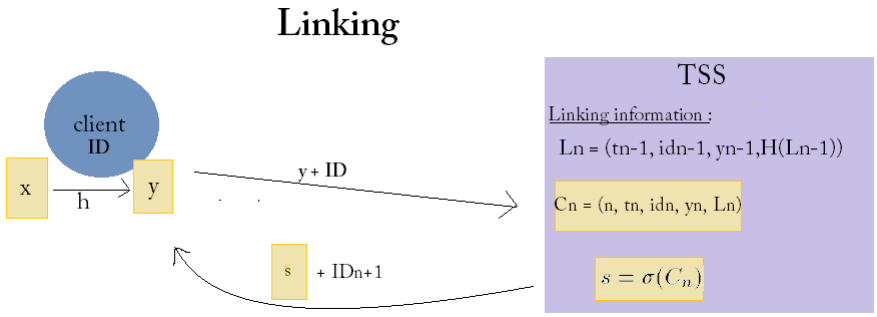

FIGURE 5 – Protocole "Lié"

Certificat 'Distribué' : Pour compléter le modèle précédent, nous allons multiplier les sources de certificats, afin d'être protégé d'une potentielle TSS corrompu. Le client piochera au hasard k identifiants de TSS (grâce à la fonction G pseduo-aléatoire, qui prendra le document hashé comme "seed").

Parralèle avec la Block-Chaine

Ce dernier protocole de certifications est très similaire à celui de la block-chaine : Un block est chainé par le protocole de Linkage. La signature d'un block est le hash du block précédent. Ensuite il est certifié, selon le protocole "Distribué" par tous les acteurs de la block-chaine, qu'on appelle des nodes.

2.2 Fonctionnement de la block-chaine

Maintenant que nous avons vu l'origine de la block-chaine, nous allons décrire ce protocole en détail. Celui-ci est expliqué dans l'article original bitcoin[5]. La block-chaine, comme son nom l'indique est constituée d'une chaine de blocks. Un block contient des transactions, un identifiant unique appelé "Noune" ainsi qu'une signature qui n'est autre que le hash 3.3 du block précédent.

Protocole

1. De nouvelles transactions sont effectuées et broadcastées à toutes les nodes.
2. Toutes les nodes collectent les transactions dans un block.
3. Lorsqu'une node termine un block (toutes les 10 minutes), elle cherche sa "Proof of work".
4. Quand la "Proof of work" est trouvée, elle la transmet à toutes les autres nodes.
5. Les nodes reçoivent le block et la "Proof of work", elles vérifient que les transactions ainsi que la proof of work sont valides.

6. Si le block est valide, les autres nodes montrent leurs accords par la création du block suivant, en utilisant le hash du block précédent comme signature du nouveau block.
7. Les nodes considéreront que la chaine la plus longue sera la chaine valide. Ainsi, si on voulait falsifier la chaine, il faudrait dépenser plus de CPU que les nodes honnêtes pour les convaincre que leur chaine est la bonne. (51% Attaque)

Spécificités

Proof of work : Ce mécanisme garanti la sécurité de la block chaine. Pour trouver la proof of work, une node doit trouver le hash qui correspond au hash du block précédant, succédé par un certain nombre de 0. Cette opération est coûteuse en CPU, car pour trouver le hash, il faut tester toutes les possibilités jusqu'à trouver la bonne. Faire l'opération inverse, en revanche est très simple (cf 3.3)

Arbre de Merkel : Pour ne pas avoir à stocker en entier l'historique des blocks pour vérifier l'intégrité de la chaine, on utilise un arbre de Merkel, appelé aussi arbre de hashage. Le principe étant de concaténer deux à deux les hashs des blocks.

Combiner et séparer les transactions : Afin encore une fois d'économiser de l'espace mémoire, les transactions sont combinées et séparées : lorsqu'une transaction est effectuée, le 'dépensier' va envoyer la totalité de son porte-monnaie. De cette manière, il n'aura pas besoin de se souvenir de la somme qu'il possède. L'argent dépensé en trop lui sera remis sous forme de change.

2.3 Axes d'améliorations de la Block-chaine : Algorand

La technologie bitcoin, dans son état actuel n'est adapté qu'à des transactions de très grosses sommes d'argent, et des dépenses occationnelles. Cela, en raison de son fort coût CPU et du temps très long pour la confirmation d'une transaction ($\sim 1h$). Des recherches sont menées, dans l'optique de démoratiser la block-chain, pour qu'elle soit adaptée à un usage commun, et pour un très grand nombre d'utilisateurs. On peut imaginer par exemple, un utilisateur de la block-chain, qui va payer son pain avec sa crypto-monaie.

Résumé de la contribution
Algorand propose une alternative au protocole Bitcoin, tout en conservant le principe de la block chaine. Le protocole d'Algorand, s'inspire du protocole de l'accord byzantin (cf. 3.3), afin d'obtenir un concensus à la fois fiable, rapide, et beacoup moins couteux en CPU que le protocole de consensus de bitcoin (qui nécessite le calcul de la Proof-of-Work, tandis que Algorand fonctionne uniquement par l'échange de messages). Algorand permet alors de réduire sigificativement le temps nécessaire à la validation d'une transaction, passant environ d'une heure à 1 minute. La validation d'un block étant 125X plus rapide que pour bitcoin.

Contribution scientifique
Le protocole d'Algorand est le suivant : Les transactions circulent entre les nodes selon le protocole gossip (cf.3.3). Les transactions qui n'ont pas été certifiées sont stockées. On les appelle pending Transaction. Un algorithme séléctionne pseudo-aléatoirement des leaders parmis l'ensemble des nodes.Ils sont élus avec une plus forte probabilité lorsqu'il possède plus de cryptomonaie. Ces leaders vont certifier ou non une pending transaction. Deux issues sont possible : soit une majoritée suffisante de leader valident la transaction, alors elle est ajouté à la block-chaine. Soit les leaders ne parvienne pas à ce mettre d'accord et la block chain va se séparer en deux branches. Les transactions qui suiveront permettront de valider ou non le block qui fait débat. Etant inspiré du protocole d'accord byzantin, ce protocole garantit, selon les même considération mathématique, que la block-chaine ne pourra pas être corrompue si au moins 2/3 des nodes sont honnêtes.

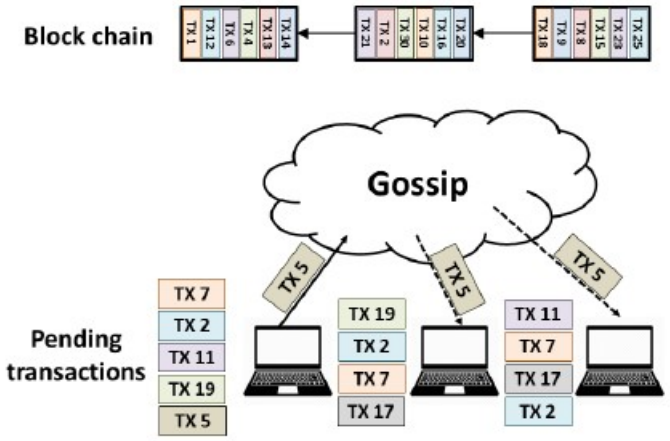

FIGURE 6 – Schéma illustrant le protocol d'algorand (source [9])

3 Culture scientifique Annexe

3.1 Protocole protégant contre la collecte des données

La recherche anonyme (Unlikability) : l'utilisateur ne peut plus être associé aux recherches qu'il effectue, il est anonyme. Il utilise le protocole "Onion Routing" (ex : RAC, "Dissent protocol"). Il est vulnérable aux nodes malicieuses et aux attaques de ré-identification (cf. 3.2)

L'envoi de fausses requêtes (Indistinguishability) : envoi de "dummy request" pour noyer la vraie requête de l'utilisateur.

Les moteurs de recherche alternatifs : ces moteurs utilisent des protocoles très lourds afin de garantir l'anonymat à leurs utilisateurs. Ils ne sont pas adaptés au traitement de grande quantité de données, ce qui les rendent inutilisables pour une usage standardisée. Par exemple, on peut citer le protocole d'encryption Homomorphique, qui est efficace

mais inutilisable à cause de sa grande complexité.

3.2 Attaque de ré-identification : exemple de SimAttaque

Une attaque de ré-identification est une attaque permettant de relier un utilisateur aux requêtes qu'il a effectuées, quand celles-ci ont été masquées par une des techniques citées précédemment (Unlikability, Indistinguishability). SimAttaque [2] est une attaque de type ré-identification qui utilise un algorithme qui calcule un coefficient de similarité, entre une requête et un historique de requêtes.

3.3 Vocabulaire scientifique

Clef de chiffrement : les clefs de chiffrement servent, comme leur nom l'indique à chiffrer/déchiffrer de l'information. On s'en sert également pour les signatures digtales, ou les codes d'authentification de messages. Il existe deux grandes catégories de clefs : les clefs de chiffrement symétriques, où la même clef sert à la fois à chiffrer et déchiffrer ; et les clefs de chiffrement asymétriques, où la clef dite publique sert au déchiffrement, et clef privée au chiffrement.

Comportement byzantin : système ne respectant pas ses propre spécifications et donnant des résultats non conformes. Ce comportement peut survenir naturellement pour des raisons physique/matériel ou âtre provoqué volontairement dans le but de faire échouer le système.

Accord byzantin : protocole de consensus, où l'on sélectionne plusieurs "leaders" parmi le groupe, pour prendre ensemble la décision pour le groupe. Ce protocole est fiable si au moins 2/3 des participants sont honnêtes.

Proxy : composant logiciel qui joue le rôle d'intermédiaire en se plaçant entre deux hôtes pour faciliter ou surveiller leurs échanges.

Software Development Kit (SDK) : ensemble d'outils logiciels destinés aux développeurs.

Fonction de Hashage : une fonction de hachage va transformer un bit-stream (ou n'importe quel type de document) en un autre bit-stream, de taille fixée ou non. C'est une fonction à sens unique, ce qui veut dire que le calcul de la fonction de hachage est facile et rapide tandis que le calcul de sa fonction inverse est pratiquement impossible.

KeyLogger : outil permettant d'enregistrer un évènement produit sur un ordinateur, en particulier l'action des touches du clavier (ou encore la capture d'écran, ou enregistrer les applications actives).

Protocole de "bavarage" (Gossip protocol) : protocole de communication, où toutes

les nodes d'un groupe sélectionnent aléatoirement un certain nombre d'autres partenaires à qui elles transmettrons l'information (en P2P). Ce protocole garantit avec une probabilité très proche de 1, que tous les participants du groupe recevront bien l'information.

Changement de contexte : dans le cadre d'un processeur qui doit ordonnancer plusieurs processus qu'il exécute pseudo-simultanément, un changement de contexte est l'opération qui consiste à sauvegarder l'état d'un processus pour en charger un autre.

Dans ce rapport sont présentées deux technologies distinctes, la block-chain et les enclaves SGX. Il est cependant possible de créer un lien entre les deux : utiliser les enclaves SGX, au service de la block-chain (ex : [4]).

Références

[1] Alexandre Adamski. Overview of Intel SGX. *Quarkslab's blog*, 2018.

[2] Albin Petit & co. SimAttack: private web search under fire. *Journal of Internet Services and Applications*, 2016.

[3] CISPA Helmholtz Center for Information Security. Fidelius : Protecting user secrets from compromised browsers. 2019.

[4] Moritz Schneider ISinisa Matetic, Karl Wüst and Ghassan Karame NEC Labs ; Srdjan Capkun ETH Zurich Kari Kostiainen, ETH Zurich. Bite : Bitcoin lightweight client privacy using trusted execution. *28th USENIX Security Symposium*, 2019.

[5] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. 2008.

[6] Todd Austin Ofir Weisse, Valeria Bertacco. Regaining lost cycles with hotcalls : A fast interface for sgx secure enclaves. 2017.

[7] P.Felber M.Pasin R.Pires V.Schiavoni S.Ben Mokhtar, A.Boutet. X-search : Revisiting privateweb search using intel sgx. *Association for Computing Machinery*, 2017.

[8] W. Scott Stornetta Stuart Haber. How to time-stamp a digital document. *Journal of Cryptology*, 1991.

[9] Silvio Micali Georgios Vlachos Nickolai Zeldovich Yossi Gilad, Rotem Hemo. Algorand : Scaling byzantine agreements for cryptocurrencies. *MIT CSAIL*, 2017.