Oblivious RAM pour des objets de tailles variables

Orateur : Léonard Assouline
leonard.assouline@ens.fr

École Normale Supérieure, CNRS, Université PSL, Paris, France

Abstract. Lors d'une communication entre un client et un serveur, il n'est pas suffisant de seulement chiffrer les données échangées pour qu'elles restent confidentielles. En analysant le schéma d'accès effectué par le client, un serveur honnête mais curieux peut apprendre de l'information sur le calcul que le client est en train de réaliser.
Un protocole d'Oblivious RAM (ORAM) rend ce schéma d'accès inintelligible. En 1993, Goldreich et Ostrovsky présentent ce concept, avec des constructions non-triviales. Beaucoup de travaux ont depuis été faits sur ce sujet, et on dénombre des dizaines de protocoles. Les constructions à base d'arbre (Tree-ORAM), présentées en 2011 et améliorées depuis, sont les plus efficaces en pratique et nécessitent une communication polylogarithmique en le nombre d'objets à stocker. La borne inférieure théorique est atteinte par l'algorithme *OptoRAMa* en 2018.
Jusque là, tous les algorithmes faisaient l'hypothèse que les objets étaient de même taille (à l'exception d'un qui fait des hypothèses fortes sur la distribution des tailles). Nous présentons des mises à jour de constructions existantes, leur permettant pour la première d'accueillir des objets de tailles variables, sans faire d'hypothèse sur la distribution des tailles.

Définition. ***Oblivious RAM***

Un protocole d'Oblivous RAM est un couple $(\texttt{Setup}, \texttt{Access})$ *où* `Setup` *prend en argument des paramètres liés à l'algorithme ainsi que le nombre d'objets* N*, et* $\texttt{Access}(op, obj)$ *exécute une opération* $op \in \{\texttt{Read}, \texttt{Write}\}$ *sur un objet* obj*.*

L'ORAM vérifie la propriété de sécurité si pour un adversaire (le serveur) computationellement non-borné deux séquences d'accès différentes sont indistingables.

L'ORAM est correct si l'exécution d'une séquence d'accès échoue (ce qui arrive en particulier si la mémoire du client déborde) avec une probabilité négligeable.

Dans ce qui suit, nous supposons que le client stocke m objets, chacun de taille $w_i \in [0, 1]$, et tels que $\sum_{i=1}^{m} \leq N$.

Un résultat sans modification de la complexité sur certains Tree-ORAM

Nous présentons une fonction `TransVar` qui prend en argument un algorithme d'ORAM à base d'arbres (i.e. les données sont stockées dans les noeuds d'un arbre) et le transforme en un ORAM compatible avec des objets de tailles variables. Sous certaines conditions, que remplissent de nombreux ORAM, le protocole ainsi obtenu est correct et sécurisé, sans rien changer à la complexité de communication initiale : polylogarithmique en N.

Une transformation générale moyennant un coût supplémentaire

Nous avons une deuxième transformation qui part de n'importe quel ORAM et en fait un ORAM pour objets de tailles variables. Pour cela nous combinons l'ORAM de départ à une solution du problème "balls-in-bins" : on obtient une table de hachage qui permet de ranger les m objets de tailles variables dans N cases que nous appelons des "super-objets". Selon la famille de fonctions de hachage choisie, cette méthode nécessite un coût supplémentaire de $O(log(N))$ ou de $O(loglog(N))$ (cette complexité est meilleure mais nécessite de stocker la table de hachage dans un ORAM secondaire).