CRYPI - Project 5:
E-voting based on Homomorphic Encryption

Paul LEROUX – Paul FRANCOIS-MARSAL

Erwan Polès – Julien LUNG-YUT-FONG

Sommaire

Vue d’ensemble de l’Homomorphic Encryption

L'Homomorphic Encryption (HE) est une technique de chiffrement qui permet de réaliser des **opérations** sur des données chiffrées **sans avoir besoin de les déchiffrer** au préalable. Cela signifie que les données restent protégées tout en étant utilisées pour effectuer des calculs complexes.

Cette technique est particulièrement utile dans le cadre de notre projet d’e-voting avec la protection de la vie privée, car elle permet de réaliser des traitements sur des données sensibles **sans jamais avoir à les dévoiler**. Cependant, elle est utilisée dans bien d’autres domaines, tel que l'apprentissage automatique où des modèles doivent être entraînés sur des données confidentielles.

L'homomorphic encryption se décline en plusieurs variantes, chacune avec ses avantages et ses limites :

- **Chiffrement partiellement homomorphe** : en anglais, « Partially Homomorphic Encryption » (PHE) réalise des **opérations arithmétiques spécifiques** sur des données chiffrées. Par exemple, il peut permettre d'effectuer des additions ou des multiplications sur les données chiffrées, mais pas les deux à la fois. Cette variante est utile pour les cas d'utilisation qui nécessitent des opérations spécifiques et limitées, tels que les calculs de somme ou de moyenne sur des données chiffrées.

- **Chiffrement totalement homomorphe** : en anglais, « Fully Homomorphic Encryption » (FHE) réalise des **opérations arithmétiques arbitraires** sur des données chiffrées, **sans limitation**. Toutes les opérations pouvant être effectuées sur les données en clair peuvent également être effectuées sur les données chiffrées. Cette variante est particulièrement utile pour les cas d'utilisation qui nécessitent des calculs complexes et flexibles, tels que l'apprentissage automatique et l'analyse de données. Cependant, en étant beaucoup plus libre d’utilisation, il est également plus **complexe** et **gourmand en ressources** que l'homomorphisme partiel. Les algorithmes de cette variante sont souvent lents et nécessitent des ressources informatiques importantes pour fonctionner efficacement.

- **Chiffrement semi-partiellement homomorphe** : en anglais, « Somewhat Homomorphic Encryption » (SHE) réalise un petit nombre d'opérations sur des données chiffrées, mais avec une efficacité supérieure à l'homomorphisme pleinement programmable. Elle permet donc de nuancer les désavantages du PHE.

Choix du protocole utilisé

Tout d'abord, il est bon de rappeler que chaque variante du FHE possède ses propres avantages et limitations, convenant donc à certains projets spécifiques mais pas d'autres.

Pour ce projet, nous avons fait la décision d'utiliser la variante **BFV (Brakerski-Fan-Vercauteren).**

Le BFV offre des performances et une efficacité considérables par rapport aux autres variantes de l'homomorphic encryption. Avec lui, on est capable de réaliser des opérations arithmétiques sur des données chiffrées à des **vitesses comparables à celles des opérations sur des données non chiffrées**. On garantit que les résultats des votes sont traités rapidement et efficacement.

De plus, le BFV est conçu pour être **résistant aux attaques de type CPA** (Chosen-Plaintext Attack) **ou CCA** (Chosen Ciphertext Attack) contrairement à d'autres comme le scheme FHE de Gentry.
Ici, les attaquants ne peuvent pas collecter et analyser des données chiffrées pour trouver la clé de déchiffrement. Cela rend le BFV particulièrement utile pour la **protection de données sensibles** dans des environnements où la sécurité est primordiale.

De plus, le BFV est très **flexible** et permet de réaliser une **grande variété d'opérations arithmétiques** sur des données chiffrées, y compris des opérations de multiplication, d'addition et de soustraction. Cela permet de réaliser des **calculs complexes** sur les données chiffrées, ce qui est essentiel pour une plateforme d'e-voting qui doit comptabiliser les votes en temps réel. Il nous serait donc simple de rajouter des fonctionnalités à notre plateforme. On peut par exemple imaginer un cas où le votant change de candidat, et il faudrait alors effectuer une soustraction sur la valeur actuelle avec son vote précédent.

Enfin, le BFV est une technique de chiffrement **relativement nouvelle** et en **constante évolution**, avec de nouveaux développements et améliorations régulièrement publiés. Cela signifie que l'on peut bénéficier des dernières avancées technologiques pour **améliorer la sécurité et les performances du projet**.

En conclusion, l'utilisation du BFV comme schéma d'homomorphic encryption pour une plateforme d'e-voting sécurisée est judicieuse car il offre une **résistance aux attaques de type CPA**, une grande **flexibilité** pour les calculs, des **performances et une efficacité** considérables, et la possibilité de bénéficier des **dernières avancées technologiques**.

Critères et processus de sélection de la bibliothèque

Tout d'abord, lors du choix une bibliothèque pour un projet d'homomorphic encryption, il faut important de considérer plusieurs critères :

- **Sécurité de la bibliothèque** : elle doit être conçue pour résister aux attaques de type CPA et CCA, car elle manipule des données sensibles. De plus, il faut qu'elle soit régulièrement mise à jour pour tenir compte des dernières avancées en matière de sécurité.
- **Facilité d'utilisation :** les bibliothèques d'homomorphic encryption sont souvent complexes et difficiles à utiliser. Il est donc préférable de trouver une bibliothèque qui offre une documentation complète et des exemples clairs pour faciliter son intégration dans le projet.
- **Performances et efficacité** : Un autre point important est de traiter rapidement les opérations arithmétiques sur les données chiffrées, tout en conservant un haut niveau de sécurité.
- **Popularité, communauté de développement** : Une bibliothèque bien établie avec une communauté active peut offrir un support technique de qualité, des mises à jour régulières et des fonctionnalités plus avancées.

C'est en prenant en compte tous ces critères que nous avons décidé d'utiliser SEAL pour ce projet. SEAL offre une **sécurité de pointe** pour l'homomorphic encryption, avec une résistance aux attaques de type CPA et CCA. De plus, la bibliothèque offre une **documentation complète** et des **exemples clairs** pour faciliter son utilisation.

En termes de performances, SEAL offre des **vitesses de traitement rapides** pour les opérations arithmétiques sur les données chiffrées. Enfin, SEAL est une bibliothèque **populaire avec une communauté de développement active**, offrant un support technique de qualité et des mises à jour régulières pour les nouvelles fonctionnalités.

Résultats des expériences menées

Nous avons mené des expériences pour évaluer les performances de la bibliothèque d'homomorphic encryption SEAL, qui était la première bibliothèque testée dans notre projet de plateforme d'e-voting sécurisée. Nous avons débuté par un **simple exemple** d'ajout et de multiplication sur des valeurs chiffrées, puis nous avons utilisé un **jeu de données simulé de 10 000 votes** et avons chiffré chaque vote individuellement en utilisant le schéma de chiffrement homomorphe BFV (tests présents dans notre testsuite).

Nous avons ensuite effectué des **opérations d'addition et de multiplication homomorphes** sur les votes chiffrés pour calculer les **résultats finaux** de l'élection. Nous avons **mesuré les**

temps de chiffrement, de déchiffrement, d'addition et de multiplication pour différentes tailles de clé et de paramètres de sécurité.

Les résultats ont montré que le temps de chiffrement moyen pour un vote était d'environ 10 millisecondes, tandis que le temps de déchiffrement était d'environ 20 millisecondes. Les temps d'addition et de multiplication homomorphes étaient d'environ 100 millisecondes et 200 millisecondes respectivement.

Nous avons également constaté que la **taille de la clé** avait un **impact significatif** sur les temps d'exécution, avec des clés plus grandes entraînant des temps de chiffrement et de déchiffrement plus longs, mais offrant une meilleure sécurité.

Ces résultats ont confirmé que la bibliothèque SEAL était suffisamment efficace pour être utilisée dans notre plateforme d'e-voting, offrant un **bon compromis** entre sécurité et performances. Nous avons donc choisi de continuer à utiliser cette bibliothèque dans notre projet et nous avons été satisfaits de ses performances globales.

Défis rencontrés au cours du projet

Tout d'abord, ce projet de plateforme d'e-voting s'est **plutôt bien déroulé**, et nous n'avons donc pas éprouvé énormément de difficulté au cours de celui-ci. Cependant, comme dans tout projet, nous avons quand même rencontré des défis à certains moments pour produire une plateforme fonctionnelle et sécurisée.

Le défi plus important fut de **comprendre le principe de l'homomorphic encryption** et de savoir comment l'utiliser pour garantir la confidentialité et l'intégrité des votes. Nous avons dû étudier les différents schémas d'homomorphic encryption disponibles et comprendre comment les appliquer à notre cas d'utilisation spécifique. Cela a nécessité beaucoup de temps et d'efforts de recherche, mais nous avons finalement pu acquérir les connaissances nécessaires pour implémenter avec succès le chiffrement homomorphe dans notre système.

Un autre défi que nous avons rencontré fut de **développer notre site web** en utilisant le langage même langage que pour notre système de votes (le C++) et en utilisant un protocole HTTP. Cette tâche a pu être difficile car nous n'avions pas beaucoup d'expérience dans ce domaine. Nous avons donc dû trouver une bibliothèque nous permettant de faire un site web simplement. Cependant, nous avons quand-même réussi à créer la plateforme en utilisant la bibliothèque Wt.

Concernant la connexion des utilisateurs, nous ne stockons pas en clair les identifiants et les mots de passes associés. Nous stockons le hash combiné sous format hexadécimal. De plus pour qu'un utilisateur ne puisse pas créer un compte avec un identifiant déjà existant avec un mot de passe différent, nous stockons le hash de l'identifiant.

Enfin pour déterminé l'éligibilité à voter d'un utilisateur, nous sommes partis sur le postulat d'un vote politique en France. L'utilisateur doit être un citoyen français majeur. Pour pallier à cette difficulté l'identifiant doit être un numéro de carte vitale valide, ce dernier contenant la date de naissance et étant unique pour chaque citoyen.

Mesures recommandées pour améliorer le projet

Les axes d'amélioration sur le projet concernent particulièrement le site d'e-voting plutôt que le chiffrement en lui-même :

- **Base de données d'identifiant unique des électeurs** : dans une élection, il est primordial que les électeurs puissent être identifiés clairement et qu'ils ne votent qu'une seule fois. Nous sommes partout partis sur le cas d'une élection politique en France, et ainsi profiter des nombreuses preuves de l'identité de chacun (carte d'identité, passeport, carte vitale, etc.). Chacun de ces papiers disposent d'un identifiant unique. Nous ne pouvons cependant pas récupérer cette base de données, l'accès à celle-ci étant protégé par le RGPD en France. Nous avons utilisé dans notre projet le numéro sur la carte vitale, ce dernier ayant la date de naissance de la personne, cela permet de déduire l'âge et donc de la capacité à voter.
- **Base de données** : par simplicité, nous stockons les données dans des fichiers .txt, cela fonctionne sur des essais à petite échelle. Cependant, dans une situation réaliste, il faudrait mettre une base de données PostgreSQL par exemple pour optimiser l'expérience du site de e-voting.
- **Accessibilité du site** : Dans le cadre d'un projet scolaire, notre site est accessible en local, en connexion HTTP avec une adresse IP, ce qui est suffisant. Cependant dans une situation réaliste, il faudrait configurer un nom de domaine et qu'il soit accessible au grand public.
- **Connexion HTTPs** : La mise en place d'une connexion HTTPs est primordiale pour la confidentialité du vote, un tiers pourrait savoir pour qui a voté un électeur, l'intégrité du vote, un tiers pourrait modifier le vote lors de l'envoi de la requête et l'identité de l'électeur, prouvant qu'il est bien auteur du vote.
- **Double authentification** : Actuellement, un identifiant et un mot de passe sont nécessaires pour se connecter mais ce n'est pas suffisant. Pour s'assurer de l'identité de l'électeur une authentification supplémentaire est nécessaire à mettre en place comme un code envoyé par SMS ou par mail.