Prédictions sécurisées en utilisant le calcul multipartite sécurisé

Projet 4

04 / 05 / 2023

Antony Rouve, Guilhem Carlet,
Thibault Saccon, Magali Jomier

Table des matières

Abstract

Le calcul multipartite sécurisé (**SMC**) permet à plusieurs parties de calculer le résultat d'une fonction publique sans pour autant révéler leurs entrées privées. Le modèle **SMC** garantit, grâce à plusieurs concepts mathématiques, que chaque participant ne dispose pas de suffisamment d'informations pour déduire les données d'entrées des autres participants.

Cette garantie de confidentialité permet de résoudre des situations où il n'est pas acceptable de divulguer des informations tout en effectuant du calcul partagé entre plusieurs acteurs.

Dans ce document, nous allons aborder le fonctionnement général du SMC, puis comment nous l'avons utilisé afin de résoudre un problème concret impliquant 2 parties en utilisant la librairie python **MPyC**. Dans notre situation, nous avons considéré que nos parties sont semi-honnêtes et qu'il n'y a aucune partie malicieuse.

Dans le cas étudié, les prédictions produites à partir des données fournies ne sont pas fiables et n'arrivent pas à détecter correctement les personnes qui vont mourir dans 10 ans.

1 Introduction

1.1 Aperçu du calcul multipartite sécurisé (**SMC**)

Lorsque plusieurs parties veulent calculer quelque chose, l'approche la plus naturelle est de désigner un tiers de confiance qui va recevoir les entrées de chaque partie afin d'effectuer l'opération concrète. Ce modèle fonctionne bien tant que l'ensemble des parties acceptent de révéler leurs données au tiers de confiance. Ce tiers de confiance représente un point unique de dysfonctionnement qui peut être compromis ou indisponible empêchant les parties d'obtenir leurs résultats.

L'intérêt du **SMC** est de toujours effectuer ces calculs sur les entrées de chaque partie d'une manière telle qu'il soit impossible de déduire les données d'entrée des autres parties. Ce modèle assure la confidentialité des données d'entrée ainsi que l'exactitude du résultat du calcul.

1.2 Partage du secret

Pour assurer la confidentialité des données, le **SMC** partage le secret donné en sous secrets qui pourra être réassemblé afin de reconstituer la donnée d'entrée. Cette étape permet de distribuer les sous secrets découpés à l'ensemble des parties intermédiaires afin de ne jamais révéler la donnée d'entrée. De cette manière, s'il n'y a aucune collaboration malicieuse, il sera impossible aux parties de déduire le secret des autres parties.

Il existe plusieurs méthodes de partage du secret permettant de découper ainsi que de rassembler une donnée d'entrée. Nous allons en détailler la méthode du partage additif ainsi que la méthode de partage de Shamir.

1.2.1 Partage additif du secret

Cette méthode de partage est l'une des plus faciles à mettre en œuvre car elle se base sur un principe mathématique très simple. Le partage additif du secret utilise des opérations d'addition afin de séparer et de réassembler le secret en parts. Il est définie de la manière suivante :

- Partage(x): $Soit\ x \in G,\ les\ éléments\ x_1, x_2 ...,\ x_n\ sont\ choisis\ tel\ que:\ x_n = x - \sum_{i=1}^{n-1} x_i$
- Réassemblage($x_1, x_2, ..., x_n$): $x = \sum_{i=1}^{n} x_i$

En ce qui concerne l'opération de partage, elle consiste à choisir n éléments aléatoires de telle sorte que leurs sommes soit égale au secret original. La possession d'une part ne révèle aucune information sur le secret originel et il n'est pas possible de déduire le secret sans posséder l'ensemble des parts.

L'opération de réassemblage du secret consiste à simplement additionner l'ensemble des parts afin de retrouver le secret initial.

Cette méthode peut s'appliquer à tous objets mathématiques qui implémentent une opération d'addition. Nous pouvons appliquer ce principe à des nombres, des chaînes de caractères ainsi qu'à des images (*diapositive n°8, 9 et 10*).

La principale faiblesse de cette méthode de partage est qu'une part du secret peut révéler beaucoup d'informations sur le secret initial en fonction de l'objet utilisé. Par exemple, si nous appliquons cette méthode de partage sur des chaînes de caractères, la possession d'une part du secret peut nous révéler la longueur ainsi que la position de certaines lettres.

De plus, si l'une des parties n'est plus en capacité de partager sa part du secret, il sera dans certains cas impossible de reconstituer le secret.

1.2.2 Partage de secret Shamir

Le partage de secret avec la méthode de Shamir est une autre approche du partage du secret. Le partage de secret Shamir se base sur le théorème d'interpolation de Lagrange permettant de reconstituer un polynôme en connaissant un ensemble de points y appartenant. Il est défini comme suit :

Il est nécessaire de choisir préalablement une valeur seuil s qui détermine la quantité requise de part à assembler afin de retrouver le secret. Les opération sont définies comme suivant:

- Partage(x):
  - Choisir un polynôme P secret de degré $s - 1$ tel que $P(X = 0) = x$
  - Une part du secret est un point (x, y) aléatoire appartenant à P tel que $P(X = x) = y$
- Réassemblage(x_1, x_2 , ..., x_n):
  - Si nous disposons de s parts du secret d'un polynôme de degré $s - 1$, alors le théorème d'interpolation de Lagrange affirme qu'il est possible de reconstruire le polynôme. De cette manière, une fois que le polynôme est interpolé, il suffit de calculer $P(X = 0)$ afin de retrouver le secret.

La méthode de Shamir est très efficace car il est facile de générer des parts du secret et que ces dernières ne révèlent pratiquement aucune information sur le polynôme secret utilisé. De plus, la définition de cette valeur de seuil est totalement indépendante du nombre de parts qui seront distribuées. C'est une méthode adaptée aux situations où la participation de l'ensemble des parties n'est pas nécessaire pour retrouver le secret initial.

1.3 Protocoles du calcul multipartie sécurisé

Avant d'aborder les protocoles existants afin de suivre le modèle du **SMC**, il est important de définir les 2 scénarios de fonctionnement.

Le scénario **semi-honnête** (*diapositive n°17*) décrit la situation dans laquelle l'ensemble des parties respectent le protocole mais essayent d'obtenir le plus d'informations possible.

Contrairement au scénario **semi-honnête**, le scénario **malicieux** correspond aux situations où les parties peuvent collaborer de manière arbitraire et sont libres de ne pas suivre le protocole afin d'obtenir le plus d'informations possible.
Dans le cadre de l'exercice, nous allons nous restreindre au cas **semi-honnête**.

Afin que l'ensemble des parties soient en mesure de calculer le résultat d'une fonction publique, il est nécessaire qu'elles suivent un protocole définissant la démarche à suivre. Nous allons détailler dans la partie suivant le protocole de Goldreich Micali Wigderson.

1.3.1 Goldreich Micali Wigderson (**GMW**)

Le protocole **GMW** se base sur plusieurs étapes afin de permettre aux parties de calculer de manière sécurisée:

- **Étape 1**: Transformer la fonction que nous souhaitons calculer en circuit logique.
  Le but de cette étape est de découper une fonction en plusieurs opérations indépendantes représentées par des portes logiques que nos parties pourront évaluer.

- **Étape 2**: Procéder au partage de secret suivant la manière additive.
  Ce protocole se base sur le partage additif du secret. L'acteur injectant des données devra découper son secret de cette manière afin de les envoyer aux parties intermédiaires qui effectueront les calculs.

- **Étape 3**: Évaluer les étapes du circuit logique de la fonction au cas par cas.
  Chaque partie va évaluer les portes logiques de manière synchronisée. Certaines opérations comme l'addition peuvent être effectuées localement sur chaque partie de manière très rapide. D'autres opérations comme la multiplication nécessitent une communication entre les parties ainsi qu'une synchronisation globale ce qui ralenti considérablement les échanges.

- **Étape 4**: Assembler les sorties pour trouver le résultat.
  Une fois l'ensemble des portes du circuit logique évalué par les parties de calcul, le parti de sortie va récupérer toutes les parts du secret détenues afin de reconstituer le résultat de sortie.

1.3.2 Autres protocoles

Il existe beaucoup d'autres protocoles de **SMC** possédant chacun leurs forces et leurs faiblesses ainsi qu'un contexte d'application.

Nous pouvons citer par exemple le protocole de Ben-Or Goldwasser Widgerson (**BGW**). Ce protocole est très similaire au protocole **GMW** de part l'évaluation de portes logiques représentant une fonction tout comme son fonctionnement en mode **Semi-Honnête**. Il supporte les opérations d'addition et de multiplication par des constantes publiques en utilisant la méthode du partage de secret de Shamir.

Les protocoles basés sur SPDZ ainsi que Tiny-OT sont des protocoles plus avancés qui sont capables de gérer des opérations comme la multiplication à l'aide de la technique des **triplets multiplicatif** ainsi que de résister à une minorité **malicieuse**. Ces protocoles sont malheureusement plus lents de part les opérations supplémentaires qu'ils effectuent.

2 Le problème

2.1 Résumé du problème

Dans ce projet, nous allons considérer 2 parties: un propriétaire de données ainsi qu'un propriétaire d'un modèle statistique capable d'effectuer des prédictions. Le propriétaire de données dispose de données de santé qu'il n'est pas possible de révéler à qui que ce soit. De même, le propriétaire du modèle ne veut pas révéler son savoir-faire qu'il va garder précieusement pour lui.

Ce projet vise à ce que le propriétaire des données reçoive une prédiction de ses données sans prendre connaissance du modèle tout en ne révélant aucune de ses données médicales.

Pour résoudre ce problème, il nous faut préserver la confidentialité des données d'entrées apportées par le propriétaire des données. Nous allons donc utiliser le modèle du calcul multipartite sécurisé pour répondre à ce besoin de confidentialité.

2.2 Choix du protocole et de la librairie

Dans le cadre de l'exercice, les deux acteurs sont considérés comme **semi-honnêtes** et nous n'aurons pas besoin d'utiliser un protocole sécurisé contre le cas **malicieux**. Ce protocole devra aussi supporter nativement l'opération de la multiplication afin de pouvoir calculer correctement la fonction de prédiction.

En ce qui concerne notre choix de librairie, notre critère de sélection majeur s'est porté sur le langage de programmation. En effet, il n'allait pas être possible d'apprendre et de maîtriser le Rust en quelques semaines. Nous avons orienté notre choix vers un langage simple d'utilisation et massivement utilisé afin de trouver une librairie de **SMC** recommandée et maintenue.
Nous avons donc choisi le langage de programmation Python pour sa simplicité ainsi que pour son élégance.

Parmi l'ensemble des librairies Python proposant des abstractions sur des protocoles de **SMC**, nous avons choisi la librairie **MPyC**:

https://github.com/lschoe/mpyc

Cette librairie nous a beaucoup attiré de par son nombre d'étoiles, la réactivité de son auteur pour répondre aux issues postées ainsi que la présence d'une multitude d'exemples dans le dossier **demos** du dépôt.

La librairie semble implémenter une version du protocole **GMW** modifiée en utilisant le partage de secret Shamir au lieu d'un partage de secret additif. Ce protocole étant adapté à notre situation, nous avons décidé de l'utiliser.

2.3 Implémentation

Tout d'abord, voici un exemple d'utilisation basique de la librairie **MPyC**, que nous avons ensuite adapté afin de répondre au problème posé :

Dans un premier temps on initialise nos parties à l'aide de la fonction `mpc.start()`. Le nombre de parties peut-être aussi choisi lors du lancement du programme à l'aide de l'argument `-M` ou alors lors en argument à `mpc.start()` pendant l''initialisation.

Ensuite, on partage de façon secrète les nombres "sécurisés "42,12,36 à l'aide de la fonction `mpc.input()`.

La prochaine étape est de déclarer une fonction coroutine qui sera exécutée sur chaque partie lors de l'appel de la fonction `mpc.output()`.

Finalement, lors de l'appel à la fonction `mpc.output()`, la bibliothèque **MPyC** va se charger de répartir les calculs de la fonction `function` sur l'ensemble des parties de telle sorte à ne jamais révéler les secrets d'entrée aux parties en utilisant les protocoles de **SMC** vu précédemment.

Script d'exemple d'utilisation de la librairie **MPyC**

```python
from mpyc.runtime import mpc

secint = mpc.SecInt()

mpc.run(mpc.start())

x_share = mpc.input(secint(42), senders=0)
y_share = mpc.input(secint(12), senders=0)
z_share = mpc.input(secint(36), senders=0)

@mpc.coroutine
async def function(x_share, y_share, z_share):
    return x_share + y_share + z_share

result = mpc.run(mpc.output(function(x_share, y_share, z_share)))

print("Result of the computation: ", result)

mpc.run(mpc.shutdown())
```

2.4 Résultats

Nous avons donc pu utiliser nos deux acteurs afin de calculer les prédictions sur les données médicales à l'aide d'un modèle.

Afin de conclure sur la fiabilité des prédictions, nous avons fait des statistiques sur celles-ci.

Avant d'étudier ces statistiques, nous devons remettre dans le contexte nos données et nos besoins sur les prédictions. Nous sommes dans un domaine médical, un domaine où l'erreur n'est que très peu acceptable, nous attendons donc des prédictions exemplairement fiables et justes. Un diagnostic médical ne doit en rien se baser sur de l'aléatoire ou sur des lois de probabilités de populations.

Avec le jeu de données fourni, nous avons pu conclure que les prédictions faites répondent dans 87,02% des cas la bonne réponse. Comme dit précédemment, dans le cadre médical de notre étude de cas, ce résultat n'est clairement pas satisfaisant (voir décevant) mais peut être acceptable dans d'autres contextes.

Pour aller plus loin dans notre analyse, nous avons calculé le nombre de fois où la prédiction est véridique sur une sous-population du jeu de données: seulement les patients qui ne développeront pas de maladie coronarienne dans 10 ans. Sur cette population, 99,00% des prédictions sont exactes. Ce résultat est satisfaisant mais nous le considérons tout de même faible dans notre contexte.

Nous avons ensuite calculé le nombre de fois où la prédiction est correcte sur la sous-population opposée: la population qui est à risque de développer une maladie coronarienne dans les 10 ans. Sur cette population, 7,23% des prédictions sont exactes. Nul besoin d'expliquer en quoi ce nombre est inacceptable, et ce, dans n'importe quel contexte.

Pour conclure sur les résultats obtenus, nous pouvons dire que les prédictions réalisées sont inadéquates et inacceptables. Les prédictions répondent dans une grande majorité des cas que le patient ne développera pas de maladie coronarienne dans les 10 années à venir. Le modèle utilisé n'est donc pas de confiance et doit être évité. Il répond presque toujours que le patient ne sera pas malade et réussit tout de même à ne pas détecter 1% des cas sur une population saine.

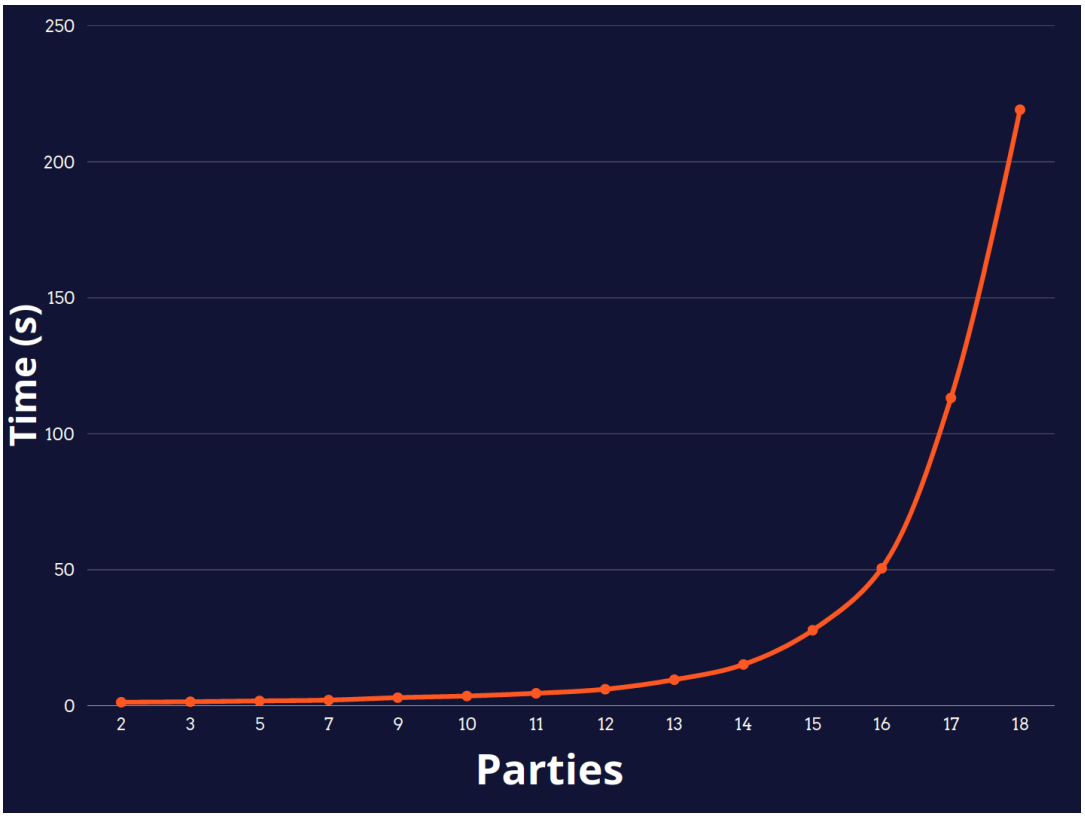
Pour notre contexte, le modèle fourni est mauvais et cela peut s'expliquer pour plusieurs raisons. Tout d'abord, il se peut qu'il ait été élaboré trop vite et sans expertise ou compétences. Une autre explication pourrait être que le modèle ait été entraîné avec un jeu de données biaisé. En effet, dans le jeu de données d'entraînement, sur les 2966 personnes, seulement 478 vont développer une maladie coronarienne. Ce jeu de données est peut-être représentatif de la réalité sur une population tirée aléatoirement mais n'est pas adapté à l'apprentissage et à l'entraînement d'un modèle de prédictions.

2.5 Performances

Afin de mesurer les performances temporelles de ce modèle, nous avons effectué des tests de performances en augmentant progressivement le nombre de parties prenantes pour effectuer les calculs. Nous avons constaté que le nombre de parties influe fortement le temps d'exécution du programme.

En effet, si nous devons calculer des prédictions sur un grand nombre de données en faisant appel à plus de 15 parties, nous atteindrons des temps de calculs inacceptables.

Temps de traitement pour un nombre de parties donné:



3 Problèmes rencontrés

3.1 Problème 1

Le premier problème que nous avons rencontré, a été dans la compréhension du sujet. En effet, nous ne connaissions rien au principe de calcul multipartite ainsi qu'à ses protocoles. Nous avons découvert les configurations **malicieuses** et **semi-honnêtes** au travers d'articles de recherches qui n'étaient pas tous triviaux à aborder.

Nous avons aussi passé beaucoup de temps à visionner des rediffusions de conférences sur Youtube ainsi que de survoler des articles de recherches à ce sujet afin d'avoir une vision beaucoup plus globale et éclairée du **SMC**.

3.2 Problème 2

Le second point qui nous a posé problème se situait sur la compréhension des méthodes exposées par la librairie. En effet, nous avons découvert cette librairie pour ce projet et il ne nous a pas été évident de faire le lien entre cette dernière avec les notions de **SMC** assimilées au travers de vidéos. Lorsque nous avons commencé à lire les exemples proposés par l'auteur de la librairie, il y a beaucoup de passages que nous ne comprenions pas.

Avec plusieurs relectures ainsi que de multiples tests, nous pensons avoir compris suffisamment le fonctionnement de cette librairie pour répondre au problème donné.

3.3 Problème 3

La dernière partie du projet qui a représenté une difficulté majeure a été un manque de documentation générale sur les différents protocoles de **SMC**. Nous avons recueilli le plus d'informations possible concernant les protocoles les plus utilisés. Cependant, nous n'avons pas réussi à comprendre en détail le fonctionnement des protocoles SPDZ tout comme le Tiny-OT ce qui nous a empêché de comparer en détail leurs forces ainsi que leurs faiblesses.

3.4 Problème 4

Le dernier problème que nous avons rencontré a été l'utilisation de la fonction de prédiction avec les parties. Nous n'avons pas réussi à utiliser la fonction exponentielle avec les parts du secret qui ont été partagées avec chaque partie. La fonction de prédiction est donc approximée et ne renvoie pas le bon résultat. Nos résultats décrits précédemment ont donc été réalisés sans **SMC** pour en conclure que le modèle est faux.

4 Améliorations possibles

Une amélioration possible pour obtenir une meilleure prédiction serait de changer de modèle. En effet, les résultats des prédictions sur le jeu de données de test présentés plus haut sont mauvais. Le modèle n'est pas capable de reconnaître les personnes à risque de maladie coronarienne au bout de 10 ans. De plus, le jeu de données d'entraînement est biaisé. Changer les données d'entraînement permettrait de mieux entraîner le modèle et de le rendre plus précis dans la détection des personnes à risque.

Ensuite, le SMC est une méthode très utilisée pour traiter des données confidentielles. Cependant, comme vu précédemment, cette méthode peut être lente pour des données volumineuses ou pour de nombreuses parties prenantes. Pour palier à ce problème, il est possible de choisir une autre méthode comme la cryptographie homomorphique. Cette technique permet de réaliser des calculs sur des données chiffrées sans avoir besoin de les déchiffrer, ce qui peut améliorer considérablement les performances. De plus, il est aussi possible de choisir un autre protocole pour le SMC qui serait plus optimisé et plus performant pour les échanges entre de nombreuses parties comme SPDZ.

5 Conclusion

En conclusion, le **SMC** présente une multitude d'avantages au coût de performances et d'une complexité non négligeable. Les garanties de confidentialité que ce modèle apporte permettent de répondre au problème mais ne permettent pas un passage à l'échelle massif dû aux multiples échanges réseau entre chaque parties et à la complexité de mise en place. Cette approche permet de résister contre des adversaires disposant d'une grande capacité de calcul ainsi que de fonctionner malgré la présence de parties malveillantes.

Cependant, le **SMC** est un domaine récent applicable à beaucoup d'autres domaines. Il est encore en pleine évolution et de nouvelles avancées pour améliorer ses performances ainsi que sa complexité sont en cours. Ils pourraient probablement permettre une utilisation viable à grandes échelles dans les années à venir.

6 Références

https://github.com/lschoe/mpyc

https://en.wikipedia.org/wiki/Garbled_circuit

https://en.wikipedia.org/wiki/Shamir%27s_secret_sharing

https://www.youtube.com/watch?v=kkMps3X_tEE

https://www.youtube.com/watch?v=iFY5SyY3IMQ

https://www.youtube.com/watch?v=xwxkp4fMWsk

https://www.youtube.com/watch?v=K54ildEW9-Q

https://www.di.ens.fr/~nitulesc/files/slides/MPC-intro.pdf

https://sands.edpsciences.org/articles/sands/pdf/2022/01/sands20210001.pdf

https://wiki.mpcalliance.org/protocols.html

https://www.youtube.com/watch?v=HOqv5xzrlFI

https://eprint.iacr.org/2019/1390.pdf

https://www.cs.purdue.edu/homes/hmaji/teaching/Fall%202017/lectures/39.pdf

https://eprint.iacr.org/2020/1577.pdf

https://link.springer.com/chapter/10.1007/978-3-030-17659-4_15

http://cyber.biu.ac.il/wp-content/uploads/2017/01/10-1.pdf

https://jonasspenger.github.io/blog/bgw-protocol-and-beaver-triples

https://eprint.iacr.org/2015/931.pdf

https://eprint.iacr.org/2020/521.pdf