Authentification biométrique par calcul multipartite sécurisé

Matthieu Colin | Alexandra Delin | Jules Diaz | Maëlys Rimbert

July 2024

EPITA

Introduction

- Deux participants :
  1. un serveur ayant une base de données de visages autorisés
  2. un client fournissant les données biométriques de son visage
- Objectif : déterminer si le client est connu par la base de données
- Solution en Python car il existe un large choix de bibliothèques de reconnaissance faciale

Figure 1: Python [16]

Face Recognition vs Deepface

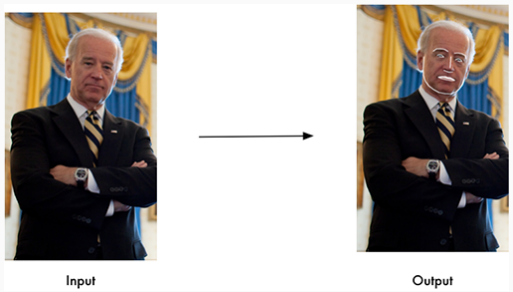

Figure 2: Démonstration de face-recognition [16]

Figure 3: Logo Deepface [16]

- Bibliothèques Python de reconnaissance faciale
- Fonctionnalités similaires, dont l'extraction d'*embeddings* correspondant à des visages
- *Embeddings* de 128 et 4096 éléments
- Problèmes de performance avec Deepface : nos machines n'étaient pas assez puissantes pour supporter les calculs sur les vecteurs

Nous avons choisi de travailler avec **face-recognition** afin de simplifier les opérations et de réduire le temps de calcul.

Calcul Multipartite Sécurisé

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange
- Opération possible sur les parties

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange
- Opération possible sur les parties
  - Addition

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange
- Opération possible sur les parties
  - Addition
  - Multiplication par un scalaire

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange
- Opération possible sur les parties
  - Addition
  - Multiplication par un scalaire
  - Multiplication

Partage de clé secrète de Shamir

- *Shamir's Secret Sharing*
  - Permet de **scinder un secret en plusieurs morceaux** appelées *parties*
  - Chaque *partie* ne donne **aucune information** sur le secret en lui même
  - L'algorithme permet de définir un seuil indiquant le nombre de *parties* minimum afin de pouvoir reconstituer le secret
- Principe simple basé sur :
  - Les propriétés des polynômes
  - L'interpolation de Lagrange
- Opération possible sur les parties
  - Addition
  - Multiplication par un scalaire
  - Multiplication
  - ...

Exemple - Seuil à deux

- Prenons le secret $s = -1$
- Formons le polynôme $q(x) = 2x - 1$
  - 2 un coefficient aléatoire
  - -1 notre secret
- Exemple de points partageables :
  - (1, 1)
  - (2, 3)

Exemple - Visualisation

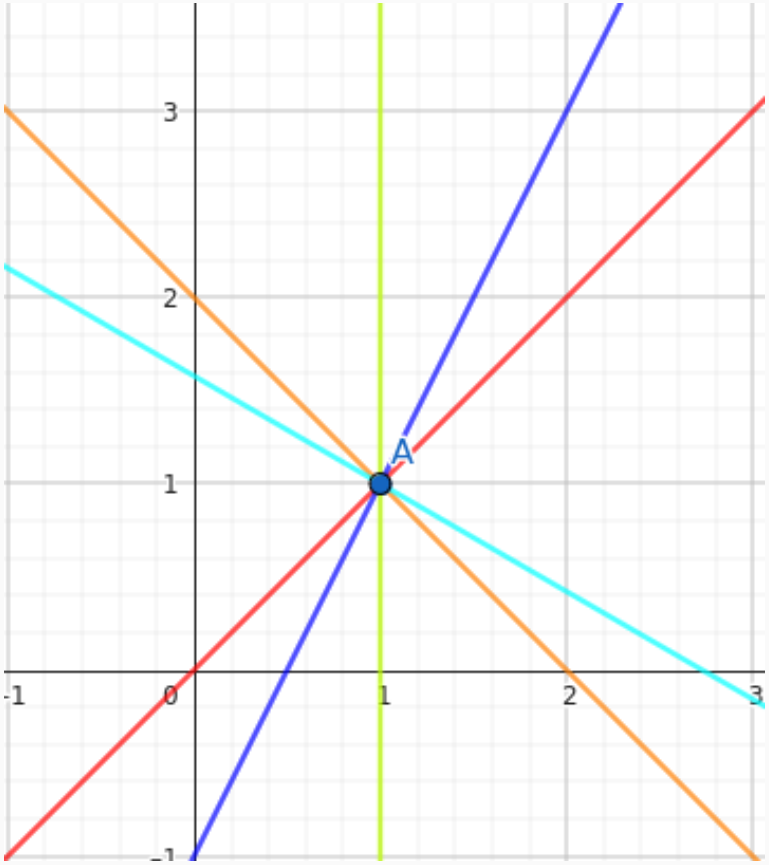

Figure 4: Infinité de solutions

Exemple - Visualisation

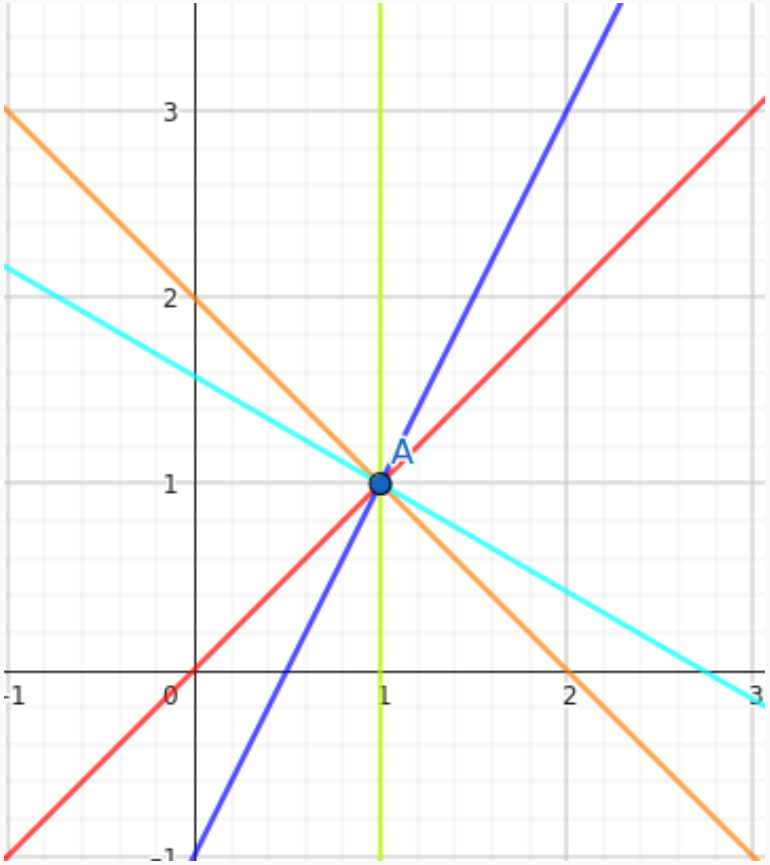

Figure 4: Infinité de solutions

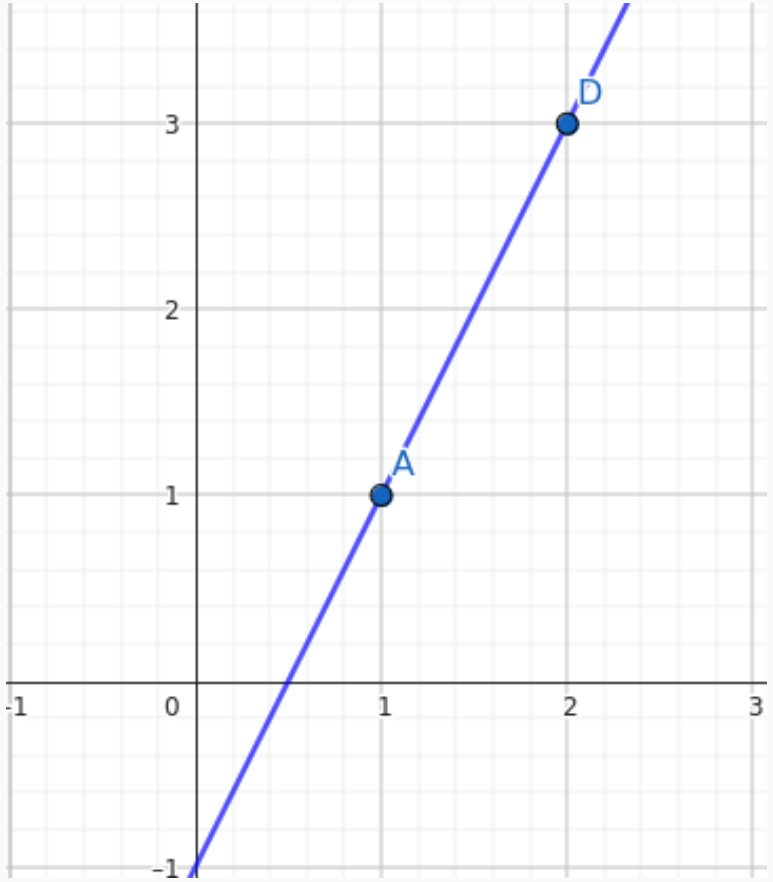

Figure 5: Solution unique

Exemple - Construction du polynôme

- Interpolation Lagrangienne :

$$L(X) = \sum_{j=0}^{n} y_i \left(\prod_{j=0, i \neq j}^{n} \frac{X - x_i}{x_j - x_i} \right)$$

$$q(x) = 1 * \frac{x-2}{1-2} + 3 * \frac{x-1}{2-1} = 2x - 1$$

Exemple - Construction du polynôme

- Interpolation Lagrangienne :

$$L(X) = \sum_{j=0}^{n} y_i \left(\prod_{j=0, i \neq j}^{n} \frac{X - x_i}{x_j - x_i} \right)$$

$$q(x) = 1 * \frac{x-2}{1-2} + 3 * \frac{x-1}{2-1} = 2x - 1$$

- Récupération du secret :

$$q(0) = -1$$

MPyC

- *Multiparty Computation in Python*

MPyC

- *Multiparty Computation in Python*
  - Panel considérable de fonctionnalités

MPyC

- *Multiparty Computation in Python*
  - Panel considérable de fonctionnalités
  - Simple d’utilisation

MPyC

- *Multiparty Computation in Python*
  - Panel considérable de fonctionnalités
  - Simple d’utilisation
  - Exemples clairs

MPyC

- *Multiparty Computation in Python*
  - Panel considérable de fonctionnalités
  - Simple d'utilisation
  - Exemples clairs
  - Bibliothèque maintenue

MPyC

- *Multiparty Computation in Python*
  - Panel considérable de fonctionnalités
  - Simple d'utilisation
  - Exemples clairs
  - Bibliothèque maintenue
- S'articule facilement avec les bibliothèques de reconnaissances faciales

Expérimentation & Résultat

7 scripts Python

- Distance euclidienne entre deux images utilisant le **SMC**
- Script d'évaluation de performances de l'algorithme MPyC
- Extraction des *faces encodings* d'un jeu d'image
- Comparaison entre la bibliothèque *DeepFace* et *Face Recognition*
- Démonstrateur (Autorisation par reconnaissance faciale)
  - Serveur
  - Client

Expérimentation & Résultat

Calcule de la distance euclidienne avec la bibliothèque MPyC

```
    embedding = secfpx.array(face_encoding)

    # Recuperation de l'image de la seconde partie
    user, server = mpc.input(embedding)

    # print('Computing the distance')
    distance = np.subtract(user, server)
    # print('Computing the euclidian distance')
    # print('Multiply')
    euclidian = np.multiply(distance, distance)
    # print('Sum')
    euclidian = np.sum(euclidian)

    # print('Printing the result')
    euclidian = await mpc.output(euclidian)
    # print('Sqrt')
    euclidian = np.sqrt(euclidian)
    # print('Result', euclidian)
    return euclidian

```

Évaluation des performances

Il faut moins de **30 ms** pour vérifier si deux images contiennent la même personne.

Nous avons déterminé que la valeur de seuil la plus optimal était proche de **0,575**

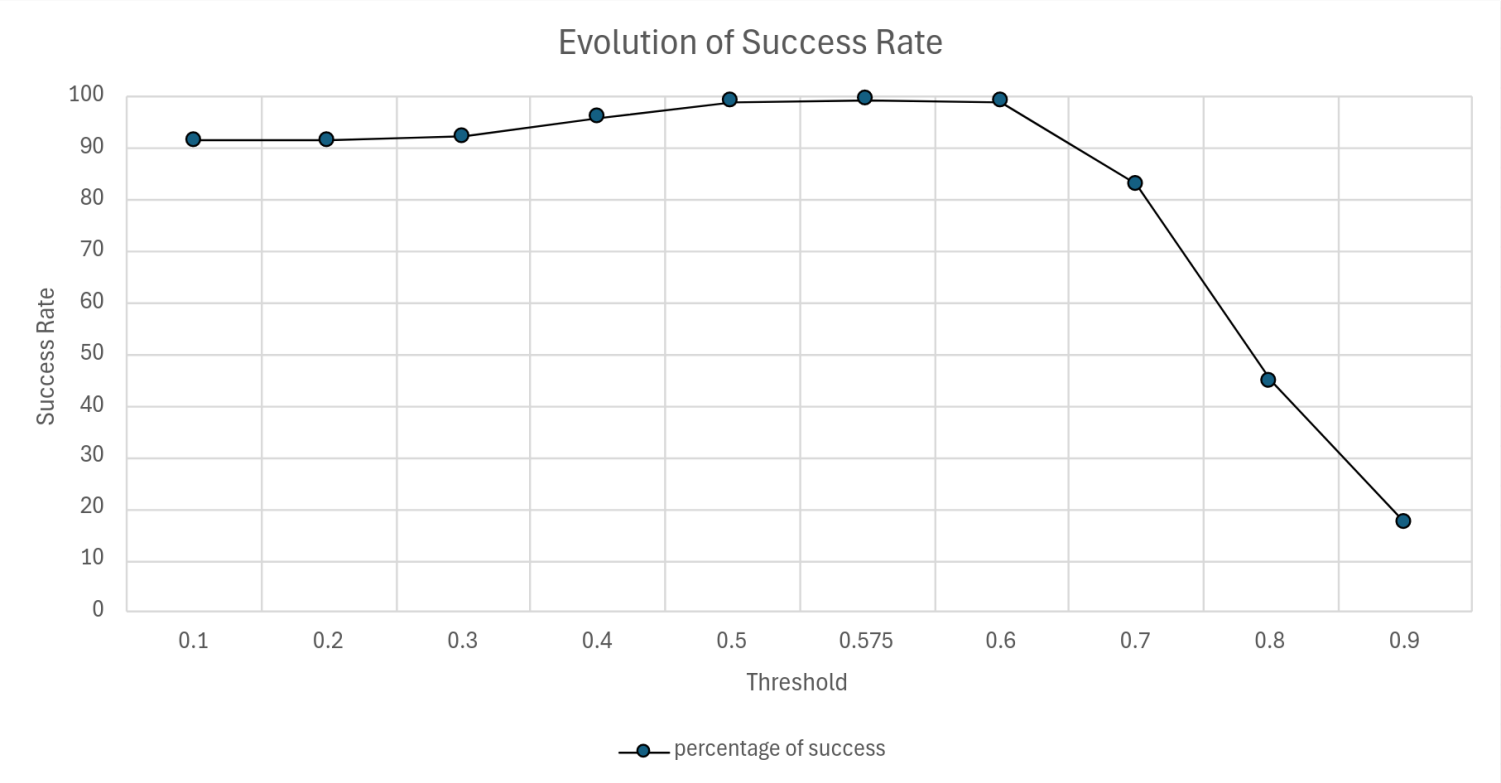

Figure 6: Courbe d'évolution du taux de succès par valeur seuil

Défis rencontrés

- Recherche de bibliothèques
- Documentations peu claires et complexes
- Calculs très consommateurs de ressources
- Difficultés pour intégrer les fonctions des bibliothèques à notre code

Mesures futures

- Potentiellement utilisable sur smartphone
- Solution sécurisée
- Processus lent à optimiser
- Scalabilité limitée
- Pas utilisable en l'état dans des grandes entreprises
- Il reste néanmoins possible de développer une solution plus rapide, fiable et sécurisée

Conclusion

- Un sujet très complexe permettant la protection des données
- De multiples applications dans la vraie vie
- Une preuve de concept fonctionnelle
- Chiffrement homomorphe

Sources i

- Démonstration de face-recognition : `https://pypi.org/project/face-recognition/`
- Logo Deepface : `https://pypi.org/project/deepface/`
- Logo Python : `https://fr.m.wikipedia.org/wiki/Fichier:Python-logo-notext.svg`

Références

- Y. Lindell, **« Secure multiparty computation, »** *Communications of the ACM*, t. 64, nº 1, p. 86-96, 2020.
- S. Dakhila, N. Ahmed et H. Shaari, **« Comparison of Two Face Recognition Machine Learning Models, »**, t. 21, p. 2022, oct. 2022. doi : 10.51984/JOPAS.V21I4.2120.
- D. Escudero, **« An Introduction to Secret-Sharing-Based Secure Multiparty Computation, »**, juin 2023.
- A. Patra, **« CSA E0 312 : Secure Computation : Lecture 6, »**, août 2015.
- M. Keller, **« MP-SPDZ : A Versatile Framework for Multi-Party Computation, »** in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. doi : 10.1145/3372297.3417872. adresse : https://doi.org/10.1145/3372297.3417872.
- ***Lattigo v5***, Online : https://github.com/tuneinsight/lattigo, EPFL-LDS, Tune Insight SA, nov. 2023.
- ***face_recognition***, Online : https://pypi.org/project/face-recognition/.

- ***deepface***, Online : https://pypi.org/project/deepface/.
- S. I. Serengil et A. Ozpinar, **« LightFace : A Hybrid Deep Face Recognition Framework, »** in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, IEEE, 2020, p. 23-27. doi : 10.1109/ASYU50717.2020.9259802. adresse : https://ieeexplore.ieee.org/document/9259802.
- ***MPyC***, Online : https://github.com/lschoe/mpyc/.
- *Wikipedia*, adresse : https://en.wikipedia.org/wiki/Secure_multi-party_computation.