IMPLEMENTATION DES TECHNIQUES DE CRYPTHOGRAPHIE OU DE SECURITE DES DONNEES EN PYTHON, APPPLICATION A UN ENSEMBLE DE DONNEES MEDICALES.

Par : NGELEKA MUSANGU Steve, DEA UNIKIN,2022-2023

I. INTRODUCTION GENERALE

L'implémentation des techniques de cryptographie ou de sécurité des données en Python est essentielle pour garantir la confidentialité, l'intégrité et la disponibilité des données sensibles, notamment dans le domaine médical. Dans ce travail, nous aborderons l'application de ces techniques à un ensemble de données médicales, mettant en lumière l'importance de protéger les informations personnelles des patients, telles que les identifiants, les antécédents médicaux, les médicaments prescrits et les allergies. La sécurisation de ces données est cruciale pour assurer la confiance des patients, se conformer aux réglementations en matière de protection des données telles que le RGPD, et prévenir les fuites ou les utilisations non autorisées des informations médicales. Au cours de cette exploration, nous examinerons comment utiliser Python pour mettre en œuvre des techniques de chiffrement, de hachage et d'autres mesures de sécurité afin de renforcer la confidentialité et l'intégrité des enregistrements médicaux.

Partant de ce qui précède la question suivante fait l'objet du présent travail :

Comment les techniques de cryptographie et de sécurité des données en Python peuvent-elles être implémentées de manière efficace pour garantir la confidentialité, l'intégrité et l'authenticité des données médicales dans le respect des normes de confidentialité telles que le RGPD (Règlement Général sur la Protection des Données) ?

Nous pensons que l'utilisation des techniques avancées de cryptographie et de sécurité des données en Python peut garantir un niveau élevé de confidentialité, d'intégrité et d'authenticité des données médicales, tout en se conformant aux normes strictes de protection des données telles que le RGPD.

II. REVUE DE LITTERATURE

Étienne ANDRÉ, Laure PETRUCCI, Université Paris 13, Sorbonne Paris Cité, **article intitulé : « La cryptographie décryptée »**

Ce travail met en lumière l'évolution historique de la cryptographie depuis ses origines anciennes jusqu'à ses applications modernes. En explorant les concepts fondamentaux tels que la cryptographie symétrique et asymétrique, les protocoles de sécurité essentiels comme SSL/TLS et PGP, ainsi que les avancées récentes dans le domaine quantique, cette synthèse souligne l'importance cruciale de la cryptographie dans la protection des données et des communications. Malgré les défis actuels, notamment liés à l'émergence de l'informatique quantique, les efforts de recherche continus visent à développer des techniques de cryptographie robustes pour garantir la sécurité et la confidentialité dans un environnement numérique en constante évolution.

Soufiane EL MOUMNI, Université Hassan II de Casablanca**, Thèse intitulée : « Implémentation à haute performance de la fonction de hachage cryptographique SHA3 sur des circuits FPGA »**

Les résultats de ce travail mettent en lumière l'importance de cette technologie dans la sécurisation des données. En examinant son état de l'art, on a constaté que diverses approches ont été explorées pour optimiser l'efficacité énergétique et la vitesse de traitement, notamment par le biais de techniques telles que la parallélisation, la pipelining et l'utilisation judicieuse des ressources FPGA. Les outils de développement et de validation jouent un rôle crucial dans la mise au point de ces implémentations, garantissant leur sécurité et leur intégrité. Dans ce travail, Les études de cas et les résultats expérimentaux ont montré des progrès significatifs en termes de débit, de latence et d'utilisation des ressources, bien que des défis persistent en matière d'optimisation et d'évolutivité. Les perspectives futures se concentrent sur l'amélioration continue de la sécurité, de la fiabilité et de l'efficacité énergétique des systèmes de hachage cryptographique sur FPGA, soulignant ainsi l'importance de la recherche continue dans ce domaine.

Mlle. SI SALEM Zahia, Mlle. TALHI Dalila, Université Mouloud MAMMERI de TIZIOUZOU, **Mémoire intitulé : « Implémentation d'une politique de sécurité par chiffrement avec les certificats dans une entreprise »**

Ce travail montre que la mise en œuvre d'une politique de sécurité par chiffrement avec les certificats dans une entreprise est cruciale pour garantir la confidentialité, l'intégrité et l'authenticité des données sensibles. Ce travail montre les fondements du chiffrement et des certificats SSL/TLS, met en lumière les normes et bonnes pratiques de sécurité, analyse les différentes méthodes de chiffrement symétriques et asymétriques, ainsi que le processus de gestion des certificats SSL/TLS. Il souligne l'importance d'une approche stratégique et systématique pour mettre en place une politique de sécurité efficace, en surmontant les défis liés à la conformité aux normes de sécurité et en s'appuyant sur des études de cas pour illustrer les bénéfices tangibles d'une telle implémentation.

En considérant tout ce qui précède, nous avons compris que l'implémentation des techniques de cryptographie ou de sécurité des données vise à résoudre les failles liées à la confidentialité, à l'intégrité, à l'authenticité et à la disponibilité des informations, en adoptant des mesures et des technologies robustes pour lutter contre les attaques et garantir un environnement numérique sécurisé, c'est qui résume l'apport du présent travail.

Partant de notre problématique, nous répondrons auxquelles que questions tirant leur source à notre sujet, parmi lesquelles nous avons retenus les suivantes :

1. Comment chiffrer les données sensibles telles que les conditions médicales et les médicaments dans le dataset en utilisant AES (Advanced Encryption Standard) avec une clé de 256 bits ?
2. Comment hasher les noms des patients pour garantir l'intégrité des données?
3. Comment masquer les dates de naissance des patients en utilisant une technique d'obfuscation?
4. Comment vérifier l'intégrité des données en utilisant un code HMAC?
5. Comment utiliser une technique de pseudonymisation pour anonymiser les noms des patients en remplaçant chaque nom par un identifiant unique ?

III. CODAGE EN PYTHON POUR LES CINQS QUESTIONS SOULEVEES CIHAUT

Pour chacune des questions, nous allons présenter les codes en python stockant les paramètres de la solution dans un dictionnaire.

III.1 Comment chiffrer les données sensibles telles que les conditions médicales et les médicaments dans le dataset en utilisant AES (Advanced Encryption Standard) avec une clé de 256 bits ?

```
[17]: %config InlineBackend.iopub_data_rate_limit = 1000000000
```

```
[7]: import pandas as pd
from cryptography.fernet import Fernet

# Charger le dataset
data = pd.read_csv("medical_records.csv")

# Générer une clé AES
key = Fernet.generate_key()
cipher_suite = Fernet(key)

# Chiffrer les données sensibles
data_to_encrypt = " ".join([row["medical_conditions"] + row["medications"] for index, row in data.iterrows()])
encrypted_data = cipher_suite.encrypt(data_to_encrypt.encode())

# Enregistrer les données chiffrées dans un fichier CSV
with open("encrypted_medical_records_Steve.csv", "wb") as f:
    f.write(encrypted_data)
```

Fig1. Solution de la première question

III.2 Comment hasher les noms des patients pour garantir l'intégrité des données?

```
[12]: import pandas as pd
import hashlib

# Charger le dataset
data = pd.read_csv("medical_records.csv")

# Hasher les noms des patients
hashed_names = [hashlib.sha256(row["name"].encode()).hexdigest() for index, row in data.iterrows()]

# Enregistrer le résultat du hashage dans une nouvelle colonne du DataFrame
data["hashed_name"] = hashed_names

# Enregistrer les données dans un fichier CSV
data.to_csv("hashed_medical_records.csv", index=False)

# Afficher les 5 premières lignes du fichier
print(data.head())
```

Fig2. Solution de la deuxième question

III.3 Comment masquer les dates de naissance des patients en utilisant une technique d'obfuscation?

```
[16]: import pandas as pd

# Charger le dataset
data = pd.read_csv("medical_records.csv")

# Obfuscation des dates de naissance
obfuscated_dates = [dob[:4] + "XXXX" for dob in data["date_of_birth"]]

# Ajouter les dates obfusquées comme une nouvelle colonne dans le DataFrame
data["obfuscated_date_of_birth"] = obfuscated_dates

# Enregistrer les données dans un fichier CSV
data.to_csv("obfuscated_medical_records.csv", index=False)

# Afficher les 5 premières lignes du fichier CSV
print(data.head())
```

Fig3. Solution de la troisième question

III.4 Comment vérifier l'intégrité des données en utilisant un code HMAC?

```
[20]: import pandas as pd
import hmac
import hashlib

# Charger le dataset
data = pd.read_csv("medical_records.csv")

# Générer une clé HMAC
key = b'secret_key'

# Calculer le HMAC pour chaque ligne de données
hmacs = [hmac.new(key, row["medical_conditions"].encode() + row["medications"].encode(), hashlib.sha256).hexdigest() for index, row in data.iterrows()]

# Ajouter les HMACs comme une nouvelle colonne dans le DataFrame
data["hmac"] = hmacs

# Enregistrer les données dans un fichier CSV
data.to_csv("medical_records_with_hmac.csv", index=False)
```

Fig4. Solution de la quatrième question

III.5 Comment utiliser une technique de pseudonymisation pour anonymiser les noms des patients en remplaçant chaque nom par un identifiant unique ?

```
[24]: import pandas as pd
      import hashlib

      # Charger le dataset
      data = pd.read_csv("medical_records.csv")

      # Générer un identifiant unique pour chaque patient en utilisant le hash de leur nom
      data['patient_id'] = [hashlib.sha256(str(name).encode()).hexdigest() for name in data['name']]

      # Supprimer les noms des patients originaux
      data.drop(columns=['name'], inplace=True)

      # Stocker les paramètres
      params = {
          "pseudonymization_technique": "SHA-256 Hash",
          "dataset_path": "medical_records.csv",
          "output_path": "medical_records_pseudonymized.csv"
      }

      # Enregistrer les données pseudonymisées dans un nouveau fichier CSV
      data.to_csv(params['output_path'], index=False)

      # Afficher les 5 premières lignes des données pseudonymisées
      print(data.head())

      # Ajouter les paramètres dans le dictionnaire
      params["first_5_rows"] = data.head().to_dict(orient="records")

      # Afficher les paramètres
      print(params)
```

Fig5. Solution de la cinquième question

IV. DISCUSSION/COMMENTAIRES SUR LES RESULTATS OBTENUS

Notre discussion va consister à expliquer les codes tout en mettant un accent sur les résultats, nous allons commenter séquentiellement, c'est-à-dire question après question.

IV.1 Commentaires sur les résultats de la première question

Les résultats obtenus sont prometteurs, avec succès le code a chiffré les données sensibles du dataset, incluant les conditions médicales et les médicaments, en utilisant la bibliothèque `cryptography` avec l'algorithme AES (Advanced Encryption Standard) et une clé robuste de 256 bits. Les données chiffrées ont été stockées dans un fichier CSV spécifique nommé **"encrypted_medical_records_Steve.csv"**, prêt à être utilisé tout en préservant la confidentialité et l'intégrité des données médicales.

Le code démarre en chargeant le dataset contenant les données à chiffrer, puis génère une clé AES de 256 bits et initialise un objet Fernet avec cette clé. Les données sensibles sont extraites du dataset et regroupées dans une chaîne unique « data_to_encrypt », qui est ensuite chiffrée à l'aide de la méthode « encrypt » de l'objet Fernet. Le résultat chiffré est enregistré dans un fichier CSV binaire nommé **"encrypted_medical_records_Steve.csv",** offrant ainsi une méthode sécurisée pour

stocker et manipuler les données médicales sensibles. Notons l'importance de sécuriser la clé de chiffrement pour garantir un déchiffrement réussi des données à l'avenir.

NOTA : Après exécution des codes, la sortie (cryptée) n'est pas visible dans l'environnement de programmation python pour la simple raison que nous utilisons un data set volumineux et que dans pareille situation la sortie ne pourra être visible que dans un fichier CSV qui sera créé automatiquement et nommé, exemple : **"encrypted_medical_records_Steve.csv"** comme est notre cas.

IV.2 Commentaires sur les résultats de la deuxième question

Le processus décrit consiste à garantir l'intégrité et la confidentialité des données médicales en hashant les noms des patients dans un dataset. Tout d'abord, le code charge les données médicales à partir d'un fichier CSV à l'aide de la bibliothèque pandas, permettant ainsi la manipulation efficace des données tabulaires. Ensuite, chaque nom de patient est passé par l'algorithme de hachage SHA256 à l'aide de la bibliothèque hashlib, ce qui crée une empreinte numérique unique et irréversible pour chaque nom. Ces empreintes, représentant les noms hashés des patients, sont ensuite ajoutées à une nouvelle colonne dans le DataFrame pandas.

Le résultat final est un dataset modifié où les noms des patients ont été remplacés par leurs équivalents hashés, assurant ainsi la confidentialité des données originales tout en garantissant l'intégrité des informations. Cette technique de hachage permet également de vérifier si les données ont été altérées, car tout changement dans les noms hashés indiquerait une altération des données. De plus, le processus de hashage utilise un algorithme cryptographique robuste, renforçant la sécurité des données médicales contre les accès non autorisés.

En enregistrant le dataset modifié dans un nouveau fichier CSV, nommé **"hashed_medical_records.csv",** vous disposez d'une version sécurisée des données médicales prête à être utilisée dans des analyses ou des applications sans compromettre la confidentialité des patients. Cette approche de hashage des données personnelles s'inscrit dans les bonnes pratiques de sécurité et de protection des données, offrant une solution efficace pour préserver l'intégrité et la confidentialité des données médicales sensibles.

IV.3 Commentaires sur les résultats de la troisième question

Le code fourni démontre une approche d'obfuscation des dates de naissance dans un dataset de dossiers médicaux. L'obfuscation est réalisée en remplaçant les quatre premiers caractères de chaque date de naissance par "XXXX", préservant ainsi partiellement la confidentialité des informations personnelles des patients. Cette technique est utile pour masquer des données sensibles tout en conservant une partie de leur utilité pour des analyses ultérieures, ce qui peut être nécessaire dans certains contextes sans compromettre la confidentialité des individus.

Cependant, il est important de noter que l'obfuscation seule n'est pas une mesure de sécurité complète et peut ne pas être suffisante pour garantir la protection des données sensibles dans tous les cas. Des approches supplémentaires, telles que le chiffrement, l'anonymisation ou d'autres techniques de protection des données, peuvent être nécessaires en fonction des exigences spécifiques de sécurité et de confidentialité. Il est également crucial de prendre en compte les réglementations et les normes de protection des données lors de la manipulation et du traitement d'informations sensibles telles que les données médicales pour assurer leur protection adéquate.

IV.4 Commentaires sur les résultats de la quatrième question

Le code fourni démontre comment calculer et utiliser des HMACs (Hash-based Message Authentication Codes) pour garantir l'intégrité des données dans un DataFrame pandas représentant des dossiers médicaux. Tout d'abord, le code charge le dataset à partir d'un fichier CSV et génère une clé HMAC statique. Ensuite, il calcule le HMAC pour chaque ligne de données en concaténant les valeurs des colonnes "medical_conditions" et "medications" pour chaque patient, puis en utilisant la clé HMAC et l'algorithme de hachage SHA-256 pour obtenir le HMAC correspondant. Les HMACs calculés sont ajoutés comme une nouvelle colonne dans le DataFrame et les données, y compris les HMACs, sont enregistrées dans un nouveau fichier CSV.

Pour vérifier l'intégrité des données, il suffit de recharger le fichier CSV contenant les HMACs calculés dans un nouveau DataFrame. Ensuite, en recalculant les HMACs pour chaque ligne en utilisant la même clé HMAC, on peut comparer les HMACs recalculés avec les HMACs déjà présents dans le DataFrame. Si les HMACs correspondent pour chaque ligne, cela signifie que les données n'ont pas été altérées depuis la génération des HMACs initiaux, offrant ainsi une vérification efficace de l'intégrité des données médicales.

NOTA : Après exécution des codes, la sortie (cryptée) n'est pas visible dans l'environnement de programmation python pour la simple raison que nous utilisons un data set volumineux et que dans pareille situation la sortie ne pourra être visible que dans un fichier CSV qui sera créé automatiquement et nommé, exemple : **" medical_records_with_hmac"** comme est notre cas.

IV.5 Commentaires sur les résultats de la cinquième question

Le code fourni illustre une méthode de pseudonymisation des données en remplaçant les noms des patients par des identifiants uniques générés à l'aide de la fonction de hachage SHA-256. Cette technique assure l'anonymat des données en ne conservant que des valeurs non identifiables, ce qui est essentiel pour protéger la vie privée des individus dans le domaine médical. En outre, le code inclut des étapes pour supprimer les noms originaux du dataset et enregistrer les données pseudonymisées dans un nouveau fichier CSV, facilitant ainsi leur utilisation pour des analyses tout en respectant les normes de confidentialité.

Le processus de pseudonymisation utilisé, basé sur le hachage cryptographique, garantit la nonrécupération des noms originaux à partir des identifiants uniques générés, ce qui renforce la sécurité et la confidentialité des données médicales. De plus, la conservation des paramètres de pseudonymisation dans un dictionnaire permet une traçabilité et une documentation claires du processus, ce qui est essentiel pour assurer la transparence et la conformité aux réglementations sur la protection des données.

V. CONCLUSION

En utilisant des techniques telles que le chiffrement AES, le hachage, l'obfuscation, le HMAC et la pseudonymisation, il est possible de sécuriser efficacement un ensemble de données médicales. Le chiffrement protège les données sensibles, le hachage garantit leur intégrité, l'obfuscation masque les informations sensibles, le HMAC vérifie l'intégrité des données et la pseudonymisation anonymise les informations personnelles des patients. Ces techniques combinées contribuent à renforcer la confidentialité, l'intégrité et la sécurité des données médicales.

NOTA : Chaque exécution des codes pour chacune des questions est stockée dans un fichier crypté précis, ce fichier est généré automatiquement après exécution et a la même extension avec le data set. Notre data set porte le nom, **medical_records.**

TABLE DES MATIERES

I. INTRODUCTION GENERALE 1
II. REVUE DE LITTERATURE 1

III. CODAGE EN PYTHON POUR LES CINQS QUESTIONS SOULEVEES CIHAUT 3
III.1 Comment chiffrer les données sensibles telles que les conditions médicales et les médicaments dans le dataset en utilisant AES (Advanced Encryption Standard) avec une clé de 256 bits ? 3
III.2 Comment hasher les noms des patients pour garantir l'intégrité des données? 3
III.3 Comment masquer les dates de naissance des patients en utilisant une technique d'obfuscation? 4
III.4 Comment vérifier l'intégrité des données en utilisant un code HMAC? 4
III.5 Comment utiliser une technique de pseudonymisation pour anonymiser les noms des patients en remplaçant chaque nom par un identifiant unique ? 5
IV. DISCUSSION/COMMENTAIRES SUR LES RESULTATS OBTENUS 5
IV.1 Commentaires sur les résultats de la première question 5
IV.2 Commentaires sur les résultats de la deuxième question 6
IV.3 Commentaires sur les résultats de la troisième question 6
IV.4 Commentaires sur les résultats de la quatrième question 7
IV.5 Commentaires sur les résultats de la cinquième question 7
V. CONCLUSION 8