1. Sécurité du protocole :

— Pourquoi utilise-t-on un challenge aléatoire plutôt qu'une valeur fixe?

Raisons principales :

-Protection contre les attaques par rejeu : Si le challenge était fixe, un attaquant pourrait capturer une réponse valide et la rejouer indéfiniment pour s'authentifier.

-Fraîcheur de l'authentification : Le challenge aléatoire garantit que chaque session d'authentification est unique et récente.

- Prévention de la pré-calcul : Avec un challenge aléatoire, un attaquant ne peut pas précalculer des réponses valides à l'avance.

2. Gestion des clés :

Pourquoi dérive-t-on une clé de session KE plutôt que d'utiliser directement k ?

Séparation des usages (Key Separation)

- La clé k est une clé à long terme, utilisée exclusivement pour l'authentification.
- La clé K_E est une clé éphémère, générée à chaque session pour le chiffrement des données.
- Ainsi, même si K_E est compromise, la clé principale k reste protégée.

Confidentialité persistante (Perfect Forward Secrecy, PFS)

- Chaque session dispose d'une clé unique dérivée selon la formule :
  K_E = HKDF(k, challenge, "GSM-KE")
- Si un attaquant parvient à obtenir la clé K_E d'une session, les autres sessions restent sécurisées.
- Même si la clé principale k est compromise plus tard, les sessions précédentes demeurent protégées, car K_E dépend également d'un défi unique qui n'est pas conservé.

3. Attaques possibles :

Que se passe-t-il si un attaquant intercepte tous les messages ?

Même s'il intercepte toutes les communications, l'attaquant ne peut ni déchiffrer les messages, ni usurper l'identité de l'utilisateur, ni rejouer l'authentification.
Il peut seulement observer le trafic (quand et combien de données sont échangées) et collecter des informations publiques, sans pouvoir exploiter la clé secrète.

Comment un attaquant pourrait-il compromettre le système ?

Vol ou piratage de la carte SIM

Si quelqu'un récupère physiquement la SIM et parvient à en extraire la clé k, il peut s'authentifier comme l'utilisateur légitime. C'est la menace la plus directe.
Mitigation : protections matérielles (puce résistante au tampering).

Compromission de la base opérateur

Si l'attaquant a accès au fichier qui stocke les clés (operator_db.csv), il obtient toutes les clés des abonnés et peut usurper n'importe qui.
Mitigation : ne pas stocker k en clair — chiffrer la base ou utiliser un HSM/KMS.

Failles d'implémentation

Un simple bug (mauvaise comparaison, fuite de mémoire) ou une attaque par canal auxiliaire (timing, power) peut révéler des informations sensibles. Par exemple, utiliser == au lieu de hmac.compare_digest ouvre la porte aux attaques par timing.
Mitigation : revue de code, tests, API résistante aux side-channels.

Annexe

Annexe – Utilisation d'une IA générative (Exercice 4 : Mini-GSM)

Outil utilisé : ChatGPT (modèle GPT-5)
Objectif : assistance à la compréhension et à la correction du code Python du projet *Mini-GSM* (fichiers : sim_card.py, network_operator.py, gsm_simulation.py, main.py).

---

1. Prompts envoyés à l'IA

(Extraits représentatifs des échanges utilisés pour la résolution de l'exercice)

- **« Je suis perdu, dis-moi le code comment régler les problème des emojis. »**
- **« Regarde ça » *(capture d'écran d'erreurs d'exécution)***
- **« Je suis perdu, explique-moi les erreurs et comment les corriger »**
- **« Résume ce texte : Que se passe-t-il si un attaquant intercepte tous les messages ? »**
- **« Comment un attaquant pourrait-il compromettre le système ? »**

---

2. Réponses principales de l'IA (résumé synthétique)

a) Corrections du code

L'IA a fourni des corrections pour :

- **network_operator.py :**
    - **Chargement correct du fichier CSV (csv.DictReader), gestion du champ k_hex ou k.**
    - **Génération du challenge avec secrets.token_bytes(16).**
    - **Vérification sécurisée de la réponse avec hmac.compare_digest.**
    - **Dérivation de la clé de session :**
    - **HKDF(algorithm=hashes.SHA256(), length=32, salt=challenge, info=b"GSM-KE")**
- **sim_card.py :**
    - **Correction du paramètre length de HKDF (32 octets).**
    - **Fermeture correcte des fichiers JSON.**
    - **HMAC-SHA256 pour la réponse.**
- **main.py :**
    - **Suppression des emojis et gestion de l'encodage CP1252 sous Windows pour éviter UnicodeEncodeError.**

- **gsm_simulation.py :**
    - **Ajout du chiffrement/déchiffrement avec AES-GCM.**

Ces modifications ont permis le bon déroulement des tests automatiques et la génération correcte des clés de session.

---

b) Explications conceptuelles générées

L'IA a aussi fourni des justifications écrites insérées dans le rapport :

- **Raisons de sécurité essentielles :**
    - **Séparation des clés (k long terme / K_E session).**
    - **Perfect Forward Secrecy : une clé de session unique par échange.**
- **Ce qu'un attaquant peut ou ne peut pas faire :**
    - **Peut observer IMSI, challenge, réponse et messages chiffrés.**
    - **Ne peut pas déchiffrer sans k ni rejouer une ancienne réponse.**
- **Attaques possibles et contre-mesures :**
    - **Vol de la SIM → extraction de k.**
    - **Piratage de la base opérateur → exposition de toutes les clés.**
    - **Failles d'implémentation (timing attacks).**
    - **Générateur aléatoire faible → rejeu possible.**
    - **MITM possible (pas d'authentification du réseau).**
    - **Force brute sur k théoriquement possible mais infaisable.**

---

4. Usage et respect des consignes

- **Les suggestions de code ont été testées localement, adaptées et documentées.**
- **Les explications textuelles ont été réécrites dans un style personnel avant insertion.**
- **Les extraits bruts de l'IA sont placés ici (en annexe) pour assurer la transparence académique demandée par l'énoncé.**