REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie

UNIVERSITE DE YAOUNDE I

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE DE YAOUNDE

DEPARTEMENT DE GENIE INFORMATIQUE

REPUBLIC OF CAMEROON
Peace – Work – Fatherland

UNIVERSITY OF YAOUNDE I

NATIONAL ADVANCED SCHOOL OF ENGINEERING OF YAOUNDE

DEPARTMENT OF COMPUTER SCIENCE

Exercices du cours : pages 13 - 16

Participant

Matricule : 22P035

Spécialité : Cybersécurité et Investigation Numérique

Noms : MBETSI DJOFANG AIME LINDSEY

Niveau : 4

Superviseur

M. MINKA MI NGUIDJOI Thierry Emmanuel

Année Scolaire : 2025–2026

Exercices pages 13 à 16

Exercice 1 — Dissertation (paradoxe de la transparence)

Le paradoxe de la transparence décrit par Byung-Chul Han dit que, dans nos sociétés numériques, nous voulons tout savoir (transparence) mais en même temps nous voulons garder notre intimité. Cette situation crée une tension : plus on donne d'informations publiques, plus la vie privée s'amenuise ; mais si on protège trop la vie privée, la société peut perdre des outils de contrôle utiles (par exemple pour détecter la corruption).

Appliqué à une enquête numérique, ce paradoxe se traduit ainsi : un État ou un enquêteur peut réclamer l'accès aux données (logs, messages, géolocalisation) pour prouver un crime et protéger les citoyens. Mais ouvrir l'accès de façon large menace la vie privée et peut être détourné (surveillance massive, profilage). L'enquête trouve la vérité mais expose fortement les personnes innocentes.

Cas concret : imaginez une fuite de fonds publics. Pour enquêter, l'administration demande les historiques de transactions et les conversations professionnelles. La transparence permet d'identifier des responsables. Mais si ces données sont rendues publiques ou utilisées sans garde-fous, elles révèlent aussi des informations sensibles (santé, opinions politiques) sur des tiers qui n'ont rien à voir. Voilà le paradoxe : la même transparence qui sert la justice met en péril l'intimité.

Solution pratique inspirée de l'éthique kantienne (langage simple) : Kant demande à agir selon des principes universels — traiter les personnes comme des fins, jamais seulement comme des moyens. Appliqué ici :

1. **Principe de finalité restreinte** : l'accès aux données est permis **seulement** pour une finalité définie (ex. enquête X), pas pour n'importe quel usage.

2. **Principe d'universalité procédural** : il faudrait des procédures identiques et publiques (qui peut demander, selon quel seuil de preuve, pour combien de temps). Ces règles seraient applicables à tous et connues de tous.

3. **Principe du respect de la dignité** : chaque accès doit minimiser l'exposition des données non pertinentes (redaction, anonymisation) ; les données sensibles doivent rester protégées.

Mise en œuvre concrète : mise en place d'un filtre légal (autorisation judiciaire), de techniques techniques (anonymisation, ZK-proofs pour prouver sans révéler les données complètes), et d'un contrôle externe (audit indépendant). Ainsi on concilie l'obligation morale d'obtenir la vérité et le respect de la personne : la transparence nécessaire à l'enquête devient circonscrite par des principes universels (comme Kant le demanderait), ce qui réduit le risque d'abus.

En résumé : la transparence aide la justice, mais elle doit être encadrée par des règles universelles et des techniques qui protègent l'intimité. C'est la voie kantienne adaptée au numérique : agir selon des règles qui respectent la dignité humaine, même dans la recherche de la vérité.

Exercice 2 — Transformation ontologique (Heidegger → numérique) — langage simple

Comparer Heidegger / adaptation numérique (points clés simples)

- Heidegger : l'homme est « être-au-monde » — l'existence se mesure par nos actions et notre présence.
- Numérique : aujourd'hui, l'identité inclut aussi un double numérique (profils, historiques, traces). L'être n'est plus seulement physique, il est aussi « être-par-ses-traces ».

Étude d'un profil social comme « être-par-la-trace » (exemple simple)

- Profil Facebook/Twitter = collection de posts, likes, photos, commentaires. Ces traces montrent habitudes, opinions, relations. Même si la personne n'est pas présente physiquement, son profil « existe » et influence la façon dont d'autres la perçoivent. On peut dire que l'individu « est » en partie par ses traces.

Impact sur la preuve légale (langage simple)

- Avantage : ces traces peuvent prouver des faits (horodatage, localisation).
- Risque : elles peuvent être manipulées (deepfakes) ou sorties de leur contexte. L'interprétation devient critique : la preuve numérique est persuasive mais nécessite une méthode rigoureuse (chaîne de custody, corroboration, métadonnées).

Exercice 3 — Calcul d'entropie (script Python simple + seuils)

Idée simple : l'entropie mesure l'imprévisibilité des octets. Un texte clair a une entropie faible ; un fichier chiffré a une entropie proche de 8 bits/octet.

Formule (rappel simple)

$$H = -\sum_{i=0}^{255} p(i) \log_2 p(i)$$

Script Python minimal (exemple)

```
# calc_entropy.py
import math
from collections import Counter

def entropy_bytes(path):
with open(path,'rb') as f:
data = f.read()
if not data:
return 0.0
counts = Counter(data)
n = len(data)
H = 0.0
for c in counts.values():
p = c / n
```

```
H -= p * math.log2(p)
return H  # bits per byte

# Usage:
# print(entropy_bytes('document.txt'))
# print(entropy_bytes('image.jpg'))
# print(entropy_bytes('cipher.aes'))
```

Interprétation simple et seuil proposé

- Texte naturel : $\approx$ 1.0–4.0 bits/octet.
- JPEG : $\approx$ 7.0–7.5 bits/octet.
- AES chiffré : $\approx$ 7.9–8.0 bits/octet.

Seuil pratique proposé :

- Si $H \geq 7.6$ bits/octet $\rightarrow$ probablement chiffré ou compressé.

Exercice 4 — Théorie des graphes appliquée aux communications téléphoniques

Exemple simple avec 5 personnes : A, B, C, D, E.

- Arêtes : AB, AC, BC, BD, CE, DE, BE.
- Degré : A=2, B=4, C=3, D=2, E=3.
- Centralité : $CD(A) = 0.5$, $CD(B) = 1$, $CD(C) = 0.75$, $CD(D) = 0.5$, $CD(E) = 0.75$.

Betweenness : B est critique car il relie souvent A est équivalent à D, A est équivalent à E.
Conclusion : B est le nœud le plus central, à surveiller en priorité.

Exercice 5 — Modélisation de l'effet papillon

Méthode :

1. Construire timeline T0 avec 1000 événements.
2. Copier en T1 et modifier un timestamp de $\pm$30s.
3. Recalculer dépendances.
4. Mesurer divergence $\delta(t)$.
5. Estimer exposant de Lyapunov : $\delta(t) \approx \delta(0)e^{\lambda t}$.

Exemple : $\delta(0) = 1$, après 3600s $\delta(3600) = 100$.

$$\lambda \approx \frac{1}{3600} \ln(100) \approx 0.00128 \text{ par seconde.}$$

Exercice 6 — Chat de Schrödinger adapté

Idée : un fichier peut être « présent/effacé » tant qu'il n'est pas figé par une image forensique. Donc l'analyse elle-même modifie parfois l'état.

Protocole pour limiter :

- Isoler la machine.
- Utiliser write-blocker.
- Faire image bit-à-bit.
- Calculer hash.
- Travailler sur copies.

Exercice 7 — Calculs sur la sphère de Bloch

État :

$$|\psi\rangle = \cos\frac{\pi}{6}|0\rangle + e^{i\pi/4}\sin\frac{\pi}{6}|1\rangle$$

Calculs :

- $\cos(\pi/6) = \sqrt{3}/2 \approx 0.866$.
- $\sin(\pi/6) = 1/2 = 0.5$.
- $P(0) = 0.75$, $P(1) = 0.25$.

Exercice 8 — Théorème de non-clonage

Principe : on ne peut pas copier un état quantique inconnu parfaitement (contradiction avec linéarité).

Conséquence : en forensique quantique, il faut prouver l'existence d'un état par des preuves statistiques, pas en le copiant.

Exercice 9 — Formalisation mathématique du paradoxe

Trois systèmes :

- Système 1 : A=0.95, C=0.10 $\rightarrow$ $AC = 0.095$, $\delta = 0.905$.
- Système 2 : A=0.60, C=0.95 $\rightarrow$ $AC = 0.57$, $\delta = 0.43$.
- Système 3 : A=0.75, C=0.60 $\rightarrow$ $AC = 0.45$, $\delta = 0.55$.

Estimation $\hbar_{num}$: Exemple : $\Delta A = 0.02$, $\Delta C = 0.03$, donc

$$\hbar_{num} \approx 2 \cdot \Delta A \cdot \Delta C = 0.0012.$$

Exercice 10 — Proof-of-concept ZK-NR

Idée : prouver qu'on connaît un document sans le révéler, et signer l'engagement.

```
import hashlib, time

def sha(x): return hashlib.sha256(x).hexdigest()

# Prover side
D = b"document bytes"
h = sha(D)
r = b"random nonce"
commit = sha((h + r))
signature = sha((commit + b"private_key_sim"))

# Verifier side
ok = (signature == sha((commit + b"private_key_sim")))
```

Overhead : calcul du hash + signature.