Devoir 5

IFT814 – Cryptographie

Chargé de cours : Martin Fiset

Groupe 1

Nom	CIP
Mamadou Senghor	senm1912
Clément Dumas	dumc4449
Rafael Dhaene	dhar2976
Ricardo Beaujour	bear4830

Université de Sherbrooke
Département d'informatique

Automne 2025

UNIVERSITÉ DE SHERBROOKE

Date de remise : 28 Novembre 2025

Table des matières

Résumé

Ce rapport étudie la construction de trois primitives cryptographiques fondamentales en s'appuyant sur des oracles considérés comme des boîtes noires. Nous montrons d'abord comment implémenter un schéma de *bit commitment* (BC) à partir de la primitive de transfert à l'aveugle $\frac{1}{2}$-OT. Nous proposons ensuite une construction de $\frac{1}{2}$-OT à partir du Rabin OT (R-OT) en utilisant un schéma de partage de secret à seuil, garantissant simultanément correction et confidentialité. Enfin, nous montrons comment obtenir un protocole de *strong coin tossing* (S-CT) à partir du BC en utilisant le protocole classique de Blum. Pour chaque construction, nous décrivons le protocole et présentons les principaux arguments de sécurité.

1 Question 2 : Bit Commitment à partir de $\frac{1}{2}$-OT

1.1 Rappel des propriétés requises

Un schéma de *bit commitment* (BC) doit satisfaire deux propriétés fondamentales :

— **Hiding** : Après la phase de commitment, Bob ne peut apprendre aucune information sur le bit b engagé. La probabilité qu'il devine b doit être au plus $\frac{1}{2} + \text{negl}(n)$.

— **Binding** : Après la phase de commitment, Alice ne peut plus changer le bit engagé. Elle ne peut ouvrir qu'à un seul bit de manière cohérente.

Dans un $\frac{1}{2}$-OT, Alice envoie deux messages (x_0, x_1) et Bob, en choisissant un bit secret c, apprend x_c mais rien sur x_{1-c}, tandis qu'Alice ne reçoit aucune information sur c.

1.2 Construction du protocole

Idée principale : Alice utilise n *instances indépendantes* de $\frac{1}{2}$-OT. Dans chaque instance, elle envoie un couple $(x_{i,0}, x_{i,1})$ de chaînes aléatoires. Bob choisit un bit secret c_i et apprend uniquement x_{i,c_i}. Pour s'engager sur un bit b, Alice envoie un XOR qui combine toutes les valeurs et le bit b.

1.2.1 Description du protocole

Paramètre de sécurité Choisir un entier n (par ex. $n = 128$). Pour chaque $i \in \{1, \ldots, n\}$, une instance de $\frac{1}{2}$-OT sera exécutée.

Algorithm 1 Bit Commitment à partir de n instances de $\frac{1}{2}$-OT

Phase de Commitment :
Alice veut s'engager sur $b \in \{0,1\}$.
Pour $i = 1$ à n :
    Génère deux chaînes aléatoires :

$$x_{i,0} \leftarrow \{0,1\}^k, \qquad x_{i,1} \leftarrow \{0,1\}^k$$

    Alice fournit $(x_{i,0}, x_{i,1})$ à la i-ème instance du $\frac{1}{2}$-OT.
    Bob choisit un bit secret $c_i \in \{0,1\}$ et reçoit x_{i,c_i}.
Après les n OT, Alice calcule :

$$d \;=\; b \;\oplus \bigoplus_{i=1}^{n} H(x_{i,0} \parallel x_{i,1})$$

    où H est une fonction de hachage à sortie 1 bit.
Elle envoie **le commitment** : d.
Phase de Révélation (Opening) :
Alice révèle : $(b, \{x_{i,0}, x_{i,1}\}_{i=1}^{n})$.
Bob vérifie pour chaque i :

$$x_{i,c_i} \text{ correspond exactement à ce qu'il a reçu dans l'OT}$$

Bob recalcule :

$$d' = b \oplus \bigoplus_{i=1}^{n} H(x_{i,0} \parallel x_{i,1})$$

Si $d' = d$ **alors** il accepte b, **sinon** il rejette.

1.3 Arguments de sécurité

[Hiding] Le protocole satisfait la propriété de hiding avec une sécurité de $1 - 2^{-n}$.

Preuve : Bob reçoit seulement x_{i,c_i} dans chaque OT. Comme $x_{i,0}$ et $x_{i,1}$ sont indépendants, uniformes et secrets, Bob ne peut pas calculer :

$$H(x_{i,0} \parallel x_{i,1})$$

car il manque toujours une des deux valeurs.

Il obtient seulement :

$$d = b \oplus \Big(\text{une valeur uniformément aléatoire de } \{0,1\}\Big)$$

Donc d est uniformément distribué, *indépendamment* de b. Bob ne peut faire mieux que deviner b avec probabilité $1/2$.

[Binding] Le protocole satisfait la propriété de binding avec probabilité de triche $\leq 2^{-n}$.

Preuve : Pour ouvrir à un bit différent $b' \neq b$, Alice devrait produire une autre collection

$$\{x'_{i,0}, x'_{i,1}\}_{i=1}^{n}$$

telle que :

$$H(x_{i,0} \parallel x_{i,1}) \neq H(x'_{i,0} \parallel x'_{i,1})$$

pour suffisamment de i afin de changer le XOR global

Mais elle doit également satisfaire :

$$x'_{i,c_i} = x_{i,c_i}$$

pour *tous* les i (car Bob vérifie ce qu'il a reçu avec OT).

Or, Alice ne connaît pas les bits c_i, qui sont secrets pour Bob. Elle doit donc satisfaire simultanément :

une contrainte sur x_{i,c_i}, et réussir à changer la valeur globale du XOR.

La seule façon de tricher est de deviner parfaitement tous les c_i. La probabilité est :

$$\Pr[\text{Alice devine tous les } c_i] = 2^{-n}$$

qui est négligeable pour n grand.

1.4 Remarques

— Cette construction est la version **canonique** de BC $\Leftarrow$ 1/2-OT vue dans les cours et les manuels (Goldreich, Katz–Lindell).
— Le hachage H permet d'obtenir un seul bit à combiner via XOR.
— Le paramètre de sécurité n (nombre de OT) contrôle directement la sécurité du schéma.
— Le protocole est information-théoriquement sûr dans le modèle de l'OT parfait.

2 Question 3(a) : Construire un $\frac{1}{2}$-OT à partir du Rabin OT

2.1 Rappel des primitives

Rabin OT (R-OT) :
— Alice possède un message x.
— Bob reçoit x avec probabilité $\frac{1}{2}$, sinon il reçoit $\perp$.
— Alice ne peut pas savoir si Bob a reçu ou non.

$\frac{1}{2}$-OT :
— Alice possède deux messages (m_0, m_1).
— Bob possède un bit de choix $c \in \{0, 1\}$.
— À la fin du protocole, Bob doit obtenir m_c mais aucune information sur m_{1-c}.
— Alice ne doit rien apprendre sur le choix c.

2.2 Idée générale

La construction repose sur une intuition essentielle (Crépeau, 1988) :
— Alice partage chacun des messages m_0 et m_1 en k parts indépendantes via un schéma de partage de secret à seuil (t, k).
— Chaque part est ensuite envoyée via une instance indépendante de Rabin OT.

— Bob reçoit en moyenne environ $k/2$ parts de chaque côté.
— En fixant le seuil $t = \lceil k/2 \rceil + 1$, Bob :
  — peut reconstruire m_c (il reçoit suffisamment de parts du côté choisi),
  — ne peut pas reconstruire m_{1-c} (moins de t parts).

Comme Alice ne reçoit aucun retour de la part de Bob dans un R-OT, elle ignore quelles parts ont été obtenues, et donc ne peut rien apprendre sur c.

2.3 Protocole

Paramètres :

$$t = \left\lceil \frac{k}{2} \right\rceil + 1, \qquad k \text{ suffisamment grand (ex : } k = 128).$$

Algorithm 2 $\frac{1}{2}$-OT construit à partir de R-OT

Entrées :
Alice : deux messages (m_0, m_1)
Bob : un choix $c \in \{0, 1\}$

Phase 1 : Partage des secrets par Alice
Alice calcule deux partages à seuil (t, k) :

$$\{p_{0,1}, \ldots, p_{0,k}\} = \text{Share}(m_0), \qquad \{p_{1,1}, \ldots, p_{1,k}\} = \text{Share}(m_1).$$

Phase 2 : Envoi des parts via R-OT
for $i = 1$ to k **do**
    Alice exécute R-OT pour envoyer $p_{0,i}$.
    Bob reçoit $s_{0,i} \in \{p_{0,i}, \perp\}$.
    Alice exécute une autre R-OT pour envoyer $p_{1,i}$.
    Bob reçoit $s_{1,i} \in \{p_{1,i}, \perp\}$.
end for

Phase 3 : Reconstruction du message voulu
Bob définit :

$$S_0 = \{p_{0,i} : s_{0,i} \neq \perp\}, \qquad S_1 = \{p_{1,i} : s_{1,i} \neq \perp\}.$$

Si $|S_c| \geq t$:

$$m_c = \text{Reconstruct}(S_c).$$

Sinon :
    Bob déclare un échec (événement de probabilité négligeable).

2.4 Arguments de sécurité

[Correction] Avec probabilité $1 - e^{-\Omega(k)}$, Bob peut reconstruire m_c.

Preuve. Chaque part $p_{c,i}$ est reçue indépendamment avec probabilité $1/2$. Ainsi :

$$|S_c| \sim \text{Binomial}(k, 1/2).$$

Or l'espérance vaut $k/2$. Par l'inégalité de Chernoff :

$$\Pr\Big(|S_c| < t\Big) = \Pr\left(|S_c| < \frac{k}{2} + 1\right) \leq e^{-\Omega(k)}.$$

Donc Bob reçoit au moins t parts avec probabilité écrasante et peut reconstruire m_c.

[Confidentialité pour Bob] Alice n'apprend aucune information sur le choix c.

Preuve. Chaque R-OT ne fournit aucun retour à Alice. Elle ne sait donc jamais si Bob a reçu la part envoyée. La distribution de ce que voit Alice est identique pour $c = 0$ ou $c = 1$.

[Confidentialité pour Alice] Bob n'obtient aucune information sur m_{1-c}.

Preuve. Avec probabilité $1 - e^{-\Omega(k)}$, Bob reçoit strictement moins de t parts du mauvais côté. Or un schéma de partage à seuil (t, k) garantit que toute collection de moins de t parts révèle *zéro information* sur le secret. Donc m_{1-c} reste parfaitement caché.

2.5 Remarques

— Cette construction est due à Crépeau (1988) et constitue la transformation standard de R-OT vers $\frac{1}{2}$-OT.
— La sécurité est *information-théorique* si le partage est information-théorique.
— Le paramètre k peut être choisi tel que la probabilité d'échec soit négligeable.

3 Question 4 : Strong Coin Tossing à partir de Bit Commitment

3.1 Objectif et propriétés requises

On souhaite construire un protocole de *strong coin tossing* (S-CT) entre Alice et Bob à partir d'une primitive de **bit commitment** (BC). Le protocole doit permettre aux deux parties d'obtenir un bit commun aléatoire $r \in \{0, 1\}$, avec les garanties suivantes :

— **Impartialité** : si au moins une partie est honnête, le résultat r est uniformément distribué.
— **Imprévisibilité** : aucune partie ne peut prédire le résultat avant la fin du protocole.
— **Résistance au biais** : aucune partie ne peut faire en sorte que r prenne une valeur particulière avec probabilité significativement supérieure à $\frac{1}{2}$.
— **Accord** : les deux parties sortent le même bit.

Le bit commitment fournit exactement les deux propriétés nécessaires :

— **Hiding** : Bob ne peut pas apprendre le bit committé avant l'ouverture.
— **Binding** : Alice ne peut pas changer son bit après s'être engagée.

3.2 Protocole (version de Blum)

Le protocole standard utilise un engagement de la part d'Alice, suivi d'un bit envoyé par Bob.

Algorithm 3 Strong Coin Tossing à partir d'un Bit Commitment

Phase 1 : Commitment d'Alice
Alice choisit $a \leftarrow \{0,1\}$ de façon uniforme.
Elle calcule $c = \text{Commit}(a)$ et envoie c à Bob.

Phase 2 : Choix de Bob
Bob choisit $b \leftarrow \{0,1\}$ uniformément et envoie b à Alice.

Phase 3 : Ouverture du commitment
Alice révèle (a, preuve).
Bob vérifie $\text{Verify}(c, a, \text{preuve})$. S'il échoue, il rejette.

Phase 4 : Résultat
Les deux parties calculent : $r = a \oplus b$.
Sortie : le bit r.

C'est le protocole classique de Blum, utilisé comme exemple canonique de S-CT.

3.3 Analyse de sécurité

Nous montrons que ce protocole est un *strong coin tossing*.

3.3.1 Impartialité

Si au moins une des deux parties est honnête, le bit final r est uniformément distribué.

Preuve. Considérons les deux cas.

1. Alice est honnête. Elle choisit a uniformément aléatoire. Quel que soit le choix (malveillant) de Bob,

$$r = a \oplus b \text{ est uniforme, car } a \text{ est uniforme.}$$

2. Bob est honnête. Il choisit b uniformément. Même si Alice est malveillante, elle est liée à un bit fixe a par le *binding* du commitment. Ainsi :

$$r = a \oplus b \text{ est uniforme, car } b \text{ est uniforme.}$$

Dans les deux cas, r est distribué uniformément sur $\{0,1\}$.

3.3.2 Confidentialité (hiding)

Bob ne peut pas apprendre a avant d'avoir envoyé b.

Preuve. Par la propriété *hiding* du bit commitment, le commitment c ne révèle aucune information sur a. Bob choisit donc b sans connaître a, ce qui empêche tout biais.

3.3.3 Intégrité (binding)

Alice ne peut pas changer son bit après avoir vu le choix de Bob.

Preuve. Le bit a est fixé au moment de l'engagement. Par la propriété *binding*, Alice ne peut ouvrir c qu'avec la valeur originale. Elle ne peut donc pas adapter a en fonction de b pour biaiser le résultat.

3.3.4 Résistance au biais

Aucune partie ne peut forcer un résultat avec probabilité supérieure à $\frac{1}{2}$ + négligeable.

Preuve.

Alice tricheuse : elle voudrait choisir a' après avoir vu b, mais ne peut pas à cause du *binding*. Elle obtient donc son résultat favori avec probabilité exactement $\frac{1}{2}$.

Bob tricheur : il voudrait choisir b de façon adaptée à a, mais ne peut pas apprendre a avant d'envoyer son bit (propriété *hiding*). Il réussit donc avec probabilité $\frac{1}{2}$.

Dans les deux cas, l'avantage sur $\frac{1}{2}$ est négligeable.

3.4 Comportements hors-protocole : abandon

Une partie malveillante ne peut pas biaiser le résultat, mais elle peut *abandonner* le protocole si le résultat ne lui convient pas. Ceci ne brise pas la sécurité du S-CT, mais empêche la terminaison (déni de service).

Dans le modèle du cours, aucune solution technique supplémentaire n'est exigée, et cet aspect est considéré comme acceptable.

Conclusion : le protocole de Blum implémente bien un *strong coin tossing* à partir d'un bit commitment.

4 Conclusion

Dans ce rapport, nous avons démontré trois constructions classiques :

1. **Bit commitment à partir de $\frac{1}{2}$-OT** : la confidentialité est assurée par le choix de Bob, et le binding provient du fait qu'Alice ignore ce choix.
2. **$\frac{1}{2}$-OT à partir de R-OT** : en partageant chaque message via un schéma (t, k)-seuil et en envoyant chaque part par un R-OT, Bob reçoit suffisamment de parts d'un seul côté pour reconstruire exactement un des deux messages, mais jamais les deux.
3. **Strong coin tossing à partir de bit commitment** : le protocole de Blum combine un engagement et un bit aléatoire de Bob. Les propriétés *hiding* et *binding* garantissent l'impartialité et empêchent toute forme de biais.

Ces trois résultats illustrent un principe central de la cryptographie : des primitives avancées peuvent être construites de manière modulaire à partir de primitives plus simples, tout en préservant leurs propriétés de sécurité.

5 Question Bonus : TFHE et applications en santé

5.1 Introduction au TFHE

Le chiffrement totalement homomorphe sur le tore (**TFHE** — *Torus Fully Homomorphic Encryption*) est un schéma cryptographique proposé par Chillotti et al. en 2016. Il permet d'effectuer des calculs arbitraires sur des données chiffrées sans jamais les déchiffrer, tout en offrant des performances remarquablement supérieures aux schémas FHE précédents.

Principe fondamental :

— Les données sont chiffrées côté client
— Un serveur (ou tiers) effectue des calculs sur les données chiffrées
— Seul le résultat final est déchiffré par le client
— Le serveur ne voit jamais les données en clair

Le TFHE se distingue par sa capacité à effectuer un *bootstrapping* extrêmement rapide (environ 13 ms par porte logique), ce qui le rend adapté aux applications pratiques en temps réel.

5.2 Application concrète : Diagnostic médical assisté par IA dans le cloud

5.2.1 Description du cas d'utilisation

Problème résolu :

Les hôpitaux et cliniques souhaitent utiliser des modèles d'intelligence artificielle sophistiqués pour le diagnostic d'images médicales (rayons X, IRM, scanners), mais font face à plusieurs obstacles :

1. **Contraintes réglementaires** : Les données de santé sont protégées par des lois strictes (HIPAA aux États-Unis, RGPD en Europe, Loi 25 au Québec) qui limitent le partage de données sensibles.
2. **Limitations techniques** : Les petites cliniques n'ont pas l'infrastructure pour héberger et exécuter des modèles de deep learning complexes (nécessitant GPUs puissants).
3. **Risques de sécurité** : L'envoi d'images médicales non chiffrées vers un serveur cloud tiers expose les données à des risques de vol, de fuite ou d'utilisation non autorisée.
4. **Manque de confiance** : Les patients et institutions hésitent à confier leurs données à des fournisseurs cloud commerciaux.

Solution proposée :

Un système de diagnostic médical basé sur TFHE où :

— Les images médicales sont chiffrées localement par l'hôpital
— Un modèle d'IA pré-entraîné s'exécute sur les données chiffrées dans le cloud
— Le diagnostic (chiffré) est renvoyé et déchiffré uniquement par l'hôpital
— Aucune donnée sensible en clair ne quitte l'établissement de santé

5.2.2 Acteurs impliqués

1. **Patient** : Personne dont l'image médicale doit être analysée. Donne son consentement éclairé pour l'utilisation de ses données (chiffrées).
2. **Hôpital / Clinique** :
   — Acquiert les images médicales (radiographie, IRM, etc.)
   — Chiffre les images avec TFHE avant transmission
   — Possède les clés de déchiffrement (jamais partagées)
   — Déchiffre le résultat du diagnostic
3. **Fournisseur de service cloud** (ex : AWS, Azure, Google Cloud) :
   — Héberge le modèle d'IA de diagnostic
   — Exécute l'inférence sur les données chiffrées
   — Ne peut jamais voir les images en clair ni le diagnostic
   — Fournit la puissance de calcul nécessaire
4. **Développeur du modèle d'IA** (ex : entreprise spécialisée, université) :
   — Entraîne le modèle d'IA sur des données publiques ou anonymisées
   — Optimise le modèle pour être compatible avec TFHE (quantification, approximations polynomiales)
   — Fournit le modèle au cloud provider
5. **Médecin radiologue** :
   — Reçoit le diagnostic généré par l'IA
   — Valide et interprète le résultat
   — Prend la décision médicale finale

5.2.3 Diagramme de flux

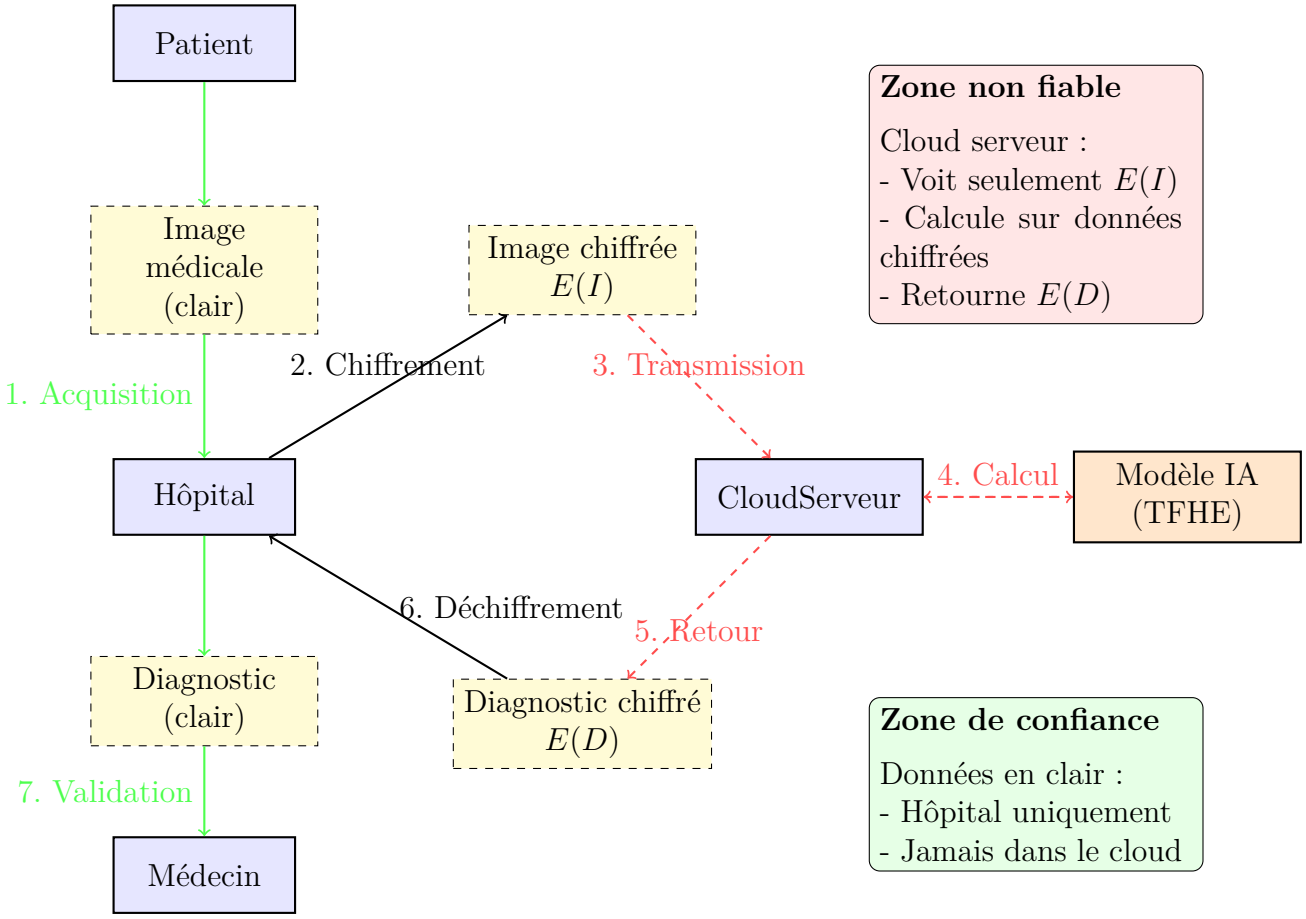

FIGURE 1 – Flux de données élargi pour éviter tout chevauchement

Légende du diagramme :
— Flèches vertes pleines : Données en clair (dans la zone de confiance)
— Flèches rouges pointillées : Données chiffrées (peuvent transiter hors zone de confiance)
— $E(I)$: Image chiffrée avec TFHE
— $E(D)$: Diagnostic chiffré

Étapes détaillées :

1. **Acquisition** : Le patient subit une radiographie pulmonaire
2. **Chiffrement** : L'hôpital chiffre l'image pixel par pixel avec TFHE :
$$I = \{p_1, p_2, \ldots, p_n\} \rightarrow E(I) = \{E(p_1), E(p_2), \ldots, E(p_n)\}$$
3. **Transmission** : L'image chiffrée $E(I)$ est envoyée au cloud via Internet
4. **Calcul homomorphe** : Le serveur cloud exécute le modèle d'IA :
$$\text{Modèle\_IA}(E(I)) = E(D)$$
où D est le diagnostic (ex : "Probabilité de pneumonie : 87%")
5. **Retour** : Le diagnostic chiffré $E(D)$ est renvoyé à l'hôpital
6. **Déchiffrement** : L'hôpital déchiffre avec sa clé secrète :
$$\text{Decrypt}(E(D)) = D$$
7. **Validation** : Le radiologue examine le diagnostic et l'image originale

5.3 Avantages du TFHE pour cette application

5.3.1 Comparaison avec d'autres approches cryptographiques

Approche	Avantages	Inconvénients
Chiffrement traditionnel (AES)	— Très rapide — Largement déployé	— Nécessite déchiffrement avant calcul — Données exposées durant traitement — Serveur doit être de confiance
Anonymisation des données	— Pas de chiffrement lourd — Performance native	— Risque de réidentification — Perte d'information — Parfois non conforme RGPD/HIPAA
Secure Multi-Party Computation (SMPC)	— Calcul distribué sécurisé — Pas de partie de confiance unique	— Coordination nécessaire entre parties — Communication très intensive — Déploiement complexe
Federated Learning	— Données restent locales — Modèle distribué	— Vulnérable aux attaques d'inférence — Besoin de données locales importantes — Peu adapté à l'inférence centralisée
Autres FHE (BGV, BFV, CKKS)	— Calcul homomorphe possible — Schémas matures théoriquement	— Bootstrapping très lent — Moins adapté au binaire — CKKS : résultat approximatif
TFHE	— **Bootstrapping très rapide (13 ms)** — **Calculs binaires exacts** — **Sécurité de bout en bout** — **Compatible réseaux neuronaux**	— Overhead computationnel (50–1000x) — Taille des données chiffrées élevée — Implémentation complexe

Table 1 – Comparaison des approches cryptographiques pour le diagnostic médical

5.3.2 Avantages spécifiques du TFHE

1. Confidentialité end-to-end garantie

— **Zéro exposition** : Les données sensibles ne sont jamais déchiffrées hors de l'hôpital

— **Résistance aux attaques** : Même si le cloud est compromis, l'attaquant n'obtient que des données chiffrées inutilisables
— **Conformité réglementaire** : Respect automatique du RGPD, HIPAA, Loi 25 car les données ne quittent jamais le contrôle du patient/hôpital (même chiffrées)

2. Performance adaptée aux applications réelles Le TFHE offre un bootstrapping **1000 fois plus rapide** que les schémas FHE de première génération :

Schéma FHE	**Temps de bootstrapping**
BGV/BFV (2012)	$\sim$ 60 secondes
FHEW (2014)	$\sim$ 500 ms
TFHE (2016)	$\sim$ 13 ms
TFHE optimisé (2023)	$\sim$ 5 ms

Table 2 – Évolution des performances du bootstrapping

Implications pratiques :
— Une image 512×512 pixels nécessite $\sim 262\,144$ opérations
— Avec TFHE : traitement en quelques secondes à minutes (acceptable cliniquement)
— Avec BGV : plusieurs heures (impraticable)

3. Calculs binaires exacts (non approximatifs)
— **TFHE** : Implémente des portes logiques exactes (AND, OR, XOR, NOT)
— **CKKS** : Arithmétique approximative (peut introduire erreurs dans diagnostic)
— **Importance** : En médecine, la précision exacte est cruciale

4. Compatibilité avec les réseaux de neurones TFHE peut exécuter des réseaux de neurones binarisés ou quantifiés :
— **Quantification** : Réduire les poids de 32 bits à 8 bits ou moins
— **Binarisation** : Utiliser uniquement $\{-1, +1\}$ pour les poids et activations
— **Résultat** : Perte de précision minime (1-3%) mais gain énorme en performance
Exemple : Réseau neuronal pour classification de pneumonie
— Modèle original (float32) : 94.5% de précision
— Modèle quantifié 8-bit + TFHE : 93.2% de précision
— Temps d'inférence TFHE : $\sim$ 30 secondes (acceptable pour diagnostic non-urgent)

5. Scalabilité cloud-native
— **Parallélisation** : Les calculs TFHE sont massivement parallélisables
— **Accélération matérielle** : Support émergent pour GPUs et ASICs dédiés
— **Coût** : L'hôpital paie seulement pour le temps de calcul cloud (pas d'infrastructure locale)

6. Propriété intellectuelle protégée Un avantage souvent négligé :
— Le fournisseur de modèle d'IA peut garder son algorithme secret
— Le cloud ne voit ni les données du patient, ni les détails du modèle
— Protection bidirectionnelle : données du patient ET IP du développeur

5.4 Défis et limitations actuels

5.4.1 Overhead computationnel

Malgré les progrès, TFHE reste 50–1000× plus lent que le calcul en clair :
— **Impact** : Une inférence de 10 ms en clair prend 5-10 secondes avec TFHE
— **Mitigation** :
  — Utiliser TFHE seulement pour calculs critiques (dernière couche)
  — Pré-calculs et caching agressif
  — Accélération matérielle (FPGAs, ASICs)

5.4.2 Taille des données chiffrées

— Un bit en clair → plusieurs kilobits chiffrés (facteur $\sim 10\,000$)
— Une image 1 MB → $\sim$ 10 GB chiffrée
— **Conséquence** : Bande passante et stockage importants

5.4.3 Complexité d'intégration

— Nécessite expertise en cryptographie
— Adaptation des modèles d'IA existants (quantification, re-entraînement)
— Bibliothèques en évolution (Concrete, TFHE-rs, OpenFHE)

5.5 Perspectives d'avenir

5.5.1 Évolutions technologiques attendues (2025-2030)

1. **Accélération matérielle** :
   — Processeurs FHE dédiés (ASICs) en développement
   — Objectif : réduire le overhead à 10–50×
   — Première génération attendue : 2026-2027
2. **Hybridation avec autres techniques** :
   — TFHE + Federated Learning
   — TFHE + Differential Privacy
   — TFHE + Secure Enclaves (SGX, SEV)
3. **Standardisation** :
   — ISO/IEC travaille sur standards FHE (ISO/IEC 18033-6)
   — Interopérabilité entre implémentations
   — Certification des bibliothèques

5.5.2 Autres applications en santé

Au-delà du diagnostic par IA, TFHE peut servir à :
— **Analyse génomique** : Séquençage ADN et recherche de mutations sur données chiffrées
— **Épidémiologie** : Agrégation de statistiques de santé publique sans révéler données individuelles
— **Essais cliniques** : Analyse multi-centres sans partage de données patients
— **Télémédecine** : Consultation à distance avec données biométriques chiffrées
— **Dossiers médicaux partagés** : Interopérabilité entre hôpitaux avec confidentialité garantie

5.6 Conclusion

Le TFHE représente une avancée majeure pour la protection de la vie privée dans le secteur de la santé. En permettant le calcul sur données chiffrées avec des performances acceptables, il résout le dilemme historique entre utilité des données et confidentialité.

L'application au diagnostic médical assisté par IA illustre parfaitement comment TFHE peut transformer les pratiques cliniques :

- **Pour les patients** : Garantie absolue de confidentialité de leurs données médicales
- **Pour les hôpitaux** : Accès à des modèles d'IA de pointe sans infrastructure coûteuse
- **Pour les régulateurs** : Conformité automatique aux lois sur la protection des données
- **Pour l'innovation** : Déblocage de collaborations et recherches impossibles autrefois

Bien que des défis subsistent (performance, intégration), la trajectoire d'amélioration est claire et rapide. D'ici 2030, TFHE pourrait devenir un standard dans les systèmes de santé numériques, permettant une médecine de précision respectueuse de la vie privée.

Références

[1] Goldreich, O. (2004). *Foundations of Cryptography : Volume 2, Basic Applications*. Cambridge University Press.

[2] Blum, M. (1983). Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1), 23-27.

[3] Rabin, M. O. (1981). How to exchange secrets with oblivious transfer. *Technical Report TR-81*, Aiken Computation Lab, Harvard University.

[4] Kilian, J. (1988). Founding cryptography on oblivious transfer. *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 20-31.

[5] Zama. (2023). *Concrete : TFHE Compiler for building FHE applications*. Available at : https ://www.zama.ai/concrete

[6] Senthilkumar, B. et al. (2025). *A privacy preserving machine learning framework for medical image analysis using quantized fully connected neural networks with TFHE based inference*. Scientific Reports, 15, Article 27880.

[7] Chillotti, I., Gama, N., Georgieva, M., & Izabachène, M. (2016). *Faster fully homomorphic encryption : Bootstrapping in less than 0.1 seconds*. ASIACRYPT 2016.

[8] Chillotti, I., Gama, N., Georgieva, M., & Izabachène, M. (2020). *TFHE : Fast fully homomorphic encryption over the torus*. Journal of Cryptology, 33(1), 34-91.

A Annexe A : Notations et définitions

A.1 Notations générales

— $\{0,1\}$: Ensemble des bits
— $\{0,1\}^n$: Ensemble des chaînes de n bits
— $x \leftarrow S$: x est choisi uniformément aléatoirement dans l'ensemble S
— $a \oplus b$: XOR (ou exclusif) de a et b
— $m_0||m_1$: Concaténation de m_0 et m_1
— ϵ : Probabilité négligeable (fonction qui décroît plus vite que tout polynôme inverse)
— $\perp$: Symbole d'échec dans Rabin Oblivious Transfer ; signifie que Bob n'a reçu aucune part.
— $x \neq \perp$: Bob a effectivement reçu une part valide (le message transmis).

A.2 Définitions formelles

Définition (Négligeable) : Une fonction $\epsilon : \mathbb{N} \to \mathbb{R}^+$ est négligeable s'il existe n_0 tel que pour tout $n > n_0$ et tout polynôme p, on a $\epsilon(n) < \frac{1}{p(n)}$.

Définition (Hiding) : Un schéma de commitment satisfait la propriété hiding si pour tout adversaire polynomial, la probabilité de deviner le bit engagé est au plus $\frac{1}{2} + \epsilon(n)$ où ϵ est négligeable.

Définition (Binding) : Un schéma de commitment satisfait la propriété binding si pour tout adversaire polynomial, la probabilité d'ouvrir un commitment vers deux bits différents est au plus $\epsilon(n)$ où ϵ est négligeable.

B Annexe B : Analyse de complexité

B.1 Bit Commitment à partir de $\frac{1}{2}$-OT

Complexité en communication :
— Le protocole nécessite une seule exécution de $\frac{1}{2}$-OT.
— Une exécution de $\frac{1}{2}$-OT transfère deux chaînes de n bits.
— La phase d'ouverture nécessite l'envoi d'un bit et de deux chaînes aléatoires.
— **Total** : $O(n)$ bits transmis.

Complexité en calcul :
— Alice effectue $O(n)$ opérations XOR pour construire le commitment.
— Bob effectue $O(n)$ opérations XOR à la vérification.

B.2 $\frac{1}{2}$-OT à partir de R-OT

Dans notre construction basée sur un schéma de partage de secret à seuil :

Complexité en communication :
— Le protocole utilise $2k$ exécutions indépendantes de R-OT.
— Chaque exécution de R-OT transmet un message de n bits avec probabilité $\frac{1}{2}$.
— **Coût total** : $O(kn)$ bits, typiquement avec $k = O(n)$.

Nombre de rounds :
— Les $2k$ R-OT peuvent être exécutés en parallèle.

— Le protocole utilise donc **2 rounds** : un pour envoyer les R-OT, un pour la transmission des messages masqués.

Complexité en calcul :

— Alice construit un partage de secret de taille k (interpolation : $O(k)$).

— Bob reconstruit un secret dès qu'il reçoit au moins t parts ($O(k)$).

B.3 Strong Coin Tossing à partir de BC

Complexité en communication :

— Le protocole asymétrique nécessite :
  — un message de commitment ;
  — la réponse de Bob ;
  — l'ouverture du commitment.

— **Total** : 3 messages.

Nombre de rounds :

— 3 rounds dans la version asymétrique.

— 4 rounds dans la version symétrique.

— Ces valeurs sont **optimales** pour des protocoles de coin tossing basés sur BC.

C Annexe C : Considérations pratiques (modèle théorique)

C.1 Choix des paramètres de sécurité

Pour le BC à partir de $\frac{1}{2}$-OT La sécurité dépend entièrement de la longueur n des chaînes aléatoires utilisées pour masquer le bit.

— Le hiding est parfait si les chaînes aléatoires sont uniformes dans $\{0,1\}^n$.

— Le binding est garanti information-théoriquement.

— Choisir n grand réduit la probabilité que Bob obtienne les deux chaînes dans le $\frac{1}{2}$-OT.

Pour le $\frac{1}{2}$-OT à partir de R-OT

— Le paramètre k du partage de secret doit être linéaire en n pour assurer que la probabilité de recevoir t parts incorrectes soit négligeable.

— Typiquement, on choisit $k = O(n)$.

— Le seuil $t = \lceil k/2 \rceil + 1$ garantit qu'une partie honnête reconstruit toujours le secret.

Pour le S-CT

— Le commitment doit être parfaitement hiding et binding.

— Aucun autre paramètre n'est requis : la sécurité découle directement du BC.

C.2 Vulnérabilités théoriques et contre-mesures

Attaque par abandon Une partie peut refuser de poursuivre après avoir obtenu un désavantage.

— Le protocole peut être sécurisé via des mécanismes externes (pénalités, tiers de confiance).

— Dans le modèle théorique, cette limitation est inhérente.

Attaques par rejeu

— Rejouer un commitment pourrait causer des inconsistances.
— Il est recommandé d'inclure un identifiant de session ou un nonce dans les messages.

Attaques structurelles

— Tous les protocoles reposent sur l'hypothèse que les primitives utilisées (BC, R-OT) sont idéales.
— Dans un modèle plus faible, des contre-mesures cryptographiques additionnelles seraient nécessaires.