# Rapport de stage — Cryptographie hybride autour de `Forrelation`

Arthur Moindrault--Cailleau*

**Abstract**

À la suite des travaux de [MBBA23] sur un protocole de cryptographie hybride d'échange de clés reposant sur le problème `Partial Matching`, la question s'est posée de trouver un protocole hybride d'échange de clés encore plus sécurisé, en s'appuyant sur le problème `Forrelation`, qui présente l'avantage d'être généralisable en $k$-`fold Forrelation`, plus difficile que `Partial Matching`. Dans le cadre de ce stage, nous avons cherché des protocoles basés sur différentes variantes de `Forrelation`, avant de chercher des protocoles basés sur des problèmes plus généraux, qui permettent une séparation en termes de difficulté classique vs quantique identique.

## 1 Introduction

### 1.1 Le modèle QCT

Le modèle QCT (*Quantum Computational Time-lock*) a été introduit dans [VA20]. C'est un modèle hybride qui peut se modéliser comme comme à la figure 1.

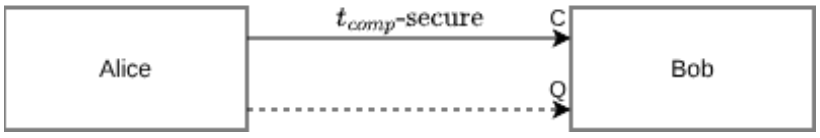


Figure 1: Schéma du modèle QCT.

Alice et Bob disposent donc de deux canaux de communication : un canal quantique $Q$ et un canal classique $C$, qui est supposé $t_{comp}$-secure, c'est-à-dire qu'une attaquante Eve ne peut pas déchiffrer les informations du canal $C$ en un temps inférieur à $t_{comp}$.

Intuitivement, une telle hypothèse sur un canal classique oblige Eve à effectuer une mesure sur le canal $Q$ avant d'avoir reçu une quelconque information sur le canal $C$, en supposant que la mémoire quantique d'Eve ne lui permet pas de conserver l'information de $Q$ plus longtemps que $t_{comp}$. Ainsi, si la mesure de $Q$ dépend de $C$, Eve n'aura pas toutes les informations nécessaires pour effectuer une mesure efficace, et aura donc potentiellement moins d'information finale que Bob, qui a accès à $C$ dès le début. La validité du modèle est discutée dans [MBBA23]. Une comparaison avec un modèle plus largement utilisé pour un échange de clé, QKD (Quantum Key Distribution), utilisé par l'algorithme [BB84], est apportée en figure 2.

*ENS Rennes, sous la direction de Romain Alléaume, équipe QURIOSITY, Télécom Paris

| | QKD | QCT (échange de clés) |
|---|---|---|
| Hypothèses | $\varnothing$ | Temps de stockage quantique restreint pour l'attaquante |
| Nombre de copies | $O(1)$ | $\tilde{O}(\sqrt{N})$ pour $\beta$PM [MBBA23] |
| Nombre de canaux | 1 quantique + 1 classique | 1 quantique + 1 classique |
| Sécurité | Basée sur des lois physiques | Basée sur des hypothèses calculatoires + Mémoire quantique bruitée |
| Mécanisme | Repère la présence d'une attaquante | Ne donne pas assez d'informations à l'attaquante |
| Communication | *Two-ways* | *One-way* ou *Two-ways* |

Figure 2: Tableau de comparaison QKD/QCT

### 1.2 Communication VS requêtes

**Definition 1.** *Un problème de communication, introduit par Yao [Yao79], peut se formaliser de la manière suivante : deux joueurs, Alice et Bob, se voient attribuer une entrée chacun, $x$ pour Alice et $y$ pour Bob. Leur but est de calculer $f(x, y)$, pour une certaine fonction $f$ décidée à l'avance.*

*La complexité de communication d'un problème $f$ est le nombre de bits échangés entre Alice et Bob dans le pire cas du meilleur protocole (avec une erreur $\varepsilon$ autorisée) :*

$$D^1_\mu(f, \varepsilon) := \min_{\pi:\mathbb{P}[f(x,y)\neq\pi(x,y)]\leq\varepsilon} CC(\pi),$$

*où $CC(\pi)$ correspond au nombre de bits transmis dans le pire cas par un protocole $\pi$, et où $x, y$ sont tirés selon la distribution $\mu$.*

Dans notre cas, nous allons nous concentrer sur un problème *one-way*, c'est-à-dire une restriction dans laquelle seule Alice est autorisée à envoyer de l'information à Bob, et Bob doit calculer $f(x, y)$.

**Definition 2.** *Un problème de requête consiste en une joueuse, Alice, qui interroge un oracle $O_f$ dans le but de calculer $f(x)$. On appelle complexité de requête le nombre maximal de requêtes qu'Alice doit faire à $O_f$ pour calculer $f(x)$.*

### 1.3 `Forrelation`

Pour établir un protocole sûr, il nous faut nous baser sur un problème « difficile ». Dans notre cas, la difficulté du problème réside dans la séparation entre sa résolution par un adversaire classique et par un adversaire quantique. `Forrelation` est un problème qui atteint une grande séparation, avec une seule requête pour un adversaire quantique, contre $\widehat{O}(\sqrt{N})$ requêtes pour un adversaire classique, où $N = 2^n$ et $n$ représente la taille des entrées quantiques [AA15]. Intuitivement, cette séparation provient de la facilité de faire une transformée de Fourier en quantique, face à la difficulté de celle-ci en classique.

**Definition 3.** *Le coefficient de forrelation $\Phi_{f,g}$ permet d'estimer la corrélation entre $f$ et la transformée de Fourier binaire $\widehat{g}$ de $g$. Il est défini pour $f, g : \{0,1\}^n \to \{-1, 1\}$ par :*

$$\Phi_{f,g} := \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} f(x)(-1)^{x\cdot y} g(y) = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} f(x)\widehat{g}(x).$$

*On montre facilement que $\Phi_{f,g} \in [-1, 1]$.*

À partir de ce coefficient, nous pouvons définir les deux problèmes `Forrelation` [AA15] et `Extremal Forrelation` [GS25] (partie suivante) :

**Definition 4.** *Le problème* ***`Forrelation`*** *consiste à distinguer le cas* $|\Phi_{f,g}| \leq \frac{1}{100}$ *(instance négative) du cas* $\Phi_{f,g} \geq \frac{3}{5}$ *(instance positive).*

Pour pouvoir établir un protocole sur ce problème, il faut définir un algorithme de génération qui permet de trouver des fonctions $f$ et $g$ respectant les contraintes du coefficient pour `Forrelation`. On définit alors les distributions $\mathcal{G}_0^F$ et $\mathcal{G}_1^F$ comme suit :

**Definition 5.** *Pour* $x, y \in \{0,1\}^n$, $F(x), G(y) \sim \mathcal{N}(0,1)$.

- *On définit alors* $\mathcal{G}_0^F$ *comme la distribution de* $f, g$ *telles que* $f = \mathrm{sgn}(F)$ *et* $g = \mathrm{sgn}(G)$, *où* sgn *est la fonction de signe (elle associe* $-1$ *à un nombre négatif et* $+1$ *à un nombre positif).*
- $\mathcal{G}_1^F$ *est la distribution de* $f, g$ *telles que* $f = \mathrm{sgn}(F)$ *et* $g = \mathrm{sgn}(\widehat{F})$, *avec* $\widehat{F}$ *la transformée de Fourier de* $F$.

Une implémentation de ces distributions est proposée en annexe 5.1.

**Proposition 1.** *Nous avons les bornes suivantes :*

- $\mathbb{P}_{f,g\sim\mathcal{G}_0^F}\left[|\Phi_{f,g}| \geq \frac{1}{100}\right] \leq \frac{10000}{N}$.
- $\mathbb{P}_{f,g\sim\mathcal{G}_1^F}\left[\Phi_{f,g} \geq \frac{3}{5}\right] \geq \frac{1}{30}$.

La preuve est donnée dans le corollaire 10 de [AA15]. Pour $n \geq 14$, on a donc accès à un algorithme probabiliste qui nous fournit $f, g$ telles que $|\Phi_{f,g}| \leq \frac{1}{100}$ avec une probabilité supérieure à 0,35. En itérant cet algorithme, on obtient donc une génération d'instances négatives. Il en est de même pour une instance positive.

### 1.4 `Extremal Forrelation`

Une variante de `Forrelation` a été étudiée en 2025 par Girish et Servedio [GS25]. L'avantage de cette variante est qu'elle permet de générer directement des fonctions à valeurs dans $\{-1, 1\}$, au lieu de passer par une fonction de signe qui rend l'étude de $\Phi_{f,g}$ plus difficile. Dans ce même article, il est conjecturé que la complexité de `Extremal Forrelation` est identique à celle de `Forrelation`, mais seule une séparation d'une requête quantique face à $\tilde{O}(N^{1/4})$ requêtes classiques est prouvée.

**Definition 6.** *Le problème* ***`Extremal Forrelation`*** *consiste à distinguer le cas* $\Phi_{f,g} = -1$ *(instance négative) du cas* $\Phi_{f,g} = 1$ *(instance positive).*

**Definition 7.** *On suppose* $n$ *pair. Les fonctions de Maiorana–McFarland sont de la forme :*

- $f(x) := \langle A_1 x, A_2 x\rangle + \langle x, a\rangle + h(A_2 x)$,
- $g(y) := \langle B_1 y, B_2 y\rangle + \langle y, b\rangle + h(B_1 y + B_1 a) + \langle B_1 a, B_2 a\rangle$.

*Avec* $A \in \mathbb{F}_2^{n\times n}$ *de rang maximal,* $B := (A^T)^{-1}$, $a \in \mathbb{F}_2^n$, $b := B^T \begin{bmatrix} B_2 \\ B_1 \end{bmatrix} a$, $h : \mathbb{F}_2^{n/2} \to \mathbb{F}_2$, *et* $A_1, A_2$ *(resp.* $B_1, B_2$*) représentant les moitiés haute et basse de* $A$ *(resp. de* $B$*).*

On peut noter qu'il existe des expressions plus simples de ces fonctions, mais que celles-ci garantissent la séparation prouvée dans [GS25]. À partir de ces fonctions, on définit alors nos distributions $\mathcal{G}_0^X$ et $\mathcal{G}_1^X$ :

**Definition 8.** *On considère $f$ et $g$ comme définies ci-dessus.*

- $\mathcal{G}_0^X$ *représente la distribution des paires* $((-1)^f, -(-1)^g)$.
- $\mathcal{G}_1^X$ *représente la distribution des paires* $((-1)^f, (-1)^g)$.

**Proposition 2.** *D'après [GS25], on a :*

- *si* $f, g \sim \mathcal{G}_0^X$, *alors* $\Phi_{f,g} = -1$ *;*
- *si* $f, g \sim \mathcal{G}_1^X$, *alors* $\Phi_{f,g} = 1$.

**Proposition 3.** *Soient $a$ et $B$ définis comme dans la définition 7. Alors :*
$\mathbb{P}[\langle B_1 a, B_2 a\rangle \mod 2 = 0] \geq \frac{1}{4}$ *et* $\mathbb{P}[\langle B_1 a, B_2 a\rangle \mod 2 = 1] \geq \frac{1}{4}$ *(pour $n \geq 2$).*

*Une preuve est proposée en annexe 5.3*

On peut alors, à partir de cette propriété, générer $f$ et $g'$ tels que $\Phi_{f,g'} = \pm 1$, avec

$$g'(y) := \langle B_1 y, B_2 y\rangle + \langle y, b\rangle + h(B_1 y + B_1 a).$$

On a donc encore ici accès à un algorithme de génération pour `Extremal Forrelation` : il suffit à Alice de générer aléatoirement $a$, ce qui, d'après la proposition précédente, lui donne au moins une chance sur quatre d'obtenir une instance positive et une chance sur quatre d'obtenir une instance négative. Le seul problème pourrait venir de la génération de $A$ (qui doit être de rang maximal), mais on apprend dans [GS25] qu'une matrice aléatoire dans $\mathbb{F}_2^{n\times n}$ est de rang maximal avec une probabilité supérieure à $1 - (n/2)2^{-n/2}$.

Une implémentation de cette génération est proposée en annexe 5.2.

# 2 Protocoles

Dans cette partie, nous allons nous intéresser à la réalisation de protocoles d'échange de clés entre deux acteurs, Alice et Bob. Alice souhaite envoyer à Bob une clé privée, qu'une adversaire malveillante Eve ne peut pas deviner.

Pour pouvoir garantir qu'un protocole est sûr, on peut faire une preuve de sécurité comme celle exposée dans [MBBA23]. L'idée générale consiste à réduire toute attaque de Eve à une mesure dès qu'elle reçoit l'information sur canal quantique. Un lien entre la complexité de communication et la probabilité pour Eve de trouver la clé peut alors permettre de conclure sur la sécurité du protocole. De plus, la complexité de communication permet de décider du nombre de copies maximal d'un même état quantique à envoyer pour garantir qu'Eve ne puisse pas retrouver la clé. En réalité, ce nombre de copies permet de corriger des erreurs de transmission (le bruit sur canal quantique est un problème majeur), et peut aussi permettre la communication à plusieurs acteurs.

Chacun des protocoles décrits ci-dessous permettent à Alice d'envoyer un unique bit d'information, mais on peut lever cette contrainte en considérant que l'on peut effectuer plusieurs rondes de ces protocoles.

### 2.1 Basé sur `Forrelation`

Une option prometteuse pour Alice est d'envoyer $f$ sur le canal classique, et $|\Psi_g\rangle$ sur le canal quantique. Bob pourra alors calculer $\langle\Psi_f|H|\Psi_g\rangle = \Phi_{f,g}$ et conclure (voir figure 3). Le problème de cette solution est qu'Eve pourrait alors calculer puis mesurer une partie de $H|\Psi_g\rangle$ (selon la borne d'Holevo [Hol73]), ce qui réduit le problème à un calcul de produit scalaire entre $f$ et $H|\Psi_g\rangle$ pour Eve.

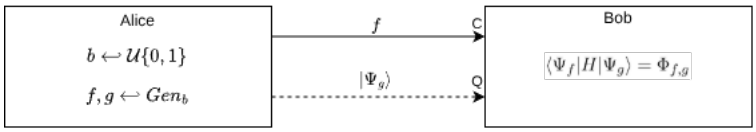


Figure 3: Schéma du protocole basé sur `Forrelation`.

### 2.2 Basé sur `Extremal Forrelation`

De la même façon que pour `Forrelation`, Alice génère $f$ et $g$. Cette fois cependant, elle ne va pas envoyer directement ces fonctions, sinon le problème serait le même. À la place, Alice envoie $A$ et $h$ sur le canal classique, et $|\Psi_a\rangle$ sur le canal quantique, comme comme à la figure 4.

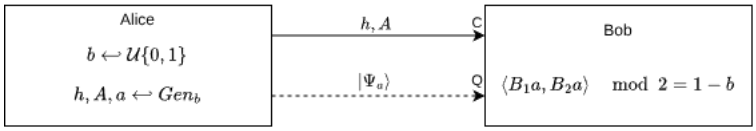


Figure 4: Schéma du protocole basé sur `Extremal Forrelation`.

Cependant, le problème n'a fait qu'être déplacé. En effet, il suffit maintenant pour Eve (et pour Bob) de calculer $\langle B_1 a, B_2 a\rangle$ pour connaître la valeur du bit qu'Alice voulait envoyer, ce qui revient encore une fois à calculer un produit scalaire.

### 2.3 `Gap Hamming`

On remarque donc que nos deux protocoles se heurtent à une réduction à un calcul de produit scalaire.

**Definition 9.** *Le problème* `Gap Hamming` *consiste à distinguer, pour* $x, y \in \{-1, 1\}^n$*, entre* $GHD(x, y) = 1$ *et* $GHD(x, y) = -1$*, où :*

$$GHD(x,y) = \begin{cases} 1 & si\ \langle x|y\rangle \leq -\sqrt{n}, \\ -1 & si\ \langle x|y\rangle \geq \sqrt{n}. \end{cases}$$

Il a été prouvé dans [CR18] que `Gap Hamming` possède une séparation en termes de complexité de communication en $O(\sqrt{N}\log N)$ en quantique contre $\Omega(N)$ en classique. Dans l'idée d'établir un protocole, ce qui est intéressant est le nombre de photons (i.e. le nombre de copies) que l'on va pouvoir envoyer à Bob avant que Eve ne puisse avoir un avantage important, et dans notre cas ce nombre sera inférieur à $O(\sqrt{N}\log N)$, et si cette borne est atteignable, alors la sécurité sera identique à celle de [MBBA23].

Une question légitime serait alors : Pourquoi tout ce travail pour un protocole "aussi sûr" que celui de [MBBA23] ? La réponse se trouve dans [AA15] : c'est une bonne brique de départ pour trouver un protocole pour $k$-fold `Forrelation`, une généralisation de `Forrelation`, qui possède l'avantage d'avoir une séparation en $k/2$ requêtes quantiques [AA15] contre $\tilde{O}(N^{1-1/k})$ requêtes classiques [BS20], et qui pourrait donc avoir une séparation en termes de communication plus ambitieuse (donc plus intéressant pour transmettre plus de copies).

On peut alors imaginer le protocole simple suivant pour `Gap Hamming` :

- Alice génère $x$ et $y$ selon le bit qu'elle veut représenter
- Elle transmet $|x\rangle$ quantiquement à Bob, et $y$ classiquement.

### 2.4 Basé sur `LWE`

`LWE` est un problème qui est conjecturé difficile classiquement comme quantiquement, mais le nombre de résultats concernant sa difficulté est encore faible. Sa construction semble malgré tout intéressante, et on peut s'en inspirer pour construire le protocole suivant (qui n'est pas *one-way*) :

**Protocole** On suppose l'existence d'un nombre premier $q$ (public).

- Bob choisit $a_1, \dots, a_m$, $s$ des vecteurs de $\mathbb{F}_q^n$ et $e_1, \dots, e_m$ des "erreurs" dans $\mathbb{R}$.
- Il envoie $\{a_i\}$, $\{t_i\}$ à Alice (pour tout $i \in \{1, \cdots, m\}$), avec $t_i := \frac{\langle a_i|s\rangle}{q} + e_i$
- Alice choisit un bit $x \in \mathbb{B}$ et un sous-ensemble $S \subseteq \{1, \cdots, m\}$.
- Elle envoie $t$ et $|a\rangle$ à Bob, avec $t := \frac{x}{2} + \sum_{s\in S} t_s$, $a := \sum_{s\in S} a_s$
- Bob calcule $t - \frac{\langle a|s\rangle}{q} = \frac{x}{2} + e$, et peut décider la valeur de $x$, en supposant $e$ "suffisamment petit".

Par "suffisamment petit", on entend ici que la valeur de $e$ permette la distinction de $x$ dans la plupart des cas. Il faut tout de même répondre à certains critères aléatoires sur cette erreur, afin de conserver l'hypothèse de difficulté de `LWE` (typiquement en tirant cette erreur suivant une gaussienne disctrète centrée en 0 [Reg24]).

Le principal problème réside ici dans le choix de $S$. Si Eve parvient à trouver $S$, alors ayant accès à $t$ et à $\{t_i\}$ après un certain temps, elle peut calculer $t - \sum_{i\in S} t_i = \frac{x}{2}$. Or, Eve peut trouver $S$ en résolvant le problème `Subset Sum` sur $(\{a_i\}, |a\rangle)$, en supposant qu'elle puisse avoir suffisamment d'informations sur $|a\rangle$. Le problème se réduit donc à `Subset Sum`.

## 3 Conclusion et perspectives

À ce jour, toutes les pistes présentées dans ce rapport sont encore étudiées afin de trouver un protocole viable, qui pourrait utiliser la difficulté de `Forrelation`. Ces travaux ont permis de mettre en lumière l'impossibilité de mettre en œuvre certains protocoles, ainsi que certaines réductions (le plus souvent à des variantes de `Gap Hamming`). Ce dernier problème semble par ailleurs prometteur, puisqu'il paraît très proche de `Forrelation`.

De plus, une des questions que font réémerger ces travaux concerne l'utilité de `Forrelation` face à [BB84] pour un échange de clés. Plus particulièrement, quels sont réellement les intérêts d'un protocole basé sur `Forrelation` ? Le modèle QCT est-il pertinent face à QKD, ou à d'autres modèles similaires comme QDM [DKM$^+$25] pour un échange de clés ?

## 4 Remerciements

Je remercie tout particulièrement Romain Alléaume, qui m'a dirigé pendant ce stage, et qui m'a permis d'éprouver en conditions réelles ma première expérience de recherche, constituée certes d'échecs, mais surtout de rebondissements dans un domaine très stimulant. Je remercie également Jeanne Lucas pour sa pédagogie et sa patience, qui m'a permis de mieux comprendre (et apprendre !) de nombreux aspects du domaine, et lui souhaite une bonne continuation pour sa thèse à venir. Enfin, je remercie l'équipe QURIOSITY pour son accueil, et pour son interdisciplinarité très fertile.

## References

[AA15] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. 2015. arXiv:1411.5729.

[BB84] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. 1984. arXiv:2003.06557.

[BS20] Nikhil Bansal and Makrand Sinha. k-forrelation optimally separates quantum and classical query complexity. 2020. arXiv:2008.07003.

[CR18] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. 2018. arXiv:1009.3460.

[DKM+25] Nico Döttling, Alexander Koch, Sven Maier, Jeremias Mechler, Anne Müller, Jörn Müller-Quade, and Marcel Tiepelt. The quantum decoherence model: Everlasting composable secure computation and more. Cryptology ePrint Archive, Paper 2025/220, 2025. URL: https://eprint.iacr.org/2025/220.

[GS25] Uma Girish and Rocco A. Servedio. Forrelation is extremally hard. 2025. arXiv:2508.02514.

[Hol73] A. S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problems of Information Transmission*, 9(3):177–183, 1973. English translation of Probl. Peredachi Inf. 9(3):3–11. URL: https://www.mathnet.ru/eng/ppi903.

[MBBA23] Francesco Mazzoncini, Balthazar Bauer, Peter Brown, and Romain Alléaume. Hybrid quantum cryptography from communication complexity. 2023. arXiv:1411.5729.

[Reg24] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. 2024. arXiv:2401.03703.

[VA20] Nilesh Vyas and Romain Alléaume. Everlasting secure key agreement with performance beyond qkd in a quantum computational hybrid security model. 2020. arXiv:2004.10173.

[Yao79] Andrew C.-C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 209–213, 1979. doi:10.1145/800135.804414.

# 5 Annexes

## 5.1 Génération pour `Forrelation`

```
import numpy as np

## Paramètres
n = 16 # OK jusqu'à 18-22 ; 16 est une valeur intéressante (14 un peu juste)
N = 1 << n
rng = np.random.default_rng()

# Pour n = 14, on est plutôt autour de 80 %
# de réussite au lieu de 40 % annoncés (dans le cas aléatoire)

# Pour le cas Forrelated, on semble tendre vers 3/5 en termes de valeur de forrelation,
# mais on est en réalité souvent au-dessus.

##Fonctions utiles
def fwht(a): #fast Walsh-Hadamard transform (multiplication optimisée par H)
    a = np.asarray(a, dtype=np.float64).copy()
    h = 1
    while h < a.size:
        for i in range(0, a.size, 2 * h):
            x = a[i:i+h].copy()
            y = a[i+h:i+2*h].copy()
            a[i:i+h] = x + y
            a[i+h:i+2*h] = x - y
        h *= 2
    return a

def sign_pm1(x): # fonction de signe
    return np.where(x >= 0, 1.0, -1.0)


def forrelation(F, G):
    HG = fwht(G)
    return np.dot(F, HG) / (N * np.sqrt(N))



##Génération
f = rng.normal(loc=0.0, scale=1.0, size=N) #f random

g = fwht(f) #g forrelated à f
g2 = rng.normal(loc=0.0, scale=1.0, size=N) # g random

# Passage au signe
F = sign_pm1(f)
G = sign_pm1(g)
```

```
G2 = sign_pm1(g2)

forr1 = forrelation(F, G)
forr2 = forrelation(F, G2)
print("forrelation =", forr1, ", ok : ", forr1 > 3/5, ", proba d'être ok >=", 1/30)
print("forrelation2 =", forr2, ", ok : ", abs(forr2) < 1/100, ", proba d'être ok >=",
0 if 1 - 100**2/N < 0 else 1 - 100**2/N)
```

### 5.2 Génération pour Extremal Forrelation

```
import numpy as np

## Paramètres
n = 16 #OK jusqu'à 18-20. Globalement plus lent que Aaronson.
N = 1 << n
k = n // 2
rng = np.random.default_rng()


##Fonctions utiles
def parity(v): #XOR de toutes les coordonnées de v
    v = np.asarray(v, dtype=np.uint8).ravel()
    return int(np.bitwise_xor.reduce(v))

def matvec_gf2(M, v): #Multiplication matricielle (dans F2)
    M = np.asarray(M, dtype=np.uint8)
    v = np.asarray(v, dtype=np.uint8).ravel()
    return np.bitwise_xor.reduce(M & v, axis=1).astype(np.uint8)

def rank_gf2(A): #Calcul du rang (dans F2)
    A = np.asarray(A, dtype=np.uint8).copy()
    n, m = A.shape
    r = 0
    for c in range(m):
        piv = np.flatnonzero(A[r:, c])
        if piv.size == 0:
            continue
        p = r + piv[0]
        if p != r:
            A[[r, p]] = A[[p, r]]
        mask = A[:, c].astype(bool)
        mask[r] = False
        A[mask] ^= A[r]
        r += 1
        if r == n:
            break
    return r

def inv_gf2(A): # Calcul de l'inverse (dans F2)
```

```
    A = np.asarray(A, dtype=np.uint8).copy()
    n = A.shape[0]
    I = np.eye(n, dtype=np.uint8)

    for c in range(n):
        piv = np.flatnonzero(A[c:, c])
        if piv.size == 0:
            raise ValueError("Matrice non inversible sur GF(2)")
        p = c + piv[0]
        if p != c:
            A[[c, p]] = A[[p, c]]
            I[[c, p]] = I[[p, c]]

        mask = A[:, c].astype(bool)
        mask[c] = False
        A[mask] ^= A[c]
        I[mask] ^= I[c]

    return I

def rand_mat(n): # Matrice aléatoire de rang maximal
    while True:
        A = rng.integers(0, 2, size=(n, n), dtype=np.uint8)
        if rank_gf2(A) == n:
            return A

def rand_vec(n): # Vecteur aléatoire
    return rng.integers(0, 2, size=n, dtype=np.uint8)

def int_to_bits(x, n): # Convertit un entier en son tableau de bits correspondant
    return np.array([(x >> i) & 1 for i in range(n - 1, -1, -1)], dtype=np.uint8)

def bits_to_int(bits): # Fait l'inverse
    bits = np.asarray(bits, dtype=np.uint8).ravel()
    out = 0
    for b in bits:
        out = (out << 1) | int(b)
    return out

def fwht(a): #Fast Walsh-Hadamard transform
    a = a.copy().astype(np.float64)
    n = a.shape[0]
    h = 1
    while h < n:
        for i in range(0, n, 2*h):
            x = a[i:i+h].copy()
            y = a[i+h:i+2*h].copy()
            a[i:i+h] = x + y
```

```
            a[i+h:i+2*h] = x - y
        h *= 2
    return a

def forrelation(F, G):
    N = len(F)
    HG = fwht(G)        # H g
    return np.dot(F, HG) / (N * np.sqrt(N))

##Eléments de base
A = rand_mat(n)
a = rand_vec(n)
B = inv_gf2(A.T)

B1 = B[:k]
B2 = B[k:]
C = np.vstack([B2, B1])          # [B2 B1]
b = matvec_gf2(B.T, matvec_gf2(C, a))

A1 = A[:k]
A2 = A[k:]

##Fonctions
#fonction h
h_table = rng.integers(0, 2, size=1 << k, dtype=np.uint8)

def h(v): #Accès à un élément de h
    return int(h_table[bits_to_int(v)])

def f_bit(x_bits): # Fonction f
    u = matvec_gf2(A1, x_bits)
    v = matvec_gf2(A2, x_bits)
    return parity(u & v) ^ parity(x_bits & a) ^ h(v)

def g_bit(y_bits): #Fonction g
    u = matvec_gf2(B1, y_bits)
    v = matvec_gf2(B2, y_bits)
    term1 = parity(u & v)
    term2 = parity(y_bits & b)
    term3 = h(u ^ matvec_gf2(B1, a))
    term4 = parity(matvec_gf2(B2, a) & matvec_gf2(B2, a))
    return term1 ^ term2 ^ term3 ^ term4

def f_sign(x_bits): # Fonction f signée
    return 1 - 2 * f_bit(x_bits)

def g_sign(y_bits): # Idem
    return 1 - 2 * g_bit(y_bits)
```

```
# Représentation finale
F = np.empty(N, dtype=np.float64)
G = np.empty(N, dtype=np.float64)

for x in range(N):
    xb = int_to_bits(x, n)
    F[x] = f_sign(xb)

for y in range(N):
    yb = int_to_bits(y, n)
    G[y] = g_sign(yb)

# 1/2 chance de valoir 1, 1/2 de valoir -1
print("forrelation =", forrelation(F, G))
```

### 5.3 Preuve de la proposition 3

*Proof.* $B$ est de rang maximal, par invariance du rang par inverse et transposition.
Donc $Ba \sim \mathcal{U}(\mathbb{F}_2^n)$, car $a \sim \mathcal{U}(\mathbb{F}_2^n)$ et $B$ représente une opération bijective.
On en déduit que $u_1, u_2 := B_1 a, B_2 a \sim \mathcal{U}(\mathbb{F}_2^{n/2})$.
De plus, $u_1$ et $u_2$ sont indépendants, d'après la déduction ci-dessus.
On a alors : $\langle B_1 a, B_2 a \rangle = \sum_{i=1}^{n/2} (u_1)_i (u_2)_i$. On remarque également que $\mathbb{P}[(u_1)_i (u_2)_i = 1] = \frac{1}{4}$, d'où $(u_1)_i (u_2)_i \sim \mathcal{B}(\frac{1}{4})$.
On en conclut : $\mathbb{P}[\langle B_1 a, B_2 a \rangle \text{ pair}] = \frac{1+(\frac{1}{2})^{n/2}}{2} \geq \frac{1}{4}$ et
$\mathbb{P}[\langle B_1 a, B_2 a \rangle \text{ impair}] = 1 - \frac{1+(\frac{1}{2})^{n/2}}{2} \geq \frac{1}{4}$. □