Attaques sur les codes QC-MDPC

Nous avons vu précédemment que l'utilisation des codes quasi-cycliques MDPC dans le cryptosystème de McEliece permet de construire un schéma de chiffrement post-quantique dont les clés ont une taille raisonnable. Cependant, ces codes étant très bien structurés, ils sont vulnérables aux attaques de types structurelles. Ces attaques permettent aux cryptanalystes de retrouver la clé secrète en vue de déchiffrer le message envoyé. La principale faille de ces types d'attaques est l'algorithme de décodage *bit-flipping*.

Nous allons, dans ce chapitre, faire une étude exhaustive de l'attaque de récupération de clé de Guo-Johansson-Stankovski ainsi que de l'attaque temporelle sur les codes quasi-cycliques MDPC.

3.1. Attaque de Guo-Johannson-Stankovski

L'attaque de récupération de clé dans [42] sur les codes QC-MDPC est une attaque de réaction reposant sur le fait que lors de l'étape du déchiffrement, le décodage itératif peut échouer avec une faible probabilité. Les auteurs ont ainsi identifié une dépendance entre la clé secrète et l'échec du décodage. Cette attaque suppose seulement qu'un adversaire est en mesure de dire quand une erreur s'est produite, par exemple parce qu'une demande de renvoi a été effectuée. Elle peut être utilisée pour construire ce que l'on appelle le spectre de distance pour la clé secrète. L'attaque se déroule en deux étapes. La première étape consiste à calculer le spectre de distance de la clé secrète (ou d'une partie de la clé secrète), basée sur l'observation d'un grand nombre de vecteurs d'erreur qui ont abouti à un échec du décodage. La deuxième étape consiste à reconstruire la clé secrète en fonction du spectre de distance.

3.1.1. Spectre de distance

Définition : Le spectre de distance d'un vecteur $h \in \mathbb{F}_2^r$, noté $d(h)$, est l'ensemble des distances θ telles qu'il existe deux bits de h non nuls à la distance θ. Les distances étant comptées de façon cyclique.

$$d(h) = \left\{ \begin{array}{l} \\ \theta : 1 \leq \theta \leq \left\lceil \frac{r}{2} \right\rceil, \exists (i,j), h_i = h_j = 1, \\ \\ \end{array} \begin{array}{r} 0 \leq i < j < r, \\ \\ \min\{j-i, r-(j-i)\} = \theta, \end{array} \right\}$$

Exemple : considérons le vecteur $h = 1001000001$, avec $r = 10$ et $w = 3$. On peut remarquer que $d(h) = \{1,3,4\}$. En effet, le premier et le dernier bit de h sont voisins (la distance étant comptée de manière cyclique), ce qui donne 1 comme spectre. Le premier et le quatrième bit sont distants de 3. Le quatrième et le dernier bit quant à eux sont distants de 4.

Notons que pour un décalage cyclique dans un sens comme dans l'autre du vecteur h, on retrouve le même spectre de distance.

3.1.2. Description de l'attaque

Le but de cette procédure d'attaque est de récupérer le premier circulant de la matrice secrète H, qui est désignée par H_0, de sa matrice publique G correspondante. Comme la matrice secrète H_0 est quasi-cyclique, il suffit de récupérer uniquement la première ligne de la matrice correspondante, h_0. L'idée principale de cette attaque est de vérifier le nombre d'échecs de décodage signalés en utilisant un certain ensemble de modèles d'erreurs différents. L'attaque génère ces schémas d'erreurs selon un certain critère.

L'idée clé est d'examiner la procédure de décodage pour différents modèles d'erreurs. L'attaque de récupération de la clé sécrète H fonctionne comme ainsi. Soit Ψ_d l'ensemble de tous les vecteurs binaires de longueur $n = 2r$, contenant exactement t nombre de ''1''. Ces t ''1'' sont placés en $\frac{t}{2}$ paires aléatoires avec une distance d entre eux sur la première moitié du vecteur d'erreur v choisi. La seconde moitié du vecteur d'erreur sera entièrement constituée de zéros, puisque l'attaque se concentre uniquement sur la récupération du premier circulant H_0. L'équation suivante donne une description mathématique de la génération de l'ensemble des vecteurs d'erreurs Ψ_d et du vecteur d'erreur v :

$$\Psi_d = \{v = (e, f) \mid w_H(f) = 0, et\ \exists\ s_1, s_2, \dots, s_t\ distinct, s.t.\ e_{s_i} = 1,$$

$$et\ s_{2i} = (s_{2i-1} + d) mod\ r\ pour\ i = 1, \dots, \frac{t}{2}\}.$$

Exemple :

Pour un code C de longueur $n = n_0 \times p = 2 \times 6 = 12$ et $t = 2$, les sous-ensembles Ψ_d suivants avec les modèles d'erreurs sont crées

$$\Psi_1 = \{110000\ 000000, 011000\ 000000, 001100\ 000000, 000110\ 000000,$$
$$000011\ 000000, 100001\ 000000\}$$

$$\Psi_2 = \{101000\ 000000, 010100\ 000000, 001010\ 000000, 000101\ 000000,$$
$$100010\ 000000, 010001\ 000000\}$$

$$\Psi_3 = \{100100\ 000000, 010010\ 000000, 001001\ 000000\}$$

Les sous-ensembles Ψ_d seront construits pour $d = 1, 2, \dots, \frac{r}{2}$. Les modèles d'erreur de ces sous-ensembles seront utilisés pour générer des informations statistiques sur le taux de réussite du décodage. Comme décrit précédemment, ce taux de réussite dépend de l'emplacement des bits d'erreur.

L'attaque effectuera M essais de décodage sur le code $QC - MDPC$, où chaque essai choisit un vecteur d'erreur à partir de Ψ_d. Cela sera fait pour $d = 1, \dots, \frac{r}{2}$, où $U = \frac{r}{2}$ est la limite supérieure. Ainsi, un total de $M \times U$ essais sera effectué, et une probabilité d'erreur de décodage peut être calculée pour chaque essai.

La motivation derrière cette attaque est qu'il existe une corrélation entre la probabilité d'erreur de décodage calculée et la fréquence d'occurrence d'une distance d dans le premier circulant H_0. En d'autres termes, plus cette distance se produit dans la matrice H_0, ou dans le vecteur h_0, plus la probabilité d'erreur de décodage calculée sera faible. Le nombre de fois que la distance se produit dans un vecteur donné est appelé multiplicité de cette distance dans ce vecteur, et est désignée par $\mu(d)$. Par conséquent, un profil de distance de h_0 peut être construit, où il donne toutes les distances disponibles en h_0, et leurs multiplicités correspondantes. Le profil de cette distance est indiqué par $D(h_0)$, et est donné par :

$$D(h_0) = \{d\ répété\ \mu(d) fois, pour\ d = 1, \dots, U\}$$

L'algorithme pour le calcul du spectre tel que présenté dans [42] est illustré ci-dessous :

Algorithme 8: Calcul du spectre de distance

Entrées : Les paramètres $n, r, w\ et\ t$ du code $QC - MDPC$ sous-jacent, le nombre d'essais M de décodage par distance.

Sorties : Le spectre de distance $D(h_0)$.

Pour toutes les distances d **faire**

Essayez de décoder les M essais en utilisant le modèle d'erreur conçu

Effectuer un test statistique pour décider de la multiplicité $\mu(d)$

Si $\mu(d) > 0$ **alors**

Ajouter d avec la multiplicité $\mu(d)$ au spectre de distance $D(h_0)$.

Exemple :

$d = 4$

$$\left.\begin{matrix} 1001 \\ 1001 \\ \vdots \\ 1001 \end{matrix}\right\} \frac{t}{2}, placés\ aléatoirements\ dans\ v \Longrightarrow v = [00100100001001 \cdots 001001 \cdots]$$

$$h_0 = [\cdots 000110100100 1100 \cdots]$$

$$\Longrightarrow \mu(d)_{h_0} = 3$$

Cet exemple montre la façon dont le vecteur d'erreur est généré lors de l'attaque et pour une distance donnée, ainsi que la façon dont la multiplicité de cette distance apparaîtra en h_0.

3.1.3. Reconstruction de la Clé secrète

Une fois que l'attaque est faite, et que $D(h_0)$ est construit avec succès, la reconstruction de h_0, et donc de H_0, peut être faite facilement. Tout d'abord, on place un ‘‘1’’ dans la première position du vecteur de reconstruction, qu'on notera h_{recons}. Un deuxième ‘‘1’’ est placé à une

position i_0, où i_0 est égal à la première distance en $D(h_0)$. Le troisième ‘‘1’’ est placé à une position i_1 quelconque et la distance entre celui-ci et les deux précédemment calculées. Si ces distances calculées apparaissent dans le profil de distance, alors la troisième est maintenue dans sa position; sinon, une nouvelle position est testée. Cette opération est répétée jusqu'à ce que les distances calculées, avec cette position i_1, apparaissent dans $D(h_0)$. La même procédure est répétée pour les quatrième, cinquième, sixième et ainsi de suite, où toutes les distances calculées entre la $n^{ième}$ et les $n-1$ ‘‘1’’ précédents doivent apparaître dans $D(h_0)$.

À la fin, h_0 sera entièrement reconstruit, et H_0 pourra alors être reconstruit en décalant cycliquement chaque entrée de h_0 sur toutes les lignes. Ensuite, le reste de la clé secrète H peut être reconstruit sous la forme H_0 en utilisant l'algèbre linéaire [26] [25].

Nous pouvons illustrer la procédure de reconstruction de la clé secrète grâce à l’algorithme suivant présenté dans [42].

Algorithme 9: Récupération de clé à partir du spectre de distance

---

Entrées : Le spectre de distance $D(h_0)$, la clé secrète partielle h_0 et la profondeur actuelle l
Sorties : La clé secrète h_0 récupérée ou le message "Il n’existe aucune clé secrète de ce type".
Paramètres récursives initiaux : Le spectre de distance $D(h_0)$, un ensemble vide pour la clé secrète, la profondeur actuelle 0.

Si $l = w$ **alors**
    Retourner h_0. //*clé secrète trouvée*
Pour toutes les clés potentielles de bits i **faire**
    Pour toutes les distances au bit clé i existent en $D(h_0)$ **faire**
        Ajouter le bit de clé i à la clé secrète h_0
        Effectuer un appel récursif avec les paramètres $D(h_0)$, h_0 et $l+1$
        Si l'appel récursif trouve la solution h_0 **alors**
            si h_0 est la clé secrète **alors**
                Retourner h_0. //*clé secrète trouvée*
        Supprimer le bit de clé i de la clé secrète h_0.

Retourner "Il n’existe aucune clé secrète de ce type".

Exemple :

Soit H_0 une matrice de contrôle de parité dont la première ligne notée h_0 comporte des 1 aux positions 2, 5, 7 et 10.

$$h_0 = [\,0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$$

On a alors : $b_2 = 1,\ b_5 = 1,\ b_7 = 1\ et\ b_{10} = 1$

Le spectre de distance $D(H_0)$ est constitué de toutes les distances d pour les quelles $1 \leq d \leq \frac{r}{2}$

$$D(H_0) = \{d(b_2, b_5), d(b_2, b_7), d(b_2, b_{10}), d(b_5, b_7), d(b_5, b_{10}), d(b_7, b_{10})\}$$

$$= \{3, 5, 4, 2, 5, 3\} = \{2, 3, 3, 4, 5, 5\}$$

Connaissant le spectre de distance $D(H_0)$, on peut reconstruire h_0.

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_3) = 2$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, 3, 3, 4, 5, 5\}$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_6) = 5 \quad \text{et} \quad d(b_3, b_6) = 3$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, \cancel{3}, 3, 4, \cancel{5}, 5\}$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_9) = 4, \quad d(b_3, b_9) = 6 \quad \text{et} \quad d(b_6, b_9) = 3$$

Pas toutes les distances $d \in D(H_0) \rightarrow echec\ du\ test$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ?\,]$$

$$d(b_1, b_{10}) = 3, \quad d(b_3, b_{10}) = 5 \quad \text{et} \quad d(b_6, b_{10}) = 4$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, \cancel{3}, \cancel{3}, 4, \cancel{5}, \cancel{5}\}$

On obtient ainsi : $h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$.

En faisant un décalage cyclique, on retrouve l'expression originale :

$$h_0 = [\,0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$$

Cette procédure peut être effectuée pour chaque matrice circulante $H_i.$ dans H. Après avoir obtenu tous les $h_i.$ on peut les combiner et construire la matrice secrète de contrôle de parité H.

Le nombre d'opérations binaires nécessaires pour rompre le système cryptographique QC-MDPC avec pour la sécurité 80-bit est estimée à 228 opérations. La décision difficile de Gallager L'algorithme de décodage était l'un des premiers algorithmes de décodage itératif pour les codes LDPC. Des algorithmes de décodage plus avancés avec de meilleures performances de décodage rendent cette la cryptanalyse pour trouver la clé privée. Il faut envoyer davantage de cryptogrammes pour obtenir suffisamment des informations statistiques pour construire le spectre de distance $D(H_i)$. Ces algorithmes de décodage d'augmenter le facteur de travail de cette cryptanalyse d'un facteur d'environ $2^{15.6}$ augmentant ainsi le facteur de travail total à $2^{43.6}$. Cependant, ce facteur de travail est encore trop faible pour que ce cryptosystème soit sécurisé.

La variante CCA-2, plus sûre, qui utilise essentiellement l'idée de brouiller les bits du message d'entrée m autour, peut également être cryptée avec cette attaque. Bien que l'attaque nécessite quelques modifications mineures et que le facteur de travail de cette variante soit plus élevé (opérations sur $2^{55.3}\ bits$), le niveau de sécurité est encore trop faible pour être d'une utilité pratique.

3.2. Attaque temporelle

L'attaque de GJS, dans [42], profite d'un minutieux calcul du taux d'échec au décodage pour les différentes classifications du vecteurs d'erreur. Pour la plupart des décodages une erreur se produit lorsqu'il faut plus qu'un certain nombre d'itérations. De ce fait, une attaque similaire peut être réalisée si un adversaire est capable de connaître le nombre d'itérations que l'algorithme effectue pour le décodage. Cela signifie que l'attaque du GJS peut être reformulée comme une attaque par canal auxiliaire très forte lorsque l'algorithme de décodage n'est pas en temps constant.

Et comme le poids du syndrome fait fuir des informations, tout paramètre corrélé à cette quantité pourrait être utilisé pour cette attaque. Un paramètre intéressant et souvent facile à mesurer est le temps.

L'algorithme de décodage des codes QC-MDPC est un algorithme itératif. Le nombre de tours nécessaires pour corriger les erreurs est variable. Cela a été étudié dans [5]. Pour les paramètres habituels pour 80 bits de sécurité, l'algorithme corrige généralement l'erreur en 3 tours, mais

dans certains cas, il en faut 4, 5 itérations, voire plus. Les mises en œuvre habituelles s'arrêtent après un certain nombre de tours (environ 10), c'est ce qui a été utilisé pour l'attaque de [42].

C'est ce qui a motivé Lequesne dans [3] à essayer de réaliser une attaque de synchronisation (attaque temporelle). Le scénario est le même que celui de l'attaque de GJS [42], mais au lieu d'observer le poids du syndrome, l'attaquant peut mesurer le nombre d'itérations nécessaires pour décoder son message. Pour obtenir le spectre, il utilise exactement le même algorithme de collecte de données : pour chaque distance dans le spectre, il calcule le nombre moyen d'itérations nécessaires pour corriger une erreur contenant cette distance. En moyenne, le nombre d'itérations est légèrement plus faible lorsque la distance apparaît dans le spectre de la clé. Cela fonctionne bien et il est possible de récupérer entièrement le spectre de distance avec 100 millions d'échantillons sur Schéma QC-MDPC de sécurité 80 bits [3].

Il s'agit de la première attaque de synchronisation sur le schéma QC-MDPC. Elle indique que nous avons besoin d'un décodage en temps constant, comme pour les QcBits [4], mais avec un algorithme de décodage.

Cette attaque a d'abord été testé sur une version simplifiée du système n'utilisant qu'un seul bloc, afin de le comparer au comportement attendu. Mais le résultat est frappant [3]. En utilisant les paramètres habituels pour une sécurité de 80 bits, avec cent mille échantillons, l'attaquant a pu récupérer tout le spectre et même les multiplicités, c'est-à-dire les distances qui apparaissent plusieurs fois dans la clé. Lorsque l'on pousse à un milliard d'échantillons, il n'est pas un lieu de confusion.

Lorsque l'expérience est effectuée sur le véritable schéma QC-MDPC avec deux blocs, on obtient des résultats similaires. L'attaque est effectuée sur chaque bloc séparément, c'est-à-dire que pour chaque modèle d'erreur, on ajoute le poids du syndrome sur les compteurs de toutes les distances présentes dans la première moitié de l'erreur pour récupérer le spectre du premier bloc. Comme il n'y a pas de corrélation entre les deux blocs la présence du deuxième bloc agit comme un bruit ajouté au poids du syndrome. La seule différence est donc qu'on aura besoin de plus d'échantillons pour réduire la variance et bien distinguer les distances dans le spectre. A noter qu'il est possible de calculer le spectre des deux blocs en même temps, de sorte qu'il n'est pas nécessaire de doubler le nombre d'échantillons pour récupérer le deuxième bloc.

Cette attaque a également été réalisée lorsqu'une autre erreur est ajoutée au syndrome, comme dans le schéma d'Ouroboros [6]. Encore une fois, cela n'ajoute qu'un bruit aléatoire et il est

possible de récupérer le spectre avec quelques millions d'échantillons pour les paramètres de sécurité 80 bits.

L’attaque temporelle est beaucoup plus rapide que l'attaque de réaction de GJS. Cette rapidité s’explique par les différences dans le nombre d'itérations qui sont beaucoup plus courantes que les erreurs de décodage. Cela permet d'obtenir plus d'informations sur les corrélations à collecter par itération.