Décalages cachés en cryptanalyse symétrique

Xavier Bonnetain María Naya-Plasencia

26 avril 2017

Inria — Équipe-projet SECRET

Plan

Introduction

Contexte

Cryptanalyse quantique

Attaquer des systèmes classiques avec un ordinateur quantique.

Oracle quantique

Requêtes en superposition

Modèle fort, mais non trivial

Problème de Simon

Le problème

Sous-groupe caché

$(\mathbb{G}, \cdot)$ un groupe

$f : \mathbb{G} \to \mathbb{G}$ telle que $f(x) = f(x \cdot s)$, trouver s.

Décalage caché

$(\mathbb{G}, \cdot)$ un groupe

$f, g : \mathbb{G} \to \mathbb{G}$ telles que $f(x) = g(x \cdot s)$, trouver s.

Simon

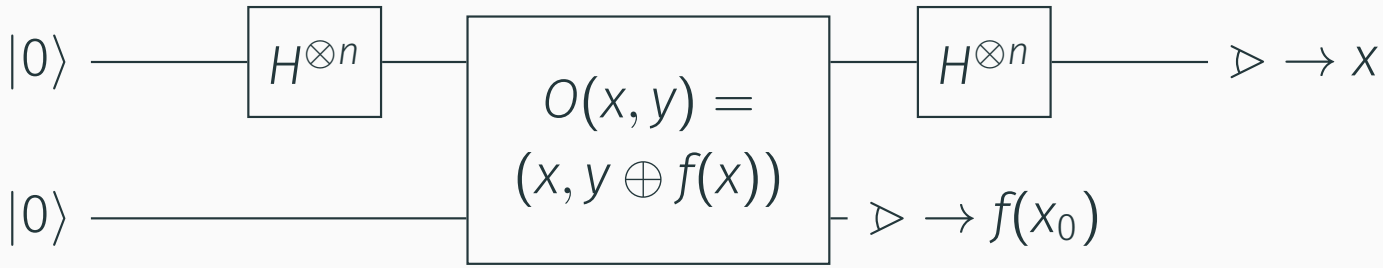
$(\mathbb{G}, \cdot) = (\mathbb{Z}_2^n, \oplus)$

Classiquement : Recherche de collision $(2^{n/2})$

Quantiquement : Algorithme de Simon [Sim94]

Algorithme de Simon [Sim94]

Circuit quantique



Routine quantique

- *Partir de* $|0\rangle\,|0\rangle$

Routine quantique

$$H \begin{cases} |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases}$$

- Partir de $|0\rangle\,|0\rangle$
- *Appliquer H, ce qui donne* $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle\,|0\rangle$

Routine quantique

$$H \begin{cases} |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \qquad O_f |x\rangle |0\rangle \longrightarrow |x\rangle |f(x)\rangle$$

- Partir de $|0\rangle |0\rangle$
- Appliquer H, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$
- *Appliquer O_f, ce qui donne* $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$

Routine quantique

$$H \begin{cases} |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \qquad O_f |x\rangle |0\rangle \longrightarrow |x\rangle |f(x)\rangle$$

- Partir de $|0\rangle |0\rangle$
- Appliquer H, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$
- Appliquer O_f, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$
- *Mesurer le second registre : on obtient $f(x_0)$, et on projette le premier registre sur les états compatibles, $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$*

Routine quantique

$$H \begin{cases} |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \qquad O_f |x\rangle |0\rangle \longrightarrow |x\rangle |f(x)\rangle$$

- Partir de $|0\rangle |0\rangle$
- Appliquer H, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$
- Appliquer O_f, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$
- Mesurer le second registre : on obtient $f(x_0)$, et on projette le premier registre sur les états compatibles, $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$
- *Réappliquer H, pour obtenir* $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} |y\rangle + (-1)^{(x_0 \oplus s) \cdot y} |y\rangle$

Routine quantique

$$H \begin{cases} |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{cases} \qquad O_f |x\rangle |0\rangle \longrightarrow |x\rangle |f(x)\rangle$$

- Partir de $|0\rangle |0\rangle$
- Appliquer H, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$
- Appliquer O_f, ce qui donne $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$
- Mesurer le second registre : on obtient $f(x_0)$, et on projette le premier registre sur les états compatibles, $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$
- Réappliquer H, pour obtenir $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} |y\rangle + (-1)^{(x_0 \oplus s) \cdot y} |y\rangle$
- *On réécrit l'état* $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle$

On mesure un élément tel que $1 + (-1)^{s \cdot y} \neq 0$, et donc un y_0 avec $y_0 \cdot s = 0$.

Algorithme de Simon [Sim94]

Algorithme au complet

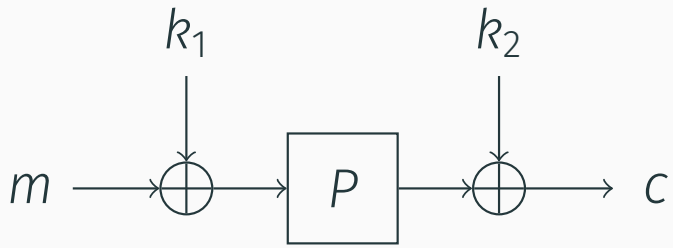
- Récupérer avec la routine quantique des éléments y_i vérifiant $y_i \cdot s = 0$, ce qui donne des équations linéaires sur les bits de s.
- Résoudre le système pour retrouver s.
- Un nombre d'équations linéaire en le nombre de bits suffit ($3n$).

Variante pour le décalage

On construit $F(b, x) = \begin{cases} f(x) & \text{si } b = 0 \\ g(x) & \text{si } b = 1 \end{cases}$, de période $(1, s)$ dans $\mathbb{Z}_2^{n+1}$.

Attaque sur Even-Mansour [KM12]

Schéma $c = E_{k_1,k_2}(m)$



Attaque

$f(x) = P(x) \oplus E_{k_1,k_2}(x)$ vérifie $f(x) = f(x \oplus k_1)$

Classique : $2^{n/2}$

Quantique : $3n$

Comment corriger ?

L'algorithme de Simon a besoin d'une structure en $\oplus$ dans la fonction.

Solution [AR17]

Remplacer les $\oplus$ par d'autres opérations, comme des multiplications de matrices, des compositions de permutations, ou des additions modulo 2^n.

Solution proposée

Approche

- Réductions des attaques vers le problème de sous-groupe caché
- Aucun algorithme polynomial n'est connu pour ces structures
- Donc la construction est sûre !

Cependant

- Pas de paramètre chiffré
- Pas d'évaluation des attaques

Problème de Kuperberg

Décalage caché dans $\mathbb{Z}_n$

$f(x) = g(x + s)$

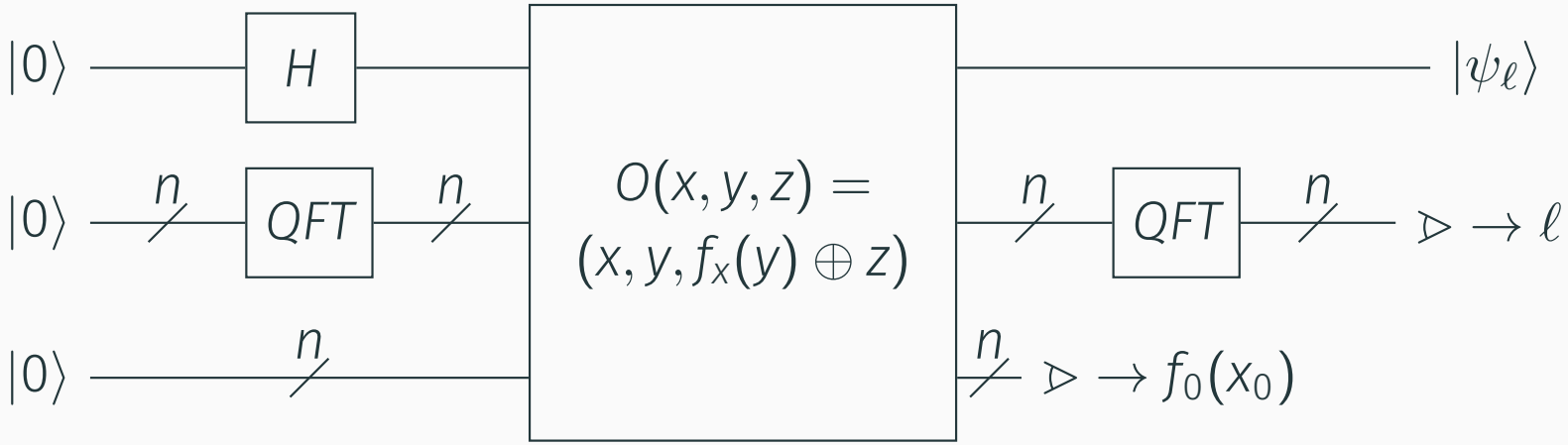
Algorithmes

- Conçu en 2003 par Kuperberg [Kup03]
- Variantes en 2004 [Reg04] et 2013 [Kup13]

Algorithme de Kuperberg

$f_0(x) = f_1(x + s)$

Routine quantique



$|\psi_\ell\rangle = |0\rangle + \chi\left(\frac{s\ell}{2^n}\right)|1\rangle$

Combinaisons de qbits

$|\psi_\ell\rangle = |0\rangle + \chi\left(\frac{s\ell}{2^n}\right)|1\rangle$

$|\psi_{2^{n-1}}\rangle = |0\rangle + (-1)^s |1\rangle$: donne la parité de s

$\text{CNOT}\,|\psi_\ell\rangle\,|\psi_m\rangle = |\psi_{\ell+m}\rangle\,|0\rangle + \chi\left(\frac{sm}{2^n}\right)|\psi_{\ell-m}\rangle\,|1\rangle$

Oublions les qbits

Problème

- On peut générer $\ell \in \mathbb{Z}_{2^n}$
- On peut faire $(\ell, m) \to \ell + m$ ou $\ell - m$
- On cherche 2^{n-1}

Remarque

Si $\ell = 2^i(1 + 2\ell')$ et $m = 2^i(1 + 2m')$,

$m + \ell = 2^{i+1}(1 + m' + \ell')$

$m - \ell = 2^{i+1}(m' - \ell')$

En combinant des éléments qui ont la même divisibilité par 2, on augmente celle du résultat.

Algorithme de Kuperberg, sans qbit

Générer N nombres aléatoires dans $\mathbb{Z}_{2^n}$
les séparer en groupes P_i d'éléments divisibles par 2^i et non par 2^{i+1}

for i := 0 to $n-2$ **do**
    while $|P_i| \geq 2$ **do**
        Prendre (a, b) dans P_i où $a+b$ ou $a-b$ a la plus grande divisibilité par 2 possible (et n'est pas 0)
        c est tiré aléatoirement dans $\{a+b, a-b\}$
        Insérer c dans le P_j idoine
    end while
end for
if $P_{n-1} \neq \emptyset$ **then return** Found
else return Failure
end if

Récupérer les autres bits

Réducibilité

Une fois qu'on a $s_0 = s \mod 2$, on peut construire $f'(x) = f(2x)$ et $g'(x) = g(2x + s_0)$, qui ont pour décalage $s' = (s - s_0)/2$, dans $\mathbb{Z}_{2^{n-1}}$.

Autre méthode

$|\psi_{2^{n-1}}\rangle$, de phase $\frac{s}{2}$, donne s_0.

$|\psi_{2^{n-2}}\rangle$ est de phase $\frac{s}{4} = \frac{s'}{2} + s_0 \frac{2^{n-2}}{2^n}$.

$|\psi_{2^{n-1}+2^{n-2}}\rangle$ est de phase $\frac{s}{4} = \frac{s'}{2} + s_0 \frac{2^{n-1}+2^{n-2}}{2^n}$.

Si $s_0 = 0$, on peut directement récupérer s_1 grâce à un $\ell = 2^{n-2} \mod 2^{n-1}$, sinon, il faut d'abord effectuer une rotation d'angle $-2\pi\frac{\ell}{2^n}$, et ainsi de suite. On cherche donc une suite d'éléments ℓ_i vérifiant $\ell_i = 2^i \mod 2^{i+1}$.

Récupérer les autres bits

On veut obtenir les nombres

```
1000…0
*100…0
**10…0
   …
***…*1
```

qui permettent de récupérer 1 bit de clé si on a récupéré tous les précédents.

Notre version

Générer N nombres aléatoires dans $\mathbb{Z}_{2^n}$
les séparer en groupes P_i d'éléments divisibles par 2^i et non par 2^{i+1}

for i := 0 to $n-2$ **do**
    while $|P_i| \geq 3$ **do**
        Prendre (a, b) dans P_i où $a+b$ ou $a-b$ a la plus grande divisibilité par 2 possible (et n'est pas 0)
        c est tiré aléatoirement dans $\{a+b, a-b\}$
        Insérer c dans le P_j idoine
    end while
end for
if $\forall i, P_i \neq \emptyset$ **then return** Found
else return Failure
end if

Estimation de la complexité

Ancien résultat [Kup03]

Complexité en $\tilde{O}(2^{1.78\sqrt{n}})$ pour un taux de réussite constant, avec une ébauche de preuve.

En simulant

Complexité approchée en requêtes en $0.7 * 2^{1.8\sqrt{n}}$, pour 90% de réussite.

Résumé

Décalage caché

$(\mathbb{G}, \cdot)$ un groupe

$f, g : \mathbb{G} \to \mathbb{G}$ telles que $f(x) = g(x \cdot s)$, trouver s.

$(\mathbb{G}, \cdot)$	Coût	Coût classique
$(\mathbb{Z}_2^n, \oplus)$	$3n$	$2^{n/2}$
$(\mathbb{Z}_{2^n}, +)$	$2^{1.8\sqrt{n}}$	$2^{n/2}$

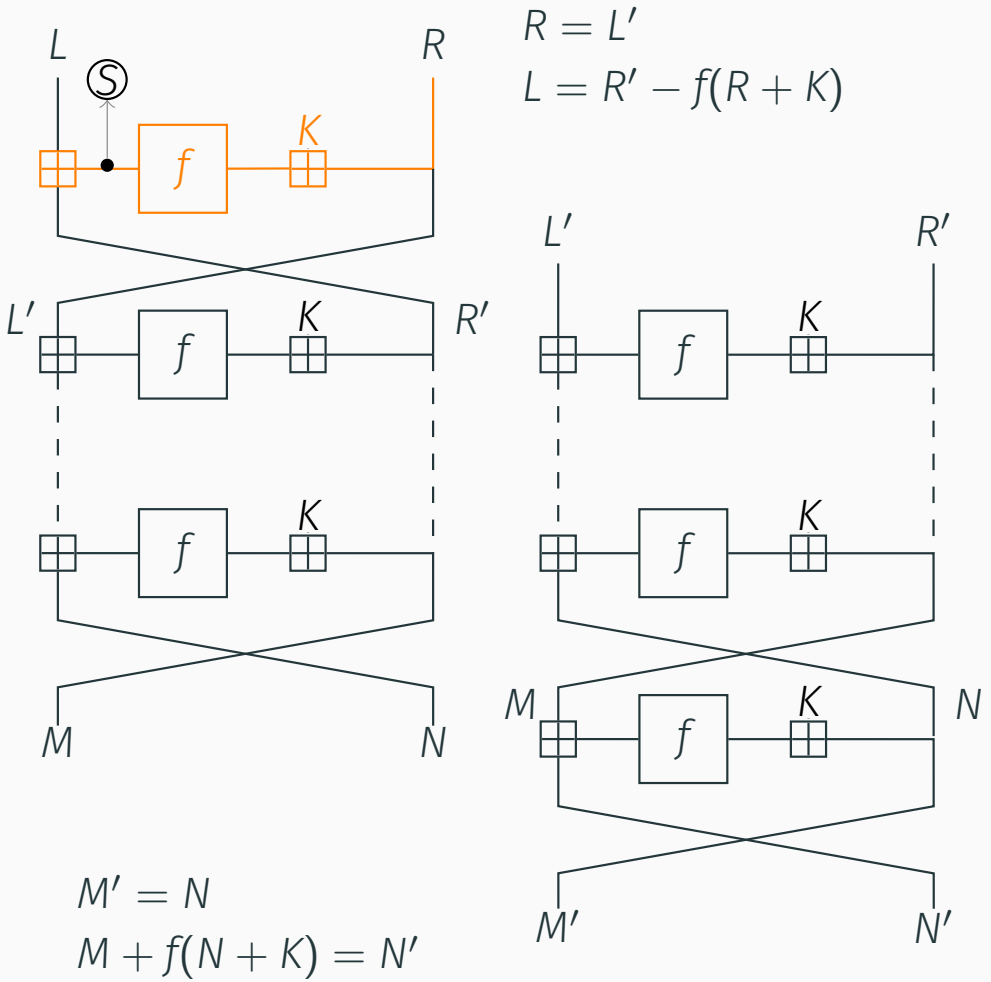
Exemple : 128 bits

$(\mathbb{Z}_2^{128}, \oplus)$: $2^{8.6}$

$(\mathbb{Z}_{2^{128}}, +)$: $2^{20.4}$

Applications en cryptanalyse

Feistel à une clé



R_0 fixé.

$E(x, R_0)_{|R} =$

$E(R_0, x - f(R_0 + K))_{|L}$

Coût

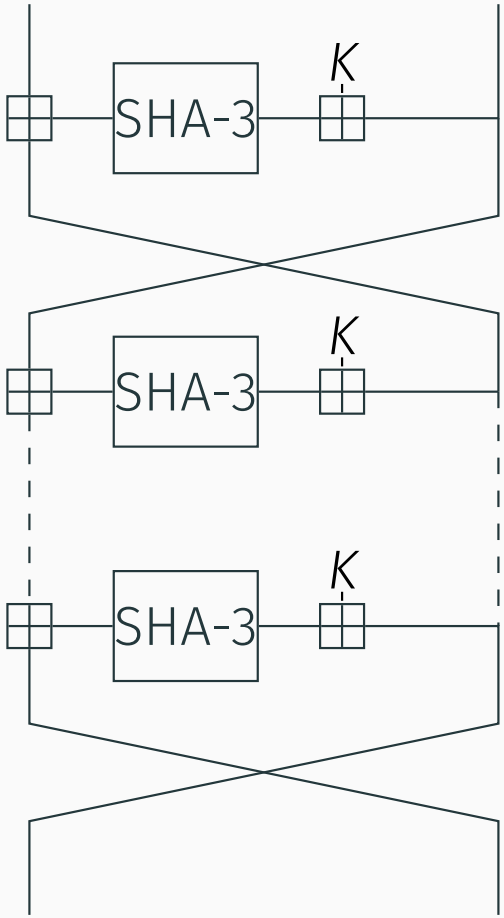
$\oplus$: $1.5n$

$+$: $2^{1.2\sqrt{n}}$

Class. : $2^{n/4}$

Plus inversion de f.

f non inversible



A partir de R_0, on récupère $f(R_0 + K)$, et on connaît f.

C'est un décalage caché !

Récupération de K

On peut construire un oracle $|x\rangle\,|0\rangle \mapsto |x\rangle\,|f(x+K)\rangle$ avec l'algorithme de Simon ou Kuperberg, puis le réappliquer pour trouver K. Les coûts se multiplient.

Coût total

f facile à inverser

clé	branche	Coût	Taille de blocs pour 128 bits de sécurité
$\oplus$	$\oplus$	$1.5n$	2^{128}
$\oplus$	$+$	$2^{1.2\sqrt{n}}$	11500
$+$	$\oplus$	$1.5n$	2^{128}
$+$	$+'$	$2^{1.2\sqrt{n}}$	11500

f non inversible

clé	branche	Coût†	Taille de blocs pour 128 bits de sécurité
$\oplus$	$\oplus$	$3n^{2*}$	2^{63}
$\oplus$	$+$	$2n2^{1.2\sqrt{n}}$	9000
$+$	$\oplus$	$2n2^{1.2\sqrt{n}}$	9000
$+$	$+'$	$2^{2.4\sqrt{n}*}$	2850

† La partie branche se fait uniquement quantiquement.

*De meilleures slide attacks existent si les lois sont les mêmes.

Bonus : Attaque sur AEZ 4.1

3 sous-clés I, J, L.

Recherche de collisions [CG16]

Fonction	Propriété
$f_1(x) = \texttt{AEZ-core}(K, (\tau), (0, x, 0, x, 0, 0))$	$f_1(x \oplus I) = f_1(x)$
$f_2(x) = \texttt{AEZ-prf}(K, (\tau, N, x, x), \tau)$	$f_2(x \oplus J) = f_2(x)$
$f_2(x) = \texttt{AEZ-prf}(K, (\tau, x, x), \tau)$	$f_3(x \oplus L) = f_3(x)$

C'est une période cachée !

Avec l'algorithme de Simon, on retrouve chaque clé en $\simeq 400$ queries.

On peut donc retrouver toute la clé en $2^{10.2}$ requêtes, autant en temps, et un coût en données de $2^{16} \ll 2^{48}$ octets.

Conclusion

Conclusion

Un algorithme efficace

Non polynomial, mais bien moins coûteux que la norme en symétrique.

Attention aux structures

Avec des $\oplus$, mais aussi des $+$.

Questions ?

References i

- Gorjan Alagic and Alexander Russell, *Quantum-secure symmetric-key cryptography based on hidden shifts*, Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III (Jean-Sébastien Coron and Jesper Buus Nielsen, eds.), Lecture Notes in Computer Science, vol. 10212, 2017, pp. 65–93.
- Colin Chaigneau and Henri Gilbert, *Is AEZ v4.1 sufficiently resilient against key-recovery attacks ?*, IACR Trans. Symmetric Cryptol. **2016** (2016), no. 1, 114–133.
- H. Kuwakado and M. Morii, *Security on the quantum-type Even-Mansour cipher*, Information Theory and its Applications (ISITA), 2012 International Symposium on, Oct 2012, pp. 312–316.

References ii

- Greg Kuperberg, *A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem*, SIAM J. Comput. **35** (2003), no. 1, 170–188.
- _____, *Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem*, 8th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2013, May 21-23, 2013, Guelph, Canada (Simone Severini and Fernando G. S. L. Brandão, eds.), LIPIcs, vol. 22, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, pp. 20–34.
- Oded Regev, *A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space*, CoRR (2004).

References iii


Daniel R. Simon, *On the Power of Quantum Cryptography*, 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994, IEEE Computer Society, 1994, pp. 116–123.