borne BCH, on en déduit que si $c(x) = (c_1, \cdots, c_n)$ est un mot du code, alors $H.c^T = 0$ implique $c(\alpha^i) = 0$ pour $1 \leqslant i \leqslant n-k$. Comme $RS_k$ a pour dimension k, cela montre que $RS_k$ est un code cyclique de zéros $\alpha, \alpha^2, \cdots, \alpha^{n-k}$ et donc de polynôme générateur $g(x) = (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{n-k})$.

**7.14.** Le sous-code sur le sous corps binaire de $RS_{n-\delta-1}$ est linéaire et cyclique puisque le code $RS_{n-\delta-1}$ est cyclique. D'après l'exercice précédent, ce code a dans l'ensemble de ses zéros $\alpha, \alpha^2, ..., \alpha^{\delta-1}$. Soit un ensemble de définition T qui contient au moins $\{1, 2, ..., \delta-1\}$. Comme le code est binaire cyclique, l'ensemble de définition contient aussi tous les éléments des 2-classes cyclotomiques modulo n associées aux éléments de l'ensemble de définition, donc $C_1 \cup ... \cup \mathbb{C}_{\delta-1} \subset T$. Réciproquement, tout mot de code binaire ayant pour zéros $\alpha, \alpha^2, ..., \alpha^{\delta-1}$ (soit le code BCH binaire avec les mêmes zéros) est contenu dans le sous-code sur le sous-corps. On peut donc en déduire que le sous-code sur le sous-corps de $RS_{n-\delta-1}$ est le code BCH binaire au sens strict de distance construite $\delta$.

**7.15.** Si $l = 1$, le décodage en liste peut décoder jusqu'à $\tau < \frac{n}{2} - \frac{1}{2}(k-1)$ erreurs, ce qui correspond au cas du décodage classique, il n'y a pas d'amélioration. Si $l = 2$, alors, pour avoir $\tau > (n-k+1)/2$, il faut $\frac{2}{3}n - k + 1 > (n-k+1)/2$, soit après simplification, $k/n \leqslant 1/3 + 1/n$.

## Chapitre 8

**8.1.** 1) On fait un tableau avec, en entrée, pour les lignes et les colonnes 00, 01 et 10 correspondant aux clés et aux clairs possibles. On voit que le chiffré 00 a 1/3 de chances d'arriver (pour des entrées et clés aléatoires) et les autres 2/9, donc il y a une fuite d'information et le fait de connaître le chiffré donne un biais sur le clair. 2) On écrit les trois possibilités sous la forme $\{0, 1, 2\}$ et les clés aussi. Le chiffrement est la somme modulo 3. De cette façon, tout est symétrique et il n'y a pas de fuite d'information.

**8.2.** 1) On attaque sur toutes les initialisations possibles ; $R_1$ est de longueur plus petite donc on aura moins d'essais à faire. On a $z = x_1$ dès que $x_2 = 0$ ou $x_3 = 0$, soit 3/4 du temps.
2) On essaie les huit initialisations possibles, l'initialisation 101 donne 101001110, qui a une corrélation proche de 3/4.

**8.3.** 1) On a $d = d_p + k(p-1)$ pour $k \in \mathbb{Z}$. Alors, $S \pmod p = m^d = m^{d_p + k(p-1)} \pmod p = m^{d_p}(m^{p-1})^k \pmod p$. Comme p est premier, $m^{p-1} = 1 \pmod p$, ce qui donne le résultat.
2) On peut vérifier directement l'égalité ou la retrouver par le théorème des restes chinois puisque l'on connaît $S \pmod q$ et $S \pmod p$.
3) Pour retrouver S, il suffit alors de calculer $S_p$ et $S_q$. Si l'on prend S avec n bits, p et q auront à peu près n/2 bits. Le calcul de S se fait en $\mathcal{O}(n^3)$ et celui de $S_p$ et $S_q$ en $\mathcal{O}(\frac{n^3}{8})$ chacun. On fait deux calculs, donc la complexité pour calculer $S_p$ et $S_q$ est en $\mathcal{O}(\frac{n^3}{4})$. Comme retrouver S a simplement le coût d'une multiplication, on gagne un facteur 4.

**8.4.** Le chiffré $c_1$ vérifie $c_1 = m^3 \pmod{N_1}$ ; de même, $c_2 = m^3 \pmod{N_2}$ et $c_3 = m^3 \pmod{N_3}$. Par les restes chinois, on peut donc retrouver $m^3 \pmod{N_1N_2N_3}$. Comme $m^3 < N_1N_2N_3$, on peut enlever le modulo et retrouver m.

**8.5.** 1) On a $g = aN + 1$. Si $B \equiv g^x \pmod{N^2}$ alors, par le développement avec le binôme de Newton, on obtient $(N+1)^x = 1 + xaN + N^2(P(N))$, où P est un certain polynôme en N. Comme on travaille modulo $N^2$, on obtient $g^x \pmod{N^2} \equiv 1 + xaN \pmod{N^2} \equiv B$, d'où $(B-1) = xaN + kN^2$ pour $k \in \mathbb{Z}$. Il en découle que $x \equiv a^{-1}\frac{(B-1)}{N} \pmod N$.
2) $C^{\phi(N)} \pmod{N^2} \equiv g^{\phi(N)m}h^{N\phi(N)} \pmod{N^2}$. Mais, comme $\phi(N^2) \equiv N\phi(N)$, $h^{N\phi(N)} \equiv 1 \pmod{N^2}$, on obtient $C^{\phi(N)} \equiv g^{\phi(N)am} \pmod{N^2}$. Or $\phi(N)a$ est inversible mod $N^2$ donc on peut appliquer le 1).
3) $\mathrm{Chiffré}(x) = g^x h_x^N$ et $\mathrm{Chiffré}(y) = g^y h_y^N$ donc $\mathrm{Chiffré}(x+y) = g^{x+y}(h_x h_y)^N$ ($h_x h_y$ peut être considéré comme un h aléatoire). De même, $\mathrm{Chiffré}(ux) = (g^x)^u (h_c^u)^N$.

**8.6.** On désigne un leader qui choisit un secret a. Il reçoit de chacun des membres du groupes M le nombre $x = g^{b_M^{-1}} \pmod p$ et renvoie $y = x^a \pmod p$. Chaque membre du groupe calcule alors $y^{b_M} \pmod p$ et obtient le même $g^a \pmod p$.

**8.7.** 1) $2 \times 256 \times 256 = 16$ kilobits. C'est beaucoup plus gros que les autres tailles de clés. 2) Pour casser le schéma, il faut être capable d'inverser la fonction de hachage pour les valeurs liées à la clé. C'est a priori infaisable d'après les propriétés de la fonction de hachage. 3) Non, car il n'y aurait pas assez de signatures possibles ($2^k$). 4) À chaque fois que l'on donne une signature, on donne des antécédents pour les valeurs de la signature. Si l'on prend deux vecteurs $a = (a_1, \cdots, a_k)$ et $b = (b_1, \cdots, b_k)$ tels que $a_i \neq b_i$, alors on peut obtenir tous les antécédents et l'on est capable de signer.

**8.8.** 1) On a bien $g^y A^r = g^{k+ar}.g^{-ar} = g^k = K \pmod p$.