TD - Construction d'une fonction de hachage

I. Construction d'une fonction de hachage : $\{0,1\}^{2m} \longrightarrow \{0,1\}^m$

1. *$p-1=2q$ et q est premier donc ses diviseurs sont $\{1,2,q,2q=p-1\}$.*
Comme d est un diviseur de $p-1$, on en déduit $d \in \{1,2,q,p-1\}$.

2. *Comme $0 \leq x_2, x_4 \leq q-1$, on a $-(q-1) \leq x_4 - x_2 \leq q-1$.*
Or q est premier; on en déduit que (x_4-x_2) est premier avec q, donc $d \neq q$.

3. *Evident :* $\alpha^{x_1}\beta^{x_2} \equiv \alpha^{x_3}\beta^{x_4} \mod p \Longleftrightarrow \alpha^{(x_1-x_3)} \equiv \beta^{(x_4-x_2)} \mod p$

4. *Si $d=1$, soit $u = (x_4-x_2)^{-1} \mod (p-1)$: $u.(x_4-x_2) = 1+k.(p-1)$ Alors* $\beta^{(x_4-x_2).u} \mod p \equiv \beta^{1+k(p-1)} \mod p \equiv \beta \mod p$ *(d'après le théorème de Fermat).*
Remplaçant dans 3., on obtient : $\beta = \alpha^{(x_1-x_3).u} \mod p$ *soit $\lambda = (x_1-x_3).u \mod p-1$, cqfd.*

5. **5.a.** *Comme $d=2$ et $p-1=2.q$, on a x_4-x_2 premier avec q d'où $u.(x_4-x_2) = 1+k.q$.*
D'où $\beta^{(x_4-x_2).u} \mod p \equiv \beta^{1+kq} \mod p \equiv \beta.(\beta^q)^k \mod p$. *Or $q = \frac{p-1}{2}$ et β est un primitif mod p.*
D'où $\beta^{p-1} = 1 \mod p$ *et* $\beta^q = \beta^{\frac{p-1}{2}} = -1 \mod p$. *Finalement* $\beta^{(x_4-x_2).u} = (-1)^k.\beta \mod p$, *cqfd.*
5.b. *Remplaçant dans 3., on obtient :* $\beta = \pm\alpha^{(x_1-x_3).u} \mod p$ *soit* $\beta = \alpha^{(x_1-x_3).u+\delta.q} \mod p$ *avec $\delta \in \{0,1\}$. On a donc $\delta = 0$, i.e. $\lambda = u.(x_1-x_3) \mod p-1$ ou sinon $\delta = 1$, i.e. $\lambda = u.(x_1-x_3)+q \mod p-1$, cqfd.*

6. *D'après les questions précédentes, on a l'algorithme suivant :*

```
AlgoCalculLogBeta( p, α, β, ; x1, x2, x3, x4 ) {
  q = (p-1)/2;
  d = pgcd(x4 - x2, p-1) ;
  if (d == 1) {
        u = (x4 - x2)^-1 mod (p-1);
        λ = (x1 - x3).u mod p-1;
  }
  else {// ici d == 2
        u = (x4 - x2)^-1 mod q;
        λ = (x1 - x3).u mod p-1;
        if (ExpoMod( α, λ, p ) == -β) λ = λ + q ;
  }
  return λ ;
}
```

Le coût se ramène à des opérations modulo $p-1$, p et q. Donc en $O(\log^{1+\epsilon} p)$, ce qui est faible même pour p grand (1024 bits par exemple). Autrement dit, si on connaît une collision pour h_1, alors on peut facilement calculer le logarithme discret de β, ce qui contredit l'hypothèse que λ est très coûteux à calculer.
Par suite, on en déduit que, sous l'hypothèse que λ est très coûteux à calculer, alors h_1 est résistante aux collisions.

II. Extension à une fonction de hachage : $\{0,1\}^* \longrightarrow \{0,1\}^m$

7. *On a*

$$\begin{array}{rccl} h_2: & (\{0,1\}^m)^4 & \to & \{0,1\}^m \\ & (x_1,x_2,x_3,x_4) & \mapsto & h_1(h_1(x_1,x_2),h_1(x_3,x_4)) \end{array}$$

8. *Si on a une collision sur h_2, alors $\exists x \neq y : h_2(x) = h_2(y)$. On a deux cas:*

- *soit $h_1(x_1,x_2) \neq h_1(y_1,y_2)$ ou $h_1(x_3,x_4) \neq h_1(y_3,y_4)$: alors comme $h_1(x_1,x_2),h_1(x_3,x_4)) = h_1(y_1,y_2),h_1(y_3,y_4))$ on a trouvé une collision sur h_1.*
- *Sinon, supposons par exemple $(x_1,x_2) \neq (y_1,y_2)$ (on peut s'y ramener par symétrie). Alors comme $h_1(x_1,x_2) = h_1(y_1,y_2)$, on a trouvé une collision sur h_1.*

Comme on suppose que h_1 est résistante aux collisions, on en déduit h_2 résistante aux collisions.

9. *Par récurrence, on montre que si h_i est résistante aux collisions, alors h_{i+1} est aussi résistante aux collisions.*
Ceci est vrai pour $i = 1$. Similairement à la question précédente, on montre que si on trouve une collision pour h_{i+1}, alors on construit aussi une collision pour h_i.
Comme h_1 est résistante aux collisions par hypothèse, alors h_i est résistante aux collisions pour tout $i \geq 2$.

10. *Soit $C(i)$ le nombre d'appels à h_1 lors de l'exécution de h_i. On a $C(i) = 2.C(i-1)+1 = 2^i.C(0) + \sum_{k=0}^{i-1} 2^k = 2^i - 1$.*
Pour un mot de n bits, on appelle donc n/m fois h_1, dont le coût est en $O(m^{1+\epsilon})$. Par suite le coût est en $O(n.m^\epsilon)$ ce que l'on peut assimiler à $O(n)$ puisque m est fixé donc constant.

11. *Si n est la taille du message, soit i tel que $2^i.m = n$, i.e. $i = \lceil log_2 \frac{n}{m} \rceil$. On hache le message avec h_i.*
Ce procédé nécessite de connaître la taille du message, donc ne marche pas à la volée. Une autre alternative, plus efficace, est d'utiliser le protocole de Merkle-Damgard comme vu en cours.