ISEC - TD 7
2024

Énoncé

Un *algorithme de mise en gage* (« *commitment scheme* ») est un algorithme $\textsc{Commit}(r,b)$ qui reçoit en argument un nombre spécifié de bits aléatoires r et un bit b. Il produit une « mise en gage » (appelons-la $\mathcal{C}$) de b, c'est-à-dire une chaine de bits dans $\{0,1\}^k$.

Pour mettre en gage un bit b, Alice calcule $\mathcal{C} \leftarrow \textsc{Commit}(r,b)$ puis envoie $\mathcal{C}$ à Bob, où r est un nonce aléatoire. Plus tard, Alice va « ouvrir »la mise en gage pour démontrer qu'elle l'avait effectuée correctement. Pour cela, Alice envoie b et r à Bob. Celui-ci peut alors calculer $\textsc{Commit}(r,b)$ et vérifier que la valeur transmise auparavant était correcte. Un schéma de mise en gage doit avoir deux propriétés :

— Il doit être « *binding* » : une mise en gage de 0 ne doit pas pouvoir être ouverte comme une mise en gage de 1 (et vice-versa).

— Il doit être « *hiding* » : les mises en gage de 0 sont indistinguables des mises en gage de 1.

Ces deux propriétés peuvent être obtenue de manière *calculatoire* (briser la propriété nécessite un énorme calcul) ou bien *absolue* (*statisticaly*, *information-theoretically*) (briser la propriété n'est pas possible même avec une puissance de calcul illimitée).

Exercice 1 : Notions de sécurité

1.a] Montrer qu'un schéma de mise en gage est *information-theoretically binding* si et seulement s'il n'existe pas de paire r, r' telle que $\textsc{Commit}(r,0) = \textsc{Commit}(r',1)$.

1.b] Montrer qu'un schéma de mise en gage est *statistically hiding* si et seulement pour tout $y \in \{0,1\}^k$, il existe une de paire r, r' telle que $y = \textsc{Commit}(r,0) = \textsc{Commit}(r',1)$.

1.c] Déduisez-en qu'il ne peut pas être les deux à la fois.

Exercice 2 : Mise en gage dérivée du chiffrement Elgamal

On considère l'algorithme de mise en gage défini par :

$$\textsc{Commit}(r,m) := (g^r, mh^r)\,,$$

où $r \in \mathbb{Z}_q$ est aléatoire, q est l'ordre du générateur g, $m \neq 0$ est le message à mettre en gage (on peut mettre en gage n'importe quel $m \in \mathbb{Z}_p^\times$, et pas juste un bit) et h est un autre générateur du groupe engendré par g modulo p ($h = g^x$ pour un certain x et h a le même ordre que g). On n'essaiera pas de mettre en gage $m = 0$, pour des raisons évidentes.

2.a] Justifier que ceci est *perfectly binding.*

2.b] Est-ce *computationally hiding* ?

Exercice 3 : Mise en gage dérivée d'un chiffrement à clef publique

La construction de l'exercice précédent peut se généraliser : pour mettre en gage une valeur m, on la chiffre avec un algorithme à clef publique. Pour ouvrir la mise en gage, on révèle l'aléa utilisé lors du chiffrement.

3.a] Justifier que ceci est *perfectly binding.*

3.b] Quelle propriété doit *computationally hiding* ?

Exercice 4 : Mise en gage de Pedersen

On considère le schéma de mise en gage inventé par Torben Pryds Pedersen en 1991 :

$$\textsc{Commit}(r, m) := g^m h^r \bmod p,$$

où r est aléatoire ; m est le message à mettre en gage ($m \in \mathbb{Z}_q$) et h est un autre générateur du groupe engendré par g modulo p (il faut ici que l'ordre du groupe, q, soit un nombre premier).

4.a] Justifier que ceci est *statistically hiding* et *computationally binding*.

Exercice 5 : Autres constructions

5.a] Proposez un algorithme de mise en gage utilisant une foncion de hachage, et étudiez sa sécurité (quelle propriété est-ce que la fonction doit avoir ?)

5.b] Même question avec un générateur pseudo-aléatoire.