École normale supérieure **2019-2020**
Département d'informatique

Introduction à la cryptologie
TD n° 9 : Zero-knowledge et Couplages.

Exercice 1. Bob connaît une suite $x_1, \ldots, x_{1000}$ de 1000 entiers compris entre 1 et 10. Il met cette suite sous la forme d'un polynôme $P \in \mathbb{Z}_p[X]$ de degré au plus 1000, tel que $P(i) = x_i$ pour $1 \leq i \leq 1000$. Ici, p est un nombre premier quelconque suffisamment grand (mettons $p > 10^6$). Alice veut vérifier que tous les entier x_i de Bob sont bien entre 1 et 10. Pour cela, en utilisant les techniques du cours, elle dispose de deux outils (ici légèrement simplifiés) :

— Elle peut choisir $\alpha \in \mathbb{Z}_p$, et interroger Bob sur la valeur de $P(\alpha)$, sans rien révéler à Bob (même pas α ou $P(\alpha)$).

— Elle peut choisir un entier n quelconque et $\alpha \in \mathbb{Z}_p$, et demander à Bob la valeur $H(\alpha)$ pour un polynôme H au choix de Bob de degré au plus n. Toujours sans rien révéler à Bob.

Décrire un protocole qui permet à Alice de s'assurer que les x_i sont tous entre 1 et 10, avec probabilité au moins 99%, en posant seulement deux questions à Bob.

Indication : On peut d'abord définir les polynômes $D = (X-1)(X-2)\cdots(X-10)$ et $T = (X-1)(X-2)\cdots(X-1000)$, puis reformuler la question d'Alice (« est-ce que $P(i) \in \{1,\ldots,10\}$ pour $i \in \{1,\ldots,1000\}$ ») sous la forme : « est-ce que un certain polynôme divise un autre polynôme », où les deux polynômes en question sont une expression simple formée à partir de D, T, P.

Exercice 2 (Comment énerver oncle Bob, premier chapitre)**.** Alice et son oncle Bob sont en vacances à la plage, et s'amusent à résoudre des sudokus. Un beau jour, oncle Bob tombe sur un sudoku particulièrement difficile, qu'il n'arrive pas à résoudre. Il le montre à Alice. Alice parvient à le résoudre, et le lendemain, elle l'annonce à Bob. Once Bob est vexé, parce qu'il y a passé plusieurs jours ; il ne la croit pas. Alice est piquée : elle veut alors prouver à oncle Bob qu'elle a réussi à résoudre le sudoku, sans rien lui apprendre sur la solution.

Soit T l'instance du problème sudoku, i.e. une grille 9×9 dont certaines cases contiennent un chiffre dans $[\![1,9]\!]$, et les autres sont vides. Soit S la solution trouvée par Alice, i.e. une grille 9×9 contenant des chiffres dans $[\![1,9]\!]$, tels que :

(a) Les chiffres apparaissant dans chacune des 9 lignes de S sont distincts. De même pour les 9 colonnes, et les 9 sous-grilles 3×3 de S.

(b) Les chiffres de S sont égaux à ceux de T partout où les cases de T sont non-vides.

1. Pourquoi utiliser une preuve *de connaissance* zero-knowledge et pas simplement une preuve zero-knowledge ?

Alice tire une permutation aléatoire $\sigma : [\![1,9]\!] \to [\![1,9]\!]$. Soit $\sigma(S)$ la grille obtenue en appliquant σ à chaque case de S. Alice commite [1] sur chaque valeur de $\sigma(S)$ (par exemple, pour chaque case contenant un chiffre c, elle tire un élément de $r \in \{0,1\}^{128}$ uniformément aléatoirement et publie $H(r \parallel \sigma(c))$, pour une fonction de hachage fixée H, et où $\parallel$ dénote la concaténation).

2. Finir ce protocole : une fois les *commitments* publiés, quelle(s) question(s) Bob peut-il poser à Alice pour s'assurer que la solution S trouvée par Alice remplit les conditions (a) et (b) (sans apprendre la solution)—de sorte que, si une des conditions n'est pas respectée, il s'en rend compte avec probabilité au moins $1/28$?
3. Combien de fois faut-il répéter le protocole pour que si Alice a triché, Bob puisse le découvrir avec probabilité au moins 99% ? Note qui pourra (ou pas) être utile pour approximer le résultat : $\log(100) \approx 4.6$.

1. du verbe franglais *commiter*.

4. Prouver que le protocole est *zero-knowledge.*

Exercice 3 (Sudoku zero-knowledge *sans interaction*)**.** Pour transformer un protocole zero-knowledge tel que celui de l'exercice précédent en un protocole non-interactif, on peut utiliser l'heuristique de Fiat-Shamir : au lieu que Bob envoie son choix de bits aléatoires à chaque itération du protocole, ces bits sont choisis par une fonction déterministe pseudo-aléatoire, typiquement une fonction de hachage. Fixons une telle fonction de hachage H.

1. Alice procède de la manière suivante : elle exécute le protocole interactif de l'exercice précédent, mais à chaque nouvelle itération, elle remplace les inputs de Bob par la sortie de H, avec comme entrée le transcript de toute la communication qui précède. Montrer que ce protocole permet à Alice de donner une preuve fausse. Quelle modification simple faut-il faire pour corriger le problème ?
2. Supposons que le protocole est itéré $\log(100)/\log(1+1/27)$ fois. Que pensez-vous de la sécurité du protocole ?

Exercice 4 (*Probabilistically Checkable Proofs* et SNARGs)**.** Un langage $\mathcal{L}$ admet une *probabilistically checkable proof* ssi il existe un algorithme P en temps polynomial probabiliste à sortie dans $\{0,1\}$ tel que : pour tout $x \in \mathcal{L}$, il existe une « preuve » y telle que $P(x,y)$ renvoie toujours 1 (accepte), et pour tout $x \notin \mathcal{L}$, et pour tout y, $P(x,y)$ renvoie 0 avec probabilité au moins 1/2. Plus exactement, on dit que $\mathcal{L}$ est dans $\mathrm{PCP}[A,B]$ ssi il existe un P comme précédemment qui de plus : (1) ne consomme que A bits d'aléa ; et (2) lit au plus B bits dans y.

1. Soit n la taille de l'entrée x. Montrer $\mathrm{PCP}[0,0] = \mathrm{PCP}[O(\log n), 0] = \mathrm{P}$.
2. Montrer $\mathrm{PCP}[O(\log n), \mathrm{poly}(n)] = \mathrm{NP}$.

Le théorème PCP (hautement non-trivial) donne $\mathrm{PCP}[O(\log n), O(1)] = \mathrm{NP}$. On souhaite utiliser ce résultat pour déduire une technique de preuve interactive succincte. Dans ce but, pour prouver $x \in \mathcal{L}$ en connaissant une preuve y, Alice construit un arbre de Merkle sur y.

3. Comment construire une preuve de connaissance succincte à partir de cet arbre de Merkle ?
4. Montrer que le protocole est succint, au sens où la quantité de bits envoyés par Alice pour convaincre Bob est $O(\lambda \log m)$, où m est la taille de la preuve y, et λ est la taille de sortie de la fonction de hachage utilisée dans l'arbre de Merkle.

Couplages. Dans toute la suite, on suppose qu'on a un couplage $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ admissible calculable efficacement avec des groupes $\mathbb{G}$, $\mathbb{G}_T$ cycliques d'ordre premier p. Pour g générateur de $\mathbb{G}$, on a donc :

- $g_T = e(g,g) \neq 1$.
- $\forall a, b \in \mathbb{Z}_p, e(g^a, g^b) = g_T^{ab}$.

Exercice 5 (Échauffement)**.**

1. Montrer que les deux propriétés ci-dessus sont vraies pour *un* générateur de $\mathbb{G}$ (au sens : il existe un générateur tel que...) si et seulement si elles sont vraies pour *tout* générateur de $\mathbb{G}$.
2. Montrer que le problème de Diffie-Hellman décisionnel est facile dans $\mathbb{G}$.
(Diffie-Hellman décisionnel : distinguer un triplet (g^a, g^b, g^{ab}) pour a, b uniformes d'un triplet (g^a, g^b, g^c) pour a, b, c uniformes.)

Exercice 6 (Échange de clef sans interaction à trois participants)**.**

1. En s'inspirant du protocole de Diffie-Hellman, décrire un protocole dans lequel trois parties A, B, C se mettent d'accord sur un secret commun en présence d'un adversaire qui observe les communications, et tel qu'il suffit à chaque participant de publier une seule valeur, indépendante des valeurs publiées par les autres participants. On suppose que le problème de Diffie-Hellman bilinéaire (*i.e.* étant donnés g^a, g^b, g^c, calculer g_T^{abc}) est difficile.
2. Supposons maintenant qu'on a une *application multilinéaire* :

$$\begin{aligned} e_n : \mathbb{G}^n &\to \mathbb{G}_T \\ (g^{a_1}, \ldots, g^{a_n}) &\mapsto e_n(g, \ldots, g)^{a_1 \cdot \cdots \cdot a_n} \end{aligned}$$

   où $e_n(g, \ldots, g) \neq 1$. Généraliser l'exercice précédent à $n+1$ participants. Sur quelle hypothèse repose maintenant la sécurité ?

Exercice 7 (Sécurité des signatures de Boneh-Boyen)**.** On définit le schéma de signature suivant. La clef secrète est $x \in \mathbb{Z}_p^*$, tiré uniformément aléatoirement. La clef publique est $u = g^x$.

- **Signature :** soit un message $m \in \mathbb{Z}_p$. On suppose $m \neq x$. La signature de m est $\sigma = g^{(x+m)^{-1}}$, où l'inversion est modulo p.
- **Vérification :** étant donnée une signature σ, on vérifier $e(\sigma, ug^m) = g_T$.

On définit le problème q**-Strong Diffie-Hellman** (q-SDH) : étant donné $(g, g^x, g^{x^2}, \ldots, g^{x^q})$ pour un x uniforme, calculer une paire $(m, g^{(x+m)^{-1}})$ (pour un m quelconque). On suppose que ce problème est difficile (pour tout q).

1. Vérifier que l'algorithme de vérification accepte une signature légitime.
2. Montrer qu'on peut utiliser le couplage e pour vérifier qu'une instance $(g, g^x, g^{x^2}, \ldots, g^{x^q})$ de q-SDH est correctement formée. Montrer de même que la version décisionnelle du problème (vérifier si une solution de q-SDH est correcte) est facile.
3. Dans la suite, nous considérons la sécurité de la signature ci-dessus contre un attaquant ayant accès à q messages connus $m_1, \ldots, m_q$ fixés d'avance, et essayant de forger une signature pour un message de son choix m^* distincts des m_i. Nous allons réduire cette sécurité à l'hypothèse q-SDH. On suppose donc qu'on a accès à un attaquant qui réussit à casser la signature au sens précédent (pour tout choix de g), et que nous avons une instance $(g, g^x, g^{x^2}, \ldots, g^{x^q})$ de q-SDH. Notre but est d'utiliser cet attaquant pour résoudre l'instance de q-SDH.
   (a) Dans un premier temps, supposons que nous connaissons magiquement x. On donne à notre attaquant une instance de l'algorithme de signature, où le générateur de $\mathbb{G}$ est :

   $$h = g^{\prod(x+m_i)}.$$

   Expliquer comment on fournit à l'attaquant les signatures des messages m_i, et comment on utilise sa sortie pour résoudre q-SDH.

   (b) Maintenant on ne suppose plus que x est magiquement connu. Montrer comment on peut quand même calculer h en utilisant l'instance q-SDH.

   (c) De même, montrer comment on peut extraire la réponse à l'instance q-SDH de la sortie de l'attaquant (sans connaissance magique de x).