TD Cryptographie

ENSSAT

17/03/2021

Exercice 15 – Chiffrement d'ElGamal

On va ici vérifier votre compréhension des concepts utilisés dans le chiffrement d'ElGamal (sur des petits nombres afin que les calculs soient rapidement effectués).

Alice et Bob veulent communiquer en utilisant ElGamal, ils publient le nombre premier $p = 5$

1. Pour compléter leur clef publique commune (p, a), peuvent-ils choisir le nombre $a = 4$? Montrer qu'ils peuvent choisir de publier $a = 2$.
2. Alice et Bob ont publié la clef commune publique $(p, a) = (5, 2)$. Bob choisit aléatoirement le nombre $\mathrm{B} = 3$ et envoie sa clef publique k_B à Alice. Alice veut envoyer le message clair $x = 4$ à Bob. Elle choisit aléatoirement le nombre $\mathrm{A} = 2$. Quel est le message crypté qu'elle envoie à Bob ?
3. Faites le travail de Bob : retrouver x en détaillant les étapes du calcul.

Corrigé

1. Pour que l'exponentielle modulaire soit bijective, il faut que a soit primitif modulo p. Or, ici $a^2 = 16 = 1 \mod 5$. Donc a n'est pas primitif et $(5, 4)$ ne peut pas être une bonne clé.
   En revanche, par un calcul rapide :
   $$a^2 = 4 \mod 5, \quad a^3 = 3 \mod 5, \quad a^4 = 1 \mod 5$$
   Ainsi, a est primitif modulo 5 et $(5, 2)$ est une bonne clé.
2. L'ensemble du protocole repose sur le protocole d'échange de clé de Diffie-Hellman, qui lui-même repose sur le fait que $a^{\mathrm{AB}} \mod p = (a^\mathrm{A})^\mathrm{B} \mod p = (a^\mathrm{B})^\mathrm{A} \mod p$.
   Bob a choisi $\mathrm{B} = 3$ et donc $k_\mathrm{B} = 2^3 = 3 \mod 5$. Il publie donc $k_\mathrm{B} = 3$ à Alice. Alice a, quant à elle, choisi $\mathrm{A} = 2$ et elle veut envoyer le message $x = 4$. Elle calcule d'abord la clé du protocole :
   $$k = (k_\mathrm{B})^\mathrm{A} = 3^2 = 4 \mod 5$$
   puis elle calcule $k \times x = 4 \times 4 = 1 \mod 5$. Elle calcule de plus $k_\mathrm{A} = a^\mathrm{A} = 2^2 = 4 \mod 5$. Elle envoie alors à Bob le couple $(k_\mathrm{A}, kx) = (4, 1)$.
3. Bob reçoit $k_\mathrm{A} = 4$ et $kx = 1$. Il a connaissance de B, donc il calcule la clé du protocole par $k = k_\mathrm{A}^\mathrm{B} = 4^3 \mod 5 = 4$. Ensuite, il termine l'inverse de k modulo $p = 5$. On obtient rapidement que $4 \times 4 = 1 \mod 5$ donc $k^{-1} = 4$ est son propre inverse.
   Enfin, il lui reste à calculer $k^{-1} \times (kx) = 4 \times 1 = 4$. Le message envoyé à l'origine est bien 4.

Exercice 19 – Test de Fermat et test de Miller

1. On a
   $$3^{340} \equiv 56 \mod 341$$
   Qu'en déduisez-vous ?
   De plus, on a
   $$2^{340} \equiv 1 \mod 341$$
   Est-ce contradictoire avec le résultat précédent ?
2. Le nombre $n = 341$ passe-t-il le test de Miller en base $a = 2$? en base $a = 4$?

Corrigé

1. Le petit théorème de Fermat indique que si p est premier et ne divise pas a, alors $a^{p-1} \equiv 1 \mod p$. Ici, 341 ne divise pas 3 et pourtant 3^{340} n'est pas congru à 1 module 341 : on peut, avec certitude, dire que 341 n'est pas premier (raisonnement par contraposée).
   En revanche, le fait que $2^{340} \equiv 1 \mod 341$ n'est pas une contradiction, puisque le théorème de Fermat indique que pour que p soit premier, il est nécessaire que $a^{p-1} \equiv 1 \mod p$ **pour tout** $a < p$.
2. On décompose tout d'abord 341 :

$$n = 341 = 85 \times 2^2 + 1 = 2^r s + 1 \text{ avec } r = 2 \text{ et } s = 85$$

   - Pour $a = 2$, on constate que $a^{10} \equiv 1 \mod 341$ et donc $a^{85} \equiv 32 \mod 341$: la condition c_1 n'est pas vérifiée. Vérifions la condition c_2 :

$$2^{2^0 s} = 2^s \equiv 32 \mod 341$$
$$2^{2^1 s} = 2^{2s} \equiv 1 \mod 341$$

     Ainsi la propriété c_2 n'est jamais réalisé, et 341 ne passe pas le test de Miller en base 2.
   - Pour $a = 4$, pour la condition c_2, on essaie :

$$4^{2^0 s} = 4^s \equiv 1 \mod 341$$
$$4^{2^1 s} = 4^{2s} \equiv 1 \mod 341$$

     Ainsi la propriété c_2 n'est jamais réalisé. Passons à la condition c_1 : on constate que $a^5 \equiv 1 \mod 341$ donc $a^{85} \equiv 1 \mod 341$: la condition c_1 est vérifiée et 341 passe le test de Miller en base 4.

> **Remarque**
>
> Il n'est pas anormal que 341 passe le test de Miller en base 4, sans être un nombre premier. L'objectif de l'exercice suivant est d'avoir une borne sur la probabilité qu'un nombre soit premier en réitérant le test de Miller.

Exercice 20 – Méthode Monte-Carlo et test de primalité

En cryptographie, plusieurs méthodes de chiffrement nécessitent de choisir aléatoirement des très grands nombres premiers (notamment les systèmes RSA et ElGamal). Pour cela une méthode couramment utilisée est d'engendrer quasi-aléatoirement des nombres impairs de k bits (où k est choisi par l'utilisateur) et de tester leur primalité. Malheureusement, les tests déterministes de primalité sont lents, bien que le problème de décision « n est-il premier ? » soit dans la classe $\mathscr{P}$ des problèmes polynomiaux en temps.

On utilise donc plutôt des algorithmes probabilistes, *i.e.* basés sur des tirages aléatoires, mais comportant un risque d'erreur lorsque l'une des réponses « oui » ou « non » est rendue (un tel algorithme s'appelle un *algorithme de Monte-Carlo*). Ces algorithmes sont généralement très efficaces (en terme de temps de calcul), et le risque d'erreur peut être rendu arbitrairemment faible en jouant sur le nombre d'itérations.

Le but de l'exercice est de décrire le test de Miller-Rabin (l'un des plus utilisées en pratique) et d'évaluer la probabilité d'erreur de la méthode.

Rappelons le résultat d'Hadamard : le nombre $\pi(n)$ de nombres premiers inférieurs ou égaux à n est, pour n très grand, approximativement égal à $\frac{n}{\ln n}$.

1. (a) On considère le nombre $\mathrm{N}(k)$ de nombres premiers entre 2^{k-1} et $2^k - 1$ inclus (si on représente les nombres en binaire, ce sont les nombres premiers qui nécessitent k bits pour être codés). Lorsque k est grand, montrer que ce nombre est approximativement égal à $\frac{2^{k-1}}{k \ln 2}$.

(b) On effectue un tirage aléatoire (uniforme) parmi les nombres impairs compris entre 2^{k-1} et 2^k-1 inclus. Quelle est, approximativement pour k grand, la probabilité que le nombre tiré soit premier ?

2. Le *test de primalité de Miller-Rabin* teste si un nombre n est premier ou composé en utilisant le calcul d'une certaine fonction booléenne $f(n,a)$, où a est un nombre entier compris entre 1 et $n-1$. Cette fonction booléenne est telle que :
   - Pour tout n premier, pour tout $a \in \{1,2,\ldots,n-1\}$, $f(n,a) = vrai$.
   - Pour tout n composé, il existe au plus $\frac{n-1}{4}$ entiers $a \in \{1,2,\ldots,n-1\}$ tels que $f(n,a) = vrai$.

   On en déduit le fonctionnement d'une des itérations de l'algorithme : on suppose que l'on dispose d'une fonction `Choisir` dont le but est de tirer « au hasard » (c-à-d de façon uniforme) un entier $a \in \{1,2,\ldots,n-1\}$ (c'est la fonction habituellement appelée `Random`) ; on calcule alors $f(n,a)$; si $f(n,a) = vrai$, l'algorithme répond « n est premier » (on dira alors que *le test de primalité est positif*) ; si $f(n,a) = faux$, l'algorithme répond « n est composé » (on dira alors que *le test de primalité est négatif*).
   (a) Écrire la fonction $f(n,a)$ dans un langage de description simple.
   (b) Dans quel cas est-on certain que la réponse donnée par l'algorithme est exacte ?
   (c) Justifier que $\frac{1}{4}$ est un majorant pour la probabilité que le test soit positif sachant que n est composé.
   (d) En réalité, on effectue bien sûr plusieurs itérations de l'algorithme avec à chaque fois un choix aléatoire de a différent. Autrement dit l'algorithme complet (sans les déclarations de variables pour ne pas alourdir le texte) est :

```
Lire(n);
i := 1;
Tant que i ⩽ m :
        Choisir a ∈ {1,2,...,n-1};
        Calculer f(n,a);
        Si f(n,a)=faux :
            Écrire(n est composé);
            Sortir;
        FinSi
        i := i+1;
FinTantQue
Écrire(n est considéré premier après m tests positifs);
```

   Les différents tirages de a sont supposés indépendants. En déduire un majorant de la probabilité que l'algorithme réponde que n est premier sachant que n est composé (après m itérations).

3. Un utilisateur tire au hasard un nombre impair $n \in \{2^{k-1}+1,\ldots,2^k-1\}$. Il souhaite évaluer le niveau de confiance qu'il peut accorder au test de primalité lorsqu'il est positif pour ce nombre n. On suppose qu'il a effectué m itérations de l'algorithme, et qu'il a obtenu la réponse « n est premier ». Calculer un majorant de la probabilité d'erreur du test, c-à-d de la probabilité que son nombre n soit en fait composé.

   Application numérique : Dans les applications industrielles du système RSA, on peut encore utiliser des nombres premiers aléatoires de 2048 bits). Un nombre n impair de 2048 bits est jugé premier après $m = 50$ itérations de l'algorithme. Calculer un majorant de la probabilité d'erreur du test.

Corrigé

1. (a) On a, pour k grand :
$$\mathrm{N}(k) \approx \pi(2^k) - \pi(2^{k-1})$$
soit, en utilisant le théorème des nombres premiers démontrés par Hadamard et la Vallée Poussin :
$$\begin{aligned}\mathrm{N}(k) &\approx \frac{2^k}{\ln(2^k)} - \frac{2^{k-1}}{\ln(2^{k-1})} \\ &\approx \frac{2^k}{k\ln(2)} - \frac{2^{k-1}}{(k-1)\ln(2)} \\ &\approx 2^{k-1}\frac{2(k-1)-k}{k(k-1)\ln(2)} \\ &\approx 2^{k-1}\frac{k-2}{k(k-1)\ln(2)} \approx \frac{2^{k-1}}{k\ln(2)}\end{aligned}$$
   (b) On choisit aléatoirement un nombre impairs compris entre 2^{k-1} et 2^k-1 inclus. Il y a $\frac{2^k-2^{k-1}}{2}$ nombres impairs. Le tirage étant uniforme, il y a équiprobabilité, et la probabilité qu'il soit premier est (pour k grand et en utilisant l'approximation vue précédemment) :
$$\begin{aligned}\mathbb{P}(\text{nombre tiré premier}) &= \frac{\mathrm{N}(k)}{\frac{2^k-2^{k-1}}{2}} \\ &\approx \frac{2\frac{2^{k-1}}{k\ln(2)}}{2^{k-1}-2^{k-2}} \\ &\approx \frac{2^k}{2^{k-1}k\ln(2)} \\ &\approx \frac{2}{k\ln(2)}\end{aligned}$$

2. (a) Cf cours.
   (b) On est sûr que le nombre est composé si l'algorithme renvoie faux. En revanche, s'il renvoie vrai, c'est soit que le nombre n est premier, soit qu'il ne l'est pas mais qu'on a tiré l'un des entiers a qui renvoie $f(a,b)$ vrai.
   (c) On suppose que n est composé. Puisqu'on choisit de manière uniforme l'entier a, l'algorithme renvoie vrai si on tombe sur l'un des entiers qui renvoie $f(n,a)$ vrai. Mais alors, par hypothèse de construction de f, on a :
$$\mathbb{P}_{n \text{ composé}}(f(a,n) \text{ renvoie vrai}) \leqslant \frac{\frac{n-1}{4}}{n-1} \leqslant \frac{1}{4}$$
   (d) On choisit aléatoirement l'entier a, et on effectue m fois la boucle. On a alors, en utilisant la question précédente, et par indépendance des tirages de a :
$$\mathbb{P}_{n \text{ composé}}(\text{renvoie vrai}) = \prod_{i=1}^{m} \mathrm{P}_{n \text{ composé}}(f(a,n) \text{ renvoie vrai}) \leqslant \frac{1}{4^m}$$

3. Définissons proprement le contexte. Un entier naturel non nul k étant fixé, on a deux expériences aléatoires, la deuxième dépendant du résultat de la première :
   - le tirage uniforme d'un nombre n parmi les nombres impairs de taille k bits ;
   - le tirage uniforme dans $[\![1, n-1]\!]$ de m nombres $a_1, \ldots, a_m$, les m tirages étant indépendants.

   La première expérience permet de définir l'évènement $\mathrm{C}_k =$ « n est composé ».
   Les deux expériences effectuées successivement permettent de définir l'évènement $\mathrm{T}_m = \mathrm{M}_1 \cap \ldots \cap \mathrm{M}_m$, où $\mathrm{M}_j =$« n passe le test de Miller en base a_j ».

   D'après la question 2(d), on a :
$$\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k) \leq 4^{-m}$$

La **probabilité d'erreur du test de primalité**, celle qui indique à l'utilisateur le niveau de confiance qu'il peut accorder à un test de primalité positif, est en fait la probabilité que n soit composé sachant qu'il a passé m fois le test avec succès, *i.e.* $\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m)$. Par la formule de Bayes :

$$\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) = \frac{\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k)\mathbb{P}(\mathrm{C}_k)}{\mathbb{P}(\mathrm{T}_m)} \quad \text{avec} \quad \mathbb{P}(\mathrm{T}_m) = \mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k)\mathbb{P}(\mathrm{C}_k) + \mathbb{P}\left(\mathrm{T}_m \,|\, \overline{\mathrm{C}_k}\right)\mathbb{P}\left(\overline{\mathrm{C}_k}\right)$$

On a $\mathbb{P}\left(\mathrm{T}_m \,|\, \overline{\mathrm{C}_k}\right) = 1$, car si n est premier alors il passe le test. Cela permet de réécrire la probabilité d'erreur sous la forme suivante :

$$\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) = \frac{1}{1 + \frac{1}{\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k)} \frac{\mathbb{P}\left(\overline{\mathrm{C}_k}\right)}{\mathbb{P}(\mathrm{C}_k)}} = \frac{1}{1 + \frac{1}{\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k)} \frac{1 - \mathbb{P}(\mathrm{C}_k)}{\mathbb{P}(\mathrm{C}_k)}} = \frac{1}{1 + \frac{1}{\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k)} \left(\frac{1}{\mathbb{P}(\mathrm{C}_k)} - 1\right)}$$

On remarque que $\frac{1}{\mathbb{P}(\mathrm{C}_k)} - 1 > 0$, donc on déduit de l'inégalité $\mathbb{P}(\mathrm{T}_m \,|\, \mathrm{C}_k) \leq 4^{-m}$ la majoration suivante :

$$\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) \leq \frac{1}{1 + 4^m \left(\frac{1}{\mathbb{P}(\mathrm{C}_k)} - 1\right)}$$

En utilisant l'approximation trouvée à la question 1(b), on en déduit une majoration numérique approchée très facile à exploiter en pratique. On a en effet, en supposant k grand, $\mathbb{P}\left(\overline{\mathrm{C}_k}\right) \approx \frac{2}{k \ln(2)}$, d'où $\mathbb{P}(\mathrm{C}_k) \approx 1 - \frac{2}{k \ln(2)}$, d'où la majoration (approximative) suivante :

$$\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) \overset{\text{approx.}}{\leq} \frac{1}{1 + 4^m \frac{2}{k \ln(2) - 2}}$$

Puisque k est grand, ainsi que 4^m même pour des valeurs raisonnables de m (disons $m \geq 10$), l'on a $1 + 4^m \frac{2}{k \ln(2) - 2} \approx \frac{2 \cdot 4^m}{k \ln(2)}$. On obtient finalement la majoration approximative suivante, très simple à exploiter en fonction de la taille k des nombres premiers qu'on veut générer et du nombre m de tests de Miller que l'on fait :

$$\mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) \overset{\text{approx.}}{\leq} \frac{k \ln(2)}{2 \cdot 4^m}$$

Cette majoration montre que **la probabilité d'erreur du test, à nombre m d'itérations fixé, augmente approximativement linéairement en fonction de la taille k** du nombre n à tester. Ainsi, si on double k et qu'on conserve le même nombre d'itérations, la probabilité d'erreur n'est que multipliée par 2, ce qui est un résultat très satisfaisant qui n'était guère prévisible !

Applications numériques. Pour $k = 1024$ bits et $m = 50$ itérations, on trouve :

$$\mathbb{P}\left(\overline{\mathrm{C}_k}\right) \approx \frac{2}{k \ln(2)} = 0,0028 \quad \text{et} \quad \mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) \leq 2,8 \cdot 10^{-28}$$

Ainsi, si l'on fait 10 000 tirages de nombres impairs de taille $k = 1024$ bits, alors on aura en moyenne tiré 28 nombres premiers (ce n'est pas si mal !), et si un nombre impair ainsi tiré passe $m = 50$ tests de Miller, alors la probabilité qu'il ne soit pas premier est inférieure ou égale à $2,8 \cdot 10^{-28}$.

Pour $k = 2048$ bits (valeur plus raisonnable en 2020) et $m = 50$ itérations, on trouve :

$$\mathbb{P}\left(\overline{\mathrm{C}_k}\right) \approx \frac{2}{k \ln(2)} = 0,0014 \quad \text{et} \quad \mathbb{P}(\mathrm{C}_k \,|\, \mathrm{T}_m) \leq 5,6 \cdot 10^{-28}$$

Noter le doublement de la probabilité d'erreur consécutif au doublement de la taille k.