Introduction à la sécurité
TD5

Ninon Devis: ninon.devis@ircam.fr

Master 2021

Problème du log discret

- Une opération particulièrement importante en cryptographie à clef publique est l'exponentiation discrète dans un groupe abélien fini G.

Soit $g \in G$, $n \in \mathbb{N}$: g^n A priori n multiplications.

- Peut être obtenu en $O(\log_2 n)$ multiplications en décomposant n en base 2:

$$n = \sum_{i=0}^{\ell-1} a_i 2^i \in \mathbb{N}. \quad a_i \in \{0,1\} \quad i \in \{0, \dots, \ell-1\}$$

ℓ longueur en bit de n

$$g^n = \prod_{i=0}^{\ell-1} g^{a_i 2^i} = g^{a_0} \times (g^{a_1})^2 \times (g^{a_2})^4 \times (g^{a_3})^8 \times \dots \times (g^{a_{\ell-1}})^{2^{\ell-1}}$$

$$= g^{a_0}(g^{a_1}(g^{a_2}(g^{a_3}(\dots(g^{a_{\ell-1}})^2)^2)^2)^2)^2$$

Problème du log discret

- Exemple pour l-1 = 3: $g^n = g^{a_0}(g^{a_1}(g^{a_2}(g^{a_3})^2)^2)^2$
  Combien y a t-il d'élévations au carré ? $3 = l-1$
  Combien y a t-il de multiplications *au pire des cas* ? $6 = 2 \times (l-1)$

- En généralisant, l'expression g^n peut être calculée en $l-1$ élévations au carré, entrelacées avec des multiplications par g^{a_i}, $i \in \{0, -, l-2\}$

- Dans le pire des cas, il faut donc $2 \times (l-1)$ multiplications pour calculer g^n.

➔ On appelle cet algorithme l'exponentiation binaire.

Exercice 1 - Multi-exponentiation

Soit G un groupe commutatif (noté multiplicativement). Pour simplifier, on peut considérer que $G = (Z/nZ)^*$ Proposer un algorithme qui, étant donnés t éléments $g_1, \dots, g_t$ du groupe G et des entiers positifs $n_1, \dots, n_t$ calcule le produit $g_1^{n1} \dots g_t^{nt} \in G$ en $O(l + 2^{t})$ multiplications dans G (où l est la taille en bits de $\max(n_1, \dots, n_t)$).

- On pose $j \in \{1, \dots, t\}$ afin de pouvoir exprimer n_j : $n_j = \sum_{i=0}^{l-1} a_{i,j} 2^i$ $a_{i,j} \in \{0,1\}$
  $i \in \{0, \dots, l-1\}$.
- Considérons d'abord le cas t = 2: on veut calculer $g_1^{n_1} g_2^{n_2}$
  Or: $g_1^{n_1} = \prod_{i=0}^{l-1} g_1^{a_{i,1}2^i}$ $g_2^{n_2} = \prod_{i=0}^{l-1} g_2^{a_{i,2}2^i}$

Exercice 1 - Multi-exponentiation

$$\dots\left(g_1^{a_{\ell-1,1}}\, g_2^{a_{\ell-1,2}}\right)^{2^{\ell-1}}$$

- D'où: $g_1^{n_1} g_2^{n_2} = \prod_{i=0}^{\ell-1} g_1^{a_{i,1} 2^i} \prod_{i=0}^{\ell-1} g_2^{a_{i,2} 2^i}$

$$= \prod_{i=0}^{\ell-1} \left(g_1^{a_{i,1}}\, g_2^{a_{i,2}}\right)^{2^i} = g_1^{a_{0,1}} g_2^{a_{0,2}} \left(g_1^{a_{1,1}} g_2^{a_{1,2}}\right)^2 \left(g_1^{a_{2,1}} g_2^{a_{2,2}}\right)^4$$

$$= g_1^{a_{0,1}} g_2^{a_{0,2}} \left(g_1^{a_{1,1}} g_2^{a_{1,2}} \left(g_1^{a_{2,1}} g^{a_{2,2}} \dots \left(g_1^{a_{\ell-1,1}} g_2^{a_{\ell-1,2}}\right)^2\right)^2\right)^2$$

- On a $\ell - 1$ élévations au carré: comme pour l'exp. binaire avec des multiplications par $g_1^{a_{i,1}} g_2^{a_{i,2}}$ $i \in \{0, \ell-2\}$

Exercice 1 - Multi-exponentiation

- Il y a en tout 4 éléments possibles de la forme $g_1^{a_{i,1}} g_2^{a_{i,2}}$

$$g_1^0 g_2^0 = 1_G \qquad g_1^1 g_2^0 = g_1 \qquad g_1^0 g_2^1 = g_2 \qquad g_1^1 g_2^1 = g_1 g_2$$

- En pré-calculant $g_1 g_2$ le nombre de multiplication nécessaires pour calculer $g_1^{n1} g_2^{n2}$ est alors égale à $2(l-1)+1$ dans le pire des cas.

- En généralisant, pour t arbitraire: $g_\epsilon = g_1^{b_1} g_2^{b_2} \cdots g_t^{b_t}$ $\quad b_i \in \{0,1\}$

- Ce pré-calcul nécessite donc 2^t multiplications dans G.
- Finalement, l'algorithme nécessite: $O(2(l-1)+2^t) = O(2^t + l)$ multiplications dans G dans le pire des cas.

Problème du log discret

- Rappels d'algèbre:
  - Groupe cyclique: groupe engendré par 1 élément
  - Pour un groupe fini, l'ordre = cardinal
  - On note <g> le groupe engendré par <g>.
  - Exemple sur Z/13Z:

$\langle 2 \rangle = \{2, 4, 8, 12, 1, 3, 5, 7, 9, 11\}$

Que l'on peut noter: $\langle 2 \rangle = \{2^i / i \leq 12\}$

  - De la même manière $\langle \underline{3} \rangle = \{1, 3, 9\}$
  - L'ordre d'un élément d'un groupe est la taille du sous groupe engendré par g.
  - Ici l'ordre de 3 est 3 dans $\mathbb{Z}/13\mathbb{Z}$

Problème du log discret

- Pour un groupe abélien G, un élément g de G, et un élément h appartenant au sous groupe multiplicatif engendré par g (<g> = $\{g^i, i \in \mathbb{Z}\}$, le problème du logarithme discret est le suivant: trouver x le plus petit entier positif tel que $h = g^x$

```
for x in range(65537):
    if pow(5, x, 65537)==42
        print(x)
        break.
```

➔ Trouver x tel que: $5^x = 42 \bmod [2^{16}+1]$ → 36432

➔ Trouver x tel que: $19139^x = 9296 \bmod [2^{16}+1]$ → 28

- On peut retrouver $x = \log_g(h)$
- x doit-il réellement parcourir toutes les valeurs jusque 65537 ?

Non, on peut s'arreter à l'ordre → `G = set([pow(19139, i, 65537) for i in range(65537)])`

`print(len(G))` ↳ 1024

On peut donc écrire `for x in range(1024).`

Pour la recherche exhaustive il suffit de faire 9 itérations

Exercice 2 - Algorithme de Shanks

Considérons un groupe multiplicatif cyclique G engendré par $g \in G$ d'ordre connu q (autrement dit, nous avons $G = \{1, g, g^2, \ldots, g^{q-1}\}$). Proposer un algorithme de résolution de logarithme discret par compromis temps-mémoire de complexité $O(\sqrt{q})$ opérations de groupe en temps et $O(\sqrt{q})$ éléments de groupe en mémoire.

- D'après l'indication: $x = x_1 T + x_0$ avec $0 \leq x_1 < T$ $\quad 0 \leq x_0 < T$

  $\hookrightarrow x < q$

- Quel serait l'algorithme "naïf" ?

  for $0 \leq x_1 < q/T$

  for $0 \leq x_0 < T$

  if $y = g^{x_0 T + x_1}$ $\rightarrow$ break c'est bon !

- Il est possible d'être plus efficace grâce à une table de hachage !

Exercice 2 - Algorithme de Shanks

- D'après l'indication: $x = x_1 T + x_0$
- En remarquant que: $h = g^{x} \iff \underline{h\left(g^{(-T)x_1}\right)} = hg^{-x_1 T} = g^{x_0}$
- Quelle est la table de hachage pertinente à construire ?

On construit la table qui liste (x_0, g^{x_0}) [baby step]

- Puis on calcule les: $(x_1, h(g^{-T})^{x_1})$ [giant step]
- jusqu'à ce que baby-step = giant-step. On retourne les (x_0, x_1) associés !

Exercice 2 - Algorithme de Shanks

$g^n \rightsquigarrow O(\log_2 n)$

- Le nombre de multiplication à effectuer est égal à:
  - pour la table des éléments g^i : $i \in \{0, \dots, T\}$ T
  - pour le calcul de g^{-T}: $g^{-T} = g^{q-T} \to O(\log(q-T))$
  - pour la recherche d'éléments $h(g^{-T})^j$: $j \in \{0, \dots, T-1\}$

    T
- Donc finalement un total de: $O(T + \log(q-T) + T) = O(T)$
- Complexité mémoire: pour la table de hachage $\to O(T)$
- Quelle valeur pertinente de T choisir ?
  - Complexité de temps: $\simeq O\left(T + \frac{q}{T}\right)$

dérivée : $1 - \frac{q}{T^2}$ $\quad T = \sqrt{q} \to$ dérivée s'annule !

Exercice 4 - Sécurité du RSA naïf

Rappels Chiffrement RSA naïf

- L'utilisateur choisi: p, q deux nombres premiers
- Il calcule: $N = pq$
- Puis: $\varphi(N) = (p-1)(q-1)$
- Il choisit ensuite un exposant public e: premier à $\varphi(N)$
- Il calcule ensuite d: l'inverse de e mod $\varphi(N)$
- *La clef publique est alors le couple (N, e) et la clef secrète est d.*

- Chiffrement: pour un message m $\in \mathbb{Z}/N\mathbb{Z}$ $\quad c = m^e \bmod N$

- Déchiffrement: pour un message c $\in \mathbb{Z}/N\mathbb{Z}$ $\quad m = c^d \bmod N$.

Exercice 4 - Sécurité du RSA naïf

Rappels Objectifs de l’adversaire:

- Bris total: l’adversaire retrouve la clef privée.
- Inversion du chiffrement: l’adversaire est capable de retrouver l’intégralité d’un message clair associé à un chiffré donné.
- Sécurité sémantique: obtenir un bit d’information sur un message clair associé à un message chiffré donné.

Exercice 4 - Sécurité du RSA naïf

4.a Montrer que le protocole de chiffrement RSA naïf n'est pas sémantiquement sûr sous une attaque à clairs choisis.

- Sous une attaque à clairs choisis: accès aux textes chiffrés et aux textes clairs correspondants.
- Sémantiquement sûr: obtenir un bit d'information sur un message clair associé à un message chiffré donné. $(m, m' \quad c)$
- En quoi RSA naïf n'est pas sémantiquement sur ?

RSA naïf est déterministe!

On applique RSA à m_0 et m_1 et on regarde quelle valeur $= c$!

Exercice 4 - Sécurité du RSA naïf

4.b Montrer que le protocole de chiffrement RSA naïf est inversible sous une attaque à un chiffré choisi

- Inversion du chiffrement: l'adversaire est capable de retrouver l'intégralité d'un message clair associé à un chiffré donné.
- Soit c* le chiffré dont on veut tirer le texte clair m*. Il nous est demandé le déchiffrement direct de c*.
- On choisit r dans Z/NZ tel que l'on ai le chiffré $c = c^* r^e$

$r \neq 1$

Soit m le déchiffrement obtenu de c.

Alors : $m^e = c^* r^e \mod N$ $\Rightarrow$ $\frac{m^e}{r^e} = \left(\frac{m}{r}\right)^e = c^* \mod N$

On peut retrouver $\frac{m}{r} \mod N$ qui est donc le texte clair associé à c^* !