Cryptographie

Chiffrement à clé publique

Gabriel Chênevert

15 décembre 2025

Aujourd'hui

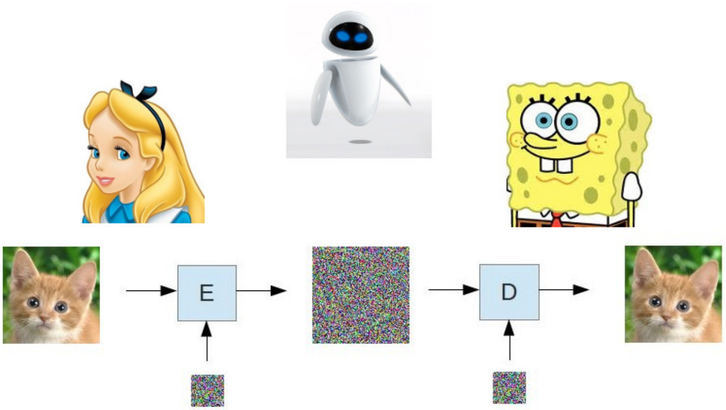
Chiffrement asymétrique

Problème du logarithme discret

Applications du DLP : Diffie-Hellman, ElGamal

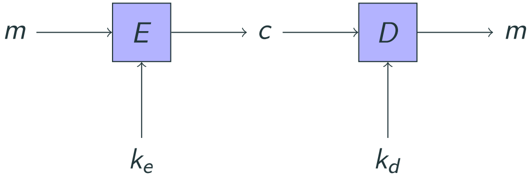
Cryptanalyse

Rappel : chiffrement symétrique



Chiffrement asymétrique

Un cryptosystème pourrait utiliser *a priori* des clés différentes pour le chiffrement et le déchiffrement :



(ce qui inclut les cryptosystèmes symétriques dans le cas particulier $k_e = k_d$)

si la connaissance de l'une des clés ne fournit pas d'information utile à propos de l'autre

alors l'une d'elle peut être rendue publique

Chiffrement à clé publique

La clé de chiffrement k_e est rendue publique (k_d gardée privée)

n'importe qui peut écrire à Bob, mais seulement lui peut lire

Implémenté par exemple dans PGP/GPG

Problèmes asymétriques célèbres

- factorisation de grands entiers

  $\Longrightarrow$ RSA (1977)

- problème du logarithme discret (DLP)

  $\Longrightarrow$ Diffie-Hellman (1976), ElGamal (1984), DSA (1993)

- logarithme discret sur les courbes elliptiques

  $\Longrightarrow$ ECC (2000+): ECDH, ECDSA, Ed25519, ...

- problème du plus court vecteur / résolution de systèmes d'équations bruitées

  $\Longrightarrow$ cryptographie basée sur les réseaux ...

Chiffrement hybride

Problème : toutes ces constructions sont *beaucoup* moins performantes que les algorithmes de chiffrement symétriques.

Par exemple : RSA nécessite des clés de 3072 bits pour 128 bits de sécurité

(15360 bits pour passer à 256 bits de sécurité . . .)

On préfère donc la plupart du temps utiliser un *chiffrement hybride*

Chiffrement hybride

Si Alice souhaite envoyer un (grand) message m à Bob en utilisant sa clé publique k_e :

- Alice choisit une clé de chiffrement symétrique k
- envoie à Bob $c = E_{\mathrm{asym}}(k_e, k) \parallel E_{\mathrm{sym}}(k, m)$
- Bob récupère k avec sa clé privée k_d
- puis déchiffre le reste du message avec k pour récupérer m

Aujourd'hui

Chiffrement asymétrique

Problème du logarithme discret

Applications du DLP : Diffie-Hellman, ElGamal

Cryptanalyse

Cadre général

On travaille dans un *groupe abélien* $(G, \boldsymbol{+})$ *i.e.*

un ensemble G d'éléments muni d'une opération interne $\boldsymbol{+}$ pour laquelle

- $a \boldsymbol{+} b = b \boldsymbol{+} a$ pour tout $a, b \in G$
- $(a \boldsymbol{+} b) \boldsymbol{+} c = a \boldsymbol{+} (b \boldsymbol{+} c)$ pour tout $a, b, c \in G$
- il existe un neutre $0_G \in G$ pour lequel $a \boldsymbol{+} 0_G = 0_G \boldsymbol{+} a = a$ pour tout $a \in G$
- chaque élément $a \in G$ admet un opposé $\overline{a}$ pour lequel $a \boldsymbol{+} \overline{a} = \overline{a} \boldsymbol{+} a = 0_G$.

Exemples de groupes abéliens

- $(\mathbb{Z}, +)$, $(\mathbb{Q}, +)$, $(\mathbb{R}, +)$, $(\mathbb{C}, +)$, $\ldots$
- $(\mathbb{Q}^*, \cdot\,)$, $(\mathbb{R}^*, \cdot\,)$, $(\mathbb{C}^*, \cdot\,)$, $\ldots$
- $(\mathbb{Z}/n\mathbb{Z}, +)$ pour $n > 1$ entier : groupe cyclique
- $((\mathbb{Z}/p\mathbb{Z})^*, \cdot\,)$ avec p premier : groupe multiplicatif à $p-1$ éléments
- courbes elliptiques

Opération itérée

Étant donné $g \in G$ et un entier $n \in \mathbb{N}$, on peut itérer $n-1$ fois l'opération :

$$n \,\boldsymbol{\times}\, g := \underbrace{g \,\boldsymbol{+}\, g \,\boldsymbol{+}\, \cdots \,\boldsymbol{+}\, g}_{n}.$$

En convenant que $0 \,\boldsymbol{\times}\, g = 0_G$ et $(-n) \,\boldsymbol{\times}\, g = n \,\boldsymbol{\times}\, \overline{g}$, cette opération d'itération satisfait les propriétés habituelles :

- $(m+n) \,\boldsymbol{\times}\, g = (m \,\boldsymbol{\times}\, g) \,\boldsymbol{+}\, (n \,\boldsymbol{\times}\, g)$ pour tout $m, n \in \mathbb{Z}$, $g \in G$
- $n \,\boldsymbol{\times}\, (g \,\boldsymbol{+}\, h) = (n \,\boldsymbol{\times}\, g) \,\boldsymbol{+}\, (n \,\boldsymbol{\times}\, h)$ pour tout $n \in \mathbb{Z}$, $g, h \in G$
- $(m \cdot n) \,\boldsymbol{\times}\, g = m \,\boldsymbol{\times}\, (n \,\boldsymbol{\times}\, g)$ pour tout $m, n \in \mathbb{Z}$, $g \in G$

NB : lorsque $\boldsymbol{+}$ est une multiplication, ce ne sont que les *lois des exposants* !

Ordre d'un élément

Definition

On appelle *ordre* d'un élément $g \in G$ le plus petit entier $n > 0$ pour lequel

$$n \times g = 0_G$$

qu'on note $\mathrm{ord}_G(g)$. S'il n'en existe pas, on convient que $\mathrm{ord}_G(g) = +\infty$.

On peut montrer qu'en général

$$n \times g = 0_G \quad \Longleftrightarrow \quad \mathrm{ord}_G(g) \text{ divise } n.$$

Definition (logarithme discret)

$$\mathrm{dlog}_G(x, g) := \ell \iff x = \ell \,\text{✖}\, g$$

La terminologie vient du cadre multiplicatif, même si les groupes d'intérêts aujourd'hui sont additifs.

Remarque : si $\ell \underset{\mathrm{ord}_G(g)}{\equiv} \ell'$ alors $\ell \,\text{✖}\, g = \ell' \,\text{✖}\, g$ (et vice-versa)

donc le logarithme discret n'est bien défini que modulo $\mathrm{ord}_G(g)$.

Exemple dans $\mathbb{Z}/2039\mathbb{Z}$ avec $+$

Prenons $g = 2$.

- $\text{ord}(2) = 2039$

- $\text{dlog}(15, 2) \equiv 1027$

(facile)

Exemple dans $(\mathbb{Z}/2039\mathbb{Z})^*$ avec $\cdot$

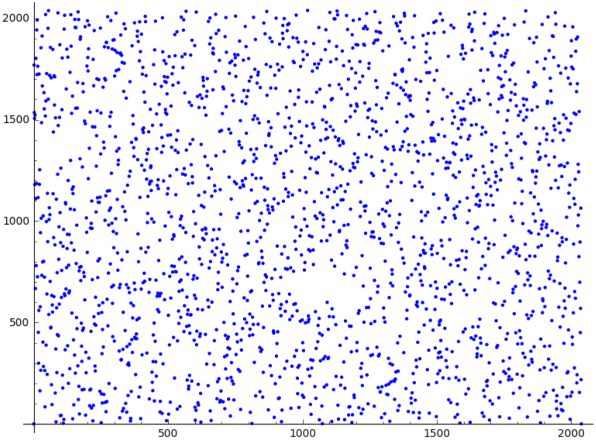
Toujours avec $g = 2$.

- $\mathrm{ord}(2) = 1019$

- $\mathrm{dlog}(15, 2) \equiv 655$

(moins facile – on peut cacher de l'information dans les exposants !)

Example: $x \underset{2039}{\equiv} 2^{\ell}$



Aujourd'hui

Chiffrement asymétrique

Problème du logarithme discret

Applications du DLP : Diffie-Hellman, ElGamal

Cryptanalyse

Partage de secret

Le chiffrement à clé publique fournit une solution au problème de partage de clé privée pour pouvoir utiliser du chiffrement symétrique sur un canal non sécurisé :

- Alice choisit une clé secrète k,
- la chiffre avec la clé publique de chiffrement de Bob,
- et lui envoie;
- Bob récupère k en utilisant sa clé privée de déchiffrement.

Y a-t-il des problèmes avec ce système ? (indice: oui)

Version symétrique

- Alice choisit k_A et l'envoie à Bob en utilisant sa clé de chiffrement publique;
- Bob choisit k_B l'envoie à Alice en utilisant sa clé de chiffrement publique;
- la secret partagé final est $k := k_A \oplus k_B$.

Mieux puisque ni Alice, ni Bob ne contrôle le secret final.

Mais deux clés publiques de chiffrement sont nécessaires. . .

Diffie-Hellman (1976)

- Alice et Bob s'entendent sur un groupe $(G, \boldsymbol{+})$ et $g \in G$ pour lequel le problème du logarithme discret est considéré difficile.
- Alice choisit m, calcule $a = m \boldsymbol{\times} g$ et l'envoie à Bob.
- Bob choisit n, calcule $b = n \boldsymbol{\times} g$ l'envoie à Alice.

Le secret partagé est

$$k := (m \cdot n) \boldsymbol{\times} g = m \boldsymbol{\times} b = n \boldsymbol{\times} a.$$

Le problème de Diffie-Hellman

L'attaquante Ève doit résoudre le problème :

connaissant a et b, retrouver k.

On *croit* que la meilleure attaque consiste à :

- calculer $m = \mathrm{dlog}_G(a, g)$ ou $n = \mathrm{dlog}_G(b, g)$
- puis calculer aisément $k = (m \cdot n) \times g$ comme le feraient Alice ou Bob.

Précautions

- DH doit **toujours** être utilisé avec de l'authentification pour éviter les attaques *personne-dans-le-milieu* (PitM)



- Bob doit vérifier qu'Alice ne fournit pas une valeur de a pour laquelle le logarithme discret est facile à calculer (et de même du côté d'Alice).

Chiffrement ElGamal (1984)

Essentiellement Diffie-Hellman + masque à usage unique

Paramètres publics: G et $g \in G$ pour lequel le DLP est difficile

Clés:

- d clé privée de déchiffrement
- $e := d \ast g$ clé publique de chiffrement

Alice souhaite envoyer un message $m \in G$ à Bob.

Chiffrement

- Alice choisit un entier aléatoire m, calcule $s := a \text{✖} g$
- Calcule le secret partagé $k = a \text{✖} e$
- Calcule le chiffré $c = m \text{✚} k$
- Transmet la paire (s, c)

Déchiffrement

À la réception d'une paire (s, c), Bob

- Calcule le secret partagé $k = d \mathbin{\pmb{\times}} s$
- Récupère $m = c \mathbin{\pmb{+}} \overline{k}$

(Les mêmes précautions que pour D-H s'appliquent)

Aujourd'hui

Chiffrement asymétrique

Problème du logarithme discret

Applications du DLP : Diffie-Hellman, ElGamal

Cryptanalyse

Attaquer le DLP

ou : *comment calculer des logarithmes discrets*

$$\mathrm{dlog}_G(x, g) = \ell \quad \Longleftrightarrow \quad x = \ell \ \text{✖}\ g$$

Algorithme naïf: recherche de ℓ par force brute

On trouvera en $\mathcal{O}(\mathrm{ord}_G(g)) \leq \mathcal{O}(|G|)$ étapes

$\implies$ on veut que l'ordre de g grand (et donc en particulier le nombre d'éléments de G)

Théorème des restes chinois

Si l'ordre de g est composé, on peut réduire la complexité du calcul du logarithme discret.

En effet, si $\mathrm{ord}_G(g) = m \cdot n$ avec m et n premiers entre eux, on peut montrer que l'équation

$$\ell \mathbin{\text{✖}} g = x$$

est équivalente au système d'équations

$$\begin{cases} \ell \mathbin{\text{✖}} (m \mathbin{\text{✖}} g) = (m \mathbin{\text{✖}} x) \\ \ell \mathbin{\text{✖}} (n \mathbin{\text{✖}} g) = (n \mathbin{\text{✖}} x) \end{cases}$$

qui permet de récupérer $\ell \mod n$ et $\ell \mod m$, donc $\ell \mod m \cdot n$.

On demande donc que $\mathrm{ord}_G(g)$ soit premier.

Exemple

Calculer $\mathrm{dlog}(9, 2)$ dans $(\mathbb{Z}/13\mathbb{Z})^*$.

Baby-step giant-step

Compromis temp/mémoire pour calculer $\ell \underset{\mathrm{ord}(g)}{\equiv} \mathrm{dlog}_G(x, g)$.

Choisissons une base β et écrivons $\ell = i\beta + j$.

Petits pas:

Calculer et stocker les valeurs de $j \,\text{✖}\, g$ pour $j \in [\![0, \beta[\![$ dans une table

Grands pas:

tant que x n'est pas dans cette table, lui soustraire $\beta \,\text{✖}\, g$.

Baby-step giant-step

En d'autres termes: les petits logarithmes ($< \beta$) sont lus dans la table pré-calculée.

Dans le cas général pour x, on cherche une version modifiée $x \,\pmb{+}\, (-i\beta) \,\pmb{\times}\, g$ pour lequel le logarithme est petit.

$$x \,\pmb{+}\, (-i\beta) \,\pmb{\times}\, g = j \,\pmb{\times}\, g \iff x = (i\beta + j) \,\pmb{\times}\, g \iff \mathrm{dlog}_G(x, g) = i\beta + j.$$

Complexité temporelle: $\mathcal{O}(\beta) + \mathcal{O}(\frac{\mathrm{ord}(g)}{\beta})$

Complexité spatiale: $\mathcal{O}(\beta)$

On prend souvent $\beta \approx \sqrt{\mathrm{ord}(g)}$ pour obtenir un complexité globale $\mathcal{O}(\sqrt{\mathrm{ord}(g)})$.

Exemple : calculer $\mathrm{dlog}_{2039}(15, 2)$

$n = 2039$

$g = 2$

$\mathrm{ord}(g) = 1019$

$\sqrt{\mathrm{ord}(g)} \approx 32$

donc on peut prendre $\beta \approx 32$ et trouver la réponse en au plus 32 grands pas

Réponse : $\ell = 655$ (facile à vérifier !)

Autres algorithmes

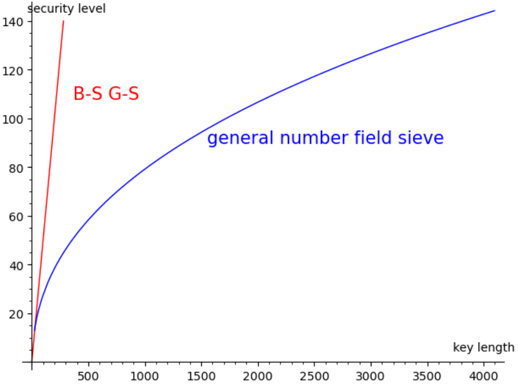
Il y a un algorithme probabiliste pour le DLP qui prend (en moyenne) $\mathcal{O}(\sqrt{\mathrm{ord}(g)})$ étapes (et $\mathcal{O}(1)$ mémoire)

Par contre : on dispose de *bien meilleurs* algorithmes pour résoudre le DLP *modulaire*

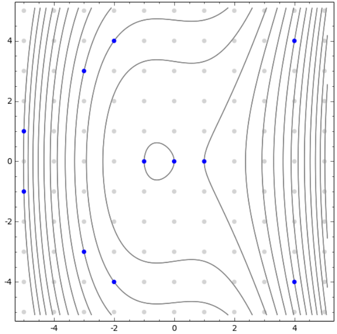
$\implies$ même longueurs de clés que pour RSA

records actuels

Calcul du DLP



Courbes elliptiques



Les meilleurs algorithmes connus sont les algorithmes *génériques*

$\Longrightarrow$ sécurité à n bits atteinte avec des clés de seulement $2n$ bits ☺