Introduction à la cryptographie

Vanessa VITSE

Université Grenoble Alpes

M1 Maths 2021

Section 4

Le problème du logarithme discret

Fonction à sens unique

Rappel : la fonction exponentielle modulaire $x \mapsto g^x \bmod p$ avec p premier, est une fonction *à sens unique*

- algorithme d'exponentiation rapide avec complexité polynomiale
- pas d'algorithme efficace (polynomial) pour calculer la fonction réciproque

Définition (DLP)

Soit G un groupe muni d'une loi de groupe notée multiplicativement. Le *problème du logarithme discret* est

étant donnés $g, h \in G$, trouver (s'il existe) x tel que $h = g^x$

- dans ce cours on considérera seulement $G = \mathbb{Z}/p\mathbb{Z}^*$ mais il existe des groupes algébriques plus efficaces utilisés en cryptographie

Fonction à sens unique

Rappel : la fonction exponentielle modulaire $x \mapsto g^x \bmod p$ avec p premier, est une fonction *à sens unique*

- algorithme d'exponentiation rapide avec complexité polynomiale
- pas d'algorithme efficace (polynomial) pour calculer la fonction réciproque

Définition (DLP)

Soit G un groupe muni d'une loi de groupe notée multiplicativement. Le *problème du logarithme discret* est

étant donnés $g, h \in G$, trouver (s'il existe) x tel que $h = g^x$

- dans ce cours on considérera seulement $G = \mathbb{Z}/p\mathbb{Z}^*$ mais il existe des groupes algébriques plus efficaces utilisés en cryptographie
- le DLP permet de construire des protocoles cryptographiques asymétriques

Section 5

Le logarithme discret pour le chiffrement

Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile

Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile
2. Alice tire au hasard $a \in [\![2, n-1]\!]$, calcule $A = g^a$, l'envoie à Bob

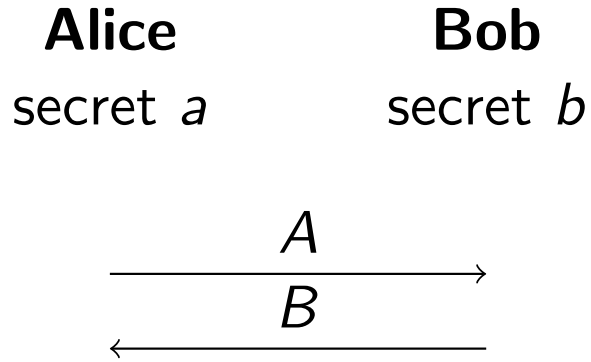
Alice

secret a

$$\xrightarrow{\quad A \quad}$$

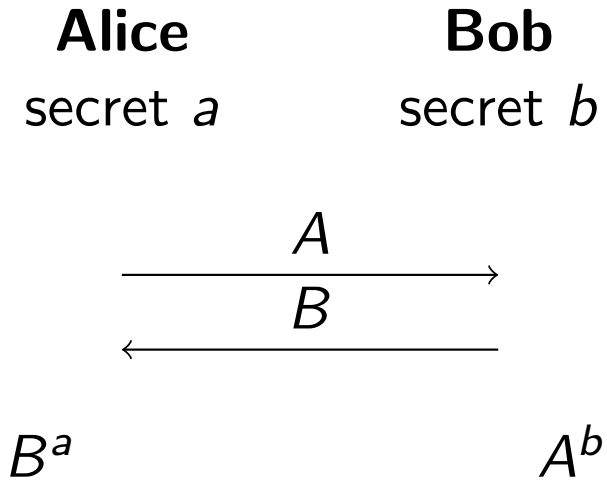
Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile
2. Alice tire au hasard $a \in [\![2, n-1]\!]$, calcule $A = g^a$, l'envoie à Bob
3. Bob tire au hasard $b \in [\![2, n-1]\!]$, calcule $B = g^b$, l'envoie à Alice



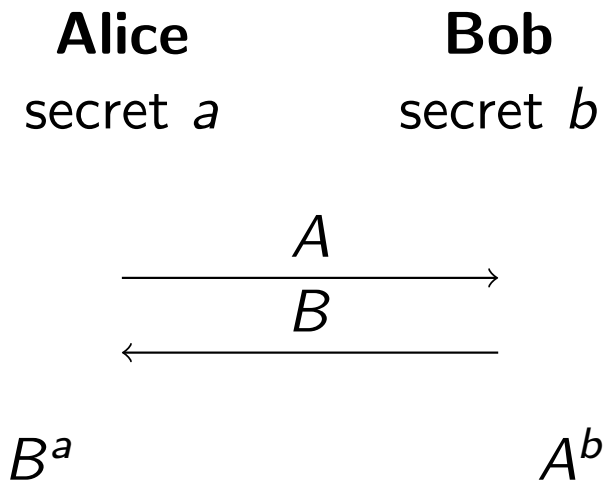
Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile
2. Alice tire au hasard $a \in [\![2, n-1]\!]$, calcule $A = g^a$, l'envoie à Bob
3. Bob tire au hasard $b \in [\![2, n-1]\!]$, calcule $B = g^b$, l'envoie à Alice
4. Alice calcule B^a, Bob calcule A^b



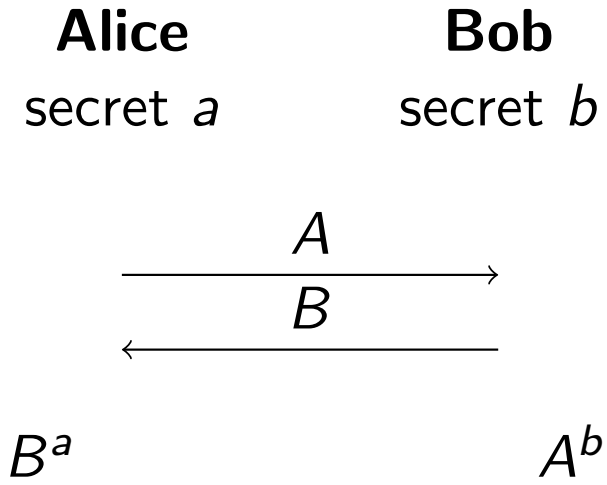
Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile
2. Alice tire au hasard $a \in [\![2, n-1]\!]$, calcule $A = g^a$, l'envoie à Bob
3. Bob tire au hasard $b \in [\![2, n-1]\!]$, calcule $B = g^b$, l'envoie à Alice
4. Alice calcule B^a, Bob calcule A^b
5. secret commun : $B^a = A^b = g^{ab}$



Échange de clef Diffie-Hellman

1. Alice et Bob choisissent publiquement un groupe G et $g \in G$ d'ordre fini n, dans lequel le DLP est difficile
2. Alice tire au hasard $a \in [\![2, n-1]\!]$, calcule $A = g^a$, l'envoie à Bob
3. Bob tire au hasard $b \in [\![2, n-1]\!]$, calcule $B = g^b$, l'envoie à Alice
4. Alice calcule B^a, Bob calcule A^b
5. secret commun : $B^a = A^b = g^{ab}$

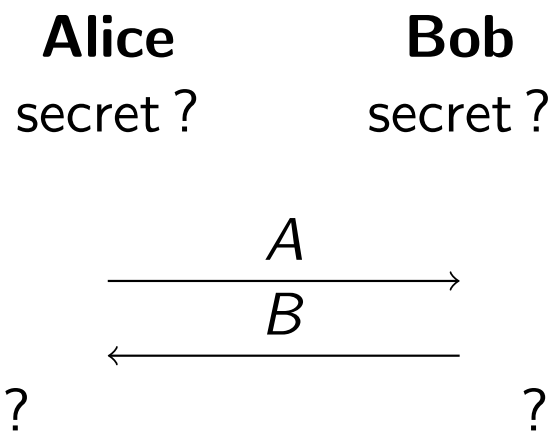


- premier protocole cryptographique ne nécessitant pas l'utilisation d'une clef secrète
- utilisable directement avec cryptosystème symétrique (ex : `https`)

Diffie-Hellman : sécurité

Problème calculatoire Diffie-Hellman (CDHP)

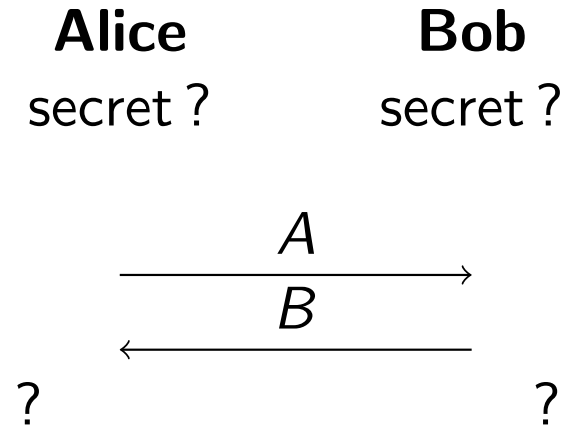
Ève l'espionne connait g, g^a, g^b.
Peut-elle retrouver g^{ab} ?



Diffie-Hellman : sécurité

Problème calculatoire Diffie-Hellman (CDHP)

Ève l'espionne connait g, g^a, g^b.
Peut-elle retrouver g^{ab} ?

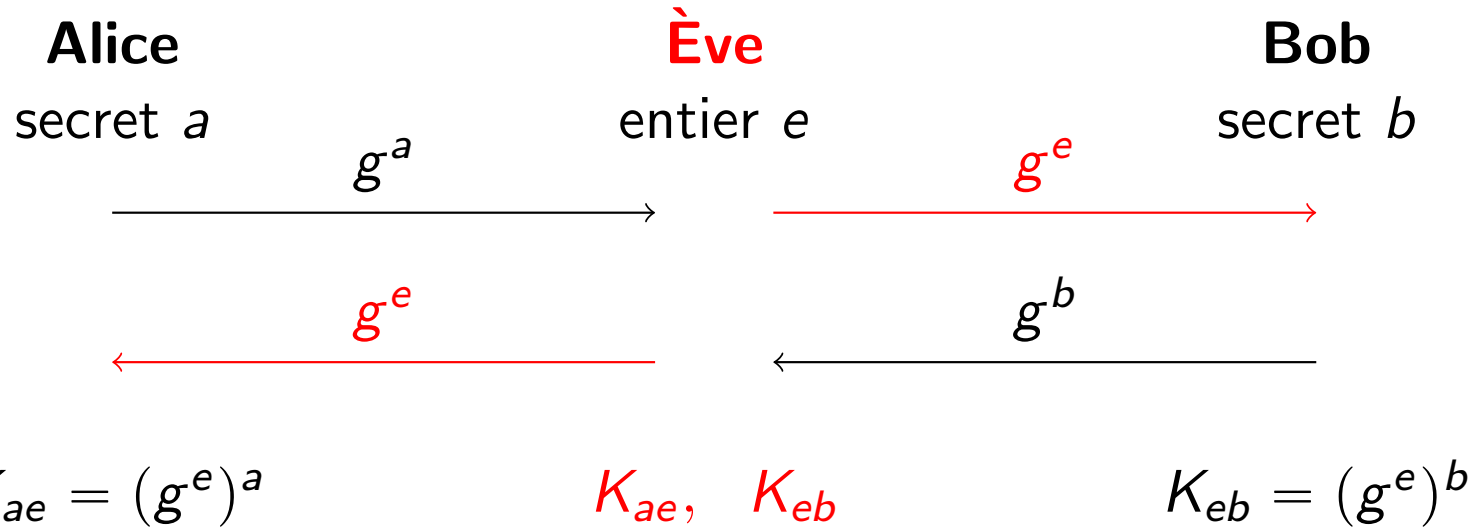


Relation CDHP/DLP

- si Ève sait résoudre le DLP, alors elle sait résoudre le CDHP
- le sens contraire est moins clair mais communément accepté comme vrai
- le protocole d'échange de clef Diffie-Hellman est donc sûr contre un attaquant passif

Attaque de "la femme-du-milieu"

Sans authentification d'Alice et Bob, un adversaire actif peut mener une attaque efficace :



Chiffrement à clef publique ElGamal

On peut définir un chiffrement asymétrique basé sur le DLP (non utilisé en pratique)

1. *Génération de clef* : Alice choisit $G = \langle g \rangle$ où le DLP est difficile, génère un entier aléatoire a et calcule $K_a = g^a$.
   Sa clef publique est (g, K_a), sa clef secrète est a.
2. *Chiffrement* : pour chiffrer $m \in G$, Bob génère une clef temporaire $K_t = g^t$ où t est un entier choisi aléatoirement et calcule $c = m(K_a)^t$.
   Le message chiffré est (K_t, c).
3. *Déchiffrement* : pour déchiffrer $((K_t, c)$, Alice calcule $(K_t)^{-a}c$ et retrouve ainsi le clair m.

Chiffrement à clef publique ElGamal

On peut définir un chiffrement asymétrique basé sur le DLP (non utilisé en pratique)

1. *Génération de clef* : Alice choisit $G = \langle g \rangle$ où le DLP est difficile, génère un entier aléatoire a et calcule $K_a = g^a$.
   Sa clef publique est (g, K_a), sa clef secrète est a.
2. *Chiffrement* : pour chiffrer $m \in G$, Bob génère une clef temporaire $K_t = g^t$ où t est un entier choisi aléatoirement et calcule $c = m(K_a)^t$.
   Le message chiffré est (K_t, c).
3. *Déchiffrement* : pour déchiffrer $((K_t, c)$, Alice calcule $(K_t)^{-a}c$ et retrouve ainsi le clair m.

- masquage du message avec $K_a^t = K_t^a = g^{at}$
- chiffrement non déterministe malléable, que l'on peut rendre non-malléable en utilisant un bourrage (OAEP ou PKCS)

Génération de paramètres

Comment choisir p premier tel que le DLP dans $(\mathbb{Z}/p\mathbb{Z})^*$ est difficile ?

Rappels :

- le groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z})^*$ est cyclique
- il admet $\varphi(p-1)$ racines primitives (générateurs) et $\varphi(n)/n \in \Omega(1/\ln\ln n)$
- un élément $g \in (\mathbb{Z}/p\mathbb{Z})^*$ est d'ordre d si (et seulement si)

$$g^d = 1 \text{ et } g^{d/q} \neq 1 \text{ pour tout diviseur premier } q \text{ de } d.$$

Génération de paramètres

Comment choisir p premier tel que le DLP dans $(\mathbb{Z}/p\mathbb{Z})^*$ est difficile ?

Rappels :

- le groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z})^*$ est cyclique
- il admet $\varphi(p-1)$ racines primitives (générateurs) et $\varphi(n)/n \in \Omega(1/\ln\ln n)$
- un élément $g \in (\mathbb{Z}/p\mathbb{Z})^*$ est d'ordre d si (et seulement si)

$$g^d = 1 \text{ et } g^{d/q} \neq 1 \text{ pour tout diviseur premier } q \text{ de } d.$$

En pratique...

- on tire un élément aléatoire et si la factorisation de $p-1$ est connue, on teste si son ordre est bien $p-1$
- si p est de taille crypto, la factorisation de $p-1$ peut être difficile
- autre possibilité : générer p un premier de Sophie Germain, g est une racine primitive ssi $g^2 \neq 1$ et $g^{(p-1)/2} \neq 1$

Section 6

Attaques génériques sur le DLP

Attaque générique

Définition

Un algorithme de calcul de DLP est générique s'il n'utilise que

- la loi de groupe
- le test d'égalité entre deux éléments

Un tel algorithme s'applique alors à n'importe quel groupe G.

Attaque générique

Définition

Un algorithme de calcul de DLP est générique s'il n'utilise que

- la loi de groupe
- le test d'égalité entre deux éléments

Un tel algorithme s'applique alors à n'importe quel groupe G.

Exemple : la recherche exhaustive
calculer g^a pour $a = 0, 1, \ldots$ jusqu'à tomber sur h la valeur cible
complexité en $O(\text{ord}(g))$

Un problème à difficulté variable

La difficulté du DLP dépend du groupe G choisi

- si $G = (\mathbb{Z}/n\mathbb{Z}, +)$, c'est très facile
- si $G = (\mathbb{Z}/p\mathbb{Z}^*, \times)$, c'est difficile (complexité sous-exponentielle en $e^{O((\log p)^{1/3}(\log\log p)^{2/3})}$)
- si G est l'ensemble des points rationnels d'une courbe elliptique définie sur $\mathbb{F}_p$, c'est encore plus difficile (complexité en $O(\sqrt{(p)})$)

Un problème à difficulté variable

La difficulté du DLP dépend du groupe G choisi

- si $G = (\mathbb{Z}/n\mathbb{Z}, +)$, c'est très facile
- si $G = (\mathbb{Z}/p\mathbb{Z}^*, \times)$, c'est difficile (complexité sous-exponentielle en $e^{O((\log p)^{1/3}(\log\log p)^{2/3})}$)
- si G est l'ensemble des points rationnels d'une courbe elliptique définie sur $\mathbb{F}_p$, c'est encore plus difficile (complexité en $O(\sqrt{(p)})$)

Dans le dernier cas, on ne sait actuellement pas faire mieux que les attaques génériques

Algorithmes génériques classiques

On considère le DLP dans G groupe d'ordre n

- Recherche exhaustive : $O(n)$ en temps et $O(1)$ en mémoire
- Pohlig-Hellman + baby-step-giant-step (cf TD) : $O(\log(n)\sqrt{p})$ en temps et $O(\sqrt{p})$ en mémoire, où p plus grand facteur premier de n
- Pohlig-Hellman + Pollard-rho : $O(\log(n)\sqrt{p})$ en temps et $O(1)$ en mémoire mais probabiliste

Algorithmes génériques classiques

On considère le DLP dans G groupe d'ordre n

- Recherche exhaustive : $O(n)$ en temps et $O(1)$ en mémoire
- Pohlig-Hellman + baby-step-giant-step (cf TD) : $O(\log(n)\sqrt{p})$ en temps et $O(\sqrt{p})$ en mémoire, où p plus grand facteur premier de n
- Pohlig-Hellman + Pollard-rho : $O(\log(n)\sqrt{p})$ en temps et $O(1)$ en mémoire mais probabiliste

Théorème de Shoup

Pour tout algorithme générique, il existe une constante c telle que l'espérance du nombre d'opérations nécessaires à résoudre le DLP dans un groupe cyclique d'ordre n est plus grande que $c\sqrt{p}$ où p est le plus grand facteur premier de n.

En particulier Pollard-Rho est optimal (hors parallélisation)

Un préliminaire incontournable : le paradoxe des anniversaires

Soit E un ensemble à N éléments, on tire avec remise des éléments dans E.

- $(Y_i)_{i\in\mathbb{N}*}$ les v.a.i.i.d. correspondantes, $Y_i \sim \mathcal{U}(E)$
- X la v.a. comptant le nombre de tirages nécessaires pour d'obtenir deux fois le même élément :

$$X = \min\{i \in \mathbb{N}^* : \exists j \in \mathbb{N}^*, j < i \text{ et } Y_i = Y_j\}.$$

Théorème

$$E(X) \leqslant \sqrt{\pi N/2} + 2$$

Preuve paradoxe des anniversaires

La probabilité qu'un élément soit distinct des $i-1$ tirés précédemment est $1-(i-1)/N$ donc

$$\Pr[X > k] = \prod_{i=1}^{k-1}(1-\frac{i}{N}) \leqslant \prod_{i=1}^{k-1} e^{-i/N} = e^{-\frac{k(k-1)}{2N}} \leqslant e^{-\frac{(k-1)^2}{2N}}.$$

Comme $\Pr[X > k] = 0$ for $k > N$, on a

$$\mathrm{E}(X) = \sum_{k\geqslant 1} k\Pr[X=k] = \sum_{k\geqslant 1} k(\Pr[X>k-1]-\Pr[X>k]) = \sum_{k\geqslant 0}\Pr[X>k]$$

Avec l'inégalité du début, on a

$$\mathrm{E}(X) \leqslant 1+\sum_{k\geqslant 0} e^{-\frac{k^2}{2N}} \leqslant 1+1+\int_0^{+\infty} e^{-\frac{t^2}{2N}}\,dt = 2+\sqrt{2N}\int_0^{+\infty} e^{-u^2}\,du$$

$$\mathrm{E}(X) \leqslant 2+\sqrt{\frac{N\pi}{2}}$$

Pollard-Rho

Soit G un groupe, $g \in G$ d'ordre r, et $h \in \langle g \rangle$

Principe

Itérer une fonction $F : G \to G$

- pour tous $\alpha, \beta \in \mathbb{Z}/r\mathbb{Z}$, il est facile de calculer $\alpha', \beta' \in \mathbb{Z}/r\mathbb{Z}$ tels que $F(g^\alpha h^\beta) = g^{\alpha'} h^{\beta'}$
- F se comporte "quasiment" comme une fonction aléatoire

Pollard-Rho

Soit G un groupe, $g \in G$ d'ordre r, et $h \in \langle g \rangle$

Principe

Itérer une fonction $F : G \to G$

- pour tous $\alpha, \beta \in \mathbb{Z}/r\mathbb{Z}$, il est facile de calculer $\alpha', \beta' \in \mathbb{Z}/r\mathbb{Z}$ tels que $F(g^\alpha h^\beta) = g^{\alpha'} h^{\beta'}$
- F se comporte "quasiment" comme une fonction aléatoire

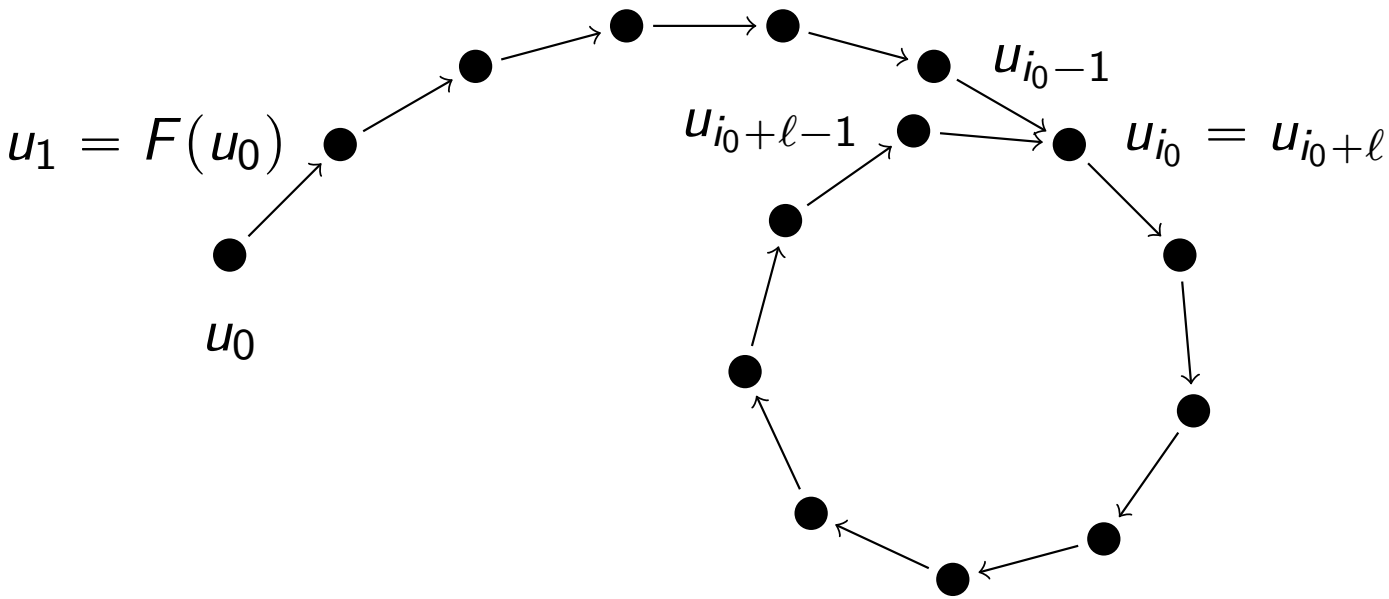
Exemple : prendre G_1, G_2, G_3 une partition de G et F

$$F(x) = \begin{cases} x^2 & \text{si } x \in G_1, \\ g \cdot x & \text{si } x \in G_2, \\ h \cdot x & \text{si } x \in G_3. \end{cases}$$

Pollard-Rho

On part de $u_0 = g^{\alpha_0} h^{\beta_0}$, on calcule $(u_i), (\alpha_i), (\beta_i)$ tels que

$$u_i = F(u_{i-1}) = F^i(u_0) = g^{\alpha_i} h^{\beta_i}.$$



Pollard-Rho

On part de $u_0 = g^{\alpha_0} h^{\beta_0}$, on calcule $(u_i), (\alpha_i), (\beta_i)$ tels que

$$u_i = F(u_{i-1}) = F^i(u_0) = g^{\alpha_i} h^{\beta_i}.$$

(u_i) est *ultimement périodique* : $\exists i_0, \ell > 0, u_{i_0} = u_{i_0+\ell}$ et cette collision donne

$$g^{\alpha_{i_0}} h^{\beta_{i_0}} = g^{\alpha_{j_0}} h^{\beta_{j_0}}.$$

Pollard-Rho

On part de $u_0 = g^{\alpha_0} h^{\beta_0}$, on calcule $(u_i), (\alpha_i), (\beta_i)$ tels que

$$u_i = F(u_{i-1}) = F^i(u_0) = g^{\alpha_i} h^{\beta_i}.$$

(u_i) est *ultimement périodique* : $\exists i_0, \ell > 0, u_{i_0} = u_{i_0+\ell}$ et cette collision donne

$$g^{\alpha_{i_0}} h^{\beta_{i_0}} = g^{\alpha_{j_0}} h^{\beta_{j_0}}.$$

Deux possibilités

- soit $\beta_{i_0} - \beta_{j_0}$ est premier à r, alors

$$\log_g(h) = -(\alpha_{i_0} - \alpha_{j_0})(\beta_{i_0} - \beta_{j_0})^{-1} \bmod r.$$

- sinon on change de u_0

Analyse de Pollard-Rho

Paradoxe des anniversaires :
La première collision est attendue après $O(\sqrt{r})$ itérations de F *si* F *est vraiment aléatoire*.
$\longrightarrow$ hypothèse fausse mais heuristique validée par l'expérience

Analyse de Pollard-Rho

Paradoxe des anniversaires :
La première collision est attendue après $O(\sqrt{r})$ itérations de F *si* F *est vraiment aléatoire*.
$\rightarrow$ hypothèse fausse mais heuristique validée par l'expérience

Détection des collisions :
Stocker tous les u_i coûte $O(\sqrt{r})$ en mémoire, on peut faire mieux :

le lièvre et la tortue de Floyd

On calcule les termes successifs de (u_i) (la tortue) et de $(v_i) = (u_{2i})$ (le lièvre) jusqu'à obtenir la première collision $u_i = v_i$.

Analyse : si le cycle (u_i) est l-périodique à partir du rang i_0, alors la collision s'obtiendra avant l'itération $i_0 + l$.

Section 7

Le logarithme discret en authentification

Principe de la signature numérique

But

Assurer l'*intégrité* du message et l'*authentification* de l'auteur

- Bob veut envoyer un message m à Alice (chiffré ou non)
- il calcule avec sa clef privée une signature σ du message et envoie (m, σ) à Alice
- Alice vérifie avec la clef publique de Bob que σ est bien la signature par Bob de m

Principe de la signature numérique

But

Assurer l'*intégrité* du message et l'*authentification* de l'auteur

- Bob veut envoyer un message m à Alice (chiffré ou non)
- il calcule avec sa clef privée une signature σ du message et envoie (m, σ) à Alice
- Alice vérifie avec la clef publique de Bob que σ est bien la signature par Bob de m

Trois fonctions dans un tel schéma :

1. fonction de génération des couples (p_K, s_k) de clefs publique / privée
2. fonction de signature $Sig : (m, s_k) \mapsto \sigma$
3. fonction de vérification $Ver : (m, \sigma, p_k) \mapsto$ Vrai ou Faux

Signature et sécurité

Falsification

Un adversaire (ne connaissant que p_k) ne doit pas pouvoir facilement

- se faire passer pour Bob, en fabriquant un couple message/signature (m, σ) valides (i.e. tel que $Ver(m, \sigma, p_k) = \text{Vrai}$)
  $\longrightarrow$ falsification *existencielle*
- signer un message donné m à la plage de Bob (i.e. fabriquer σ tel que $Ver(m, \sigma, p_k) = \text{Vrai}$)
  $\longrightarrow$ falsification *universelle*

Signature et sécurité

Falsification

Un adversaire (ne connaissant que p_k) ne doit pas pouvoir facilement

- se faire passer pour Bob, en fabriquant un couple message/signature (m, σ) valides (i.e. tel que $Ver(m, \sigma, p_k) = \text{Vrai}$)
  $\rightarrow$ falsification *existencielle*
- signer un message donné m à la plage de Bob (i.e. fabriquer σ tel que $Ver(m, \sigma, p_k) = \text{Vrai}$)
  $\rightarrow$ falsification *universelle*

Remarque : on veut des signatures courtes, taille fixée à n bits *indépendamment* de la longeur de m

- recherche exhaustive : falsification universelle en $O(2^n)$
- paradoxe des anniversaires : falsification existencielle en $O(2^{n/2})$

Fonctions de hachage cryptographiques

Définition

Une **fonction de hachage** est une application $H : \{0,1\}^* \to \{0,1\}^n$

- prend en entrée un chaine de bits de taille arbitraire
- retourne rapidement un chaine de bits de taille fixée n

Nombreuses utilités : bases de données, indexation, ... et crypto !

Remarques :

- chaque "haché" $y \in \{0,1\}^n$ admet de nombreux antécédents
- en cryptographie, on veut que de tels antécédents soient difficiles à trouver

Conditions cryptographiques

- *résistance pré-image :*
  $y \in \{0;1\}^n$ donné $\rightarrow$ difficile de trouver x tel que $H(x) = y$
- *résistance 2nd-préimage :*
  $x \in \{0;1\}^*$ et $y = H(x)$ donnés $\rightarrow$ difficile de trouver $x' \neq x$ tel que $H(x') = y$.
- *résistance au collision :*
  difficile de trouver $x \neq x'$ tel que $H(x) = H(x')$.

Conditions cryptographiques

- *résistance pré-image :*
  $y \in \{0;1\}^n$ donné $\rightarrow$ difficile de trouver x tel que $H(x) = y$
- *résistance 2nd-préimage :*
  $x \in \{0;1\}^*$ et $y = H(x)$ donnés $\rightarrow$ difficile de trouver $x' \neq x$ tel que $H(x') = y$.
- *résistance au collision :*
  difficile de trouver $x \neq x'$ tel que $H(x) = H(x')$.

Idéalement

- pour résoudre 1 et 2, pas mieux que force brute en $O(2^n)$
- pour résoudre 3, pas mieux que les anniversaires en $O(2^{n/2})$

Conditions cryptographiques

- *résistance pré-image :*
  $y \in \{0;1\}^n$ donné $\rightarrow$ difficile de trouver x tel que $H(x) = y$
- *résistance 2nd-préimage :*
  $x \in \{0;1\}^*$ et $y = H(x)$ donnés $\rightarrow$ difficile de trouver $x' \neq x$ tel que $H(x') = y$.
- *résistance au collision :*
  difficile de trouver $x \neq x'$ tel que $H(x) = H(x')$.

Idéalement

- pour résoudre 1 et 2, pas mieux que force brute en $O(2^n)$
- pour résoudre 3, pas mieux que les anniversaires en $O(2^{n/2})$

Ex d'application : assurer l'intégrité d'un logiciel à télécharger en publiant son haché sur un site de confiance

Schéma d'identification de Schnorr

L'authentification est un intérêt majeur de la crypto à clef publique

Schéma d'identification basé sur le DLP

Soit G groupe d'ordre premier r et $h = g^a$, a secret d'Alice.
G, r, g et h sont publiés sur un serveur de confiance.

Alice veut montrer à Bob que c'est bien elle qui communique et non Eve (pas de "femme-au-milieu")

- Bob récupère les paramètres publics d'Alice sur le serveur de confiance
- Alice prouve à Bob qu'elle connaît a **sans le lui révéler**

(en pratique : Alice et/ou Bob sont des serveurs, pas des individus !)

Voir les détails en TD

Schéma de signature de Schnorr

Version non-interactive du protocole précédent (transformation de *Fiat-Shamir*)

1. **Génération de clefs :** Alice choisit $G = \langle g \rangle$ d'ordre r et une fonction de hachage H à valeurs dans $\mathbb{Z}/r\mathbb{Z}$.
   Elle génère un aléa a et calcule $h = g^a$: sa clef publique est (G, g, h, r, H), sa clef secrète est a.
2. **Signature :** pour signer le message m, Alice choisit au hasard $0 \leqslant k < r$, et calcule g^k, $s_1 = H(m||g^k)$ et $s_2 = k + as_1 \bmod r$.
   La signature de m est (s_1, s_2).
3. **Vérification :** Bob récupère les paramètres publics d'Alice, puis teste si $s_1 = H(m||g^{s_2} h^{-s_1})$.

Schéma de signature de Schnorr

Version non-interactive du protocole précédent (transformation de *Fiat-Shamir*)

1. **Génération de clefs :** Alice choisit $G = \langle g \rangle$ d'ordre r et une fonction de hachage H à valeurs dans $\mathbb{Z}/r\mathbb{Z}$.
   Elle génère un aléa a et calcule $h = g^a$: sa clef publique est (G, g, h, r, H), sa clef secrète est a.
2. **Signature :** pour signer le message m, Alice choisit au hasard $0 \leqslant k < r$, et calcule g^k, $s_1 = H(m||g^k)$ et $s_2 = k + as_1 \bmod r$.
   La signature de m est (s_1, s_2).
3. **Vérification :** Bob récupère les paramètres publics d'Alice, puis teste si $s_1 = H(m||g^{s_2} h^{-s_1})$.

Par rapport à l'authentification : le *challenge* s_1 de Bob est remplacé par $H(m||g^k)$, "presque" aléatoire si H est sûre

Sécurité du schéma de signature

Signature de Schnorr

1. Clef publique (g, h), clef privée a tel que $h = g^a$
2. Signature d'un message m : couple (s_1, s_2) où $s_1 = H(m||g^k)$, $s_2 = k + as_1 \bmod r$, et k aléa
3. Vérification de $(m, (s_1, s_2))$: test $s_1 \stackrel{?}{=} H(m||g^{s_2} h^{-s_1})$.

Protocole correct : une signature valide passe la vérification

Sécurité du schéma de signature

Signature de Schnorr

1. Clef publique (g, h), clef privée a tel que $h = g^a$
2. Signature d'un message m : couple (s_1, s_2) où $s_1 = H(m||g^k)$, $s_2 = k + as_1 \bmod r$, et k aléa
3. Vérification de $(m, (s_1, s_2))$: test $s_1 \stackrel{?}{=} H(m||g^{s_2} h^{-s_1})$.

Sécurité :

- Si H non sûre : falsification existencielle possible
  choisir (s_1, s_2), calculer $g' = g^{s_2} h^{-s_1}$, puis trouver m tel que $H(m||g') = s_1$
- Si H sûre : la sortie de H est considérée aléatoire
  s_1 ne peut qu'être obtenu par calcul de type $H(m||g')$ avec $g' \in G$.
  Trouver ensuite s_2 qui convient revient à trouver le DL de $g' h^{s_1}$ en base g.

Sécurité du schéma de signature

Signature de Schnorr

1. Clef publique (g, h), clef privée a tel que $h = g^a$
2. Signature d'un message m : couple (s_1, s_2) où $s_1 = H(m||g^k)$, $s_2 = k + as_1 \bmod r$, et k aléa
3. Vérification de $(m, (s_1, s_2))$: test $s_1 \stackrel{?}{=} H(m||g^{s_2} h^{-s_1})$.

Attention : le k utilisé ("nonce") doit être vraiment aléatoire.
En particulier, si un même k est utilisé deux fois, il est facile de retrouver a !