Chiffrement El Gamal et application au vote électronique

Florian Le Manach

Journée des Mathématiques du lycée Pothier

16 avril 2024

Sommaire

1. Introduction
   - Définition
   - Histoire
   - Un modèle formel
2. Le chiffrement El Gamal
   - L'algorithme
   - Choix du groupe
   - Courbes elliptiques
3. Premières applications
   - Signature
   - Échange de clefs Diffie-Hellman
   - Le protocole TLS
4. Vote électronique
   - Système de vote transparent
   - Présentation de Belenios
   - Propriétés cryptographiques
   - Récapitulatif

Le principe de la cryptographie

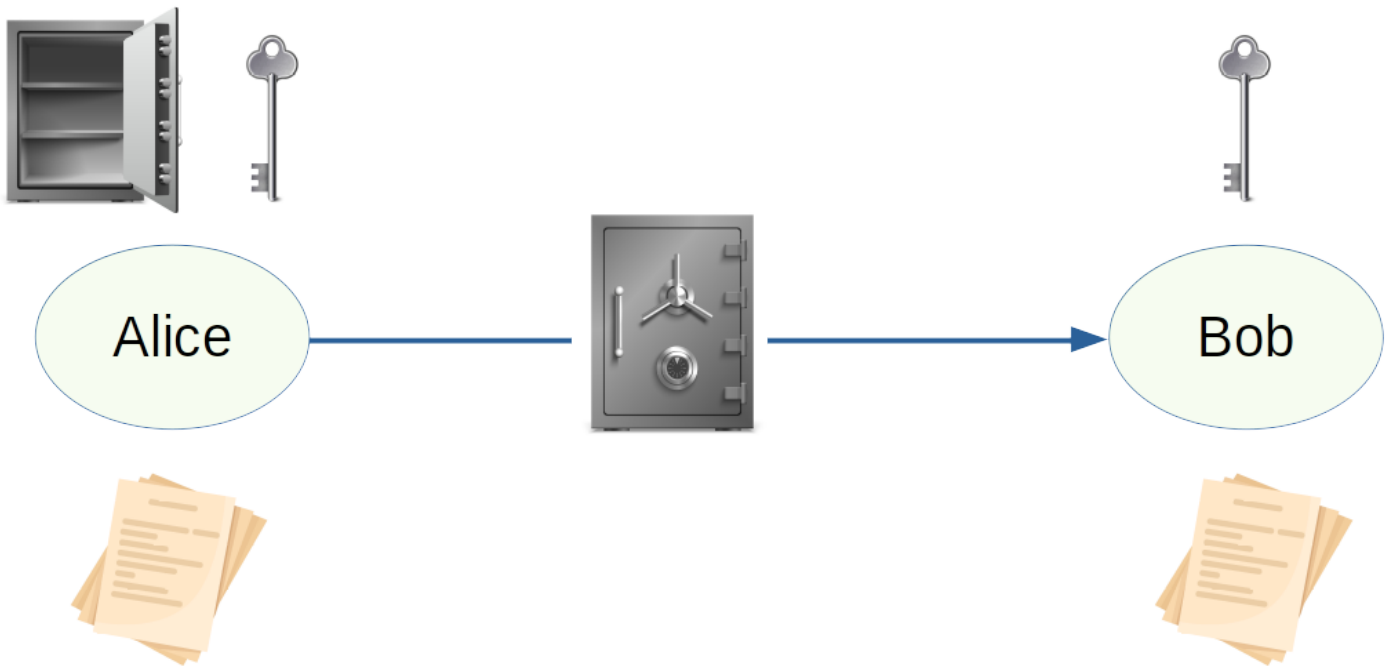
La cryptographie est la discipline qui traite de la transmission confidentielle de données. Elle a pour but de rendre inintelligible un message pour ceux qui ne sont pas habilités à en prendre connaissance.

Le principe de la cryptographie

La cryptographie est la discipline qui traite de la transmission confidentielle de données. Elle a pour but de rendre inintelligible un message pour ceux qui ne sont pas habilités à en prendre connaissance.

Le principe de la cryptographie

La cryptographie est la discipline qui traite de la transmission confidentielle de données. Elle a pour but de rendre inintelligible un message pour ceux qui ne sont pas habilités à en prendre connaissance.



L'âge artisanal (jusqu'à la première guerre mondiale)

L'âge artisanal (jusqu'à la première guerre mondiale)

- Chiffrement par permutation : le scytale ($\sim$ VII^e siècle av. J.-C.)

L'âge artisanal (jusqu'à la première guerre mondiale)

- Chiffrement par permutation : le scytale ($\sim$ VII^e siècle av. J.-C.)

- Chiffrement par décalage : le chiffre de César ($\sim$ I^e siècle av. J.-C.)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

L'âge artisanal (jusqu'à la première guerre mondiale)

- Chiffrement par permutation : le scytale ($\sim$ VII^e siècle av. J.-C.)
- Chiffrement par décalage : le chiffre de César ($\sim$ I^e siècle av. J.-C.)
- Principes de Kerckhoffs (1883)

La sécurité d'un système ne doit pas être fondée sur son caractère secret.
Seule une donnée de petite taille (la clef) doit assurer la sécurité.

L'âge artisanal (jusqu'à la première guerre mondiale)

- Chiffrement par permutation : le scytale ($\sim$ VII^e siècle av. J.-C.)
- Chiffrement par décalage : le chiffre de César ($\sim$ I^e siècle av. J.-C.)
- Principes de Kerckhoffs (1883)
- Chiffrement par substitution : le chiffre de Vigenère (1586)

  Utilise le même principe que le chiffre de César sauf que le décalage de chaque lettre du texte en clair peut différer en fonction de la position de la lettre dans le texte. Ce décalage est calculé à l'aide d'une clef secrète. Cet algorithme, contrairement au chiffre de César, résiste à l'analyse par fréquences mais a été cassé par Kasiski en 1863.

L'âge technique (de la première guerre mondiale à 1975)

L'âge technique (de la première guerre mondiale à 1975)

- Automatisation des permutations et substitutions

Enigma

L'âge technique (de la première guerre mondiale à 1975)

- Automatisation des permutations et substitutions

- Un chiffrement théoriquement parfait : le chiffre de Vernam (1917) dont le principe est le même que le chiffre de Vigenère avec les conditions suivantes :
  1. la clef est choisie de manière totalement aléatoire ;
  2. la clef ne doit être utilisée qu'une seule fois ;
  3. la clef doit être aussi longue que le message à chiffrer.

  Shannon a prouvé dans « *A Mathematical Theory of Communications* » en 1949 que ce chiffrement garde un secret parfait.

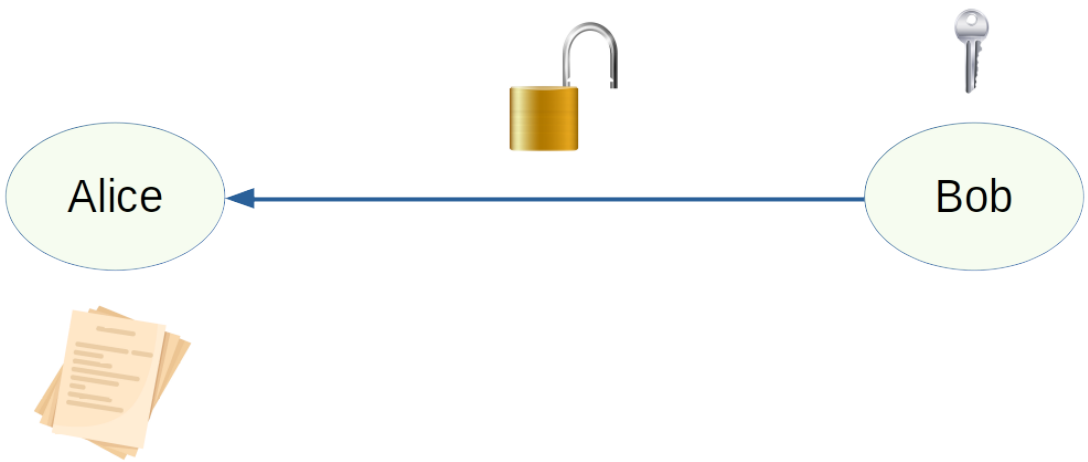
L'âge paradoxale (à partir de 1976)

L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).

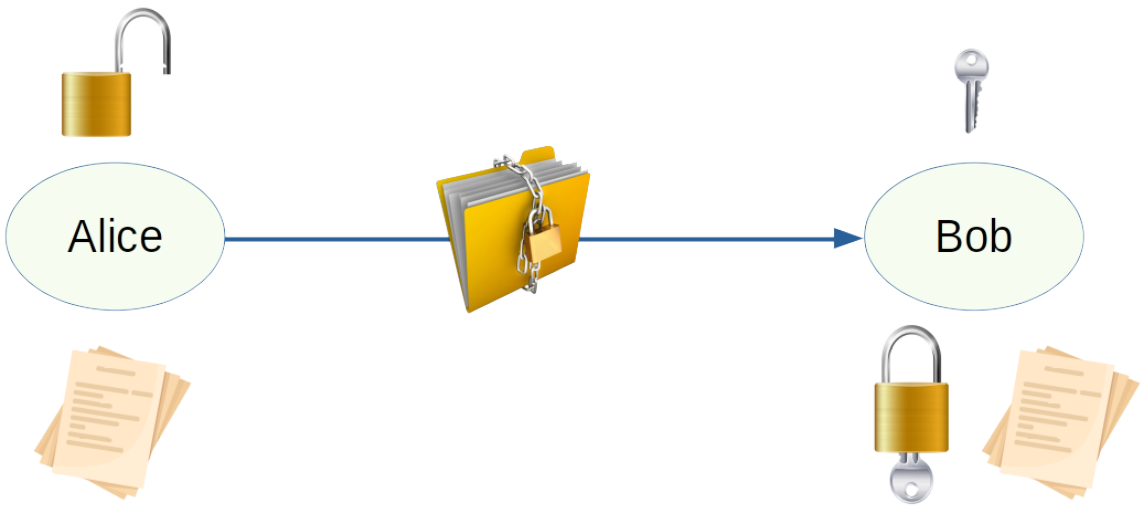
L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).



L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).



L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).

- En 1978, Rivest, Shamir et Adleman publient un algorithme de chiffrement asymétrique basé sur le problème de la factorisation.

L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).
- En 1978, Rivest, Shamir et Adleman publient un algorithme de chiffrement asymétrique basé sur le problème de la factorisation.
- En 1985, El Gamal publie un algorithme de chiffrement asymétrique basé sur le problème du logarithme discret.

L'âge paradoxale (à partir de 1976)

- En 1976, Diffie et Hellman publient l'article « *New Directions in Cryptography* » qui introduit le concept de cryptographie à clef publique (ou asymétrique).
- En 1978, Rivest, Shamir et Adleman publient un algorithme de chiffrement asymétrique basé sur le problème de la factorisation.
- En 1985, El Gamal publie un algorithme de chiffrement asymétrique basé sur le problème du logarithme discret.
- En 2000, le NIST lance un concours international pour sélectionner l'algorithme de chiffrement symétrique qui deviendra le nouveau standard du gouvernement américain. L'algorithme Rijndael est sélectionné et devient AES.

  Les chiffrements symétriques sont très rapides mais nécessitent le partage d'une clef secrète.

Cadre général : le modèle de Shannon

Un cryptosystème est la donnée de $(\mathscr{M}, \mathscr{C}, \mathscr{K}, c, d)$ où

- $\mathscr{M}$ l'ensemble des messages possibles ;
- $\mathscr{C}$ l'ensemble des chiffrés ;
- $\mathscr{K}$ l'ensemble des clefs ;
- $c : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{M}, \mathscr{C})$ une fonction servant à chiffrer ;
- $d : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{C}, \mathscr{M})$ une fonction servant à déchiffrer.

vérifiant $|\mathscr{M}| = |\mathscr{C}|$ et

Cadre général : le modèle de Shannon

Un cryptosystème est la donnée de $(\mathscr{M}, \mathscr{C}, \mathscr{K}, c, d)$ où

- $\mathscr{M}$ l'ensemble des messages possibles ;
- $\mathscr{C}$ l'ensemble des chiffrés ;
- $\mathscr{K}$ l'ensemble des clefs ;
- $c : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{M}, \mathscr{C})$ une fonction servant à chiffrer ;
- $d : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{C}, \mathscr{M})$ une fonction servant à déchiffrer.

vérifiant $|\mathscr{M}| = |\mathscr{C}|$ et

- Pour un chiffrement symétrique : $\forall k \in \mathscr{K},\ c(k) = d(k)^{-1}$.

Cadre général : le modèle de Shannon

Un cryptosystème est la donnée de $(\mathscr{M}, \mathscr{C}, \mathscr{K}, c, d)$ où

- $\mathscr{M}$ l'ensemble des messages possibles ;
- $\mathscr{C}$ l'ensemble des chiffrés ;
- $\mathscr{K}$ l'ensemble des clefs ;
- $c : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{M}, \mathscr{C})$ une fonction servant à chiffrer ;
- $d : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{C}, \mathscr{M})$ une fonction servant à déchiffrer.

vérifiant $|\mathscr{M}| = |\mathscr{C}|$ et

- Pour un chiffrement symétrique : $\forall k \in \mathscr{K},\ c(k) = d(k)^{-1}$.
- Pour un chiffrement asymétrique :
  $\forall k_{pub} \in \mathscr{K}_{pub},\ \exists k_{priv} \in \mathscr{K}_{priv},\ c(k_{pub}) = d(k_{priv})^{-1}$
  et tel que la fonction $c(k_{pub})$ soit à « sens unique » c'est-à-dire facile à calculer mais difficile à inverser sans la clef privée.

Cadre général : le modèle de Shannon

Un cryptosystème est la donnée de $(\mathscr{M}, \mathscr{C}, \mathscr{K}, c, d)$ où

- $\mathscr{M}$ l'ensemble des messages possibles ;
- $\mathscr{C}$ l'ensemble des chiffrés ;
- $\mathscr{K}$ l'ensemble des clefs ;
- $c : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{M}, \mathscr{C})$ une fonction servant à chiffrer ;
- $d : \mathscr{K} \longrightarrow \mathscr{B}(\mathscr{C}, \mathscr{M})$ une fonction servant à déchiffrer.

vérifiant $|\mathscr{M}| = |\mathscr{C}|$ et

- Pour un chiffrement symétrique : $\forall k \in \mathscr{K},\ c(k) = d(k)^{-1}$.
- Pour un chiffrement asymétrique :
  $\forall k_{pub} \in \mathscr{K}_{pub},\ \exists k_{priv} \in \mathscr{K}_{priv},\ c(k_{pub}) = d(k_{priv})^{-1}$
  et tel que la fonction $c(k_{pub})$ soit à « sens unique » c'est-à-dire facile à calculer mais difficile à inverser sans la clef privée.

- Pour AES : $|\mathscr{M}| = |\mathscr{C}| = 2^{128}$ et $|\mathscr{K}| = 2^b$ avec $b \in \{128, 192, 256\}$.
- Pour RSA : $|\mathscr{M}| = |\mathscr{C}| = 2^b$ et $|\mathscr{K}| \simeq 2^b$ avec $b \in \{1024, 2048, 4096\}$.

Le chiffrement El Gamal

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g. C'est-à-dire

$$G = \langle g \rangle = \{e, g, g^2, \cdots, g^{N-1}\}.$$

On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.

Le chiffrement El Gamal

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g. C'est-à-dire

$$G = \langle g \rangle = \{e, g, g^2, \cdots, g^{N-1}\}.$$

On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.

- Clef privée : $x \in [\![0, N-1]\!]$.
- Clef publique : $(g, h) \in G^2$.
- Ensemble des messages : $\mathscr{M} = G$; ensemble des chiffrés : $\mathscr{C} = G^2$.
- Chiffrement d'un message $m \in G$: $(m.h^t, g^t)$ avec $t \in [\![0, N-1]\!]$ choisi aléatoirement.
- Déchiffrement d'un chiffré $(c_1, c_2) \in G^2$: $c_1.c_2^{-x}$.

Le chiffrement El Gamal

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g. C'est-à-dire

$$G = \langle g \rangle = \{e, g, g^2, \cdots, g^{N-1}\}.$$

On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.

- Clef privée : $x \in [\![0, N-1]\!]$.
- Clef publique : $(g, h) \in G^2$.
- Ensemble des messages : $\mathscr{M} = G$; ensemble des chiffrés : $\mathscr{C} = G^2$.
- Chiffrement d'un message $m \in G$: $(m.h^t, g^t)$ avec $t \in [\![0, N-1]\!]$ choisi aléatoirement.
- Déchiffrement d'un chiffré $(c_1, c_2) \in G^2$: $c_1.c_2^{-x}$.

Soit $(G, \times)$ un groupe cyclique engendré par g. Pour $h \in G$ on définit

$$\log_g(h) = \min\left(\left\{x \in \mathbb{N} \mid h = g^x\right\}\right).$$

Problème du logarithme discret : Étant donné g et g^x calculer $x = \log_g(g^x)$.
Problème de Diffie-Hellman : Étant donné g, g^a et g^b calculer g^{ab}.
Objectif : Trouver un groupe G dans lequel ces deux problèmes sont difficiles

Comment choisir le groupe ?

Mauvais choix :

- $G = (\mathbb{Z}/n\mathbb{Z}, +)$:

  Soit k un générateur de G (k premier avec n) et $x \in [\![0, n-1]\!]$. On pose $h = xk$.
  Comme $k \wedge n = 1$, on obtient par l'algorithme d'Euclide étendu $(u, v) \in G^2$ tel que $ku + nv = 1$ donc $ku = 1$ dans G.
  Finalement $u = k^{-1}$ puis $x = hu$.

Comment choisir le groupe ?

Mauvais choix :

- $G = (\mathbb{Z}/n\mathbb{Z}, +)$:

  Soit k un générateur de G (k premier avec n) et $x \in [\![0, n-1]\!]$. On pose $h = xk$.
  Comme $k \wedge n = 1$, on obtient par l'algorithme d'Euclide étendu $(u, v) \in G^2$ tel que $ku + nv = 1$ donc $ku = 1$ dans G.
  Finalement $u = k^{-1}$ puis $x = hu$.

- Dans $(\mathbb{R}_+^*, \times)$ le sous-groupe monogène $G = \langle a \rangle = \left\{a^k,\ k \in \mathbb{Z}\right\}$
  car $\log_a(x)$ se calcule facilement.

Comment choisir le groupe ?

Mauvais choix :

- $G = (\mathbb{Z}/n\mathbb{Z}, +)$:

  Soit k un générateur de G (k premier avec n) et $x \in [\![0, n-1]\!]$. On pose $h = xk$.
  Comme $k \wedge n = 1$, on obtient par l'algorithme d'Euclide étendu $(u, v) \in G^2$ tel que $ku + nv = 1$ donc $ku = 1$ dans G.
  Finalement $u = k^{-1}$ puis $x = hu$.

- Dans $(\mathbb{R}_+^*, \times)$ le sous-groupe monogène $G = \langle a \rangle = \left\{a^k,\ k \in \mathbb{Z}\right\}$
  car $\log_a(x)$ se calcule facilement.

Une bonne idée :

Soit K un corps (commutatif) fini. Alors $K^\times$ est cyclique.

Comment choisir le groupe ?

Mauvais choix :

- $G = (\mathbb{Z}/n\mathbb{Z}, +)$:

  Soit k un générateur de G (k premier avec n) et $x \in [\![0, n-1]\!]$. On pose $h = xk$.
  Comme $k \wedge n = 1$, on obtient par l'algorithme d'Euclide étendu $(u, v) \in G^2$ tel que $ku + nv = 1$ donc $ku = 1$ dans G.
  Finalement $u = k^{-1}$ puis $x = hu$.

- Dans $(\mathbb{R}_+^*, \times)$ le sous-groupe monogène $G = \langle a \rangle = \left\{a^k,\ k \in \mathbb{Z}\right\}$
  car $\log_a(x)$ se calcule facilement.

Une bonne idée :

Soit K un corps (commutatif) fini. Alors $K^\times$ est cyclique.

Soit p un nombre premier. Considérons $G = \left((\mathbb{Z}/p\mathbb{Z})^\times, \times\right)$.

Comment choisir p ?

Réduction de Pohlig-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g et $h = g^x$ avec $x \in [\![0, N-1]\!]$.

Réduction de Pohlig-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g et $h = g^x$ avec $x \in [\![0, N-1]\!]$.

Étape 1 : Si $N = p^\alpha$ avec p premier (assez petit) et $\alpha \in \mathbb{N}^*$
On décompose x en base p :

$$x = x_0 + x_1 p + x_2 p^2 + \cdots + x_{\alpha-1} p^{\alpha-1}.$$

Alors, puisque $xp^{\alpha-1} \equiv x_0 p^{\alpha-1} \ [p^\alpha]$, on a

$$h^{p^{\alpha-1}} = g^{xp^{\alpha-1}} = g^{x_0 p^{\alpha-1}} = \left(g^{p^{\alpha-1}}\right)^{x_0}.$$

Trouver x_0 revient à déterminer le logarithme discret de $h^{p^{\alpha-1}}$ dans le sous-groupe cyclique $H = \langle g^{p^{\alpha-1}} \rangle$ d'ordre p.

Une fois $x_0, \cdots, x_{k-1}$ déterminés on trouve x_k en remarquant que

$$y = h^{p^{k-1}} g^{-\left(x_0 p^{k-1} + \cdots + x_{k-1} p^{\alpha-2}\right)} = g^{x_k p^{\alpha-1}} = \left(g^{p^{\alpha-1}}\right)^{x_k}.$$

On détermine x_k en calculant le logarithme discret de y dans le sous-groupe cyclique H.

Réduction de Pohlig-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g et $h = g^x$ avec $x \in [\![0, N-1]\!]$.

Étape 2 : Si la décomposition en facteurs premier de N est $N = \prod_{i=1}^{n} p_i^{\alpha_i}$

On considère $g_i = g^{N/p_i^{\alpha_i}}$ et $G_i = \langle g_i \rangle$ qui est sous-groupe cyclique d'ordre $p_i^{\alpha_i}$.

D'après l'étape 1 on calcule x_i le logarithme de $h_i = h^{N/p_i^{\alpha_i}} = g_i^x$ dans G_i. Dès lors

$$g_i^{x_i} = h_i = g_i^x \quad \text{donc} \quad x_i \equiv x \ [p_i^{\alpha_i}].$$

Finalement puisque pour tout $i \in [\![1, n]\!]$, $x \equiv x_i \ [p_i^{\alpha_i}]$, on peut retrouver x par le théorème des restes chinois.

Conclusion : Calculer le logarithme discret dans G revient à calculer le logarithme discret dans des groupes cycliques d'ordre p_i.
Si tous les p_i sont assez petits alors on peut facilement résoudre le problème du logarithme discret.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.
- On choisit donc p un nombre premier de Sophie Germain, c'est-à-dire tel que $p = 2q+1$ avec q premier.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.
- On choisit donc p un nombre premier de Sophie Germain, c'est-à-dire tel que $p = 2q+1$ avec q premier.
- Dans ce cas G possède 4 sous-groupes : les sous-groupes triviaux $\{1\}$ et G, le sous-groupe $\{-1, 1\}$ et H un sous-groupe d'ordre q.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.
- On choisit donc p un nombre premier de Sophie Germain, c'est-à-dire tel que $p = 2q+1$ avec q premier.
- Dans ce cas G possède 4 sous-groupes : les sous-groupes triviaux $\{1\}$ et G, le sous-groupe $\{-1, 1\}$ et H un sous-groupe d'ordre q.
- Si on prend g un générateur de G et $x \in [\![0, p-2]\!]$ alors x est pair ssi g^x est un carré dans G. On peut déterminer si g^x est un carré en calculant le symbole de Legendre $\left(\frac{g^x}{p}\right)$.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.
- On choisit donc p un nombre premier de Sophie Germain, c'est-à-dire tel que $p = 2q+1$ avec q premier.
- Dans ce cas G possède 4 sous-groupes : les sous-groupes triviaux $\{1\}$ et G, le sous-groupe $\{-1, 1\}$ et H un sous-groupe d'ordre q.
- Si on prend g un générateur de G et $x \in [\![0, p-2]\!]$ alors x est pair ssi g^x est un carré dans G. On peut déterminer si g^x est un carré en calculant le symbole de Legendre $\left(\frac{g^x}{p}\right)$.
- On prendra alors le groupe cyclique H d'ordre q qui est premier.

Un bon choix de groupe pour l'algorithme El Gamal

Soit p un nombre premier. On considère $G = \left((\mathbb{Z}/p\mathbb{Z})^{\times}, \times\right)$.

- Pour résister à l'attaque de Pohlig-Hellman le cardinal de G (c'est-à-dire $p-1$) ne doit pas avoir (que) des petits facteurs premiers.
- On choisit donc p un nombre premier de Sophie Germain, c'est-à-dire tel que $p = 2q+1$ avec q premier.
- Dans ce cas G possède 4 sous-groupes : les sous-groupes triviaux $\{1\}$ et G, le sous-groupe $\{-1, 1\}$ et H un sous-groupe d'ordre q.
- Si on prend g un générateur de G et $x \in [\![0, p-2]\!]$ alors x est pair ssi g^x est un carré dans G. On peut déterminer si g^x est un carré en calculant le symbole de Legendre $\left(\frac{g^x}{p}\right)$.
- On prendra alors le groupe cyclique H d'ordre q qui est premier.
- Dans la pratique on prend

$$p = 2^n - 2^{n-64} - 1 + 2^{64}\left(\left\lfloor 2^{n-130}\pi \right\rfloor + k\right)$$

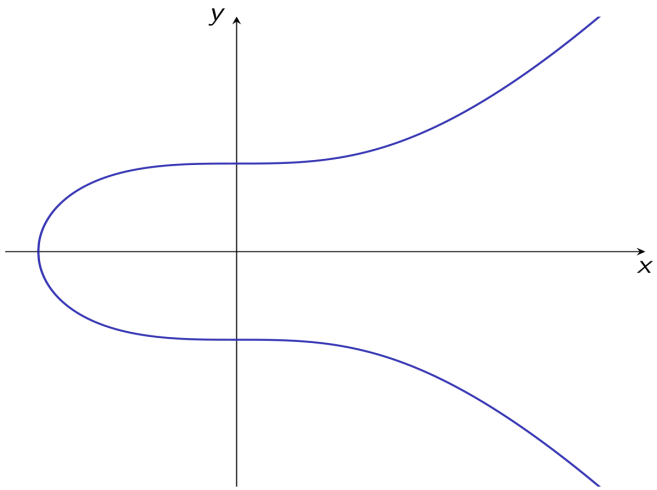
avec $n \in \{2048, 3072, 4096, 6144, 8192\}$ et k le plus petit entier positif tel que p soit nombre premier de Sophie Germain.

Courbes elliptiques

Soit p un nombre premier et $(a, b) \in \left(\mathbb{Z}/p\mathbb{Z}\right)^2$. L'ensemble

$$\mathcal{C} = \left\{(x, y) \in \left(\mathbb{Z}/p\mathbb{Z}\right)^2 \mid y^2 = x^3 + ax + b\right\} \cup \{O\}$$

peut être muni d'une loi de groupe.



En choisissant bien p, a et b ce groupe est cyclique et le problème du logarithme discret est difficile. L'algorithme El Gamal peut-être utilisé sur ces courbes elliptiques.

Par exemple pour la courbe Secp256k1 utilisée pour signer les transactions Bitcoin :

- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
- $a = 0$ et $b = 7$

Fonctionnement général de la signature

L'objectif de la signature numérique d'un document est de garantir **l'authenticité** et **l'intégrité** du document grâce à un procédé cryptographique asymétrique.

Fonctionnement général de la signature

L'objectif de la signature numérique d'un document est de garantir **l'authenticité** et **l'intégrité** du document grâce à un procédé cryptographique asymétrique.

Fonction de hachage : c'est une fonction H à sens unique qui prend en entrée des données de longueur quelconque et rend une valeur de taille fixe. On souhaite de plus qu'il n'y ait pas/peu de x et y « voisins » tels que $H(x) = H(y)$.

Fonctionnement général de la signature

L'objectif de la signature numérique d'un document est de garantir **l'authenticité** et **l'intégrité** du document grâce à un procédé cryptographique asymétrique.

Fonction de hachage : c'est une fonction H à sens unique qui prend en entrée des données de longueur quelconque et rend une valeur de taille fixe. On souhaite de plus qu'il n'y ait pas/peu de x et y « voisins » tels que $H(x) = H(y)$.

On considère un cryptosystème asymétrique tel que $\mathscr{M} = \mathscr{C}$. Soit (k_{pub}, k_{priv}) une paire de clefs et H une fonction de hachage à valeurs dans $\mathscr{M}$. Prenons m un message à signer (potentiellement $m \notin \mathscr{M}$).

- Signature : $s = d(k_{priv})(H(m))$
- Transmission : (m, s)
- Vérification : $c(k_{pub})(s) == H(m)$

Le (EC)DSA : (Elliptic Curve) Digital Signature Algorithm

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.
On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.
Finalement on considère une fonction de hachage H à valeurs dans $[\![0, p-1]\!]$ et une bijection $\phi : G \to \mathbb{Z}/N\mathbb{Z}$ qui se calcule efficacement.

- Clef privée : x.
- Clef publique : (g, h, H, ϕ).
- Signature d'un message m : $(a, b) \in G \times \mathbb{Z}/N\mathbb{Z}$ défini par :
  1. on choisit aléatoirement $k \in \mathbb{Z}/N\mathbb{Z}$ inversible
  2. on calcule $a = g^k$
  3. on calcule $b = (H(m) - x\phi(a))k^{-1}$ dans $\mathbb{Z}/N\mathbb{Z}$
- Vérification de la signature (a, b) du message m : on vérifie que

$$g^{H(m)} = h^{\phi(a)} a^b.$$

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit
$\alpha \in]\!]0, N[\![$ secret

Alice

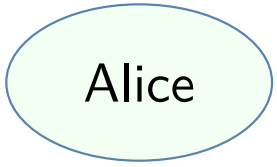
Bob

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit
$\alpha \in]\!]0, N[\!]$ secret

Bob choisit
$\beta \in]\!]0, N[\!]$ secret



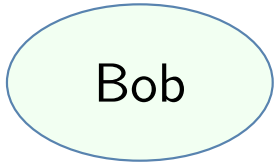


Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit
$\alpha \in]\!]0, N[\!]$ secret

Bob choisit
$\beta \in]\!]0, N[\!]$ secret

Alice

Bob

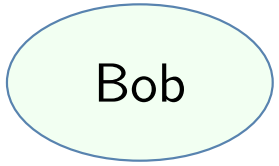
Alice calcule $a = g^{\alpha}$

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit
$\alpha \in]\!]0, N[\!]$ secret

Bob choisit
$\beta \in]\!]0, N[\!]$ secret



Alice calcule $a = g^{\alpha}$

Bob calcule $b = g^{\beta}$

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

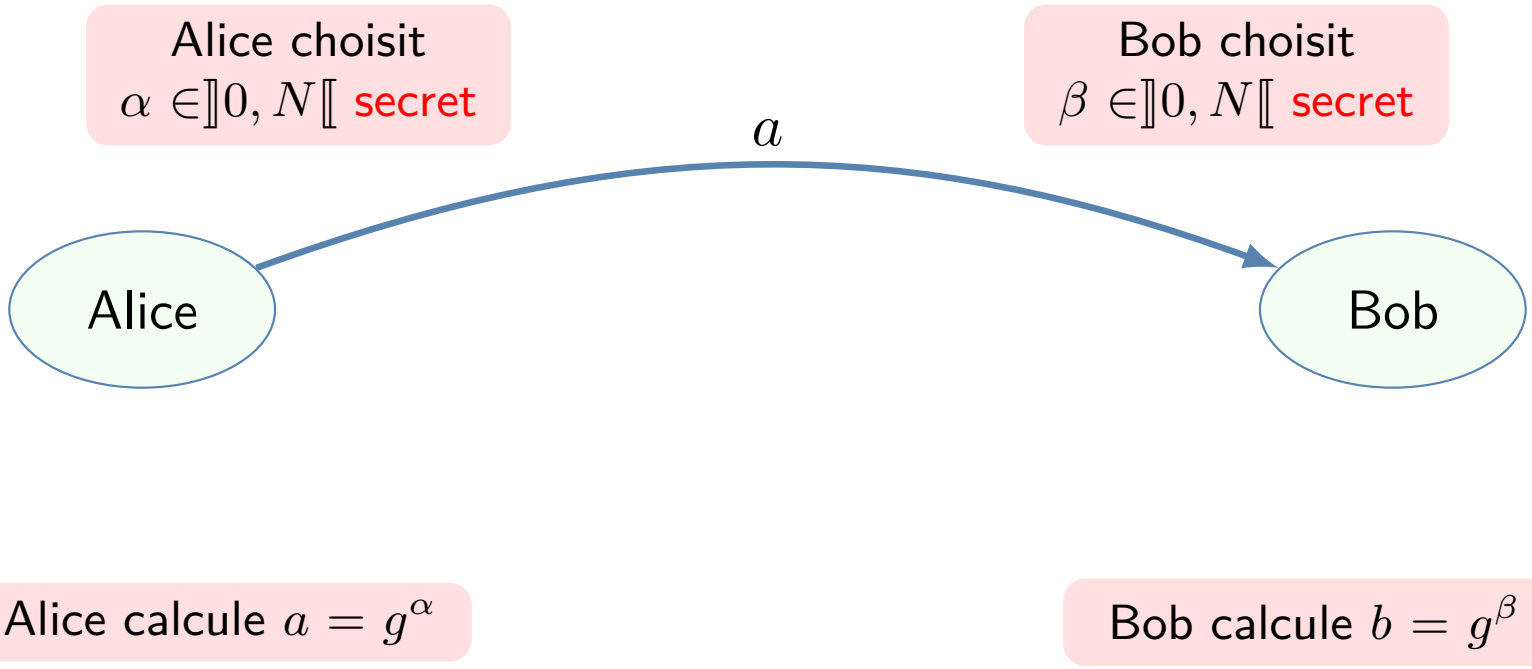


Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

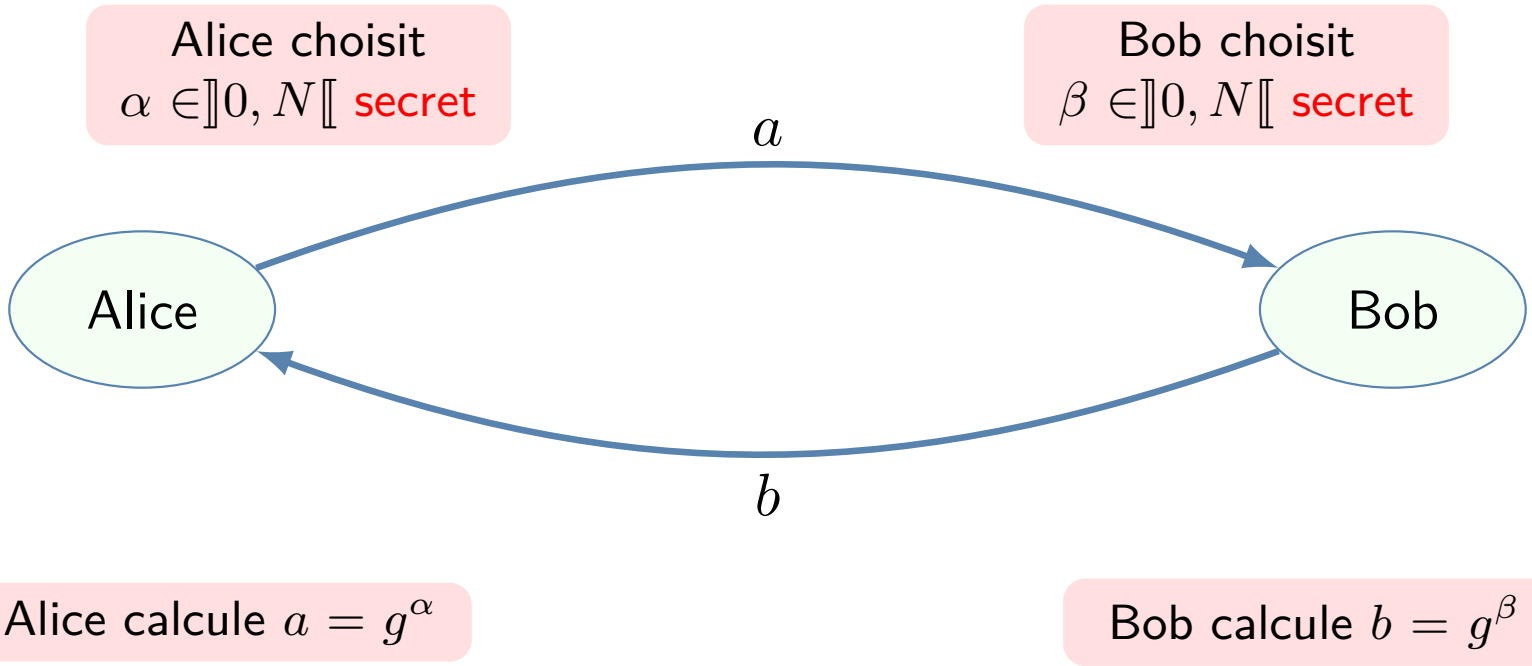


Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

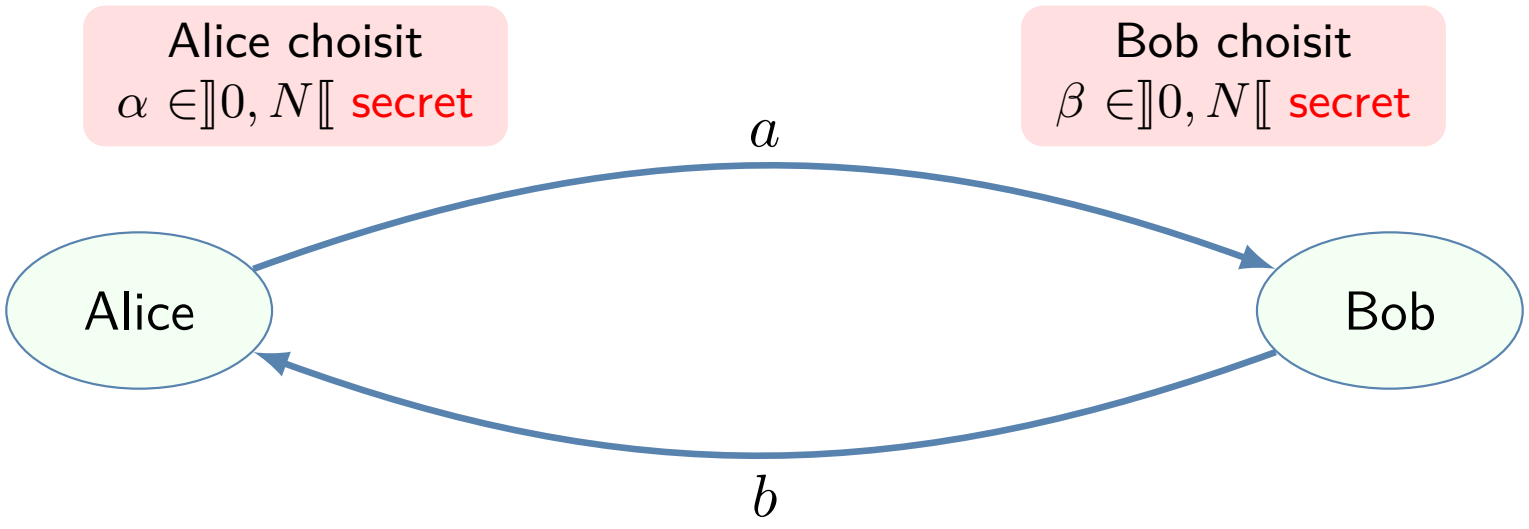
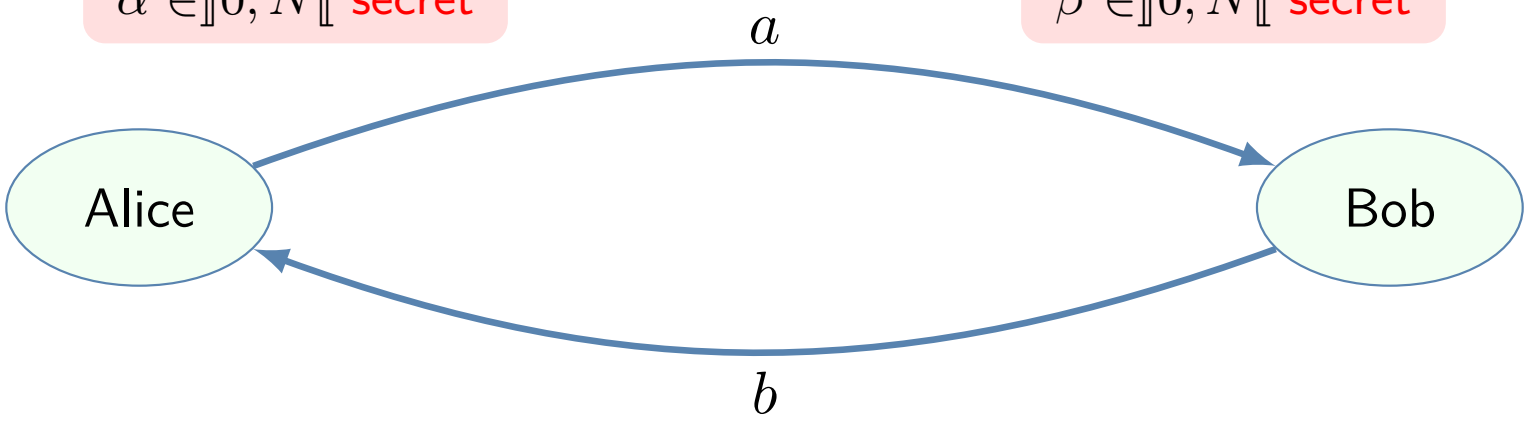


Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit $\alpha \in]\!]0, N[\![$ secret

Bob choisit $\beta \in]\!]0, N[\![$ secret



Alice calcule $a = g^{\alpha}$

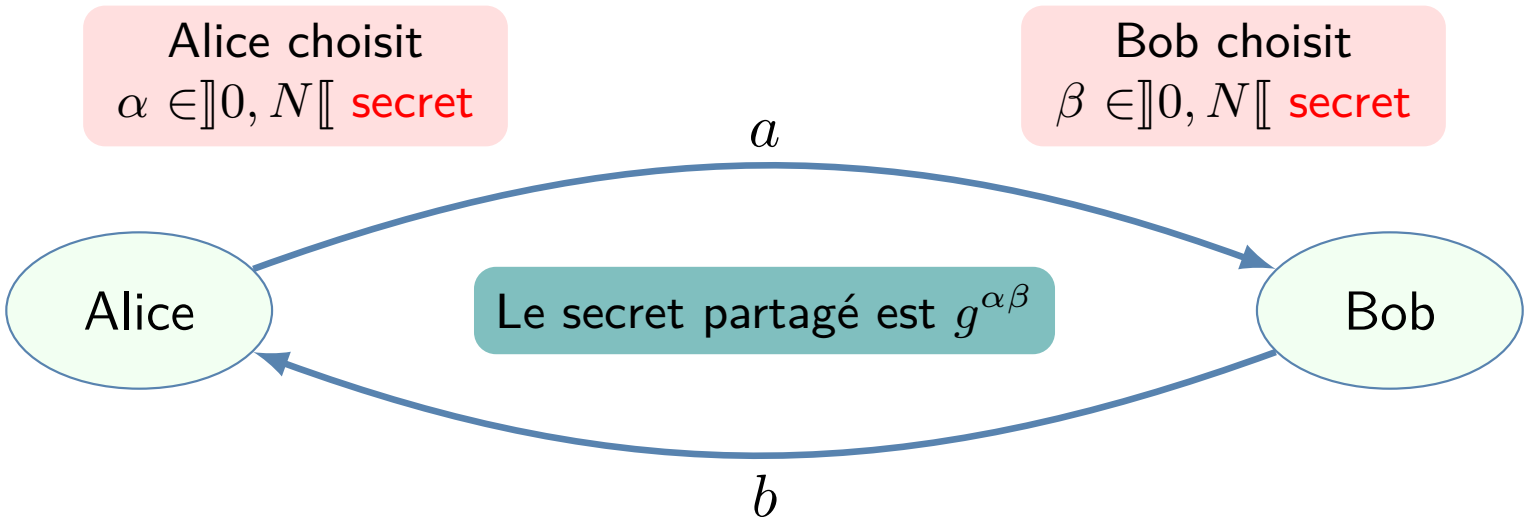
Bob calcule $b = g^{\beta}$

Alice calcule $b^{\alpha} = \left(g^{\beta}\right)^{\alpha}$

Bob calcule $a^{\beta} = (g^{\alpha})^{\beta}$

Schéma du protocole d'échange de clefs Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.



Alice calcule $a = g^{\alpha}$

Bob calcule $b = g^{\beta}$

Alice calcule $b^{\alpha} = \left(g^{\beta}\right)^{\alpha}$ $= g^{\alpha\beta} =$ Bob calcule $a^{\beta} = (g^{\alpha})^{\beta}$

Attaque de l'homme du milieu

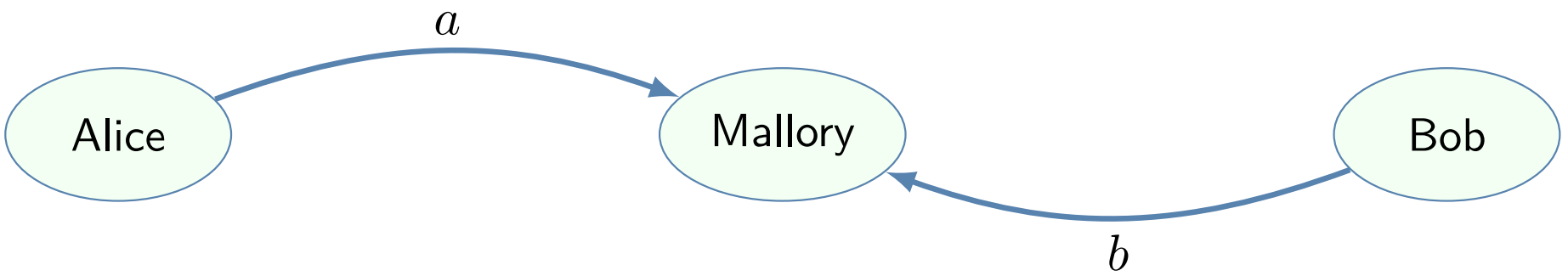
Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Attaque de l'homme du milieu

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit
$\alpha \in]\!]0, N[\![$ secret
et calcule $a = g^{\alpha}$

Bob choisit
$\beta \in]\!]0, N[\![$ secret
et calcule $b = g^{\beta}$



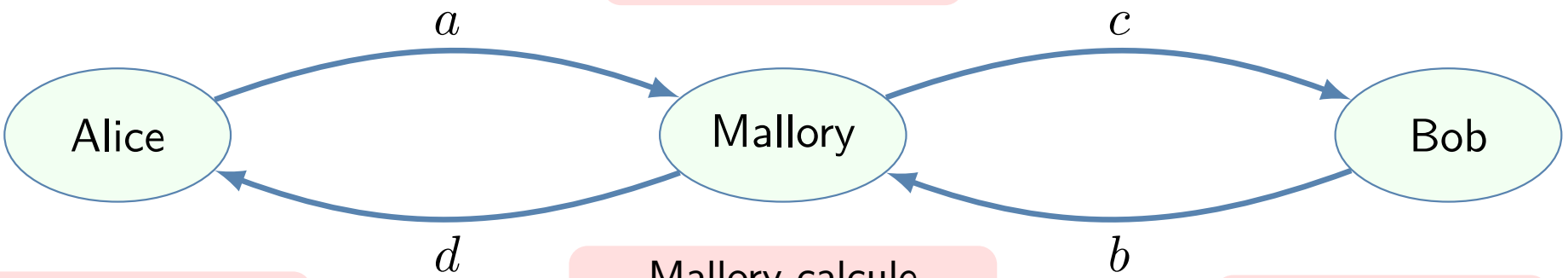
Attaque de l'homme du milieu

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

Alice choisit $\alpha \in]\!]0, N[\![$ secret et calcule $a = g^\alpha$

Mallory choisit $(\gamma, \delta) \in]\!]0, N[\![^2$ et calcule $c = g^\gamma$ et $d = g^\delta$

Bob choisit $\beta \in]\!]0, N[\![$ secret et calcule $b = g^\beta$



Alice calcule $d^\alpha = \left(g^\delta\right)^\alpha$

Mallory calcule $a^\delta = (g^\alpha)^\delta$ et $b^\gamma = \left(g^\beta\right)^\gamma$

Bob calcule $c^\beta = (g^\gamma)^\beta$

Attaque de l'homme du milieu

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g dans lequel le problème de Diffie-Hellman est difficile.

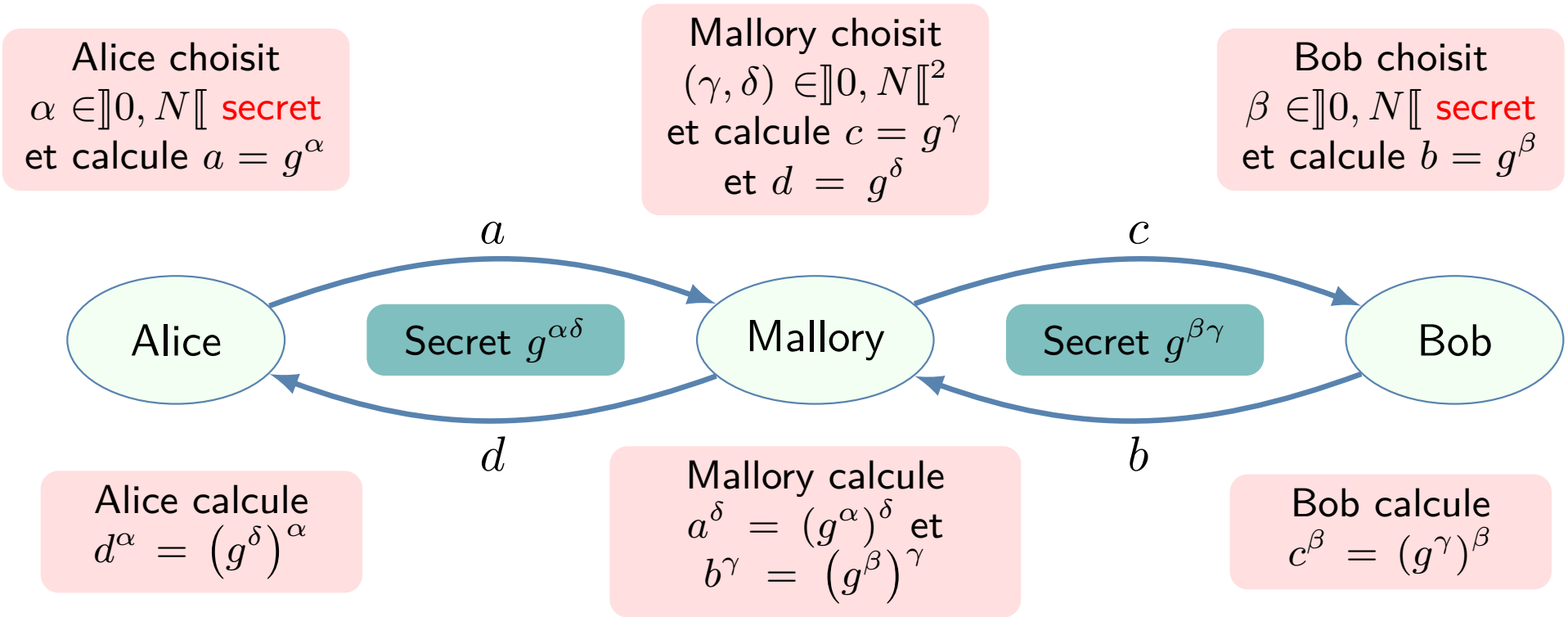
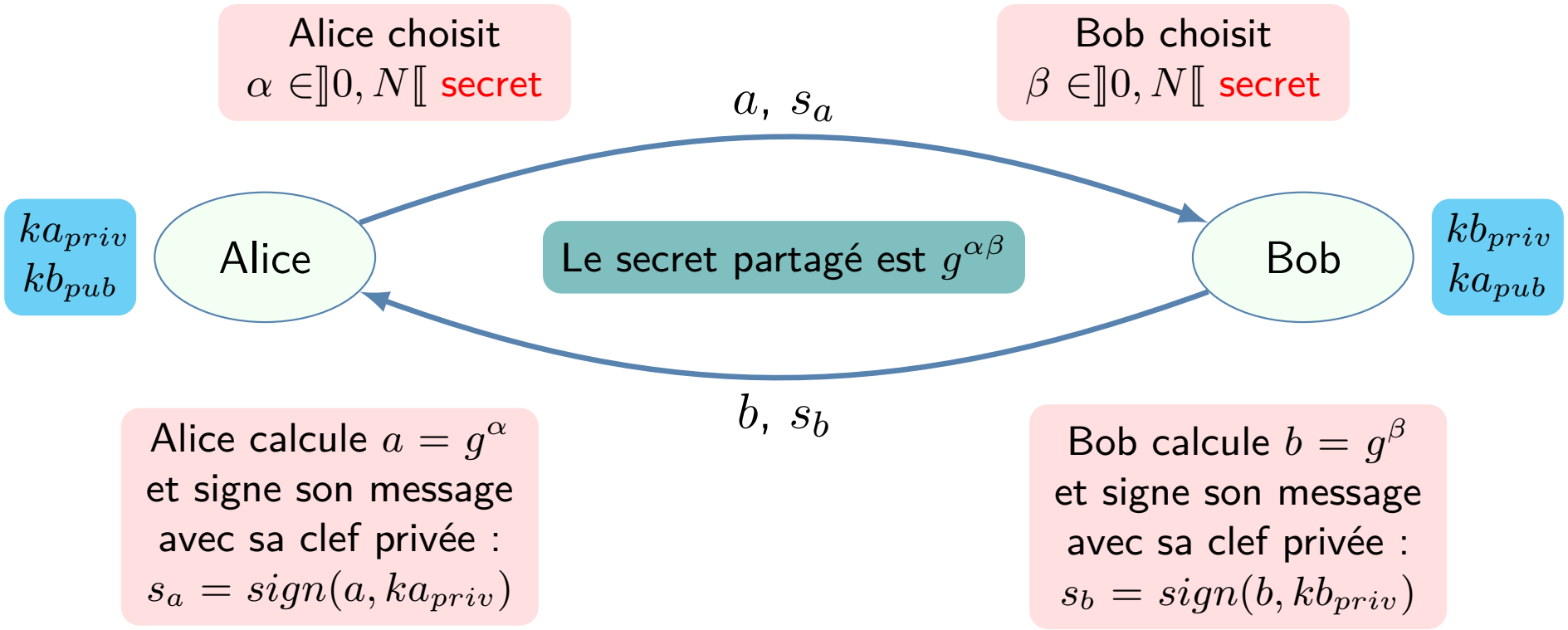


Schéma final du protocole d'échange de clefs Diffie-Hellman

Alice et Bob génèrent chacun un couple de clefs publiques/privées (ka_{pub}, ka_{priv}) et (kb_{pub}, kb_{priv}) pour signer leurs messages.

Schéma final du protocole d'échange de clefs Diffie-Hellman

Alice et Bob génèrent chacun un couple de clefs publiques/privées (ka_{pub}, ka_{priv}) et (kb_{pub}, kb_{priv}) pour signer leurs messages.



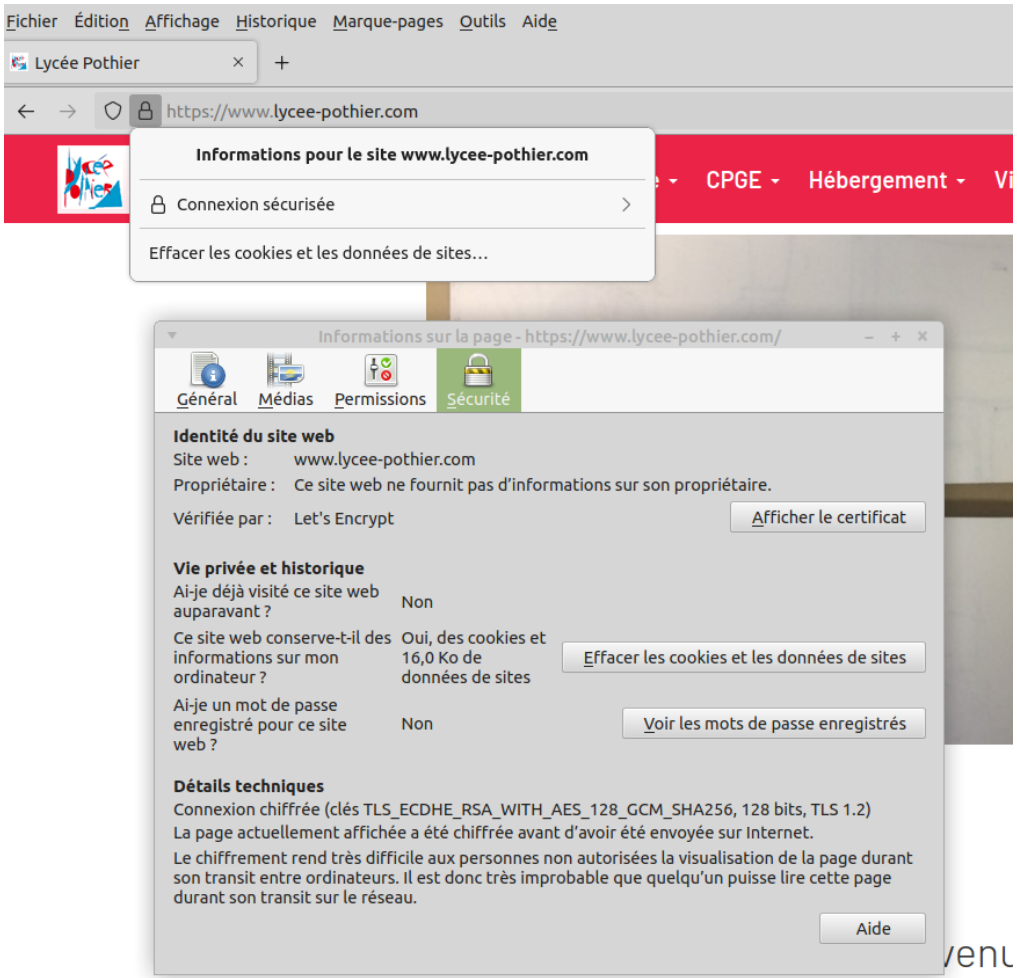
Description du protocole TLS (Transport Layer Security)

TLS est un protocole de sécurisation d'échanges sur un réseau informatique qui s'ajoute par dessus des protocoles historiques non sécurisés (par exemple HTTP/DNS pour le web ou IMAP/SMTP pour les mails)

Fonctionnement d'une connexion sécurisée par TLS entre un client (navigateur) et un serveur (web) :

1. Le client contacte le serveur et demande la mise en place de connexion sécurisée par TLS
2. Le serveur envoie au client son « certificat » qui contient sa clef publique et une signature numérique émise par une autorité de certification
3. Le client vérifie le certificat du serveur
4. Le client et le serveur génèrent et s'échangent une clef commune via l'échange de clef de Diffie-Hellman
5. Le client et le serveur commencent à s'échanger des données en les chiffrant avec un algorithme de chiffrement symétrique et la précédente clef

Exemple d'utilisation du protocole TLS



Quelques statistiques :

- En 2024 plus de 85% des sites utilisent https par défaut
- En 2021 les certificats sont signés à
  - 75% par RSA
  - 25% par El Gamal sur courbes elliptiques (ECDSA)

$$\underbrace{\text{ECDHE}}_{\text{échange clef}} - \underbrace{\text{RSA}}_{\text{auth}} - \underbrace{\text{AES128GCM}}_{\text{chiffrement symétrique}} - \underbrace{\text{SHA256}}_{\text{hachage}}$$

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final
5. Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final
5. Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi
6. Chaque votant doit pouvoir vérifier que son vote est dans l'urne et n'est pas modifié

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final
5. Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi
6. Chaque votant doit pouvoir vérifier que son vote est dans l'urne et n'est pas modifié
7. Un votant ne doit pas pouvoir prouver son vote à un tiers

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final
5. Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi
6. Chaque votant doit pouvoir vérifier que son vote est dans l'urne et n'est pas modifié
7. Un votant ne doit pas pouvoir prouver son vote à un tiers
8. Tout cela même si les autorités sont malhonnêtes

Propriétés d'un vote idéal

1. L'urne doit être publique et le vote vérifiable
2. Seuls les inscrits votent
3. Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. Tout le monde doit pouvoir vérifier le comptage final
5. Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi
6. Chaque votant doit pouvoir vérifier que son vote est dans l'urne et n'est pas modifié
7. Un votant ne doit pas pouvoir prouver son vote à un tiers
8. Tout cela même si les autorités sont malhonnêtes

Recommandation de la CNIL :

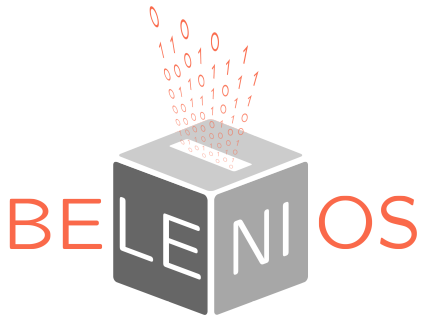
- Niveau 1 : le minimum à avoir 2 4 (avec vérification par les autorités) 5
- Niveau 2 : transparence de l'urne 1 3
- Niveau 3 : vérification de l'intégrité du vote par des outils tiers 8

Protocole de vote Belenios

Protocole et logiciel développés au LORIA (Laboratoire Lorrain de Recherche en Informatique et ses Applications) à partir de 2012 par

- **Stéphane Glondu** : ingénieur de recherche, développeur principal
- **Véronique Cortier** : directrice de recherche CNRS, spécialiste en analyse formelle de la sécurité des protocoles
- **Pierrick Gaudry** : chargé de recherche CNRS, spécialiste de la théorie algorithmique des nombres et de la cryptographie à clé publique

Diffusé sous licence libre (AGPL) et utilisé dans plus de 1400 élections en 2020 (au sein d'universités, d'associations, etc.)



`https://www.belenios.org`

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
   - David arrive et certains électeurs ont déjà voté. Leurs bulletins sont visibles dans l'urne.

Urne (page web publique)	
Alice	$\{V_A\}_{pub_E}$
Bob	$\{V_B\}_{pub_E}$
Carol	$\{V_C\}_{pub_E}$

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
   - David arrive et certains électeurs ont déjà voté. Leurs bulletins sont visibles dans l'urne.
   - David choisit son vote V_D qui vaut 0 ou 1

Urne (page web publique)	
Alice	$\{V_A\}_{pub_E}$
Bob	$\{V_B\}_{pub_E}$
Carol	$\{V_C\}_{pub_E}$

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
   - David arrive et certains électeurs ont déjà voté. Leurs bulletins sont visibles dans l'urne.
   - David choisit son vote V_D qui vaut 0 ou 1
   - il chiffre son vote avec la clef publique pub_E : $\{V_D\}_{pub_E}$

Urne (page web publique)	
Alice	$\{V_A\}_{pub_E}$
Bob	$\{V_B\}_{pub_E}$
Carol	$\{V_C\}_{pub_E}$

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
   - David arrive et certains électeurs ont déjà voté. Leurs bulletins sont visibles dans l'urne.
   - David choisit son vote V_D qui vaut 0 ou 1
   - il chiffre son vote avec la clef publique pub_E : $\{V_D\}_{pub_E}$

Urne (page web publique)	
Alice	$\{V_A\}_{pub_E}$
Bob	$\{V_B\}_{pub_E}$
Carol	$\{V_C\}_{pub_E}$
David	$\{V_D\}_{pub_E}$

Fonctionnement simplifié de Belenios

Exemple d'un vote type référendum avec deux choix : 0 ou 1

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
   - David arrive et certains électeurs ont déjà voté. Leurs bulletins sont visibles dans l'urne.
   - David choisit son vote V_D qui vaut 0 ou 1
   - il chiffre son vote avec la clef publique pub_E : $\{V_D\}_{pub_E}$

Urne (page web publique)	
Alice	$\{V_A\}_{pub_E}$
Bob	$\{V_B\}_{pub_E}$
Carol	$\{V_C\}_{pub_E}$
David	$\{V_D\}_{pub_E}$

3. **Dépouillement** : le chiffrement doit avoir la propriété d'être homomorphe :

$$\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$$

Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

Problèmes de ce modèle simplifié

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffré les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
3. **Dépouillement** : $\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$.
   Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

Problèmes de ce modèle simplifié

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffré les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
3. **Dépouillement** : $\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$.
   Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

- Chacune des autorités peut casser le secret du vote
  Solution : chiffrement à seuil

Problèmes de ce modèle simplifié

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffré les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
3. **Dépouillement** : $\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$.
   Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

- Chacune des autorités peut casser le secret du vote
  Solution : chiffrement à seuil
- Un votant peut voter plusieurs fois.
  Solution : signer chaque vote

Problèmes de ce modèle simplifié

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffré les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
3. **Dépouillement** : $\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$.
   Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

- Chacune des autorités peut casser le secret du vote
  Solution : chiffrement à seuil
- Un votant peut voter plusieurs fois.
  Solution : signer chaque vote
- Un votant peut envoyer $\{100\}_{pub_E}$ au lieu de $\{0\}_{pub_E}$ ou $\{1\}_{pub_E}$.

Problèmes de ce modèle simplifié

1. **Initialisation du vote** : génération d'un couple de clef $(pub_E, priv_E)$ pour chiffré les votes. La clef pub_E est distribuée aux votants et la clef $priv_E$ est partagée aux autorités de contrôle (commission électorale, candidats, ...)
2. **Vote** : Chaque votant chiffre son vote avec la clef publique pub_E puis son vote chiffré est publié sur une page publique
3. **Dépouillement** : $\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}$.
   Seul le résultat final doit être déchiffré par les autorités de contrôle avec la clef privée $priv_E$.

- Chacune des autorités peut casser le secret du vote
  Solution : chiffrement à seuil
- Un votant peut voter plusieurs fois.
  Solution : signer chaque vote
- Un votant peut envoyer $\{100\}_{pub_E}$ au lieu de $\{0\}_{pub_E}$ ou $\{1\}_{pub_E}$.
- Les votants ne peuvent vérifier le résultat du vote.
  Solution : preuve à divulgation nulle de connaissance

Le chiffrement El Gamal est homomorphe

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g.
On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.

- Clef privée du vote : $priv_E = x \in [\![0, N-1]\!]$.
- Clef publique du vote : $pub_E = (g, h) \in G^2$.
- Chiffrement d'un vote $v \in \{0, 1\}$: $(g^v.h^t \ , \ g^t)$ avec $t \in [\![0, N-1]\!]$ choisi aléatoirement.

Le chiffrement El Gamal est homomorphe

Soit $(G, \times)$ un groupe cyclique d'ordre N engendré par g.
On choisit aléatoirement $x \in [\![0, N-1]\!]$ et on calcule $h = g^x$.

- Clef privée du vote : $priv_E = x \in [\![0, N-1]\!]$.
- Clef publique du vote : $pub_E = (g, h) \in G^2$.
- Chiffrement d'un vote $v \in \{0, 1\}$: $(g^v.h^t \ , \ g^t)$ avec $t \in [\![0, N-1]\!]$ choisi aléatoirement.

On considère n votes $(v_1, \cdots, v_n) \in \{0, 1\}^n$ chiffrés puis publiés :

$$\left(g^{v_1}.h^{t_1} \ , \ g^{t_1}\right), \quad \cdots \quad , \left(g^{v_n}.h^{t_n} \ , \ g^{t_n}\right).$$

Le dépouillement s'obtient en faisant

$$\left(\prod_{i=1}^{n} g^{v_i}.h^{t_i} \ , \ \prod_{i=1}^{n} g^{t_i}\right) = \left(g^v.h^t \ , \ g^t\right) = (c_1, c_2) \text{ avec } v = \sum_{i=1}^{n} v_i \text{ et } t = \sum_{i=1}^{n} t_i$$

que les autorités peuvent déchiffrer en faisant $c_1.c_2^{-x} = g^v$. On peut alors récupérer le résultat du vote v qui est de taille raisonnable.

Chiffrement à seuil : partage de clef secrète de Shamir (1979)

On chiffre un message avec une clef privée $c \in \mathbb{F}_p$ avec p premier.
Le but est de créer n clefs privées dont au minimum k clefs sont nécessaires pour déchiffrer le message.

Chiffrement à seuil : partage de clef secrète de Shamir (1979)

On chiffre un message avec une clef privée $c \in \mathbb{F}_p$ avec p premier.
Le but est de créer n clefs privées dont au minimum k clefs sont nécessaires pour déchiffrer le message.

1. On génère $k-1$ coefficients dans $\mathbb{F}_p$ notés $a_1, \cdots, a_{k-1}$ et on considère le polynôme

$$P = c + \sum_{i=1}^{k-1} a_i X^i.$$

Le polynôme doit rester secret et ne pas être partagé.

Chiffrement à seuil : partage de clef secrète de Shamir (1979)

On chiffre un message avec une clef privée $c \in \mathbb{F}_p$ avec p premier.
Le but est de créer n clefs privées dont au minimum k clefs sont nécessaires pour déchiffrer le message.

1. On génère $k-1$ coefficients dans $\mathbb{F}_p$ notés $a_1, \cdots, a_{k-1}$ et on considère le polynôme

$$P = c + \sum_{i=1}^{k-1} a_i X^i.$$

   Le polynôme doit rester secret et ne pas être partagé.

2. On génère n éléments deux à deux distincts de $\mathbb{F}_p$ notés $x_1, \cdots, x_n$ et on calcule $y_1 = P(x_1), \cdots, y_n = P(x_n)$.

Chiffrement à seuil : partage de clef secrète de Shamir (1979)

On chiffre un message avec une clef privée $c \in \mathbb{F}_p$ avec p premier.
Le but est de créer n clefs privées dont au minimum k clefs sont nécessaires pour déchiffrer le message.

1. On génère $k-1$ coefficients dans $\mathbb{F}_p$ notés $a_1, \cdots, a_{k-1}$ et on considère le polynôme

$$P = c + \sum_{i=1}^{k-1} a_i X^i.$$

   Le polynôme doit rester secret et ne pas être partagé.
2. On génère n éléments deux à deux distincts de $\mathbb{F}_p$ notés $x_1, \cdots, x_n$ et on calcule $y_1 = P(x_1), \cdots, y_n = P(x_n)$.
3. On distribue à chacune des n autorités un couple (x_i, y_i).

Chiffrement à seuil : partage de clef secrète de Shamir (1979)

On chiffre un message avec une clef privée $c \in \mathbb{F}_p$ avec p premier.
Le but est de créer n clefs privées dont au minimum k clefs sont nécessaires pour déchiffrer le message.

1. On génère $k-1$ coefficients dans $\mathbb{F}_p$ notés $a_1, \cdots, a_{k-1}$ et on considère le polynôme

$$P = c + \sum_{i=1}^{k-1} a_i X^i.$$

   Le polynôme doit rester secret et ne pas être partagé.
2. On génère n éléments deux à deux distincts de $\mathbb{F}_p$ notés $x_1, \cdots, x_n$ et on calcule $y_1 = P(x_1), \cdots, y_n = P(x_n)$.
3. On distribue à chacune des n autorités un couple (x_i, y_i).

Pour trouver c il faut reconstituer le polynôme P en l'interpolant à l'aide d'au minimum k couples (x_i, y_i) car $\deg(P) = k - 1$.

On peut faire en sorte que c reste secret et que l'algorithme déchiffre automatiquement le message.

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

1					7		9	
	3			2				8
		9	6			5		
		5	3			9		
	1			8				2
6					4			
3							1	
	4							7
		7				3		

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

1					7		9	
	3			2				8
		9	6			5		
		5	3			9		
	1			8				2
6					4			
3							1	
	4							7
		7				3		

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

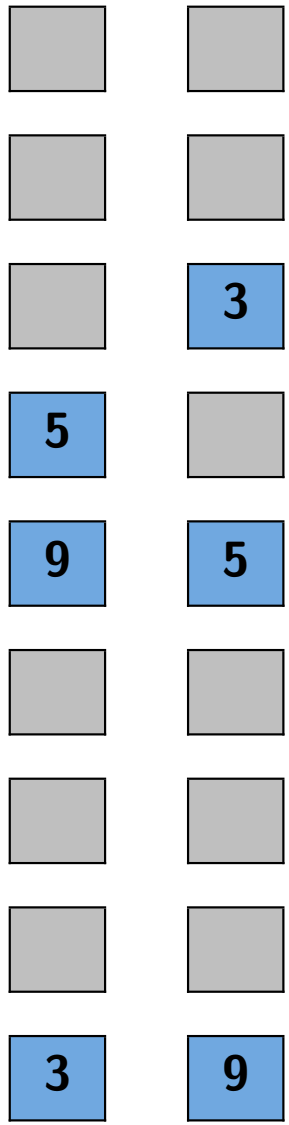
1					7		9	
	3			2				8
		9	6					
		5	3					
	1			8				2
6					4			
3							1	
	4							7
		7						

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

1					7		9	
	3			2				8
		9	6					
		5	3					
	1			8				2
6					4			
3							1	
	4							7
		7						

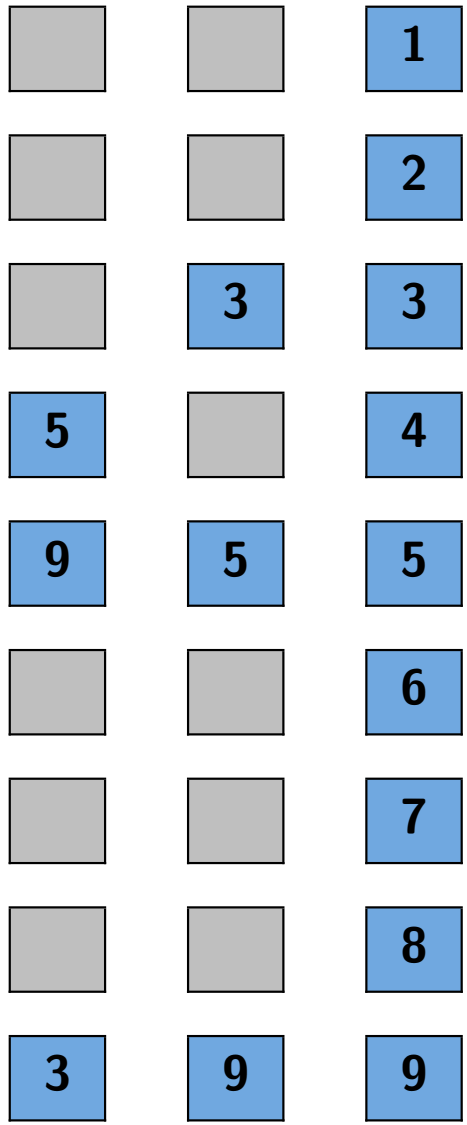


Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

1					7		9	
	3			2				8
		9	6					
		5	3					
	1			8				2
6					4			
3							1	
	4							7
		7						



Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

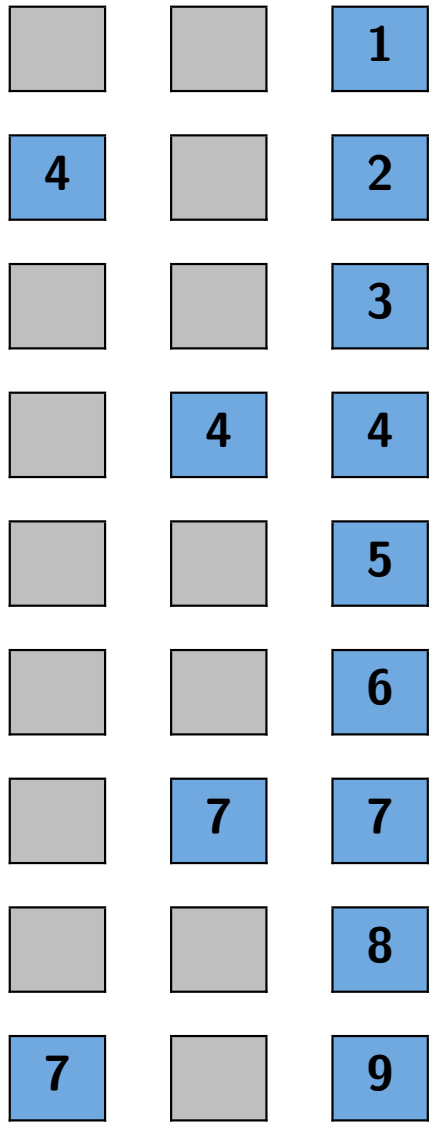
1					7		9	
	3			2				8
		9	6			5		
		5	3			9		
	1			8				2
6					4			
3							1	
	4							7
		7				3		

Preuve à divulgation nulle de connaissance (Zero Knowledge Proof)

Objectif : prouver à quelqu'un que l'on détient un secret (ou qu'une propriété est vraie) sans dévoiler la moindre information sur ce secret

Illustration : une grille de sudoku

1					7		9	
	3			2				8
		9	6			5		
		5	3			9		
	1			8				2
6					4			
3							1	
		7				3		



Prouver l'intégrité d'un bulletin

Il existe plusieurs preuves à divulgation nulle de connaissance :

- étant donné g^x prouver que l'on possède x ;
- étant donné $(g^{m_1}h^{t_1}, g^{t_1})$ et $(g^{m_2}h^{t_2}, g^{t_2})$ deux messages chiffrés par El Gamal prouver que $m_1 = m_2$ c'est-à-dire que les deux chiffrés chiffrent le même message ;
- étant donné P et Q deux assertions qui possèdent une preuve à divulgation nulle de connaissance, on peut les combiner et prouver « P ou Q ».

Les deux derniers points permettent de prouver que l'on chiffre $v \in \{0, 1\}$ par El Gamal.

Prouver l'intégrité d'un bulletin

Il existe plusieurs preuves à divulgation nulle de connaissance :

- étant donné g^x prouver que l'on possède x ;
- étant donné $(g^{m_1}h^{t_1}, g^{t_1})$ et $(g^{m_2}h^{t_2}, g^{t_2})$ deux messages chiffrés par El Gamal prouver que $m_1 = m_2$ c'est-à-dire que les deux chiffrés chiffrent le même message ;
- étant donné P et Q deux assertions qui possèdent une preuve à divulgation nulle de connaissance, on peut les combiner et prouver « P ou Q ».

Les deux derniers points permettent de prouver que l'on chiffre $v \in \{0, 1\}$ par El Gamal.

Soit $G = \langle g \rangle$ un groupe cyclique d'ordre N, $h = g^x$ avec $x \in [\![0, N-1]\!]$ et $v \in \{0, 1\}$.

On chiffre v avec El Gamal et la clef publique (g, h) : $(g^v h^t, g^t)$ avec $t \in [\![0, N-1]\!]$.

- **P** choisit aléatoirement $(w, r, d) \in [\![0, N-1]\!]^3$ et calcule puis envoie à **V** :
  - si $v = 0$: $(a_0, b_0) = (h^w, g^w)$ et $(a_1, b_1) = (h^{td+r}, g^{td+r})$
  - si $v = 1$: $(a_0, b_0) = (g^d h^{td+r}, g^{td+r})$ et $(a_1, b_1) = (h^w, g^w)$
- **V** choisit aléatoirement $e \in [\![0, N-1]\!]$ et l'envoie à **P**
- **P** calcule puis envoie à **V** :
  - si $v = 0$: $(d_0, d_1) = (e - d, d)$ et $(r_0, r_1) = (w - td_0, r)$
  - si $v = 1$: $(d_0, d_1) = (d, e - d)$ et $(r_0, r_1) = (r, w - td_1)$
- **V** vérifie que $e = d_0 + d_1$ et pour $i \in \{0, 1\}$, $a_i = g^{vd_i} h^{td_i + r_i}$ et $b_i = g^{td_i + r_i}$.

Fonctionnement final de Belenios

1. **Initialisation du vote** :
   - Génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes
   - La clef pub_E est distribuée aux votants
   - La clef $priv_E$ est partagée aux autorités de contrôle avec un seuil
   - Chaque votant reçoit une clef privée (envoyée par une autorité externe) pour pouvoir signer son vote

Fonctionnement final de Belenios

1. **Initialisation du vote** :
   - Génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes
   - La clef pub_E est distribuée aux votants
   - La clef $priv_E$ est partagée aux autorités de contrôle avec un seuil
   - Chaque votant reçoit une clef privée (envoyée par une autorité externe) pour pouvoir signer son vote
2. **Vote** : Chaque votant chiffre son vote avec la clef publique du vote pub_E, signe son vote avec sa clef privée puis prouve la validité de son vote avec une preuve à divulgation nulle de connaissance. Tout cela est publié sur une page publique.

Urne (page web publique)

Alice	$\{V_A\}_{pub_E}$	$[\{V_A\}_{pub_E}]_{priv_A}$	$ZKProof(V_A \in \{0,1\})$
Bob	$\{V_B\}_{pub_E}$	$[\{V_B\}_{pub_E}]_{priv_B}$	$ZKProof(V_B \in \{0,1\})$
Carol	$\{V_C\}_{pub_E}$	$[\{V_C\}_{pub_E}]_{priv_C}$	$ZKProof(V_C \in \{0,1\})$

Fonctionnement final de Belenios

1. **Initialisation du vote** :
   - Génération d'un couple de clef $(pub_E, priv_E)$ pour chiffrer les votes
   - La clef pub_E est distribuée aux votants
   - La clef $priv_E$ est partagée aux autorités de contrôle avec un seuil
   - Chaque votant reçoit une clef privée (envoyée par une autorité externe) pour pouvoir signer son vote
2. **Vote** : Chaque votant chiffre son vote avec la clef publique du vote pub_E, signe son vote avec sa clef privée puis prouve la validité de son vote avec une preuve à divulgation nulle de connaissance. Tout cela est publié sur une page publique.

Urne (page web publique)

Alice	$\{V_A\}_{pub_E}$	$[\{V_A\}_{pub_E}]_{priv_A}$	$ZKProof(V_A \in \{0,1\})$
Bob	$\{V_B\}_{pub_E}$	$[\{V_B\}_{pub_E}]_{priv_B}$	$ZKProof(V_B \in \{0,1\})$
Carol	$\{V_C\}_{pub_E}$	$[\{V_C\}_{pub_E}]_{priv_C}$	$ZKProof(V_C \in \{0,1\})$

3. **Dépouillement** : Les autorités réunies calculent

$$\{V_1\}_{pub_E} \times \cdots \times \{V_n\}_{pub_E} = \{V_1 + \cdots + V_n\}_{pub_E}.$$

puis déchiffrent le résultat.
Elles publient une preuve à divulgation nulle de connaissance du résultat.

Propriétés du protocole Belenios

1. ✔ L'urne doit être publique et le vote vérifiable
2. ? Seuls les inscrits votent
3. ✔ Chacun peut vérifier que les votes viennent d'inscrits et que chaque votant ne vote qu'une fois
4. ✔ Tout le monde doit pouvoir vérifier le comptage final
5. ✔ Le vote doit être secret : l'urne ne doit pas permettre de savoir qui a voté quoi
6. ✔ Chaque votant doit pouvoir vérifier que son vote est dans l'urne et n'est pas modifié
7. ✘ Un votant ne doit pas pouvoir prouver son vote à un tiers
8. ? Tout cela même si les autorités sont malhonnêtes

Fin

Merci pour votre attention et merci à Alice et Bob !

Sources

Articles historiques

- SHANNON, Claude Elwood. A mathematical theory of communication. The Bell system technical journal, 1948, vol. 27, no 3, p. 379-423.
- DIFFIE, Whitfield et HELLMAN, Martin E. New directions in cryptography. In : Democratizing Cryptography : The Work of Whitfield Diffie and Martin Hellman. 2022. p. 365-390.
- ELGAMAL, Taher. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 1985, vol. 31, no 4, p. 469-472.

Cours généralistes

- ZÉMOR, Gilles. Cours de cryptographie. Cassini, 2000.
- LEGRANDGÉRARD, Yves. Cours de Cryptographie. Laboratoire IRIF - Université Paris Diderot, 2019
- BERRY, Gérard. Où va l'informatique ? - Les aspects scientifiques de la sécurité informatique. Cours au collège de France, 2019

Documents spécialisés

- VITSE, Vanessa. Attaques algébriques du problème du logarithme discret sur courbes elliptiques. 2011. Thèse de doctorat. Université de Versailles-Saint Quentin en Yvelines.
- CORTIER, Véronique, GAUDRY, Pierrick, et GLONDU, Stéphane. Belenios : a simple private and verifiable electronic voting system. Foundations of Security, Protocols, and Equational Reasoning : Essays Dedicated to Catherine A. Meadows, 2019, p. 214-238.

Annexe 1 : Quid de l'implémentation

Exemple : vérifier qu'un mot de passe entré est correct

```
def verif_clef(mot : str) -> bool :
    clef = "mot_de_passe"

    if len(mot) != len(clef) :
        return False

    for i in range(len(clef)) :
        if mot[i] != clef[i] :
            return False

    return True
```

Annexe 1 : Quid de l'implémentation

Exemple : vérifier qu'un mot de passe entré est correct

```
def verif_clef(mot : str) -> bool :
    clef = "mot_de_passe"

    if len(mot) != len(clef) :
        return False

    for i in range(len(clef)) :
        if mot[i] != clef[i] :
            return False

    return True
```

Un problème de temps

- verif_clef("abc_de_passe") met 2.46e-07 s
- verif_clef("mot_et_passe") met 4.31e-07 s
- verif_clef("mot_de_passe") met 7.69e-07 s
- verif_clef("test") met 1.04e-07 s

Annexe 1 : Quid de l'implémentation

Une autre solution :

```
def verif_clef(mot : str) -> bool :
    clef = "mot_de_passe"

    resultat = True

    if len(mot) != len(clef) :
        resultat = False

    for i in range(len(clef)) :
        if resultat and mot[i] != clef[i] :
            resultat = False

    return resultat
```

Dans certains langages, le compilateur «optimise» par défaut le code en le transformant : il remplace `resultat=False` par **`return`** `False` ce qui casse la sécurité.

La bonne méthode est d'utiliser une fonction de hachage.

Annexe 2 : Canaux cachés

Les attaques par canaux cachés utilisent les failles matérielles pour récupérer des informations sur le secret utilisé.
On peut exploiter ces failles en :

- analysant la tension : certaines opérations ont une consommation de courant spécifique ;
- mesurant le temps d'exécution ;
- écoutant les bruits : écouter le bruit d'un processeur avec un smartphone peut permettre de découvrir la clef secrète RSA lors d'un déchiffrement (CVE-2013-4576)

Annexe 2 : Canaux cachés

Les attaques par canaux cachés utilisent les failles matérielles pour récupérer des informations sur le secret utilisé.
On peut exploiter ces failles en :

- analysant la tension : certaines opérations ont une consommation de courant spécifique ;
- mesurant le temps d'exécution ;
- écoutant les bruits : écouter le bruit d'un processeur avec un smartphone peut permettre de découvrir la clef secrète RSA lors d'un déchiffrement (CVE-2013-4576)

Les attaques par injection de fautes consistent à perturber l'exécution d'un algorithme en :

- provoquant des sur tensions ;
- utilisant un laser ;
- en chauffant.

C'est en faisant des erreurs que l'on apprend !

Annexe 3 : L'algorithme RSA

Génération de clefs :

1. On choisit deux nombres premiers p et q
2. On calcule $n = pq$
3. On calcule $\omega = (p-1)(q-1)$ et on choisit $d \in [\![1, \omega]\!]$ premier avec ω
4. On choisit $e \in [\![1, \omega]\!]$ tel que $ed \equiv 1\ [\omega]$

- Clef privée : $(n, d) \in \mathbb{N}^2$
- Clef publique : $(n, e) \in \mathbb{N}^2$
- Ensemble des messages : $\mathscr{M} = \mathbb{Z}/n\mathbb{Z}$; ensemble des chiffrés : $\mathscr{C} = \mathbb{Z}/n\mathbb{Z}$
- Chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$: m^e
- Déchiffrement d'un chiffré $c \in \mathbb{Z}/n\mathbb{Z}$: c^d

Problème de la factorisation : Étant donné $n = pq$ avec p et q deux grands nombres premiers trouver p et q.

Annexe 4 : Preuves de sécurité El Gamal

Au sens de Shannon la sécurité parfaite est impossible pour les chiffrements asymétriques contrairement au chiffrements symétriques (cf chiffre de Vernam).

Buts de l'attaquant :

- **BRK** - Cassage total : retrouver la clé de déchiffrement
- **OW** - Sens unique : déchiffrer un message « challenge »
- **IND** - Indistinguabilité : à partir d'un chiffré, extraire une information sur le message correspondant (à part la taille)

Moyens de l'attaquant :

- **KPA** - Clairs connus : il voit des couples clairs/chiffrés
- **CPA** - Clairs choisis : il chiffre des messages de son choix
- **CCA** - Chiffrés choisis : il peut faire déchiffrer des messages arbitraires sauf le « challenge »

Problèmes difficiles : Soit $(G, \times)$ un groupe cyclique engendré par g

- **DLOG** - Logarithme discret : étant donné g et g^x calculer $x = \log_g(g^x)$
- **CDH** - Calculer Diffie-Hellman : étant donné g, g^a et g^b calculer g^{ab}
- **DDH** - Décider Diffie-Hellman : étant donné g, g^a, g^b et g^c décider si $g^c = g^{ab}$

Annexe 4 : Preuves de sécurité El Gamal

Buts de l'attaquant :

- **BRK** - Cassage total : retrouver la clé de déchiffrement
- **OW** - Sens unique : déchiffrer un message « challenge »
- **IND** - Indistinguabilité : à partir d'un chiffré, extraire une information sur le message correspondant (à part la taille)

Moyens de l'attaquant :

- **KPA** - Clairs connus : il voit des couples clairs/chiffrés
- **CPA** - Clairs choisis : il chiffre des messages de son choix
- **CCA** - Chiffrés choisis : il peut faire déchiffrer des messages arbitraires sauf le « challenge »

Problèmes difficiles : Soit $(G, \times)$ un groupe cyclique engendré par g

- **DLOG** - Logarithme discret : étant donné g et g^x calculer $x = \log_g(g^x)$
- **CDH** - Calculer Diffie-Hellman : étant donné g, g^a et g^b calculer g^{ab}
- **DDH** - Décider Diffie-Hellman : étant donné g, g^a, g^b et g^c décider si $g^c = g^{ab}$

- En terme de sécurité on a **BRK** $\Longrightarrow$ **OW** $\Longrightarrow$ **IND**
- On a **DLOG** $\Longrightarrow$ **CDH** $\Longrightarrow$ **DDH**
- **BRK-CPA** $\Longrightarrow$ **DLOG** ; **OW-CPA** $\Longrightarrow$ **CDH** ; **IND-CPA** $\Longrightarrow$ **DDH**
- Vulnérabilité du chiffrement El Gamal : **OW-CCA**

Annexe 5 : Chiffrement symétrique

- Shanon introduit deux propriétés que devrait satisfaire un bon algorithme de chiffrement symétrique : la **diffusion** et la **confusion**.
- La propriété de **diffusion** signifie que des changements minimes dans les données en entrée se traduisent par des changements importants dans les données en sortie.
- La propriété de **confusion** mesure la complexité de l'interdépendance entre la clé, le clair et le chiffré. Plus cette complexité est grande, meilleur est l'algorithme.
- En pratique, la **diffusion** est un processus essentiellement **linéaire** alors qu'au contraire la **confusion** s'appuie sur des opérateurs **non-linéaires**.

- L'idée principale est de construire une fonction de chiffrement f_K (pour une clef K) par **itérations successives** d'une fonction simple g_K : $f_K = g_K^d$.
- Le comportement de g^d peut être imprévisible (pour d assez grand), même si g est très simple (cf système dynamique).
- Dans la pratique on « découpe » la clef K en plusieurs sous-clefs $K_1, \cdots, K_d$ et on utilise

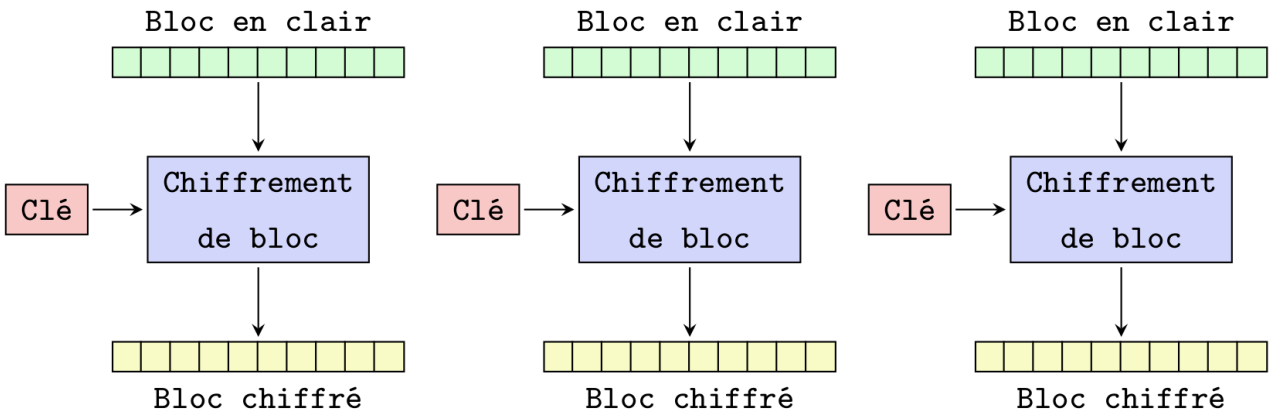
$$f_K = g_{K_d} \circ \cdots \circ g_{K_1}.$$

- Pour AES-128 : $d = 10$, pour AES-192 : $d = 12$ et pour AES-256 : $d = 14$.

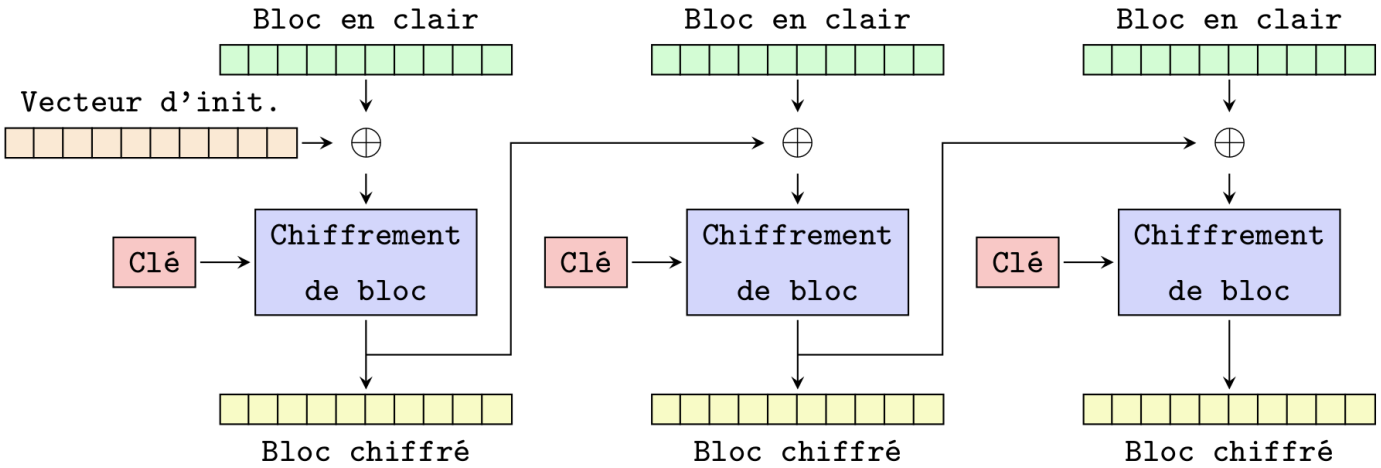
Annexe 6 : Mode opératoire de chiffrement par bloc

- Il y a deux grandes familles de chiffrements symétriques : les **chiffrements par bloc** (ex. AES) et les **chiffrements par flot** (ex. Chacha)
- Les chiffrements par bloc ne peuvent chiffrer que des blocs de taille fixe (ex. pour AES les blocs font 128 bits). Pour chiffrer des données plus importantes il faut les découper en petits blocs et chiffrer chaque bloc. Pour cela il existe plusieurs modes.

Annexe 6 : Mode opératoire de chiffrement par bloc



Une mauvaise idée (mode ECB) : le problème majeur de ce mode est que si plusieurs blocs sont identiques alors leur chiffré sera identique.



Une idée (parmi d'autres) intéressante (mode CBC)

Annexe 7 : Taille des clefs

Comparaison de la taille des clefs (en bits)

AES	RSA	El Gamal	Taille Groupe El Gamal
80	1 024	160	1 024
128	3 072	256	3 072
192	7 680	384	7 680
256	15 360	512	15 360

Figure – NIST Recommendations (2020)

Recommandation ANSSI : AES ≥ 128 bits et RSA ≥ 3072 bits

	RSA-1024	ECC-160
Génération de clefs	0.958s	0.219s
Chiffrement	0.063s	1.147s
Déchiffrement	0.062s	0.015s

Figure – Performance sur un processeur dual core 2.4GHz équipé de 2GB de mémoire vive

Complexité du meilleur algorithme connu pour

- Factoriser $N = pq$ avec p et q premiers : $e^{\sqrt{2\ln(N)\ln(\ln(N))}}$
- Calculer le logarithme discret sur corps fini de cardinal N : $e^{c\ln(N)^{\frac{1}{3}}\ln(\ln(N))^{\frac{2}{3}}}$
- Calculer le logarithme discret sur courbe elliptique de cardinal N : $O(\sqrt{N})$