

Schéma k -sur- n sous Tails : formulation mathématique rigoureuse

BC69 21A8 5B8D DBB5 F3A6 EB81 9055 4E6A 6924 F3C7

4 décembre 2025

Résumé

Ce texte formalise, dans un cadre entièrement mathématique, le schéma k -sur- n mis en œuvre par la procédure opérationnelle sous Tails pour la gestion d'un secret maître, telle qu'elle est décrite dans le document `procedure_kofn_tails_v1.html` du dépôt [1] dont l'empreinte SHA-256 est

ed223dd055108b0c08cd6d7b3011962d5bdc5f92dd38760c5c6a202982fcde84

Le protocole repose sur les briques cryptographiques suivantes :

- le partage de secret de Shamir [2] sur un corps fini pour répartir un secret maître S selon un seuil k -sur- n ;
- la dérivation de clés symétriques à partir de S et des mots de passe par **HKDF** (**HMAC-based Key Derivation Function**) [4] et **PBKDF2** (**Password-Based Key Derivation Function 2**) [5] ;
- une clé de signature **Ed25519** (Edwards-curve Digital Signature Algorithm) [6] dérivée de S (schéma de signature à base de courbe elliptique) ;
- un chiffrement hybride **RSA-4096 / AES-256-GCM** (**Advanced Encryption Standard** en mode **Galois/Counter Mode**) [7, 8, 10] pour les fichiers destinés aux membres du groupe k -sur- n , la clé RSA privée étant elle-même chiffrée par AES-256-GCM à partir de S ;

Nous détaillons, pour chaque type d'objet manipulé, les garanties effectives en termes de confidentialité, d'intégrité et d'authenticité :

- **Parts de secret** : confidentialité et intégrité assurées par AES-GCM avec une clé dérivée du mot de passe du porteur (PBKDF2) ;
- **Signatures de fichiers** : intégrité et authenticité cryptographiques assurées par Ed25519 à partir de la clé dérivée de S ;
- **Fichiers chiffrés hybrides RSA–AES** : confidentialité et intégrité du contenu assurées par AES-GCM ; l'authenticité de la *source* n'est en revanche pas garantie en l'absence de signature dédiée ;
- **Clés publiques** : leur authenticité doit être vérifiée par des canaux hors-bande (vérification humaine, échange préalable sur canal sécurisé, etc.).

Les objets sont définis dans leur structure algébrique naturelle (corps finis, groupes, espaces de clés), puis mis en correspondance explicite avec les formats de fichiers et les scripts décrits dans `procedure_kofn_tails_v1.html`, de façon à assurer une traçabilité complète entre le modèle mathématique et la mise en œuvre opérationnelle. Dans toute la suite, l'expression « procédure Tails » désigne précisément ce document HTML identifié par l'empreinte SHA-256 ci-dessus.

Table des matières

Table des notations	4
1 Cadre algébrique de base	5
1.1 Préambule et motivation	5
1.2 Arithmétique modulaire et groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^\times$	5
1.3 Théorèmes fondamentaux d'isomorphisme	9
1.4 Corps finis et extensions	11
1.5 Mots binaires et conventions d'encodage	17
2 Secret maître et schéma de Shamir	18
2.1 Secret maître	18
2.2 Schéma (k, n) de Shamir	18
2.3 Confidentialité parfaite du schéma de Shamir	19
3 Fonctions de hachage, HMAC, HKDF, PBKDF2	20
3.1 Fonction de hachage	20
3.2 Fonction de hachage cryptographique SHA-256	20
3.3 HMAC	23
3.4 Fonction d'encodage d'entier	23
3.5 HKDF	23
3.6 PBKDF2	24
4 Structure de groupe et signature Ed25519	24
4.1 Courbe elliptique Ed25519	24
4.2 Encodage des points et scalaires	26
4.3 Décodage des points Ed25519	26
4.4 Clés Ed25519	30
4.5 Algorithme de signature Ed25519	31
4.6 Algorithme de vérification Ed25519	31
5 RSA, MGF1 et OAEP	32
5.1 Chiffrement asymétrique RSA	32
5.2 MGF1	35
5.3 RSA-OAEP	36

6	AES-GCM	38
6.1	AES comme permutation de bloc	38
6.2	Correspondance des corps finis	38
6.3	Transformations de base	39
6.4	Expansion de clé AES-256	40
6.5	Algorithme de chiffrement complet	41
6.6	Corps $\mathbb{F}_{2^{128}}$ et GHASH	42
6.7	Mode AES-GCM	42
6.8	Dénombrement des permutations vs. clés	45
7	Construction globale du schéma k-sur-n	46
7.1	Paramètres et constantes	46
7.2	Cérémonie initiale	46
7.3	Protection des parts	47
7.4	Chiffrement hybride des fichiers	47
8	Correspondance avec les scripts Tails v1	47
8.1	Secret maître et partage Shamir	48
8.2	Clés dérivées, sels HKDF et Ed25519	48
8.3	PBKDF2, sels S_i et chiffrement des parts	48
8.4	RSA, OAEP et chiffrement hybride	48
8.5	Signature Ed25519 avec contexte	49
8.6	Gestion sécurisée de la mémoire	49
8.7	Résumé sur les sels publics	49
9	Résumé des propriétés de sécurité	49
A	Annexe A — Corps finis et notation \mathbb{F}_{2^n}	51
A.1	Caractéristique et sous-corps premier	51
A.2	Cardinal d'un corps fini	52
A.3	Frobenius et polynôme $X^{p^n} - X$	52
A.4	Polynôme $X^{p^n} - X$ et polynômes irréductibles	55
A.5	Cyclicité du groupe multiplicatif	57
A.6	Unicité de \mathbb{F}_{p^n} à isomorphisme près	59
	Références	60

Table des notations

$\{0, 1\}^t$	Mots binaires de longueur t
$\{0, 1\}^*$	Mots binaires de longueur finie (union sur $t \geq 0$)
$\ x\ $	Longueur en bits de $x \in \{0, 1\}^*$
$x \parallel y$	Concaténation de mots binaires
$x \oplus y$	XOR bit-à-bit (x, y de même longueur)
$\gcd(a, b)$	Plus grand commun diviseur de a et b
$\text{val}_{\text{be}}(b)$	Encodage big-endian $b \in \{0, 1\}^t \rightarrow \{0, \dots, 2^t - 1\}$
$\text{val}_{\text{le}}(b)$	Encodage little-endian $b \in \{0, 1\}^t \rightarrow \{0, \dots, 2^t - 1\}$
$S \in \{0, 1\}^{256}$	Secret maître (32 octets)
s_0	Entier $\text{val}_{\text{be}}(S) \in \{0, \dots, 2^{256} - 1\}$
$P = 2^{521} - 1$	Premier de Mersenne, cardinal de \mathbb{F}_P
\mathbb{F}_P	Corps fini à P éléments
$\mathbb{F}_P[X]$	Polynômes à coefficients dans \mathbb{F}_P
$\mathcal{P}_{<k}$	Polynômes de degré $< k$ dans $\mathbb{F}_P[X]$
(x_i, y_i)	Part de Shamir du participant i ($\in \mathbb{F}_P^2$)
$q = 2^{255} - 19$	Cardinal du corps de base de Ed25519
$E(\mathbb{F}_q)$	Groupe des points de la courbe elliptique
$B \in E(\mathbb{F}_q)$	Point de base d'ordre premier ℓ
ℓ	Ordre premier de B ($\approx 2^{252}$)
\mathbb{Z}_ℓ	Anneau $\{0, \dots, \ell - 1\}$ modulo ℓ
$a \in \mathbb{Z}_\ell$	Clé privée Ed25519
$A = aB$	Clé publique Ed25519
(n, e, d)	Paramètres RSA : module, exposants public/privé
h	Fonction de hachage $h : \{0, 1\}^* \rightarrow \{0, 1\}^d$
HMAC_h	MAC HMAC basé sur h (avec h fonction de hachage sous-jacente)
HKDF_h	Dérivation de clé HKDF (RFC 5869) basée sur h
PBKDF2_h	Dérivation par mot de passe (PBKDF2, RFC 8018) basée sur h
MGF1_h	Génération de masque MGF1 (RFC 8017) basée sur h
RSA-OAEP	RSA avec rembourrage OAEP (RFC 8017)
AES-GCM_K	AES-GCM sous clé symétrique K
IKM	Input Keying Material (ici S)
salt_{Ed}	Sel HKDF constant pour K_{Ed}
W	Sel HKDF aléatoire par cérémonie pour K_{RSA}
info_{Ed}	Contexte HKDF pour Ed25519
info_{RSA}	Contexte HKDF pour RSA
P_i	Mot de passe du participant i
S_i	Sel PBKDF2 individuel
K_i	Clé symétrique pour chiffrer la part i
K_{Ed}	Graine Ed25519 dérivée de S
K_{RSA}	Clé pour chiffrer la clé privée RSA
K_{AES}	Clé de session AES
N, N_i	Nonces AES-GCM
\perp	Symbole d'échec/rejet

1 Cadre algébrique de base

1.1 Préambule et motivation

Cette section rassemble les notions d'algèbre et d'arithmétique modulaire utilisées dans la suite du document. Nous suivons une progression logique :

- L'**arithmétique modulaire** (entiers, pgcd, inverses) définit le socle concret pour l'anneau $\mathbb{Z}/n\mathbb{Z}$ et prépare le corps $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$.
- Les **théorèmes d'isomorphisme** généralisent ces structures, s'appuyant sur les exemples concrets.
- Les **corps finis et extensions** s'appuient sur les deux précédents (utilisation des isomorphismes pour l'unicité de \mathbb{F}_p).
- Les **mots binaires et encodages** formalisent la conversion vers les représentations informatiques.

1.2 Arithmétique modulaire et groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^\times$

Théorème 1.1 (Identité de Bézout (lemme de Bachet-Bézout)). Soient $a, b \in \mathbb{Z}$ deux entiers non tous les deux nuls. Il existe $u, v \in \mathbb{Z}$ tels que :

$$au + bv = \gcd(a, b)$$

Démonstration. Soit $d = \gcd(a, b)$. Considérons l'ensemble :

$$S = \{ax + by \mid x, y \in \mathbb{Z} \text{ et } ax + by > 0\}$$

L'ensemble S est non vide. Puisque a et b ne sont pas tous deux nuls, au moins l'un des nombres $|a|$ ou $|b|$ est strictement positif. Ainsi, si $a \neq 0$, alors $a \cdot 1 + b \cdot 0 = |a| > 0$, donc $|a| \in S$; si $b \neq 0$, alors $a \cdot 0 + b \cdot 1 = |b| > 0$, donc $|b| \in S$.

Soit m le plus petit élément de S . Par le principe du bon ordre (l'ensemble des entiers positifs est bien ordonné, donc tout sous-ensemble non vide d'entiers positifs possède un plus petit élément), S a un plus petit élément que nous notons m . Par définition de S , il existe $u_0, v_0 \in \mathbb{Z}$ tels que :

$$m = au_0 + bv_0 > 0.$$

m divise a et b . Effectuons la division euclidienne de a par m : $a = mq + r$ avec $0 \leq r < m$. Alors :

$$r = a - mq = a - (au_0 + bv_0)q = a(1 - u_0q) + b(-v_0q).$$

Si $r > 0$, alors $r \in S$ et $r < m$, ce qui contredit la minimalité de m . Donc $r = 0$, et m divise a . De même, en divisant b par m , on montre que m divise b .

m est le plus grand diviseur commun. Puisque m divise a et b , et que $m = au_0 + bv_0$, tout diviseur commun de a et b divise aussi m . En effet, si d' divise a et b , alors $a = d'a'$ et $b = d'b'$, donc :

$$m = d'a'u_0 + d'b'v_0 = d'(a'u_0 + b'v_0),$$

et d' divise m .

En particulier, $d = \gcd(a, b)$ divise m . Mais puisque m divise a et b , et que d est le plus grand diviseur commun (au sens de la divisibilité), on a $m \leq d$. Comme d divise m , on a aussi $d \leq m$. Donc $m = d$.

Conclusion. Nous avons montré que le plus petit élément positif m de S est égal à $\gcd(a, b)$, et qu'il s'écrit $m = au_0 + bv_0$ pour certains $u_0, v_0 \in \mathbb{Z}$. Ainsi, $\gcd(a, b) = au_0 + bv_0$. \square

Remarque 1.2 (Nombres premiers entre eux). Deux entiers a et b sont dits *premiers entre eux* (ou *copremiers*) si et seulement si $\gcd(a, b) = 1$. Cette propriété est fondamentale pour l'inversibilité modulo n et pour les théorèmes d'Euler et de Fermat.

Théorème 1.3 (Théorème de Gauss (lemme d'Euclide)). Soient a, b, c des entiers. Si a et b sont premiers entre eux et a divise bc , alors a divise c .

Démonstration. Puisque $\gcd(a, b) = 1$, par l'identité de Bézout, il existe des entiers u, v tels que :

$$au + bv = 1$$

Multiplions cette égalité par c :

$$auc + bvc = c$$

Par hypothèse, a divise bc , donc a divise bvc . De plus, a divise évidemment auc . Donc a divise la somme $auc + bvc = c$. \square

Définition 1.4 (Anneau $\mathbb{Z}/n\mathbb{Z}$ et groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^\times$). Soit $n \in \mathbb{N}^*$. L'anneau $\mathbb{Z}/n\mathbb{Z}$ est l'ensemble $\{0, 1, \dots, n-1\}$ muni des opérations $+$ et \cdot modulo n .

Le groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^\times$ est l'ensemble des éléments inversibles de $\mathbb{Z}/n\mathbb{Z}$:

$$(\mathbb{Z}/n\mathbb{Z})^\times = \{a \in \{0, 1, \dots, n-1\} \mid \gcd(a, n) = 1\}.$$

Remarque 1.5 (Inversibilité modulo n). L'élément $a \in \mathbb{Z}/n\mathbb{Z}$ est inversible si et seulement si $\gcd(a, n) = 1$. En effet, si $\gcd(a, n) = 1$, alors par l'identité de Bézout, il existe u, v tels que $au + nv = 1$, donc $au \equiv 1 \pmod{n}$. Réciproquement, si a est inversible, il existe b tel que $ab \equiv 1 \pmod{n}$, donc $ab - kn = 1$ pour un certain k , ce qui implique que tout diviseur commun à a et n divise 1, donc $\gcd(a, n) = 1$.

Définition 1.6 (Indicatrice d'Euler). L'indicatrice d'Euler $\varphi(n)$ est définie pour tout entier $n \geq 1$ comme le nombre d'entiers compris entre 1 et n qui sont premiers avec n :

$$\varphi(n) = \#\{k \in \{1, 2, \dots, n\} \mid \gcd(k, n) = 1\}$$

Proposition 1.7 (Valeur de $\varphi(n)$ pour $n = pq$). Si p et q sont deux nombres premiers distincts et $n = pq$, alors :

$$\varphi(n) = (p-1)(q-1)$$

Démonstration. Parmi les $n = pq$ entiers de 1 à n , les entiers qui ne sont pas premiers avec n sont :

- Les multiples de p : $p, 2p, 3p, \dots, qp$ il y en a q
- Les multiples de q : $q, 2q, 3q, \dots, pq$ il y en a p

L'entier pq a été compté deux fois. Par le principe d'inclusion-exclusion :

$$\varphi(n) = n - q - p + 1 = pq - p - q + 1 = (p-1)(q-1)$$

□

Théorème 1.8 (Théorème de Lagrange pour les groupes finis). Soit G un groupe fini et H un sous-groupe de G . Alors l'ordre de H divise l'ordre de G :

$$|H| \mid |G|$$

En particulier, pour tout $g \in G$, l'ordre de g divise $|G|$.

Démonstration. Considérons la relation d'équivalence sur G définie par :

$$x \sim y \iff x^{-1}y \in H$$

Vérifions que c'est une relation d'équivalence :

- Réflexivité : $x^{-1}x = e \in H$, donc $x \sim x$.
- Symétrie : si $x \sim y$, alors $x^{-1}y \in H$. Comme H est un sous-groupe, l'inverse $(x^{-1}y)^{-1} = y^{-1}x$ est aussi dans H , donc $y \sim x$.
- Transitivité : si $x \sim y$ et $y \sim z$, alors $x^{-1}y \in H$ et $y^{-1}z \in H$. Le produit $(x^{-1}y)(y^{-1}z) = x^{-1}z$ est dans H , donc $x \sim z$.

Les classes d'équivalence sont les *classes à gauche* $gH = \{gh : h \in H\}$.

Pour montrer que toutes les classes à gauche ont le même cardinal que H , considérons l'application $f : H \rightarrow gH$ définie par $f(h) = gh$. Cette application est :

- **Surjective** : Par définition de gH
- **Injective** : Si $gh_1 = gh_2$, alors en multipliant à gauche par g^{-1} , on obtient $h_1 = h_2$

Ainsi, $|gH| = |H|$ pour tout $g \in G$.

Puisque les classes d'équivalence forment une partition de G , on a :

$$|G| = \sum_{\text{classes}} |gH| = (\text{nombre de classes}) \times |H|$$

Donc $|H|$ divise $|G|$.

Pour la deuxième partie, si $g \in G$, considérons le sous-groupe engendré par g , noté $\langle g \rangle = \{g^n : n \in \mathbb{Z}\}$.

Puisque G est fini, les puissances successives de g ne peuvent être toutes distinctes. Il existe donc deux entiers $i < j$ tels que $g^i = g^j$, d'où $g^{j-i} = e$. Ainsi, l'ensemble $\{k \in \mathbb{N}^* : g^k = e\}$ est non vide, et par le principe du bon ordre, il possède un plus petit élément, noté d . Cet entier d est appelé l'ordre de g .

Montrons que $\langle g \rangle = \{e, g, g^2, \dots, g^{d-1}\}$. Soit $n \in \mathbb{Z}$. Par division euclidienne, on peut écrire $n = qd + r$ avec $0 \leq r < d$, et donc $g^n = g^{qd+r} = (g^d)^q \cdot g^r = e^q \cdot g^r = g^r$. Ainsi, tout élément de $\langle g \rangle$ est l'un des g^r avec $0 \leq r < d$. Ces éléments sont distincts, car si $g^i = g^j$ avec $0 \leq i < j < d$, alors $g^{j-i} = e$ avec $0 < j-i < d$, ce qui contredit la minimalité de d .

Donc $\langle g \rangle = \{e, g, g^2, \dots, g^{d-1}\}$ a exactement d éléments, ce qui signifie que le cardinal du sous-groupe $\langle g \rangle$ est égal à l'ordre de g .

Par la première partie du théorème, l'ordre de $\langle g \rangle$ (c'est-à-dire son cardinal) divise $|G|$, donc l'ordre de g divise $|G|$. □

Théorème 1.9 (Théorème d'Euler). Si a et n sont premiers entre eux, alors :

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Démonstration. L'ensemble $(\mathbb{Z}/n\mathbb{Z})^\times$ est un groupe multiplicatif (définition précédente) d'ordre $\varphi(n)$. Pour tout $a \in (\mathbb{Z}/n\mathbb{Z})^\times$, par le théorème de Lagrange (Théorème 1.8) appliqué au sous-groupe $\langle a \rangle$, l'ordre de a divise $\varphi(n)$. Donc $a^{\varphi(n)} = 1$ dans $(\mathbb{Z}/n\mathbb{Z})^\times$, c'est-à-dire $a^{\varphi(n)} \equiv 1 \pmod{n}$. \square

Théorème 1.10 (Petit théorème de Fermat). Si p est premier et a n'est pas divisible par p , alors :

$$a^{p-1} \equiv 1 \pmod{p}$$

Démonstration. C'est un cas particulier du théorème d'Euler, car pour p premier, $\varphi(p) = p - 1$. \square

Proposition 1.11 (Algorithme d'Euclide étendu et inverse modulaire). Soient $a, n \in \mathbb{Z}$ avec $\gcd(a, n) = 1$. L'algorithme d'Euclide étendu calcule des entiers u, v tels que

$$au + nv = 1.$$

En particulier, $u \bmod n$ est l'inverse de a modulo n .

Algorithme d'Euclide étendu. Soient $a \in \{1, 2, \dots, n-1\}$ et n premier avec a . L'algorithme calcule l'inverse de a modulo n .

Initialisation : Posons $r_0 \leftarrow n, r_1 \leftarrow a, u_0 \leftarrow 0, u_1 \leftarrow 1$. Par définition, on a :

$$r_0 = n = a \cdot 0 + n \cdot 1 \quad \text{et} \quad r_1 = a = a \cdot 1 + n \cdot 0.$$

Ainsi, en introduisant des coefficients $v_0 = 1$ et $v_1 = 0$ (non stockés explicitement), on obtient les relations initiales :

$$r_0 = au_0 + nv_0, \quad r_1 = au_1 + nv_1.$$

Invariants de boucle : À chaque étape de l'algorithme, on maintient les deux propriétés suivantes pour les valeurs courantes de r_0, r_1, u_0, u_1 :

1. **Expression linéaire :** Il existe des entiers v_0, v_1 tels que

$$r_0 = au_0 + nv_0 \quad \text{et} \quad r_1 = au_1 + nv_1.$$

2. **PGCD préservé :** $\gcd(r_0, r_1) = \gcd(n, a) = 1$.

Déroulement de la boucle : Tant que $r_1 \neq 0$:

1. Calculer $q \leftarrow \lfloor r_0/r_1 \rfloor$ (division euclidienne).
2. Mettre à jour :

$$\begin{aligned} r_2 &\leftarrow r_0 - q \cdot r_1, \\ u_2 &\leftarrow u_0 - q \cdot u_1. \end{aligned}$$

3. Décaler les variables :

$$\begin{aligned} r_0 &\leftarrow r_1, & r_1 &\leftarrow r_2, \\ u_0 &\leftarrow u_1, & u_1 &\leftarrow u_2. \end{aligned}$$

Preuve de l'invariant (1) : Supposons qu'avant une itération on ait $r_0 = au_0 + nv_0$ et $r_1 = au_1 + nv_1$. Alors, après calcul de q et mise à jour, on a :

$$\begin{aligned} r_2 &= r_0 - qr_1 \\ &= (au_0 + nv_0) - q(au_1 + nv_1) \\ &= a(u_0 - qu_1) + n(v_0 - qv_1) \\ &= au_2 + nv_2, \end{aligned}$$

en posant $v_2 = v_0 - qv_1$. Après le décalage, les nouvelles valeurs r_0, r_1, u_0, u_1 vérifient à nouveau la forme linéaire avec les coefficients correspondants.

Preuve de l'invariant (2) : La propriété $\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$ assure que le PGCD reste inchangé à chaque itération. Comme initialement $\gcd(n, a) = 1$, on a $\gcd(r_0, r_1) = 1$ tout au long.

Terminaison et résultat : La suite (r_i) est strictement décroissante et positive, donc r_1 finit par devenir nul. À la fin, on a $r_1 = 0$ et $r_0 = \gcd(n, a) = 1$ (par l'invariant 2). D'après l'invariant 1 appliqué à l'avant-dernière itération, on a alors :

$$1 = r_0 = au_0 + nv_0.$$

Ainsi, modulo n , on obtient $au_0 \equiv 1 \pmod{n}$. L'inverse de a modulo n est donc u_0 modulo n (c'est-à-dire $u_0 \bmod n$, pour le ramener dans $\{0, \dots, n-1\}$). \square

Remarque 1.12 (À retenir pour RSA). Pour RSA, nous utiliserons :

- Le théorème d'Euler (ou le petit théorème de Fermat) dans la preuve de correction (section 5.3).
- L'algorithme d'Euclide étendu pour calculer $d = e^{-1} \bmod \varphi(n)$.

1.3 Théorèmes fondamentaux d'isomorphisme

Définition 1.13 (Homomorphisme de groupes). Soient (G, \cdot) et $(H, *)$ deux groupes. Une application $f : G \rightarrow H$ est un *homomorphisme de groupes* si pour tous $g_1, g_2 \in G$:

$$f(g_1 \cdot g_2) = f(g_1) * f(g_2).$$

Le *noyau* de f est $\ker(f) = \{g \in G \mid f(g) = e_H\}$.

Définition 1.14 (Homomorphisme d'anneaux). Soient $(A, +_A, \cdot_A)$ et $(B, +_B, \cdot_B)$ deux anneaux. Une application $f : A \rightarrow B$ est un *homomorphisme d'anneaux* si pour tous $a_1, a_2 \in A$:

$$\begin{aligned} f(a_1 +_A a_2) &= f(a_1) +_B f(a_2) \\ f(a_1 \cdot_A a_2) &= f(a_1) \cdot_B f(a_2) \\ f(1_A) &= 1_B \end{aligned}$$

Le *noyau* de f est $\ker(f) = \{a \in A \mid f(a) = 0_B\}$.

Remarque 1.15. Les deux premières conditions impliquent automatiquement $f(0_A) = 0_B$. En effet, pour tout $a \in A$, on a $f(a) = f(a +_A 0_A) = f(a) +_B f(0_A)$, d'où $f(0_A) = 0_B$ par simplification dans le groupe additif $(B, +_B)$.

Théorème 1.16 (Premier théorème d'isomorphisme pour les groupes). Soit $f : G \rightarrow H$ un homomorphisme de groupes. Alors :

1. $\ker(f)$ est un sous-groupe distingué de G
2. $\text{Im}(f)$ est un sous-groupe de H
3. $G/\ker(f) \simeq \text{Im}(f)$

Démonstration. (1) et (2) sont des vérifications directes. Pour (3), définissons :

$$\bar{f} : G/\ker(f) \rightarrow \text{Im}(f), \quad g\ker(f) \mapsto f(g).$$

Vérifions que \bar{f} est bien définie : si $g_1\ker(f) = g_2\ker(f)$, alors $g_1^{-1}g_2 \in \ker(f)$, donc $f(g_1^{-1}g_2) = e_H$, d'où $f(g_1) = f(g_2)$. Ceci montre que \bar{f} ne dépend pas du choix du représentant de la classe $g\ker(f)$, ce qui est possible car $\ker(f)$ est distingué.

Montrons que \bar{f} est un homomorphisme :

$$\bar{f}((g_1\ker(f))(g_2\ker(f))) = \bar{f}(g_1g_2\ker(f)) = f(g_1g_2) = f(g_1)f(g_2) = \bar{f}(g_1\ker(f))\bar{f}(g_2\ker(f)).$$

L'injectivité vient de : si $\bar{f}(g\ker(f)) = e_H$, alors $f(g) = e_H$, donc $g \in \ker(f)$, d'où $g\ker(f) = \ker(f)$ (classe de l'élément neutre).

La surjectivité est immédiate par définition de $\text{Im}(f)$.

Ainsi \bar{f} est un isomorphisme. □

Définition 1.17 (Idéal d'un anneau). Un sous-ensemble I d'un anneau A est un *idéal* si :

1. $(I, +)$ est un sous-groupe de $(A, +)$
2. Pour tout $a \in A$ et tout $x \in I$, on a $a \cdot x \in I$ et $x \cdot a \in I$ (stabilité par multiplication par tout élément de l'anneau)

Théorème 1.18 (Premier théorème d'isomorphisme pour les anneaux). Soit $f : A \rightarrow B$ un homomorphisme d'anneaux. Alors :

1. $\ker(f)$ est un idéal de A
2. $\text{Im}(f)$ est un sous-anneau de B
3. $A/\ker(f) \simeq \text{Im}(f)$

Démonstration. (1) Montrons que $\ker(f)$ est un idéal. Soient $x, y \in \ker(f)$. Alors $f(x - y) = f(x) - f(y) = 0_B - 0_B = 0_B$, donc $x - y \in \ker(f)$. Soit $a \in A$ et $x \in \ker(f)$. Alors $f(a \cdot x) = f(a) \cdot f(x) = f(a) \cdot 0_B = 0_B$, donc $a \cdot x \in \ker(f)$. De même, $x \cdot a \in \ker(f)$. Ainsi $\ker(f)$ est un idéal.

(2) Soient $b_1, b_2 \in \text{Im}(f)$. Il existe $a_1, a_2 \in A$ tels que $f(a_1) = b_1$ et $f(a_2) = b_2$. Alors $b_1 - b_2 = f(a_1) - f(a_2) = f(a_1 - a_2) \in \text{Im}(f)$ et $b_1 \cdot b_2 = f(a_1) \cdot f(a_2) = f(a_1 \cdot a_2) \in \text{Im}(f)$. De plus, $1_B = f(1_A) \in \text{Im}(f)$. Donc $\text{Im}(f)$ est un sous-anneau de B .

(3) Définissons $\bar{f} : A/\ker(f) \rightarrow \text{Im}(f)$ par $\bar{f}(a + \ker(f)) = f(a)$. Cette application est bien définie car si $a + \ker(f) = a' + \ker(f)$, alors $a - a' \in \ker(f)$, donc $f(a - a') = 0_B$,

soit $f(a) = f(a')$. C'est un homomorphisme d'anneaux car :

$$\begin{aligned}\bar{f}((a + \ker(f)) + (b + \ker(f))) &= \bar{f}(a + b + \ker(f)) = f(a + b) = f(a) + f(b) \\ &= \bar{f}(a + \ker(f)) + \bar{f}(b + \ker(f)), \\ \bar{f}((a + \ker(f)) \cdot (b + \ker(f))) &= \bar{f}(a \cdot b + \ker(f)) = f(a \cdot b) = f(a) \cdot f(b) \\ &= \bar{f}(a + \ker(f)) \cdot \bar{f}(b + \ker(f)), \\ \bar{f}(1_A + \ker(f)) &= f(1_A) = 1_B.\end{aligned}$$

L'injectivité : si $\bar{f}(a + \ker(f)) = 0_B$, alors $f(a) = 0_B$, donc $a \in \ker(f)$, d'où $a + \ker(f) = 0_A + \ker(f)$. La surjectivité est évidente. Donc \bar{f} est un isomorphisme. \square

Remarque 1.19 (Utilisation en cryptographie). Ces théorèmes justifient plusieurs isomorphismes fondamentaux :

- $\mathbb{Z}/p\mathbb{Z} \simeq \mathbb{F}_p$ (corps premier) - essentiel pour RSA et Shamir
- $\mathbb{F}_2[X]/(m(X)) \simeq \mathbb{F}_{2^s}$ pour AES
- $\mathbb{Z}_\ell \simeq \langle B \rangle$ pour Ed25519 - isomorphisme entre scalaires et points du sous-groupe

Ils garantissent que différentes représentations d'une même structure algébrique sont équivalentes, ce qui permet de choisir la représentation la plus efficace pour l'implémentation.

1.4 Corps finis et extensions

Théorème 1.20 (Unicité du corps à p éléments). Soit p un nombre premier. Alors :

1. $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ est un corps à p éléments.
2. Tout corps K de cardinal p est isomorphe à \mathbb{F}_p .
3. Cet isomorphisme est unique : si $\phi_1, \phi_2 : K \rightarrow \mathbb{F}_p$ sont deux isomorphismes, alors $\phi_1 = \phi_2$.

Démonstration. (1) Puisque p est premier, tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible modulo p , donc $\mathbb{Z}/p\mathbb{Z}$ est un corps.

(2) Soit K un corps de cardinal p . Puisque K est fini, sa caractéristique est un nombre premier. Comme $p \cdot 1_K = 0$ (d'après le théorème de Lagrange appliqué au groupe additif), la caractéristique divise p , donc est égale à p .

Considérons l'homomorphisme d'anneaux canonique :

$$\iota : \mathbb{Z} \rightarrow K, \quad n \mapsto n \cdot 1_K$$

Son noyau est un idéal de \mathbb{Z} . Comme tout idéal de \mathbb{Z} est principal, il existe un entier $m \geq 0$ tel que $\ker \iota = m\mathbb{Z}$. Puisque $\iota(p) = p \cdot 1_K = 0$, on a $p \in \ker \iota$, donc m divise p . Comme ι n'est pas l'application nulle (car $\iota(1) = 1_K \neq 0$), on a $m = p$.

Ainsi $\ker \iota = p\mathbb{Z}$. Par le théorème fondamental d'isomorphisme (Théorème 1.18), on obtient un homomorphisme injectif :

$$\bar{\iota} : \mathbb{Z}/p\mathbb{Z} \hookrightarrow K$$

Puisque les deux ensembles ont le même cardinal fini p , $\bar{\iota}$ est un isomorphisme.

(3) Soient $\phi_1, \phi_2 : K \rightarrow \mathbb{F}_p$ deux isomorphismes. Alors $\psi = \phi_2^{-1} \circ \phi_1$ est un automorphisme de K .

Montrons que K est un \mathbb{F}_p -espace vectoriel de dimension 1. D'abord, comme K est de caractéristique p , le sous-corps premier (voir A.1) de K est isomorphe à \mathbb{F}_p . On peut donc identifier \mathbb{F}_p au sous-corps de K engendré par 1_K , et définir une structure de \mathbb{F}_p -espace vectoriel sur K par la multiplication interne de K .

Pour la dimension : si $\dim_{\mathbb{F}_p} K = d$, alors K est isomorphe à \mathbb{F}_p^d comme espace vectoriel, donc $|K| = p^d$. Or $|K| = p$, donc $p^d = p$ et $d = 1$. Ainsi, K est un \mathbb{F}_p -espace vectoriel de dimension 1.

Un automorphisme d'anneaux de K est en particulier un automorphisme de \mathbb{F}_p -espaces vectoriels. Tout endomorphisme d'un espace vectoriel de dimension 1 est la multiplication par un scalaire. Il existe donc $\lambda \in \mathbb{F}_p^*$ tel que $\psi(x) = \lambda \cdot x$ pour tout $x \in K$.

En particulier, $\psi(1_K) = \lambda \cdot 1_K = \lambda$. Comme ψ est un isomorphisme d'anneaux, on a $\psi(1_K) = 1_K$, donc $\lambda = 1_K = 1$ (dans \mathbb{F}_p). Ainsi $\psi = \text{id}_K$, ce qui implique $\phi_1 = \phi_2$. \square

Définition 1.21 (Corps fini premier). Soit p un nombre premier. On note

$$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$$

le corps fini à p éléments. Les opérations arithmétiques dans \mathbb{F}_p sont effectuées modulo p . Plus précisément, pour $a, b \in \mathbb{F}_p$:

- $a + b = (a + b) \bmod p$
- $a \cdot b = (a \cdot b) \bmod p$
- L'inverse additif de a est $p - a$ (modulo p)
- L'inverse multiplicatif de $a \neq 0$ est l'unique élément $a^{-1} \in \mathbb{F}_p$ tel que $a \cdot a^{-1} \equiv 1 \pmod{p}$

Exemple 1.22 (Corps \mathbb{F}_5). Le corps $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$ avec les opérations modulo 5.

Tables d'addition et de multiplication :

+	0	1	2	3	4	×	0	1	2	3	4
0	0	1	2	3	4	0	0	0	0	0	0
1	1	2	3	4	0	1	0	1	2	3	4
2	2	3	4	0	1	2	0	2	4	1	3
3	3	4	0	1	2	3	0	3	1	4	2
4	4	0	1	2	3	4	0	4	3	2	1

Calcul de l'inverse de 3 modulo 5 : on cherche x tel que $3x \equiv 1 \pmod{5}$. En inspectant la table, $3 \times 2 = 6 \equiv 1 \pmod{5}$, donc $3^{-1} = 2$.

Définition 1.23 (Corps \mathbb{F}_P pour le partage de Shamir). Dans le cadre du schéma k -sur- n , on utilise un corps fini premier particulier de cardinal

$$P = 2^{521} - 1$$

qui est un nombre premier de Mersenne. Ce choix est motivé par :

- Sa taille (521 bits) est suffisamment grande pour garantir la sécurité du partage de secret.

- Les opérations modulo P peuvent être optimisées grâce à la forme $2^{521} - 1$ (réduction rapide).
- Le corps \mathbb{F}_P permet de représenter de manière injective un secret maître $S \in \{0, 1\}^{256}$ (car $P > 2^{256}$).

Ce corps sera le support algébrique des polynômes de Shamir utilisés pour le partage du secret maître.

Définition 1.24 (Anneau de polynômes). Soit \mathbb{K} un corps commutatif. L'anneau des polynômes à coefficients dans \mathbb{K} est l'ensemble des suites (a_0, a_1, a_2, \dots) d'éléments de \mathbb{K} qui sont presque toutes nulles (c'est-à-dire qu'il existe un rang N tel que $a_n = 0$ pour tout $n \geq N$). Un tel polynôme s'écrit usuellement $P(X) = a_0 + a_1X + a_2X^2 + \dots + a_dX^d$ où d est le plus grand indice tel que $a_d \neq 0$ (degré).

Cas particulier : Le polynôme nul $(0, 0, 0, \dots)$ est noté 0 . Par convention, on pose $\deg(0) = -\infty$ (avec les conventions $-\infty < n$ pour tout $n \in \mathbb{N}$ et $-\infty + n = -\infty$).

Les opérations sont :

- Addition : $(a_n) + (b_n) = (a_n + b_n)$
- Multiplication : $(a_n) \cdot (b_n) = (c_n)$ avec $c_n = \sum_{i+j=n} a_i b_j$

On note cet anneau $\mathbb{K}[X]$.

Lemme 1.25 (Division euclidienne dans l'anneau $\mathbb{K}[X]$). Soient $A(X), B(X) \in \mathbb{K}[X]$ avec $B(X) \neq 0$. Il existe un unique couple $(Q(X), R(X)) \in \mathbb{K}[X]^2$ tel que :

$$A(X) = B(X)Q(X) + R(X) \quad \text{avec} \quad \deg(R(X)) < \deg(B(X)) \quad \text{ou} \quad R(X) = 0.$$

Démonstration. Existence. On procède par récurrence forte sur $\deg(A)$.

- Si $A = 0$ ou $\deg(A) < \deg(B)$, on pose $Q = 0$ et $R = A$, et on a bien $A = B \cdot 0 + R$ avec $\deg(R) < \deg(B)$ ou $R = 0$.

- Supposons $\deg(A) \geq \deg(B)$. Écrivons :

$$A(X) = a_n X^n + \dots + a_0, \quad B(X) = b_m X^m + \dots + b_0,$$

avec $a_n \neq 0$, $b_m \neq 0$ et $n \geq m$. Formons le polynôme :

$$A_1(X) = A(X) - \frac{a_n}{b_m} X^{n-m} B(X).$$

Le terme de degré n s'annule, donc $\deg(A_1) < n$ ou $A_1 = 0$. Par hypothèse de récurrence forte, il existe Q_1, R_1 tels que :

$$A_1 = BQ_1 + R_1, \quad \text{avec} \quad \deg(R_1) < \deg(B) \quad \text{ou} \quad R_1 = 0.$$

Alors :

$$A = \frac{a_n}{b_m} X^{n-m} B + A_1 = B \left(\frac{a_n}{b_m} X^{n-m} + Q_1 \right) + R_1,$$

ce qui donne l'existence avec $Q = \frac{a_n}{b_m} X^{n-m} + Q_1$ et $R = R_1$.

Unicité. Supposons deux décompositions :

$$A = BQ + R = BQ' + R',$$

avec $\deg(R), \deg(R') < \deg(B)$ ou $R, R' = 0$. Alors :

$$B(Q - Q') = R' - R.$$

Si $Q \neq Q'$, alors $\deg(B(Q - Q')) \geq \deg(B)$ (car $\mathbb{K}[X]$ est intègre). Mais $\deg(R' - R) < \deg(B)$ (car $\deg(R' - R) \leq \max(\deg(R), \deg(R')) < \deg(B)$). Contradiction. Donc $Q = Q'$, et par suite $R = R'$. \square

Lemme 1.26 (Nombre de racines d'un polynôme). Soit \mathbb{K} un corps et soit $f \in \mathbb{K}[X]$ un polynôme non nul de degré $d \geq 0$. Alors f admet au plus d racines dans \mathbb{K} .

Démonstration. Par récurrence sur d . Le cas $d = 0$ est immédiat (un polynôme constant non nul n'a aucune racine). Pour $d \geq 1$, si f a une racine $a \in \mathbb{K}$, alors par division euclidienne dans $\mathbb{K}[X]$, on peut écrire $f(X) = (X - a)g(X)$ avec $g \in \mathbb{K}[X]$ de degré $d - 1$.

Soit maintenant $b \neq a$ une autre racine de f . Alors $0 = f(b) = (b - a)g(b)$. Comme $b - a \neq 0$ et que \mathbb{K} est un corps (donc intègre), on en déduit $g(b) = 0$. Ainsi, toutes les racines de f distinctes de a sont des racines de g . Par hypothèse de récurrence, g a au plus $d - 1$ racines, donc f a au plus d racines. \square

Proposition 1.27 (Anneau principal des polynômes). Soit \mathbb{K} un corps. Alors $\mathbb{K}[X]$ est un anneau principal, c'est-à-dire que tout idéal de $\mathbb{K}[X]$ est engendré par un unique polynôme unitaire (le générateur unitaire de l'idéal).

Démonstration. Soit J un idéal non nul de $\mathbb{K}[X]$. Considérons l'ensemble :

$$D = \{\deg(P(X)) \mid P(X) \in J, P(X) \neq 0\}.$$

Comme J est non nul, D est une partie non vide de \mathbb{N} . Soit d_0 le plus petit élément de D , et soit $P_0(X) \in J$ non nul de degré d_0 . On peut supposer $P_0(X)$ unitaire (quitte à le normaliser). Montrons que $J = (P_0(X))$.

Soit $A(X) \in J$. Par division euclidienne, il existe $Q(X), R(X) \in \mathbb{K}[X]$ tels que :

$$A(X) = P_0(X)Q(X) + R(X) \quad \text{avec} \quad \deg(R(X)) < \deg(P_0(X)) \quad \text{ou} \quad R(X) = 0.$$

Or $R(X) = A(X) - P_0(X)Q(X) \in J$ (car $A(X), P_0(X) \in J$). Si $R(X) \neq 0$, alors $\deg(R(X)) < d_0$, ce qui contredit la minimalité de d_0 . Donc $R(X) = 0$ et $A(X) = P_0(X)Q(X) \in (P_0(X))$. Ainsi $J \subset (P_0(X))$. L'inclusion inverse est évidente car $P_0(X) \in J$. Donc $J = (P_0(X))$. \square

Définition 1.28 (Idéal maximal). Soit A un anneau commutatif unitaire. Un idéal I de A est dit *maximal* si $I \neq A$ et si pour tout idéal J de A tel que $I \subseteq J \subseteq A$, on a soit $J = I$, soit $J = A$.

Proposition 1.29 (Quotient par un idéal maximal). Soit A un anneau commutatif unitaire et I un idéal maximal de A . Alors l'anneau quotient A/I est un corps.

Démonstration. Soit $\bar{x} \neq 0$ dans A/I . Alors $x \notin I$. Considérons l'idéal $J = I + (x)$ (l'idéal engendré par I et x). Puisque I est maximal et $I \subsetneq J$ (car $x \in J$ mais $x \notin I$), on a nécessairement $J = A$.

Ainsi, il existe $i \in I$ et $a \in A$ tels que :

$$1 = i + a \cdot x$$

En passant au quotient dans A/I , on obtient :

$$\bar{1} = \bar{i} + \bar{a} \cdot \bar{x} = 0 + \bar{a} \cdot \bar{x} = \bar{a} \cdot \bar{x}$$

Donc \bar{a} est l'inverse de \bar{x} dans A/I . Ainsi, tout élément non nul de A/I est inversible, et A/I est un corps. \square

Corollaire 1.30 (Construction de corps par quotient avec un polynôme irréductible). Soit \mathbb{K} un corps commutatif et soit $m(X) \in \mathbb{K}[X]$ un polynôme irréductible. Alors l'anneau quotient $\mathbb{K}[X]/(m(X))$ est un corps.

Démonstration. Soit $I = (m(X))$ l'idéal engendré par $m(X)$.

Puisque $\mathbb{K}[X]$ est principal (Proposition 1.27), et que $m(X)$ est irréductible, l'idéal I est maximal. En effet, soit J un idéal tel que $I \subset J \subset \mathbb{K}[X]$. Alors $J = (p(X))$ pour un certain $p(X) \in \mathbb{K}[X]$. Puisque $m(X) \in J$, on a $p(X) \mid m(X)$. Mais $m(X)$ est irréductible, donc soit $p(X)$ est une constante non nulle (auquel cas $J = \mathbb{K}[X]$), soit $p(X)$ est associé à $m(X)$ (auquel cas $J = I$). Ainsi, I est maximal.

Par la Proposition 1.29, le quotient $\mathbb{K}[X]/I$ est un corps. \square

Théorème 1.31 (Théorème de Moore). Pour tout $q = p^n$ avec p premier et $n \geq 1$, il existe un corps fini de cardinal q , unique à isomorphisme près. On note alors \mathbb{F}_{p^n} n'importe quel corps fini de cardinal p^n .

Ce théorème est essentiellement là pour justifier que la notation \mathbb{F}_{p^n} (en particulier \mathbb{F}_{2^n}) n'est pas ambiguë : tous les corps finis de cardinal p^n sont isomorphes, quel que soit le modèle concret choisi.

Commentaire. Existence. En pratique, on construit un corps de cardinal p^n comme quotient $\mathbb{F}_p[X]/(m(X))$, où $m(X) \in \mathbb{F}_p[X]$ est un polynôme irréductible de degré n (on admet ici l'existence d'un tel polynôme). Par la Proposition 1.30, ce quotient est un corps. Vu comme \mathbb{F}_p -espace vectoriel de base $1, \bar{X}, \dots, \bar{X}^{n-1}$, il a donc exactement p^n éléments. C'est ce modèle concret que nous utiliserons pour la construction de \mathbb{F}_{p^n} (par exemple \mathbb{F}_{2^8} pour AES).

Unicité. Une preuve complète du fait que tout corps fini de cardinal p^n est isomorphe à tout autre repose sur quelques notions de théorie des corps que nous ne développons pas dans le corps du texte. Ces éléments sont développés en détail dans l'Annexe A, où l'on donne une preuve complète du théorème de Moore. \square

Remarque 1.32 (Construction pratique). En pratique, pour construire \mathbb{F}_{p^n} , on choisit un polynôme irréductible $m(X) \in \mathbb{F}_p[X]$ de degré n et on pose :

$$\mathbb{F}_{p^n} = \mathbb{F}_p[X]/(m(X))$$

Les éléments sont les polynômes de degré au plus $n - 1$ à coefficients dans \mathbb{F}_p , et les opérations sont effectuées modulo $m(X)$.

Cette construction est fondamentale pour de nombreux schémas cryptographiques, par exemple AES utilise $\mathbb{F}_{2^8} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$.

Exemple 1.33 (Corps \mathbb{F}_4 comme $\mathbb{F}_2[X]/(X^2 + X + 1)$). Le polynôme $X^2 + X + 1$ est irréductible sur \mathbb{F}_2 (il n'a pas de racine dans \mathbb{F}_2 : $0^2 + 0 + 1 = 1$, $1^2 + 1 + 1 = 1$). On a $\mathbb{F}_4 \simeq \mathbb{F}_2[X]/(X^2 + X + 1)$.

Les éléments de ce corps sont les polynômes de degré au plus 1 : $0, 1, X, X + 1$. Notons α la classe de X modulo $X^2 + X + 1$, donc $\alpha^2 + \alpha + 1 = 0$, soit $\alpha^2 = \alpha + 1$ (car $-1 = 1$ dans \mathbb{F}_2).

Tables d'opérations (en notant les éléments comme $0, 1, \alpha, \alpha + 1$) :

+	0	1	α	$\alpha + 1$	×	0	1	α	$\alpha + 1$
0	0	1	α	$\alpha + 1$	0	0	0	0	0
1	1	0	$\alpha + 1$	α	1	0	1	α	$\alpha + 1$
α	α	$\alpha + 1$	0	1	α	0	α	$\alpha + 1$	1
$\alpha + 1$	$\alpha + 1$	α	1	0	$\alpha + 1$	0	$\alpha + 1$	1	α

Calcul de l'inverse de α par l'algorithme d'Euclide étendu :

Dans $\mathbb{F}_2[X]$, on cherche l'inverse de X modulo $X^2 + X + 1$, c'est-à-dire un polynôme $u(X) \in \mathbb{F}_2[X]$ de degré au plus 1 tel que :

$$X \cdot u(X) \equiv 1 \pmod{X^2 + X + 1}$$

Appliquons l'algorithme d'Euclide étendu à $A(X) = X^2 + X + 1$ et $B(X) = X$:

1. Division de $A(X)$ par $B(X)$:

$$X^2 + X + 1 = X \cdot (X + 1) + 1$$

Donc $Q_1(X) = X + 1$ et $R_1(X) = 1$.

2. Division de $B(X)$ par $R_1(X)$:

$$X = 1 \cdot X + 0$$

Le pgcd est donc $1 = R_1(X)$.

3. Remontons l'algorithme pour exprimer 1 comme combinaison de $X^2 + X + 1$ et X :

$$1 = (X^2 + X + 1) - X \cdot (X + 1)$$

Ainsi, modulo $X^2 + X + 1$, on a :

$$X \cdot (X + 1) \equiv -1 \equiv 1 \pmod{X^2 + X + 1}$$

car dans \mathbb{F}_2 , $-1 = 1$.

Donc l'inverse de X modulo $X^2 + X + 1$ est $X + 1$. En termes d'éléments de \mathbb{F}_4 , cela donne $\alpha^{-1} = \alpha + 1$.

Vérification dans la table : $\alpha \times (\alpha + 1) = 1$, donc $\alpha^{-1} = \alpha + 1$.

Remarque 1.34 (À retenir sur les corps finis).

- $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ est un corps si et seulement si p est premier.
- Pour tout $a \in \mathbb{F}_p$, $a \neq 0$, il existe un unique inverse $a^{-1} \in \mathbb{F}_p$.
- Un polynôme irréductible $m(X)$ sur \mathbb{F}_2 donne un corps $\mathbb{F}_{2^m} \simeq \mathbb{F}_2[X]/(m(X))$.
- Tout corps fini de cardinal $q = p^n$ est isomorphe à $\mathbb{F}_p[X]/(m(X))$ pour un polynôme irréductible $m(X)$ de degré n .

1.5 Mots binaires et conventions d'encodage

Les primitives cryptographiques sont spécifiées en termes de mots binaires (octets, séquences d'octets) tandis que nous raisonnons souvent dans des corps finis \mathbb{F}_n . Cette section fixe les conventions d'encodage qui permettent de passer de l'un à l'autre sans ambiguïté.

Définition 1.35 (Mots binaires). Pour $t \in \mathbb{N}$, on note

$$\{0, 1\}^t$$

l'ensemble des suites (b_{t-1}, \dots, b_0) de bits, avec la convention que $\{0, 1\}^0 = \{\varepsilon\}$ contient le mot vide. On pose

$$\{0, 1\}^* = \bigcup_{t \geq 0} \{0, 1\}^t,$$

et pour $x \in \{0, 1\}^*$, on note $\|x\|$ sa longueur (en bits).

Définition 1.36 (Concaténation et XOR). La concaténation est l'application

$$\| : \{0, 1\}^a \times \{0, 1\}^b \rightarrow \{0, 1\}^{a+b}$$

qui à (x, y) associe la suite obtenue en juxtaposant x et y .

Le XOR est l'application

$$\oplus : \{0, 1\}^t \times \{0, 1\}^t \rightarrow \{0, 1\}^t$$

définie par $(x, y) \mapsto (x_0 \oplus y_0, \dots, x_{t-1} \oplus y_{t-1})$, où \oplus désigne l'addition dans \mathbb{F}_2 .

Définition 1.37 (Encodage big-endian). Pour $t \in \mathbb{N}$ et $b = (b_{t-1}, \dots, b_0) \in \{0, 1\}^t$, on définit

$$\text{val}_{\text{be}}(b) = \sum_{j=0}^{t-1} b_j 2^{t-1-j} \in \{0, \dots, 2^t - 1\}.$$

Cet encodage fournit une bijection canonique entre $\{0, 1\}^t$ et $\{0, \dots, 2^t - 1\}$.

Définition 1.38 (Encodage little-endian). Pour $t \in \mathbb{N}$ et $b = (b_{t-1}, \dots, b_0) \in \{0, 1\}^t$, on définit

$$\text{val}_{\text{le}}(b) = \sum_{j=0}^{t-1} b_j 2^j \in \{0, \dots, 2^t - 1\}.$$

Cet encodage est utilisé dans les standards Ed25519.

Remarque 1.39 (Convention d'écriture). Dans la suite du document :

- L'encodage big-endian est utilisé pour :
 - Le partage de Shamir (représentation des éléments de \mathbb{F}_p)
 - Les représentations internes dans \mathbb{F}_p pour RSA
 - Les compteurs AES-GCM
- L'encodage little-endian est utilisé pour :
 - Ed25519 (scalaires et points)
 - Certaines opérations internes d'AES
- Le contexte déterminera clairement quel encodage est employé

2 Secret maître et schéma de Shamir

2.1 Secret maître

Définition 2.1 (Secret maître). On fixe un entier

$$P = 2^{521} - 1$$

et le corps \mathbb{F}_P . Le secret maître est un mot binaire

$$S \in \{0, 1\}^{256}$$

tiré uniformément. On définit

$$s_0 = \text{val}_{\text{be}}(S) \in \{0, \dots, 2^{256} - 1\}, \quad s = s_0 \bmod P \in \mathbb{F}_P.$$

Comme $P > 2^{256} - 1$, on a $s = s_0$ dans \mathbb{F}_P , ce qui garantit que l'application $S \mapsto s$ est une injection de $\{0, 1\}^{256}$ dans \mathbb{F}_P .

Ainsi, S est la représentation binaire utilisée pour la dérivation de clés, et $s \in \mathbb{F}_P$ est la représentation dans le corps fini utilisée pour le partage de secret.

2.2 Schéma (k, n) de Shamir

Définition 2.2 (Partage de Shamir). Soient des entiers n, k vérifiant $2 \leq k \leq n$ et $n < P$. On définit le schéma (k, n) de Shamir sur \mathbb{F}_P comme suit.

- À partir de $s \in \mathbb{F}_P$, on choisit indépendamment et uniformément $a_1, \dots, a_{k-1} \in \mathbb{F}_P$.
- On définit

$$f(X) = s + a_1X + a_2X^2 + \dots + a_{k-1}X^{k-1} \in \mathcal{P}_{<k}.$$

- Pour $i \in \{1, \dots, n\}$, on fixe $x_i = i \in \mathbb{F}_P$ (en identifiant l'entier i à son image dans \mathbb{F}_P) et on pose

$$y_i = f(x_i) \in \mathbb{F}_P.$$

- La part i -ème est le couple $(x_i, y_i) \in \mathbb{F}_P^2$.

Définition 2.3 (Algorithmes Share et Reconstruct). On définit formellement les applications :

- Share : $\mathbb{F}_P \rightarrow (\mathbb{F}_P^2)^n$ qui à s associe la famille $((x_i, y_i))_{1 \leq i \leq n}$.
- Reconstruct : $(\mathbb{F}_P^2)^k \rightarrow \mathbb{F}_P$ qui, à k points distincts (x_{i_j}, y_{i_j}) , associe $s = f(0)$ obtenu par interpolation de Lagrange.

Théorème 2.4 (Unicité de l'interpolation). Soient $x_1, \dots, x_k \in \mathbb{F}_P$ deux à deux distincts et $y_1, \dots, y_k \in \mathbb{F}_P$. Il existe un unique $f \in \mathcal{P}_{<k}$ tel que $f(x_j) = y_j$ pour tout $j \in \{1, \dots, k\}$.

Démonstration. Par le lemme 1.26 sur le nombre de racines, deux polynômes de degré $< k$ coïncidant en k points distincts sont égaux. L'existence s'obtient par la formule d'interpolation de Lagrange :

$$f(X) = \sum_{j=1}^k y_j \ell_j(X), \quad \ell_j(X) = \prod_{\substack{m=1 \\ m \neq j}}^k \frac{X - x_m}{x_j - x_m} \in \mathbb{F}_P[X].$$

Les dénominateurs $x_j - x_m$ sont inversibles dans \mathbb{F}_P car les x_j sont distincts. On vérifie $\ell_j(x_i) = \delta_{ij}$, d'où $f(x_j) = y_j$. \square

Propriété 2.5 (Reconstruction du secret). Soit $I = \{i_1, \dots, i_k\}$ avec $i_1 < \dots < i_k$. Soit f le polynôme défini par le schéma de Shamir. Alors

$$s = f(0) = \sum_{j=1}^k y_{i_j} \lambda_j, \quad \lambda_j = \ell_j(0) = \prod_{\substack{m=1 \\ m \neq j}}^k \frac{-x_{i_m}}{x_{i_j} - x_{i_m}} \in \mathbb{F}_P.$$

Démonstration. On applique le théorème 2.4 avec $(x_j, y_j) = (x_{i_j}, y_{i_j})$. On obtient

$$f(X) = \sum_{j=1}^k y_{i_j} \ell_j(X).$$

Comme $f(X) = s + a_1X + \dots + a_{k-1}X^{k-1}$, on a $f(0) = s$. En évaluant en $X = 0$, on obtient

$$s = f(0) = \sum_{j=1}^k y_{i_j} \ell_j(0),$$

avec $\lambda_j = \ell_j(0)$. \square

2.3 Confidentialité parfaite du schéma de Shamir

Proposition 2.6. Soit $t < k$ et $I = \{i_1, \dots, i_t\} \subset \{1, \dots, n\}$. On suppose que (s, a_1, \dots, a_{k-1}) est uniforme dans \mathbb{F}_P^k . Alors, pour toute réalisation fixée des parts $((x_{i_j}, y_{i_j}))_{1 \leq j \leq t}$ et pour tous $s_0, s_1 \in \mathbb{F}_P$,

$$\mathbb{P}[s = s_0 \mid (x_{i_j}, y_{i_j})] = \mathbb{P}[s = s_1 \mid (x_{i_j}, y_{i_j})].$$

En particulier, la loi a posteriori de s conditionnellement à $t < k$ parts reste uniforme sur \mathbb{F}_P .

Démonstration. Les t équations $f(x_{i_j}) = y_{i_j}$ forment un système linéaire de rang t (la matrice de Vandermonde partielle est de rang plein car les x_{i_j} sont distincts). Ce système impose t contraintes indépendantes sur les k variables (s, a_1, \dots, a_{k-1}) .

Pour toute valeur fixée $s_0 \in \mathbb{F}_P$, le système restreint aux (a_1, \dots, a_{k-1}) a t équations indépendantes sur $k - 1$ variables, donc admet exactement P^{k-1-t} solutions. Ce nombre est indépendant de s_0 .

Par Bayes et l'uniformité a priori sur \mathbb{F}_P^k :

$$\mathbb{P}[s = s_0 \mid \text{parts}] = \frac{\mathbb{P}[\text{parts} \mid s = s_0] \cdot \mathbb{P}[s = s_0]}{\mathbb{P}[\text{parts}]} = \frac{P^{-t} \cdot P^{-1}}{P^{-t}} = \frac{1}{P}.$$

Ainsi, la loi conditionnelle reste uniforme sur \mathbb{F}_P . \square

3 Fonctions de hachage, HMAC, HKDF, PBKDF2

3.1 Fonction de hachage

Définition 3.1 (Fonction de hachage cryptographique). Une fonction de hachage cryptographique est une application

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^d,$$

pour un d fixé, qui doit satisfaire les propriétés suivantes :

- **Résistance à la préimage** : Étant donné $y \in \{0, 1\}^d$, il est difficile de trouver x tel que $h(x) = y$.
- **Résistance à la seconde préimage** : Étant donné x , il est difficile de trouver $x' \neq x$ tel que $h(x) = h(x')$.
- **Résistance aux collisions** : Il est difficile de trouver $x \neq x'$ tels que $h(x) = h(x')$.

Dans ce document, h désigne une telle fonction fixée une fois pour toutes (typiquement SHA-256 avec $d = 256$).

3.2 Fonction de hachage cryptographique SHA-256

Définition 3.2 (SHA-256 selon FIPS 180-4 [9]). SHA-256 est une fonction de hachage cryptographique de la famille SHA-2 qui produit une empreinte de 256 bits. Elle opère sur des blocs de 512 bits en utilisant une structure de Merkle-Damgård avec renforcement de Merkle.

Soit $M \in \{0, 1\}^*$ un message de longueur L bits. L'algorithme procède comme suit :

1. **Préparation du message** : Ajout d'un bit 1, de k bits 0, et de la longueur L encodée sur 64 bits (big-endian), où k est le plus petit entier non négatif tel que la longueur totale soit un multiple de 512.
2. **Découpage** : Division du message préparé en N blocs de 512 bits : $M^{(1)}, M^{(2)}, \dots, M^{(N)}$.
3. **Initialisation** : Définition de 8 variables de 32 bits initialisées avec des constantes spécifiques (les racines carrées des 8 premiers nombres premiers).
4. **Boucle de compression** : Application itérative d'une fonction de compression C à chaque bloc, mettant à jour les 8 variables.
5. **Sortie** : Concaténation des 8 variables finales (chacune de 32 bits) pour former l'empreinte de 256 bits.

Préparation du message et découpage

Définition 3.3 (Padding SHA-256). Soit $M \in \{0, 1\}^*$ un message de longueur L bits. On construit le message préparé M' comme suit :

$$M' = M \parallel 1 \parallel 0^k \parallel \text{len}_{64}(L)$$

où :

- 1 est un bit unique de valeur 1,
- 0^k est une suite de k bits 0, avec k choisi tel que $L + 1 + k + 64 \equiv 0 \pmod{512}$,

— $\text{len}_{64}(L)$ est la représentation binaire de L sur 64 bits en convention *big-endian*.

Le message M' a ainsi une longueur multiple de 512 bits.

Définition 3.4 (Découpage en blocs de 512 bits). Le message préparé M' est divisé en N blocs de 512 bits :

$$M' = M^{(1)} \parallel M^{(2)} \parallel \dots \parallel M^{(N)}.$$

Chaque bloc $M^{(i)}$ est interprété comme 16 mots de 32 bits $M_0^{(i)}, M_1^{(i)}, \dots, M_{15}^{(i)}$ en lecture *big-endian* :

$$M^{(i)} = M_0^{(i)} \parallel M_1^{(i)} \parallel \dots \parallel M_{15}^{(i)}.$$

Constantes et fonctions internes

Définition 3.5 (Constantes initiales SHA-256). Les 8 variables initiales (registres) sont définies par les constantes suivantes (en hexadécimal) :

$$\begin{array}{ll} H_0^{(0)} = 6a09e667 & H_1^{(0)} = bb67ae85 \\ H_2^{(0)} = 3c6ef372 & H_3^{(0)} = a54ff53a \\ H_4^{(0)} = 510e527f & H_5^{(0)} = 9b05688c \\ H_6^{(0)} = 1f83d9ab & H_7^{(0)} = 5be0cd19 \end{array}$$

Ces valeurs correspondent aux 32 bits de poids fort des parties fractionnaires des racines carrées des 8 premiers nombres premiers.

Définition 3.6 (Constantes de tour SHA-256). Chaque tour t (avec $0 \leq t \leq 63$) utilise une constante K_t de 32 bits, définie comme les 32 bits de poids fort des parties fractionnaires des racines cubiques des 64 premiers nombres premiers. Les premières constantes sont :

$$\begin{array}{llll} K_0 = 428a2f98 & K_1 = 71374491 & K_2 = b5c0fbcf & K_3 = e9b5dba5 \\ K_4 = 3956c25b & K_5 = 59f111f1 & K_6 = 923f82a4 & K_7 = ab1c5ed5 \\ K_8 = d807aa98 & K_9 = 12835b01 & K_{10} = 243185be & K_{11} = 550c7dc3 \\ K_{12} = 72be5d74 & K_{13} = 80deb1fe & K_{14} = 9bdc06a7 & K_{15} = c19bf174 \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

Définition 3.7 (Fonctions internes SHA-256). Pour des mots de 32 bits x, y, z , on définit les fonctions suivantes :

$$\begin{aligned} \text{Ch}(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) && \text{(choix)} \\ \text{Maj}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) && \text{(majorité)} \\ \Sigma_0(x) &= \text{ROTR}^2(x) \oplus \text{ROTR}^{13}(x) \oplus \text{ROTR}^{22}(x) \\ \Sigma_1(x) &= \text{ROTR}^6(x) \oplus \text{ROTR}^{11}(x) \oplus \text{ROTR}^{25}(x) \\ \sigma_0(x) &= \text{ROTR}^7(x) \oplus \text{ROTR}^{18}(x) \oplus \text{SHR}^3(x) \\ \sigma_1(x) &= \text{ROTR}^{17}(x) \oplus \text{ROTR}^{19}(x) \oplus \text{SHR}^{10}(x) \end{aligned}$$

où ROTR^n désigne la rotation vers la droite de n bits, SHR^n le décalage vers la droite de n bits (remplissage par des zéros), \wedge le ET bit à bit, \oplus le XOR, et \neg la négation bit à bit.

Planning des mots

Définition 3.8 (Planning des mots). Pour chaque bloc $M^{(i)}$, on construit 64 mots de 32 bits W_0, \dots, W_{63} appelés planning des mots. Les 16 premiers mots sont les 16 mots du bloc courant :

$$W_t = M_t^{(i)} \quad \text{pour } 0 \leq t \leq 15.$$

Les 48 mots suivants sont calculés récursivement pour $16 \leq t \leq 63$:

$$W_t = \sigma_1(W_{t-2}) \boxplus W_{t-7} \boxplus \sigma_0(W_{t-15}) \boxplus W_{t-16},$$

où \boxplus désigne l'addition modulo 2^{32} .

Fonction de compression

Définition 3.9 (Fonction de compression SHA-256). La fonction de compression C prend en entrée un état de 256 bits (représenté par 8 mots de 32 bits $H^{(i-1)} = (H_0^{(i-1)}, \dots, H_7^{(i-1)})$) et un bloc de message $M^{(i)}$ de 512 bits, et produit un nouvel état $H^{(i)}$ de 256 bits.

Algorithme de la fonction de compression :

1. Préparer le planning des mots W_0, \dots, W_{63} à partir de $M^{(i)}$.
2. Initialiser 8 variables de travail (a, b, c, d, e, f, g, h) avec l'état courant :

$$\begin{aligned} a &\leftarrow H_0^{(i-1)}, & b &\leftarrow H_1^{(i-1)}, & c &\leftarrow H_2^{(i-1)}, & d &\leftarrow H_3^{(i-1)}, \\ e &\leftarrow H_4^{(i-1)}, & f &\leftarrow H_5^{(i-1)}, & g &\leftarrow H_6^{(i-1)}, & h &\leftarrow H_7^{(i-1)}. \end{aligned}$$

3. Pour $t = 0$ à 63 :

$$\begin{aligned} T_1 &\leftarrow h \boxplus \Sigma_1(e) \boxplus \text{Ch}(e, f, g) \boxplus K_t \boxplus W_t, \\ T_2 &\leftarrow \Sigma_0(a) \boxplus \text{Maj}(a, b, c), \\ h &\leftarrow g, & g &\leftarrow f, & f &\leftarrow e, & e &\leftarrow d \boxplus T_1, \\ d &\leftarrow c, & c &\leftarrow b, & b &\leftarrow a, & a &\leftarrow T_1 \boxplus T_2. \end{aligned}$$

4. Mettre à jour l'état par addition modulo 2^{32} :

$$\begin{aligned} H_0^{(i)} &\leftarrow H_0^{(i-1)} \boxplus a, & H_1^{(i)} &\leftarrow H_1^{(i-1)} \boxplus b, \\ H_2^{(i)} &\leftarrow H_2^{(i-1)} \boxplus c, & H_3^{(i)} &\leftarrow H_3^{(i-1)} \boxplus d, \\ H_4^{(i)} &\leftarrow H_4^{(i-1)} \boxplus e, & H_5^{(i)} &\leftarrow H_5^{(i-1)} \boxplus f, \\ H_6^{(i)} &\leftarrow H_6^{(i-1)} \boxplus g, & H_7^{(i)} &\leftarrow H_7^{(i-1)} \boxplus h. \end{aligned}$$

Algorithme complet et sortie

Définition 3.10 (Algorithme SHA-256). Les étapes sont :

1. **Préparation** : Construire M' à partir du message M en appliquant le padding.
2. **Découpage** : Diviser M' en N blocs de 512 bits $M^{(1)}, \dots, M^{(N)}$.
3. **Initialisation** : Initialiser l'état $H^{(0)}$ avec les 8 constantes initiales.
4. **Boucle sur les blocs** : Pour $i = 1$ à N , appliquer la fonction de compression C :

$$H^{(i)} = C(H^{(i-1)}, M^{(i)}).$$

5. **Sortie** : L'empreinte finale de 256 bits est la concaténation des 8 mots de 32 bits de l'état final $H^{(N)}$ en ordre *big-endian* :

$$\text{SHA-256}(M) = H_0^{(N)} \parallel H_1^{(N)} \parallel H_2^{(N)} \parallel H_3^{(N)} \parallel H_4^{(N)} \parallel H_5^{(N)} \parallel H_6^{(N)} \parallel H_7^{(N)}.$$

3.3 HMAC

Définition 3.11 (Longueur de bloc et masques internes). On fixe un entier $k_{\text{blk}} \in \mathbb{N}^*$, appelé longueur de bloc de HMAC. On se donne deux mots binaires

$$\text{ipad}, \text{opad} \in \{0, 1\}^{k_{\text{blk}}}$$

appelés respectivement masque interne et masque externe. Dans les spécifications usuelles [3], ce sont les octets $0x36$ (pour **ipad**) et $0x5C$ (pour **opad**) répétés de façon à obtenir k_{blk} bits.

Définition 3.12 (Normalisation de clé pour HMAC). On définit une application

$$\text{NormKey} : \{0, 1\}^* \rightarrow \{0, 1\}^{k_{\text{blk}}}$$

qui, pour toute clé $K \in \{0, 1\}^*$,

- si $\|K\| > k_{\text{blk}}$, alors $K_{\text{blk}} = \text{troncature}(h(K))$ aux k_{blk} premiers bits ;
- si $\|K\| < k_{\text{blk}}$, alors $K_{\text{blk}} = K \parallel 0^{k_{\text{blk}} - \|K\|}$;
- si $\|K\| = k_{\text{blk}}$, alors $K_{\text{blk}} = K$.

Ainsi, pour tout $K \in \{0, 1\}^*$, la valeur $\text{NormKey}(K)$ est un mot binaire de longueur exactement k_{blk} .

Définition 3.13 (HMAC basé sur h). On fixe une fonction de hachage

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^d.$$

On définit la fonction

$$\text{HMAC}_h : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^d$$

par la formule :

$$\text{HMAC}_h(K, M) = h((\text{NormKey}(K) \oplus \text{opad}) \parallel h((\text{NormKey}(K) \oplus \text{ipad}) \parallel M)).$$

3.4 Fonction d'encodage d'entier

Définition 3.14 (Fonction INT_4). On définit l'application

$$\text{INT}_4 : \{0, \dots, 2^{32} - 1\} \rightarrow \{0, 1\}^{32}$$

qui à un entier i associe son encodage big-endian sur 4 octets (32 bits) :

$$\text{INT}_4(i) = (b_{31}b_{30} \dots b_0) \text{ où } i = \sum_{j=0}^{31} b_j 2^{31-j}.$$

3.5 HKDF

Définition 3.15 (HKDF (HMAC-based Key Derivation Function) [4]). Soit $h : \{0, 1\}^* \rightarrow \{0, 1\}^d$ une fonction de hachage. On définit la fonction de dérivation de clé HKDF selon RFC 5869 :

$$\text{HKDF}_h : \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{N} \rightarrow \{0, 1\}^L$$

comme suit. Soient $\text{IKM} \in \{0, 1\}^*$ (input keying material), $\text{salt} \in \{0, 1\}^*$, $\text{info} \in \{0, 1\}^*$, et $L \in \mathbb{N}$ (longueur de sortie en octets). On pose :

- $\text{PRK} = \text{HMAC}_h(\text{salt}, \text{IKM})$ (extraction)
- On définit itérativement pour $i = 1, 2, \dots$ jusqu'à obtenir L octets :

$$\begin{aligned} T_1 &= \text{HMAC}_h(\text{PRK}, \text{info} \parallel \text{INT}_4(1)) \\ T_2 &= \text{HMAC}_h(\text{PRK}, T_1 \parallel \text{info} \parallel \text{INT}_4(2)) \\ &\vdots \\ T_i &= \text{HMAC}_h(\text{PRK}, T_{i-1} \parallel \text{info} \parallel \text{INT}_4(i)) \end{aligned}$$

- La sortie est la troncature aux L octets de $T_1 \parallel T_2 \parallel \dots$

3.6 PBKDF2

Définition 3.16 (PBKDF2 (Password-Based Key Derivation Function 2) [5]). On fixe un paramètre d'itération $c \in \mathbb{N}^*$ et un hachage h . On définit

$$\text{PBKDF2}_h : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{N}^* \times \mathbb{N} \rightarrow \{0, 1\}^{8L}$$

où L est le nombre d'octets souhaités. Pour un mot de passe $P \in \{0, 1\}^*$, un sel $S \in \{0, 1\}^*$ et un entier $L \geq 1$, on définit, pour $i \geq 1$:

$$\begin{aligned} U_1 &= \text{HMAC}_h(P, S \parallel \text{INT}_4(i)), \\ U_j &= \text{HMAC}_h(P, U_{j-1}) \quad (2 \leq j \leq c), \end{aligned}$$

puis

$$F(P, S, c, i) = U_1 \oplus U_2 \oplus \dots \oplus U_c.$$

La sortie de PBKDF2 est le préfixe aux $8L$ bits de la concaténation

$$F(P, S, c, 1) \parallel F(P, S, c, 2) \parallel \dots$$

4 Structure de groupe et signature Ed25519

4.1 Courbe elliptique Ed25519

Définition 4.1 (Courbe elliptique Ed25519 (Edwards-curve Digital Signature Algorithm) [6]). Soit $q = 2^{255} - 19$ et \mathbb{F}_q le corps fini correspondant. La courbe elliptique Ed25519 est définie par l'équation de Twisted Edwards :

$$E : -x^2 + y^2 = 1 + dx^2y^2 \quad \text{sur } \mathbb{F}_q$$

où $d = -\frac{121665}{121666} \in \mathbb{F}_q$, avec la division interprétée comme la multiplication par l'inverse modulo q .

Définition 4.2 (Loi de groupe sur $E(\mathbb{F}_q)$). Soient $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ deux points de $E(\mathbb{F}_q)$. L'addition est définie par :

$$P_1 + P_2 = (x_3, y_3) = \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 + x_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

où toutes les opérations arithmétiques sont effectuées dans \mathbb{F}_q . L'élément neutre est le point $\mathcal{O} = (0, 1)$.

Proposition 4.3 (Cardinal du groupe). Le groupe $E(\mathbb{F}_q)$ est abélien fini de cardinal :

$$\#E(\mathbb{F}_q) = 8\ell$$

où ℓ est le nombre premier $\ell = 2^{252} + 27742317777372353535851937790883648493$.

Remarque 4.4 (Admission du cardinal). Le calcul du cardinal de la courbe Ed25519 est admis. Il peut être obtenu par l'algorithme de Schoof-Elkies-Atkin (SEA) et a été vérifié de manière indépendante par plusieurs implémentations.

Définition 4.5 (Point de base standard Ed25519). On fixe le point de base standard $B \in E(\mathbb{F}_q)$ d'ordre ℓ , dont les coordonnées affines sont :

$$y_B = 46316835694926478169428394003475163141307993866256225615783033603165251855960$$

$$x_B = 15112221349535400772501151409588531511454012693041857206046113283949847762202$$

Ces valeurs satisfont l'équation de la courbe.

Remarque 4.6 (Admission de l'ordre du point de base). L'ordre premier ℓ du point de base B est admis. Cette propriété essentielle pour la sécurité cryptographique a été vérifiée par la communauté.

Définition 4.7 (Sous-groupe cyclique principal). Le sous-groupe cyclique d'ordre ℓ est :

$$\langle B \rangle = \{aB \mid a \in \mathbb{Z}_\ell\}$$

où aB désigne l'addition B avec elle-même a fois (avec la convention $0B = \mathcal{O}$).

Proposition 4.8 (Isomorphisme $\mathbb{Z}_\ell \simeq \langle B \rangle$). L'application

$$\psi : \mathbb{Z}_\ell \rightarrow \langle B \rangle, \quad a \mapsto aB$$

est un isomorphisme de groupes. Plus précisément :

1. ψ est un homomorphisme de groupes.
2. $\ker(\psi) = \{0\}$ (injectivité).
3. ψ est surjectif par définition de $\langle B \rangle$.

Démonstration. Vérifions les propriétés :

1. **Homomorphisme** : Pour tous $a, b \in \mathbb{Z}_\ell$, on a

$$\psi(a + b) = (a + b)B = aB + bB = \psi(a) + \psi(b),$$

où l'addition dans \mathbb{Z}_ℓ est modulo ℓ et l'addition dans $\langle B \rangle$ est l'addition de points sur la courbe elliptique.

2. **Noyau** : Soit $a \in \mathbb{Z}_\ell$ tel que $\psi(a) = \mathcal{O}$ (élément neutre). Alors $aB = \mathcal{O}$. Puisque B est d'ordre ℓ , cela implique que ℓ divise a . Comme $a \in \{0, \dots, \ell - 1\}$, on a nécessairement $a = 0$. Donc $\ker(\psi) = \{0\}$.
3. **Surjectivité** : Par définition, $\langle B \rangle = \{aB \mid a \in \mathbb{Z}_\ell\}$, donc tout élément de $\langle B \rangle$ est de la forme aB pour un certain $a \in \mathbb{Z}_\ell$, c'est-à-dire $\psi(a)$.

Ainsi, ψ est un isomorphisme de groupes. □

Remarque 4.9 (Interprétation du premier théorème d'isomorphisme). Le premier théorème d'isomorphisme pour les groupes (Théorème 1.16) permet d'interpréter cet isomorphisme. Pour $G = \mathbb{Z}_\ell$, $H = \langle B \rangle$, et $f = \psi$, on a :

- $\ker(\psi) = \{0\}$ qui est un sous-groupe distingué de \mathbb{Z}_ℓ
- $\text{Im}(\psi) = \langle B \rangle$
- Par le théorème : $\mathbb{Z}_\ell / \ker(\psi) \simeq \text{Im}(\psi)$, c'est-à-dire $\mathbb{Z}_\ell / \{0\} \simeq \langle B \rangle$.

Comme le quotient $\mathbb{Z}_\ell / \{0\}$ est canoniquement isomorphe à \mathbb{Z}_ℓ lui-même, on retrouve bien l'isomorphisme $\mathbb{Z}_\ell \simeq \langle B \rangle$.

Remarque 4.10 (Utilisation cryptographique). Seuls les points du sous-groupe $\langle B \rangle$ d'ordre premier ℓ sont utilisés pour la cryptographie. Le clamping dans Ed25519 garantit que les scalaires appartiennent à \mathbb{Z}_ℓ , et l'isomorphisme $\mathbb{Z}_\ell \simeq \langle B \rangle$ assure que chaque scalaire correspond à un unique point du sous-groupe, et réciproquement.

4.2 Encodage des points et scalaires

Définition 4.11 (Encodage des points et scalaires). L'encodage canonique d'un point $P = (x, y) \in E(\mathbb{F}_q)$ est défini par :

$$\text{enc}(P) = \text{bytes}_{\text{le}}(y) \parallel p \in \{0, 1\}^{256}$$

où $\text{bytes}_{\text{le}}(y)$ est la représentation little-endian de y sur 255 bits (32 octets, le bit de poids fort ignoré), et p est le bit de parité de x , c'est-à-dire le bit le moins significatif de x .

L'encodage d'un scalaire $s \in \mathbb{Z}_\ell$ est sa représentation little-endian sur 32 octets.

Remarque 4.12 (Encodage compressé). L'encodage utilisé pour Ed25519 est l'encodage compressé standard où seul le coordonnée y est stockée avec un bit de parité pour permettre la reconstruction de x . L'encodage est en little-endian conformément au standard [6].

4.3 Décodage des points Ed25519

Définition 4.13 (Symbole de Legendre). Soit p un nombre premier impair et a un entier. Le symbole de Legendre $\left(\frac{a}{p}\right)$ est défini par :

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{si } p \mid a \\ 1 & \text{si } a \text{ est un carré modulo } p \\ -1 & \text{sinon} \end{cases}$$

Théorème 4.14 (Critère d'Euler). Soit p un nombre premier impair et a un entier non divisible par p . Alors :

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}$$

En particulier, $a^{(p-1)/2} \equiv 1 \pmod{p}$ si et seulement si a est un carré modulo p , et $a^{(p-1)/2} \equiv -1 \pmod{p}$ si et seulement si a n'est pas un carré modulo p .

Démonstration. Si a est un carré modulo p , alors $a \equiv b^2 \pmod{p}$ pour un b non divisible par p , donc $a^{(p-1)/2} \equiv b^{p-1} \equiv 1 \pmod{p}$ par le théorème de Fermat.

Réciproquement, dans \mathbb{F}_p^* qui est cyclique d'ordre $p-1$, considérons l'homomorphisme $\phi : x \mapsto x^2$. Son noyau est $\{\pm 1\}$ d'ordre 2, donc son image (les carrés) est d'ordre $(p-1)/2$. L'équation $x^{(p-1)/2} = 1$ a au plus $(p-1)/2$ solutions, et tous les carrés la vérifient. Donc les non-carrés vérifient $a^{(p-1)/2} = -1$. \square

Lemme 4.15 (Lemme de Gauss pour les résidus quadratiques). Soit p un nombre premier impair et a un entier non divisible par p . Considérons l'ensemble :

$$A = \{a, 2a, 3a, \dots, \frac{p-1}{2}a\}$$

Pour chaque élément $ka \in A$, on considère son *résidu modulo p* , c'est-à-dire l'unique entier r_k tel que $1 \leq r_k \leq p-1$ et $r_k \equiv ka \pmod{p}$.

On sépare ces résidus en deux groupes :

- Les résidus r_1, \dots, r_k qui sont inférieurs ou égaux à $p/2$
- Les résidus s_1, \dots, s_μ qui sont supérieurs à $p/2$

où k est le nombre de résidus dans le premier groupe et μ est le nombre de résidus dans le second groupe. Alors :

$$\left(\frac{a}{p}\right) = (-1)^\mu$$

Démonstration. Étape 1 : Les nombres $r_1, \dots, r_k, p-s_1, \dots, p-s_\mu$ sont distincts

Les r_i sont par définition dans $\{1, \dots, \lfloor p/2 \rfloor\}$. Pour les s_j , comme $s_j > p/2$, on a $p-s_j \in \{1, \dots, \lfloor p/2 \rfloor\}$.

Supposons par l'absurde qu'il existe i, j tels que $r_i = p-s_j$. Alors :

$$r_i + s_j = p$$

Mais $r_i \equiv \alpha a \pmod{p}$ et $s_j \equiv \beta a \pmod{p}$ pour certains $\alpha, \beta \in \{1, \dots, \frac{p-1}{2}\}$. Donc :

$$\alpha a + \beta a \equiv 0 \pmod{p} \Rightarrow (\alpha + \beta)a \equiv 0 \pmod{p}$$

Puisque $p \nmid a$, on doit avoir $p \mid (\alpha + \beta)$. Mais $2 \leq \alpha + \beta \leq p-1$, contradiction.

Ainsi, ces $k + \mu = \frac{p-1}{2}$ nombres forment une permutation de $\{1, 2, \dots, \frac{p-1}{2}\}$.

Étape 2 : Relations produits

Le produit de tous les éléments de $\{1, 2, \dots, \frac{p-1}{2}\}$ est $(\frac{p-1}{2})!$. Donc :

$$r_1 \cdots r_k (p-s_1) \cdots (p-s_\mu) = \left(\frac{p-1}{2}\right)!$$

Modulo p , on a $p-s_j \equiv -s_j \pmod{p}$, donc :

$$r_1 \cdots r_k (p-s_1) \cdots (p-s_\mu) \equiv r_1 \cdots r_k (-s_1) \cdots (-s_\mu) = (-1)^\mu r_1 \cdots r_k s_1 \cdots s_\mu \pmod{p}$$

Ainsi :

$$(-1)^\mu r_1 \cdots r_k s_1 \cdots s_\mu \equiv \left(\frac{p-1}{2}\right)! \pmod{p} \tag{1}$$

D'autre part, le produit des éléments de A est :

$$a \cdot 2a \cdot 3a \cdots \frac{p-1}{2}a = a^{(p-1)/2} \left(\frac{p-1}{2} \right)!$$

Modulo p , ce produit est congru au produit des résidus des éléments de A , c'est-à-dire $r_1 \cdots r_k s_1 \cdots s_\mu$. Donc :

$$a^{(p-1)/2} \left(\frac{p-1}{2} \right)! \equiv r_1 \cdots r_k s_1 \cdots s_\mu \pmod{p} \quad (2)$$

Étape 3 : Conclusion

En substituant (2) dans (1), on obtient :

$$(-1)^\mu a^{(p-1)/2} \left(\frac{p-1}{2} \right)! \equiv \left(\frac{p-1}{2} \right)! \pmod{p}$$

Puisque $\left(\frac{p-1}{2} \right)!$ n'est pas divisible par p , on peut simplifier :

$$(-1)^\mu a^{(p-1)/2} \equiv 1 \pmod{p} \quad \Rightarrow \quad a^{(p-1)/2} \equiv (-1)^\mu \pmod{p}$$

Par le critère d'Euler (Théorème 4.14), on a $a^{(p-1)/2} \equiv \left(\frac{a}{p} \right) \pmod{p}$, donc :

$$\left(\frac{a}{p} \right) = (-1)^\mu$$

□

Lemme 4.16 (Caractère quadratique de 2). Soit q un nombre premier impair. Alors :

$$\left(\frac{2}{q} \right) = (-1)^{\frac{q^2-1}{8}}$$

En particulier, pour $q \equiv 5 \pmod{8}$, on a $\left(\frac{2}{q} \right) = -1$.

Démonstration. Nous appliquons le lemme 4.15 avec $a = 2$. Soit

$$A = \{2, 4, 6, \dots, 2 \cdot \frac{q-1}{2}\} = \{2, 4, \dots, q-1\}$$

Soit μ le nombre d'éléments de A dont le reste modulo q est supérieur à $q/2$. Alors par le lemme de Gauss :

$$\left(\frac{2}{q} \right) = (-1)^\mu$$

Les éléments de A sont tous dans l'intervalle $[2, q-1]$. Un élément $2k$ (avec $1 \leq k \leq \frac{q-1}{2}$) a un reste modulo q supérieur à $q/2$ si et seulement si $2k > q/2$, c'est-à-dire $k > q/4$.

Le nombre de tels k est donc :

$$\mu = \left\lfloor \frac{q-1}{2} \right\rfloor - \left\lfloor \frac{q}{4} \right\rfloor$$

Écrivons $q = 8m + r$ avec $r \in \{1, 3, 5, 7\}$. Alors :

$$\frac{q^2 - 1}{8} = \frac{(8m + r)^2 - 1}{8} = 8m^2 + 2mr + \frac{r^2 - 1}{8}$$

D'autre part, calculons μ pour chaque cas :

- Si $q = 8m + 1$, alors $\mu = 4m - 2m = 2m$ (pair)
- Si $q = 8m + 3$, alors $\mu = 4m + 1 - 2m = 2m + 1$ (impair)
- Si $q = 8m + 5$, alors $\mu = 4m + 2 - (2m + 1) = 2m + 1$ (impair)
- Si $q = 8m + 7$, alors $\mu = 4m + 3 - (2m + 1) = 2m + 2$ (pair)

Donc $\left(\frac{2}{q}\right) = (-1)^\mu = (-1)^{\frac{q^2-1}{8}}$.

Pour $q \equiv 5 \pmod{8}$, on a $q = 8m + 5$, donc $\frac{q^2-1}{8} = 8m^2 + 10m + 3$ qui est impair, donc $\left(\frac{2}{q}\right) = -1$. \square

Théorème 4.17 (Structure des racines carrées dans \mathbb{F}_q pour $q \equiv 5 \pmod{8}$). Soit $q = 2^{255} - 19$ et $a \in \mathbb{F}_q$ un élément non nul. Si a est un carré dans \mathbb{F}_q , alors l'ensemble de ses racines carrées est $\{x, -x\}$ où :

$$x = a^{(q+3)/8} \quad \text{ou} \quad x = a^{(q+3)/8} \cdot 2^{(q-1)/4}$$

De plus, ces deux racines carrées ont des bits de parité opposés.

Démonstration. Soit $q = 2^{255} - 19 \equiv 5 \pmod{8}$. On peut écrire $q = 8k + 5$ avec $k \in \mathbb{Z}$, donc $(q + 3)/8 = k + 1$ est un entier.

Supposons que a est un carré dans \mathbb{F}_q . Calculons $x_0 = a^{(q+3)/8}$:

$$x_0^2 = (a^{(q+3)/8})^2 = a^{(q+3)/4} = a \cdot a^{(q-1)/4}$$

Soit $b = a^{(q-1)/4}$. Puisque a est un carré, le critère d'Euler (Théorème 4.14) donne $a^{(q-1)/2} = 1$, donc $b^2 = a^{(q-1)/2} = 1$, ce qui implique $b = \pm 1$.

On distingue deux cas :

- Si $b = 1$, alors $x_0^2 = a \cdot 1 = a$, donc x_0 est une racine carrée de a .
- Si $b = -1$, alors $x_0^2 = a \cdot (-1) = -a$. Soit $i = 2^{(q-1)/4}$. Alors :

$$i^2 = (2^{(q-1)/4})^2 = 2^{(q-1)/2}$$

Par le lemme précédent, pour $q \equiv 5 \pmod{8}$, on a $\left(\frac{2}{q}\right) = -1$, donc par le critère d'Euler (Théorème 4.14), $2^{(q-1)/2} = -1$. Ainsi $i^2 = -1$ et $x = x_0 \cdot i$ vérifie $x^2 = x_0^2 \cdot i^2 = (-a) \cdot (-1) = a$, donc x est une racine carrée de a .

Dans les deux cas, on obtient une racine carrée x de a . L'autre racine carrée est $-x$. Puisque la caractéristique de \mathbb{F}_q est différente de 2, on a $x \neq -x$. En effet, si $x = -x$, alors $2x = 0$, ce qui impliquerait $x = 0$, mais $a = x^2 = 0$, contradiction avec a non nul.

De plus, les bits de parité de x et $-x$ sont opposés. En effet, si on représente les éléments de \mathbb{F}_q par des entiers entre 0 et $q - 1$, alors $-x$ est représenté par $q - x$. Comme q est impair, $q - x$ a la parité opposée à x . \square

Définition 4.18 (Décodage des points Ed25519). Le décodage d'un point compressé est l'application :

$$\text{dec} : \{0, 1\}^{256} \rightarrow E(\mathbb{F}_q) \cup \{\perp\}$$

définie comme suit. Soit $P_{\text{enc}} \in \{0, 1\}^{256}$ un point encodé :

1. Interpréter P_{enc} comme un entier en little-endian $0 \leq N < 2^{256}$. Noter p le bit de poids fort de N (bit d'indice 255), et y l'entier obtenu en annulant ce bit, c'est-à-dire en prenant les 255 bits de poids faible. On identifie alors y avec un élément de $\{0, 1\}^{255}$ et $p \in \{0, 1\}$.
2. Interpréter y (les 255 bits de poids faible) comme un élément de \mathbb{F}_q (en little-endian) et le noter y_P .
3. Calculer $x_P^2 = \frac{y_P^2 - 1}{dy_P^2 + 1}$ dans \mathbb{F}_q .
4. Si x_P^2 n'est pas un carré dans \mathbb{F}_q , retourner \perp .
5. Calculer une racine carrée x_P de x_P^2 en utilisant le théorème précédent.
6. Parmi les deux racines carrées $\{x_P, -x_P\}$, choisir celle dont le bit de parité (bit le moins significatif) est égal à p .
7. Retourner le point $P = (x_P, y_P)$.

Remarque 4.19 (Implémentation efficace de la racine carrée). En pratique, on calcule d'abord :

$$x = a^{(q+3)/8}$$

puis on vérifie :

- Si $x^2 = a$, alors x est une racine carrée de a .
- Si $x^2 = -a$, alors $x \cdot 2^{(q-1)/4}$ est une racine carrée de a .
- Sinon, a n'est pas un carré dans \mathbb{F}_q .

On ajuste ensuite la parité en choisissant entre la racine obtenue et son opposée selon le bit p stocké dans l'encodage.

4.4 Clés Ed25519

Définition 4.20 (Clampage Ed25519). Soit $H_1 \in \{0, 1\}^{256}$, que l'on interprète comme une suite de 32 octets $(h_0, h_1, \dots, h_{31})$ en little-endian. Le clampage Ed25519 est l'application :

$$\text{clamp} : \{0, 1\}^{256} \rightarrow \mathbb{Z}_\ell$$

définie par les opérations suivantes sur les octets :

$$\begin{aligned} h_0 &\leftarrow h_0 \wedge \text{0xF8} \quad (\text{bits 0-2 à 0}), \\ h_{31} &\leftarrow (h_{31} \wedge \text{0x7F}) \vee \text{0x40} \quad (\text{bit 255 à 0, bit 254 à 1}), \end{aligned}$$

puis on pose

$$a = \text{val}_\ell(h_0, h_1, \dots, h_{31}) \bmod \ell.$$

Définition 4.21 (Génération de clés Ed25519). Soit $K_{\text{Ed}} \in \{0, 1\}^{256}$. On calcule :

$$H = \text{SHA-512}(K_{\text{Ed}}) = H_1 \parallel H_2 \quad \text{avec } H_1, H_2 \in \{0, 1\}^{256}$$

La clé privée est :

$$a = \text{clamp}(H_1) \in \mathbb{Z}_\ell$$

La clé publique est :

$$A = aB \in E(\mathbb{F}_q)$$

4.5 Algorithme de signature Ed25519

Définition 4.22 (Signature Ed25519 [6]). L'algorithme de signature $\text{Sign} : \mathbb{Z}_\ell \times \{0, 1\}^* \rightarrow \{0, 1\}^{512}$ est défini comme suit :

Pour $(a, M) \in \mathbb{Z}_\ell \times \{0, 1\}^*$:

1. Calculer $H = \text{SHA-512}(K_{\text{Ed}}) = H_1 \parallel H_2$
2. Définir $r = \text{val}_{\text{le}}(\text{SHA-512}(H_2 \parallel M)) \bmod \ell$
3. Calculer $R = rB \in E(\mathbb{F}_q)$
4. Calculer $k = \text{val}_{\text{le}}(\text{SHA-512}(\text{enc}(R) \parallel \text{enc}(A) \parallel M)) \bmod \ell$
5. Calculer $s = (r + k \cdot a) \bmod \ell$
6. La signature est $\sigma = \text{enc}(R) \parallel \text{bytes}_{\text{le}}(s)$

4.6 Algorithme de vérification Ed25519

Définition 4.23 (Vérification Ed25519 [6]). L'algorithme de vérification $\text{Verify} : E(\mathbb{F}_q) \times \{0, 1\}^* \times \{0, 1\}^{512} \rightarrow \{\text{OK}, \perp\}$ est défini comme suit :

Pour $(A, M, \sigma) \in E(\mathbb{F}_q) \times \{0, 1\}^* \times \{0, 1\}^{512}$:

1. Parser $\sigma = R_{\text{enc}} \parallel s_{\text{enc}}$ avec $R_{\text{enc}}, s_{\text{enc}} \in \{0, 1\}^{256}$
2. Décoder $R = \text{dec}(R_{\text{enc}}) \in E(\mathbb{F}_q)$
3. Décoder $s = \text{val}_{\text{le}}(s_{\text{enc}}) \in \mathbb{Z}_\ell$
4. Vérifier que A et R sont des points valides sur $E(\mathbb{F}_q)$
5. Vérifier que $s \in \{0, \dots, \ell - 1\}$
6. Calculer $k = \text{val}_{\text{le}}(\text{SHA-512}(R_{\text{enc}} \parallel \text{enc}(A) \parallel M)) \bmod \ell$
7. Vérifier l'équation :

$$sB = R + kA$$

Si toutes les vérifications réussissent, retourner OK, sinon \perp .

Théorème 4.24 (Correction de la vérification Ed25519). Pour toute paire (a, A) générée valide et tout message M ,

$$\text{Verify}(A, M, \text{Sign}(a, M)) = \text{OK}$$

Démonstration. Soit $\sigma = (R, s)$ une signature valide. Alors :

$$\begin{aligned} sB &= (r + k \cdot a)B \\ &= rB + k \cdot (aB) \\ &= R + kA \end{aligned}$$

L'équation de vérification est donc satisfaite. □

Remarque 4.25 (Signature contextuelle dans le schéma k -sur- n). Dans notre implémentation, on signe non pas M directement mais le message contextualisé :

$$M' = \text{"kofn-ed25519-v1"} \parallel \text{SHA256}(A) \parallel M$$

où A est la clé publique Ed25519. Cela empêche la réutilisation des signatures hors contexte et garantit la liaison avec la clé maîtresse du schéma.

Remarque 4.26 (Sécurité des signatures). La sécurité d'Ed25519 repose sur la difficulté du problème du logarithme discret dans le groupe $\langle B \rangle$ et sur les propriétés de résistance aux collisions de SHA-512. Le clamping empêche les attaques par canaux auxiliaires et garantit que le scalaire est dans le sous-groupe principal.

5 RSA, MGF1 et OAEP

5.1 Chiffrement asymétrique RSA

Définition 5.1 (Clés RSA (Rivest–Shamir–Adleman)). On choisit deux nombres premiers p, q de taille comparable (typiquement $p, q \approx 2^{2048}$ pour un module de 4096 bits) et on pose $n = pq$.

On calcule $\varphi(n) = (p-1)(q-1)$ (voir Proposition 1.7).

On choisit un exposant public e tel que $\gcd(e, \varphi(n)) = 1$ (typiquement $e = 65537$). L'exposant privé d est l'inverse de e modulo $\varphi(n)$:

$$d \equiv e^{-1} \pmod{\varphi(n)}.$$

La clé publique est (n, e) et la clé privée est (n, d) .

Remarque 5.2 (Taille, primalité et génération concrète d'une clé RSA de 4096 bits). Lorsque l'on parle d'une « clé RSA de 4096 bits », on fait référence à la taille du module $n = pq$, c'est-à-dire au nombre de bits de son écriture binaire. Par convention, un module RSA de 4096 bits vérifie

$$2^{4095} \leq n < 2^{4096},$$

ce qui équivaut à dire que n a exactement 4096 bits en binaire.

Taille exacte du module. En pratique, les bibliothèques cryptographiques génèrent deux nombres premiers p et q de 2048 bits (avec le bit de poids fort forcé à 1 pour garantir la taille), puis calculent $n = pq$. Si l'on modélise p et q comme uniformément répartis parmi les entiers de 2048 bits, on peut écrire

$$p = 2^{2047}u, \quad q = 2^{2047}v, \quad \text{avec } u, v \in [1, 2[.$$

On obtient alors $n = pq = 2^{4094}uv$. Le module n a 4095 bits si et seulement si $uv < 2$, ce qui conduit à

$$\mathbb{P}(n \text{ a } 4095 \text{ bits}) = \mathbb{P}(uv < 2) = \int_1^2 \left(\frac{2}{u} - 1 \right) du = 2 \ln 2 - 1 \approx 0,386,$$

et donc

$$\mathbb{P}(n \text{ a } 4096 \text{ bits}) = 1 - (2 \ln 2 - 1) = 2 - 2 \ln 2 \approx 0,614.$$

Autrement dit, si l'on se contentait de choisir p et q de 2048 bits puis de prendre $n = pq$, on obtiendrait un module de 4096 bits dans un peu plus de 60 % des cas.

Pour garantir que le module a bien 4096 bits, on utilise en pratique l'une des deux stratégies suivantes :

- soit on applique un *rejection sampling* : on génère p et q (premiers de 2048 bits), on calcule $n = pq$, puis on rejette les couples pour lesquels n n'a que 4095 bits, jusqu'à obtenir un n vérifiant $\lfloor \log_2 n \rfloor + 1 = 4096$;
- soit on contraint légèrement la forme de p et de q (en fixant notamment quelques bits de poids fort et la taille minimale) de manière à ce que le produit pq soit, sauf cas exceptionnel, automatiquement dans l'intervalle $[2^{4095}, 2^{4096})$, tout en rejetant les rares cas restants où la longueur ne serait pas exactement 4096 bits.

Dans les deux cas, la taille annoncée de la clé (4096 bits) correspond effectivement à la taille mathématique du module. Dans notre implémentation, qui s'appuie sur la bibliothèque Python `cryptography` et donc sur OpenSSL, c'est essentiellement la seconde stratégie qui est utilisée : p et q sont générés avec une forme légèrement contrainte, puis la génération est répétée tant que le module $n = pq$ n'a pas exactement 4096 bits.

Génération pratique de premiers de 2048 bits. En pratique, les bibliothèques cryptographiques génèrent p et q de la manière suivante :

1. *Tirage d'un candidat impair de 2048 bits.* On tire aléatoirement un entier N de 2048 bits à l'aide d'un générateur pseudo-aléatoire cryptographiquement sûr, on force le bit de poids fort à 1 (pour garantir la taille) et le bit de poids faible à 1 (pour que N soit impair).
2. *Filtrage par petits premiers.* On élimine rapidement les candidats manifestement composés en testant la divisibilité de N par une liste de petits nombres premiers (par exemple les premiers ≤ 1000).
3. *Test de primalité probabiliste (Miller–Rabin).* Pour les candidats survivants, on applique plusieurs itérations du test de Miller–Rabin. On écrit $N - 1 = 2^s d$ avec d impair, puis, pour chaque base a choisie (aléatoire ou selon une liste standard), on examine la suite

$$a^d, a^{2d}, a^{4d}, \dots, a^{2^{s-1}d} \pmod{N}.$$

Si, pour une base donnée, cette suite ne se comporte pas comme elle le ferait modulo un nombre premier, on peut conclure que N est composé et rejeter ce candidat. Si N passe le test pour toutes les bases choisies, on le déclare *premier probable*.

On montre que, pour un entier impair composé N , la proportion de bases a pour lesquelles le test de Miller–Rabin échoue à détecter la composité est au plus $1/4$. Ainsi, si l'on utilise k bases indépendantes, la probabilité qu'un nombre *composé* passe tous les tests est majorée par $(1/4)^k = 2^{-2k}$. Avec un nombre raisonnable de bases (quelques dizaines au plus), la probabilité de déclarer « premier » un entier en réalité composé devient négligeable (par exemple $\leq 2^{-128}$), ce qui est largement suffisant pour les usages cryptographiques.

Ordre de grandeur du nombre de premiers disponibles. Le nombre de nombres premiers de 2048 bits est colossal : le théorème des nombres premiers donne l'approximation

$$\#\{\text{premiers de 2048 bits}\} \approx \frac{2^{2047}}{2048 \ln 2} \approx 10^{613}.$$

Même en rejetant une fraction non négligeable des couples (p, q) pour des raisons de taille du module ou de tests de primalité, l'espace de recherche reste astronomique, et ces contraintes n'affaiblissent pas la sécurité de RSA.

Théorème 5.3 (Correction du chiffrement RSA). Pour tout message $M \in \{0, \dots, n-1\}$ et toute clé RSA valide (n, e, d) , on a :

$$(M^e)^d \equiv M \pmod{n}.$$

Démonstration. Puisque $ed \equiv 1 \pmod{\varphi(n)}$, il existe k tel que $ed = 1 + k(p-1)(q-1)$.

Modulo p :

- Si $p \mid M$: alors $M \equiv 0 \pmod{p}$, donc $M^{ed} \equiv 0 \equiv M \pmod{p}$.
- Si $p \nmid M$: par le petit théorème de Fermat (théorème 1.10), $M^{p-1} \equiv 1 \pmod{p}$, donc

$$M^{ed} = M^{1+k(p-1)(q-1)} = M \cdot (M^{p-1})^{k(q-1)} \equiv M \cdot 1^{k(q-1)} = M \pmod{p}.$$

Ainsi, dans tous les cas, $M^{ed} \equiv M \pmod{p}$, donc $p \mid (M^{ed} - M)$.

Modulo q :

- Si $q \mid M$: alors $M \equiv 0 \pmod{q}$, donc $M^{ed} \equiv 0 \equiv M \pmod{q}$.
- Si $q \nmid M$: par le petit théorème de Fermat (théorème 1.10), $M^{q-1} \equiv 1 \pmod{q}$, donc

$$M^{ed} = M^{1+k(p-1)(q-1)} = M \cdot (M^{q-1})^{k(p-1)} \equiv M \cdot 1^{k(p-1)} = M \pmod{q}.$$

Ainsi, dans tous les cas, $M^{ed} \equiv M \pmod{q}$, donc $q \mid (M^{ed} - M)$.

Soit $N = M^{ed} - M$. Nous avons montré que $p \mid N$ et $q \mid N$. Puisque p et q sont premiers distincts, ils sont premiers entre eux.

Comme $p \mid N$, on peut écrire $N = p \cdot K$ pour un certain entier K . Puisque $q \mid N = p \cdot K$ et que $\gcd(p, q) = 1$, par le théorème de Gauss (théorème 1.3), $q \mid K$. Donc $K = q \cdot L$ pour un certain entier L , et ainsi :

$$N = p \cdot K = p \cdot q \cdot L = n \cdot L,$$

ce qui montre que $n \mid N$, c'est-à-dire :

$$M^{ed} \equiv M \pmod{n}.$$

□

Remarque 5.4 (Risque lorsque $\gcd(M, n) \neq 1$). Si $\gcd(M, n) \neq 1$, alors M est divisible par p ou par q , ce qui permettrait à un attaquant de factoriser n en calculant $\gcd(M, n)$.

Bien que théoriquement possible, cette attaque est pratiquement irréalisable :

- Pour un module RSA de 4096 bits, la probabilité qu'un message aléatoire M (de 4096 bits après encodage OAEP) ait un facteur commun avec n est d'environ 2^{-2048} .
- Cette probabilité est bien inférieure à l'inverse du nombre estimé de particules dans l'univers observable ($\approx 10^{80} \approx 2^{266}$).
- Même en chiffrant un milliard de messages par seconde pendant l'âge de l'univers ($\approx 4 \times 10^{17}$ secondes), le nombre total de messages serait d'environ $10^{26} \approx 2^{86}$, et l'espérance du nombre de messages « vulnérables » serait de $2^{86} \cdot 2^{-2048} = 2^{-1962}$, ce qui reste bien inférieur à 1.

- Si un tel événement se produisait par miracle, le chiffrement et le déchiffrement fonctionneraient parfaitement normalement, et personne ne s'en rendrait compte sans calculer explicitement $\gcd(M, n)$.

En pratique, cette attaque n'est donc pas une préoccupation réaliste pour la sécurité de RSA avec des paramètres standard.

Définition 5.5 (Paramètres concrets pour le schéma k -sur- n). Dans notre implémentation :

- Module n : 4096 bits ;
- Exposant public $e = 65537$;
- Exposant privé $d \equiv e^{-1} \pmod{\varphi(n)}$.

Remarque 5.6 (Sécurité RSA). La sécurité de RSA repose sur la difficulté de la factorisation du module n . Pour un module de 4096 bits, construit comme indiqué ci-dessus, cela offre une sécurité suffisante selon les standards actuels.

5.2 MGF1

Définition 5.7 (Fonction de génération de masque MGF1 (Mask Generation Function 1) [7]). Soit $h : \{0, 1\}^* \rightarrow \{0, 1\}^d$ une fonction de hachage. On définit

$$\text{MGF1}_h : \{0, 1\}^* \times \mathbb{N} \rightarrow \{0, 1\}^\ell$$

par :

$$\text{MGF1}_h(Z, \ell) = T_1 \parallel T_2 \parallel \dots \parallel T_{\lceil \ell/d \rceil} \quad \text{tronqué à } \ell \text{ bits}$$

où

$$T_i = h(Z \parallel \text{INT}_4(i-1)) \quad \text{pour } i = 1, 2, \dots, \lceil \ell/d \rceil$$

et $\text{INT}_4(j)$ est l'encodage big-endian de j sur 4 octets.

Propriété 5.8 (Propriétés de MGF1). MGF1 est déterministe et peut générer des masques de longueur arbitraire. Sa sécurité repose sur les propriétés de résistance aux collisions et de préimage de la fonction de hachage sous-jacente h .

Remarque 5.9 (Rôle crucial de MGF1 dans OAEP). MGF1 joue un rôle essentiel dans OAEP en empêchant les attaques par blocs indépendants :

- **Transformation de taille** : Passer d'une entrée fixe à une sortie de longueur arbitraire
- **Distribution uniforme** : Garantir que le masque n'a pas de motifs détectables
- **Non-corrélation** : Assurer que chaque partie du masque est unique grâce au compteur
- **Entrelacement cryptographique** : Créer des dépendances non linéaires entre `maskedSeed` et `maskedDB` via la structure :

$$\begin{cases} \text{maskedDB} = \text{DB} \oplus \text{MGF1}_h(\text{seed}) \\ \text{maskedSeed} = \text{seed} \oplus \text{MGF1}_h(\text{maskedDB}) \end{cases}$$

- **Prévention d'attaques par blocs** : Toute modification d'un bit dans une partie affecte de manière imprévisible l'ensemble du message, rendant impossible les attaques ciblées sur des blocs individuels
- **Sécurité prouvée** : Permettre les preuves formelles de sécurité d'OAEP contre les attaques adaptatives (IND-CCA2)

Sans MGF1, un attaquant pourrait manipuler séparément les différentes parties du message encodé, réduisant la sécurité à celle de schémas de padding vulnérables comme PKCS#1 v1.5. MGF1 est donc la « colle cryptographique » qui assure l'indissociabilité des composants d'OAEP.

5.3 RSA-OAEP

Définition 5.10 (Paramètres RSA-OAEP (Optimal Asymmetric Encryption Padding) [7]). Soient k la taille en octets du module n (pour n de 4096 bits, $k = 512$), et $hLen$ la taille de sortie de la fonction de hachage en octets (pour SHA-256, $hLen = 32$). On fixe :

$$k_0 = k - 2hLen - 2, \quad k_1 = hLen$$

où k_0 est la longueur maximale du message en octets et k_1 la longueur de l'aléa r .

Définition 5.11 (Encodage OAEP). L'encodage OAEP pour un message $M \in \{0, 1\}^{8k_0}$, un label $L \in \{0, 1\}^*$ et un aléa $r \in \{0, 1\}^{8k_1}$ est défini comme suit :

1. Calculer $lHash = h(L)$
2. Former la chaîne $PS = 0^{8(k-k_0-2hLen-2)}$ (padding de zéros)
3. Construire le bloc $DB = lHash \parallel PS \parallel 0x01 \parallel M$
4. Calculer $maskedDB = DB \oplus \text{MGF1}_h(r, 8(k - hLen - 1))$
5. Calculer $maskedSeed = r \oplus \text{MGF1}_h(maskedDB, 8hLen)$
6. Le message encodé est $EM = maskedSeed \parallel maskedDB \in \{0, 1\}^{8k}$

Définition 5.12 (Chiffrement RSA-OAEP). Soient (n, e) une clé publique RSA, k_0, k_1 les paramètres de taille définis ci-dessus. On définit :

$$\begin{aligned} \text{RSA-OAEP}_{(n,e)} : \{0, 1\}^{8k_0} \times \{0, 1\}^{8k_1} &\rightarrow \{0, \dots, n-1\} \\ \text{RSA-OAEP}_{(n,e)}(M; r) &= \text{val}_{\text{be}}(EM)^e \bmod n \end{aligned}$$

où EM est le résultat de l'encodage OAEP de M avec l'aléa r .

Définition 5.13 (Déchiffrement RSA-OAEP). Soient (n, d) une clé privée RSA. On définit :

$$\begin{aligned} \text{RSA-OAEP}_{(n,d)}^{-1} : \{0, \dots, n-1\} &\rightarrow \{0, 1\}^{8k_0} \cup \{\perp\} \\ \text{RSA-OAEP}_{(n,d)}^{-1}(C) &= \begin{cases} M & \text{si le décodage réussit} \\ \perp & \text{sinon} \end{cases} \end{aligned}$$

où $C \in \{0, \dots, n-1\}$ est le **texte chiffré** (ciphertext) obtenu par $\text{RSA-OAEP}_{(n,e)}(M; r)$.

Le déchiffrement procède comme suit :

1. Calculer $EM = C^d \bmod n$ et convertir en binary sur $8k$ bits

2. Parser $EM = \text{maskedSeed} \parallel \text{maskedDB}$ avec $\text{maskedSeed} \in \{0, 1\}^{8hLen}$
3. Calculer $\text{seed} = \text{maskedSeed} \oplus \text{MGF1}_h(\text{maskedDB}, 8hLen)$
4. Calculer $DB = \text{maskedDB} \oplus \text{MGF1}_h(\text{seed}, 8(k - hLen - 1))$
5. Parser $DB = lHash' \parallel PS \parallel 0x01 \parallel M$ où PS est une suite de zéros
6. Vérifier que $lHash' = h(L)$ et que PS contient bien que des zéros
7. Si toutes les vérifications passent, retourner M , sinon \perp

Remarque 5.14 (Statut de l'aléa r dans OAEP). Contrairement aux sels publics utilisés dans HKDF ou PBKDF2, l'aléa r dans OAEP est « cryptographiquement protégé » :

- **Non public** : r n'est pas stocké en clair ni transmis publiquement
- **Masqué cryptographiquement** : Il est caché dans le chiffré via $\text{maskedSeed} = r \oplus \text{MGF1}_h(\text{maskedDB})$
- **Récupération au déchiffrement** : Seul le possesseur de la clé privée peut retrouver r
- **Rôle éphémère** : r est utilisé une seule fois puis « jeté »

La sécurité d'OAEP repose sur le fait que r reste « imprévisible » pour un attaquant au moment du chiffrement.

Remarque 5.15 (Paramètres numériques pour RSA-4096 et SHA-256). Pour un module RSA de 4096 bits ($k = 512$ octets) avec SHA-256 ($hLen = 32$ octets), on a :

$$k_0 = 512 - 2 \times 32 - 2 = 446 \text{ octets}$$

Cette capacité de 446 octets est amplement suffisante pour une clé AES-256 de 32 octets, avec une marge importante pour les données de padding.

Théorème 5.16 (Correction de RSA-OAEP). Pour toute clé (n, e, d) valide, pour tout message M de longueur $8k_0$ bits et pour tout aléa $r \in \{0, 1\}^{8k_1}$,

$$\text{RSA-OAEP}_{(n,d)}^{-1}(\text{RSA-OAEP}_{(n,e)}(M; r)) = M.$$

Démonstration. La correction découle de la structure réversible de l'encodage OAEP. Les opérations de masquage utilisant MGF sont réversibles car déterministe, et les vérifications assurent l'intégrité du message. \square

Remarque 5.17 (Propriétés de sécurité). RSA-OAEP offre une sécurité prouvée dans le modèle de l'oracle aléatoire contre les attaques adaptatives à chiffrés choisis (IND-CCA2). Le padding OAEP prévient les attaques par canal auxiliaire et garantit l'intégrité du message.

Définition 5.18 (Utilisation dans le schéma k -sur- n). Dans notre contexte, RSA-OAEP est utilisé pour chiffrer des clés AES-256 de 32 octets. Pour un module RSA de 4096 bits ($k = 512$) avec SHA-256 ($hLen = 32$), on a :

$$k_0 = 512 - 2 \times 32 - 2 = 446 \text{ octets}$$

Ce qui est amplement suffisant pour une clé AES-256 de 32 octets, avec une marge importante de 414 octets pour le padding OAEP.

6 AES-GCM

6.1 AES comme permutation de bloc

Définition 6.1 (AES-256 (Advanced Encryption Standard) selon FIPS 197 [8]). On fixe une taille de bloc de 128 bits et une taille de clé de 256 bits. Pour chaque clé $K \in \{0, 1\}^{256}$, on dispose d'une permutation de blocs

$$E_K : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128},$$

bijective, avec réciproque notée $D_K = E_K^{-1}$.

L'algorithme AES-256 opère sur un *état* (state) représenté comme une matrice 4×4 d'octets, notée :

$$\text{state} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

où chaque $s_{r,c}$ est un octet ($0 \leq r < 4, 0 \leq c < 4$).

Note : L'algorithme complet de chiffrement (fonction CIPHER) sera détaillé à la section 6.5.

Définition 6.2 (Mapping entrée-état selon FIPS 197). Le mapping entre le bloc d'entrée $in \in \{0, 1\}^{128}$ et l'état suit l'ordre *colonne-major* :

$$s[r, c] = in[8 \cdot (4c + r) \dots 8 \cdot (4c + r) + 7] \quad \text{pour } 0 \leq r < 4, 0 \leq c < 4$$

En représentation octet, cela équivaut à :

$$s[r, c] = in[4c + r] \quad \text{pour } 0 \leq r < 4, 0 \leq c < 4$$

où in est vu comme un tableau de 16 octets $in[0] \dots in[15]$.

Définition 6.3 (Paramètres AES-256). Pour AES-256, on a les paramètres fixes suivants :

- $Nk = 8$ (nombre de mots de 32 bits dans la clé)
- $Nb = 4$ (nombre de mots de 32 bits dans le bloc/état)
- $Nr = 14$ (nombre de tours)

6.2 Correspondance des corps finis

Remarque 6.4 (Corps $\text{GF}(2^8)$ d'AES). Le corps $\text{GF}(2^8)$ utilisé dans AES correspond à la construction suivante dans notre formalisme :

$$\mathbb{F}_{2^8} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$$

où :

- \mathbb{F}_2 est le corps à 2 éléments (corps premier)
- Le polynôme $m(X) = X^8 + X^4 + X^3 + X + 1$ est irréductible sur \mathbb{F}_2
- Chaque élément de \mathbb{F}_{2^8} est un polynôme de degré ≤ 7 à coefficients dans \mathbb{F}_2

— Un octet $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ représente le polynôme $b_7X^7 + b_6X^6 + \dots + b_1X + b_0$

Remarque 6.5 (Différence avec le corps de Shamir). Le corps \mathbb{F}_P utilisé pour le partage de Shamir (avec $P = 2^{521} - 1$) est de caractéristique différente :

- \mathbb{F}_P : corps premier de caractéristique P (premier)
- \mathbb{F}_{2^8} : corps de caractéristique 2

Cette différence est fondamentale et explique pourquoi les opérations arithmétiques (addition, multiplication) diffèrent radicalement entre Shamir et AES.

Proposition 6.6 (Opérations dans les corps de caractéristique 2). Dans \mathbb{F}_{2^8} :

- **Addition** : équivaut au XOR bit-à-bit
- **Soustraction** : identique à l'addition ($a - b = a + b$)
- **Multiplication** : multiplication polynomiale suivie de réduction modulo le polynôme irréductible

6.3 Transformations de base

Définition 6.7 (S-box AES). La S-box (table de substitution) AES est une table fixe de 256 octets définie dans le standard FIPS 197. Pour un octet d'entrée xy en hexadécimal (où x est le nibble de poids fort et y le nibble de poids faible), la valeur de substitution est donnée par la table suivante :

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Cette table est notée $SBox : \{0, 1\}^8 \rightarrow \{0, 1\}^8$.

Définition 6.8 (SUBBYTES()). La transformation SUBBYTES() applique la S-box à chaque octet de l'état :

$$s'_{r,c} = SBox(s_{r,c}) \quad \text{pour } 0 \leq r < 4, 0 \leq c < 4$$

Bien que la S-box soit définie par un algorithme (inversion dans $\text{GF}(2^8)$ suivie d'une transformation affine), dans la pratique elle est implémentée comme une table de consultation fixe pour des raisons de performance.

Remarque 6.9 (Construction algorithmique de la S-box). La S-box peut être générée algorithmiquement pour tout octet $b \neq 0$ par :

1. Calculer l'inverse multiplicatif b^{-1} dans $\text{GF}(2^8)$ modulo $m(x) = x^8 + x^4 + x^3 + x + 1$
2. Appliquer la transformation affine :

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

où $c = 0\text{x}63 = \{01100011\}$.

Pour $b = 0$, on utilise $0^{-1} = 0$ par convention.

Définition 6.10 (SHIFTRROWS()). La transformation SHIFTRROWS() décale cycliquement les lignes de l'état :

- Ligne 0 : pas de décalage
- Ligne 1 : décalage de 1 position vers la gauche
- Ligne 2 : décalage de 2 positions vers la gauche
- Ligne 3 : décalage de 3 positions vers la gauche

Formellement : $s'_{r,c} = s_{r,(c+r) \bmod 4}$ pour $0 \leq r < 4$, $0 \leq c < 4$.

Définition 6.11 (MIXCOLUMNS()). La transformation MIXCOLUMNS() traite chaque colonne de l'état comme un polynôme sur $\text{GF}(2^8)$ et le multiplie modulo $x^4 + 1$ par le polynôme fixe :

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Ceci équivaut à la multiplication matricielle :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{pour } 0 \leq c < 4$$

6.4 Expansion de clé AES-256

Définition 6.12 (Structure des mots de clé). Un mot $w[i]$ est une séquence de 32 bits interprétée comme 4 octets :

$$w[i] = (w[i]_0, w[i]_1, w[i]_2, w[i]_3)$$

où $w[i]_0$ est l'octet de poids fort et $w[i]_3$ l'octet de poids faible.

Définition 6.13 (KEYEXPANSION() pour AES-256). L'expansion de clé génère $4 \times (Nr + 1) = 60$ mots de 32 bits à partir de la clé initiale.

Soit $key[0..7]$ les 8 mots initiaux de la clé. Pour $i = 0$ à 59 :

- Si $i < 8$: $w[i] = key[i]$
- Si $i \geq 8$ et $i \bmod 8 = 0$:

$$w[i] = w[i - 8] \oplus \text{SubWord}(\text{RotWord}(w[i - 1])) \oplus \text{Rcon}[i/8]$$

— Si $i \geq 8$ et $i \bmod 8 = 4$:

$$w[i] = w[i - 8] \oplus \text{SubWord}(w[i - 1])$$

— Sinon :

$$w[i] = w[i - 8] \oplus w[i - 1]$$

où :

- $\text{RotWord}([a, b, c, d]) = [b, c, d, a]$
- $\text{SubWord}([a, b, c, d]) = [\text{SBox}(a), \text{SBox}(b), \text{SBox}(c), \text{SBox}(d)]$
- $\text{Rcon}[j] = [x^{j-1}, \{00\}, \{00\}, \{00\}]$ avec $x = \{02\}$

Définition 6.14 ($\text{ADDROUNDKEY}()$). La transformation $\text{ADDROUNDKEY}()$ combine l'état avec la clé de tour par XOR. Pour chaque colonne c ($0 \leq c < 4$), on a :

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus w[4 \times \text{round} + c]$$

où $w[i]$ est le i -ème mot du schedule de clés et round est le numéro du tour.

6.5 Algorithme de chiffrement complet

Définition 6.15 ($\text{CIPHER}()$ pour AES-256). L'algorithme de chiffrement complet est :

1. $\text{state} \leftarrow \text{in}$ (copie de l'entrée dans l'état selon le mapping colonne-major)
2. $\text{state} \leftarrow \text{ADDROUNDKEY}(\text{state}, w[0..3])$
3. Pour $\text{round} = 1$ à $Nr - 1$:
 - (a) $\text{state} \leftarrow \text{SUBBYTES}(\text{state})$
 - (b) $\text{state} \leftarrow \text{SHIFTRROWS}(\text{state})$
 - (c) $\text{state} \leftarrow \text{MIXCOLUMNS}(\text{state})$
 - (d) $\text{state} \leftarrow \text{ADDROUNDKEY}(\text{state}, w[4 \times \text{round}..4 \times \text{round} + 3])$
4. $\text{state} \leftarrow \text{SUBBYTES}(\text{state})$
5. $\text{state} \leftarrow \text{SHIFTRROWS}(\text{state})$
6. $\text{state} \leftarrow \text{ADDROUNDKEY}(\text{state}, w[4 \times Nr..4 \times Nr + 3])$
7. $\text{out} \leftarrow \text{state}$ (conversion selon le mapping colonne-major inverse)

Définition 6.16 (Arithmétique dans $\text{GF}(2^8)$). Le corps fini $\text{GF}(2^8)$ est défini par le polynôme irréductible :

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Chaque octet $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ représente le polynôme :

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

L'addition est le XOR bit-à-bit. La multiplication est la multiplication polynomiale modulo $m(x)$.

Remarque 6.17 (Implémentation efficace). La multiplication par $\{02\}$ (notée $\text{XTIMES}()$) peut être implémentée efficacement :

$$\text{XTIMES}(b) = \begin{cases} (b \ll 1) & \text{si } b_7 = 0 \\ (b \ll 1) \oplus \{1b\} & \text{si } b_7 = 1 \end{cases}$$

Cette opération est utilisée dans $\text{MIXCOLUMNS}()$ et $\text{KEYEXPANSION}()$.

6.6 Corps $\mathbb{F}_{2^{128}}$ et GHASH

Définition 6.18 (Corps $\mathbb{F}_{2^{128}}$). On fixe un polynôme irréductible $P(X) \in \mathbb{F}_2[X]$ de degré 128 (par exemple $X^{128} + X^7 + X^2 + X + 1$) et on définit

$$\mathbb{F}_{2^{128}} = \mathbb{F}_2[X]/(P(X)).$$

On fixe une bijection entre $\{0, 1\}^{128}$ et $\mathbb{F}_{2^{128}}$ en identifiant le bloc (b_{127}, \dots, b_0) au polynôme $\sum_{i=0}^{127} b_i X^i \bmod P(X)$.

Définition 6.19 (Sous-clé de hachage H). Pour une clé AES K , on définit

$$H = E_K(0^{128}) \in \{0, 1\}^{128},$$

que l'on identifie à un élément de $\mathbb{F}_{2^{128}}$.

Définition 6.20 (GHASH (Galois Hash) selon SP 800-38D [10]). Soit $H \in \mathbb{F}_{2^{128}}$. Pour une séquence de blocs $X_1, \dots, X_m \in \{0, 1\}^{128}$, la fonction GHASH est définie récursivement par :

$$Y_0 = 0^{128}, \quad Y_i = (Y_{i-1} \oplus X_i) \cdot H \quad \text{pour } i = 1, \dots, m$$

Le résultat est Y_m . Cette définition est équivalente à :

$$\text{GHASH}_H(X_1, \dots, X_m) = \sum_{j=1}^m X_j \cdot H^{m-j+1}$$

6.7 Mode AES-GCM

Définition 6.21 (AES-GCM (Galois/Counter Mode) selon SP 800-38D [10]). Pour une clé $K \in \{0, 1\}^{256}$, on modélise AES-GCM comme un couple d'applications

$$\begin{aligned} \text{AES-GCM}_K &: \{0, 1\}^{96} \times \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^* \times \{0, 1\}^{128} \\ \text{AES-GCM}_K^{-1} &: \{0, 1\}^{96} \times \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^{128} \rightarrow \{0, 1\}^* \cup \{\perp\} \end{aligned}$$

où :

- Le premier argument est un nonce $N \in \{0, 1\}^{96}$
- Le deuxième argument est les données authentifiées associées (AAD) $A \in \{0, 1\}^*$
- Le troisième argument est le message $M \in \{0, 1\}^*$
- La sortie est un couple (C, T) avec C le texte chiffré et T le tag (128 bits)

Remarque 6.22 (Rôle d'AES dans AES-GCM). Le chiffrement AES est utilisé à deux endroits essentiels dans GCM :

1. **Dans GCTR** : Pour générer la séquence de masquage en chiffrant les compteurs J_i :

$$E_K(J_i) \quad \text{servant de masque pour } M_i \oplus E_K(J_i)$$

2. **Pour calculer H** : La sous-clé de hachage est obtenue par :

$$H = E_K(0^{128})$$

qui est utilisée dans toutes les multiplications GHASH.

Ainsi, AES fournit à la fois la confidentialité (via GCTR) et la base de l'authentification (via H).

Définition 6.23 (Structure mathématique de GCM). Le mode GCM combine :

- Le chiffrement en mode compteur (GCTR) utilisant E_K pour la confidentialité
- La fonction d'authentification GHASH dans $\mathbb{F}_{2^{128}}$ utilisant $H = E_K(0^{128})$ pour l'intégrité

Soit le polynôme irréductible :

$$P(X) = X^{128} + X^7 + X^2 + X + 1 \in \mathbb{F}_2[X]$$

qui définit le corps $\mathbb{F}_{2^{128}} = \mathbb{F}_2[X]/(P(X))$.

Définition 6.24 (Sous-clé de hachage H). Pour une clé AES K , on définit la sous-clé de hachage :

$$H = E_K(0^{128}) \in \{0, 1\}^{128}$$

Cette valeur est identifiée à un élément de $\mathbb{F}_{2^{128}}$ et utilisée dans toutes les multiplications GHASH.

Définition 6.25 (Fonction d'incrémentation GCM). Soit $\text{incr}_s : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$ qui incrémente les s bits de droite comme un entier big-endian. Pour GCM, $s = 32$:

$$\text{incr}_{32}(X) = \text{MSB}_{96}(X) \parallel \text{INT}_4^{-1}((\text{val}_{\text{be}}(\text{LSB}_{32}(X)) + 1) \bmod 2^{32})$$

Définition 6.26 (Génération des compteurs). À partir du nonce $N \in \{0, 1\}^{96}$, on définit :

$$J_0 = \begin{cases} N \parallel 0^{31}1 & \text{si } \|N\| = 96 \\ \text{GHASH}_H(\varepsilon, N) & \text{sinon} \end{cases}$$

Les compteurs successifs sont :

$$J_i = \text{incr}_{32}(J_{i-1}) \quad \text{pour } i = 1, 2, \dots$$

Définition 6.27 (Fonction GCTR utilisant AES). Pour une clé K , un compteur initial $ICB \in \{0, 1\}^{128}$, et une entrée $X \in \{0, 1\}^*$:

$$\begin{aligned} \text{GCTR}_K(ICB, X) = & \\ \text{Si } X = \varepsilon : & \varepsilon \\ \text{Sinon} : & \text{Soit } m = \lceil \|X\|/128 \rceil \\ & \text{Pour } i = 1 \text{ à } m - 1 : \\ & \quad CB_i = \text{incr}_{32}(CB_{i-1}) \text{ avec } CB_0 = ICB \\ & \quad Y_i = X_i \oplus \mathbf{E}_K(\mathbf{CB}_i) \quad \text{AES utilisé ici} \\ & \quad CB_m = \text{incr}_{32}(CB_{m-1}) \\ & \quad Y_m = X_m \oplus \text{MSB}_{\|X_m\|}(\mathbf{E}_K(\mathbf{CB}_m)) \quad \text{AES utilisé ici} \\ & \text{Résultat } Y_1 \parallel \dots \parallel Y_m \end{aligned}$$

Définition 6.28 (Fonction GHASH utilisant $H = E_K(0^{128})$). Soit $H = \mathbf{E_K}(0^{128}) \in \mathbb{F}_{2^{128}}$. Pour des données $X \in \{0, 1\}^*$:

$$\begin{aligned} \text{GHASH}_H(X) = & \\ & \text{Soit } X_1, \dots, X_m \leftarrow \text{Partition}_{128}(X) \\ & Y_0 = 0^{128} \\ & \text{Pour } i = 1 \text{ à } m : \\ & \quad Y_i = (Y_{i-1} \oplus X_i) \cdot H \quad \text{dans } \mathbb{F}_{2^{128}} \\ & \text{Résultat } Y_m \end{aligned}$$

Équivalent polynomial :

$$\text{GHASH}_H(X_1, \dots, X_m) = X_1 \cdot H^m \oplus X_2 \cdot H^{m-1} \oplus \dots \oplus X_m \cdot H$$

Définition 6.29 (Algorithme de chiffrement GCM). Pour K, N, A, M :

1. $H = \mathbf{E_K}(0^{128})$ **AES utilisé ici**
2. $J_0 = \begin{cases} N \parallel 0^{31}1 & \text{si } \|N\| = 96 \\ \text{GHASH}_H(\varepsilon, N) & \text{sinon} \end{cases}$
3. $C = \text{GCTR}_K(\text{incr}_{32}(J_0), M)$ **AES utilisé dans GCTR**
4. $u = 128 \cdot \lceil \|C\|/128 \rceil - \|C\|$
5. $v = 128 \cdot \lceil \|A\|/128 \rceil - \|A\|$
6. $S = \text{GHASH}_H(A \parallel 0^v \parallel C \parallel 0^u \parallel \text{len}_{64}(A) \parallel \text{len}_{64}(C))$
7. $T = \text{MSB}_t(\text{GCTR}_K(J_0, S))$ **AES utilisé dans GCTR**
8. Retourner (C, T)

où $\text{len}_{64}(X)$ représente la longueur de X en bits sur 64 bits.

Définition 6.30 (Algorithme de déchiffrement GCM). Pour K, N, A, C, T :

1. $H = \mathbf{E_K}(0^{128})$ **AES utilisé ici**
2. $J_0 = \begin{cases} N \parallel 0^{31}1 & \text{si } \|N\| = 96 \\ \text{GHASH}_H(\varepsilon, N) & \text{sinon} \end{cases}$
3. $P = \text{GCTR}_K(\text{incr}_{32}(J_0), C)$ **AES utilisé dans GCTR**
4. $u = 128 \cdot \lceil \|C\|/128 \rceil - \|C\|$
5. $v = 128 \cdot \lceil \|A\|/128 \rceil - \|A\|$
6. $S = \text{GHASH}_H(A \parallel 0^v \parallel C \parallel 0^u \parallel \text{len}_{64}(A) \parallel \text{len}_{64}(C))$
7. $T' = \text{MSB}_t(\text{GCTR}_K(J_0, S))$ **AES utilisé dans GCTR**
8. Si $T' = T$ retourner P , sinon \perp

Théorème 6.31 (Correction d'AES-GCM). Pour toute clé K , tout nonce N , toutes données A , et tout message M :

$$\text{AES-GCM}_K^{-1}(N, A, \text{AES-GCM}_K(N, A, M)) = M$$

Esquisse. La correction découle de :

- La réversibilité de GCTR qui utilise E_K de façon déterministe
- La linéarité de GHASH dans $\mathbb{F}_{2^{128}}$ utilisant $H = E_K(0^{128})$

- L'utilisation cohérente des mêmes compteurs J_i et de la même clé K

□

Remarque 6.32 (Double usage d'AES dans GCM). AES est utilisé de deux manières différentes :

1. **Confidentialité** : Dans GCTR, $E_K(J_i)$ génère une séquence de masquage pour chiffrer le message
2. **Authentification** : $E_K(0^{128})$ produit H , la base de toutes les opérations GHASH

Cette dualité fait de GCM un mode authentifié efficace utilisant une seule primitive cryptographique.

Remarque 6.33 (Sécurité GCM). La sécurité d'AES-GCM repose sur :

- L'indistinguabilité de E_K comme permutation pseudo-aléatoire (PRP)
- L'unicité des nonces (pour éviter la réutilisation des compteurs)
- La résistance aux collisions de GHASH dans $\mathbb{F}_{2^{128}}$

Remarque 6.34 (Utilisation dans le schéma k -sur- n). Dans notre contexte :

- AES-256 est utilisé avec des clés de 256 bits
- Nonces de 96 bits garantissant $J_0 = N \parallel 0^{31}1$
- Données authentifiées associées généralement vides : $A = \varepsilon$
- Tags de 128 bits ($t = 128$)

6.8 Dénombrement des permutations vs. clés

Remarque 6.35 (Injectivité pratique de l'application clépermutation). L'espace des blocs possibles est $\{0, 1\}^{128}$, qui contient

$$N = 2^{128}$$

éléments. L'ensemble de toutes les permutations de $\{0, 1\}^{128}$ est donc le groupe symétrique

$$\text{Sym}(\{0, 1\}^{128}) \simeq \mathfrak{S}_{2^{128}},$$

qui contient

$$|\text{Sym}(\{0, 1\}^{128})| = (2^{128})!$$

permutations possibles.

L'espace des clés AES-256 contient $|\mathcal{K}| = 2^{256}$ clés possibles. L'application

$$\Phi : \mathcal{K} \rightarrow \text{Sym}(\{0, 1\}^{128}), \quad K \mapsto E_K$$

ne peut donc, au plus, réaliser que 2^{256} permutations parmi les $(2^{128})!$ permutations théoriquement possibles sur $\{0, 1\}^{128}$.

En utilisant l'approximation de Stirling $\log_2(n!) \approx n(\log_2 n - \log_2 e)$,

$$\log_2((2^{128})!) \approx 2^{128}(\log_2(2^{128}) - \log_2 e) = 2^{128}(128 - \log_2 e) \approx 2^{128} \cdot 126,56,$$

alors que

$$\log_2 |\mathcal{K}| = \log_2(2^{256}) = 256.$$

On a donc

$$\log_2 |\text{Sym}(\{0, 1\}^{128})| \approx 126,56 \cdot 2^{128} \gg 256 = \log_2 |\mathcal{K}|,$$

ce qui signifie que le nombre de permutations de blocs possibles est astronomiquement plus grand que le nombre de clés AES-256 possibles.

Si l'on modélise AES-256 comme une *permutation pseudo-aléatoire* (PRP), la probabilité de collision (existence de $K \neq K'$ tels que $E_K = E_{K'}$) est d'environ $2^{-\Omega(2^{128})}$, totalement négligeable à toute échelle pratique.

7 Construction globale du schéma k -sur- n

7.1 Paramètres et constantes

Définition 7.1 (Paramètres globaux). On fixe :

- un entier $n \geq 2$ (nombre de participants) ;
- un entier k avec $2 \leq k \leq n$ (seuil) ;
- le corps \mathbb{F}_P avec $P = 2^{521} - 1$;
- la fonction de hachage h (par exemple SHA-256) ;
- les fonctions dérivées HMAC_h , HKDF_h , PBKDF2_h ;
- un schéma de signature Ed25519 sur $(E(\mathbb{F}_q), \langle B \rangle)$;
- un schéma RSA-OAEP pour des modules de taille 4096 bits ;
- AES-GCM avec blocs de 128 bits et clés de 256 bits.

Définition 7.2 (Constantes de contexte et sels). On fixe les constantes suivantes :

- $\text{info}_{\text{Ed}} \in \{0, 1\}^*$, chaîne ASCII (par exemple "ed25519-master-key") ;
- $\text{info}_{\text{RSA}} \in \{0, 1\}^*$, (par exemple "rsa-wrap-key") ;
- un sel HKDF public pour Ed25519, $\text{salt}_{\text{Ed}} \in \{0, 1\}^*$, typiquement l'ASCII de "ed25519-salt" ;
- pour chaque cérémonie, un sel HKDF public pour RSA, $W \in \{0, 1\}^{\ell_{\text{wrap}}}$, tiré uniformément ;
- pour chaque participant i , un sel PBKDF2 individuel $S_i \in \{0, 1\}^{\ell_S}$ (avec ℓ_S fixé, par exemple 128 bits), tiré uniformément ;
- un nombre d'itérations $c \in \mathbb{N}^*$ pour PBKDF2 ;
- pour RSA-OAEP, un label $L \in \{0, 1\}^*$ (souvent vide).

7.2 Cérémonie initiale

Définition 7.3 (Encodage des éléments de \mathbb{F}_P). Soit $\phi : \mathbb{F}_P \rightarrow \{0, 1\}^{521}$ l'isomorphisme qui à $a \in \mathbb{F}_P$ associe sa représentation binaire big-endian sur $\lceil \log_2 P \rceil = 521$ bits.

Définition 7.4 (Étape de génération). La cérémonie initiale réalise les étapes suivantes :

- 1) Tirer $S \in \{0, 1\}^{256}$ uniformément et définir $s = \text{val}_{\text{be}}(S) \bmod P \in \mathbb{F}_P$.
- 2) Appliquer $\text{Share}(s)$ pour obtenir les parts $(x_i, y_i)_{1 \leq i \leq n}$.
- 3) Dériver la clé

$$K_{\text{Ed}} = \text{HKDF}_h(S, \text{salt}_{\text{Ed}}, \text{info}_{\text{Ed}}, 32) \in \{0, 1\}^{256}.$$

- 4) Dériver à partir de K_{Ed} la paire de clés Ed25519 (a, A) .
- 5) Générer une paire RSA (n, e, d) avec n de 4096 bits.
- 6) Tirer $W \in \{0, 1\}^{\ell_{\text{wrap}}}$ et dériver la clé

$$K_{\text{RSA}} = \text{HKDF}_h(S, W, \text{info}_{\text{RSA}}, L_{\text{RSA}}) \in \{0, 1\}^{8L_{\text{RSA}}}$$

avec $L_{\text{RSA}} = 32$ (clé AES-256 de 32 octets).

- 7) Encoder la clé privée RSA en un mot binaire M_{RSA} (format PEM ou DER), puis calculer

$$(C_{\text{RSA}}, T_{\text{RSA}}) = \text{AES-GCM}_{K_{\text{RSA}}}(N_{\text{RSA}}, M_{\text{RSA}})$$

pour un nonce $N_{\text{RSA}} \in \{0, 1\}^{96}$.

- 8) Stocker $(W, N_{\text{RSA}}, C_{\text{RSA}}, T_{\text{RSA}})$ dans `rsa_wrapped.json`.

7.3 Protection des parts

Définition 7.5 (Sérialisation des parts). Pour chaque part (x_i, y_i) , on construit un dictionnaire JSON contenant les champs `x`, `y`, `k`, `n`, et `pub_hash`. On encode ce dictionnaire en UTF-8 pour obtenir le message à chiffrer $M_i \in \{0, 1\}^*$.

Définition 7.6 (Dérivation des clés individuelles). Pour chaque participant i , on fixe un mot de passe $P_i \in \{0, 1\}^*$ et on définit

$$K_i = \text{PBKDF2}_h(P_i, S_i, c, L_i) \in \{0, 1\}^{8L_i},$$

où L_i est choisi en fonction de la clé AES à dériver (par exemple $L_i = 32$ pour une clé AES-256).

Définition 7.7 (Chiffrement des parts). Pour une part sérialisée M_i , on calcule

$$(C_i, T_i) = \text{AES-GCM}_{K_i}(N_i, M_i)$$

avec un nonce $N_i \in \{0, 1\}^{96}$. Le tuple (S_i, N_i, C_i, T_i) est stocké dans la partie personnelle du coffre du participant i .

7.4 Chiffrement hybride des fichiers

Définition 7.8 (Chiffrement hybride d'un fichier). Soit un fichier $F \in \{0, 1\}^*$. On procède comme suit :

- 1) Tirer une clé de session $K_{\text{AES}} \in \{0, 1\}^{256}$.
- 2) Tirer un nonce $N \in \{0, 1\}^{96}$.
- 3) Calculer $(C, T) = \text{AES-GCM}_{K_{\text{AES}}}(N, F)$.
- 4) Calculer $E_K = \text{RSA-OAEP}_{(n,e)}(K_{\text{AES}}; r)$ pour un aléa $r \in \{0, 1\}^{k_1}$.

Le quadruplet (E_K, N, C, T) est le chiffrement hybride de F .

8 Correspondance avec les scripts Tails v1

Cette section établit la correspondance entre les objets formalisés ci-dessus et les éléments concrets (scripts, fichiers) de la procédure Tails v1.

8.1 Secret maître et partage Shamir

- Le secret $S \in \{0, 1\}^{256}$ est généré par `os.urandom(32)` dans le script `ceremony_generate.py` (variable `S_int`).
- La conversion $s = \text{val}_{\text{be}}(S) \bmod P \in \mathbb{F}_P$ est implicite dans l'implémentation Python.
- Les partages (x_i, y_i) sont obtenus par évaluation d'un polynôme $f \in \mathcal{P}_{<k}$ via la fonction `shamir_split`.
- La reconstruction utilise l'interpolation de Lagrange dans `shamir_reconstruct` avec calcul explicite des coefficients λ_j .

8.2 Clés dérivées, sels HKDF et Ed25519

- La clé $K_{\text{Ed}} = \text{HKDF}_h(S, \text{salt}_{\text{Ed}}, \text{info}_{\text{Ed}}, 32)$ est instanciée via :

```
HKDF(SHA256, 32, salt=b"ed25519-salt",  
      info=b"ed25519-master-key")
```

- De même, $K_{\text{RSA}} = \text{HKDF}_h(S, W, \text{info}_{\text{RSA}}, 32)$ utilise `info=b"rsa-wrap-key"` avec un sel aléatoire W stocké dans `rsa_wrapped.json`.
- La paire (a, A) est construite via `Ed25519PrivateKey.from_private_bytes(ed_seed)`, où `ed_seed` est K_{Ed} . L'API effectue le clamping et calcule $A = aB$.

8.3 PBKDF2, sels S_i et chiffrement des parts

- Pour chaque participant i , le script tire un sel S_i (16 octets) et dérive $K_i = \text{PBKDF2}_h(P_i, S_i, c, 32)$ avec $c = 501000$ itérations.
- Les parts sont sérialisées en JSON avec les champs `x`, `y`, `k`, `n`, et `pub_hash`, puis chiffrées par AES-GCM.
- L'enveloppe chiffrée inclut les métadonnées nécessaires pour la vérification lors de la reconstruction.

8.4 RSA, OAEP et chiffrement hybride

- La paire RSA (n, e, d) est générée avec $n = 4096$ bits et $e = 65537$.
- La clé privée RSA est sérialisée en PEM puis chiffrée sous AES-GCM avec K_{RSA} , produisant `rsa_wrapped.json`.
- Le script `rsa_hybrid_encrypt.py` implémente le chiffrement hybride :
 - Génération de $K_{\text{AES}} \in \{0, 1\}^{256}$
 - Chiffrement de F par $\text{AES-GCM}_{K_{\text{AES}}}(N, F)$
 - Chiffrement de K_{AES} par $\text{RSA-OAEP}_{(n,e)}$
- Le déchiffrement dans `kofn_rsa_decrypt.py` inverse ces étapes après reconstruction de S et déverrouillage de la clé RSA.

8.5 Signature Ed25519 avec contexte

- Les signatures utilisent un contexte déterministe :

`contexte = "kofn-ed25519-v1" || SHA256(A) || M`

pour éviter la réutilisation hors contexte.

- La vérification dans `verify_ed25519.py` utilise le même contexte, assurant l'authenticité même avec la même clé publique.

8.6 Gestion sécurisée de la mémoire

- Les secrets éphémères (S , seeds) sont stockés dans des `bytearray` pour permettre l'effacement explicite via `secure_wipe`.
- Cette mesure atténue les risques d'exposition en RAM après usage.

8.7 Résumé sur les sels publics

- Sel HKDF Ed25519 saltEd :
 - valeur : chaîne ASCII constante (par exemple `"ed25519-salt"`);
 - génération : aucun tirage, valeur codée en dur dans le script;
 - stockage : implicite dans le code, public et identique pour toutes les cérémonies.
- Sel HKDF RSA (wrap salt) W :
 - valeur : bitstring aléatoire `wrap_salt` généré par `os.urandom(WRAP_SALT_SIZE)`;
 - génération : une fois par cérémonie de génération de la clé RSA;
 - stockage : champ `"salt"` dans `rsa_wrapped.json`, encodé en Base64, public.
- Sels PBKDF2 S_i :
 - valeur : bitstrings aléatoires individuels, un par participant, générés par `os.urandom(SALT_SIZE)`;
 - génération : lors de la création ou de la mise à jour de la protection de la part du participant i ;
 - stockage : dans la partie personnelle du coffre du participant, aux côtés de (N_i, C_i, T_i) , public mais lié au participant.

9 Résumé des propriétés de sécurité

Définition 9.1 (Modèle de sécurité). On considère un adversaire \mathcal{A} ayant accès aux oracles :

- $\mathcal{O}_{\text{Share}}$: renvoie jusqu'à $t < k$ parts
- $\mathcal{O}_{\text{Sign}}$: signe des messages avec la clé Ed25519
- $\mathcal{O}_{\text{Decrypt}}$: déchiffre des textes chiffrés

Théorème 9.2 (Confidentialité conditionnelle). Soit \mathcal{A} un adversaire polynomial n'ayant accès qu'à $t < k$ parts. Alors pour toute fonction f calculable en temps polynomial :

$$|\mathbb{P}[\mathcal{A}(f(S)) = 1] - \mathbb{P}[\mathcal{A}(f(U)) = 1]| \leq \epsilon(\lambda)$$

où U est uniforme dans $\{0, 1\}^{256}$ et ϵ est négligeable dans le paramètre de sécurité λ .

Théorème 9.3 (Intégrité des signatures et des chiffrés). Sous les hypothèses que HKDF, PBKDF2, AES-GCM et RSA-OAEP sont cryptographiquement sûrs, il est computationnellement difficile pour un adversaire polynomial de forger une signature Ed25519 valide ou de modifier un texte chiffré AES-GCM sans être détecté.

Remarque 9.4. Une preuve complète de sécurité nécessiterait un modèle d'adversaire précis (oracles, ressources de calcul) et l'utilisation de techniques de réduction dans des modèles comme l'oracle aléatoire. On ne la développe pas ici.

A Annexe A — Corps finis et notation \mathbb{F}_{2^n}

Cette annexe rassemble les compléments d'algèbre nécessaires pour justifier entièrement l'utilisation de la notation \mathbb{F}_{2^n} (et, plus généralement, \mathbb{F}_{p^n}) dans le reste du document. Toutes les démonstrations reposent sur des arguments élémentaires portant sur des objets finis : polynômes sur \mathbb{F}_p , groupes abéliens finis, dénombrements. Dans cette annexe, on détaille les éléments de théorie des corps finis qui justifient complètement le Théorème 1.31. On travaille toujours avec un nombre premier p et des corps de caractéristique p .

A.1 Caractéristique et sous-corps premier

Définition A.1 (Caractéristique d'un corps). Soit K un corps. La *caractéristique* de K est le plus petit entier $p \geq 1$ tel que

$$\underbrace{1 + 1 + \cdots + 1}_{p \text{ fois}} = 0$$

s'il existe, et 0 sinon.

Proposition A.2. Si K est un corps fini, alors sa caractéristique est un nombre premier p .

Démonstration. Comme K est fini, le sous-groupe additif engendré par 1 est lui aussi fini. Il existe donc un entier $p \geq 1$ minimal tel que $p \cdot 1 = 0$. On montre que p est premier : si $p = ab$ avec $1 < a, b < p$, on a

$$0 = p \cdot 1 = (ab) \cdot 1 = a \cdot (b \cdot 1),$$

et comme K est sans diviseur de zéro (c'est un corps), cela implique $b \cdot 1 = 0$ ou $a \cdot 1 = 0$, en contradiction avec la minimalité de p . \square

Proposition A.3 (Sous-corps premier). Soit K un corps de caractéristique $p > 0$. L'application

$$\varphi : \mathbb{F}_p \longrightarrow K, \quad \bar{n} \longmapsto n \cdot 1_K$$

est un homomorphisme injectif de corps. Son image est un sous-corps de K , appelé *sous-corps premier* de K , isomorphe à \mathbb{F}_p .

Démonstration. L'application est bien définie : dans \mathbb{F}_p , on a $\bar{n} = \bar{m}$ si et seulement si p divise $n - m$. Dans ce cas, il existe $q \in \mathbb{Z}$ tel que $n - m = pq$, et donc dans K ,

$$(n - m) \cdot 1_K = pq \cdot 1_K = p \cdot (q \cdot 1_K) = 0$$

puisque K est de caractéristique p . On a donc bien $n \cdot 1_K = m \cdot 1_K$, ce qui montre que φ ne dépend pas du représentant choisi.

Pour tout $\bar{n}, \bar{m} \in \mathbb{F}_p$, on vérifie directement

$$\varphi(\bar{n} + \bar{m}) = (n + m) \cdot 1_K = n \cdot 1_K + m \cdot 1_K = \varphi(\bar{n}) + \varphi(\bar{m}),$$

et

$$\varphi(\bar{n} \bar{m}) = (nm) \cdot 1_K = (n \cdot 1_K)(m \cdot 1_K) = \varphi(\bar{n}) \varphi(\bar{m}),$$

ainsi que $\varphi(1) = 1_K$. L'application φ est donc un homomorphisme de corps.

Pour l'injectivité, supposons $\varphi(\bar{n}) = 0$. Alors $n \cdot 1_K = 0$. Par définition de la caractéristique p , cela implique que p divise n , donc $\bar{n} = 0$ dans \mathbb{F}_p . Ainsi, $\ker(\varphi) = \{0\}$ et φ est injective. Son image est donc un sous-corps de K isomorphe à \mathbb{F}_p , appelé sous-corps premier de K . \square

Ainsi, tout corps fini K contient un sous-corps isomorphe à \mathbb{F}_p , et on peut voir K comme une extension finie de \mathbb{F}_p .

A.2 Cardinal d'un corps fini

Proposition A.4. Soit K un corps fini de caractéristique p , et soit k le sous-corps premier de K (isomorphe à \mathbb{F}_p). Alors K est un k -espace vectoriel de dimension finie n , et

$$|K| = p^n.$$

Démonstration. Par la Proposition A.3, k est un sous-corps de K , isomorphe à \mathbb{F}_p . On définit sur K une structure de k -espace vectoriel en posant :

- pour $x, y \in K$, la somme vectorielle est l'addition dans K ;
- pour $\lambda \in k$ (donc $\lambda \in K$) et $x \in K$, la multiplication scalaire est donnée par

$$\lambda \cdot x := \lambda x,$$

où le produit est celui de K .

Les axiomes d'espace vectoriel se vérifient alors directement à partir des axiomes de corps dans K :

- la distributivité de la multiplication dans K donne $\lambda \cdot (x + y) = \lambda x + \lambda y$ et $(\lambda + \mu) \cdot x = \lambda x + \mu x$;
- l'associativité de la multiplication donne $(\lambda \mu) \cdot x = \lambda \cdot (\mu \cdot x)$;
- l'élément neutre $1_k = 1_K$ vérifie $1_k \cdot x = x$ pour tout $x \in K$.

Ainsi, K est bien un k -espace vectoriel.

Comme K est fini, il existe une base finie (e_1, \dots, e_n) de K sur k . Tout élément $x \in K$ s'écrit alors de manière unique comme combinaison linéaire

$$x = \lambda_1 e_1 + \dots + \lambda_n e_n, \quad \lambda_i \in k.$$

Il y a exactement p^n n -uplets $(\lambda_1, \dots, \lambda_n)$ puisque $|k| = p$. Donc $|K| = p^n$. \square

Cela justifie l'écriture « $|K| = p^n$ » et le fait de noter un corps fini K de cardinal p^n par \mathbb{F}_{p^n} .

A.3 Frobenius et polynôme $X^{p^n} - X$

Théorème A.5 (Théorème fondamental de l'arithmétique). Tout entier $n \geq 2$ peut s'écrire comme un produit fini de nombres premiers. De plus, cette écriture est unique à l'ordre près des facteurs.

Démonstration. Nous divisons la preuve en deux parties : existence puis unicité.

Existence. Nous montrons par récurrence sur $n \geq 2$ que tout entier n s'écrit comme un produit de nombres premiers (en autorisant le cas d'un seul facteur premier).

- Pour $n = 2$, l'entier 2 est premier, la propriété est donc vraie.
- Supposons la propriété vraie pour tous les entiers m tels que $2 \leq m < n$, et montrons-la pour n .
 - Si n est premier, alors n est déjà un produit de nombres premiers (un seul facteur).
 - Si n n'est pas premier, il existe des entiers a, b tels que $1 < a < n$, $1 < b < n$ et $n = ab$. Par hypothèse de récurrence, a et b s'écrivent chacun comme produit de nombres premiers. Le produit $n = ab$ est alors lui-même un produit de nombres premiers.

Par récurrence, tout entier $n \geq 2$ s'écrit comme produit de nombres premiers.

Unicité. Supposons que l'on ait deux écritures de n en produit de nombres premiers :

$$n = p_1 p_2 \cdots p_r = q_1 q_2 \cdots q_s,$$

où les p_i et les q_j sont des nombres premiers (pas nécessairement distincts). Nous allons montrer que $r = s$ et, à permutation près, $p_i = q_i$ pour tout i .

Comme p_1 est premier et divise le produit $q_1 \cdots q_s$, le Théorème 1.3 (lemme d'Euclide) implique que p_1 divise l'un des q_j . Or chacun des q_j est premier, donc si un nombre premier p_1 divise q_j , on a nécessairement $p_1 = q_j$. En renommant si besoin les q_j , on peut supposer $p_1 = q_1$.

En simplifiant par $p_1 = q_1$ (qui est non nul), on obtient

$$\frac{n}{p_1} = p_2 \cdots p_r = q_2 \cdots q_s.$$

Si $r = 1$, alors $n = p_1$ est premier et il n'y a rien d'autre à prouver. Sinon, l'entier n/p_1 est ≥ 2 et strictement plus petit que n . On applique le même raisonnement à n/p_1 (récurrence sur le nombre total de facteurs premiers) et on conclut que $r = s$ et que, à réordonnement près, $p_i = q_i$ pour tout i .

Ainsi, la décomposition de n en produit de nombres premiers est unique à l'ordre près des facteurs. □

Proposition A.6 (Endomorphisme de Frobenius). Soit K un corps de caractéristique p . L'application

$$\varphi : K \longrightarrow K, \quad x \longmapsto x^p$$

est un homomorphisme de corps, appelé *endomorphisme de Frobenius*.

Démonstration. Pour $x, y \in K$, on a dans \mathbb{Z}

$$(x + y)^p = \sum_{k=0}^p \binom{p}{k} x^k y^{p-k}.$$

Pour $0 < k < p$, montrons que le coefficient binomial $\binom{p}{k}$ est multiple de p . Dans \mathbb{Z} , on dispose de l'égalité

$$\binom{p}{k} k! = p(p-1)(p-2) \cdots (p-k+1).$$

Le produit du membre de droite contient le facteur p , donc p divise $\binom{p}{k} k!$.

Montrons maintenant que p ne divise pas $k!$. Par le Théorème A.5, on peut écrire la décomposition en facteurs premiers de $k!$ sous la forme

$$k! = \prod_{i=1}^r \ell_i^{\alpha_i},$$

où les ℓ_i sont des nombres premiers et $\alpha_i \geq 1$. Chaque ℓ_i divise l'un des entiers $1, 2, \dots, k$, donc $\ell_i \leq k < p$. Si l'on supposait que p divise $k!$, alors p devrait être égal à l'un des ℓ_i , ce qui entraînerait une contradiction avec l'hypothèse $k < p$. Ainsi, p ne divise pas $k!$.

Comme p divise $\binom{p}{k} k!$ et que p ne divise pas $k!$, les seuls diviseurs positifs possibles de p étant 1 et p , on a donc $\gcd(p, k!) = 1$. Le Théorème 1.3 s'applique alors avec $a = p$ et $b = k!$, ce qui implique que p divise $\binom{p}{k}$.

Dans un corps K de caractéristique p , cela signifie que l'image de $\binom{p}{k}$ est nulle : si $\binom{p}{k} = pq$ dans \mathbb{Z} , alors dans K

$$\binom{p}{k} \cdot 1_K = (pq) \cdot 1_K = p \cdot (q \cdot 1_K) = 0,$$

puisque $p \cdot 1_K = 0$. Pour $0 < k < p$, on a donc bien $\binom{p}{k} = 0$ dans K .

Dans le développement binomial de $(x+y)^p$, tous les termes intermédiaires disparaissent donc dans K , et il reste

$$(x+y)^p = x^p + y^p.$$

La compatibilité à la multiplication est immédiate :

$$(xy)^p = x^p y^p,$$

et enfin $\varphi(1) = 1^p = 1$, donc φ respecte l'addition, la multiplication et l'élément neutre. C'est donc un homomorphisme de corps. \square

Proposition A.7. Si K est un corps fini de caractéristique p , l'endomorphisme de Frobenius $\varphi : x \mapsto x^p$ est bijectif.

Démonstration. Un homomorphisme de corps est injectif dès que son noyau est réduit à $\{0\}$. Mais si $\varphi(x) = 0$, alors $x^p = 0$ implique $x = 0$ dans un corps. Ainsi φ est injectif. Comme K est fini, une application injective est automatiquement surjective, donc φ est bijective. \square

Corollaire A.8. Soit K un corps fini de cardinal p^n . Pour tout $x \in K$, on a

$$x^{p^n} = x.$$

En identifiant \mathbb{F}_p avec le sous-corps premier de K (Proposition A.3), on peut considérer $X^{p^n} - X$ comme un polynôme de $\mathbb{F}_p[X] \subseteq K[X]$. Autrement dit, tout élément de K est racine du polynôme $X^{p^n} - X \in \mathbb{F}_p[X]$.

Démonstration. Notons $q = |K| = p^n$. Pour $x = 0$, on a trivialement $0^q = 0$.

Supposons maintenant $x \in K^\times$. Le groupe multiplicatif K^\times a pour cardinal $q - 1$. Par le théorème de Lagrange sur les groupes finis, l'ordre de x divise $q - 1$, donc $x^{q-1} = 1$. En multipliant par x , on obtient

$$x^q = x,$$

c'est-à-dire $x^{p^n} = x$.

Ainsi, pour tout $x \in K$ (nul ou non nul), on a bien $x^{p^n} = x$, ce qui signifie que tout élément de K est racine du polynôme $X^{p^n} - X \in \mathbb{F}_p[X]$. \square

Lemme A.9 (Corps de décomposition d'un polynôme). Soit K un corps et soit $P(X) \in K[X]$ un polynôme non constant. Il existe une extension de corps $L \supseteq K$ dans laquelle P se scinde dans $L[X]$ en produit de facteurs linéaires.

Démonstration. On procède par récurrence sur le degré $d = \deg(P) \geq 1$.

Si $d = 1$, le polynôme P est déjà linéaire et se scinde sur K ; on peut prendre $L = K$.

Supposons $d \geq 2$ et le résultat vrai pour tous les polynômes de degré strictement inférieur à d . Comme $K[X]$ est un anneau principal euclidien, tout polynôme non constant admet un facteur irréductible non constant. Choisissons un tel facteur irréductible $f_1(X)$ de $P(X)$ dans $K[X]$.

On considère alors le quotient

$$K_1 = K[X]/(f_1).$$

C'est un corps, et si l'on note α_1 la classe de X dans K_1 , on a $f_1(\alpha_1) = 0$ dans K_1 . Le polynôme P admet donc une racine α_1 dans l'extension K_1 de K .

Nous regardons maintenant P comme élément de $K_1[X]$. Comme $f_1(X)$ est un facteur de P et admet α_1 comme racine, on peut écrire

$$P(X) = (X - \alpha_1)^m Q_1(X),$$

où $m \geq 1$ et $Q_1(X) \in K_1[X]$ ne s'annule pas en $X = \alpha_1$. Le degré de Q_1 est strictement inférieur à d .

Par hypothèse de récurrence appliquée au corps K_1 et au polynôme Q_1 , il existe une extension de corps L de K_1 dans laquelle Q_1 se scinde en produit de facteurs linéaires. Dans ce même corps L , le facteur $(X - \alpha_1)^m$ est déjà produit de facteurs linéaires. Ainsi, dans $L[X]$, le polynôme P s'écrit comme produit de facteurs linéaires.

Comme $K \subseteq K_1 \subseteq L$, le corps L est bien une extension de K dans laquelle P se scinde. \square

A.4 Polynôme $X^{p^n} - X$ et polynômes irréductibles

Dans cette sous-section, on fixe un entier $n \geq 1$. D'après le Lemme A.9 appliqué au polynôme $X^{p^n} - X \in \mathbb{F}_p[X]$, il existe une extension de corps L de \mathbb{F}_p dans laquelle $X^{p^n} - X$ se scinde en produit de facteurs linéaires. Nous travaillerons dans une telle extension L (que l'on garde fixée pour ce n).

Proposition A.10. Dans l'extension L fixée ci-dessus, l'ensemble des racines du polynôme $X^{p^n} - X \in \mathbb{F}_p[X]$ forme un sous-corps S de L contenant \mathbb{F}_p . En particulier, S est un corps fini de caractéristique p .

Démonstration. Posons

$$S = \{x \in L \mid x^{p^n} = x\}.$$

Par définition, S est exactement l'ensemble des racines de $X^{p^n} - X$ dans L .

Montrons d'abord que S est stable par les opérations de corps. Soient $x, y \in S$. Alors

$$(x + y)^{p^n} = x^{p^n} + y^{p^n} = x + y,$$

où l'on a utilisé la Proposition A.6 (endomorphisme de Frobenius) itérée n fois. De même,

$$(xy)^{p^n} = x^{p^n} y^{p^n} = xy.$$

Enfin, si $x \in S$ est non nul, alors x^{-1} vérifie

$$(x^{-1})^{p^n} = (x^{p^n})^{-1} = x^{-1}.$$

Ainsi, S est stable par addition, multiplication et passage à l'inverse (pour les éléments non nuls), et contient 0 et 1. C'est donc un sous-corps de L .

Comme S est constitué des racines d'un polynôme de degré p^n , le Lemme 1.26 implique que S est fini (et même $|S| \leq p^n$). En particulier, S est un corps fini de caractéristique p . \square

Proposition A.11. La factorisation de $X^{p^n} - X$ dans $\mathbb{F}_p[X]$ est

$$X^{p^n} - X = \prod_{\substack{f \text{ irréductible,} \\ f \in \mathbb{F}_p[X], \deg(f) \mid n}} f(X),$$

le produit portant sur tous les polynômes irréductibles unitaires dont le degré divise n , chaque facteur apparaissant une seule fois.

Démonstration. Soit $f \in \mathbb{F}_p[X]$ irréductible unitaire et soit α une racine de f dans une extension de corps de \mathbb{F}_p (par exemple dans le corps quotient $\mathbb{F}_p[X]/(f)$). Alors f est le polynôme minimal de α sur \mathbb{F}_p et $\mathbb{F}_p(\alpha)$ est un corps de cardinal p^d , où $d = \deg(f)$ (cf. Proposition A.4). On a $\alpha^{p^d} = \alpha$ (Corollaire A.8 appliqué à $\mathbb{F}_p(\alpha)$), donc α est racine de $X^{p^d} - X$.

Si d divise n , alors $p^d - 1$ divise $p^n - 1$, et on vérifie que

$$X^{p^n} - X = (X^{p^d} - X) \circ (\cdots),$$

ce qui implique que toute racine de $X^{p^d} - X$ est racine de $X^{p^n} - X$. En particulier, α est racine de $X^{p^n} - X$, donc le polynôme minimal f divise $X^{p^n} - X$ dans $\mathbb{F}_p[X]$.

Réciproquement, si f divise $X^{p^n} - X$, toute racine α de f satisfait $\alpha^{p^n} = \alpha$, donc $\alpha \in \mathbb{F}_{p^n}$ et $\mathbb{F}_p(\alpha) \subset \mathbb{F}_{p^n}$, ce qui impose que $|\mathbb{F}_p(\alpha)| = p^d$ divise p^n , donc $d \mid n$.

Enfin, comme $X^{p^n} - X$ n'a pas de racine multiple (sa dérivée est $-1 \neq 0$), il est sans facteur carré, donc chaque polynôme irréductible apparaît au plus une fois dans la factorisation. En regroupant les facteurs associés à chaque degré, on obtient la factorisation annoncée. \square

Corollaire A.12 (Existence de polynômes irréductibles de degré n). Pour tout entier $n \geq 1$, il existe au moins un polynôme irréductible dans $\mathbb{F}_p[X]$ de degré exactement n .

Démonstration. Notons I_d le nombre de polynômes irréductibles unitaires de degré d dans $\mathbb{F}_p[X]$. La Proposition A.11 donne

$$X^{p^n} - X = \prod_{d|n} \prod_{\deg(f)=d} f(X),$$

donc la somme des degrés des facteurs vaut

$$p^n = \deg(X^{p^n} - X) = \sum_{d|n} d I_d.$$

Si I_n était nul, on aurait

$$p^n = \sum_{\substack{d|n \\ d < n}} d I_d \leq \sum_{\substack{d|n \\ d < n}} d p^d < \sum_{k=0}^{n-1} p^k = \frac{p^n - 1}{p - 1} < p^n,$$

ce qui est impossible. Donc $I_n > 0$. □

Lemme A.13 (Somme des valeurs de l'indicatrice d'Euler). Pour tout entier $n \geq 1$, on a

$$\sum_{d|n} \varphi(d) = n.$$

Démonstration. On compte le nombre d'entiers k tels que $1 \leq k \leq n$ de deux façons.

Pour chaque entier $k \in \{1, \dots, n\}$, posons $d = \gcd(k, n)$. Alors d est un diviseur de n , et on peut écrire $k = d \cdot k'$ avec $1 \leq k' \leq n/d$ et $\gcd(k', n/d) = 1$. Inversement, pour tout diviseur $d | n$ et tout entier k' tel que $1 \leq k' \leq n/d$ et $\gcd(k', n/d) = 1$, le produit $k = dk'$ vérifie $1 \leq k \leq n$ et $\gcd(k, n) = d$.

Ainsi, pour chaque diviseur d de n , le nombre d'entiers k entre 1 et n tels que $\gcd(k, n) = d$ est exactement égal au nombre d'entiers k' entre 1 et n/d qui sont premiers avec n/d , c'est-à-dire $\varphi(n/d)$.

En regroupant les entiers k selon la valeur de $\gcd(k, n)$, on obtient donc

$$n = \#\{1, \dots, n\} = \sum_{d|n} \varphi\left(\frac{n}{d}\right).$$

En faisant le changement de variable $e = n/d$, on voit que e parcourt exactement l'ensemble des diviseurs de n , et l'on obtient

$$n = \sum_{e|n} \varphi(e),$$

ce qui est la formule annoncée. □

A.5 Cyclicité du groupe multiplicatif

Lemme A.14. Soit K un corps et soit G un sous-groupe fini du groupe multiplicatif K^\times . Alors G est un groupe cyclique.

Démonstration. Notons $n = |G|$. Pour chaque diviseur d de n , on introduit les ensembles suivants :

- $A_d = \{x \in G \mid \text{l'ordre de } x \text{ est exactement } d\}$ et $a_d = |A_d|$;
- $B_d = \{x \in G \mid \text{l'ordre de } x \text{ divise } d\}$ et $b_d = |B_d|$.

Les ensembles A_d pour $d \mid n$ forment une partition de G , donc

$$n = |G| = \sum_{d \mid n} a_d. \quad (\text{A.1})$$

Par définition, un élément $x \in G$ appartient à B_d si et seulement si son ordre m divise d ; autrement dit,

$$b_d = \sum_{m \mid d} a_m \quad \text{pour tout } d \mid n. \quad (\text{A.2})$$

D'autre part, B_d est exactement l'ensemble des solutions dans G de l'équation $x^d = 1$. Comme $G \subseteq K^\times$, ces solutions sont aussi des solutions dans K du polynôme $X^d - 1 \in K[X]$. Par le Lemme 1.26, ce polynôme a au plus d racines dans K , donc

$$b_d \leq d \quad \text{pour tout } d \mid n. \quad (\text{A.3})$$

Nous allons maintenant montrer, par récurrence sur d , que pour tout diviseur d de n ,

$$a_d \leq \varphi(d), \quad (\text{A.4})$$

où φ désigne l'indicatrice d'Euler.

Pour $d = 1$, le seul élément d'ordre 1 est l'élément neutre 1_G , donc $a_1 = 1 = \varphi(1)$.

Supposons la propriété (A.4) vraie pour tous les diviseurs stricts de d , et montrons-la pour d . En utilisant (A.2), (A.3) et le Lemme A.13, on obtient :

$$b_d = \sum_{m \mid d} a_m = a_d + \sum_{\substack{m \mid d \\ m < d}} a_m \leq a_d + \sum_{\substack{m \mid d \\ m < d}} \varphi(m) \leq d = \sum_{m \mid d} \varphi(m) = \varphi(d) + \sum_{\substack{m \mid d \\ m < d}} \varphi(m).$$

En retranchant $\sum_{m \mid d, m < d} \varphi(m)$ des deux côtés, on obtient bien $a_d \leq \varphi(d)$.

En particulier, pour $d = n$, on a $a_n \leq \varphi(n)$.

Montrons maintenant qu'il existe un élément d'ordre n dans G . Supposons par l'absurde que $a_n = 0$. Alors, comme $a_d \leq \varphi(d)$ pour tout $d \mid n$ et que $\varphi(n) \geq 1$, on a

$$\sum_{d \mid n} a_d \leq \sum_{\substack{d \mid n \\ d < n}} \varphi(d) = \sum_{d \mid n} \varphi(d) - \varphi(n) < \sum_{d \mid n} \varphi(d) = n,$$

où l'on a utilisé le Lemme A.13 pour la dernière égalité. Cela contredit (A.1), qui affirme que $\sum_{d \mid n} a_d = n$.

Ainsi, $a_n > 0$: il existe au moins un élément de G d'ordre n . Un tel élément engendre tout le groupe G , qui est donc cyclique. \square

En particulier, pour un corps fini K de cardinal p^n , le groupe multiplicatif K^\times est cyclique d'ordre $p^n - 1$.

Proposition A.15 (Élément primitif). Soit K un corps fini de cardinal p^n . Il existe un élément $\alpha \in K$ tel que

$$K^\times = \langle \alpha \rangle.$$

En particulier, $K = \mathbb{F}_p(\alpha)$ et le polynôme minimal de α sur \mathbb{F}_p est de degré n .

Démonstration. Par le Lemme A.14, K^\times est cyclique : il existe α tel que $K^\times = \langle \alpha \rangle$. Le sous-corps $\mathbb{F}_p(\alpha)$ contient alors tous les éléments non nuls de K , donc $\mathbb{F}_p(\alpha) = K$. Par la Proposition A.4, le degré $[\mathbb{F}_p(\alpha) : \mathbb{F}_p]$ vaut n , ce qui signifie que le polynôme minimal de α sur \mathbb{F}_p est de degré n . \square

A.6 Unicité de \mathbb{F}_{p^n} à isomorphisme près

On peut maintenant démontrer complètement l'unicité de \mathbb{F}_{p^n} .

Théorème A.16 (Unicité à isomorphisme près). Soient K et L deux corps finis de cardinal p^n . Alors K et L sont isomorphes en tant que corps.

Démonstration. Par la Proposition A.15, il existe $\alpha \in K$ tel que $K = \mathbb{F}_p(\alpha)$ et le polynôme minimal m_α de α sur \mathbb{F}_p est de degré n . D'après la Proposition A.11, tout polynôme irréductible de degré n sur \mathbb{F}_p divise $X^{p^n} - X$. En particulier, m_α divise $X^{p^n} - X$.

Considérons le corps L . Comme $|L| = p^n$, le Corollaire A.8 implique que tout élément de L est racine de $X^{p^n} - X$, donc le polynôme m_α se factorise dans $L[X]$ et admet une racine $\beta \in L$.

Par définition du polynôme minimal, l'application

$$\Phi : \mathbb{F}_p[X]/(m_\alpha) \longrightarrow \mathbb{F}_p(\alpha), \quad \overline{P(X)} \longmapsto P(\alpha)$$

est un isomorphisme de corps, et de même

$$\Psi : \mathbb{F}_p[X]/(m_\alpha) \longrightarrow \mathbb{F}_p(\beta), \quad \overline{P(X)} \longmapsto P(\beta)$$

en est un. On obtient donc un isomorphisme

$$\theta : \mathbb{F}_p(\alpha) \longrightarrow \mathbb{F}_p(\beta), \quad \alpha \longmapsto \beta.$$

Or $\mathbb{F}_p(\alpha) = K$ (Proposition A.15) et $\mathbb{F}_p(\beta)$ est un sous-corps de L contenant \mathbb{F}_p . Son cardinal vaut $p^{\deg(m_\alpha)} = p^n$, donc $\mathbb{F}_p(\beta)$ est de cardinal p^n . Comme L est lui-même de cardinal p^n , on en déduit $\mathbb{F}_p(\beta) = L$. L'isomorphisme θ est donc un isomorphisme de corps $K \simeq L$. \square

Le Théorème A.16, combiné à la construction par quotient $\mathbb{F}_p[X]/(m(X))$ (Corollaire A.12 et Proposition 1.30), donne la version complète du Théorème 1.31 utilisé dans le corps du texte : pour tout $q = p^n$, il existe un corps fini de cardinal q , unique à isomorphisme près, et l'on peut le noter sans ambiguïté \mathbb{F}_{p^n} .

Références

- [1] Philippe Rackette. (2025). *K of N Tails - Documentation and Implementation*. <https://github.com/philipperackette/kofn-tails>.
- [2] Shamir, A. (1979). *How to share a secret*. Communications of the ACM, 22(11), 612-613.
- [3] Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC : Keyed-Hashing for Message Authentication*. RFC 2104.
- [4] Krawczyk, H., & Eronen, P. (2010). *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. RFC 5869.
- [5] Moriarty, K., Kaliski, B., & Rusch, A. (2017). *PKCS #5 : Password-Based Cryptography Specification Version 2.1*. RFC 8018.
- [6] Josefsson, S., & Liusvaara, I. (2017). *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032.
- [7] Moriarty, K., Kaliski, B., Jonsson, J., & Rusch, A. (2016). *PKCS #1 : RSA Cryptography Specifications Version 2.2*. RFC 8017.
- [8] National Institute of Standards and Technology. (2001). *Advanced Encryption Standard (AES)*. FIPS PUB 197.
- [9] National Institute of Standards and Technology. (2015). *Secure Hash Standard (SHS)*. FIPS PUB 180-4.
- [10] Dworkin, M. (2007). *Recommendation for Block Cipher Modes of Operation : Galois/Counter Mode (GCM) and GMAC*. NIST Special Publication 800-38D.