université de BORDEAUX

Master Cryptologie et Sécurité Informatique

Attaque par canal auxiliaire sur la signature ECDSA et réduction de réseau

Étudiants :
Paul Capgrand
Florian Duzes
Paul Chauveau

Professeur :
Guilhem Castagnos

12 février 2025

Table des matières

1 Introduction

Dans l'Histoire, la signature est un mécanisme qui remonte au moins à 3100 avant J.C [NDE93]. On a pu observer sur des tablettes de gestion de blé, de la cité sumérienne d'Uruk, le nom "Kushim". Cette signature permet d'affirmer une identité à un travail, un auteur à un écrit... En revanche, les historiens considèrent que cette marque n'est pas encore une signature et que c'est plutôt durant l'Antiquité que la signature devient moderne. Elle devient un moyen de distinguer et d'authentifier des documents.

Désormais à l'ère informatique [Ano06], la signature se transpose au numérique. Premièrement décrite par Whitfield Diffie et Martin Hellman en 1976 [DH76] sur l'idée de fonctions à sens unique à trappe, c'est en 1978 dans « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems » [RA78] que Ronald Rivest, Adi Shamir et Leonard Adleman ouvrent la voie pour une première implémentation.

Pour être utile, une signature numérique doit posséder les propriétés suivantes :

— Authentique : l'identité du signataire doit pouvoir être retrouvée de manière certaine
— Infalsifiable : une signature ne peut pas être reproduite
— Unique : la signature fait partie du document signé et ne peut être appliquée sur un autre document
— Inaltérable : une fois un document signé, on ne peut plus le modifier
— Irrévocable : la personne qui signe ne peut le contester

Ainsi, lorsque de nouveaux protocoles sont avancés pour réaliser des signatures numériques, des études sont nécessaires pour vérifier que ces caractéristiques sont préservées. Puis, enfin, un consortium de spécialiste, actuellement le NIST, donne son aval pour la standardisation d'un protocole. La signature RSA , première à être standardisée, est actuellement obsolète car elle ne présente plus assez de garanties de sécurité. Sa successeure, DSA est obsolète depuis 2023. Elle fut mise au point en 1991 à partir des cryptosystèmes de Schnorr et ElGamal [NIS94] et utilisée dès 1994 [Ano05].

Aujourd'hui, c'est la signature ECDSA qui est le standard de signature. Héritière de DSA, elle utilise les courbes elliptiques à la place des groupes d'entier modulo un grand nombre premier. Ce standard [NIS09], validé dès 2009, est actuellement largement déployé dans les systèmes informatiques contemporains car beaucoup plus compact pour un même niveau de sécurité que DSA.

L'objectif de ces travaux de recherche est de compromettre les propriétés de la signature ECDSA.

ANNONCE DU PLAN

2 Attaque par canal auxiliaire

2.1 Description rapide

En informatique, on évolue dans un monde binaire électronique. Ou il y a du courant et on représente la valeur "1", ou il n'y en a pas et on a la valeur "0". Ces variations de niveau d'électricité ont lieu sur un support matériel : une puce ou un fil de courant par exemple. Ainsi, lorsque l'on programme, toutes les instructions ont une empreinte électronique.

Une attaque par canal auxiliaire consiste à lire ses empreintes depuis l'extérieur du composant. Il s'agit d'attaquer un système sans remettre en cause sa sécurité théorique. On exploite des failles matérielles ou logicielles. Si en lisant la consommation du composant que j'attaque, je suis capable d'en avoir une image bit à bit au fil du temps. Alors, il est aisé de transcrire le binaire, puis de l'analyser pour savoir quel est le fonctionnement du composant, malgré la volonté de son concepteur.

Les attaques par canal auxiliaire sont nombreuses et permettent de classer les composants en niveau de sécurité. Dans le domaine des cartes à puce, des laboratoires testent et rendent des verdicts sur la sécurité les composants qui sont utilisés dans nos cartes bancaires, dans les voitures ou dans les téléphones.

Ces attaques peuvent être distinguées par leur caractère invasif. Dans le cas invasif, une interaction avec le matériel est réalisée. En général, ces interactions sont réalisées avec une sonde qui observe les échanges d'informations sur une zone physique, ou avec l'application d'une faute (modification d'un bit). Une attaque par injection de faute consiste à perturber le fonctionnement normal d'un composant grâce l'application d'une perturbation laser, lumineuse ou électromagnétique. Dans le second cas, on effectue une lecture depuis l'extérieur du composant. On a l'analyse de consommation électrique, comme décrit précédemment, l'analyse temporelle (où on observe le temps effectuer pour réaliser certaines opérations) ou encore l'analyse électromagnétique.

Une attaque par canal auxiliaire consiste à lire un support par une méthode décrite, ou d'autre encore existante, puis de réaliser une analyse de cette lecture pour en déduire le processus de fonctionnement du support. Les fabricants de tels supports réalisent eux-mêmes ces attaques pour pouvoir mettre en place des contre-mesures adaptées. Ces protections peuvent être matérielles ou logicielles.

On nomme trace l'information recueillie lors d'une lecture par canal auxiliaire. On utilisera ce terme par la suite pour désigner les éléments que l'on utilise pour mener nos attaques. En pratique, une trace peut être soumise à des bruits parasites générés par l'appareil écouté. C'est-à-dire que l'information obtenue n'est pas directement lisible sur la trace. Dans notre cas, nous aurons des traces parfaites, non bruitées. Cela nous amène à vous décrire plus précisément le cadre de notre étude.

2.2 Cadre de notre étude

L'objectif de notre étude est d'évaluer la sécurité de la signature ECDSA face aux attaques par canal auxiliaire. Comme décrit précédemment, mener une attaque par canal auxiliaire demande matériel et outils de précision. Elle doit être configuré spécifiquement pour l'architecture attaquée.

Nous nous détachons de ces contraintes physiques en conceptualisant un modèle simplifié d'étude. Ce modèle comprend plusieurs modules, principalement pour générer des traces :

1 - Un calculateur de paramètres
1 - Un générateur de signatures
1 - Un simulateur d'extracteur de bits
2 - Un module de mise en réseau
2 - Un calculateur de clé depuis les bits extraits

Les modules (1) nous permettent de générer rapidement des signatures et les modules (2) nous permettent de les attaquer en faisant une hypothèse sur le nombre de bits connus. Cela nous permet ensuite d'évaluer les différents types d'attaques réalisables sur les signatures étudiées.

À ces modules, on peut ajouter des outils de visualisation, de tests automatiques et de mémorisation. L'ensemble constitue la partie programmé de ce projet.

3 Digital Signature Algorithm

De nombreuses attaques par canal auxiliaire ont été proposé sur ce cryptosystème. Howgrave-Graham et Smart ont développé en 1999 [NA 01] des attaques fondées sur la réduction de réseaux euclidiens contre ces signatures. Ces méthodes ont connu de nombreux développements, s'adaptant aux différentes techniques d'exponentiations et contre-mesures utilisées pour implanter cet algorithme de signature. En nous appuyant sur cet article « Lattice Attacks on Digital Signature Schemes », nous allons étudier la stratégie nécessaire à la réalisation de ces attaques de récupération de clé privé.

3.1 Analyse

La signature DSA repose sur la sécurité du logarithme discret, mais la mise au point de l'algorithme LLL [LLL82] à permis la mise au point d'attaques où une partie de l'information est connue.

Nous allons décrire une attaque par réseau, dont la connaissance d'un certain nombre de messages m_i issus d'une clé privé, et d'une proportion de bits de clé éphémères y_i, nous permet de retrouver la clé secrète.

3.1.1 Mise en place

Alice souhaite signer ses messages. Elle va publier un groupe G et un élément de ce groupe g. Ce groupe doit être d'ordre p, un nombre premier de plus de 160 bits de longs, on a donc $\#G = p$. Il est aussi nécessaire d'avoir une fonction f de G dans $\mathbb{Z}/p\mathbb{Z}$. En pratique, on utilise une fonction de hachage publique.

Sa clé secrète est x, et elle va ensuite publier $h = g^x$, comme vérifieur de signature.

Alice a un message $m \in \mathbb{Z}/p\mathbb{Z}$, voici l'équation de la signature b :

$$m \equiv by - xf\left(g^y\right) \pmod{p} \tag{1}$$

Avec $y \in \{1, \ldots, p-1\}$ sa clé éphémère et b la signature associé. Le triplé (m, g^y, b) permet ensuite de vérifier la signature selon cette equation :

$$g^m * h^{f(g^y)} = (g^y)^b$$

3.1.2 Stratégie de l'attaque

L'idée de l'attaque se base sur la récupération d'entropie de bits de la clé. Si Alice signe suffisamment de messages avec sa clé privé, alors, si l'attaquant connaît par un canal auxiliaire des bits d'informations des clés éphémères, il est possible de retrouver cette clé privée.

Admettons que l'on puisse recupèrer h signatures, alors grâce à la construction 1, on peut construire h équations. Nos inconnus sont x et les y_i :

$$m_i - b_i y_i + xf\left(g^{y_i}\right) \equiv 0 \pmod{p} \tag{2}$$

Et avec quelques passes, on peut réarranger nos équations, avec A et B deux entiers, sous cette forme $y_i + x * A_i + B_i \equiv 0 \pmod{p}$. Avec un pivot de Gauss d'une équation de notre choix (ici on choisit la dernière équation, la h), on peut éliminer l'inconnu x. Nos équations seront alors de la forme :

$$y_i + y_h * A'_i + B'_i \equiv 0 \pmod{p} \tag{3}$$

3.1.3 Information sur les clés éphémères

Maintenant, on se tourne vers nos clés éphémères y_i. Si on connait y_i, alors on peut retrouver x directement par la resolution de 1. Ici on suppose que l'on connait une partie de l'information. Grâce à une attaque par canal auxiliaire, on peut admettre la connaissance de quelques bits de la clé. Cette connaissance peut-être continue ou discontinue, c'est-à-dire que l'on connaît des bits voisins, ou seulement des bits épars. Dans cette section 3.1 on admet la connaissance de bloc de clé. Cette connaissance peut alors être représenter mathématiquement par :

$$y_i = \alpha'_i + 2^{\lambda_i} z_i + 2^{\mu_i} \alpha''_i \tag{4}$$

Ainsi, on connait α'_i, α''_i, λ_i et μ_i. Notre seule inconnue est z_i. On comprend que nos variables sont conditionnés sous :

$$0 \leq z'_i < 2^{\lambda_i}, 0 \leq z_i < X_i = 2^{\mu_i - \lambda_i}, \lambda_i < \mu_i \text{ and } 0 \leq z''_i$$

L'objectif désormais, est de résoudre ces équations en trouvant z_i. Voici comment se présente chaque équation :

$$z_i + s_i z_h + t_i \equiv 0 \pmod{p} \tag{5}$$

3.1.4 Enchaînement mathématique

Dans cette section, nous allons observer plus en détail comment s'applique la transformation de l'information. Nous recevons tout d'abord les informations de 1.

$m_i \equiv by_i - xf\left(g^{y_i}\right) \pmod{p}$
$\Leftrightarrow -by_i + xf\left(g^{y_i}\right) + m_i = 0 \pmod{p}$
$\Leftrightarrow y_i + xf\left(g^{y_i}\right)(-b)^{-1} + m_i(-b)^{-1} = 0 \pmod{p}$

Ainsi, on peut s'implifier dans un premier temps, c'est équations en définissant $A_i = f\left(g^{y_i}\right)(-b)^{-1}$ et $B_i = m_i(-b)^{-1}$. On obtient nos équations sous la forme :

$$y_i + xA_i + B_i = 0 \pmod{p}$$

Ensuite, on souhaite faire disparaitre la variable x de notre ensemble d'équation. Elle correspond à la clé secrète d'Alice et on a admis notre absence d'information sur celle-ci :
$y_h + xA_h + B_h = 0 \pmod{p}$
$\Leftrightarrow xA_h = -y_h - B_h \pmod{p}$
$\Leftrightarrow x = -y_h A_h^{-1} - B_h A_h^{-1} \pmod{p}$

Maintenant que x est simplifié, on va pouvoir faire le pivot de Gauss décrit dans la section 3.1.2. Donc :
$y_i + (-y_h A_h^{-1} - B_h A_h^{-1})A_i + B_i = 0 \pmod{p}$
$\Leftrightarrow y_i + -y_h A_h^{-1} A_i - B_h A_h^{-1} A_i + B_i = 0 \pmod{p}$
On va à nouveau simplifier notre équation en définissant $A'_i = A_h^{-1} A_i$ et $B'_i = -B_h A_h^{-1} A_i + B_i$, ce qui nous rend avec 3 : $y_i + y_h * A'_i + B'_i = 0 \pmod{p}$

Enfin, on va intégrer l'information que l'on a pu récupérer de notre attaque par canal auxiliaire. Donc, chaque équations est modifiées pour présenter le changement :
$\alpha'_i + 2^{\lambda_i} z_i + 2^{\mu_i} \alpha''_i + (\alpha'_h + 2^{\lambda_h} z_h + 2^{\mu_h} \alpha''_h) * A'_i + B'_i = 0 \pmod{p}$
$\Leftrightarrow 2^{\lambda_i} z_i + (\alpha'_h + 2^{\lambda_h} z_h + 2^{\mu_h} \alpha''_h) * A'_i + B'_i + \alpha'_i + 2^{\mu_i} \alpha''_i = 0 \pmod{p}$
$\Leftrightarrow z_i 2^{\lambda_i} + z_h 2^{\lambda_h} A'_i + (\alpha'_h + 2^{\mu_h} \alpha''_h) A'_i + B'_i + \alpha'_i + 2^{\mu_i} \alpha''_i = 0 \pmod{p}$
$\Leftrightarrow z_i 2^{\lambda_i} + z_h 2^{\lambda_h} A'_i + (\alpha'_h + 2^{\mu_h} \alpha''_h) A'_i + B'_i + \alpha'_i + 2^{\mu_i} \alpha''_i = 0 \pmod{p}$

$$\Leftrightarrow z_i + z_h 2^{\lambda_h} A'_i 2^{-\lambda_i} + ((\alpha'_h + 2^{\mu_h}\alpha''_h)A'_i + B'_i + \alpha'_i + 2^{\mu_i}\alpha''_i)2^{-\lambda_i} = 0 \pmod{p}$$

Ainsi, en définissant $s_i = 2^{\lambda_h} A'_i 2^{-\lambda_i}$ et $t_i = ((\alpha'_h + 2^{\mu_h}\alpha''_h)A'_i + B'_i + \alpha'_i + 2^{\mu_i}\alpha''_i)2^{-\lambda_i}$, on obtient notre forme simplifié 5 :

$$z_i + s_i z_h + t_i = 0 \pmod{p}$$

3.1.5 Réduction de réseau et solution

Maintenant equipé d'un système d'équations, nous allons pouvoir le réduire comme un problème de réseau. Tout d'abord, nous allons rappeler ici pourquoi utiliser les réseaux euclidiens.

Un réseau L est un sous-groupe discret de l'espace associé. Cette structure mathématique peut être représenter par une base $\mathbb{B}$ de d vecteurs indépendants $\{b_1, \dots b_d\}$. L est engendré par $\mathbb{B}$ lorsqu'on réalise des combinaisons linéaires des d vecteurs. Utiliser les réseaux nous permet d'accéder aux problèmes associé à $\mathbb{B}$: est-ce la base la plus simple qui traduise L ? Par plus simple, on entend en général, la base qui a les vecteurs les plus courts (problème SVP) ou les plus proche entre eux (problème CVP). Ces problèmes sont de classe NP-difficile. Cela signifie que lorsque l'on souhaite simplifier, réduire notre base $\mathbb{B}$ en temps polynomial, il faut accepter d'obtenir une approximation de la base idéale. On nomme γ-approximation la base obtenue. En acceptant cette imprécision de $\gamma > 1$, on peut utiliser l'algorithme de Lenstra–Lenstra–Lovász [LLL82].

Pour circuler du problème de solution d'équations à celui de recherche de plus court vecteur, on utilise la matrice A :

$$A = \begin{pmatrix} -1 & s_1 & s_2 & \dots & s_n \\ 0 & p & 0 & \cdots & 0 \\ 0 & 0 & p & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & \cdots & \cdots & p \end{pmatrix} \in M_{(n+1),(n+1)}(\mathbb{Z})$$

L'hypothèse de l'attaque développé par Howgraham-graham et Smart [NA 01] repose sur la recherche du plus court vecteur. Par construction de ce réseau $L = \{\mathbf{x}A : \mathbf{x} \in \mathbb{Z}^{n+1}\}$ issu de A et d'un vecteur quelconque $\mathbf{x} \in \mathbb{Z}^{n+1}$, si on considère le vecteur :

$$\mathbf{t} = (0, t_1, t_2, \dots, t_n) \in \mathbb{Z}^{n+1}$$

Alors on peut retrouver les clés éphémères,

$$\mathbf{x}A - \mathbf{t} = (z_0, z_1, \dots, z_n) \in \mathbb{Z}^{n+1}$$

3.1.6 Mesure de la durée de l'attaque

Cette attaque est une simulation académique, ainsi mesurer le temps effectif de cette attaque est plus difficile. Cela dépend des compétences des attaquants et de la qualités des traces récupérés. Le temps nécessaire à la récupération de signature, leur analyse est ici ignoré.

On considère que la récupération des signatures, la reformulation des équations et la construction de la matrice doivent être faite en amont. Une fois ces points validés, alors l'attaque peut être lancé :

Algorithm 1 « Lattice Attacks on Digital Signature Schemes »

Require: $nb\_signature, epsilon, Zq, g, x$ ▷ Paramètres pour la génération des signatures
$\beta = generate\_signature(nb\_signature, epsilon, Zq, g, x)$
$eq = create\_equation(\beta)$
$A = matrix\_from\_equations(eq)$
$Z = plus\_court\_vecteur(A)$
$verification(\beta, Z)$

3.1.7 Point de vigilance

Nous avons rencontrer des difficultés lors de la création d'un paramètre publié par Alice. Le groupe G doit être d'ordre p, un grand nombre premier avec suffisamment de bits de sécurité. Prendre un grand nombre premier p_2 aléatoirement, puis créer un groupe G_2 avec, n'est pas la bonne méthode. Le groupe sera trop petit : $\#G_2 = p_2 - 1$. Il faut reprendre la méthode décrite par le *FIPS 186 : Digital Signature Standard (DSS)* [NIS94].

On produit un grand nombre premier q aléatoirement. Puis on va chercher p tel que $q|(p-1)$. Sans reprendre la méthode décrite dans le document, on obtient un tel p. Finalement, on prend aléatoirement un élément h de $\mathbb{Z}/p\mathbb{Z}$, on calcule $e = (p-1)/q$, enfin, on publie $G = h^e$.

3.2 Résultat

```
equations, ephemeral_keys, known_parts = data_gen(n+1,epsilon)
equations = rearrange_equations(equations,n+1)
A = matrix_from_equations(equations,n)
start = time.time() # data treatment doesn't count in atk duration
Z = attack(A, equations)
end = time.time()
offset = ZZ(160*epsilon)
elapsed = end - start
z0_found = Z[index+1]*2^offset + known_parts[index]
actual_z0 = ZZ(ephemeral_keys[index])
correct = (z0_found == actual_z0)
```

On remarque que le temps des attaques est similaire peut importe les valeurs de p, q, g, x et epsilon. La valeur qui affecte la durée de l'attaque est `n` qui est le nombre de signatures exploitées et donc la taille de la matrice pour la réduction de réseau puis Babai. Le temps similaire pour les faibles valeurs de `n` peuvent s'expliquer par le coût quasiment constant d'initialisation des algos LLL et Babai en Sage (chargement des classes, précalculs).

	Actual Value of n required	Time (sec)
.500	1	0.2742
.250	4	0.2601
.100	10	0.2625
.050	25	0.3490
.025	+100	Infaisable ?

Avec tes tailles de p et q similaires à l'article, on a de meilleures performances avec des petites valeurs de epsilon (peu de bits de la clé éphémère), pour de grandes valeurs de epsilon, l'article présente de meilleurs performances.

3.3 Regard sur les paramètres de sécurités

LLL, BKZ, papier de recherche, et L4

4 ECDSA

4.1 Contexte et motivation

4.2 Changement d'implémentation

4.3 Résultats

5 Conclusion

6 Mémo

6.1 Paramètres

Si on reprend toutes les étapes depuis le début :

1. La taille minimale de modules premiers est de 2048 bits pour une utilisation ne devant pas dépasser la fin de l'année 2030.

2. Pour une utilisation à partir de l'année 2031, la taille minimale de modules premiers est de 3072 bits.

3. On emploiera des sous-groupes dont l'ordre est multiple d'un nombre pre- mier d'au moins 250 bits.

```python
from Cryptodome.PublicKey import ECC
mykey = ECC.generate(curve='p256')
```

$$A = \begin{pmatrix} -1 & s_1 & s_2 & \dots & s_n \\ 0 & p & 0 & \cdots & 0 \\ 0 & 0 & p & & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & \dots & \cdots & \cdots & p \end{pmatrix} \in M_{(n+1),(n+1)}(\mathbb{Z})$$

From Lattice Attacks on Digital Signature Schemes[NA 01].

		Actual Value of n required	Time in Seconds
.500	2	2	0.0102
.250	4	4	0.0360
.100	10	11	0.4428
.050	20	30	8.6970
.025	40	-	Infeasible

Also from Lattice Attacks on Digital Signature Schemes[NA 01].

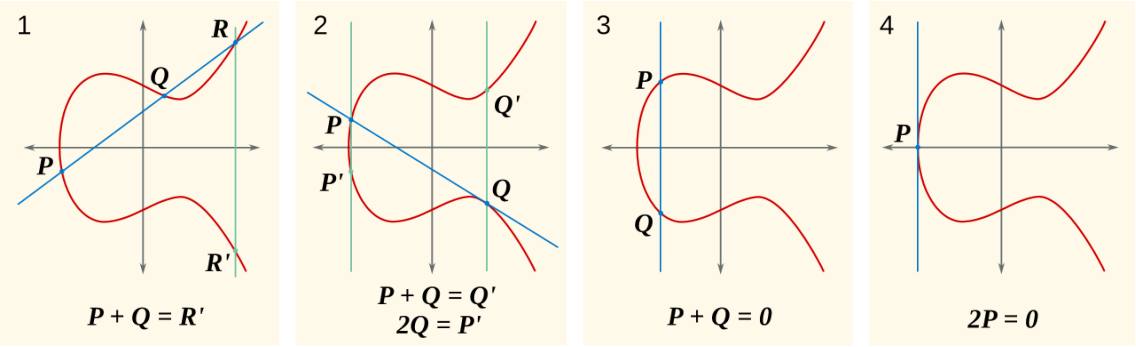

FIGURE 1 – Différents cas d'addition dans les courbes elliptiques[Wik24]

7 Bibliographie

Références

[NDE93] Hans J. Nissen, Peter Damerow et Robert K. Englund. *The Administrative Activities of Kushim". Archaic Bookkeeping : Early Writing and Techniques of Economic Administration in the Ancient Near East.* Trad. par paul Larsen. University of Chicago Press, 1993. isbn : 0-226-58659-6.

[Ano06] Anonymes. *Signature numérique*. 2006. url : https://fr.wikipedia.org/wiki/Signature_num%C3%A9rique#.

[DH76] W. Diffie et M. Hellman. « New directions in cryptography ». In : *IEEE Transactions on Information Theory* 22.6 (1976), p. 644-654. doi : 10.1109/TIT.1976.1055638.

[RA78] Adi Shamir Ronald L. Rivest et Leonard M. Adleman. « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems ». In : *Communications of the ACM* 21 (1978). doi : 10.1145/359340.359342.

[NIS94] NIST. *FIPS 186 : Digital Signature Standard (DSS)*. 1994. url : https://web.archive.org/web/20131213131144/http://www.itl.nist.gov/fipspubs/fip186.htm.

[Ano05] Anonymes. *Digital Signature Algorithm*. 2005. url : https://en.wikipedia.org/wiki/Digital_Signature_Algorithm.

[NIS09] NIST. « FIPS 186-3 : Digital Signature Standard (DSS) ». In : (2009). url : https://csrc.nist.gov/files/pubs/fips/186-3/final/docs/fips_186-3.pdf.

[NA 01] N.P. Smart N.A. Howgrave-Graham. « Lattice Attacks on Digital Signature Schemes ». In : *Designs Codes and Cryptography* (2001). url : https://www.researchgate.net/publication/225240686_Lattice_Attacks_on_Digital_Signature_Schemes.

[LLL82] A.K. Lenstra, H.W. Lenstra et L. Lovász. « Factoring polynomials with rational coefficients ». In : (1982).

[Wik24] Wikipédia. *Courbe elliptique — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 15-septembre-2024]. 2024. url : http://fr.wikipedia.org/w/index.php?title=Courbe_elliptique&oldid=218649671.