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

Capgrand Paul - Chauveau Paul - Duzes Florian

Master Cryptologie et Sécurité Informatique

université de BORDEAUX

April 15, 2025

Introduction

1999 : Howgrave-Graham et Smart [HS01]

"Lattice Attacks on Digital Signature Schemes"

2001 : Publication dans le journal *Designs, Codes and Cryptography*

Introduction

Outils développés en *SageMath* :

1. Un générateur de paramètres DSA
2. Un générateur de signatures et de traces pour tous les protocoles étudiés
3. Les fonctions de l'attaque de notre référence
4. Un programme pour filtrer et agréger les résultats avant de les tracer avec *numpy*, *pandas*, *matplotlib* et *seaborn*

Sommaire

Préambule

Réseaux Euclidiens

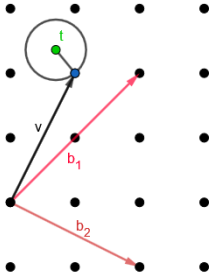
Un réseau L est un sous-groupe discret de $\mathbb{R}^n$.

Cette structure peut être décrite par une base $\mathcal{B}$ de d vecteurs indépendants $\{b_1, \dots b_d\}$. En posant A la matrice dont les lignes sont les d vecteurs de $\mathcal{B}$, on peut écrire :

$$L = \{\mathbf{x}A : \mathbf{x} \in \mathbb{Z}^n\}$$

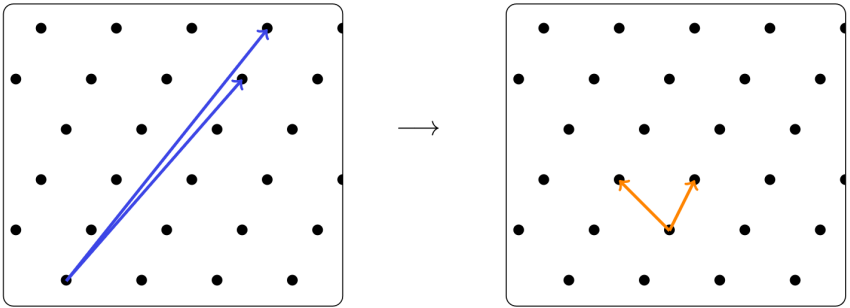
Closest Vector Problem

- Pour un vecteur $\mathbf{t}$ de $\mathbb{R}^n$, trouver le vecteur de L le plus proche.
- NP-Difficile



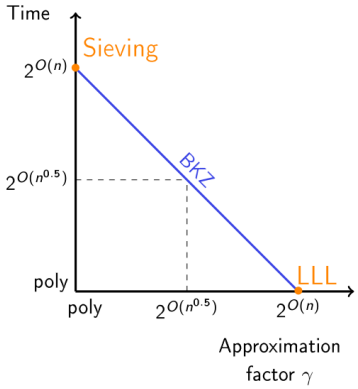
CVP [SSG24]

Réduction de base



Exemple de réduction de réseaux [Pel24]

Algorithme de réduction de réseau



Comparaison du facteur d'approximation et le temps de calcul entre LLL, BKZ et Sieving [Pel24]

Approximation du CVP

Babaï :

$$\gamma = 2\left(\frac{2}{\sqrt{3}}\right)^d$$

avec d le rang du réseau.

Algorithme du plan proche de Babai :

1. une base $\mathcal{B} \in \mathbb{Z}^{d \times n}$
2. un vecteur cible $t \in \mathbb{Z}^n$

Une réduction de réseau avant de projeter itérativement t sur chaque vecteur de base réduit successif. La projection arrondie est ensuite soustraite de t pour obtenir un nouveau vecteur plus proche du point du réseau.

Digital Signature Algorithm

La sécurité de la signature DSA, repose sur le problème du logarithme discret dans le groupe $(Z/p\mathbb{Z})^\times$ avec p premier et suffisamment grand.

Paramètres publics:

1. p_{1024} et q_{160}, deux nombres premiers et tel que $q|(p-1)$, [NIS94]
2. g un générateur de $(Z/p\mathbb{Z})^\times$

Clé secrète : $x \leftarrow \mathbb{Z}/q\mathbb{Z}$

Clé publique : $h = g^x$

$$(p, q, g, h)$$

Protocole de signature

f une fonction de hachage : SHA-1

Soit $m \in \mathbb{Z}/q\mathbb{Z}$, $y \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$

$$b \equiv (m + xf\,(g^y))y^{-1} \pmod{q} \tag{1}$$

$$(g^y, b)$$

Pour vérifier la signature :

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

Interêt des courbes elliptiques

Table: Taille de clés pour une sécurité équivalente entre la signature DSA et la signature avec les courbes elliptiques (EC). Issu de [HMV04], p.19

	Security level (bits)				
	80	112	128	192	256
DSA paramètre p	1024	2048	3072	8192	15360
EC paramètre n	160	224	256	384	512

Elliptic Curve DSA

E une courbe elliptique d'ordre n un nombre premier, soit P un point de E et f notre fonction de hachage.

Clé secrète $x \leftarrow \mathbb{Z}/n\mathbb{Z}$
Clé publique $Q = xP$

$$r \xleftarrow{\$} \mathbb{Z}/n\mathbb{Z}, rP = (x_R, y_R)$$

La signature est alors donnée par $\sigma = (\sigma_1, \sigma_2) = (x_R \mod n, s)$, où :

$$s \equiv r^{-1}(x(x_R \mod n) + f(m)) \pmod{n}. \tag{2}$$

Signature ECDSA - vérification

Vérification de (σ_1, σ_2) :

1. $u_1 \equiv f(m)\sigma_2^{-1} \pmod{n}$
2. $u_2 \equiv \sigma_1\sigma_2^{-1} \pmod{n}$

$(x_1, y_1) = u_1P + u_2Q$

$$\sigma_1 \equiv x_1 \tag{3}$$

Traces & Préparation

Illustration d'une trace

Appelons x la valeur dont on veut récupérer les bits d'informations, admettons par exemple que x s'écrit ainsi :

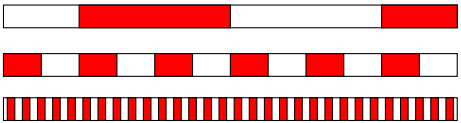
$$x = 10101101010011110001110100111 10$$

L'information inconnue de x, en rouge sur les schémas ci-dessous, peut être organisée de différentes manières. La plus simple étant le cas contigu où juste un bloc de bits est manquant :

Cas contigu :

Illustration d'une trace - cas non contigu

Mais nous prenons aussi en compte le cas où l'information manquante est séparée en plusieurs blocs :

Cas non-contigu : $x =$ 

Attaque

Signatures et équations

Attaque par canal auxiliaire => bits d'information sur les clés éphémères y_i
Objectif : retrouver entièrement une clé éphémère et d'en déduire la clé privée x
On récupère h signatures => h équations pour $1 \leq i \leq h$:

$$m_i - b_i y_i + xf\left(g^{y_i}\right) \equiv 0 \pmod{q} \tag{4}$$

On peut ensuite réarranger nos équations, avec A et B entiers, sous cette forme $y_i + xA_i + B_i \equiv 0 \pmod{q}$. Pivot de Gauss pour exprimer x en fonction de y_h :

$$y_i + y_h \times A'_i + B'_i \equiv 0 \pmod{q} \tag{5}$$

Simplification des équations

$$y_i = \alpha'_i + 2^{\lambda_i} z_i + 2^{\mu_i} \alpha''_i \tag{6}$$

y_i :

On connaît les α'_i, α''_i, λ_i et μ_i. Nos inconnues sont les z_i et on définit X_i leurs bornes supérieures :

$$0 \leq z_i < X_i = 2^{\mu_i - \lambda_i}$$

On simplifie une dernière fois nos équations pour obtenir :

$$z_i + s_i z_h + t_i = 0 \pmod{q} \tag{7}$$

Réseau et CVP

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

Réseau $L = \{\mathbf{x}A : \mathbf{x} \in \mathbb{Z}^{n+1}\}$ issu de A. Un vecteur $\mathbf{v}$ de L s'exprime ainsi :

$$\mathbf{v} = (-x_0,\, x_0 s_1 + x_1 q,\, \dots,\, x_0 s_n + x_n q) \in \mathbb{Z}^{n+1}$$

$$z_i \equiv -z_h s_i - t_i \pmod{q}$$

$$z_i \equiv -z_h s_i - t_i \pmod{q}$$

En prenant :

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

On sait qu'il existe :

$$\mathbf{v} - \mathbf{t} = (z_h,\ z_1,\ \ldots,\ z_n) \in \mathbb{Z}^{n+1}$$

$$\|\mathbf{v} - \mathbf{t}\|^2 \leq \sum_{i=0}^{n} X_i^2$$

Non-contigu

$$y_i = z'_i + \sum_{j=1}^{d} z_{i,j} 2^{\lambda_{i,j}}$$

y_i :

Notre système d'équation devient :

$$z_{i,1} + \sum_{j=2}^{d} s_{i,j} z_{i,j} + \sum_{j=1}^{d} r_{i,j} z_{0,j} + t_i \equiv 0 \pmod{q}$$

$$A = \left(\begin{array}{c|c} -I_{d(n+1)-n} & R^t \\ & S \\ \hline 0 & -qI_n \end{array} \right) \times D$$

Où $R = (r_{i,j})$ et S correspond à la matrice

$$S = \begin{pmatrix} \mathbf{s}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{s}_n \end{pmatrix} \in M_{n(d-1),n}(\mathbb{Z})$$

avec $\mathbf{s}_i$ le vecteur colonne $(s_{i,j})_{j=2}^{d}$.

Résultats

Comparaison avec l'article

Table: Comparaison de résultats entre [HS01] et notre implémentation, réalisé sur bergman

Epsilon	Bits connus	Nombre d'équations[1]		Temps (secondes)	
		Nous	Eux	Nous	Eux
0.25	40	4	4	0.00428	0.0360
0.1	16	10	11	0.00956	0.4428
0.05	8	24	30	0.05998	8.6970
0.03125	5	80	/	7163.72	*échec*

[1] Le nombre d'équations présenté est le minimum requis pour que l'attaque fonctionne.

Comparaison avec l'article - cas non contigu

Table: Comparaison de résultats entre [HS01] et notre implémentation, réalisé sur lautrec

Epsilon	d	Nombre d'équations		Temps (secondes)	
		Nous	Eux	Nous	Eux
0.5	4	2	2	0.0067	0.304
	8	2	2	0.032	1.135
	16	3 (6%)	/	0.899	*échec*
0.25	2	4	4	0.050	0.393
	4	4 (95%)	4	0.024	1.785
	8	5 (3%)	/	0.452	*échec*
0.1	2	12 (88%)	12	0.087	6.256
	4	/	/	*échec*	*échec*

Considérations matérielles

Table: configurations des machines utilisées

Machine	modèle CPU	RAM (GO)
bergman	Intel i9-13900	64
lautrec	Intel Xeon E-2236	32
jolicoeur	AMD Opteron 6276	124
wylde	Intel Xeon W-1250P	48
moore	Intel Xeon W-1250P	48
plomet	Intel Xeon W-1290	64

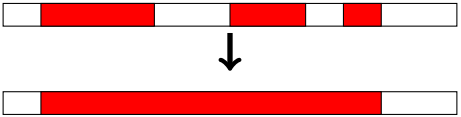
Configuration plausible pour l'ordinateur des chercheurs en 1999 :

- Intel Pentium II - 32 bits, cadencé entre 233 et 450 MHz
- DDR RAM 256 Megaoctets

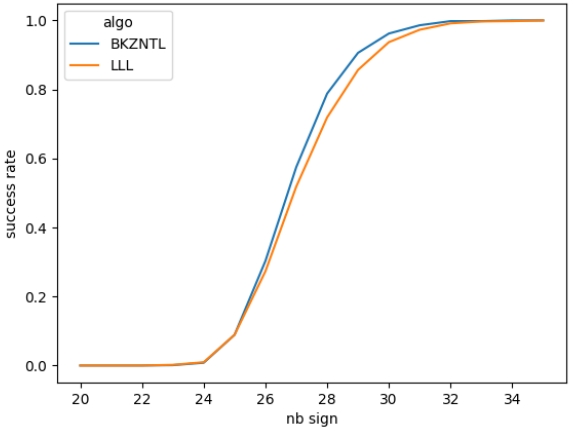
Remarque

Table: Extrait de résultat sur DSA historique, non contigu sur lautrec, contigu sur plomet

	Epsilon	Nombre d'équations	Temps (secondes)	Réussite
Non Contigu (d=16)	0.4	6	142.297	*False*
Contigu	0.03125	80	7163.72	*True*

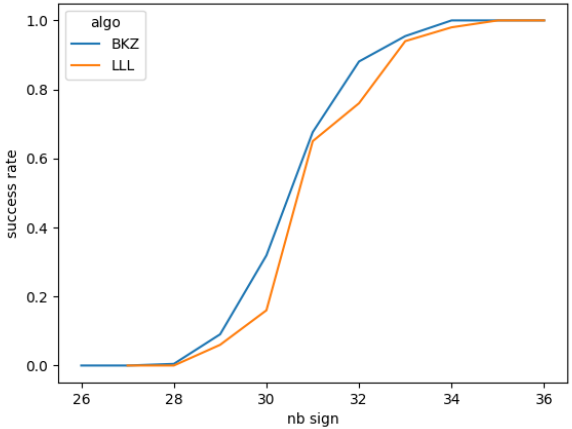


DSA 1024 160 - 8 bits / epsilon=0.05



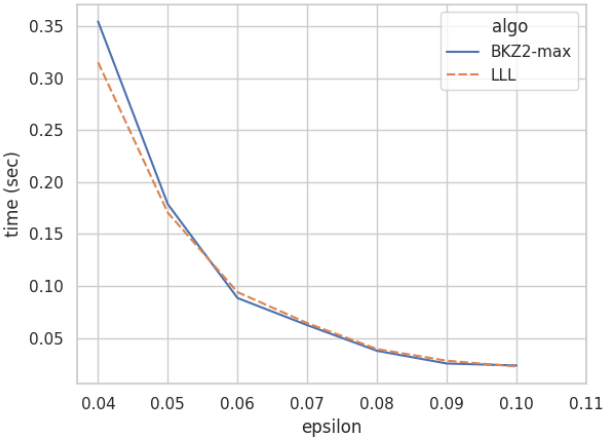
Probabilité de succès de l'attaque en fonction du nombre de signatures sur jolicoeur

ECDSA P-256 - 10 bits / epsilon=0.04



Probabilité de succès de l'attaque en fonction du nombre de signatures sur jolicoeur

ECDSA P-256 - comparaison LLL / BKZ-NTL



Comparaison du temps de calcul en fonction de la valeur d'epsilon sur jolicoeur

Conclusion

Références

[HMV04] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[HS01] N.A. Howgrave-Graham and N.P. Smart. "Lattice Attacks on Digital Signature Schemes". In: *Designs Codes and Cryptography* (2001).

[KEF] Paul Kirchner, Thomas Espitau, and Pierre-Alain Fouque. "Towards Faster Polynomial-Time Lattice Reduction". In: *Advances in Cryptology – CRYPTO 2021*.

[NIS94] NIST. *FIPS 186 : Digital Signature Standard (DSS)*. 1994.

[Pel24] Alice Pellet-mary. "Part II: algorithmic problems and algorithms". In: *Cryptologie Post-Quantique*. 2024.

[RH23] Keegan Ryan and Nadia Heninger. *Fast Practical Lattice Reduction through Iterated Compression*. Cryptology ePrint Archive, Paper 2023/237. 2023.

[SSG24] Maria Sabani, Ilias Savvas, and Georgia Garani. "Learning with Errors: A Lattice-Based Keystone of Post-Quantum Cryptography". In: *Signals* 5 (Apr. 2024), pp. 216–243.

[Xu+23] Luyao Xu et al. "Improved Attacks on (EC)DSA with Nonce Leakage by Lattice Sieving with Predicate". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023.2 (2023), pp. 568–586.

Ouverture

Records

[Xu+23] => algorithmes de réduction de réseau et des techniques de criblage avec prédicat. Records en termes de temps d'exécution et de nombre d'échantillons.

Table: Comparaison avec les précédents records des attaques par réseau sur (EC)DSA. Chaque colonne correspond à la taille des courbes et chaque ligne correspond au nombre de bits de fuite de clé éphémère par signature.

	4-bit	**3-bit**	**2-bit**	**1-bit**
112-bit	-	-	-	[Xu+23]
160-bit	-	[2002]	[2013],[2021],[2022]	/
256-bit	[2019],[2020]	[2021],[2022]	[Xu+23]	/
384-bit	[2021],[2022]	[Xu+23]	/	/

w-NAF

Table: Tableau de Conversion entre Différentes Formes NAF

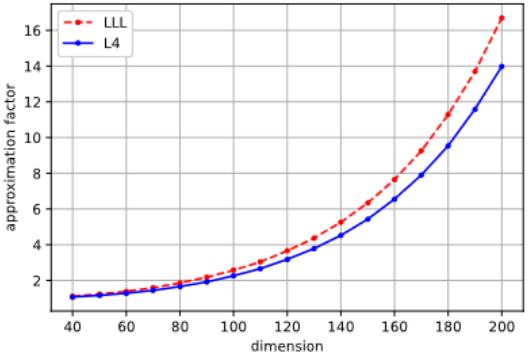
Décimal	**Binaire**	**1-NAF**	**2-NAF**	**3-NAF**
9	1001	1001	1001	$1000(-7)$
11	1011	$10(-1)0(-1)$	1003	$1000(-5)$
29	11101	$100(-1)01$	$10000(-3)$	$10000(-3)$
42	101010	101010	$300(-3)0$	100050
85	1010101	1010101	$100300(-3)$	50005
170	10101010	10101010	$100300(-3)0$	500050

Algorithme de réduction

Table: Tableau de comparaison des tailles de réseau après application de Flatter, LLL et une étude antérieure

Dimension	Nombre de bits	fpLLL (s)	[KEF]	[RH23]
128	100000	3831	400	69
256	10000	2764	200	83
384	10000	10855	780	246

Algorithme de réduction



Comparaison de performances entre LLL et L4

Annexe

Implémentation

```
def hide_contiguous(y,epsilon, q) :
    if not 0 < epsilon <= 1 :
        raise ValueError(" - epsilon doit être entre 0 et 1.")
    y = int(y)

    total = len(bin(q)[2:])
    nb_bits = floor(total*epsilon)     # nb de bit de y connus
    unknow = total - nb_bits           # nb de bits inconnus

    # séparation en deux blocs initiaux ou finaux
    __lambda = floor(random() * nb_bits)
    mu = unknow + __lambda

    #application de masque
    a = y & ((1 << __lambda) - 1)
    b = (y >> __lambda) & ((1 << (mu - __lambda)) - 1)
    c = y >> mu

    return a,b,c,__lambda,mu
```