Étude du compromis temps-mémoire dans les chaînes arc-en-ciel

1 Introduction

Soit f une fonction à sens unique (chiffrement ou hachage) opérant sur un espace de clés de taille N. Étant donné un chiffré $y = f(k^*)$, on cherche à retrouver la clé secrète k^*.

Deux approches extrêmes s'offrent à nous :

— **Force brute** : tester les N clés une par une.
Temps $\mathcal{O}(N)$, mémoire $\mathcal{O}(1)$.

— **Table de correspondance complète** : pré-calculer et stocker tous les couples $(k, f(k))$.
Temps de recherche $\mathcal{O}(1)$ (ou $\mathcal{O}(\log N)$ avec tri), mémoire $\mathcal{O}(N)$.

Les chaînes arc-en-ciel, introduites par Oechslin [2] en amélioration des tables de Hellman [1], réalisent un *compromis* entre ces deux extrêmes : on échange du temps de pré-calcul contre de la mémoire afin de réduire simultanément le coût en ligne et le stockage requis.

2 Principe des chaînes arc-en-ciel

2.1 Construction d'une chaîne

On se donne :

— une fonction à sens unique $f : \mathcal{K} \to \mathcal{C}$ (ex. AES avec clé réduite),

— une famille de fonctions de réduction $R_j : \mathcal{C} \to \mathcal{K}$, pour $j = 0, 1, \ldots, t-1$, où chaque R_j ramène un chiffré dans l'espace des clés.

Le point crucial est que les R_j doivent être **toutes distinctes**. C'est ce qui distingue les tables arc-en-ciel des tables de Hellman, où une seule fonction de réduction est utilisée par table.

À partir d'une clé initiale k_0, on construit une chaîne de longueur t :

$$k_0 \xrightarrow{f} c_0 \xrightarrow{R_0} k_1 \xrightarrow{f} c_1 \xrightarrow{R_1} k_2 \longrightarrow \cdots \xrightarrow{f} c_{t-1} \xrightarrow{R_{t-1}} k_t$$

On ne stocke que le couple (k_0, k_t) : le début et la fin.

2.2 Phase de pré-calcul

On choisit m clés de départ $k_0^{(1)}, \ldots, k_0^{(m)}$ aléatoirement dans $\mathcal{K}$ et on construit m chaînes de longueur t. On obtient une table de m couples $(k_0^{(i)}, k_t^{(i)})$.

— **Coût mémoire** : m entrées.
— **Coût du pré-calcul** : $m \times t$ évaluations de f.

2.3 Phase d'attaque

Étant donné le chiffré cible $y = f(k^*)$, on cherche si y apparaît comme résultat intermédiaire d'une chaîne.

Pour chaque position j de $t-1$ à 0 :

1. On calcule le candidat de fin de chaîne :

$$y_j = R_{t-1}\big(f\big(\cdots R_j(y) \cdots\big)\big)$$

   c'est-à-dire qu'on applique R_j, puis alternativement f et $R_{j+1}, R_{j+2}, \ldots, R_{t-1}$ pour atteindre la position finale.
2. On cherche y_j parmi les $k_t^{(i)}$ de la table (recherche en $\mathcal{O}(\log m)$ si la table est triée).
3. En cas de correspondance, on reconstruit la chaîne depuis $k_0^{(i)}$ jusqu'à trouver k_j tel que $f(k_j) = y$.

Le coût en ligne de l'attaque est dominé par les $\frac{t(t+1)}{2} \approx \frac{t^2}{2}$ évaluations de f.

3 Analyse du compromis temps-mémoire

3.1 Couverture de l'espace de clés

Chaque chaîne de longueur t traverse t clés distinctes (en l'absence de collisions). Avec m chaînes, le nombre maximal de clés couvertes est $m \times t$.

Pour couvrir l'intégralité de l'espace $\mathcal{K}$, on veut donc :

$$m \times t \;\geq\; N \tag{1}$$

En pratique, on prend $m \times t$ légèrement supérieur à N pour compenser les collisions inévitables entre chaînes.

3.2 Courbe de compromis

Notons $M = m$ la mémoire (nombre d'entrées stockées) et $T = t^2/2$ le temps d'attaque en ligne. Avec la contrainte de couverture $m \times t = N$, on a $t = N/m$, d'où :

$$\boxed{T \;=\; \frac{t^2}{2} \;=\; \frac{N^2}{2\,M^2}} \qquad \text{soit} \qquad T \cdot M^2 \;=\; \frac{N^2}{2} \tag{2}$$

C'est la **courbe de compromis temps-mémoire**. On peut librement régler le curseur entre les deux extrêmes :

— *Plus de mémoire, moins de temps* : augmenter M (plus de chaînes, plus courtes).
— *Moins de mémoire, plus de temps* : diminuer M (moins de chaînes, plus longues).

3.3 Paramètres optimaux

Proposition 1. *Au point d'équilibre $M \approx T$, les paramètres optimaux satisfont :*

$$\boxed{M \;\approx\; T \;\approx\; N^{2/3}}$$

Démonstration. En imposant $M = T$ dans la relation (2) :

$$M \cdot M^2 = \frac{N^2}{2} \quad \Longrightarrow \quad M^3 = \frac{N^2}{2} \quad \Longrightarrow \quad M = \left(\frac{N^2}{2}\right)^{1/3} = \frac{N^{2/3}}{2^{1/3}} \;\approx\; 0{,}79 \cdot N^{2/3}$$

La longueur de chaîne correspondante est :

$$t = \frac{N}{M} \approx \frac{N}{N^{2/3}} = N^{1/3}$$

Le coût du pré-calcul reste $m \times t = N$ (comparable à une recherche exhaustive), mais il n'est effectué **qu'une seule fois** et peut être réutilisé pour attaquer autant de chiffrés que l'on souhaite. □

Remarque 1. *Le gain par rapport à la force brute est considérable. Par exemple, pour $N = 2^{40}$:*

Méthode	*Mémoire*	*Temps en ligne*
Force brute	$\mathcal{O}(1)$	$2^{40} \approx 10^{12}$
Table complète	2^{40}	$\mathcal{O}(1)$
Rainbow table	$\mathbf{2^{27} \approx 10^8}$	$\mathbf{2^{27} \approx 10^8}$

On passe d'un coût de 10^{12} à environ 10^8 en temps et en mémoire : un gain d'un facteur $\approx 10\,000$.

4 Analyse de la probabilité de succès

4.1 Modèle sans collisions

En l'absence de collisions, chaque chaîne couvre exactement t clés distinctes. La probabilité qu'une clé secrète k^* choisie uniformément *ne soit pas* couverte par la table est :

$$P(\text{échec}) = \left(1 - \frac{1}{N}\right)^{m \times t} \;\approx\; e^{-mt/N} \tag{3}$$

Avec $m \times t = N$ (couverture exacte) : $P(\text{succès}) \approx 1 - e^{-1} \approx 63\,\%$.

Pour atteindre une probabilité plus élevée, on prend $m \times t = c \cdot N$ avec $c > 1$:

$$P(\text{succès}) \approx 1 - e^{-c}$$

4.2 Modèle avec collisions

En pratique, des collisions réduisent la couverture. Lorsque deux chaînes produisent la même clé à la même colonne j, elles *fusionnent* et couvrent les mêmes clés pour le reste de la chaîne.

Soit m_j le nombre de clés **distinctes** à la colonne j. L'évolution est donnée par la récurrence (analogue au paradoxe des anniversaires) :

$$\boxed{m_0 = m, \qquad m_{j+1} = N\left(1 - e^{-m_j/N}\right)} \tag{4}$$

À chaque étape, les m_j entrées produisent $m_{j+1} < m_j$ sorties distinctes. La probabilité de succès tenant compte des collisions est alors :

$$P(\text{succès}) = 1 - \prod_{j=0}^{t-1}\left(1 - \frac{m_j}{N}\right) \approx 1 - \exp\left(-\frac{1}{N}\sum_{j=0}^{t-1} m_j\right) \tag{5}$$

4.3 Probabilité avec les paramètres optimaux

Proposition 2. *Avec un facteur de couverture $c = mt/N \approx 3$ à 4 et des fonctions de réduction bien choisies, la probabilité de succès atteint environ* **86 %** *à* **90 %***.*

Justification. En résolvant la récurrence (4) par l'approximation continue $\frac{dm}{dj} \approx -\frac{m^2}{2N}$, on obtient :

$$m(j) = \frac{2N \cdot m_0}{m_0 \cdot j + 2N}$$

La couverture totale est alors :

$$\Sigma = \sum_{j=0}^{t-1} m_j \approx \int_0^t m(j)\, dj = 2N \ln\left(\frac{m_0 \cdot t}{2N} + 1\right) = 2N \ln\left(\frac{c}{2} + 1\right)$$

La probabilité de succès devient :

$$P(\text{succès}) \approx 1 - e^{-\Sigma/N} = 1 - e^{-2\ln(c/2+1)} = 1 - \frac{1}{(c/2+1)^2}$$

Application numérique pour différentes valeurs de $c = mt/N$:

$c = mt/N$	Σ/N (couverture effective)	P(succès)
1	$\approx 0{,}81$	$\approx 56\,\%$
2	$\approx 1{,}39$	$\approx 75\,\%$
3	$\approx 1{,}83$	$\approx 84\,\%$
4	$\approx 2{,}20$	$\approx 89\,\%$
5	$\approx 2{,}50$	$\approx 92\,\%$

Ainsi, avec $c \approx 4$ (soit $mt \approx 4N$), on atteint une probabilité de succès d'environ **89 %**, ce qui est cohérent avec les résultats expérimentaux obtenus dans notre implémentation ($m = 500$, $t = 500$, $N = 2^{16} = 65\,536$, $c \approx 3{,}8$, succès observé : 9/10). □

Remarque 2. *Pour augmenter la probabilité au-delà de 90 % sans augmenter excessivement m ou t, on peut utiliser plusieurs tables arc-en-ciel indépendantes (avec des familles de réduction différentes). Avec ℓ tables, chacune de probabilité p :*

$$P_\ell(\textit{succès}) = 1 - (1-p)^\ell$$

Par exemple, $\ell = 2$ tables avec $p = 75\,\%$ donnent $P_2 \approx 93{,}7\,\%$.

5 Importance des fonctions de réduction distinctes

L'utilisation de fonctions de réduction *différentes* à chaque colonne est essentielle. Si toutes les R_j sont identiques (table de Hellman classique), deux chaînes qui collisionnent à n'importe quelle position fusionnent définitivement, ce qui augmente considérablement le taux de collisions et réduit drastiquement la couverture effective.

Dans notre implémentation, chaque R_j applique un *tweak* dérivé de j par un générateur pseudo-aléatoire (xorshift) :

$$R_j(c) = \mathtt{trunc}_n(c) \oplus \mathtt{tweak}(j)$$

où $\mathtt{trunc}_n$ extrait les n bits de poids fort du chiffré. Cela garantit que $R_i \neq R_j$ pour $i \neq j$, et donc que deux chaînes ne peuvent fusionner que si elles collisionnent à la **même colonne**.

6 Résumé

Paramètre	**Notation**	**Valeur optimale**
Espace de clés	N	—
Nombre de chaînes (mémoire)	m	$\sim N^{2/3}$
Longueur des chaînes	t	$\sim N^{1/3}$
Temps d'attaque en ligne	$T \approx t^2/2$	$\sim N^{2/3}$
Pré-calcul (une seule fois)	$m \times t$	$\sim N$
Courbe de compromis	$T \cdot M^2$	$= N^2/2$
Probabilité de succès	P	$\approx$ 86–90 %

Les chaînes arc-en-ciel réalisent un compromis temps-mémoire efficace : au lieu de payer N en temps (force brute) ou N en mémoire (table complète), on ne paie que $N^{2/3}$ pour les deux, au prix d'un pré-calcul de coût N effectué une seule fois.

Références

[1] M. E. Hellman, *A Cryptanalytic Time-Memory Trade-Off*, IEEE Transactions on Information Theory, vol. 26, no. 4, pp. 401–406, 1980.

[2] P. Oechslin, *Making a Faster Cryptanalytic Time-Memory Trade-Off*, Advances in Cryptology – CRYPTO 2003, LNCS vol. 2729, pp. 617–630, Springer, 2003.