Des trappes dans les clés ?

conférence SSTIC - juin 2003

Eric Wegrzynowski

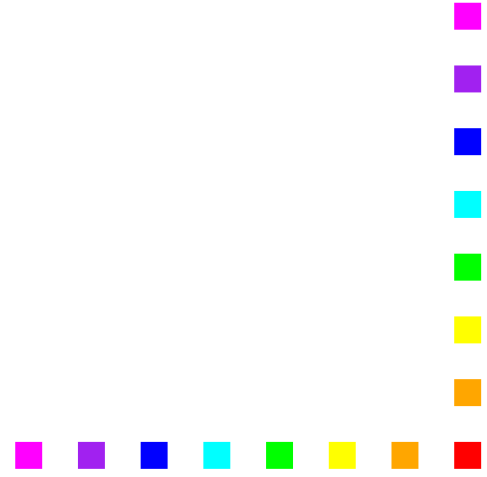
Eric.Wegrzynowski@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille

Université des Sciences et Technologies de Lille

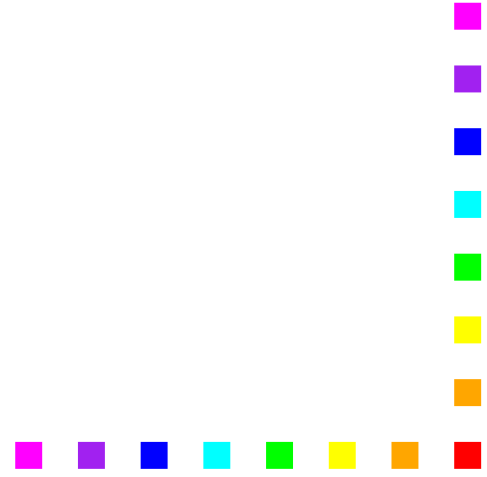
Plan de l’exposé

- Introduction



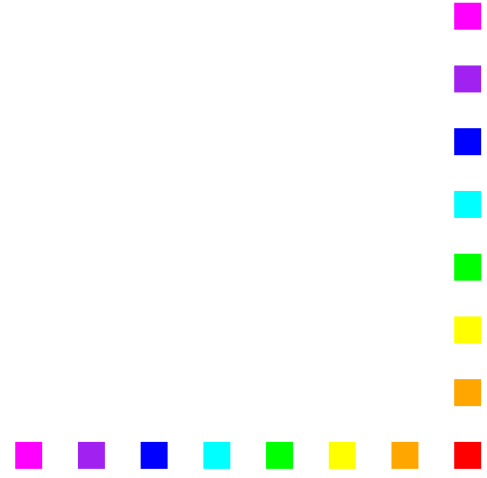
Plan de l’exposé

- Introduction
- Faiblesse des clés ou trappes ?



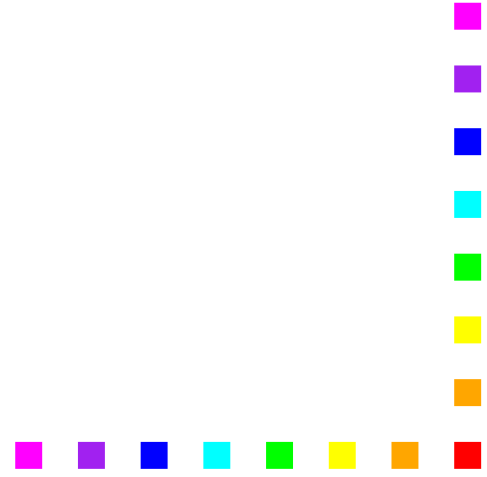
Plan de l’exposé

- Introduction
- Faiblesse des clés ou trappes ?
- Exemple de générateur à trappes



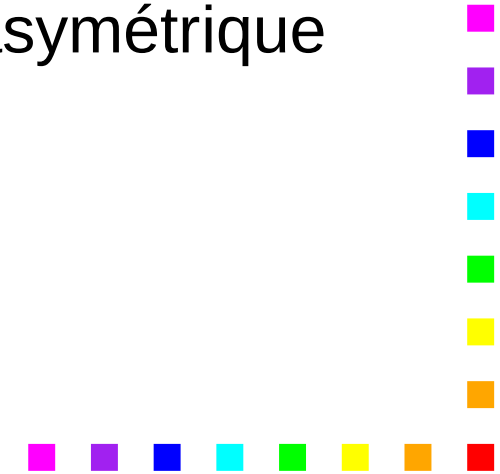
Plan de l’exposé

- Introduction
- Faiblesse des clés ou trappes ?
- Exemple de générateur à trappes
  - de clés de systèmes de chiffrement symétrique



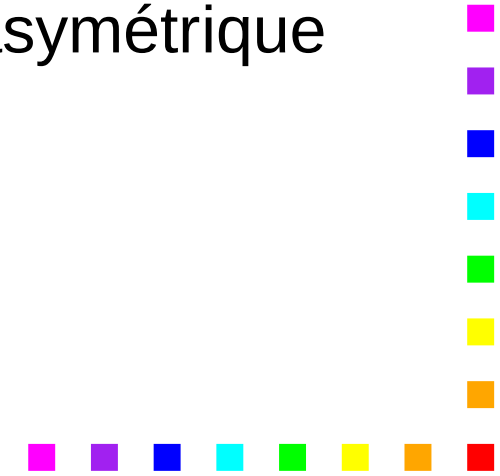
Plan de l’exposé

- Introduction
- Faiblesse des clés ou trappes ?
- Exemple de générateur à trappes
  - de clés de systèmes de chiffrement symétrique
  - de clés de systèmes de chiffrement asymétrique

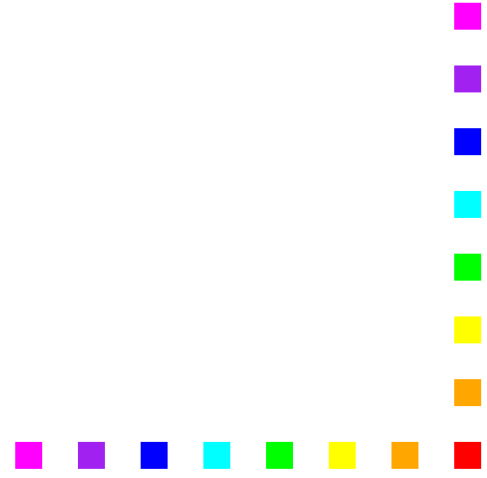


Plan de l’exposé

- Introduction
- Faiblesse des clés ou trappes ?
- Exemple de générateur à trappes
  - de clés de systèmes de chiffrement symétrique
  - de clés de systèmes de chiffrement asymétrique
- Questions en suspens

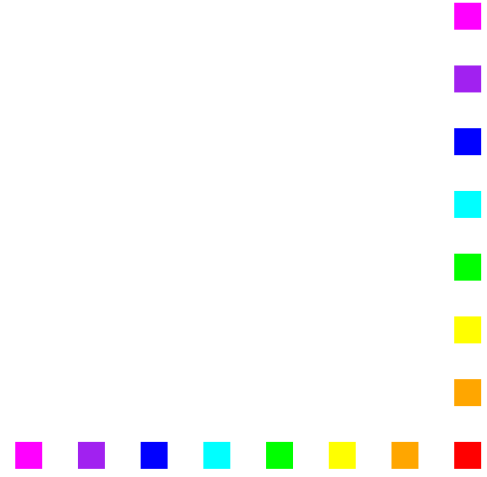


Introduction



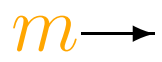
La cryptographie

But premier : assurer la confidentialité des messages



La cryptographie

But premier : assurer la confidentialité des messages

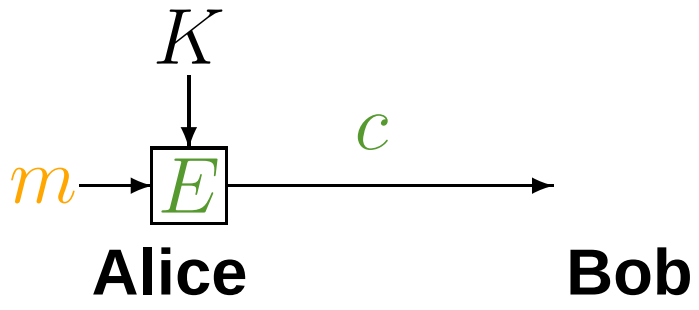


Alice **Bob**

m = message confidentiel

La cryptographie

But premier : assurer la confidentialité des messages



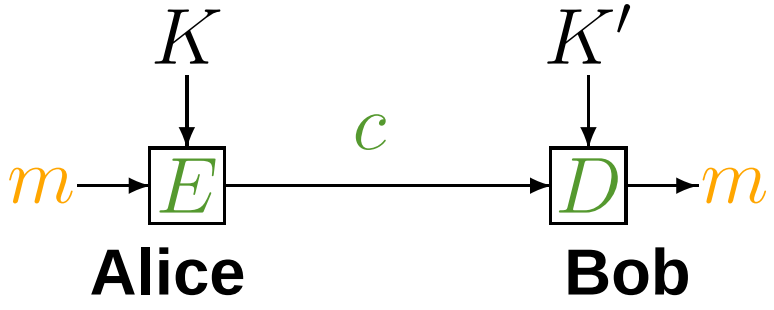
m = message confidentiel

E = système de chiffrement, K = clé de chiffrement

c = message chiffré

La cryptographie

But premier : assurer la confidentialité des messages



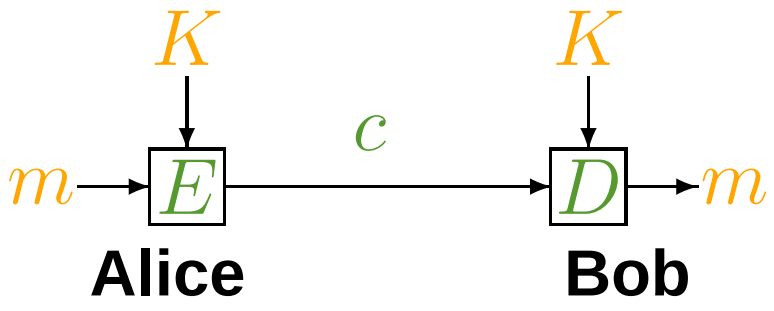
m = message confidentiel

E = système de chiffrement, K = clé de chiffrement

c = message chiffré

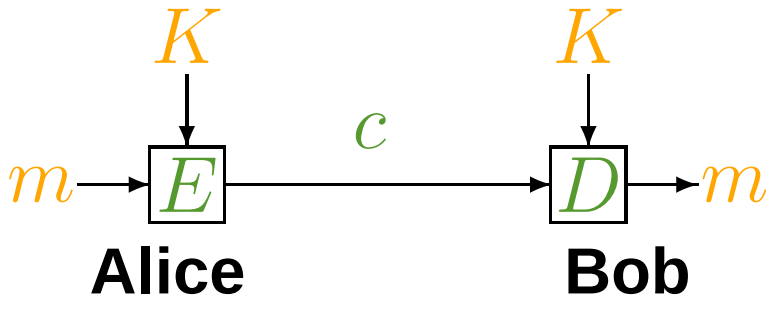
D = système de déchiffrement, K' = clé de déchiffrement

Chiffrement symétrique



clé de chiffrement = clé de déchiffrement ($K = K'$)

Chiffrement symétrique

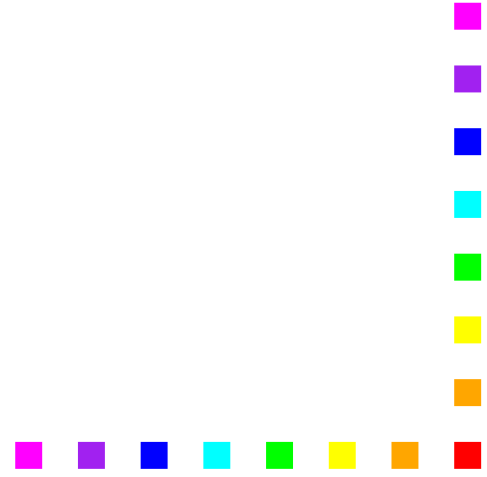


clé de chiffrement = clé de déchiffrement ($K = K'$)

$\Longrightarrow$ clés secrètes

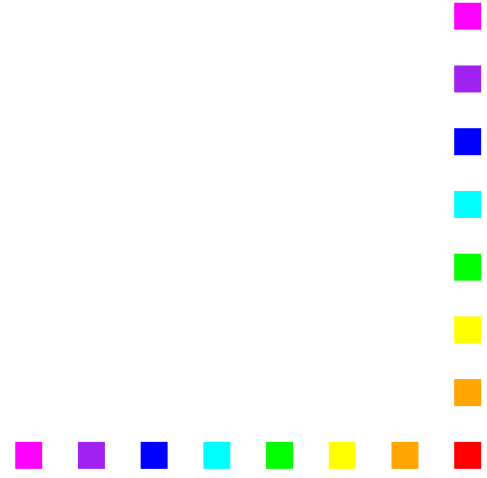
Exemples de chiffrements symétriques

- **D**ata **E**ncryption **S**tandard (DES)



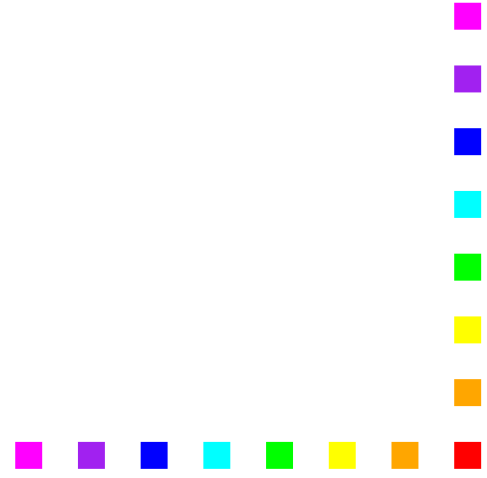
Exemples de chiffrements symétriques

- **D**ata **E**ncryption **S**tandard (DES)
- **A**dvanced **E**ncryption **S**tandard (AES)



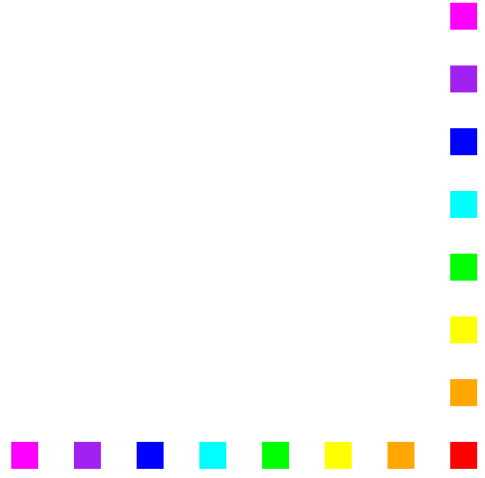
Exemples de chiffrements symétriques

- **D**ata **E**ncryption **S**tandard (DES)
- **A**dvanced **E**ncryption **S**tandard (AES)
- **R**ivest **C**ipher n° **4** (RC4)



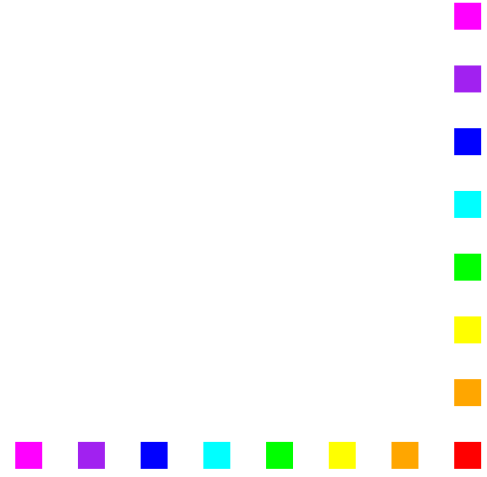
Exemples de chiffrements symétriques

- **D**ata **E**ncryption **S**tandard (DES)
- **A**dvanced **E**ncryption **S**tandard (AES)
- **R**ivest **C**ipher n° **4** (RC4)
- et beaucoup d'autres encore . . .



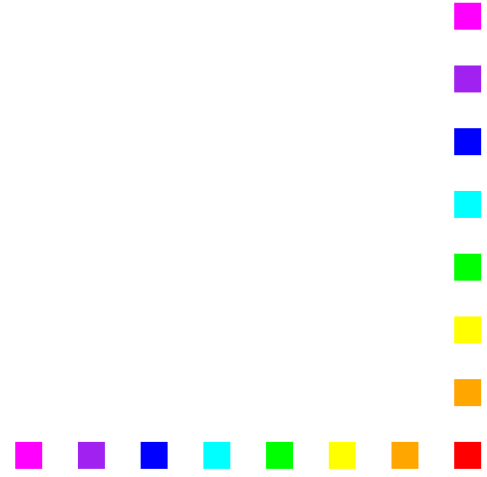
Inconvénients du chiffrement symétrique

- Nécessité de partager un secret commun :
  la clé



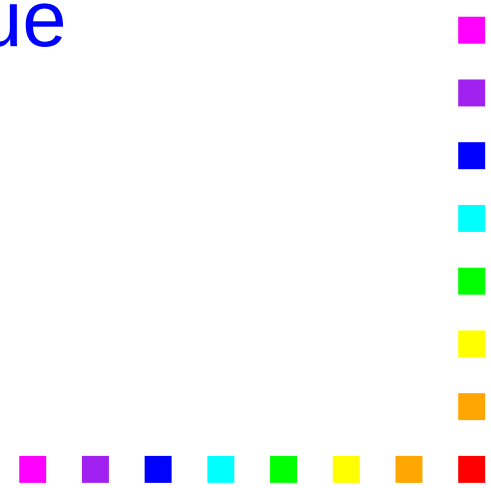
Inconvénients du chiffrement symétrique

- Nécessité de partager un secret commun :
  la clé
- Comment convenir d'une clé ?

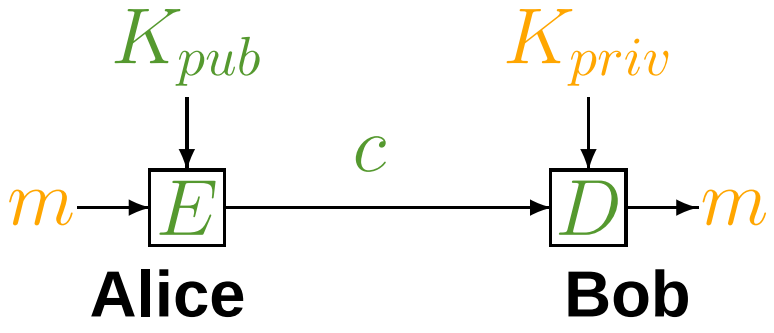


Inconvénients du chiffrement symétrique

- Nécessité de partager un secret commun :
  la clé
- Comment convenir d'une clé ?
- Une réponse :
  la cryptographie asymétrique

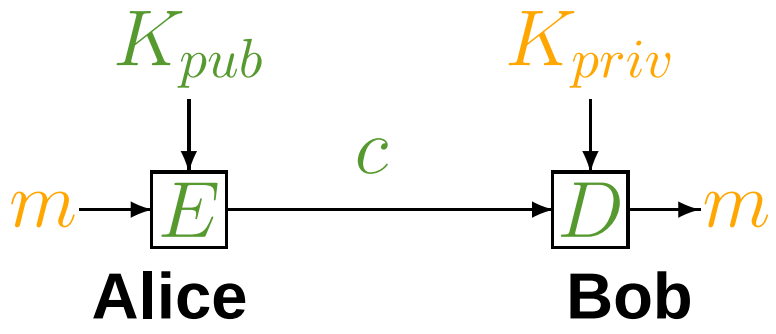


Chiffrement asymétrique



clé de chiffrement $\neq$ clé de déchiffrement ($K_{pub} \neq K_{priv}$)

Chiffrement asymétrique



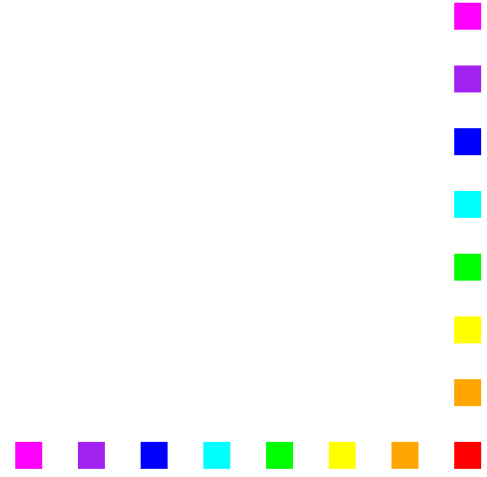
clé de chiffrement $\neq$ clé de déchiffrement ($K_{pub} \neq K_{priv}$)

K_{pub} = clé de chiffrement publique

K_{priv} = clé de déchiffrement privée

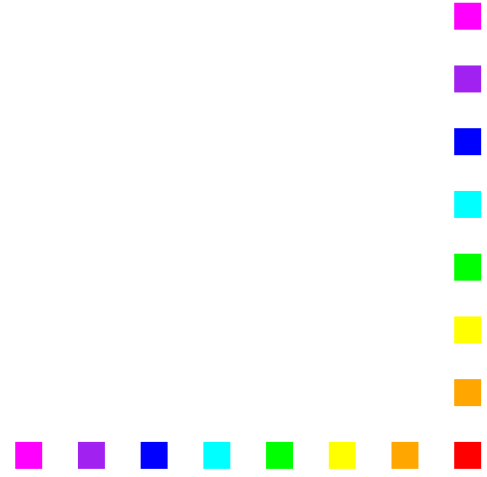
Exemples de chiffrements asymétriques

- **R**ivest, **S**hamir, **A**dleman (RSA)



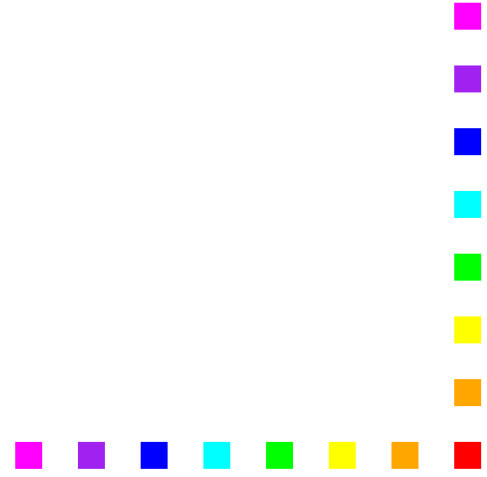
Exemples de chiffrements asymétriques

- **R**ivest, **S**hamir, **A**dleman (RSA)
- El Gamal



Exemples de chiffrements asymétriques

- **R**ivest, **S**hamir, **A**dleman (RSA)
- El Gamal
- courbes elliptiques

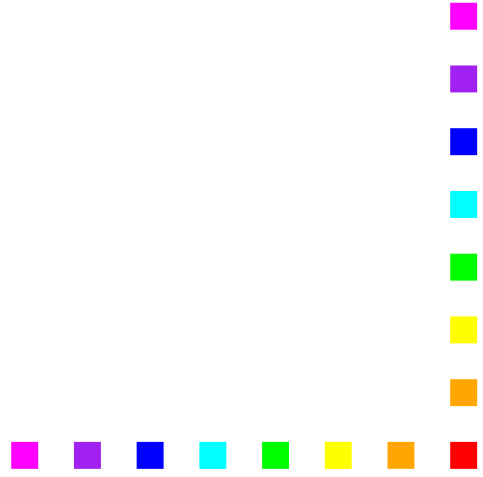


Exemples de chiffrements asymétriques

- **R**ivest, **S**hamir, **A**dleman (RSA)
- El Gamal
- courbes elliptiques
- et beaucoup d’autres encore . . .

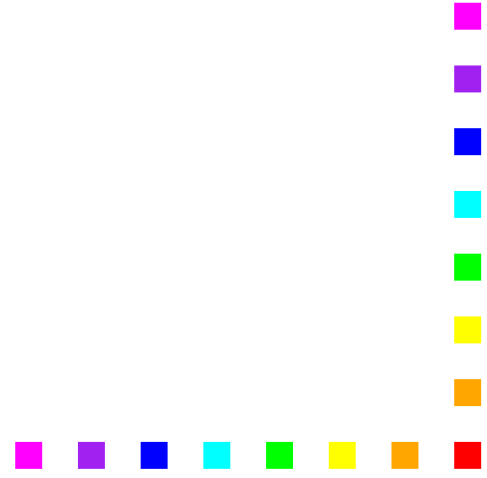
Inconvénients du chiffrement asymétrique

- opérations de chiffrement/déchiffrement plus lentes qu'en crypto à clés secrètes



Inconvénients du chiffrement asymétrique

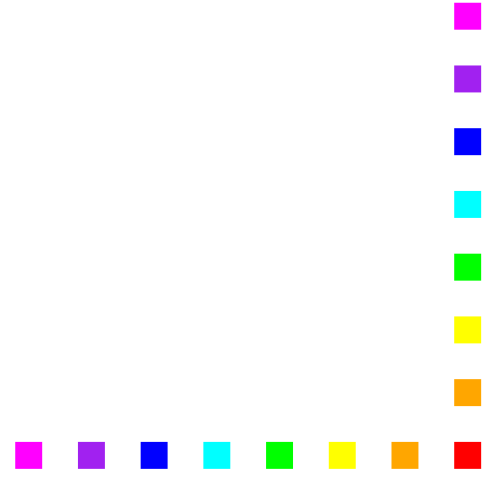
- opérations de chiffrement/déchiffrement plus lentes qu'en crypto à clés secrètes
- dans la pratique, utilisation des clés publiques pour échanger une clé d'un système de chiffrement conventionnel



Quelle sécurité ?

Sécurité assurée si

- bons algos de chiffrement/déchiffrement

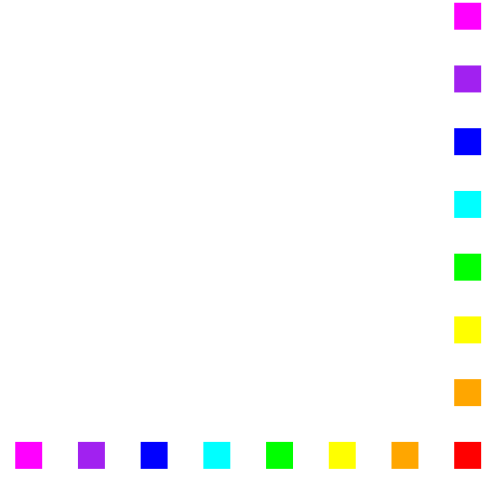


Quelle sécurité ?

Sécurité assurée si

- bons algos de chiffrement/déchiffrement
- principe de Kerkhoffs (1883) :

  la sécurité ne doit reposer que sur le secret de la clé.

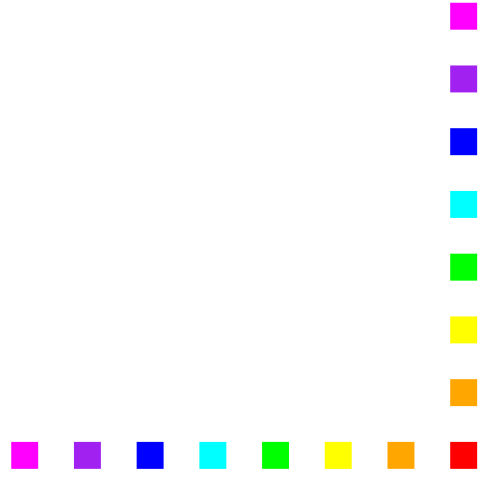


Quelle sécurité ?

Sécurité assurée si

- bons algos de chiffrement/déchiffrement
- principe de Kerkhoffs (1883) :

  la sécurité ne doit reposer que sur le secret de la clé.
- un espace de clés suffisamment grand



Quelle sécurité ?

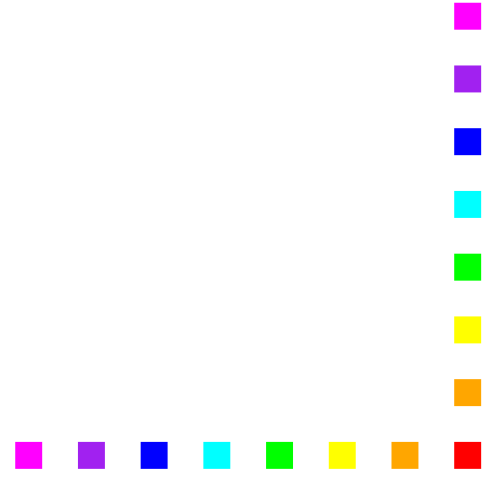
Sécurité assurée si

- bons algos de chiffrement/déchiffrement
- principe de Kerkhoffs (1883) :

  la sécurité ne doit reposer que sur le secret de la clé.
- un espace de clés suffisamment grand
- bons générateurs de clés cryptographiques

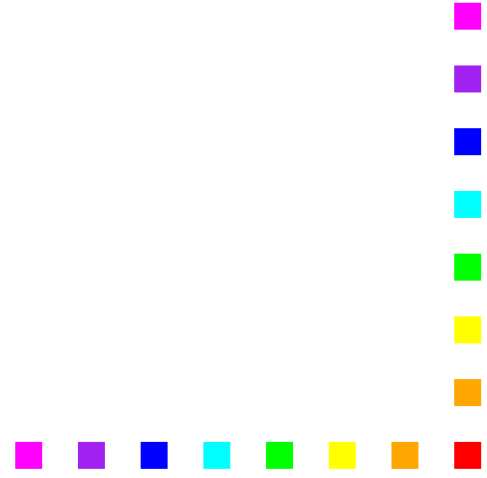
mais . . .

- comment les clés sont-elles fabriquées ?



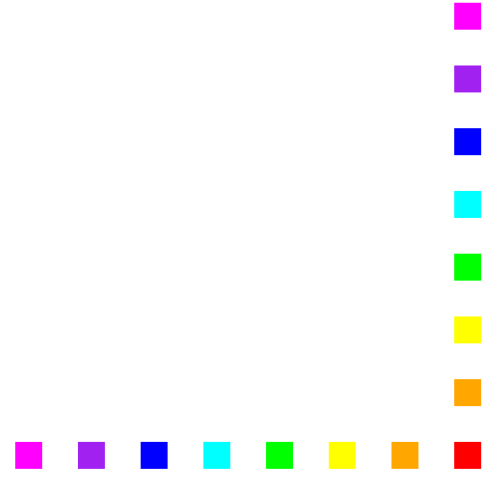
mais . . .

- comment les clés sont-elles fabriquées ?
- taille des clés = mesure de sécurité ?



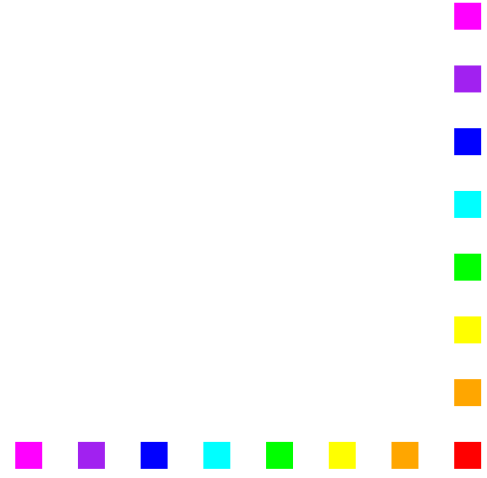
mais . . .

- comment les clés sont-elles fabriquées ?
- taille des clés = mesure de sécurité ?
- qualité des générateurs ?



mais . . .

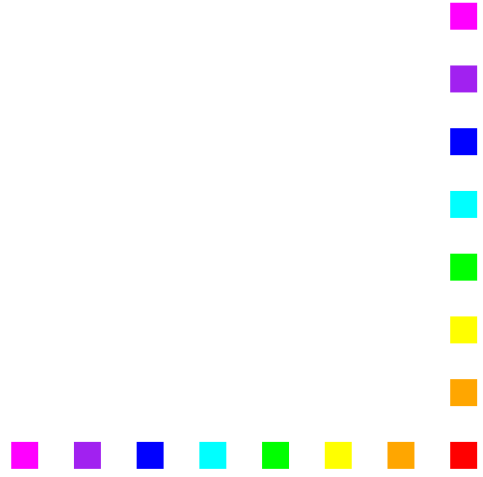
- comment les clés sont-elles fabriquées ?
- taille des clés = mesure de sécurité ?
- qualité des générateurs ?
- peut-on faire confiance aux outils de chiffrement ?



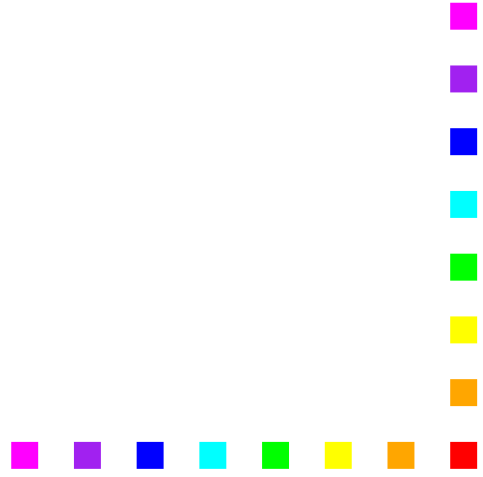
mais . . .

- comment les clés sont-elles fabriquées ?
- taille des clés = mesure de sécurité ?
- qualité des générateurs ?
- peut-on faire confiance aux outils de chiffrement ?

  cf affaire CryptoAG

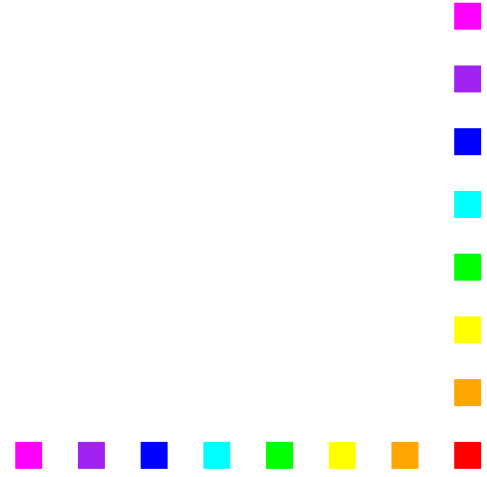


Clés faibles, trappes



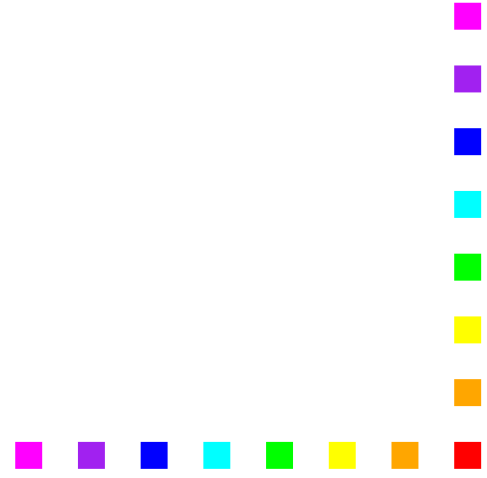
Qu’est-ce qu’une clé ?

- Clés symétriques : une chaîne de bits quelconque



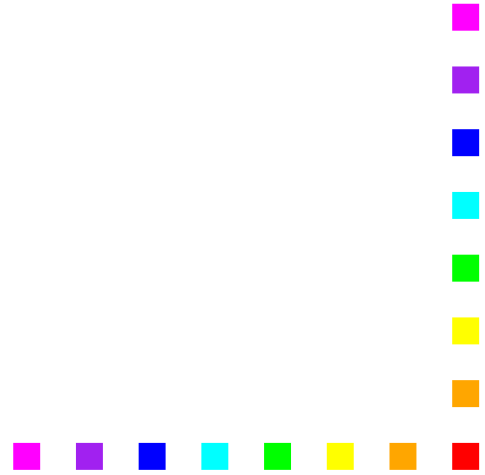
Qu’est-ce qu’une clé ?

- Clés symétriques : une chaîne de bits quelconque
- le plus souvent d’une taille n fixée ($n = 56$ bits pour le DES, 128, 192 ou 256 pour l’AES)



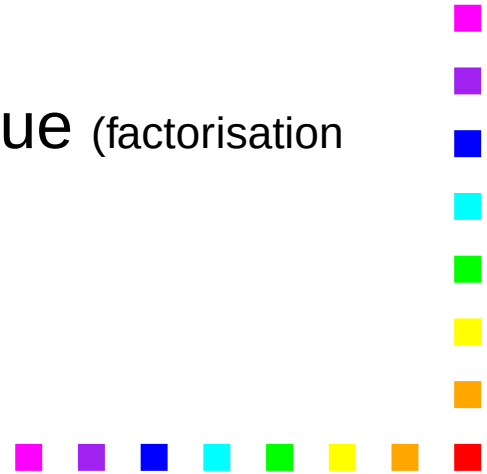
Qu’est-ce qu’une clé ?

- Clés symétriques : une chaîne de bits quelconque
- le plus souvent d’une taille n fixée ($n = 56$ bits pour le DES, 128, 192 ou 256 pour l’AES)
- Clés asymétriques : des données structurées (des entiers pour RSA)



Qu’est-ce qu’une clé ?

- Clés symétriques : une chaîne de bits quelconque
- le plus souvent d’une taille n fixée ($n = 56$ bits pour le DES, 128, 192 ou 256 pour l’AES)
- Clés asymétriques : des données structurées (des entiers pour RSA)
- rendant difficile un problème mathématique (factorisation pour RSA)

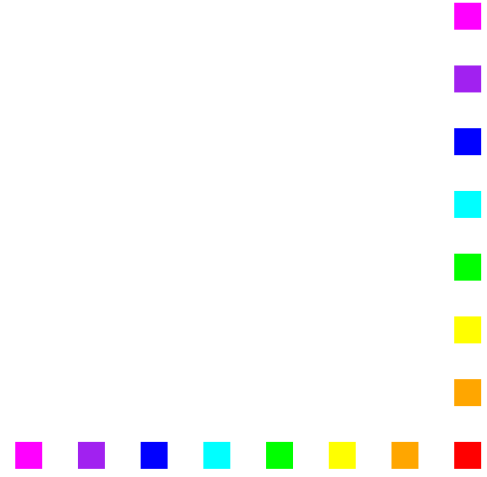


Qu’est-ce qu’une clé ?

- Clés symétriques : une chaîne de bits quelconque
- le plus souvent d’une taille n fixée ($n = 56$ bits pour le DES, 128, 192 ou 256 pour l’AES)
- Clés asymétriques : des données structurées (des entiers pour RSA)
- rendant difficile un problème mathématique (factorisation pour RSA)
- la taille est liée à la difficulté du problème à traiter (au moins 1024 bits pour RSA)

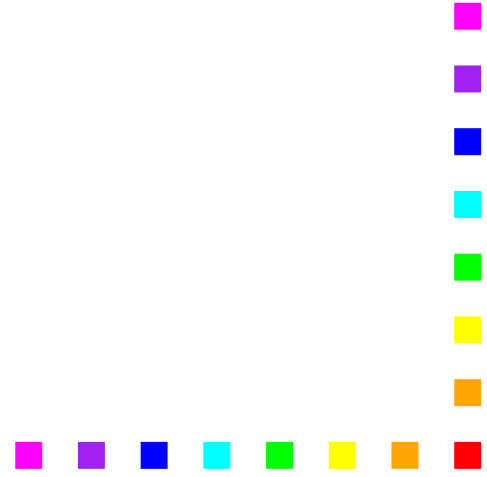
Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé



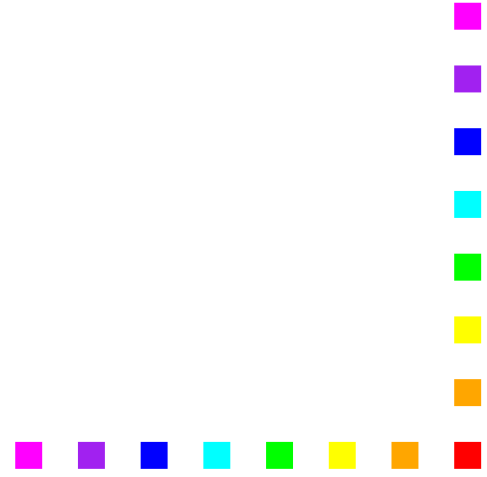
Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé
- nbre de clés = fonction de la taille



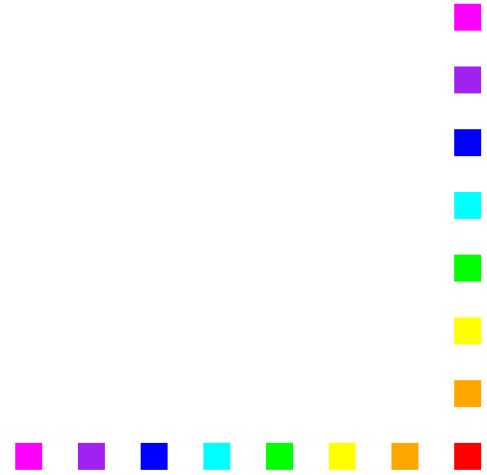
Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé
- nbre de clés = fonction de la taille
- nbre de clés = 2^n dans le cas des clés symétriques



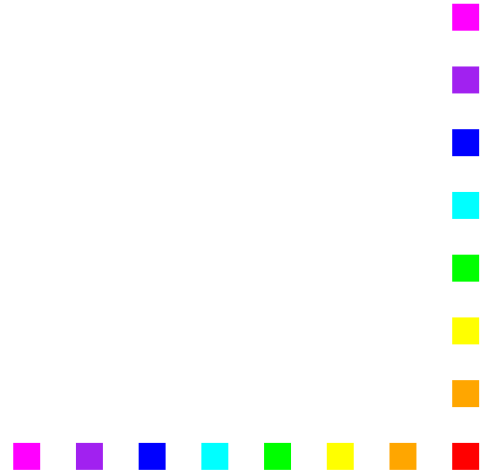
Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé
- nbre de clés = fonction de la taille
- nbre de clés = 2^n dans le cas des clés symétriques
- Entropie = nbre moyen de questions binaires pour déterminer une clé



Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé
- nbre de clés = fonction de la taille
- nbre de clés = 2^n dans le cas des clés symétriques
- Entropie = nbre moyen de questions binaires pour déterminer une clé
- entropie $\leq$ taille

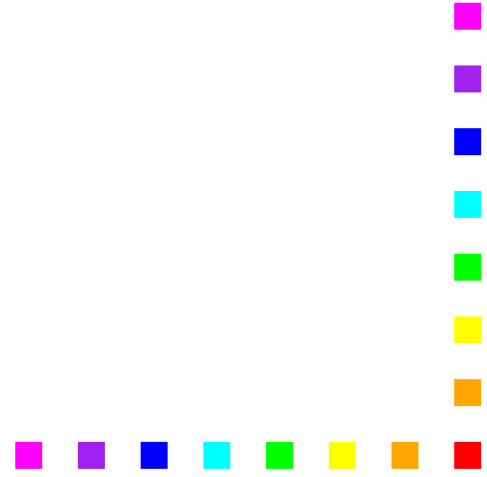


Taille vs Entropie

- Taille = nbre n de bits pour représenter une clé
- nbre de clés = fonction de la taille
- nbre de clés = 2^n dans le cas des clés symétriques
- Entropie = nbre moyen de questions binaires pour déterminer une clé
- entropie $\leq$ taille
- entropie = taille dans le cas des clés symétriques, si toutes clés équiprobables

Condition nécessaire

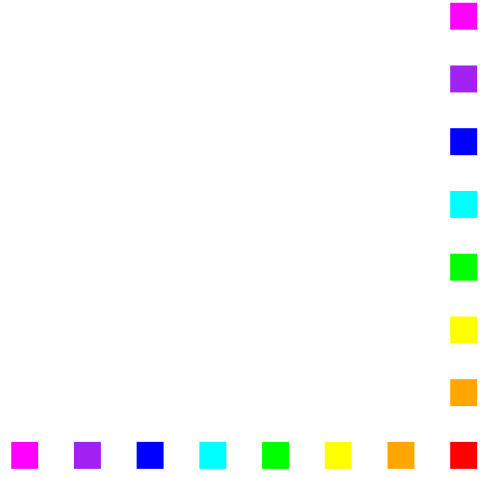
Critère important = entropie



Condition nécessaire

Critère important = entropie

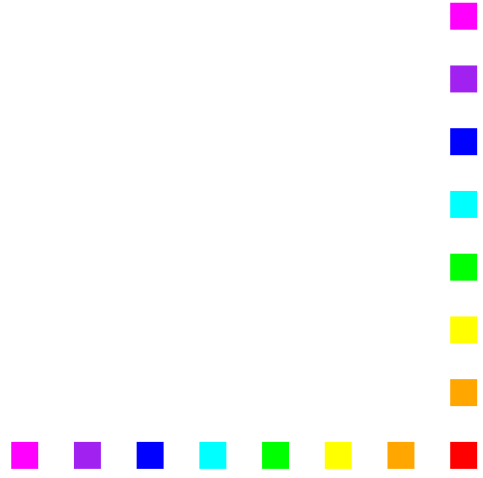
- entropie maximale $\Rightarrow$ bon générateur de clé



Condition nécessaire

Critère important = entropie

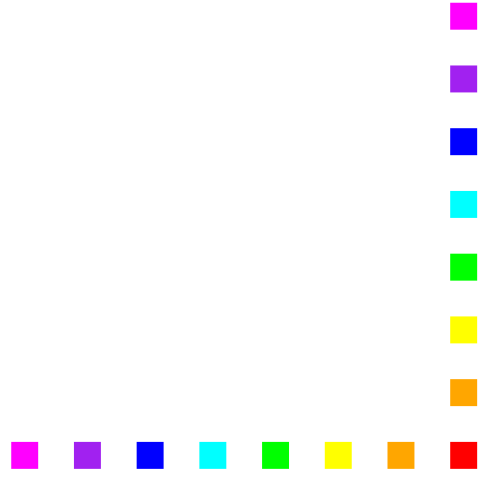
- entropie maximale $\Rightarrow$ bon générateur de clé
- $\Rightarrow$ distribution de probabilité uniforme sur l'espace des clés



Condition nécessaire

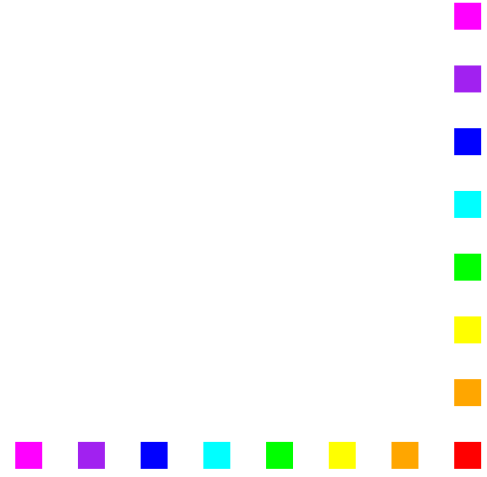
Critère important = entropie

- entropie maximale $\Rightarrow$ bon générateur de clé
- $\Rightarrow$ distribution de probabilité uniforme sur l'espace des clés
- $\Rightarrow$ bon générateur d'alea



Clés faibles ou trappes ?

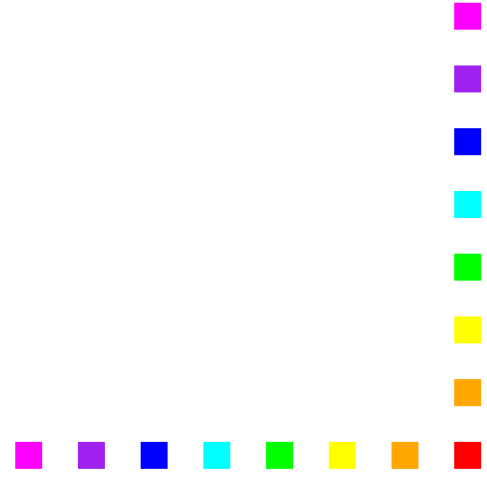
Mauvais générateur de clés = générateur à entropie non maximale



Clés faibles ou trappes ?

Mauvais générateur de clés = générateur à entropie non maximale

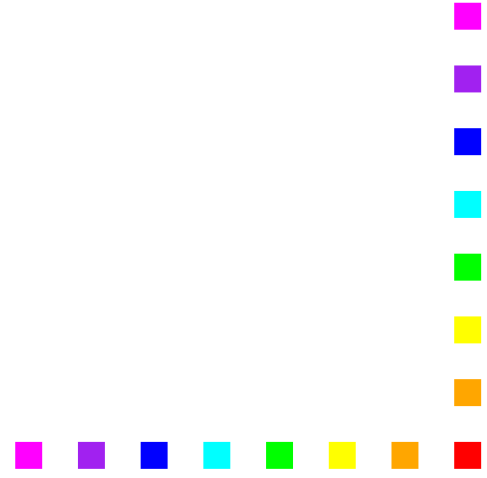
- non intentionnel : générateur faible



Clés faibles ou trappes ?

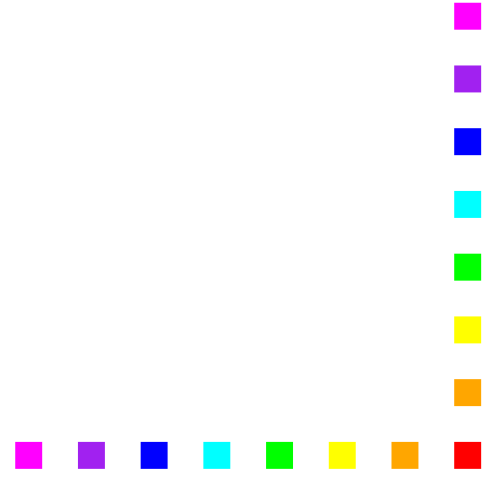
Mauvais générateur de clés = générateur à entropie non maximale

- non intentionnel : générateur faible
- intentionnel : générateur à trappes



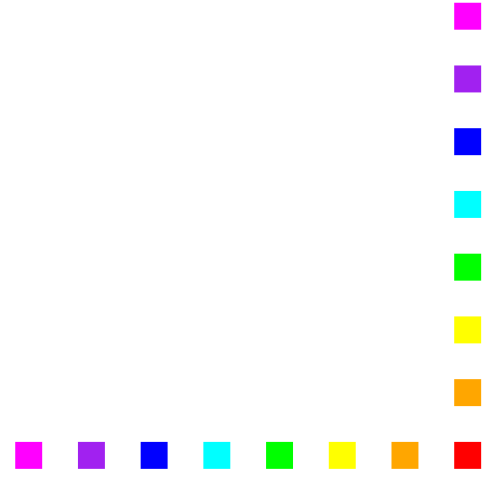
Comment ?

- choix d’une distribution de probabilité biaisée (non uniforme) sur l’espace des clés



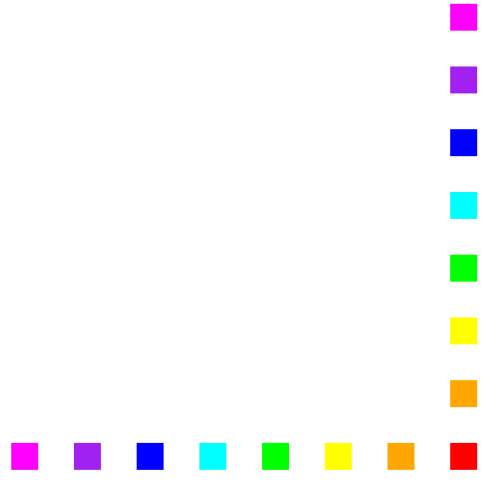
Comment ?

- choix d’une distribution de probabilité biaisée (non uniforme) sur l’espace des clés
- limitation de l’espace des clés possibles



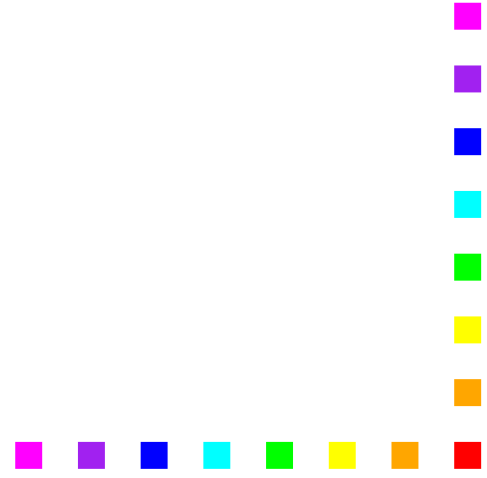
Des trappes dans les clés secrètes ?

Un exemple



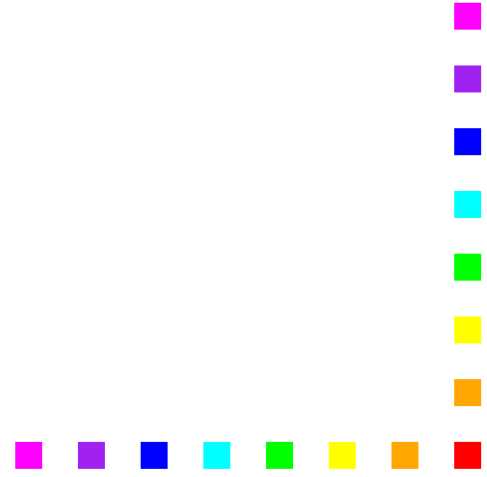
Principe

- limiter l’espace des clés possibles



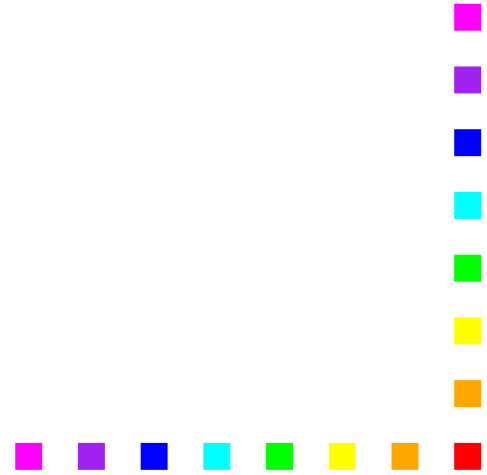
Principe

- limiter l’espace des clés possibles
- de façon la moins visible possible



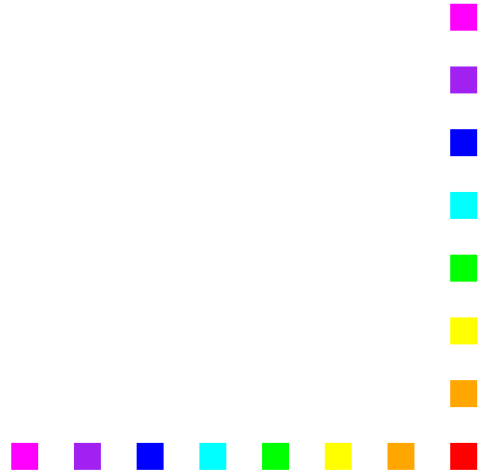
Utilisation de codes linéaires

- choix de k clés de longueur n (linéairement indépendantes) :
  $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ $(k < n)$



Utilisation de codes linéaires

- choix de k clés de longueur n (linéairement indépendantes) : $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ $(k < n)$
- tirage au sort de k bits aléatoires $b_1, b_2, \ldots, b_k$



Utilisation de codes linéaires

- choix de k clés de longueur n (linéairement indépendantes) : $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ $(k < n)$
- tirage au sort de k bits aléatoires $b_1, b_2, \ldots, b_k$
- combinaison linéaire des clés de base avec ces bits

$$\mathbf{c} = \sum_{i=1}^{k} b_i.\mathbf{c}_i$$

Utilisation de codes linéaires

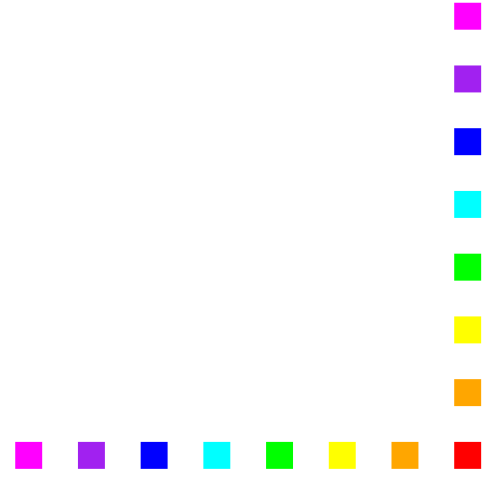
- choix de k clés de longueur n (linéairement indépendantes) : $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ $(k < n)$
- tirage au sort de k bits aléatoires $b_1, b_2, \ldots, b_k$
- combinaison linéaire des clés de base avec ces bits

$$\mathbf{c} = \sum_{i=1}^{k} b_i.\mathbf{c}_i$$

espace des clés générées = sous-espace de dimension k

Exemple

- longueur des clés $n = 10$, nombre de clés fixées $k = 4$

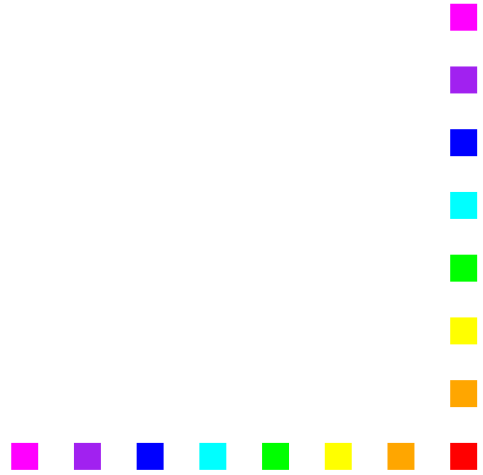


Exemple

- longueur des clés $n = 10$, nombre de clés fixées $k = 4$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$



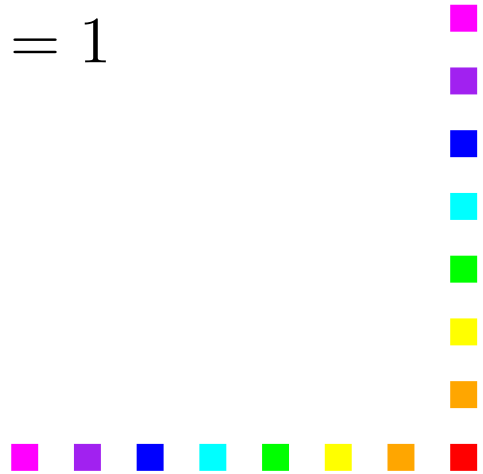
Exemple

- longueur des clés $n = 10$, nombre de clés fixées $k = 4$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$

- bits tirés au sort : $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1$



Exemple

- longueur des clés $n = 10$, nombre de clés fixées $k = 4$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$

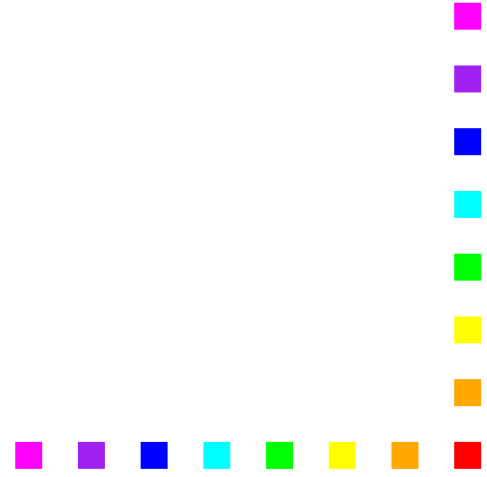
- bits tirés au sort : $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1$
- clé générée

$$010001011 + 001010101$$

$$+000101100 \quad = \quad 011110010$$

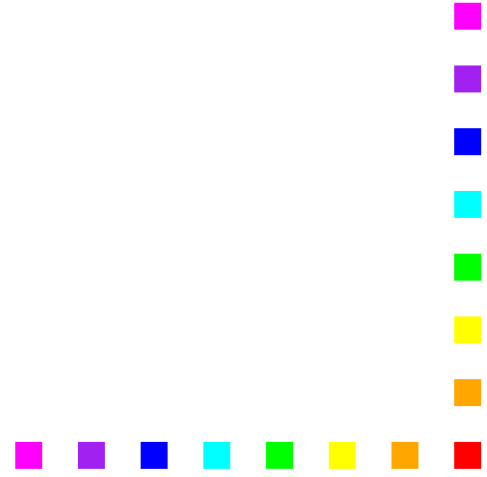
Exploitation de la trappe

- espace des clés limité à 2^k clés (au lieu de 2^n)



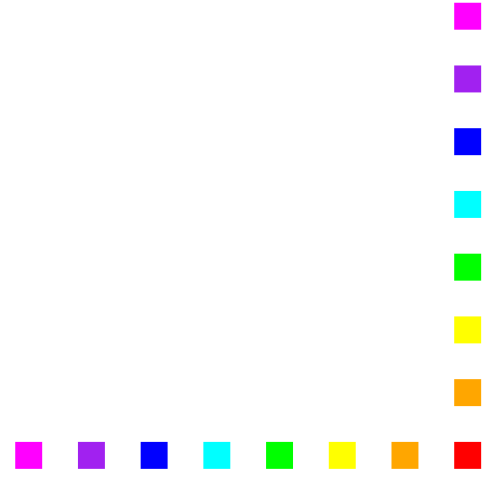
Exploitation de la trappe

- espace des clés limité à 2^k clés (au lieu de 2^n)
- $\Rightarrow$ entropie maximale = k bits (au lieu de n)



Exploitation de la trappe

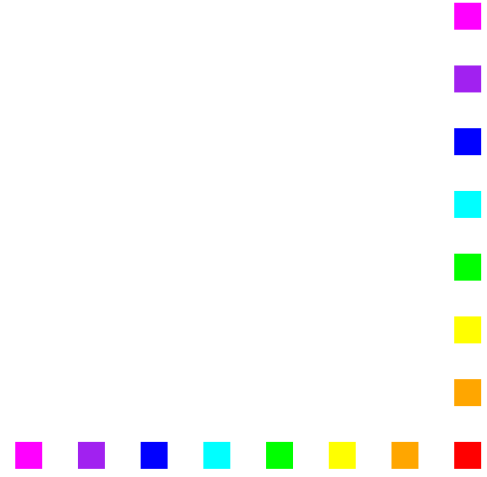
- espace des clés limité à 2^k clés (au lieu de 2^n)
- $\Rightarrow$ entropie maximale = k bits (au lieu de n)
- $\Rightarrow$ recherche exhaustive envisageable si $k < 60$



Détection de la trappe

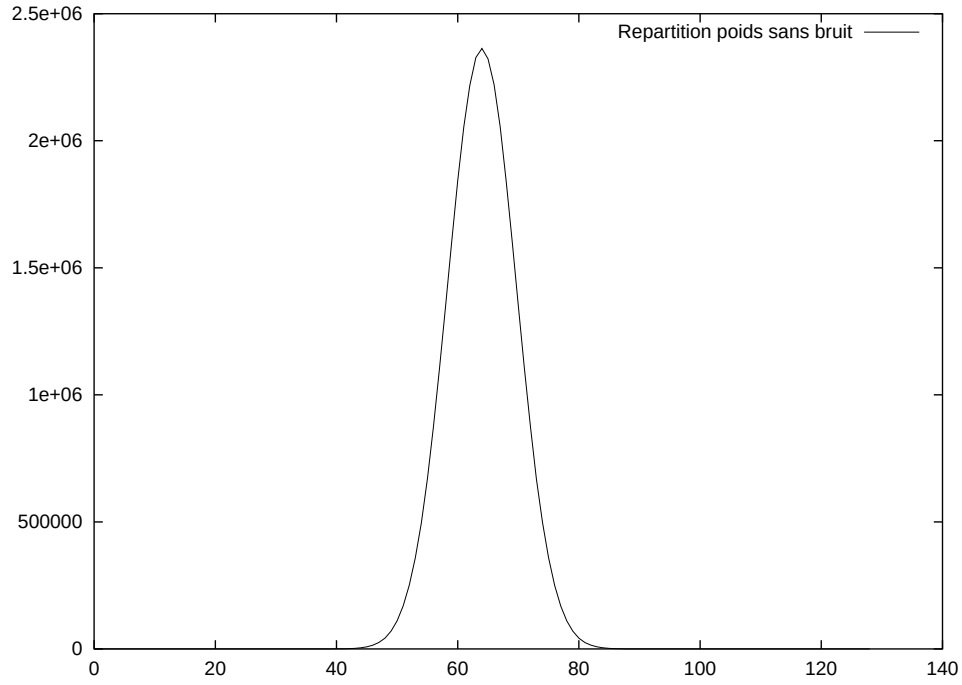
Si clés de base bien choisies, bonnes propriétés statistiques des motifs

- équilibre
- poids
- …



Répartition des poids

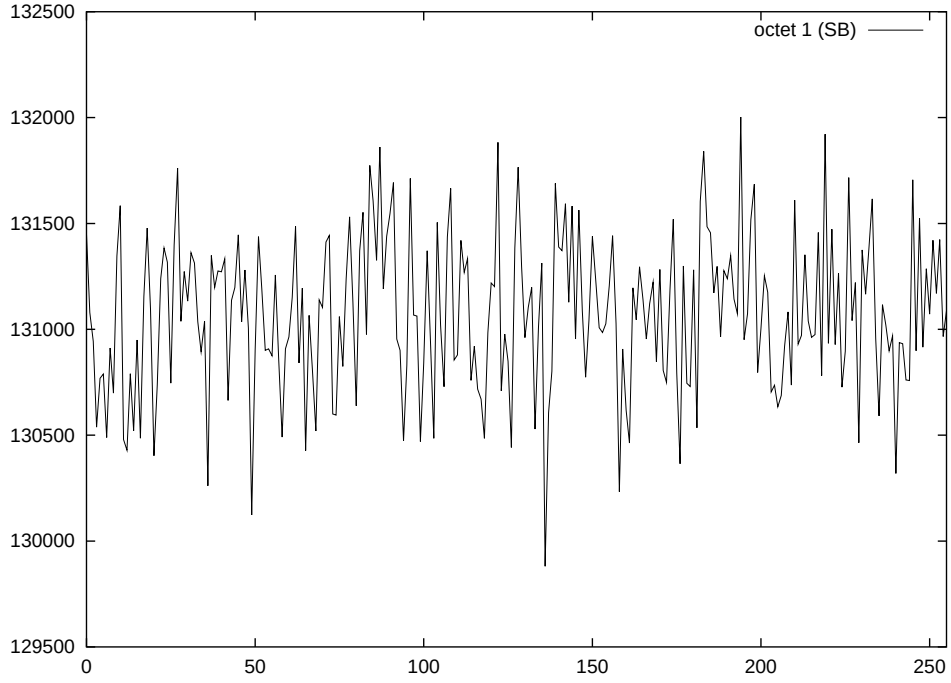
Longueur des clés $n = 128$, dimension du sous-espace $k = 50$, taille échantillon =2^{25}



Répartition selon le poids

Répartition par octets

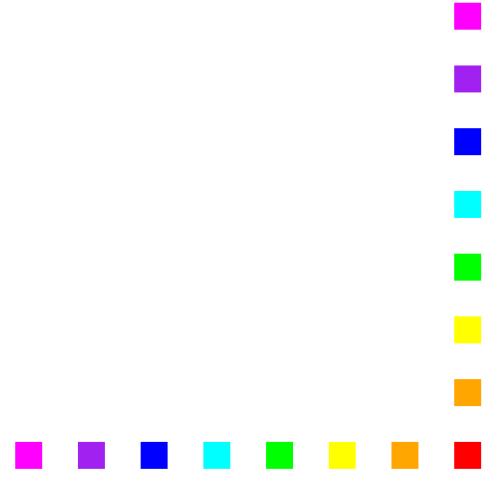
Longueur des clés $n = 128$, dimension du sous-espace $k = 50$, taille échantillon =2^{25}



Répartition selon le premier octet

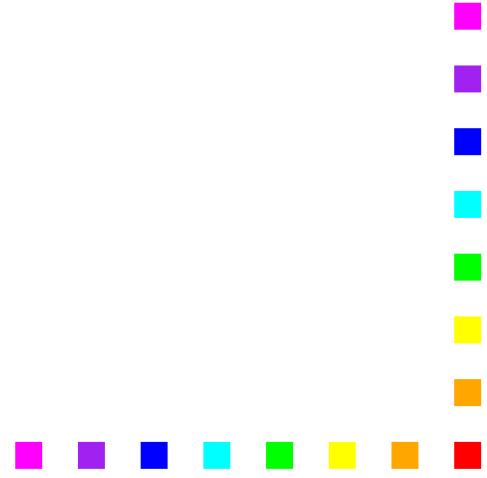
Mais . . .

- détection algébrique aisée



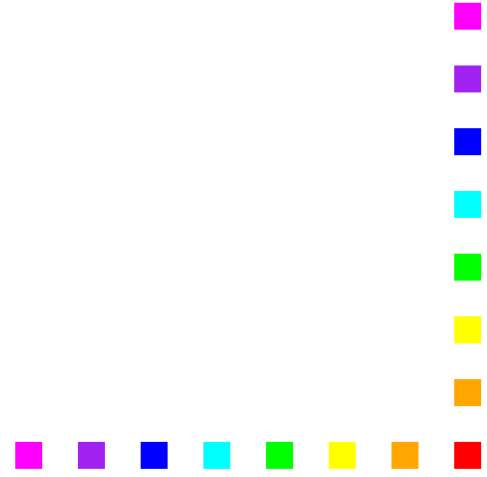
Mais . . .

- détection algébrique aisée
- par calcul de rang de matrices



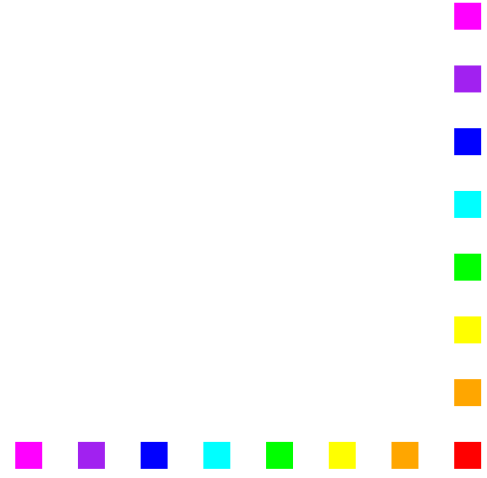
Mais . . .

- détection algébrique aisée
- par calcul de rang de matrices
- besoin de générer $n + l$ clés pour détecter le biais (proba fausse alarme $< 1/1000$ si $l = 10$)



Amélioration de la trappe

- en introduisant du bruit



Amélioration de la trappe

- en introduisant du bruit
- clé générée = combinaison linéaire des clés de base + bruit aléatoire

$$\mathbf{c} = \sum_{i=1}^{k} b_i.\mathbf{c}_i + \mathbf{e}$$

Amélioration de la trappe

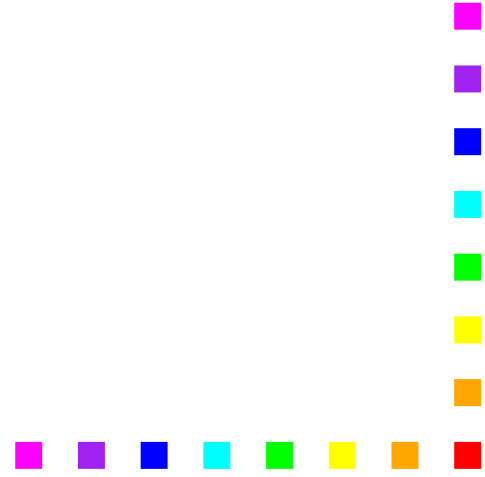
- en introduisant du bruit
- clé générée = combinaison linéaire des clés de base + bruit aléatoire

$$\mathbf{c} = \sum_{i=1}^{k} b_i.\mathbf{c}_i + \mathbf{e}$$

- paramètres du générateur = clés de base + poids du bruit $\leq t$

Exemple

- $n = 10$, $k = 4$, $t = 1$

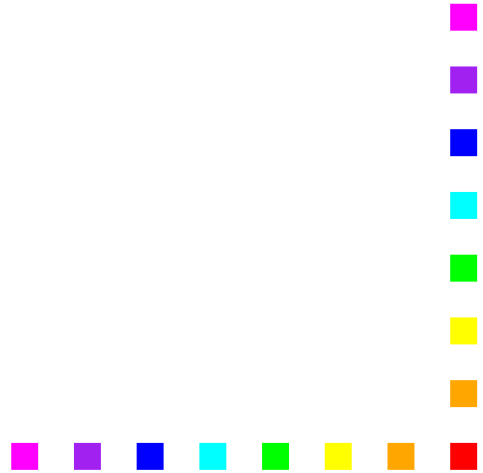


Exemple

- $n = 10$, $k = 4$, $t = 1$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$



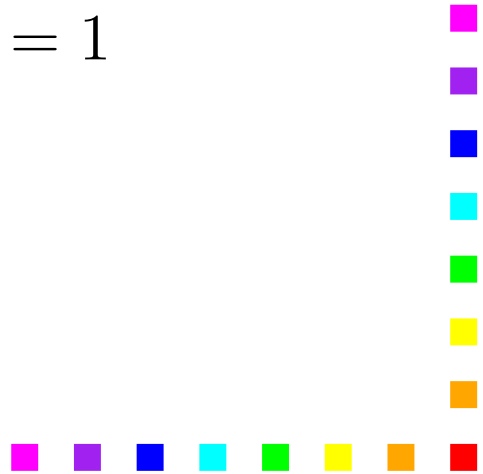
Exemple

- $n = 10$, $k = 4$, $t = 1$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$

- bits tirés au sort : $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1$



Exemple

- $n = 10$, $k = 4$, $t = 1$
- clés de base

$$\mathbf{c}_1 = 100011001, \mathbf{c}_2 = 010001011,$$

$$\mathbf{c}_3 = 001010101, \mathbf{c}_4 = 000101100$$

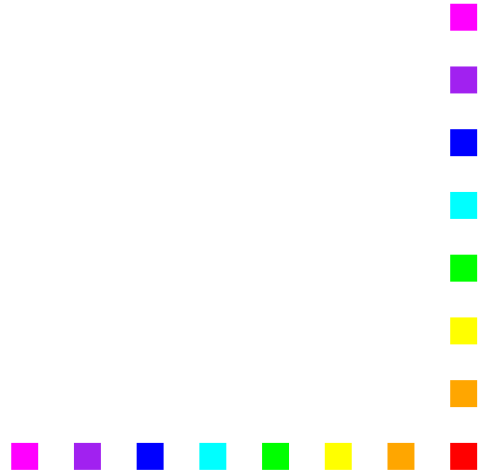
- bits tirés au sort : $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 1$
- clé générée

$$010001011 + 001010101$$

$$+000101100+ \overbrace{0100000000}^{\text{bruitalatoire}} = 001110010$$

Exploitation de la trappe

- espace des clés limité à $2^k . \sum_{i=0}^{t} \left(\begin{array}{c} n \\ i \end{array} \right)$ clés



Exploitation de la trappe

- espace des clés limité à $2^k . \sum_{i=0}^{t} \left(\begin{array}{c} n \\ i \end{array} \right)$ clés
- $\Rightarrow$ entropie maximale = $k + \log_2 \sum_{i=0}^{t} \left(\begin{array}{c} n \\ i \end{array} \right)$ bits

  ($\approx k + \log_2 n$ si $t = 1$)

Exploitation de la trappe

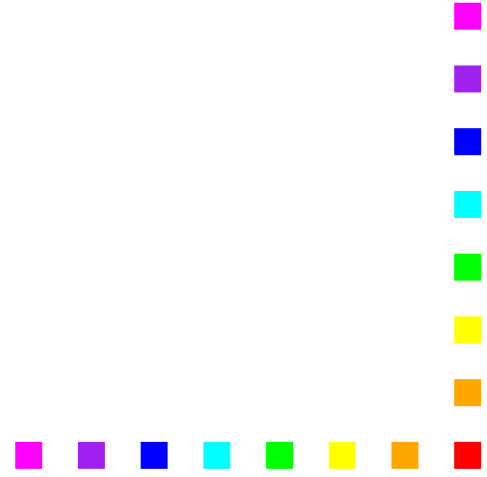
- espace des clés limité à $2^k . \sum_{i=0}^{t} \left(\begin{array}{c} n \\ i \end{array} \right)$ clés
- $\Rightarrow$ entropie maximale = $k + \log_2 \sum_{i=0}^{t} \left(\begin{array}{c} n \\ i \end{array} \right)$ bits

  ($\approx k + \log_2 n$ si $t = 1$)
- $\Rightarrow$ recherche exhaustive envisageable pour k, t convenablement choisis

Détection de la trappe

Si clés de base bien choisies, bonnes propriétés statistiques des motifs

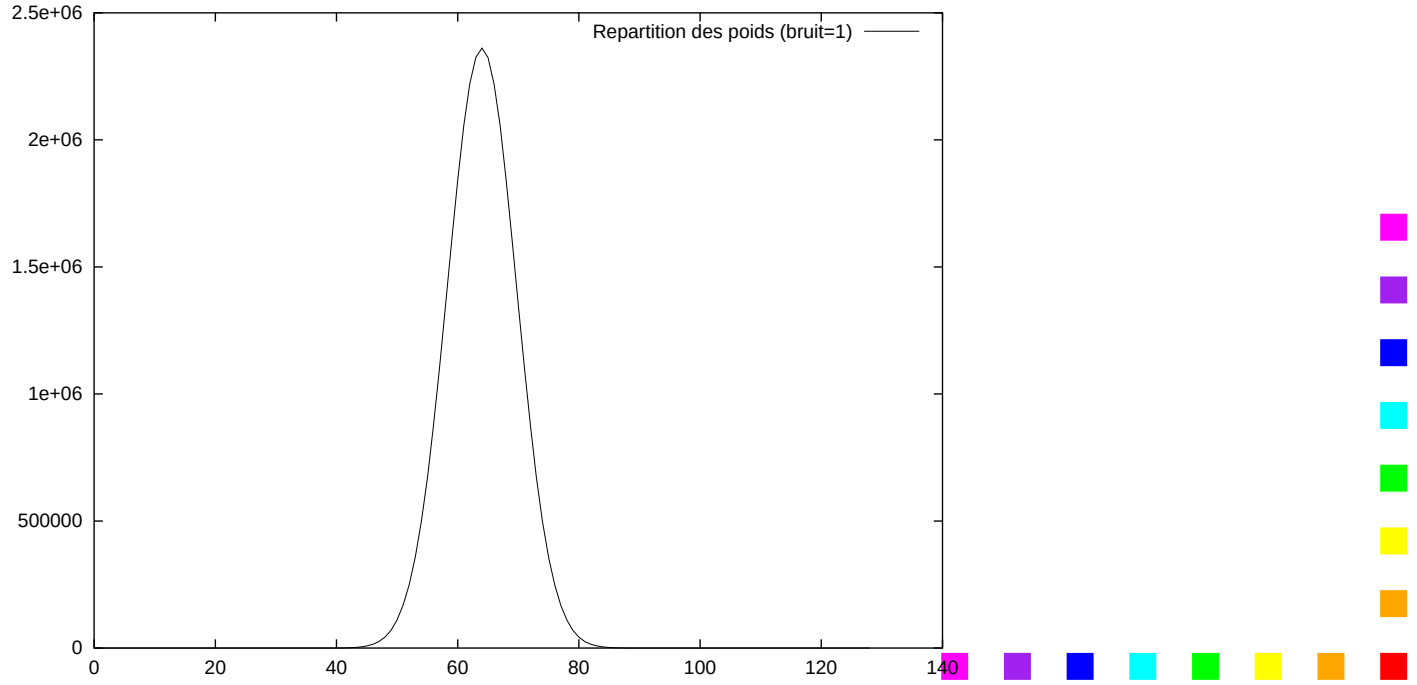
- équilibre
- poids
- détection par calcul de rang impossible



Répartition des poids

Longueur des clés $n = 128$, dimension du sous-espace $k = 50$, taille échantillon =2^{25},

poids bruit $t = 1$

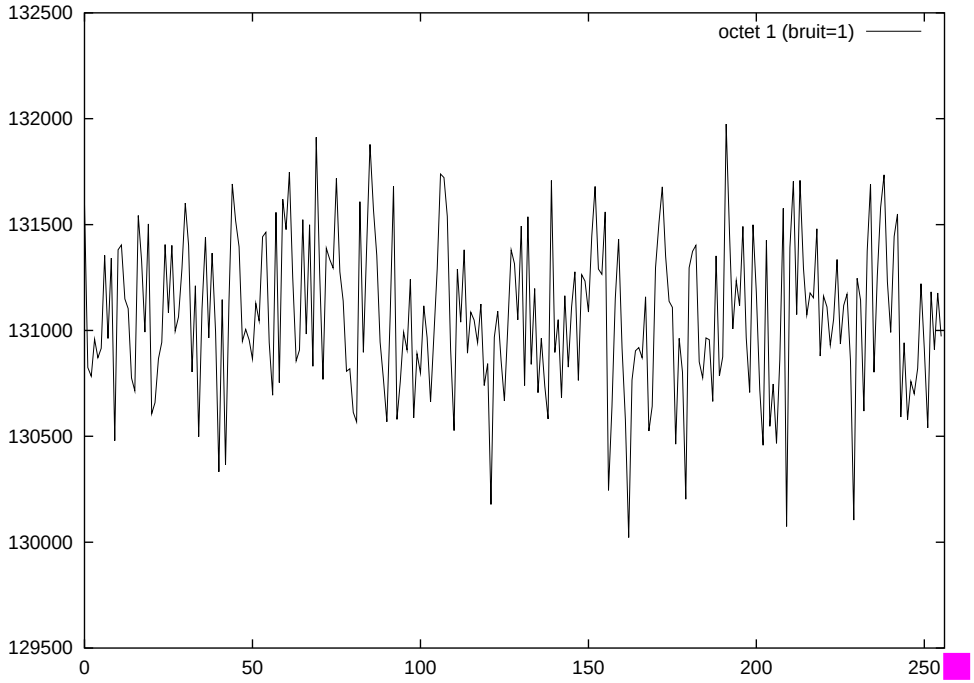


Répartition selon le poids

Répartition par octets

Longueur des clés $n = 128$, dimension du sous-espace $k = 50$, taille échantillon =2^{25},

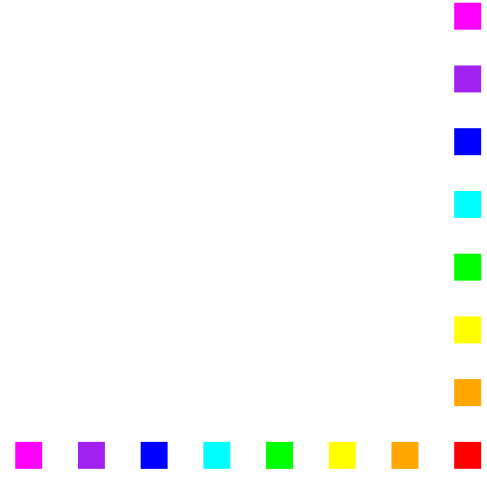
poids bruit $t = 1$



Répartition selon le premier octet

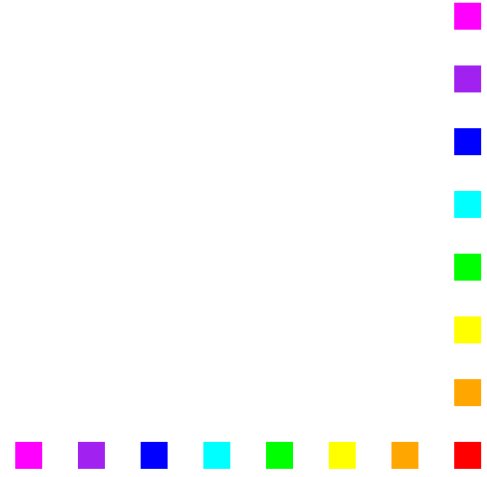
Mais . . .

- détection de collisions possible



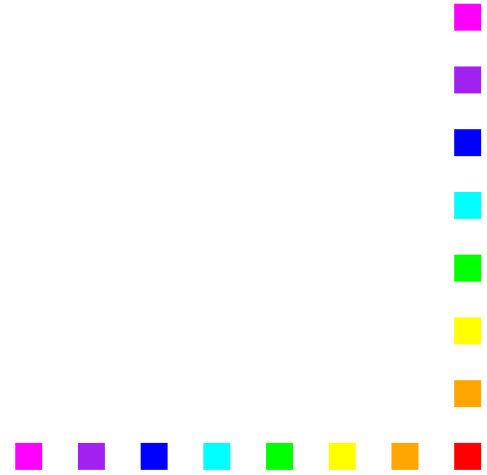
Mais . . .

- détection de collisions possible
- effectuée dans le cas $n = 128$, $k = 50$, $t = 1$



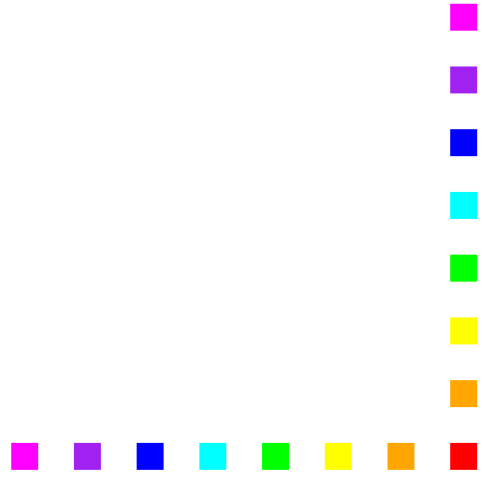
Mais . . .

- détection de collisions possible
- effectuée dans le cas $n = 128$, $k = 50$, $t = 1$
- test gourmand en calcul (ordre de la racine carrée de l'entropie)



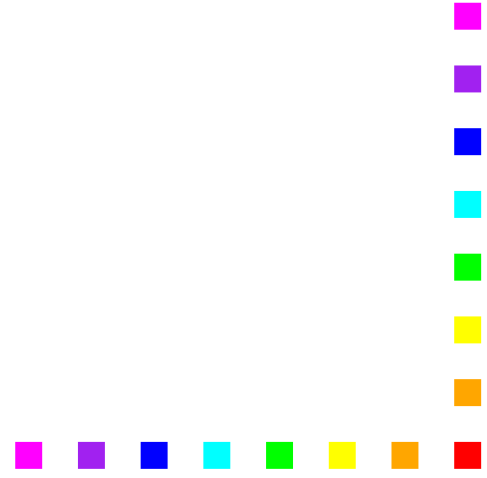
Des trappes dans les clés publiques ?

Le cas de RSA



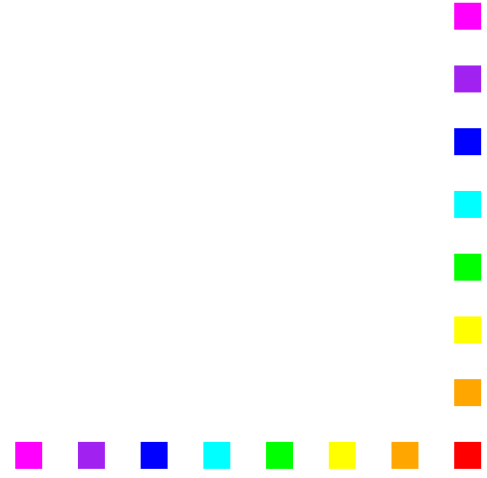
Le système RSA

- proposé par Rivest, Shamir et Adleman en 1978



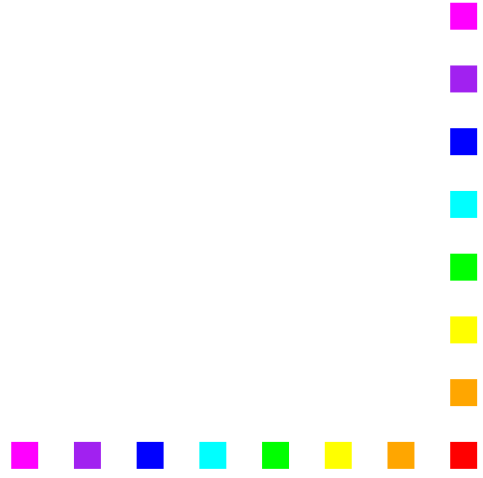
Le système RSA

- proposé par Rivest, Shamir et Adleman en 1978
- très utilisé aujourd'hui



Le système RSA

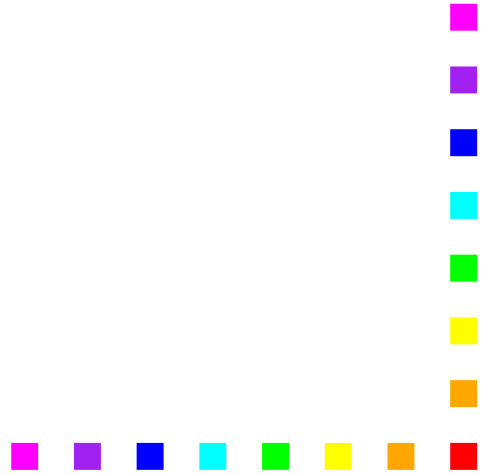
- proposé par Rivest, Shamir et Adleman en 1978
- très utilisé aujourd'hui
- repose sur la difficulté de factoriser les entiers



RSA comment ça marche ?

Clé publique : $(n, e) \in \mathbb{N}^2$ avec

- $n = p \times q$ produit de deux nombres premiers distincts,
- e entier inversible modulo $\varphi(n) = (p-1)(q-1)$

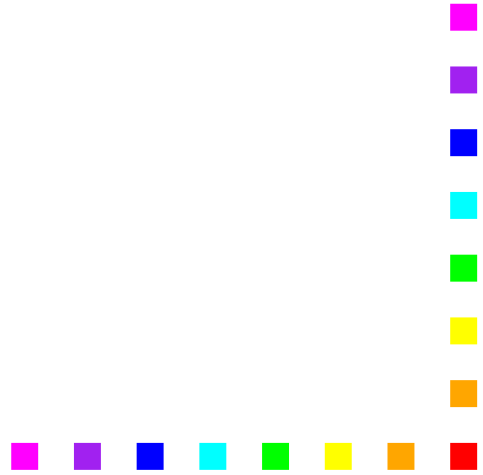


RSA comment ça marche ?

Clé publique : $(n, e) \in \mathbb{N}^2$ avec

- $n = p \times q$ produit de deux nombres premiers distincts,
- e entier inversible modulo $\varphi(n) = (p-1)(q-1)$

Exemple : $n = 17 \times 23 = 391,\ \varphi(n) = 16 \times 22 = 352,\ \ e = 3$



RSA comment ça marche ?

Clé publique : $(n, e) \in \mathbb{N}^2$ avec

- $n = p \times q$ produit de deux nombres premiers distincts,
- e entier inversible modulo $\varphi(n) = (p-1)(q-1)$

Exemple : $n = 17 \times 23 = 391,\ \varphi(n) = 16 \times 22 = 352,\ \ e = 3$

Clé privée : $d = e^{-1} \pmod{\varphi(n)}$

RSA comment ça marche ?

Clé publique : $(n, e) \in \mathbb{N}^2$ avec

- $n = p \times q$ produit de deux nombres premiers distincts,
- e entier inversible modulo $\varphi(n) = (p-1)(q-1)$

Exemple : $n = 17 \times 23 = 391,\ \varphi(n) = 16 \times 22 = 352,\ \ e = 3$

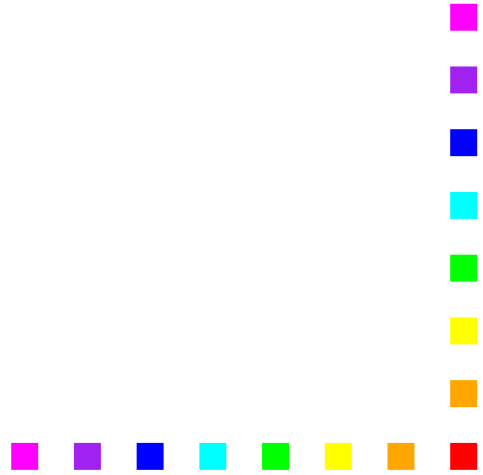
Clé privée : $d = e^{-1} \pmod{\varphi(n)}$

Exemple : $d = 1/3 \pmod{352} = 235$

RSA (2)

Chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$

$$c = m^e \pmod{n}$$

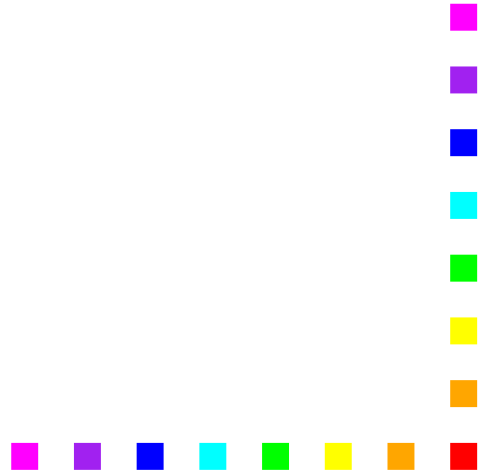


RSA (2)

Chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$

$$c = m^e \pmod{n}$$

Exemple : $m = 13, \quad c = 13^3 \pmod{391} = 242$



RSA (2)

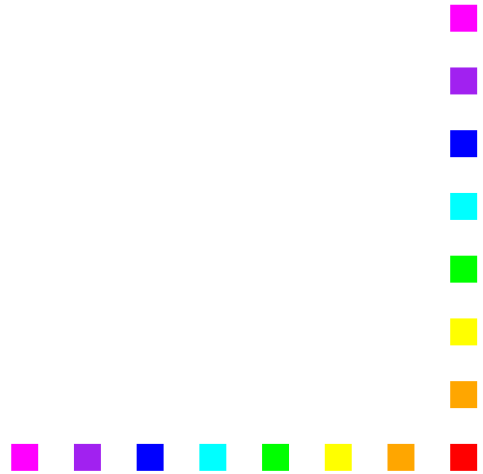
Chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$

$$c = m^e \pmod{n}$$

Exemple : $m = 13, \quad c = 13^3 \pmod{391} = 242$

Déchiffrement d'un message $c \in \mathbb{Z}/n\mathbb{Z}$

$$m = c^d \pmod{n}$$



RSA (2)

Chiffrement d'un message $m \in \mathbb{Z}/n\mathbb{Z}$

$$c = m^e \pmod{n}$$

Exemple : $m = 13, \quad c = 13^3 \pmod{391} = 242$

Déchiffrement d'un message $c \in \mathbb{Z}/n\mathbb{Z}$

$$m = c^d \pmod{n}$$

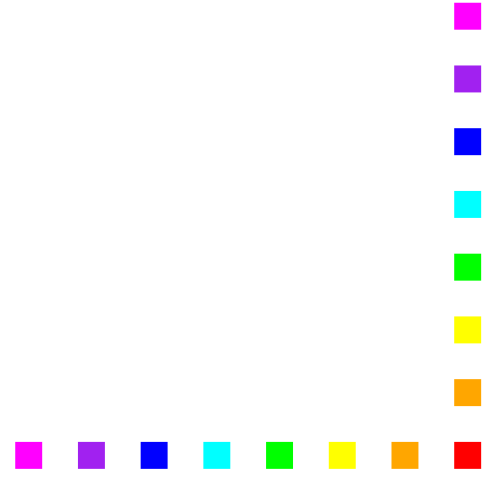
Exemple : $242^{235} \pmod{391} = 13$

Le problème RSAP

- le problème *RSAP* :

  données : une clé publique RSA (n, e) et $c \in \mathbb{N}$

  but : trouver $m \in \mathbb{N}$ tel que $c = m^e \pmod{n}$



Le problème RSAP

- le problème *RSAP* :

  données : une clé publique RSA (n, e) et $c \in \mathbb{N}$

  but : trouver $m \in \mathbb{N}$ tel que $c = m^e \pmod{n}$

- le problème *FACTOR* :

  donnée : un entier composé n

  but : trouver p et q tels que $n = p \times q$

Clairement, $RSAP <_P FACTOR$

Le problème RSAP

- le problème *RSAP* :
  données : une clé publique RSA (n, e) et $c \in \mathbb{N}$
  but : trouver $m \in \mathbb{N}$ tel que $c = m^e \pmod{n}$
- le problème *FACTOR* :
  donnée : un entier composé n
  but : trouver p et q tels que $n = p \times q$

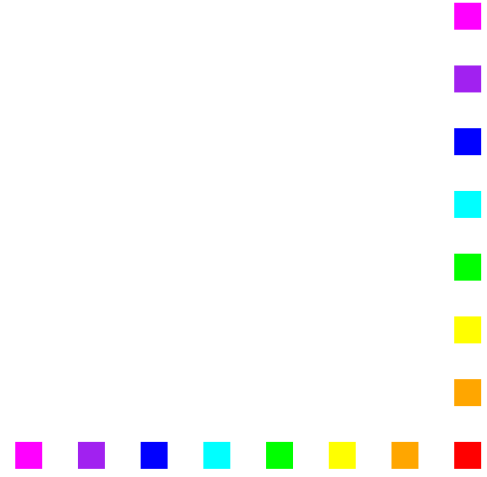
Clairement, $RSAP <_P FACTOR$
réciproque non connue.

Une variante de *RSAP*

Le problème *RSAP'* :

données : une clé publique RSA (n, e)

but : trouver d tel que $d = e^{-1} \pmod{\varphi(n)}$



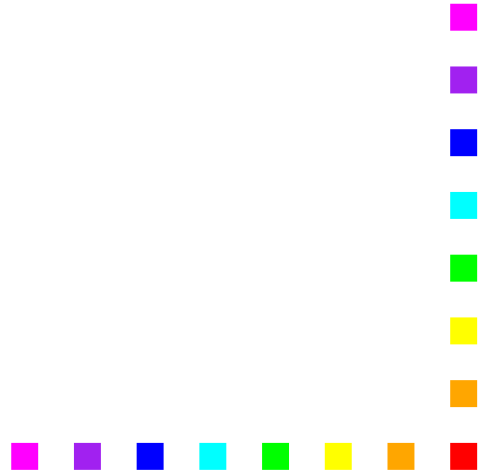
Une variante de *RSAP*

Le problème *RSAP'* :

données : une clé publique RSA (n, e)

but : trouver d tel que $d = e^{-1} \pmod{\varphi(n)}$

Clairement, $RSAP <_P RSAP' <_P FACTOR$



Une variante de *RSAP*

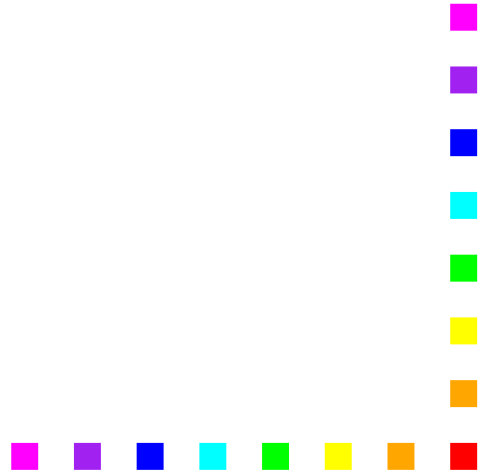
Le problème *RSAP'* :

données : une clé publique RSA (n, e)

but : trouver d tel que $d = e^{-1} \pmod{\varphi(n)}$

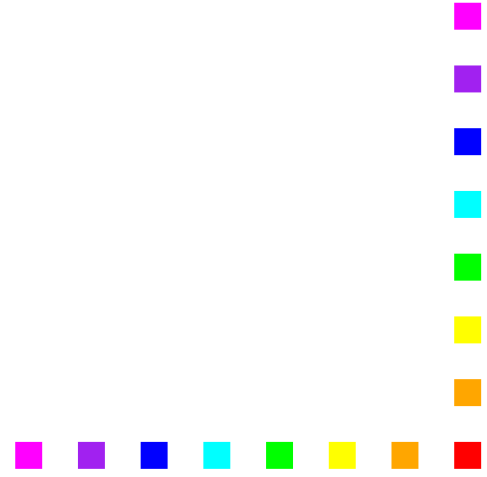
Clairement, $RSAP <_P RSAP' <_P FACTOR$

et les réciproques ?



Un algorithme probabiliste

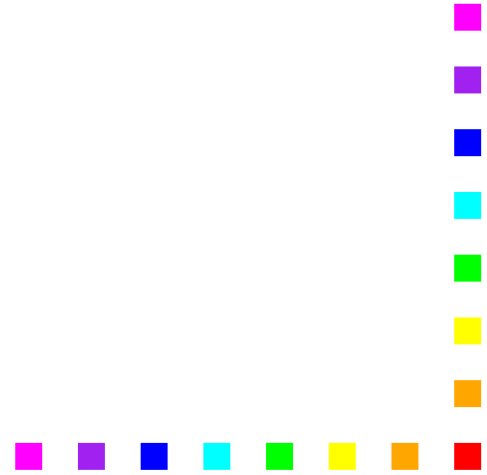
Connaissant n, e et d, on peut factoriser n :



Un algorithme probabiliste

Connaissant n, e et d, on peut factoriser n :

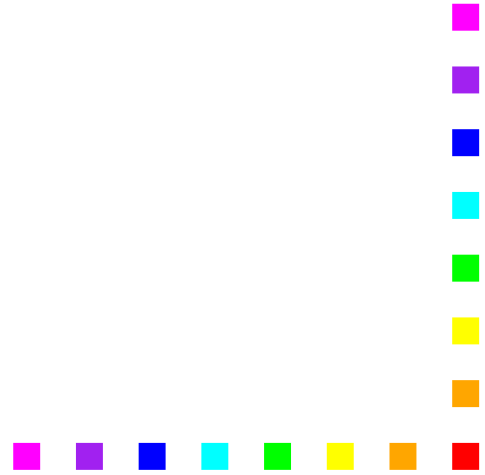
- calculer u impair et s tels que $ed - 1 = u \times 2^s$



Un algorithme probabiliste

Connaissant n, e et d, on peut factoriser n :

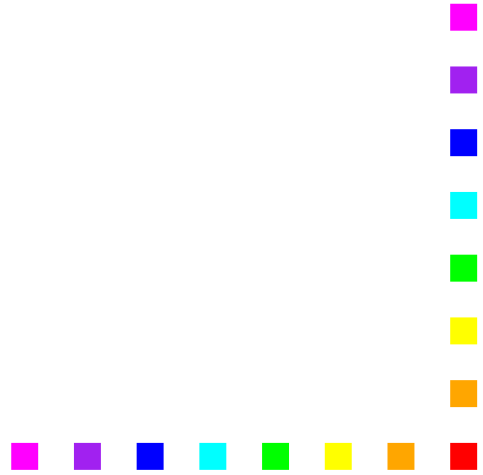
- calculer u impair et s tels que $ed - 1 = u \times 2^s$
- choisir $2 \leq a \leq n - 2$ au hasard, premier avec n



Un algorithme probabiliste

Connaissant n, e et d, on peut factoriser n :

- calculer u impair et s tels que $ed - 1 = u \times 2^s$
- choisir $2 \leq a \leq n - 2$ au hasard, premier avec n
- calculer $b_t = a^{u.2^t} \pmod{n}$ depuis $t = 0$ jusqu'à obtenir $b_t = 1$



Un algorithme probabiliste

Connaissant n, e et d, on peut factoriser n :

- calculer u impair et s tels que $ed - 1 = u \times 2^s$
- choisir $2 \leq a \leq n - 2$ au hasard, premier avec n
- calculer $b_t = a^{u.2^t} \pmod{n}$ depuis $t = 0$ jusqu'à obtenir $b_t = 1$
- calculer le pgcd de n et de $b_{t-1} - 1$

Un algorithme probabiliste

Connaissant n, e et d, on peut factoriser n :

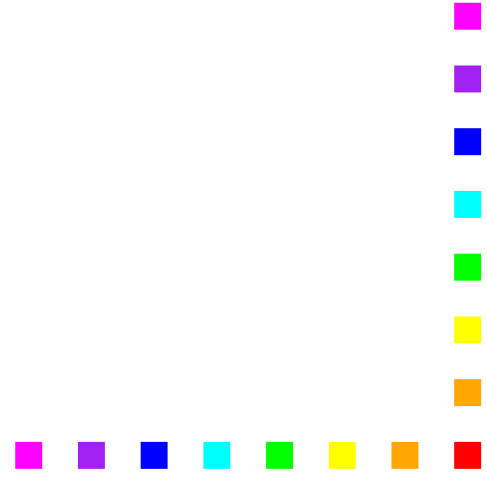
- calculer u impair et s tels que $ed - 1 = u \times 2^s$
- choisir $2 \leq a \leq n - 2$ au hasard, premier avec n
- calculer $b_t = a^{u.2^t} \pmod{n}$ depuis $t = 0$ jusqu'à obtenir $b_t = 1$
- calculer le pgcd de n et de $b_{t-1} - 1$

avec proba 1/2, le pgcd est un facteur premier de n

Pas trop petit d !

- Un danger : choisir la clé privée d petite

  Le problème *RSAP'* est polynomial si $d < n^{1/4}$ (M. Wiener, 1990)



Pas trop petit d !

- Un danger : choisir la clé privée d petite

  Le problème *RSAP'* est polynomial si $d < n^{1/4}$ (M. Wiener, 1990)

- repose sur le fait qu'il existe alors un entier k tel que

$$|\frac{e}{n} - \frac{k}{d}| < \frac{1}{d^2}$$

Pas trop petit d !

- Un danger : choisir la clé privée d petite

  Le problème *RSAP'* est polynomial si $d < n^{1/4}$ (M. Wiener, 1990)

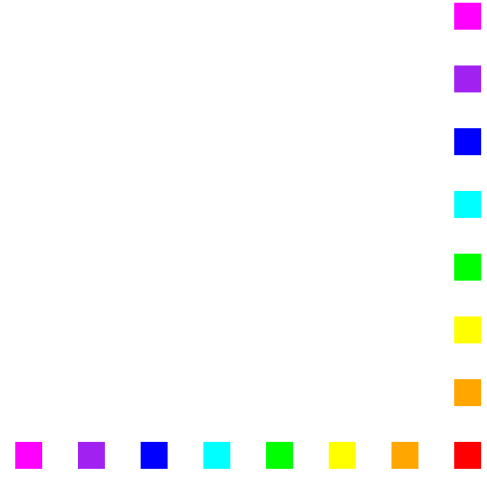
- repose sur le fait qu'il existe alors un entier k tel que

$$|\frac{e}{n} - \frac{k}{d}| < \frac{1}{d^2}$$

- obtenu par développement en fraction continue de $\frac{e}{n}$

Exemple

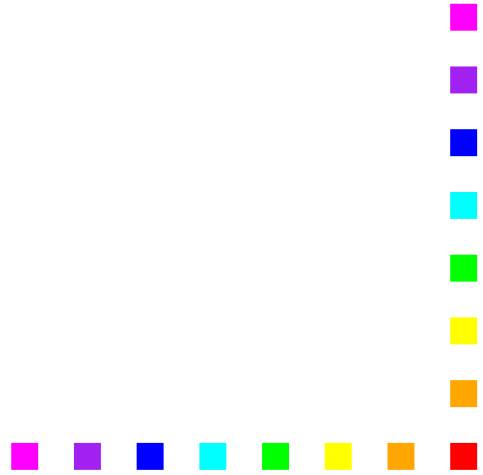
- $n = 85067$, $e = 12997$



Exemple

- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$



Exemple

- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$

- chiffrement d'un message $c = 100^e \pmod{n} = 11607$

Exemple

- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$

- chiffrement d'un message $c = 100^e \pmod{n} = 11607$
- essais de déchiffrement

Exemple

- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$

- chiffrement d'un message $c = 100^e \pmod{n} = 11607$
- essais de déchiffrement
- $c^6 \pmod{n} = 83569 \neq 100$

Exemple

- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$

- chiffrement d'un message $c = 100^e \pmod{n} = 11607$
- essais de déchiffrement
- $c^6 \pmod{n} = 83569 \neq 100$
- $c^7 \pmod{n} = 51449 \neq 100$

Exemple

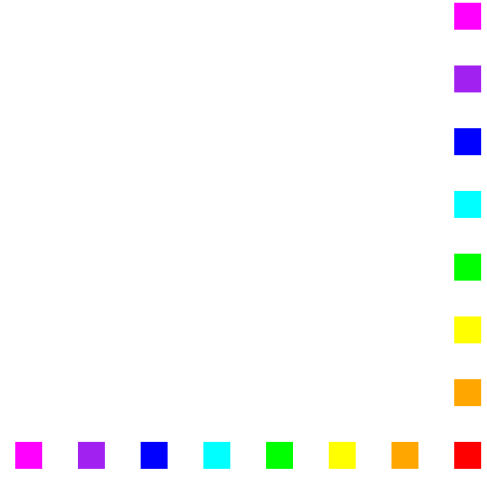
- $n = 85067$, $e = 12997$
- développement en fraction continue

$$\frac{1}{6}, \frac{1}{7}, \frac{2}{13}, \frac{11}{72}, \frac{266}{1741}, \ldots$$

- chiffrement d'un message $c = 100^e \pmod{n} = 11607$
- essais de déchiffrement
- $c^6 \pmod{n} = 83569 \neq 100$
- $c^7 \pmod{n} = 51449 \neq 100$
- $c^{13} \pmod{n} = 100$ GAGNE !

Un générateur de clés RSA à trappes

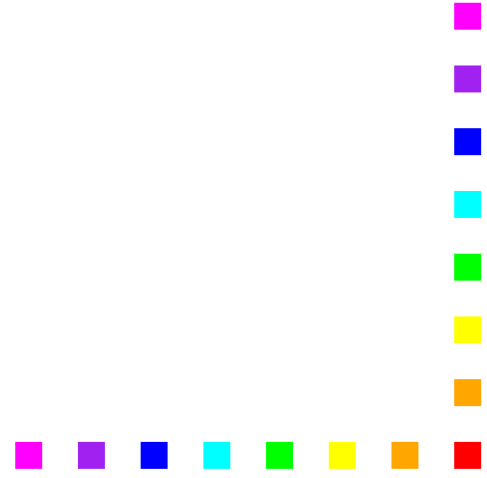
dû à Claude Crépeau et Alain Slakmon en utilisant l'attaque de Wiener (`http://crypto.cs.mcgill.ca/~crepeau/RSA`)



Un générateur de clés RSA à trappes

dû à Claude Crépeau et Alain Slakmon en utilisant l'attaque de Wiener (`http://crypto.cs.mcgill.ca/~crepeau/RSA`)

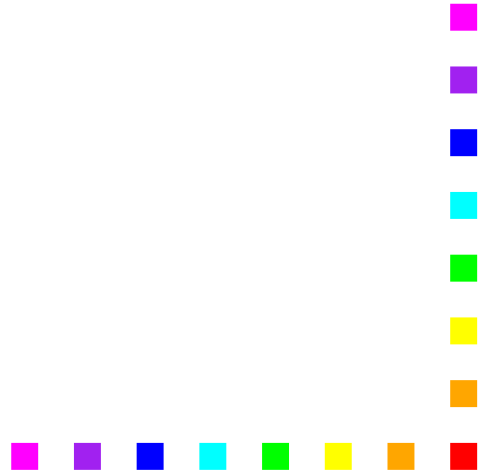
- un grand entier pair M est fixé (et caché : c'est la trappe)



Un générateur de clés RSA à trappes

dû à Claude Crépeau et Alain Slakmon en utilisant l'attaque de Wiener (`http://crypto.cs.mcgill.ca/~crepeau/RSA`)

- un grand entier pair M est fixé (et caché : c'est la trappe)
- générer une paire de clés RSA $(n, e1)$, $d1$ avec $d1$ petit jusqu'à $e1 + M$ premier avec $\varphi(n)$



Un générateur de clés RSA à trappes

dû à Claude Crépeau et Alain Slakmon en utilisant l'attaque de Wiener (`http://crypto.cs.mcgill.ca/~crepeau/RSA`)

- un grand entier pair M est fixé (et caché : c'est la trappe)
- générer une paire de clés RSA $(n, e1)$, $d1$ avec $d1$ petit jusqu'à $e1 + M$ premier avec $\varphi(n)$
- retourner n, $e = e1 + M$ et $d = e^{-1} \pmod{\varphi(n)}$

Un générateur de clés RSA à trappes

dû à Claude Crépeau et Alain Slakmon en utilisant l'attaque de Wiener (`http://crypto.cs.mcgill.ca/~crepeau/RSA`)

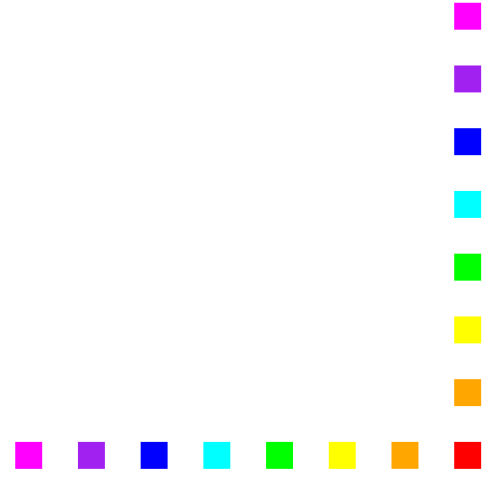
- un grand entier pair M est fixé (et caché : c'est la trappe)
- générer une paire de clés RSA $(n, e1)$, $d1$ avec $d1$ petit jusqu'à $e1 + M$ premier avec $\varphi(n)$
- retourner n, $e = e1 + M$ et $d = e^{-1} \pmod{\varphi(n)}$

n, e et d forment une paire de clés avec d de l'ordre de n

...

Utilisation de la trappe

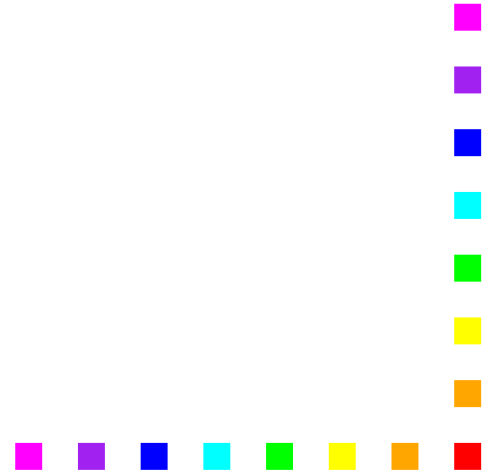
. . . mais la trappe permet de réoudre le problème *RSAP'*



Utilisation de la trappe

. . . mais la trappe permet de réoudre le problème *RSAP'*

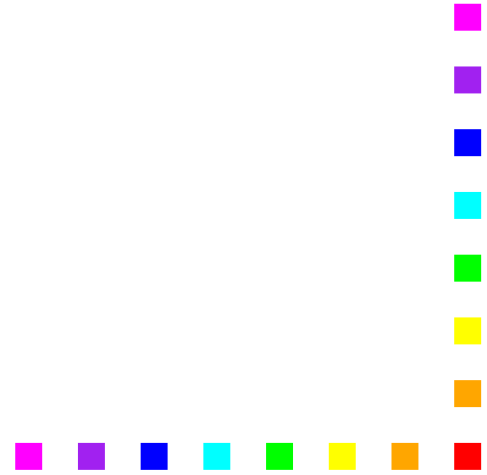
- calculer $e1 = e - M$



Utilisation de la trappe

. . . mais la trappe permet de réoudre le problème *RSAP'*

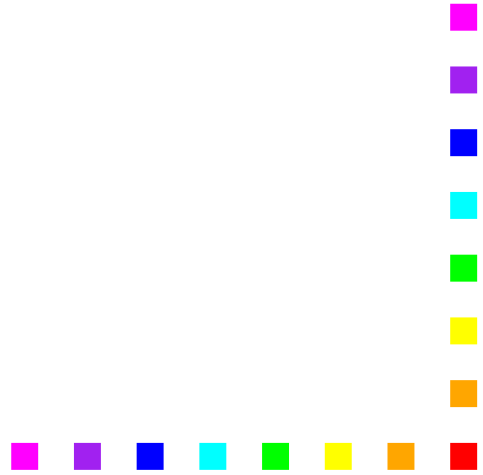
- calculer $e1 = e - M$
- retrouver $d1$ par la méthode de Wiener



Utilisation de la trappe

. . . mais la trappe permet de réoudre le problème *RSAP'*

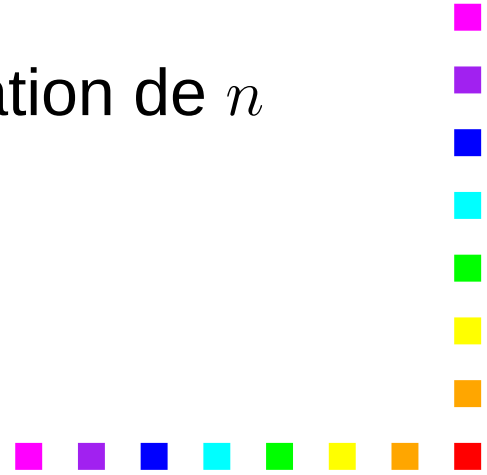
- calculer $e1 = e - M$
- retrouver $d1$ par la méthode de Wiener
- factoriser n à l'aide de $e1$ et $d1$



Utilisation de la trappe

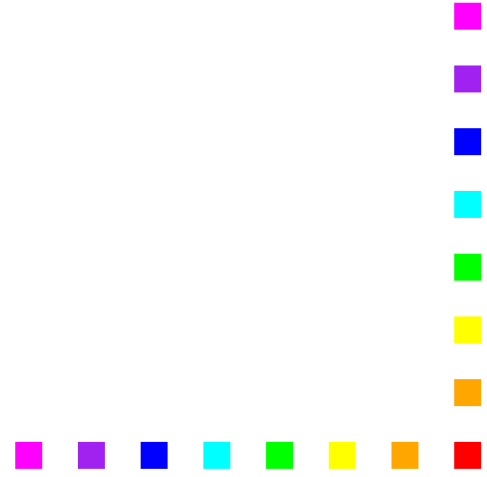
. . . mais la trappe permet de réoudre le problème *RSAP'*

- calculer $e1 = e - M$
- retrouver $d1$ par la méthode de Wiener
- factoriser n à l'aide de $e1$ et $d1$
- calculer d en connaissant e et la factorisation de n



Une parade

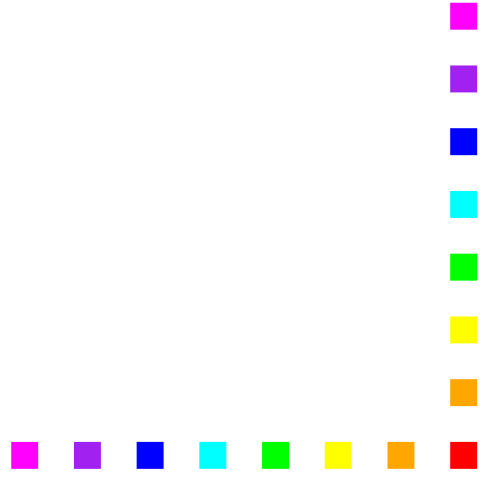
proposée par Serge Vaudenay.



Une parade

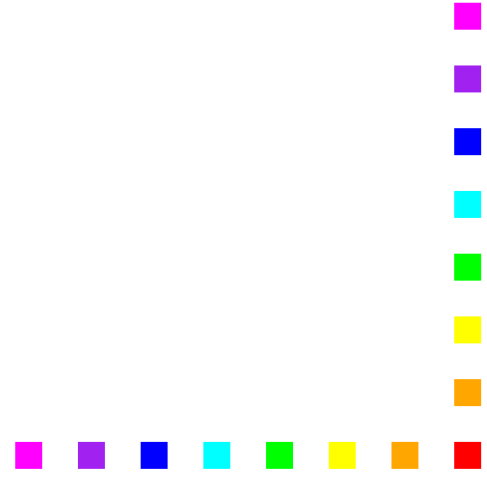
proposée par Serge Vaudenay.

repose sur des réductions de e modulo un petit facteur premier de $\varphi(n)$



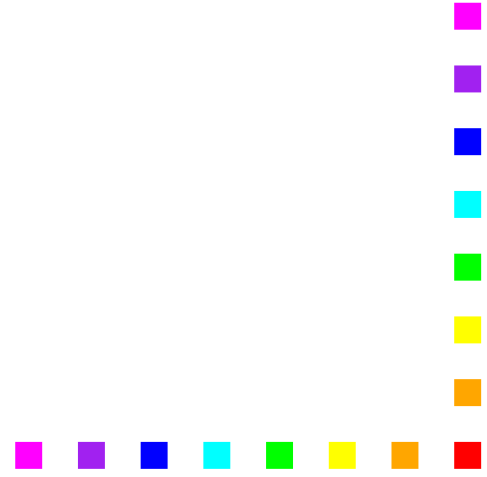
Questions en suspens

- des trappes pour quoi faire ?



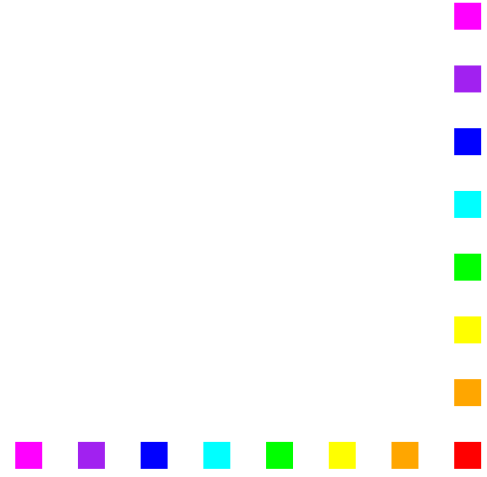
Questions en suspens

- des trappes pour quoi faire ?
- pour qui ?



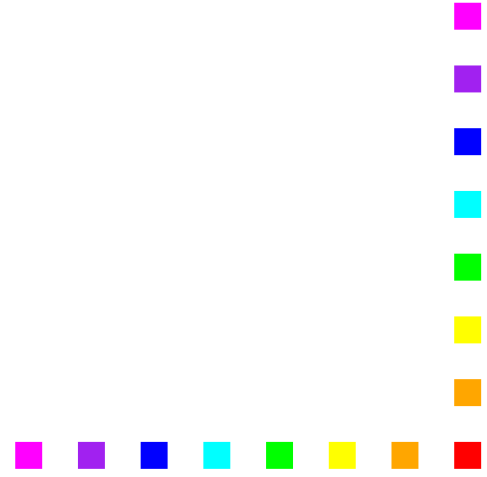
Questions en suspens

- des trappes pour quoi faire ?
- pour qui ?
- existence de trappes dans les outils d’aujourd’hui ?



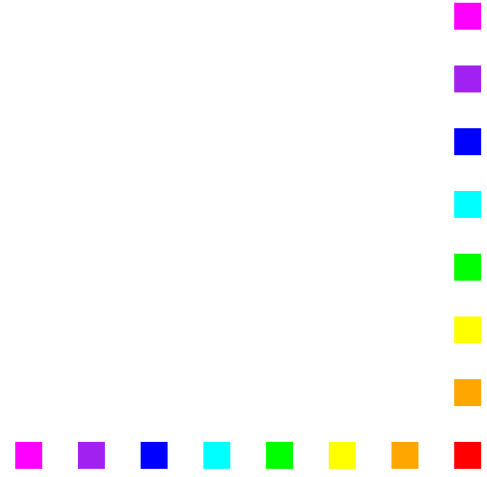
Questions en suspens

- des trappes pour quoi faire ?
- pour qui ?
- existence de trappes dans les outils d’aujourd’hui ?
- quels types de trappes ?



Questions en suspens

- des trappes pour quoi faire ?
- pour qui ?
- existence de trappes dans les outils d'aujourd'hui ?
- quels types de trappes ?
- comment les détecter ?



Merci

Questions ?