Peut-on faire confiance à la cryptographie moderne ?

Léo Perrin
@lpp_crypto

Mardi 7 avril 2020
Parlons Maths !

La cryptographie est **partout**...
mais peut-on lui faire confiance ?

Outline

Plan de la section à venir

Dans l'ancien temps

T E X T E
U

Chiffrement de César (avec décalage de 1) : on opère *a priori* sur des lettres.

Dans l'ancien temps

T E X T E

U F

Chiffrement de César (avec décalage de 1) : on opère *a priori* sur des lettres.

Dans l'ancien temps

T E X T E

U F Y

Chiffrement de César (avec décalage de 1) : on opère *a priori* sur des lettres.

Dans l'ancien temps

T E X T E

U F Y U F

Chiffrement de César (avec décalage de 1) : on opère *a priori* sur des lettres.

Des chiffres et des lettres

En fait, on n'opère pas vraiment sur les lettres !

En décomposant les étapes :

Des chiffres et des lettres

En fait, on n'opère pas vraiment sur les lettres !

En décomposant les étapes :

1. T est la 20ème lettre de l'alphabet

Des chiffres et des lettres

En fait, on n'opère pas vraiment sur les lettres !

En décomposant les étapes :

1. T est la 20ème lettre de l'alphabet
2. Décalage de $1 : 20 + 1 = 21$

Des chiffres et des lettres

En fait, on n'opère pas vraiment sur les lettres !

En décomposant les étapes :

1. T est la 20ème lettre de l'alphabet
2. Décalage de 1 : $20 + 1 = 21$
3. On remplace T par la 21ème lettre : U

Des chiffres et des lettres

En fait, on n'opère pas vraiment sur les lettres !

En décomposant les étapes :

1. T est la 20ème lettre de l'alphabet
2. Décalage de 1 : $20 + 1 = 21$
3. On remplace T par la 21ème lettre : U

Principe général

1. on transforme les lettres en nombres,
2. on fait des opérations sur ces nombres,
3. puis on re-transforme ces nombres en lettres.

Numérisation

On peut tout transformer en nombres !

Texte : Chaque lettre, chaque ponctuation, chaque smiley a un code unique

Symbole	a	A	!	-
Nombre	97	65	33	45

Numérisation

On peut tout transformer en nombres !

Texte : Chaque lettre, chaque ponctuation, chaque smiley a un code unique

Symbole	a	A	!	-
Nombre	97	65	33	45

Image : Chaque pixel devient 3 nombres, par exemple

$$(100, 30, 200)$$

où le premier nombre donne la quantité de rouge, le deuxième de vert et le dernier de bleu.

Son : C'est plus compliqué mais ça marche aussi

Opérations sur les nombres

Un ordinateur ne sait faire qu'une seule chose : calculer.

Opérations sur les nombres

Un ordinateur ne sait faire qu'une seule chose : calculer.

Ça tombe bien : ça nous suffit !

Opérations sur les nombres

Un ordinateur ne sait faire qu'une seule chose : calculer.

Ça tombe bien : ça nous suffit !

Pour chiffrer n'importe quelle données, il suffit de pouvoir chiffrer des séquences de nombres !

Retour sur César

Comment décrire formellement le chiffrement de César ?

- On suppose que le texte a été numérisé, donc on opère sur une séquence de nombres entre 0 et 25.
- La clef aussi est un nombre k entre 0 et 25.

Retour sur César

Comment décrire formellement le chiffrement de César ?

- On suppose que le texte a été numérisé, donc on opère sur une séquence de nombres entre 0 et 25.
- La clef aussi est un nombre k entre 0 et 25.

```
Pour chaque nombre x dans la sequence:
    remplacer x par x+k
    si x vaut 26 ou plus, on le remplace par x-26
```

Définition

Timbre soviétique à l'effigie d'Al-Khwarizmi
source : wikipedia

Un algorithme est une séquence non ambigüe d'opérations permettant de résoudre un problème.

Définition

Timbre soviétique à l'effigie d'Al-Khwarizmi
source : wikipedia

Un algorithme est une séquence non ambigüe d'opérations permettant de résoudre un problème.

Un algorithme de chiffrement est donc une séquence non ambigüe d'opérations permettant de chiffrer des données.

Définition

Timbre soviétique à l'effigie d'Al-Khwarizmi
source : wikipedia

Un algorithme est une séquence non ambigüe d'opérations permettant de résoudre un problème.

Un algorithme de chiffrement est donc une séquence non ambigüe d'opérations permettant de chiffrer une séquence de nombres.

Fonction

En informatique, une fonction est un algorithme qui a des entrées et sorties bien définies.

Fonction

En informatique, une fonction est un algorithme qui a des entrées et sorties bien définies.

Exemple : Chiffre-César

On peut définir `chiffre-césar(s, k)`, où

- `s` est une séquence de nombres,
- `k` est un nombre (la clef).

Fonction

En informatique, une fonction est un algorithme qui a des entrées et sorties bien définies.

Exemple : Chiffre-César

On peut définir `chiffre-césar(s, k)`, où

- `s` est une séquence de nombres,
- `k` est un nombre (la clef).

Pas besoin de recopier l'algorithme à chaque fois qu'on veut l'utiliser : écrire "`chiffre-césar`" suffit !

Fonction

En informatique, une fonction est un algorithme qui a des entrées et sorties bien définies.

Exemple : Chiffre-César

On peut définir `chiffre-césar(s, k)`, où

- `s` est une séquence de nombres,
- `k` est un nombre (la clef).

Pas besoin de recopier l'algorithme à chaque fois qu'on veut l'utiliser : écrire "`chiffre-césar`" suffit !

Librairie

Un ensemble de fonctions informatique forme une "librairie". En pratique, le chiffrement d'une grande partie de nos données est accompli par des fonctions trouvées dans quelques librairies.

"Appliquer une fonction"

Les fonctions sont des raccourcis pour raccourcir la description d'un algorithme.

Exemple d'actualité : "faites une pâte brisée"

Un exemple pas du tout aléatoire : SPARX

```
void sparx_encrypt(uint16_t * x, uint16_t k[][2*R_S]) {
    unsigned int s, r, b;
    for (s=0 ; s<N_S ; s++) {
        for (b=0 ; b<N_B ; b++)
            for (r=0 ; r<R_S ; r++) {
                x[2*b  ] ^= k[N_B*s + b][2*r    ];
                x[2*b+1] ^= k[N_B*s + b][2*r + 1];
                A(x[2*b], x[2*b+1]);
            }
        L(x);
    }
    for (b=0 ; b<N_B ; b++) {
        x[2*b  ] ^= k[N_B*N_S][2*b  ];
        x[2*b+1] ^= k[N_B*N_S][2*b+1];
    }
}
```

"Texte" : 4 nombres (entre 0 et 65535),

Clef : 8 nombres (entre 0 et 65535).

Cet algorithme est beaucoup plus sophistiqué que celui de César... mais fondamentalement, il fait le même type de choses !

Plan de la section à venir

L'importance du choix

- Pour pouvoir chiffrer/déchiffrer, il faut que les interlocuteurs utilisent les mêmes algorithmes…
- et il faut qu'ils soient sécurisés !

L'importance du choix

- Pour pouvoir chiffrer/déchiffrer, il faut que les interlocuteurs utilisent les mêmes algorithmes…
- et il faut qu'ils soient sécurisés !

`Chiffrement1(message, clef)`

`Déchiffrement2(message, clef)`

[ÉCHEC]

L'importance du choix

- Pour pouvoir chiffrer/déchiffrer, il faut que les interlocuteurs utilisent les mêmes algorithmes…
- et il faut qu'ils soient sécurisés !

`Chiffrement1(message, clef)`

`Déchiffrement1(message, clef)`

[SUCCÈS]

Fonctionnement de la recherche en cryptographie (1/2)

Conception

- Quel type d'algorithme faire ?
- Comment éviter les attaques connues ?
- Y'a-t-il d'autres types d'attaques ?

Fonctionnement de la recherche en cryptographie (1/2)

Conception

- Quel type d'algorithme faire ?
- Comment éviter les attaques connues ?
- Y'a-t-il d'autres types d'attaques ?

Cryptanalyse

Fonctionnement de la recherche en cryptographie (1/2)

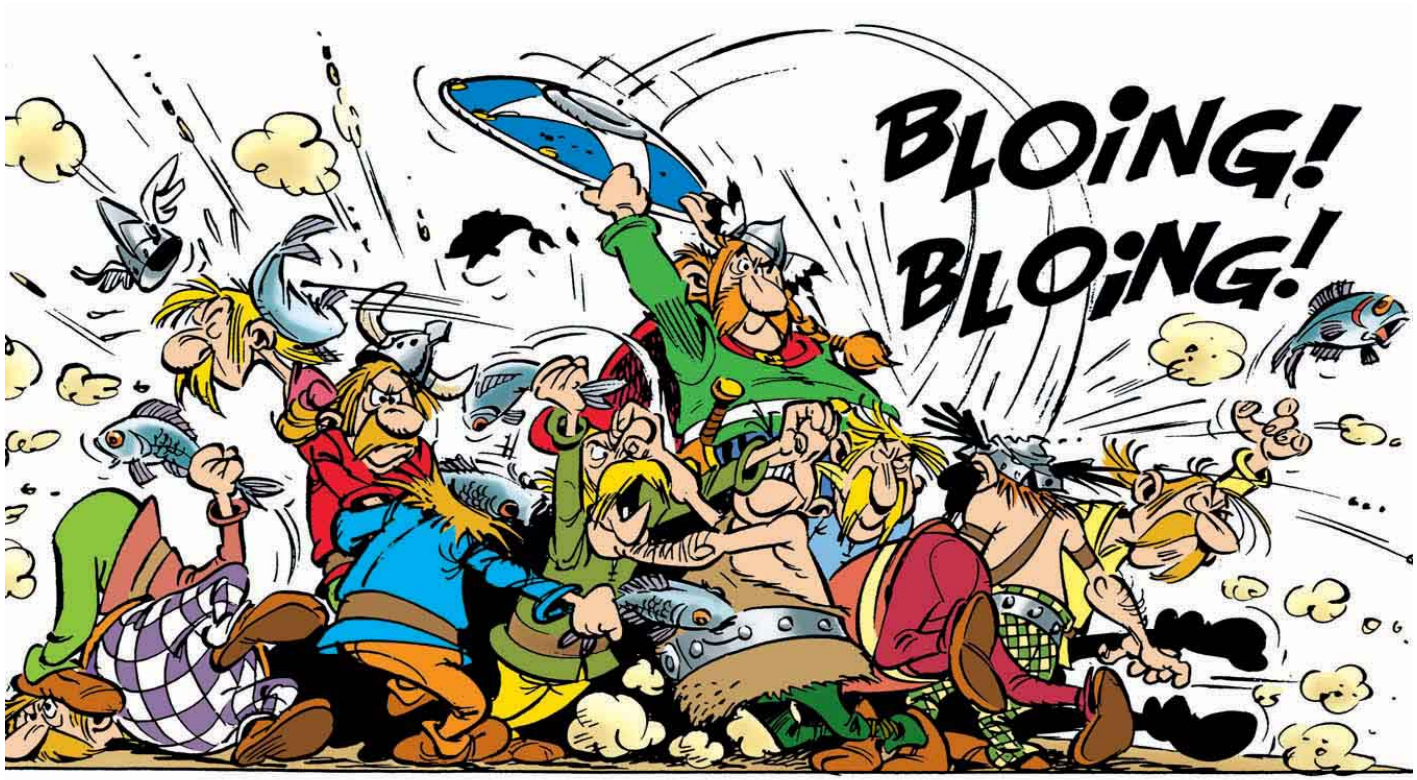
Conception

- Quel type d'algorithme faire ?
- Comment éviter les attaques connues ?
- Y'a-t-il d'autres types d'attaques ?

Cryptanalyse

À MORT LES ALGOS !!1 !

Fonctionnement de la recherche en cryptographie (2/2)

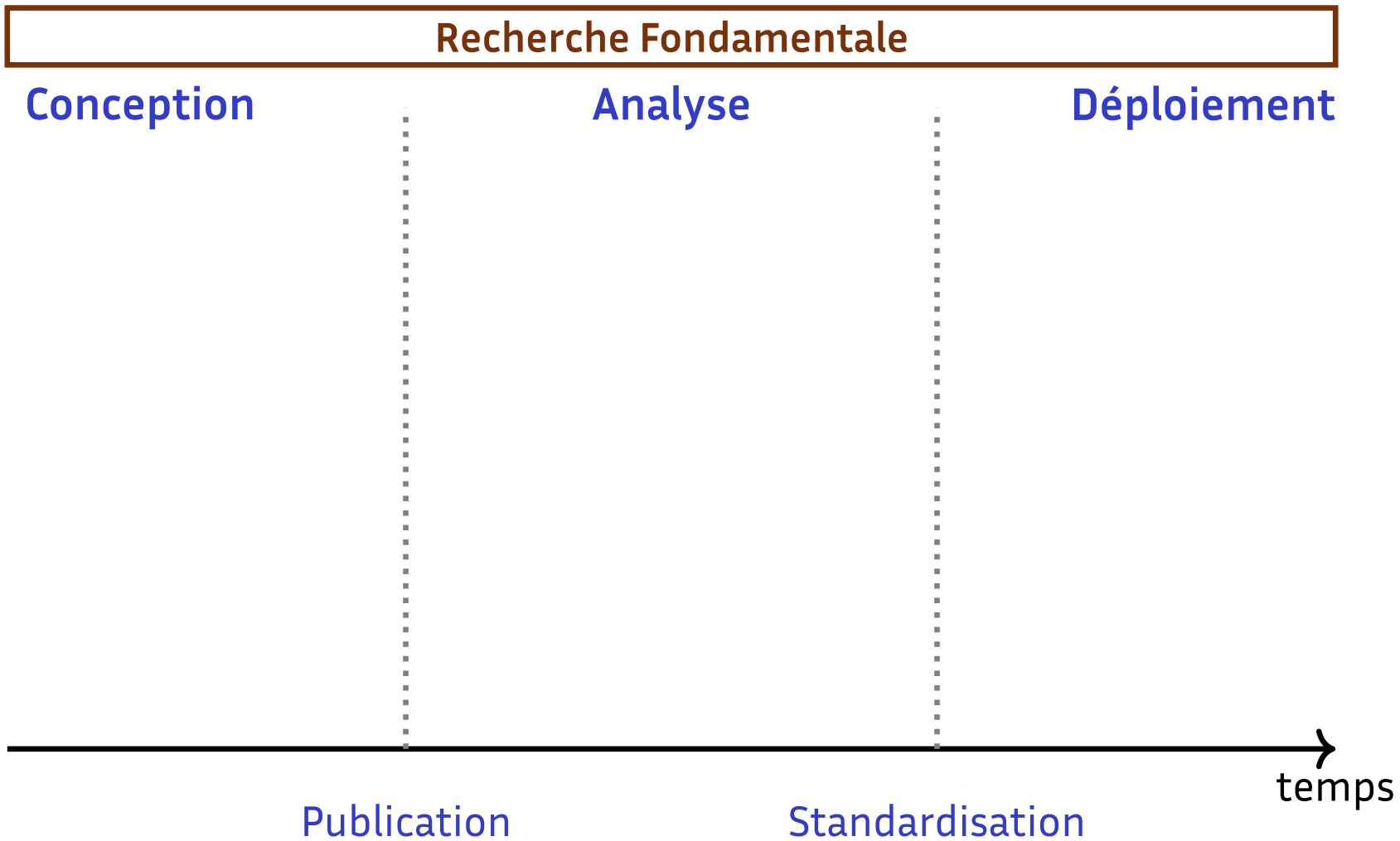


La recherche en cryptographie, allégorie

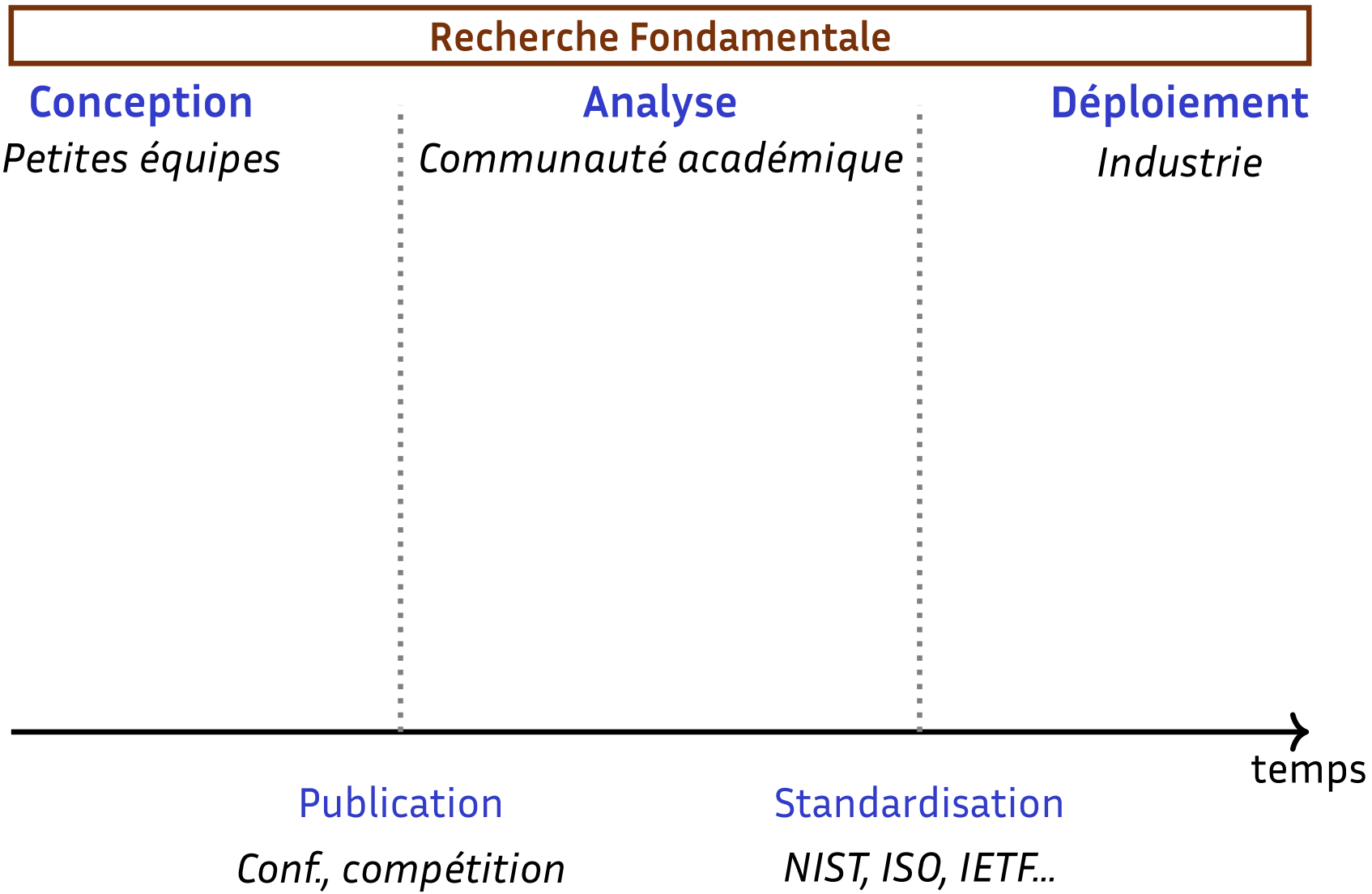
Cycle de vie d'un algorithme cryptographique

Recherche Fondamentale

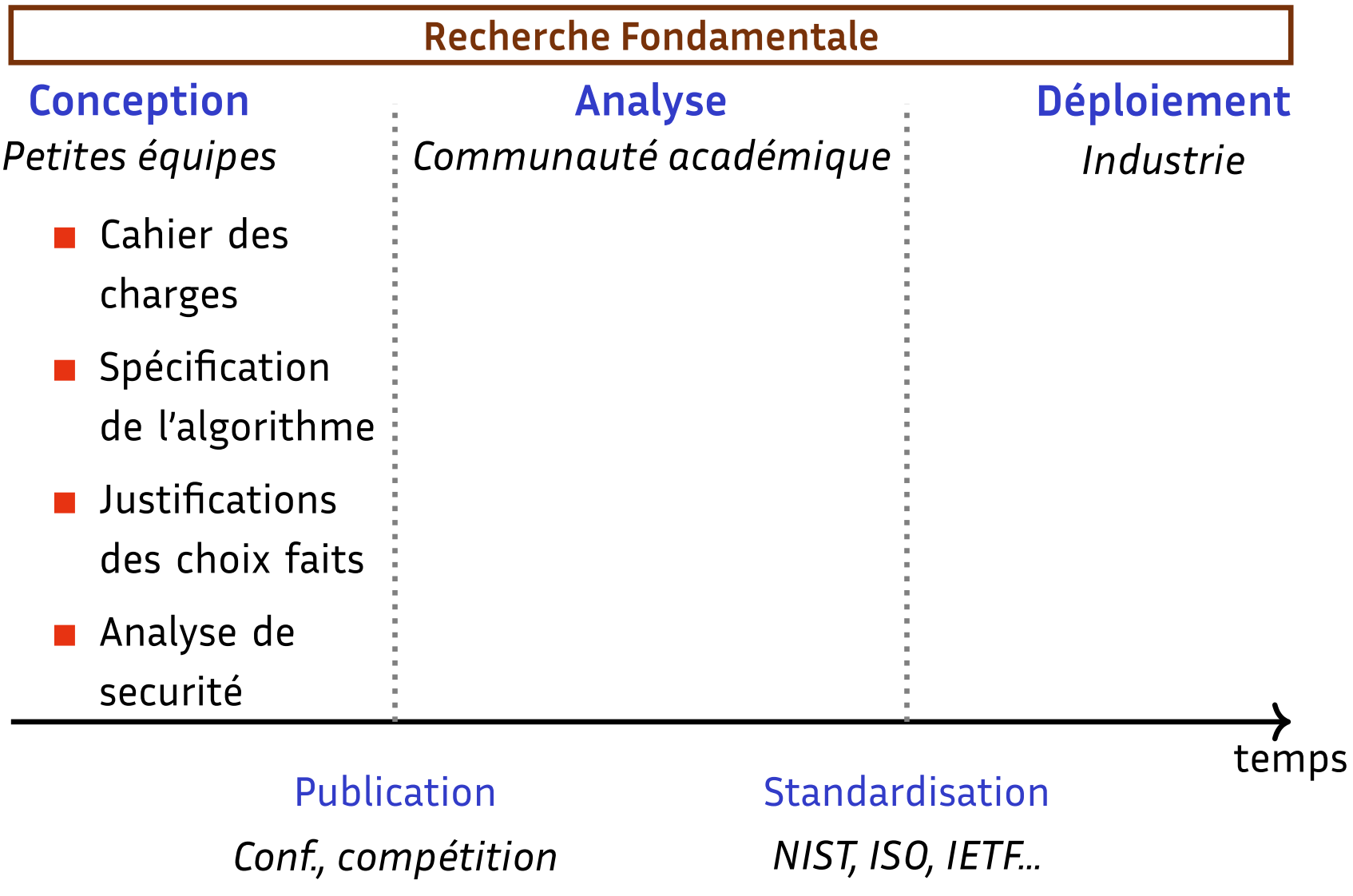
Cycle de vie d'un algorithme cryptographique



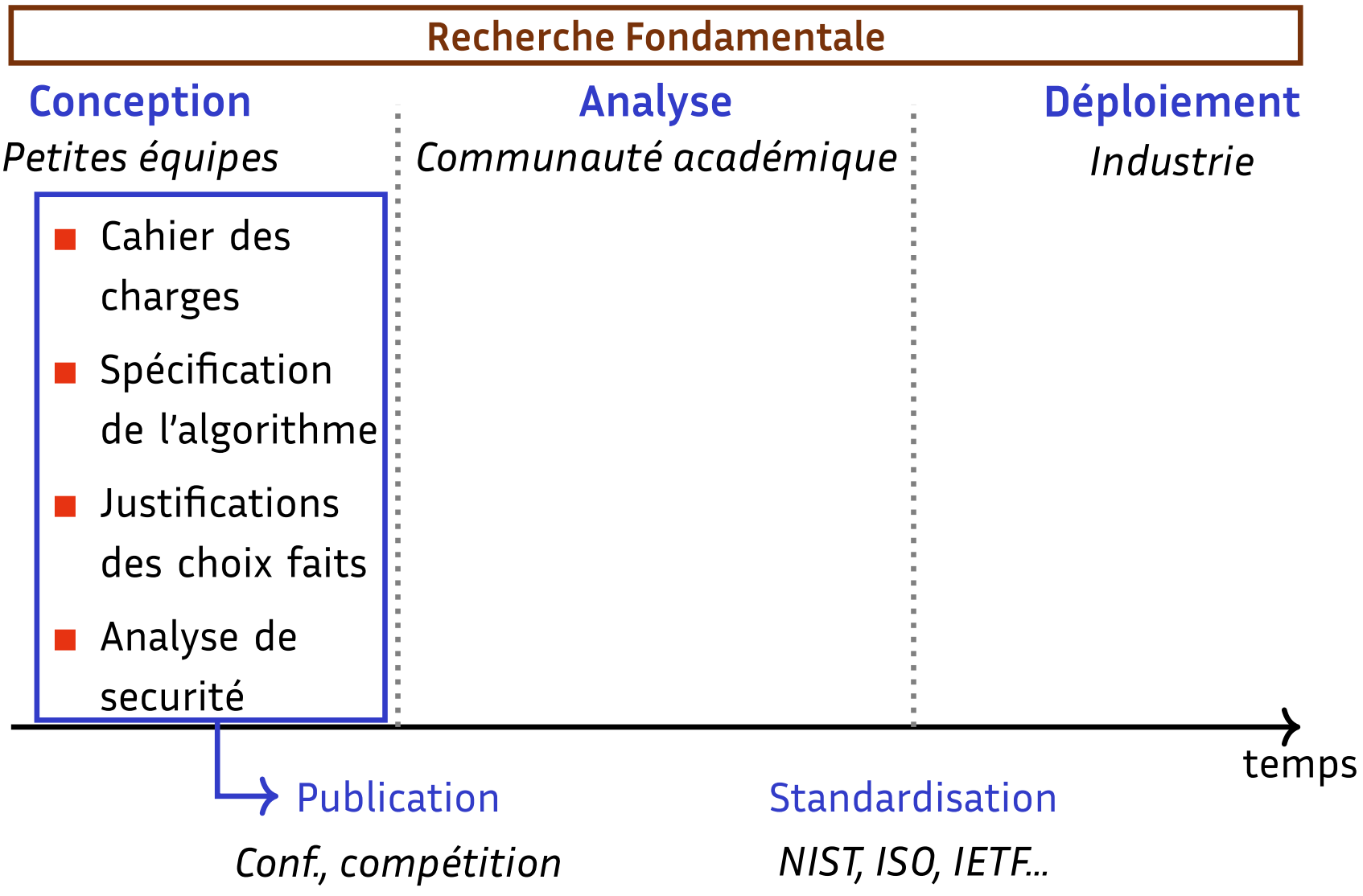
Cycle de vie d'un algorithme cryptographique



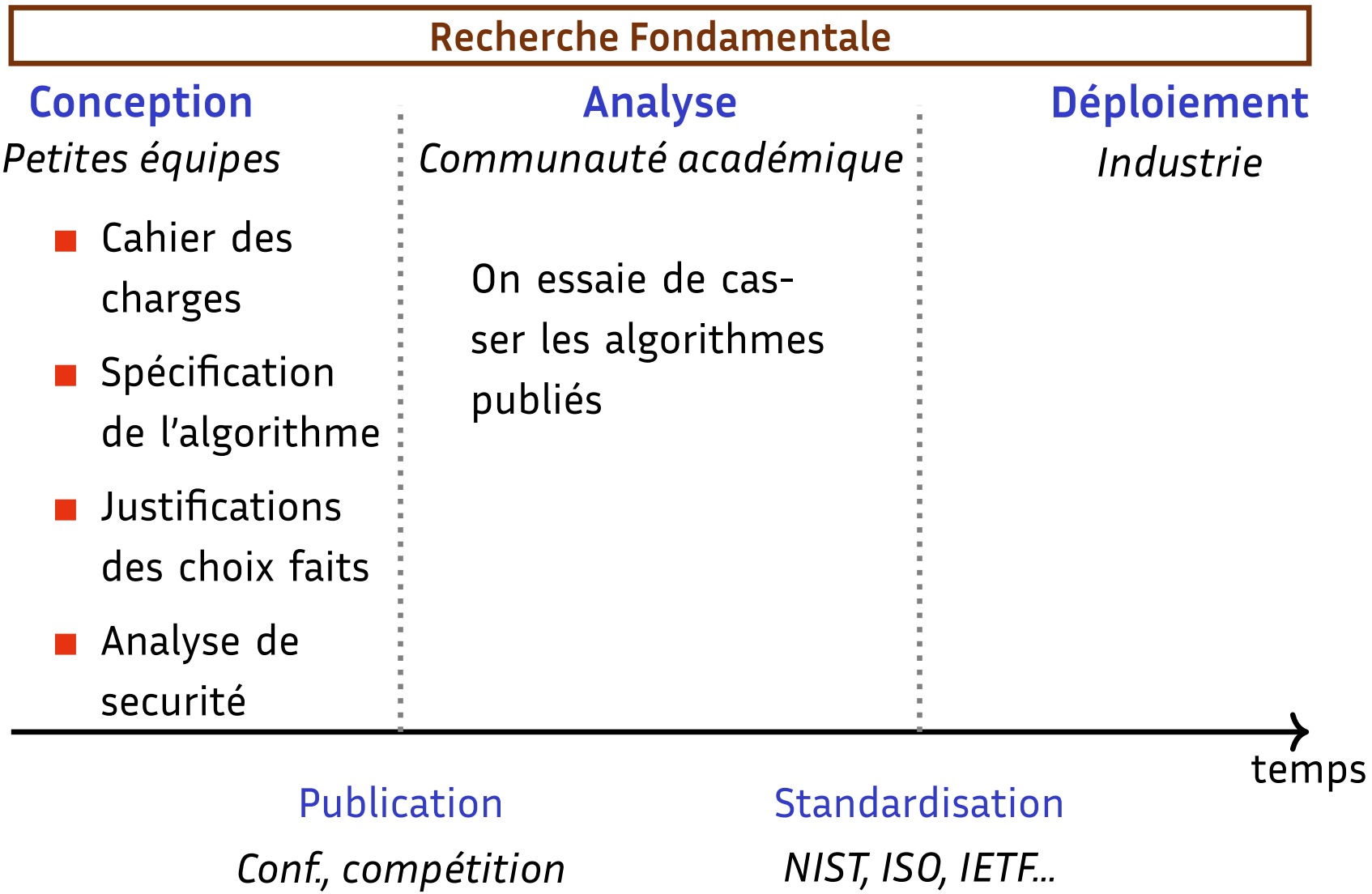
Cycle de vie d'un algorithme cryptographique



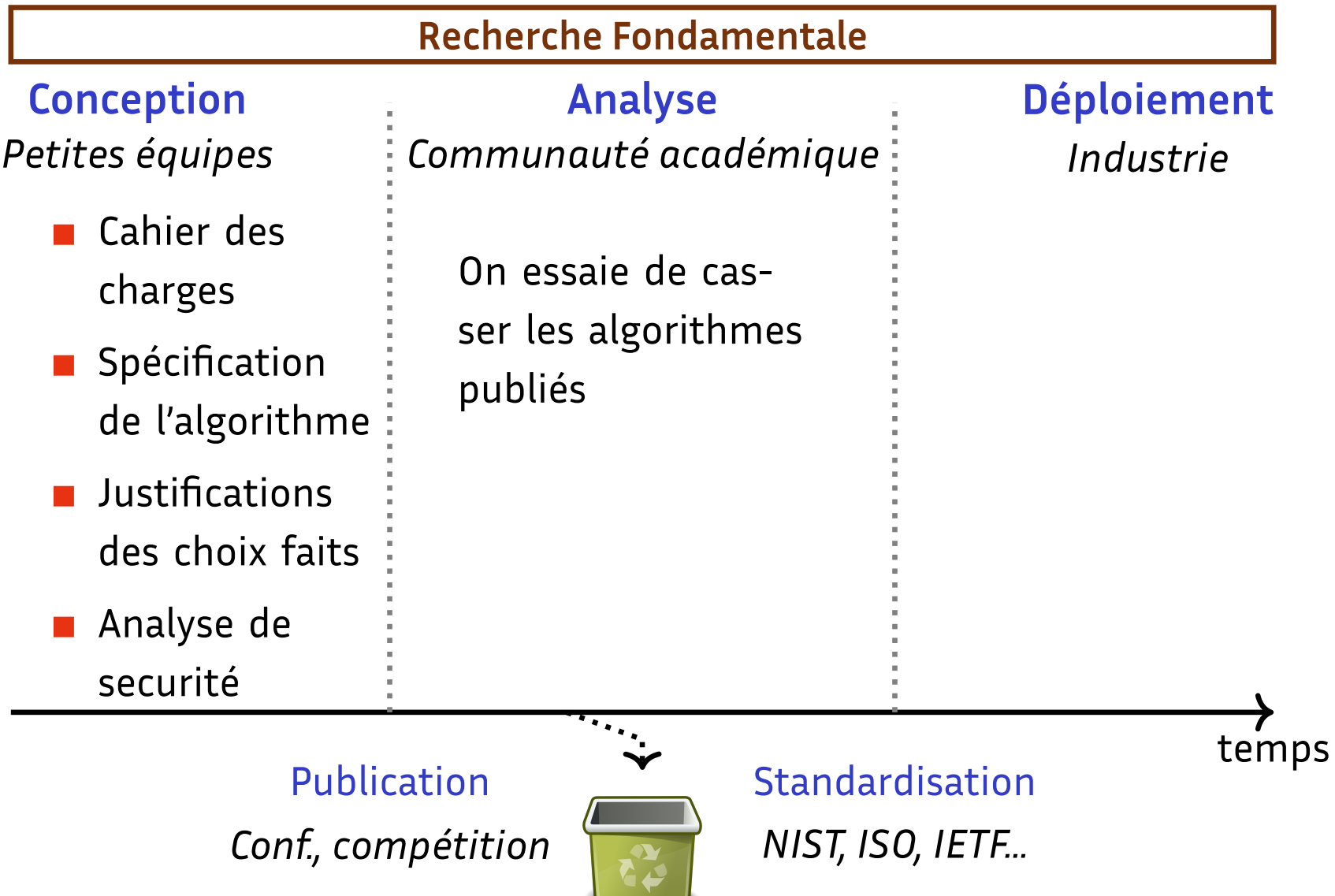
Cycle de vie d'un algorithme cryptographique



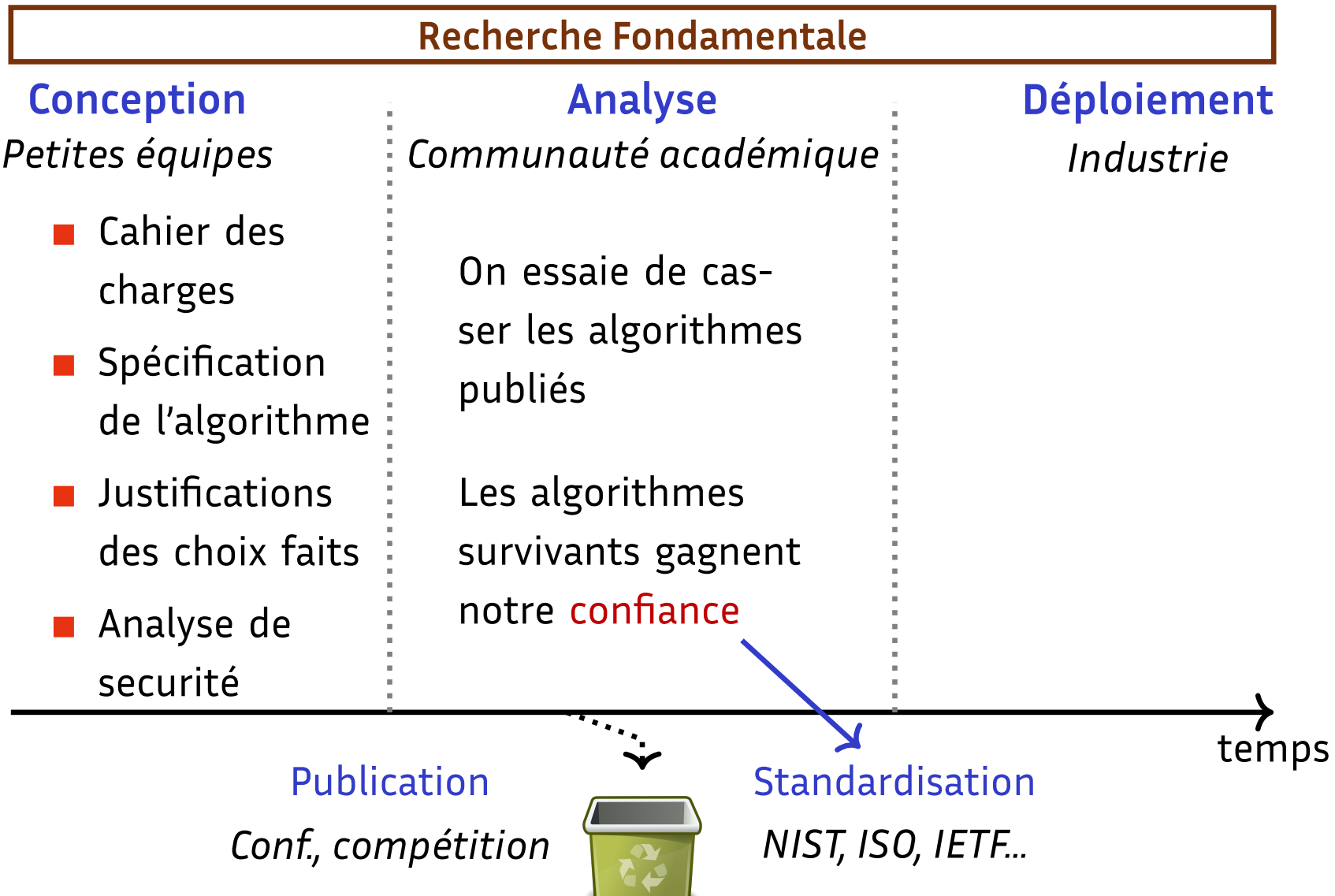
Cycle de vie d'un algorithme cryptographique



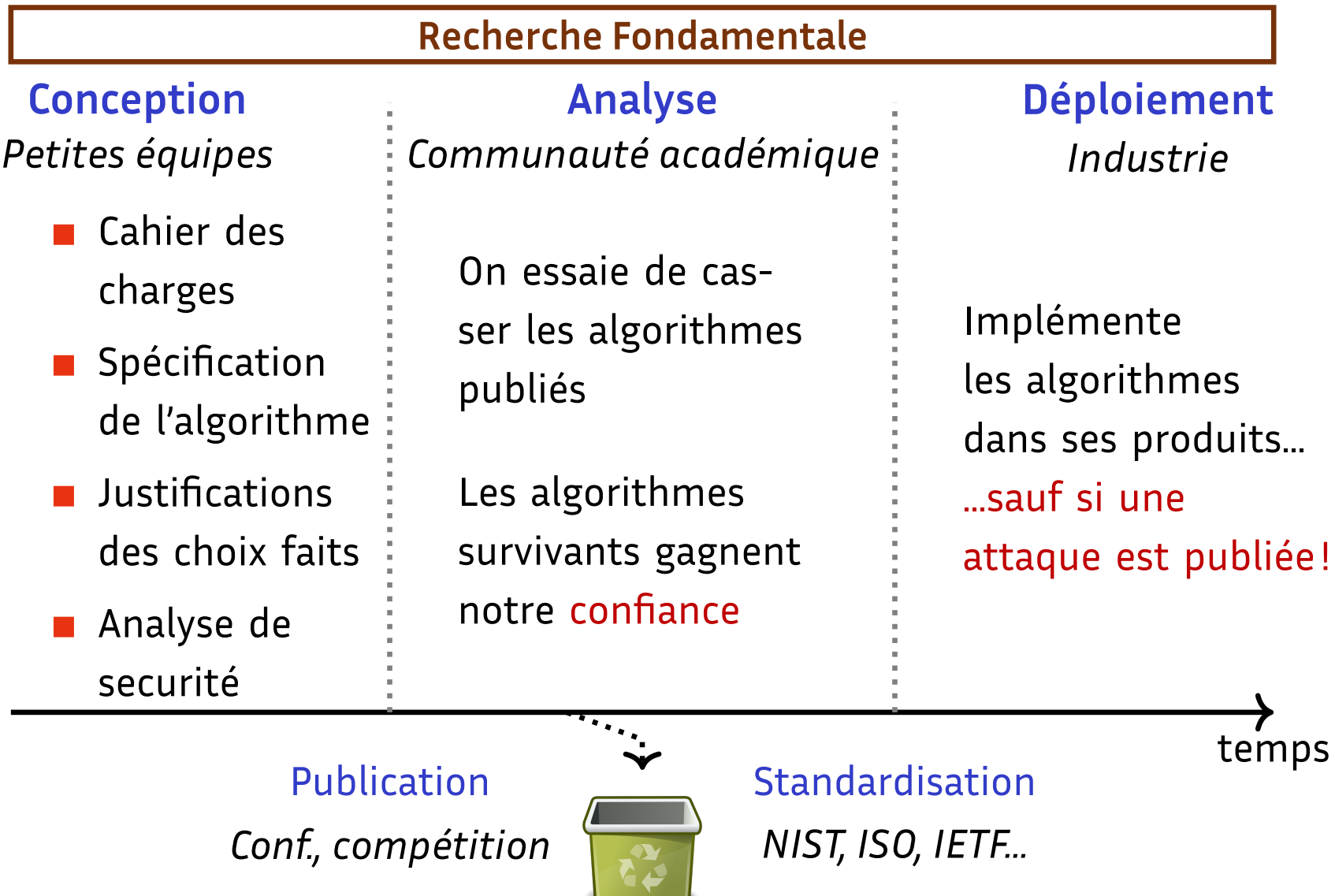
Cycle de vie d'un algorithme cryptographique



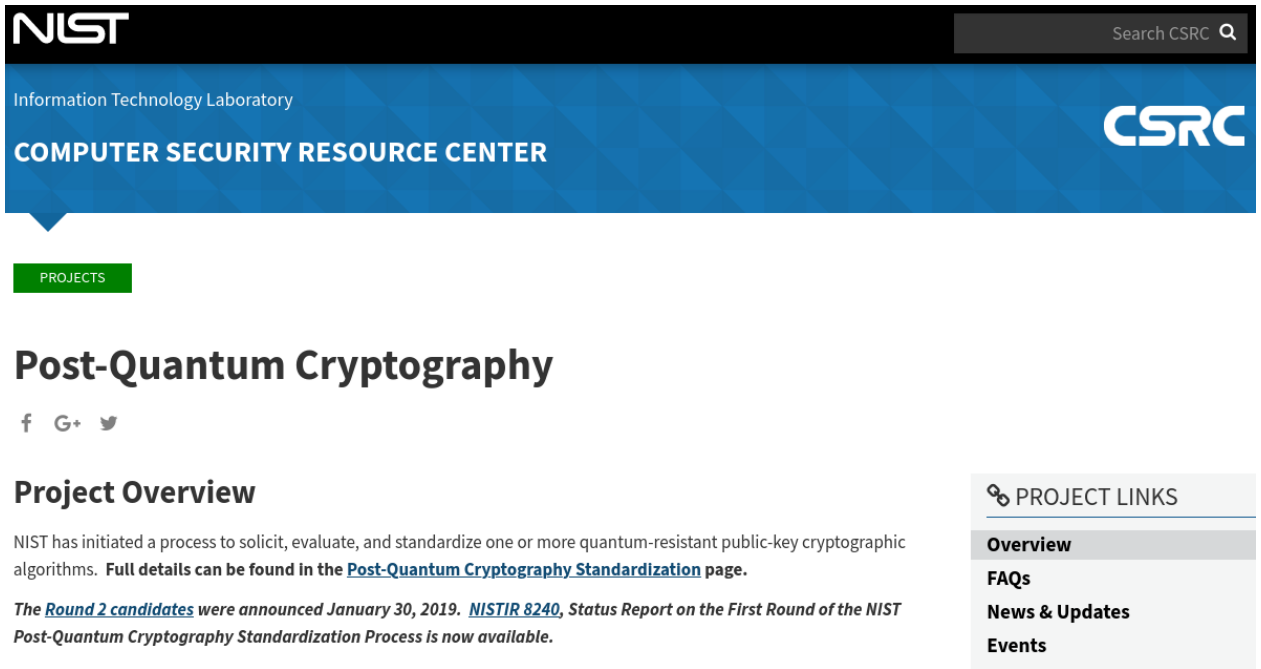
Cycle de vie d'un algorithme cryptographique



Cycle de vie d'un algorithme cryptographique



Post-Quantique



Les ordinateurs quantiques vont casser un grand nombres d'algorithmes de chiffrement (ceux à clef publique comme RSA).

$\Longrightarrow$ Il nous faut de nouveaux algorithmes !

Cryptographie légère

Les objets connectés ne peuvent pas supporter le coût (pourtant bas !) des algorithmes actuels.

$\Longrightarrow$ Il nous faut de nouveaux algorithmes !

Plan de la section à venir

Le coût de la cryptographie

Quand un ordinateur suit un algorithme, il consomme de l'électricité

- l'électricité n'est pas gratuite,
- les piles/batteries se vident...

Le coût de la cryptographie

Quand un ordinateur suit un algorithme, il consomme de l'électricité

- l'électricité n'est pas gratuite,
- les piles/batteries se vident…

Si on ne sécurise rien, alors ça ne coûte rien !

Manager à qui l'ingénieur sécurité explique l'importance de la cryptographie, allégorie.

Célèbre marque de voiture allemande

Des chercheurs démontrent la vulnérabilité des commandes d'ouverture à distance sur les voitures de série. [redacted] est en première ligne.

[redacted] n'en a pas fini avec Flavio D. Garcia.

Cet ingénieur en informatique à l'université de Birmingham reviendra, à l'occasion de la conférence Usenix dont l'**édition 2016** a lieu du 10 au 12 août à Austin (Texas), sur les conclusions établies dans son **rapport** « Lock it and still lose it ».

Des conclusions peu flatteuses pour le groupe automobile allemand : la quasi-totalité des véhicules qu'il a vendu ces 20 dernières années* peuvent être déverrouillés sans clés… et sans plip, du nom de cette télécommande aujourd'hui livrée en standard avec la plupart des voitures de tourisme.

`https://www.itespresso.fr/securite-it-hackers-voiture-cles-136067.html`

"SNCF" hollandaise

`https://en.wikipedia.org/wiki/OV-chipkaart#Security` (en)

Source image : wikipedia

Casser le processus

Conception

Petites équipes

- Cahier des charges
- Spécification de l'algorithme
- Justifications des choix faits
- Analyse de securité

Analyse

Communauté académique

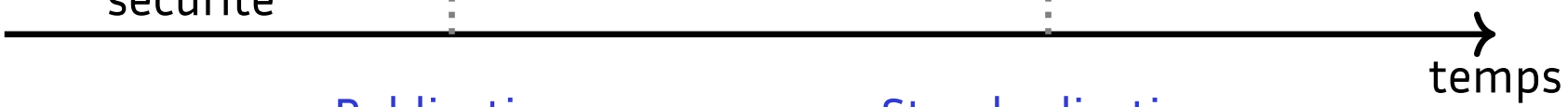
On essaie de casser les algorithmes publiés

Les algorithmes survivants gagnent notre confiance

Déploiement

Industrie

Implémente les algorithmes dans ses produits...
...sauf si une attaque est publiée !



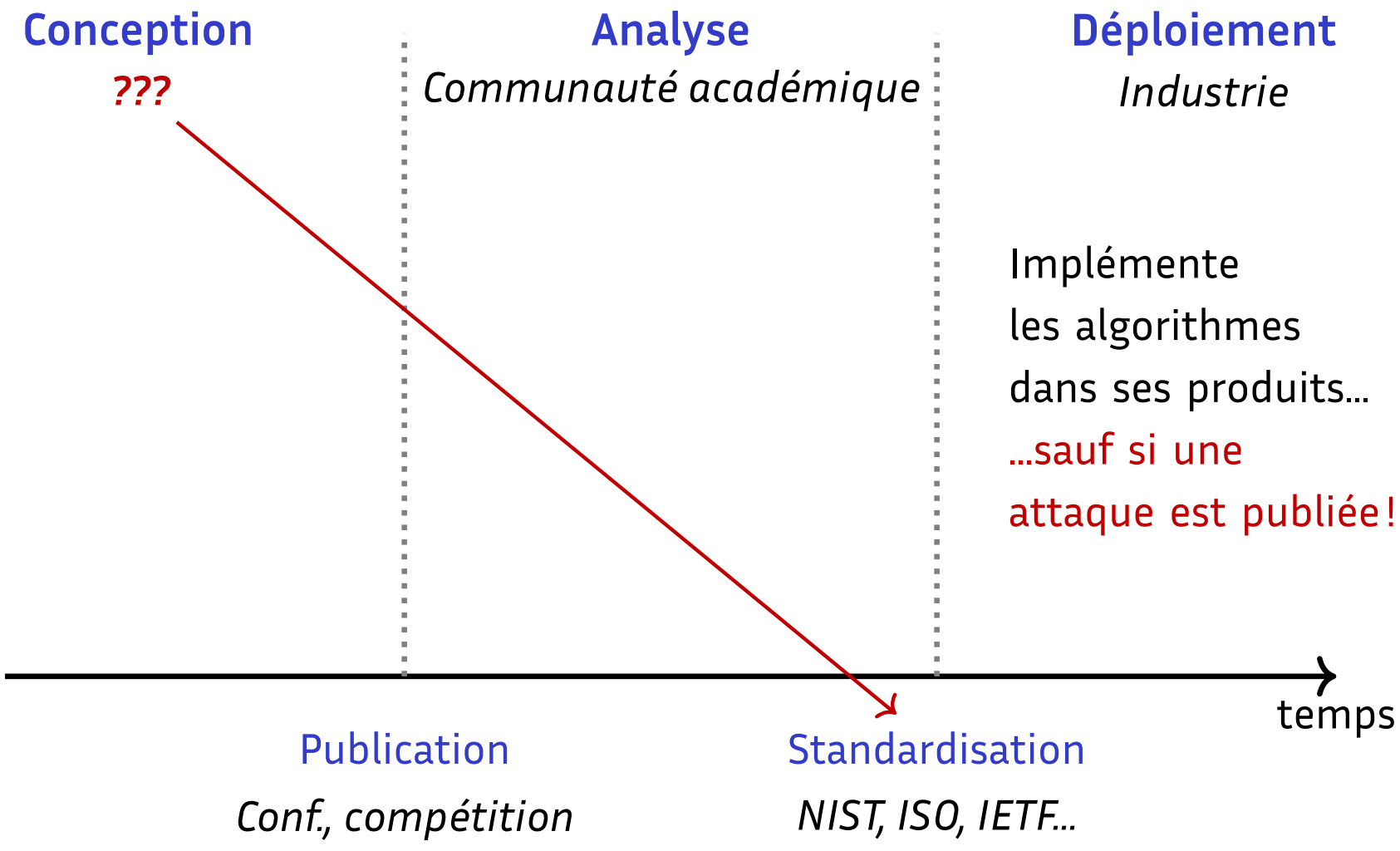
Publication

Conf., compétition

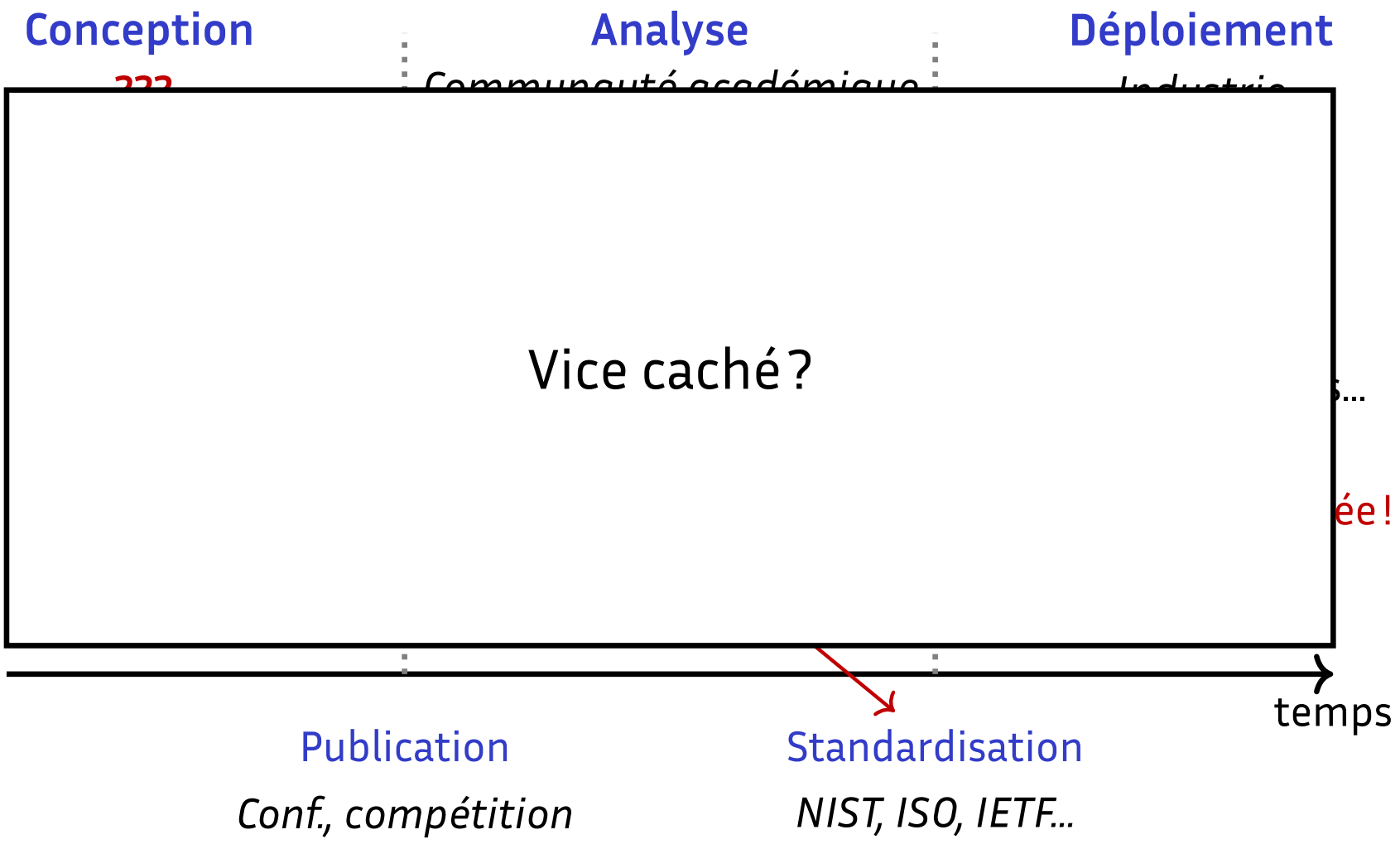
Standardisation

NIST, ISO, IETF...

Casser le processus



Casser le processus



DUAL EC

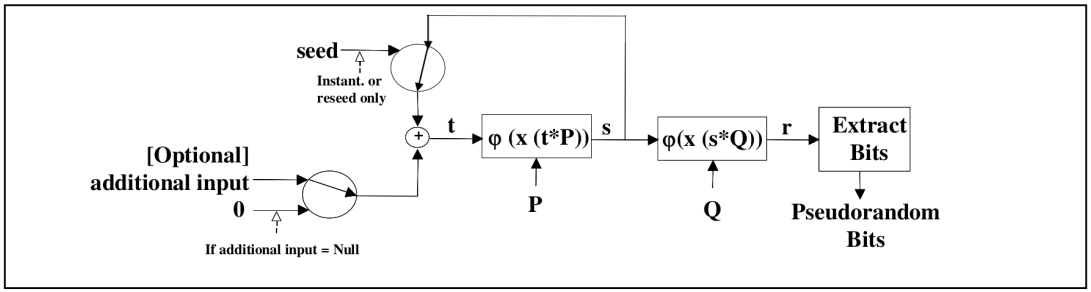


Figure 13: Dual_EC_DRBG

Pour aller plus loin (en anglais).

- Le fonctionnement de la backdoor : `https://eprint.iacr.org/2015/767`
- L'utilisation de cet algorithme a grandement facilité des attaques contre un logiciel de sécurité (Juniper). Détails : `https://eprint.iacr.org/2016/376`

Pourquoi implémenter un tel algorithme ?

`https://www.numerama.com/magazine/27869-la-nsa-aurait-paye-10-millions-de-dollars-pour-corrompre-rsa-qui-dement.html`

Kuznyechik

- "Kuznyechik" veut dire "sauterelle"
- C'est le nom du chiffrement par bloc standard en Russie depuis 2015.
- Il chiffre un nombre de $[0, 2^{128} - 1]$ avec une clef dans $[0, 2^{256} - 1]$.

Kuznyechik

- "Kuznyechik" veut dire "sauterelle"
- C'est le nom du chiffrement par bloc standard en Russie depuis 2015.
- Il chiffre un nombre de $[0, 2^{128} - 1]$ avec une clef dans $[0, 2^{256} - 1]$.
- Ses auteurs essaient d'en promouvoir l'usage partout dans le monde (pas juste en Russie)

Kuznyechik

- "Kuznyechik" veut dire "sauterelle"
- C'est le nom du chiffrement par bloc standard en Russie depuis 2015.
- Il chiffre un nombre de $[0, 2^{128} - 1]$ avec une clef dans $[0, 2^{256} - 1]$.
- Ses auteurs essaient d'en promouvoir l'usage partout dans le monde (pas juste en Russie)
- Aucunes explications sur le pourquoi du comment de sa conception n'ont été données !

Boîte-S

π' = (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241. 187, 20, 205, 95, 193, 249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182).

L'algorithme est très simple :

```
À la donnée de x entre 0 et 255:
    renvoyez la x-ieme entrée du tableau
```

Question naturelle

D'où sort ce tableau ?

Question naturelle

D'où sort ce tableau ?

questioned is the S-box π. This S-box was chosen from Streebog hash-function and it was synthesized in 2007. Note that through many years of cryptanalysis no weakness of this S-box was found. The S-box π was obtained by pseudo-random search and the following properties were taken into account.

[...]

No secret structure was enforced during construction of the S-box. At the same time, it is obvious that for any transformation a lot of representations are possible (see, for example, a lot of AES S-box representations).

"On l'a généré aléatoirement"

Question naturelle

D'où sort ce tableau ?

questioned is the S-box π. This S-box was chosen from Streebog hash-function and it was synthesized in 2007. Note that through many years of cryptanalysis no weakness of this S-box was found. The S-box π was obtained by pseudo-random search and the following properties were taken into account.

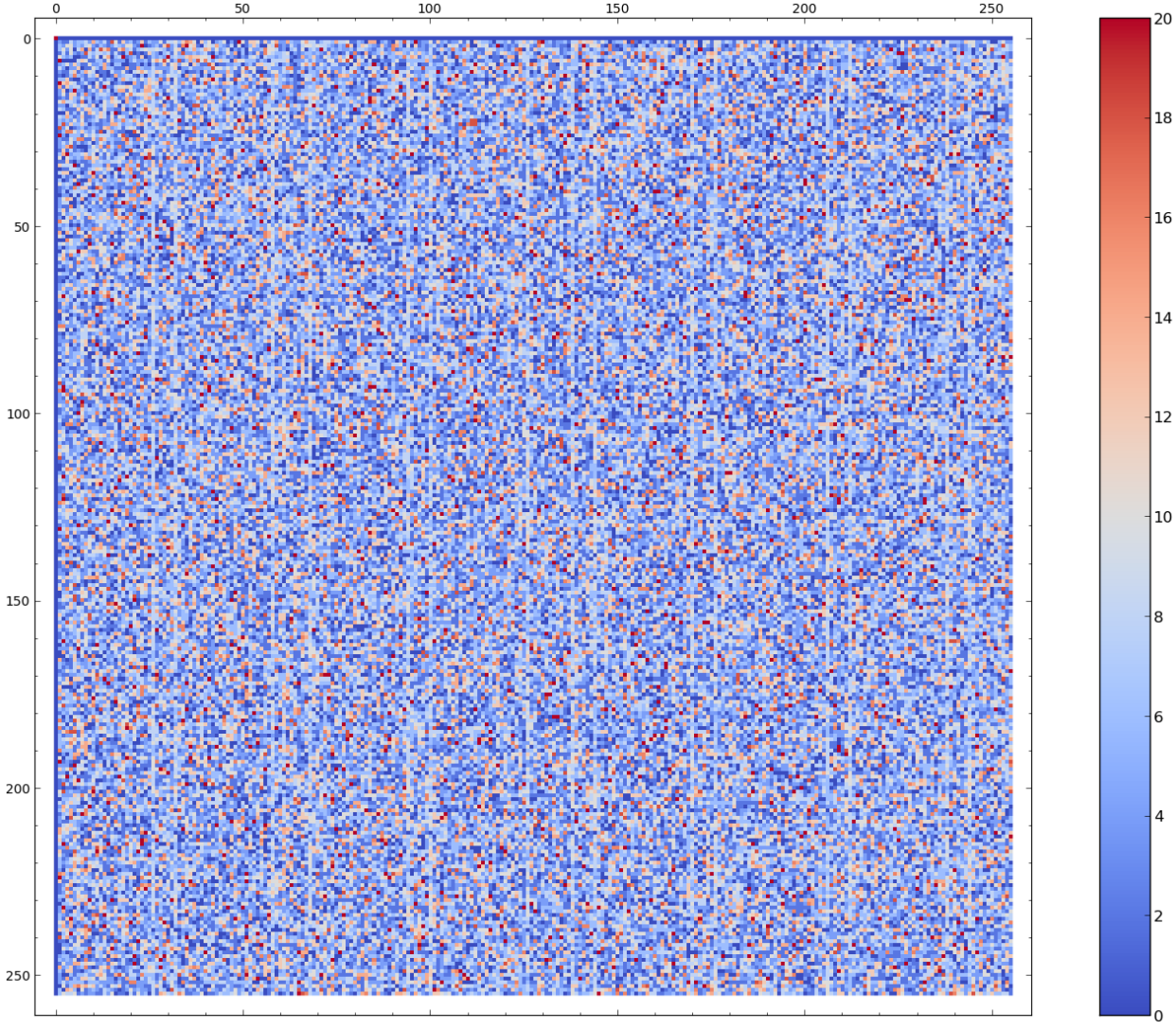
[...]

No secret structure was enforced during construction of the S-box. At the same time, it is obvious that for any transformation a lot of representations are possible (see, for example, a lot of AES S-box representations).

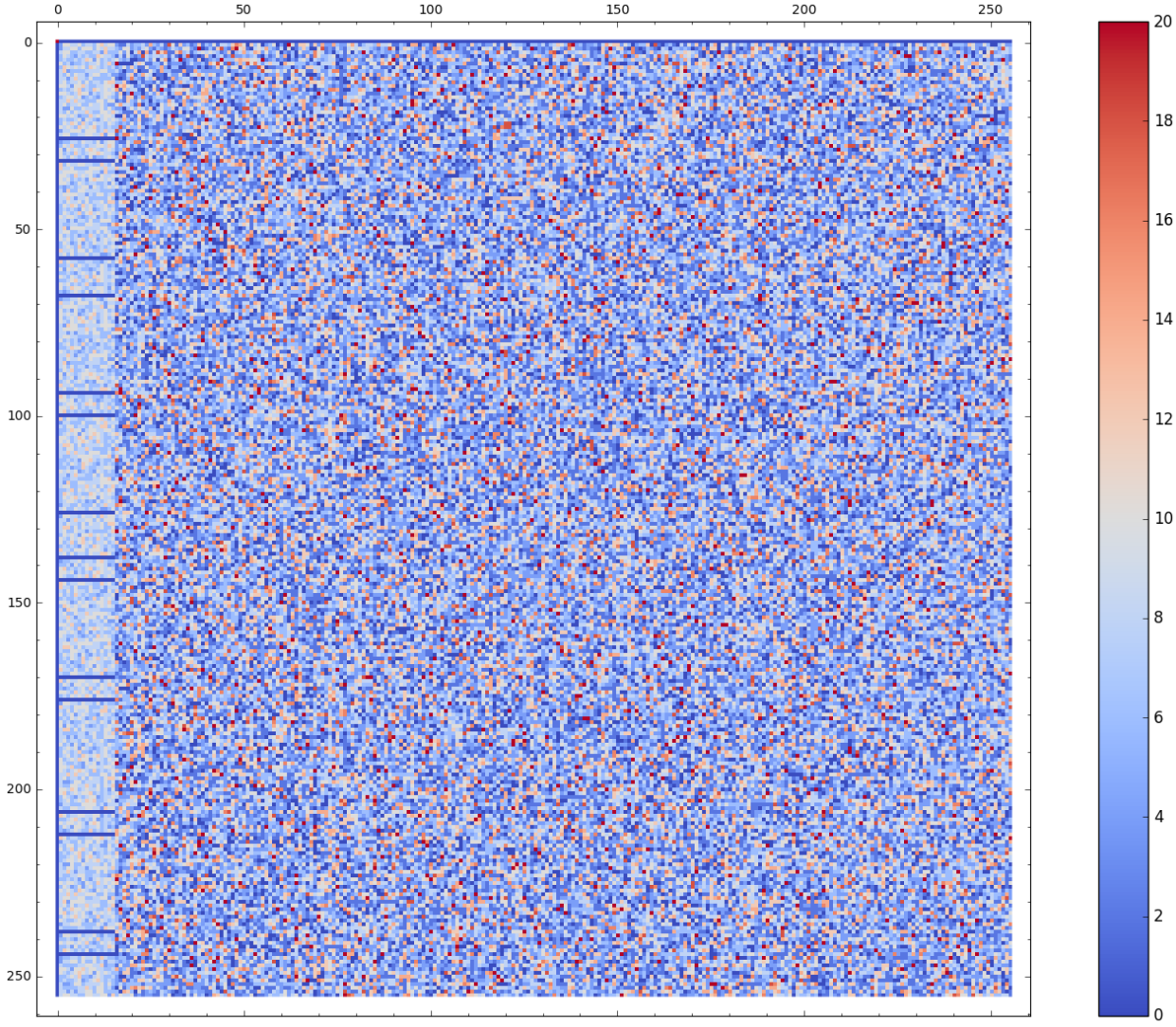
"On l'a généré aléatoirement"

"Par contre on a perdu notre algorithme de génération, donc on ne peut pas vous le montrer ! Désolés !"

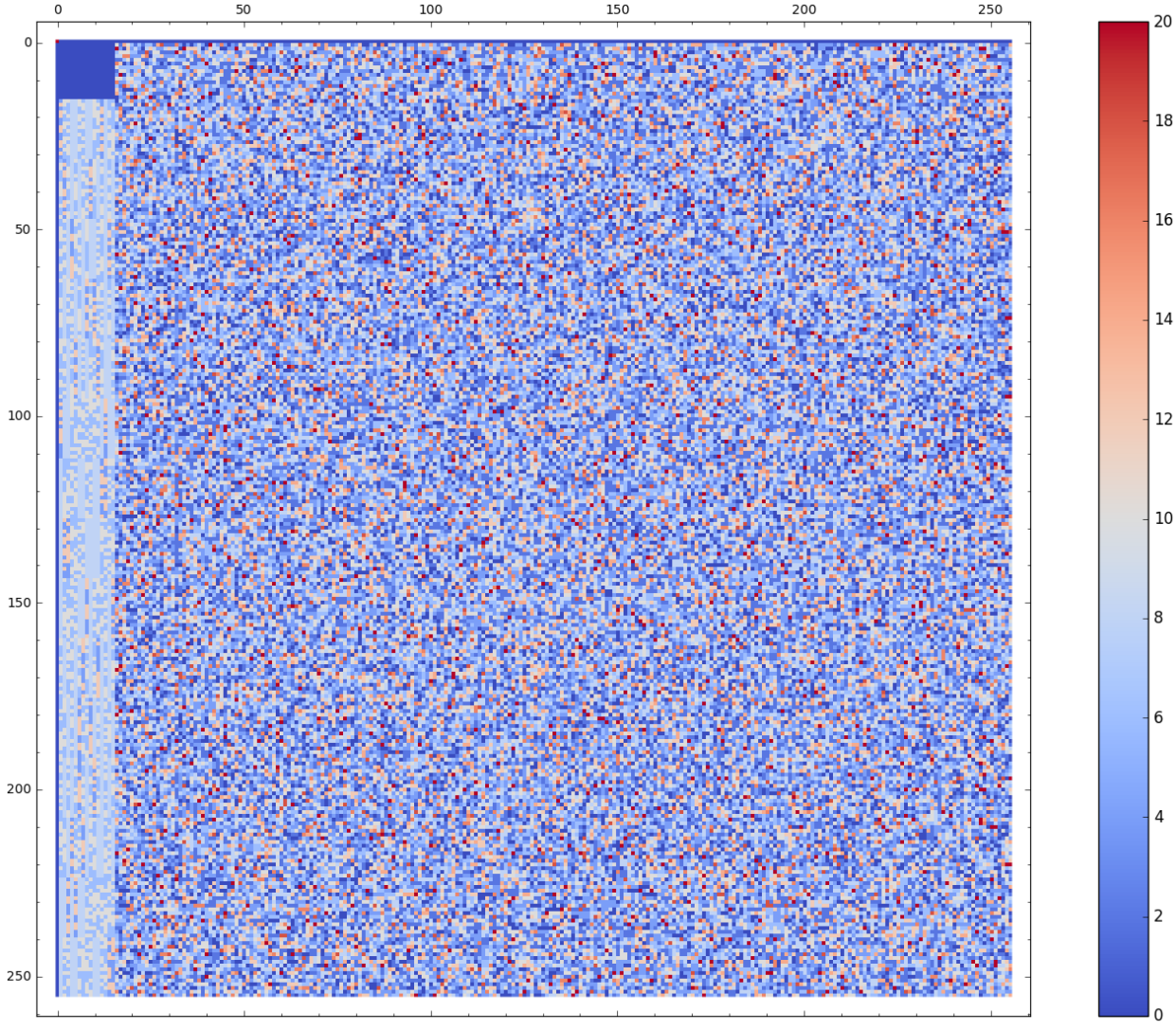
Représentation graphique de π



Ré-ordonnement des lignes



Représentation graphique de $\approx \pi$



La structure de π

$$\pi \begin{cases} \mathbb{F}_{2^8} & \longrightarrow \mathbb{F}_{2^8} \\ 0 & \mapsto \kappa(0)\,, \\ \left(\alpha^{2^m+1}\right)^j & \mapsto \kappa\left(2^m - j\right), \text{ for } 1 \leq j \leq 2^m - 1\,, \\ \alpha^{i+(2^m+1)j} & \mapsto \kappa\left(2^m - i\right) \oplus \left(\alpha^{2^m+1}\right)^{s(j)}, \text{ for } 0 < i, 0 \leq j < 2^m - 1\,. \end{cases}$$

La structure de π

$$\pi \begin{cases} \mathbb{F}_{2^8} & \longrightarrow \mathbb{F}_{2^8} \\ 0 & \mapsto \kappa(0)\,, \\ \left(\alpha^{2^m+1}\right)^j & \mapsto \kappa(2^m - j), \text{ for } 1 \leq j \leq 2^m - 1\,, \\ \alpha^{i+(2^m+1)j} & \mapsto \kappa(2^m - i) \oplus \left(\alpha^{2^m+1}\right)^{s(j)}, \text{ for } 0 < i, 0 \leq j < 2^m - 1\,. \end{cases}$$

"Non mais ça ne veut rien dire, il y a toujours une écriture aussi simple pour une fonction mathématique !"

Comment prouver que c'est faux ?

Taille d'un algorithme

Un algorithme est lui aussi un nombre !

1. On écrit l'algorithme propremment
2. On transforme le texte en séquence de nombres avec une taille maximale (typiquement, entre 0 et 255)
3. On les concatène.

Taille d'un algorithme

Un algorithme est lui aussi un nombre !

1. On écrit l'algorithme propremment
2. On transforme le texte en séquence de nombres avec une taille maximale (typiquement, entre 0 et 255)
3. On les concatène.

Example

A	J	O	U	T	E	R		1
065	074	079	085	084	069	082	032	049

Résultat : 65 074 079 085 084 069 082 032 049

Combien y'a-t-il de composants comme π ?

π est une permutation : elle change l'ordre des nombres entre 0 et 255.

À chaque sortie possible correspond exactement une entrée (pas collision).

Combien y'a-t-il de composants comme π ?

π est une permutation : elle change l'ordre des nombres entre 0 et 255.

À chaque sortie possible correspond exactement une entrée (pas collision).

$$\underbrace{\pi(0)}_{256\ \textit{possibilites}} \times \underbrace{\pi(1)}_{255\ \textit{possibilites}} \times \underbrace{\pi(2)}_{254\ \textit{possibilites}} \times \ldots \underbrace{\pi(255)}_{1\ \textit{possibilite}}$$

Au total, il y $256! \approx 10^{507}$ permutations de $[0, 255]$, c'est-à-dire d'algorithmes qui font le même genre de choses que π

Combien y'en a-t-il d'aussi simples que π ?

Definition

Complexité de Kolmogorov (simplifiée) La complexité d'un algorithme correspond à la taille du nombre le représentant.

"Combien y'a-t-il de permutations aussi simple que π ?" devient "Quelle est la taille de l'algorithme implémentant π ?"

La taille de π

Si on utilise la table originale, on trouve… plus que 507.

C'est logique : pour chaque permutation, un programme similaire existe. π n'a rien d'exceptionnel à ce niveau.

La taille de π

Si on utilise la table originale, on trouve... plus que 507.

C'est logique : pour chaque permutation, un programme similaire existe. π n'a rien d'exceptionnel à ce niveau.

Avec la structure que j'ai trouvée

```
p(x){unsigned char*k="@`rFTDVbpPB
vdtfR@\xacp?\xe2>4\xa6\xe9{z\xe3q
5\xa7\xe8",a=2,l=0,b=17;while(x&&
(l++,a^x))a=2*a^a/128*29;return l
%b?k[l%b]^k[b+l/b]^b:k[l/b]^188;}
```

En utilisant son équation mathématique secrète, on peut faire un programme qui tient sur 348 chiffres, et même beaucoup moins !

`https://codegolf.stackexchange.com/questions/186498/proving-that-a-russian-cryptographic-standard-is-too-structured`

Ça coince !

Il y a 10^{507} permutations comme π, mais il y a au plus 10^{349} algorithmes "au moins aussi simples" que π.

Ça coince !

Il y a 10^{507} permutations comme π, mais il y a au plus 10^{349} algorithmes "au moins aussi simples" que π.

Du coup, le nombre de permutations "au moins aussi simples" que π est au moins $10^{507-349} = 10^{158}$ fois plus petit que l'ensemble des permutations !

Ça coince !

Il y a 10^{507} permutations comme π, mais il y a au plus 10^{349} algorithmes "au moins aussi simples" que π.

Du coup, le nombre de permutations "au moins aussi simples" que π est au moins $10^{507-349} = 10^{158}$ fois plus petit que l'ensemble des permutations !

En fait, la probabilité qu'une permutation soit au moins aussi simple que π est environ la probabilité de gagner au LOTO 60 fois d'affilé.

Contre argument officiel

"On a calculé cette probabilité différemment et on est arrivés à 10^{-57}, ce qui est plutôt élevé".

Contre argument officiel

"On a calculé cette probabilité différemment et on est arrivés à 10^{-57}, ce qui est plutôt élevé".

$$0,\underbrace{000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 00}_{\text{56 zéros}}1 \approx 1$$

Contre argument officiel

"On a calculé cette probabilité différemment et on est arrivés à 10^{-57}, ce qui est plutôt élevé".

$$0,\underbrace{000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 000\ 00}_{\text{56 zéros}}1 \approx 1$$

L'argumentaire de la délégation russe, allégorie

Conclusion

Peut-on faire confiance à la cryptographie moderne ?

Conclusion

Peut-on faire confiance à la cryptographie moderne ?

Oui ... Mais !

Les bons algorithmes existent, encore faut-il les utiliser.

Conclusion

Peut-on faire confiance à la cryptographie moderne ?

Oui ... Mais !

Les bons algorithmes existent, encore faut-il les utiliser.

Merci !