# CRYPTA

Ludovic Perret

Sorbonne Universités, UPMC Univ Paris 06, INRIA Paris
LIP6, PolSyS Project, Paris, France
GitHub CRYPTA : https://github.com/lperret/EtuCrypta.git
DRAFT – Ne pas imprimer.

2017 – 2018




# Plan du cours

# Plan du cours

# RC4 [R. Rivest, 1987]

- Génération d'une suite chiffrante à partir d'un tableau $S$ de 256 **octets** et d'une clef $K$.

## Phase d'initialisation (RC4-KSA)

- **Pour** $i, 0 \leq i \leq 255$ **faire** $S[i] := i$ **FinPour**
- $j := 0$
- **Pour** $i, 0 \leq i \leq 255$ **faire**

  $j := (j + S[i] + K[i \mod \mathrm{len(K)}]) \bmod 2^8$
  Échanger$\big(S[i], S[j]\big)$
- **FinPour**

# RC4 [R. Rivest, 1987]

## Génération de la suite chiffrante (RC4-PRGA)

- $i := 0 \quad j := 0$
- **Pour** $k, 0 \leq k \leq \text{no} - 1$ **faire**

  $i := (i + 1) \bmod 2^8 \quad j := (j + S[i]) \bmod 2^8$

  Échanger$\big(S[i], S[j]\big)$

  $t := (S[i] + S[j]) \bmod 2^8$

  $\text{SuiteChiffrante}[k] := S[t]$
- **FinPour**
- **Return** SuiteChiffrante

# Faiblesses de RC4 [N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, Jacob C. N. Schuldt]

Biais sur la suite chiffrante : `http://www.isg.rhul.ac.uk/tls/`

# Compétition eStream


*Source :* `http://www.ecrypt.eu.org/stream/`

- Nouvelle génération de chiffrement à flot
- 2004 – 2008
- Révision en 2012

| Profile 1 (Soft, 128 bit) | Profile 2 (Hard, 80 bit) |
|---|---|
| HC-128 | Grain v1 |
| Rabbit | MICKEY 2.0 |
| Salsa20/12 | Trivium |
| SOSEMANUK | |

# Compétition eStream

ECRYPT II

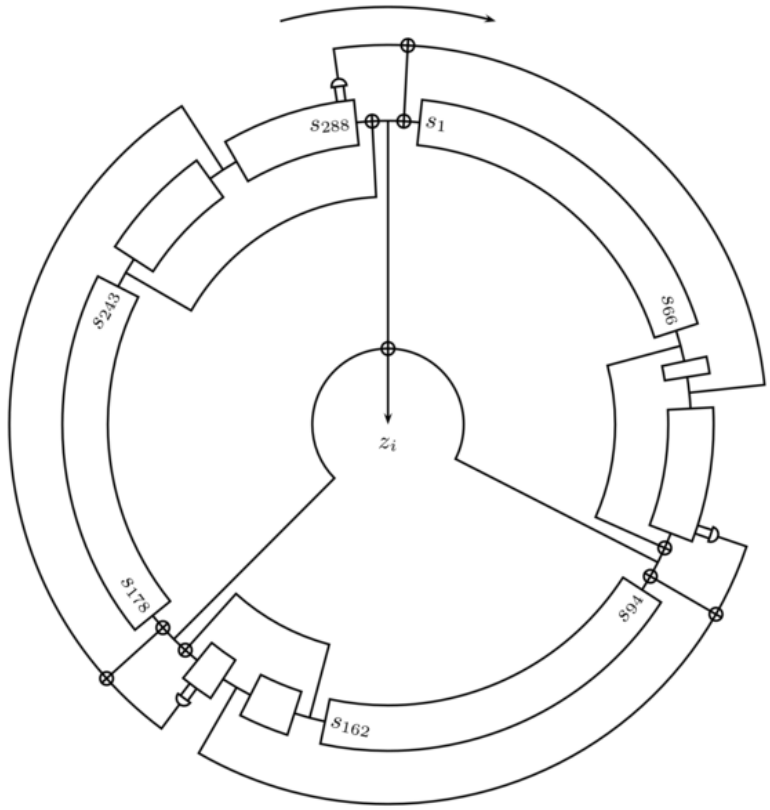
*Source :* `http://www.ecrypt.eu.org/stream/`

- Nouvelle génération de chiffrement par flot, 2004 – 2008
- Révision en 2012

| Type 1 (Soft) | Type 2 (Hard) |
|---|---|
| HC-128 | Grain v1 |
| Rabbit | MICKEY 2.0 |
| Salsa20/12 | Trivium |
| SOSEMANUK | |

# Plan du cours

# Trivium (Christophe De Cannière, Bart Preneel)




*Source :* `http://www.ecrypt.eu.org/stream/e2-trivium.html`

- Clé de 80 bit, registre de 288 bit
- 3 registres de 93, 84 et 111 bit
- Suite chiffrante de taille $\leq 2^{64}$ (pour une clé secrète)

# Description de Trivium

## Principe

- Inti. clé (80 bit) et IV (80 bit)
- Génération de flot

for $i := 1$ to $N$ do

$t_1 := s_{66} + s_{93}$

$t_2 := s_{162} + s_{177}$

$t_3 := s_{243} + s_{288}$

$z_i := t_1 + t_2 + t_3$

# Description de Trivium

## Principe

- Inti. clé (80 bit) et IV (80 bit)
- Génération de flot

for $i := 1$ to $N$ do

$t_1 := s_{66} + s_{93}$

$t_2 := s_{162} + s_{177}$

$t_3 := s_{243} + s_{288}$

$z_i := t_1 + t_2 + t_3$

$t_1 := s_{66} + s_{93} + s_{91}s_{92} + s_{171}$

$t_2 := s_{162} + s_{177} + s_{175}s_{176} + s_{264}$

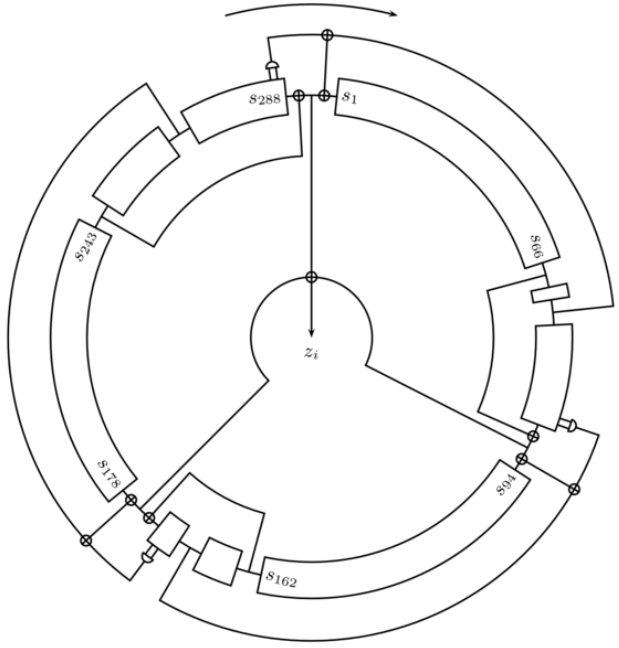
$t_3 := s_{243} + s_{288} + s_{286}s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) := (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \cdots, s_{177}) := (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \cdots, s_{288}) := (t_2, s_{178}, \ldots, s_{287})$

end for



# Description de Trivium – Initialisation

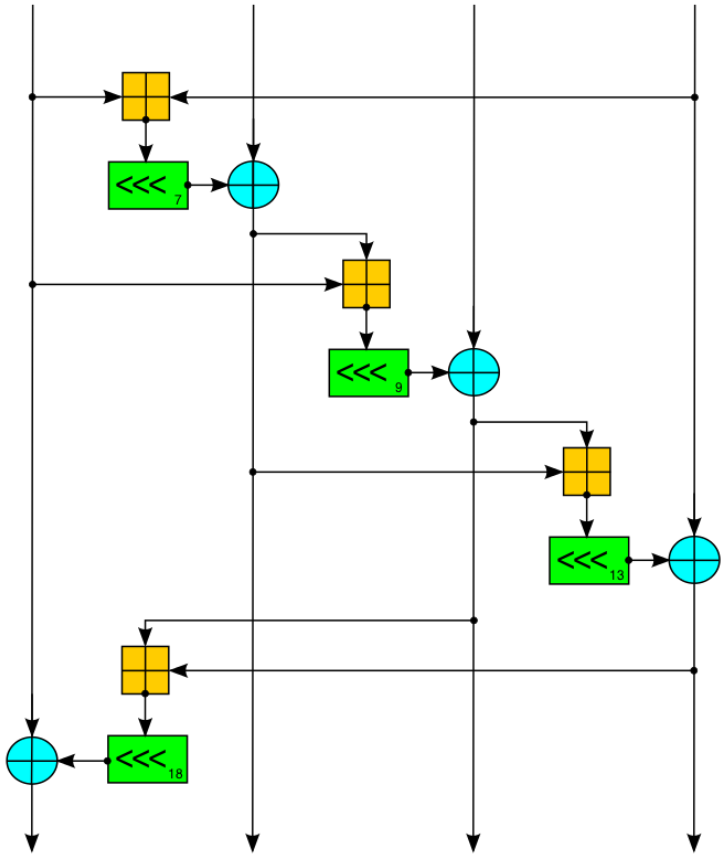
$(s_1, s_2, \ldots, s_{93}) := (K_1, K_2, \ldots, K_{80}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \cdots, s_{177}) := (\mathrm{IV}_1, \mathrm{IV}_2, \ldots, \mathrm{IV}_{80}, 0, \ldots, 0)$
$(s_{178}, s_{279}, \cdots, s_{288}) := (0, 0, \ldots, 1, 1, 1)$
for $i := 1$ to $4 \times 288$ do
    $t_1 := s_{66} + s_{93}$
    $t_2 := s_{162} + s_{177}$
    $t_3 := s_{243} + s_{288}$
    $z_i := t_1 + t_2 + t_3$

# Description de Trivium – Initialisation

$(s_1, s_2, \ldots, s_{93}) := (K_1, K_2, \ldots, K_{80}, 0, \ldots, 0)$
$(s_{94}, s_{95}, \cdots, s_{177}) := (\mathrm{IV}_1, \mathrm{IV}_2, \ldots, \mathrm{IV}_{80}, 0, \ldots, 0)$
$(s_{178}, s_{279}, \cdots, s_{288}) := (0, 0, \ldots, 1, 1, 1)$
for $i := 1$ to $4 \times 288$ do
$t_1 := s_{66} + s_{93}$
$t_2 := s_{162} + s_{177}$
$t_3 := s_{243} + s_{288}$
$z_i := t_1 + t_2 + t_3$
$t_1 := t_1 + s_{91} s_{92} + s_{171}$
$t_2 := t_2 + s_{175} s_{176} + s_{264}$
$t_3 := t_3 + s_{286} s_{287} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) := (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \cdots, s_{177}) := (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{279}, \cdots, s_{288}) := (t_2, s_{178}, \ldots, s_{287})$
end for

# Plan du cours

# Salsa20 (Dan Bernstein)

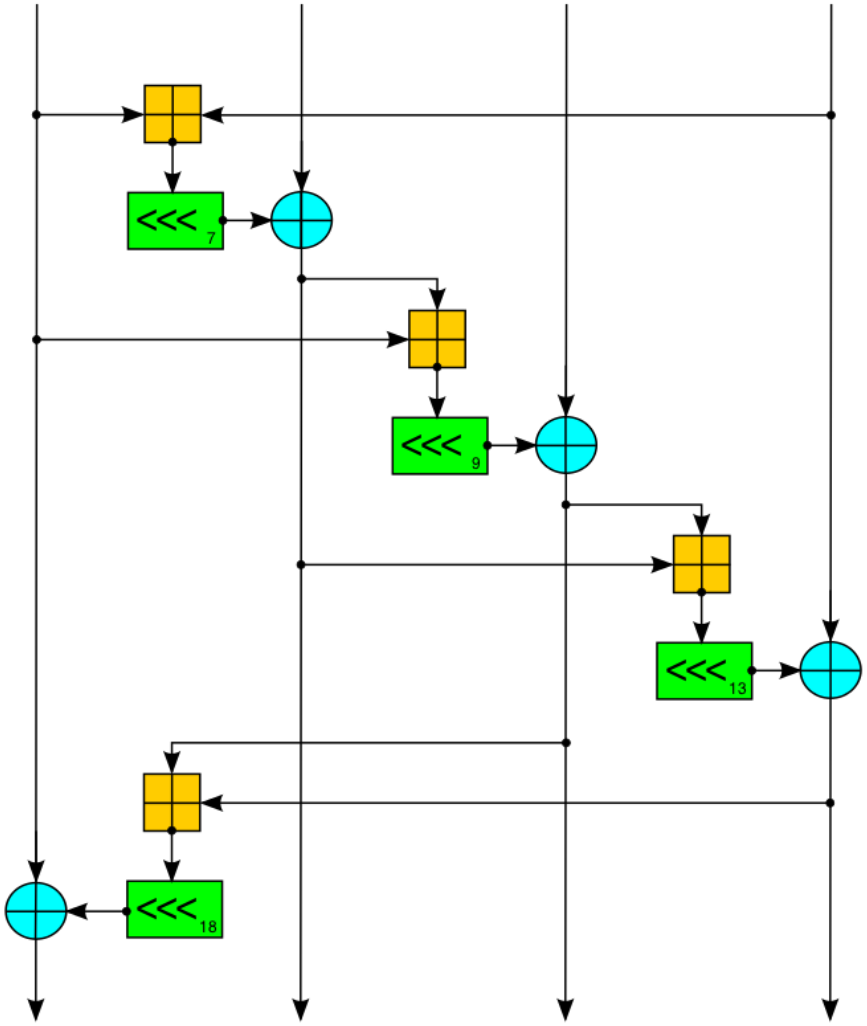



- ARX (Add, Rotate, XOR)
- Clé de 128 ou 256 bit
- Hachage en mode compteur

http://www.ecrypt.eu.org/stream/e2-salsa20.html

# Plan du cours

# Quarterround




$$y_0, y_1, y_2, y_3 \in \{0, \ldots, 2^{32} - 1\}$$

$$\begin{aligned} z_1 &:= y_1 \oplus ((y_0 \boxplus y_3) \lll 7) \\ z_2 &:= y_2 \oplus ((z_1 \boxplus y_0) \lll 9) \\ z_3 &:= y_3 \oplus ((z_2 \boxplus z_1) \lll 13) \\ z_0 &:= y_0 \oplus ((z_3 \boxplus z_2) \lll 18) \end{aligned}$$

- $\oplus$ : addition modulo 2
- $\boxplus$ : addition modulo $2^{32}$
- $\lll$ : rotation à gauche

# RowRound

$y_0, y_1, y_2, y_3, , \ldots, y_{15} \in \{0, \ldots, 2^{32} - 1\}$

$$\begin{aligned}
(z_0, z_1, z_2, z_3) &:= \text{quarterround}(y_0, y_1, y_2, y_3) \\
(z_5, z_6, z_7, z_4) &:= \text{quarterround}(y_5, y_6, y_7, y_4) \\
(z_{10}, z_{11}, z_8, z_9) &:= \text{quarterround}(y_{10}, y_{11}, y_8, y_9) \\
(z_{15}, z_{12}, z_{13}, z_{14}) &:= \text{quarterround}(y_{15}, y_{12}, y_{13}, y_{14})
\end{aligned}$$

$$\begin{pmatrix}
y_0 & y_1 & y_2 & y_3 \\
y_4 & y_5 & y_6 & y_7 \\
y_8 & y_9 & y_{10} & y_{11} \\
y_{12} & y_{13} & y_{14} & y_{15}
\end{pmatrix}$$

# ColumnRound

$x_0, x_1, x_2, x_3, \ldots, x_{15} \in \{0, \ldots, 2^{32} - 1\}$

$$
\begin{aligned}
(y_0, y_4, z_8, y_{12}) &:= \text{quarterround}(x_0, x_4, x_8, x_{12}) \\
(y_5, y_9, y_{13}, y_1) &:= \text{quarterround}(x_5, x_9, x_{13}, x_1) \\
(y_{10}, y_{14}, y_2, y_6) &:= \text{quarterround}(x_{10}, x_{14}, x_2, x_6) \\
(y_{15}, y_3, y_7, y_{11}) &:= \text{quarterround}(x_{15}, x_3, x_7, x_{11})
\end{aligned}
$$

$$
\begin{pmatrix}
x_0 & x_1 & x_2 & x_3 \\
x_4 & x_5 & x_6 & x_7 \\
x_8 & x_9 & x_{10} & x_{11} \\
x_{12} & x_{13} & x_{14} & x_{15}
\end{pmatrix}
$$

# Hachage `Salsa20`

$x = (x_0, x_1, x_2, \ldots, x_{15})$ avec $x_i \in \{0, \ldots, 2^{32} - 1\}$

$$\texttt{HachSalsa20}(x) = x + \text{doubleround}^{12}(x),$$

avec $\text{doubleround}(x) = \text{rowround}\big(\text{columnround}(x)\big)$.

# Hachage Salsa20

$x = (x_0, x_1, x_2, \ldots, x_{15})$ avec $x_i \in \{0, \ldots, 2^{32} - 1\}$

$$\texttt{HachSalsa20}(x) = x + \text{doubleround}^{12}(x),$$

avec $\text{doubleround}(x) = \text{rowround}\big(\text{columnround}(x)\big)$.

$$\texttt{HachSalsa20} : 16 \cdot 32 = 512 \text{ bit } \rightarrow 16 \cdot 32 = 512 \text{ bit}$$

# Plan du cours

# Expansion Salsa20

$k_0, k_1, n$ des séquences de $16 \cdot 8 = 128$ bit :

$$\texttt{ExpSalsa20}_{k_0,k_1}(n) = \texttt{HachSalsa20}(\sigma_0, k_0, \sigma_1, n, \sigma_2, k_1, \sigma_3)$$

$\sigma_0, \sigma_1, \sigma_2, \sigma_3$ des séquences fixées de $4 \cdot 8$ bit.

$$\texttt{ExpSalsa20} : 256 \times 128 \text{ bit } \rightarrow 16 \cdot 32 = 512 \text{ bit}$$

# Expansion Salsa20

$k_0, k_1, n$ des séquences de $16 \cdot 8 = 128$ bit :

$$\texttt{ExpSalsa20}_{k_0,k_1}(n) = \texttt{HachSalsa20}(\sigma_0, k_0, \sigma_1, n, \sigma_2, k_1, \sigma_3)$$

$\sigma_0, \sigma_1, \sigma_2, \sigma_3$ des séquences fixées de $4 \cdot 8$ bit.

$$\texttt{ExpSalsa20} : 256 \times 128 \text{ bit } \rightarrow 16 \cdot 32 = 512 \text{ bit}$$

$k, n$ des séquences $16 \cdot 8 = 128$ bit :

$$\texttt{ExpSalsa20}_k(n) = \texttt{HachSalsa20}(\tau_0, k, \tau_1, n, \tau_2, k, \tau_3)$$

$\tau_0, \tau_1, \tau_2, \tau_3$ des séquences fixées de $4 \cdot 8$ bit.

$$\texttt{ExpSalsa20} : 128 \times 128 \text{ bit } \rightarrow 16 \cdot 32 = 512 \text{ bit}$$

# Plan du cours

# Chiffrement `Salsa20`

- $k$ une séquence de $16 \cdot 8 = 128$ bit ou $32 \cdot 8 = 256$ bit
- $v$ une séquence de $8 \cdot 8 = 64$ bit :

$$\texttt{Salsa20}_k(v) = [(\texttt{ExpSalsa20}_k(v, \underline{i}))|i \in \{0, \ldots, 2^{64} - 1\}],$$

avec $\underline{i} = (i_0, i_1, \ldots, i_7)$ une séquence de $8 \cdot 8$ bit tel que

$$i = i_0 + 2^8 i_1 + \cdots + 2^{56} i_7.$$

# Chiffrement Salsa20

- $k$ une séquence de $16 \cdot 8 = 128$ bit ou $32 \cdot 8 = 256$ bit
- $v$ une séquence de $8 \cdot 8 = 64$ bit :

$$\texttt{Salsa20}_k(v) = [(\texttt{ExpSalsa20}_k(v, \underline{i}))|i \in \{0, \ldots, 2^{64} - 1\}],$$

avec $\underline{i} = (i_0, i_1, \ldots, i_7)$ une séquence de $8 \cdot 8$ bit tel que

$$i = i_0 + 2^8 i_1 + \cdots + 2^{56} i_7.$$

$m$ une séquence de $\ell$ octets, avec $\ell \in \{0, \ldots, 2^{70}\}$.

$$c = m \oplus \texttt{Salsa20}_k(v).$$

# Chiffrement `Salsa20`

- $k$ une séquence de $16 \cdot 8 = 128$ bit ou $32 \cdot 8 = 256$ bit
- $v$ une séquence de $8 \cdot 8 = 64$ bit :

$$\texttt{Salsa20}_k(v) = [(\texttt{ExpSalsa20}_k(v, \underline{i}))|i \in \{0, \ldots, 2^{64} - 1\}],$$

avec $\underline{i} = (i_0, i_1, \ldots, i_7)$ une séquence de $8 \cdot 8$ bit tel que

$$i = i_0 + 2^8 i_1 + \cdots + 2^{56} i_7.$$

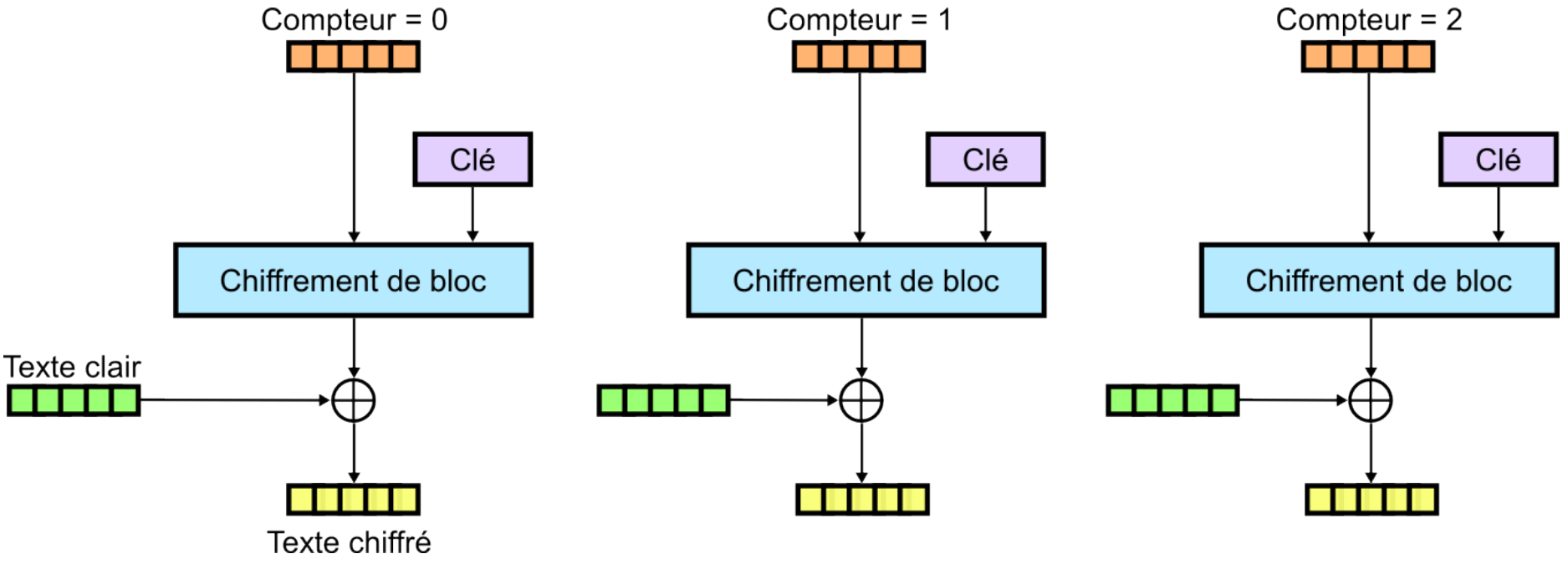
$m$ une séquence de $\ell$ octets, avec $\ell \in \{0, \ldots, 2^{70}\}$.

$$c = m \oplus \texttt{Salsa20}_k(v).$$

- Variante de `Salsa20` (`ChaCha20`) dans TLS
- Fonction de hachage BLAKE/BLAKE2

# Plan du cours

# Mode CTR




*Source :* `https://fr.wikipedia.org/`

# Plan du cours

# Plan du cours

# Paradoxe des anniversaires

## Proposition

Soient $H : X \to Y, x_1, \ldots, x_k$ des éléments distincts de $X$ tirés aléatoirement, et $y_i = H(x_i)$, pour tout $i, 1 \leq i \leq k$.

$$\Pr(\exists \text{ collision}) \approx 1 - e^{-\frac{k(k-1)}{2N}}, \text{ avec } N = |Y|.$$

# Paradoxe des anniversaires

## Démonstration.

- On suppose que les $y_i$ sont des éléments aléatoires de $Y$.

# Paradoxe des anniversaires

## Démonstration.

- On suppose que les $y_i$ sont des éléments aléatoires de $Y$.
- Nous avons $N = |Y|$. La probabilité que $y_{i+1} \notin \{y_1, \ldots, y_i\}$ est $p_{i+1} = (1 - i/N)$.

# Paradoxe des anniversaires

## Démonstration.

- On suppose que les $y_i$ sont des éléments aléatoires de $Y$.
- Nous avons $N = |Y|$. La probabilité que $y_{i+1} \notin \{y_1, \ldots, y_i\}$ est $p_{i+1} = (1 - i/N)$.
- La probabilité que les $y_1, \ldots, y_k$ – tirés dans cet ordre – soient distincts est

$$P = \prod_{i=0}^{k-1} p_{i+1} = \prod_{i=0}^{k-1} (1 - i/N).$$

# Paradoxe des anniversaires

## Démonstration.

- On suppose que les $y_i$ sont des éléments aléatoires de $Y$.
- Nous avons $N = |Y|$. La probabilité que $y_{i+1} \notin \{y_1, \ldots, y_i\}$ est $p_{i+1} = (1 - i/N)$.
- La probabilité que les $y_1, \ldots, y_k$ – tirés dans cet ordre – soient distincts est

$$P = \prod_{i=0}^{k-1} p_{i+1} = \prod_{i=0}^{k-1} (1 - i/N).$$

- La probabilité de non-collision est donc $P$.

# Paradoxe des anniversaires

## Démonstration.

- On suppose que les $y_i$ sont des éléments aléatoires de $Y$.
- Nous avons $N = |Y|$. La probabilité que $y_{i+1} \notin \{y_1, \ldots, y_i\}$ est $p_{i+1} = (1 - i/N)$.
- La probabilité que les $y_1, \ldots, y_k$ – tirés dans cet ordre – soient distincts est

$$P = \prod_{i=0}^{k-1} p_{i+1} = \prod_{i=0}^{k-1} (1 - i/N).$$

- La probabilité de non-collision est donc $P$.
- En approchant $1 - x$ par $e^{-x}$ pour $x$ proche de 0, on obtient : $P \simeq \prod_{i=0}^{k-1} e^{-\frac{i}{N}} = e^{-\frac{k(k-1)}{2N}}$.

# Collision

## Proposition

Soit $H : X \to Y$ une fonction de hachage, avec $|X| \geq |Y|$ et $|Y| = N$.
Pour trouver une collision avec probabilité $\geq 1/2$, il “suffit" de hacher :

$$\mathcal{O}(\sqrt{N}) \text{ éléments de } X.$$

## Autrement dit . . .

Pour avoir une probabilité $\geq 1/2$ de trouver une collision, il suffit de hacher un peu plus de $\sqrt{N}$ éléments de $X$.

# Preuve

## Démonstration.

Notons $\epsilon = 1 - P$, la probabilité d'avoir au moins une collision.
Exprimons $k$ en fonction de $\epsilon$ et $N$ :

$$\epsilon \simeq 1 - e^{-\frac{k(k-1)}{2N}} \Rightarrow -\frac{k(k-1)}{2N} \simeq \ln(1-\epsilon).$$

Ainsi, $k^2 - k \simeq 2N \ln(\frac{1}{1-\epsilon})$. En ignorant le terme $-k$, on obtient :

$$k \simeq \sqrt{2N \ln\left(\frac{1}{1-\epsilon}\right)}.$$

Pour $\epsilon = 1/2$, on trouve $k \simeq 1.18 \cdot \sqrt{N}$. □

# Illustration

- Supposons que $X$ est un ensemble d'individus
- $Y$ l'ensemble des 365 jours d'une année non bissextile
- $H(x)$, le jour de l'anniversaire d'une personne de $X$ (on suppose que $X$ comporte plus de 365 personnes)
- On obtient $k \simeq 1.18 \cdot \sqrt{365} \simeq 1.18 \cdot 19.10 \simeq 22.5$

# Plan du cours

# Fonction de compression

**Problème**

Comment gérer une donnée de taille variable ?

**Définition**

fonction de compression : fonction qui transforme toute chaîne d'une taille fixée $r + n$ en une chaîne de taille $n$.

$$f : \{0,1\}^{r+n} \quad \mapsto \quad \{0,1\}^{n}.$$

# Construction de Merle-Damgård – (I)

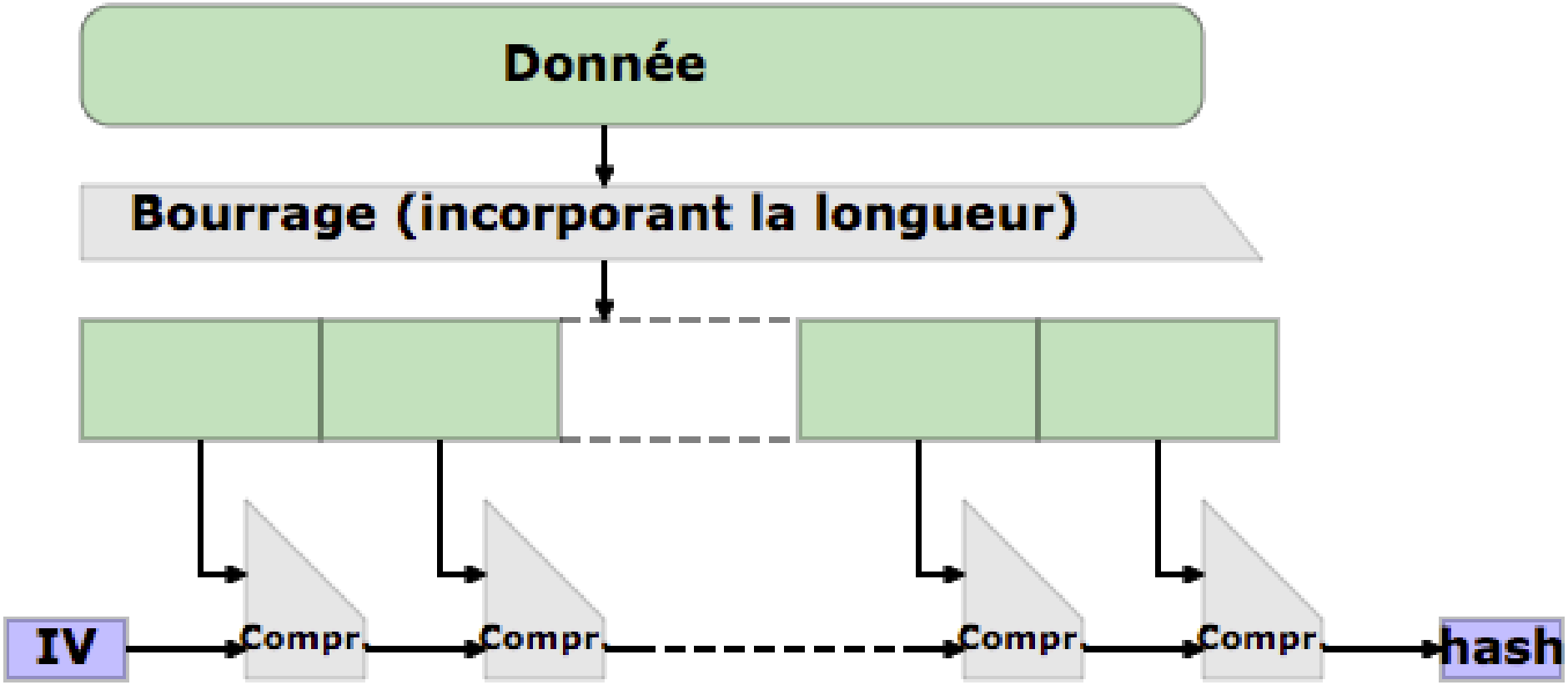
La chaîne $x$ (de longueur arbitraire) à hacher subit un prétraitement (padding) qui la transforme en $t$ blocs de $r$ bits $x_1, \ldots, x_t$.

- $\mathrm{IV} \in \{0,1\}^n$ une valeur initiale (ou vecteur d'initialisation),
- $f : \{0,1\}^r \times \{0,1\}^n \mapsto \{0,1\}^n$ une fonction de compression.

On calcule :

$$H_0 = \mathrm{IV},\ H_i = f(H_{i-1}, x_i),\ 1 \leq i \leq t.$$

# Merkle-Damgård




*Source :* `https://fr.wikipedia.org/`

**Remarque**

Fonction de hachage $\iff$ Fonction de compression

# Plan du cours

# SHA2 (SHA256, SHA384 et SHA512)

## SHA = Secure Hash Algorithm

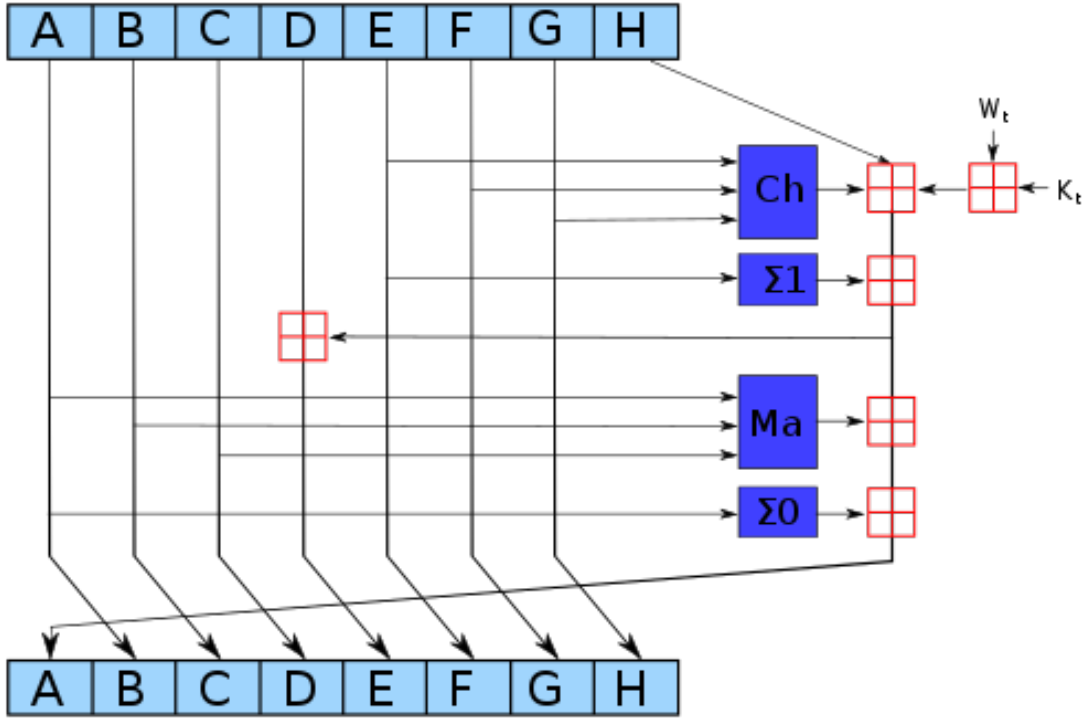
- Merkle-Damgård
- Fonction de compression ; taille des blocs $\in \{512, 1024\}$ bit
- Emprunte $\in \{224, 256, 384, 512\}$ bit

## Fonction de Compression – SHA256

- Variables de chaînage de 256 bits ($A, B, C, D, E, F, G, H$)
- 64 étapes élémentaires (tours)
- Expansion du bloc de message

  16 mots (32 bits) vers 64 mots

# SHA2 – Tour




*Source :* `https://fr.wikipedia.org/`

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$Ma(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$
$$\Sigma_0(x) = \mathrm{ROT}^2(x) \oplus \mathrm{ROT}^{13}(x) \oplus \mathrm{ROT}^{22}(x)$$
$$\Sigma_1(x) = \mathrm{ROT}^6(x) \oplus \mathrm{ROT}^{11}(x) \oplus \mathrm{ROT}^{25}(x)$$

# SHA12 – (II)

Expansion de message : $W_i = m_i, \forall i, 0 \leq i \leq 15$, et

$$W_t = \sigma_0(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16},$$

pour $t, 16 \leq i \leq 63$.

$$\sigma_0(x) = \mathrm{ROT}^7(x) \oplus \mathrm{ROT}^{18}(x) \oplus \mathrm{SHR}^3(x)$$
$$\sigma_1(x) = \mathrm{ROT}^{17}(x) \oplus \mathrm{ROT}^{19}(x) \oplus \mathrm{SHR}^{10}(x)$$

# Plan du cours

# Compétition SHA3

Xiaoyun Wang, Hongbo Yu.
*How to Break MD5 and Other Hash Functions.*
EUROCRYPT 2005.

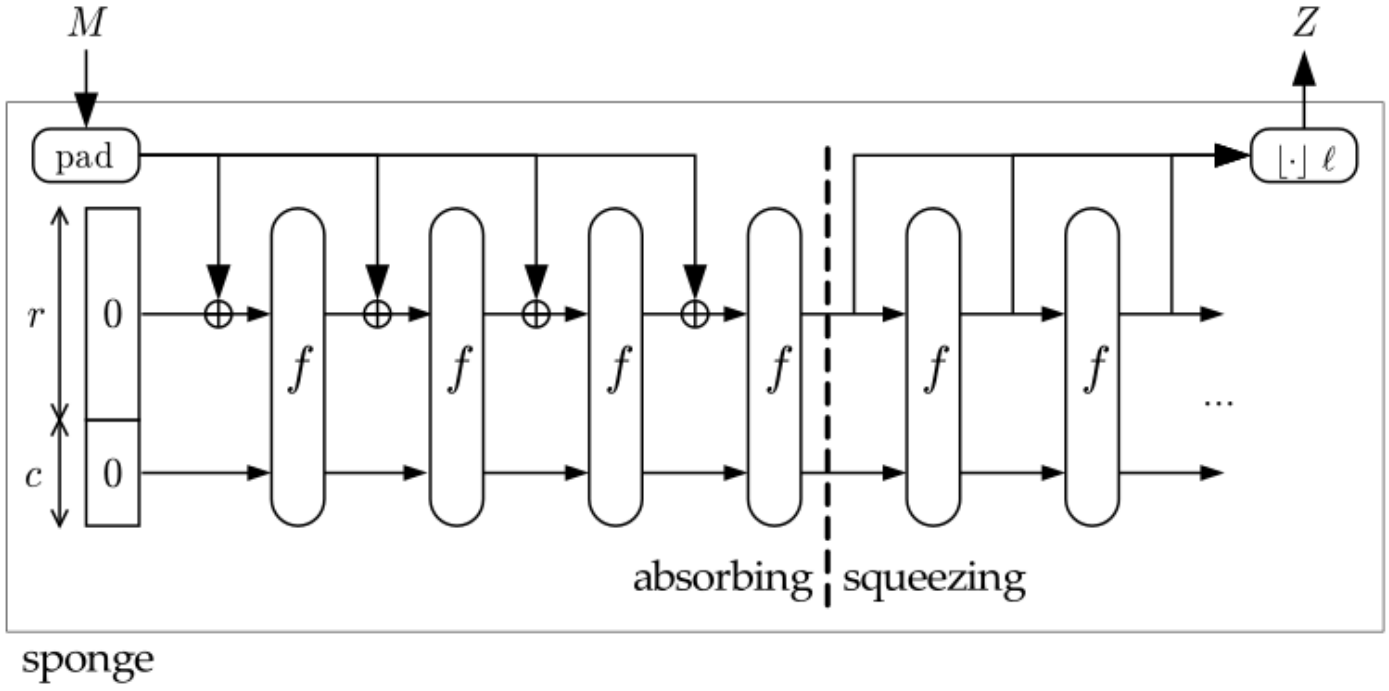
## Nouveau standard

- 64 soumissions (2008)
- 14 candidats en phase 2
- 5 candidats en phase 3 (2010)

## SHA3, 2012

- Keccak (G. Bertoni, J. Daemen, M. Peeters, G. Van Assche)

# Construction sponge



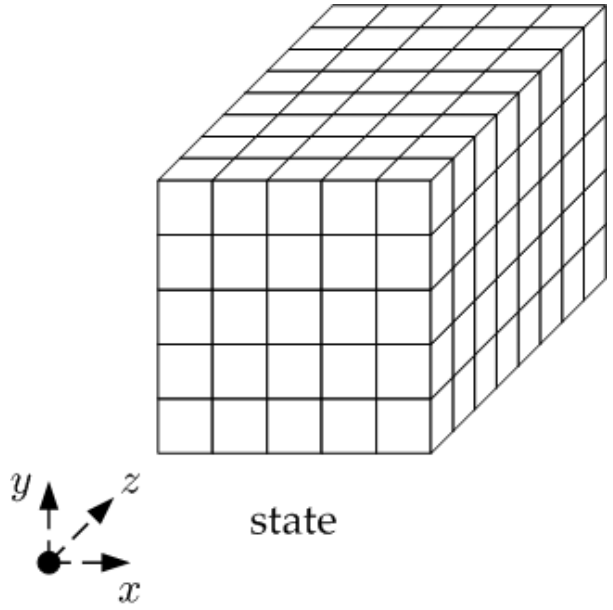

*Source :* `https://keccak.team/sponge_duplex.html`

- $r$, le taux (taille d'un bloc)
- $c$, la capacité
- $f$, permutation sur $b = r + c$ bit

## Niveau de sécurité

$$2^{c/2}$$

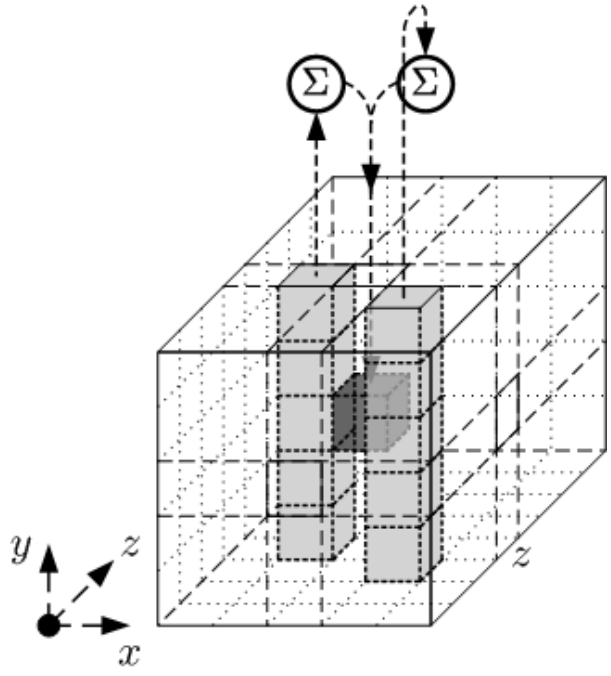
# Keccak-f – Structure de donnée




*Source :* `https://keccak.team`

- $2^\ell$ tableaux de $5 \times 5$ bit, avec $\ell \in \{1, 2, 5, 8, 16, 32, 64\}$
- $b = 25 \times 2^\ell$

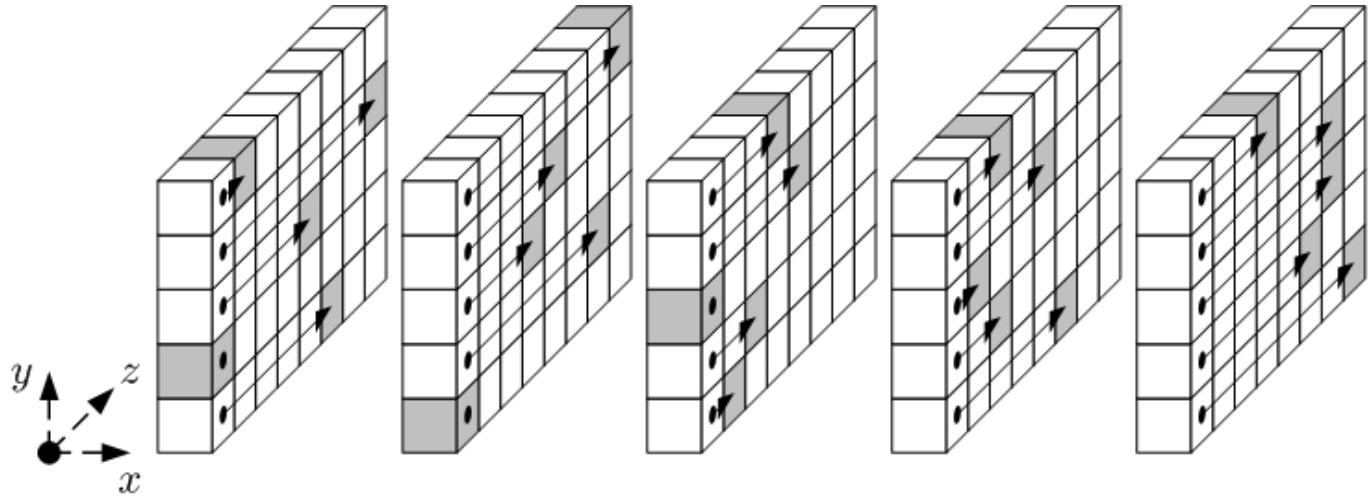
# Keccak-f – Fonction $\theta$




*Source :* `https://keccak.team`

$$a[i][j][k] := a[i][j][k] \oplus \sum_{j'=0}^{4} a[i-1][j'][k] \oplus \sum_{j'=0}^{4} a[i+1][j'][k-1]$$

# Keccak-f – Fonction $\rho$



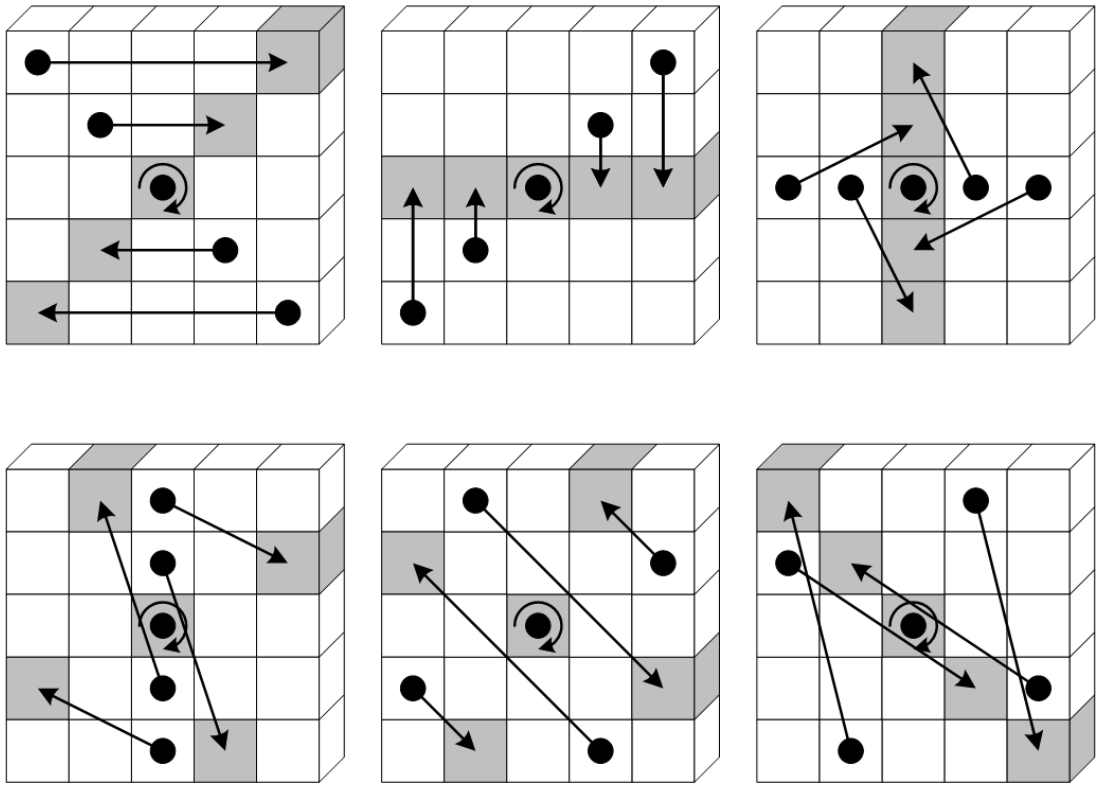

*Source :* `https://keccak.team`

$$a[i][j][k] := a[i][j][k - (t+1)(t+2)/2],$$

avec $t = -1$ si $i = j = 0$ ; sinon $t, 0 \leq t \leq 24$ et :

$$\begin{pmatrix} i \\ j \end{pmatrix} \equiv \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 0 \\ 1 \end{pmatrix} \bmod 5.$$

# Keccak-f – Fonction $\pi$

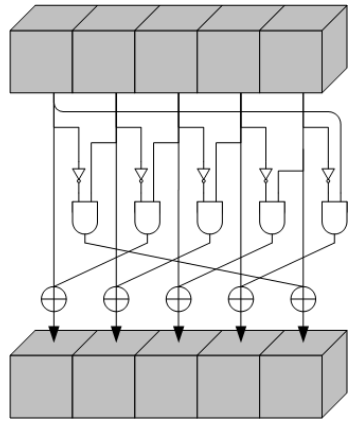


*Source :* `https://keccak.team`

$$a[i][j] := a[i'][j'], \text{avec}$$

$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix}$$

# Keccak-f – Fonction $\chi$



*Source :* `https://keccak.team`

$$a[i] := a[i] + (a[i+1] + 1)a[i+2].$$

# Keccak-f

On répète $n_r = 12 + 2\ell$ fois :

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta.$$

avec $\iota$ :

$$a := a + RC[i_r].$$

# Keccak-f

On répète $n_r = 12 + 2\ell$ fois :

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta.$$

avec $\iota$ :

$$a := a + RC[i_r].$$

## SHA3

| | sortie | $r$ | $c$ | Collision |
|---|---|---|---|---|
| SHA3-224 | 224 | 1152 | 448 | 112 |
| SHA3-256 | 256 | 1088 | 512 | 128 |
| SHA3-384 | 384 | 832 | 768 | 192 |

# Plan du cours

# Plan du cours

# WEP

## Wired Equivalent Privacy (a.k.a. *Weak Encryption Protocol*)

- Protocole permettant de sécuriser les réseaux sans fil de type Wi-Fi
- Fournir aux réseaux sans fil une confidentialité comparable à celle d'un réseau local filaire classique
- Norme de sécurité introduite dans IEEE 802.11 (1999)

# WEP – Quelques Caractéristiques

- Utilisation d'une **clef secrète** partagée entre les entités communicantes et l'Acces Point (AP)
  - **Chiffrement à clef secrète (RC4)** pour assurer la confidentialité des donnés (trames, paquets)
  - **Code linéaire CRC32** pour assurer l'intégrité

# CRC32

## Cyclic Redundancy Check

Le contrôle de redondance cyclique (CRC) est un moyen de tester l'intégrité des données (i.e. détecter des erreurs).

- Code correcteur ; pas une primitive cryptographique

# CRC32 – Encodage

- **Encodage.** Pour un CRC$n$, on ajoute $n(=32)$ bit au message (trame) à envoyer.
- $\mathbf{M}' = X^n \cdot \mathbf{M}(X)$ est le message correspondant aux bit de la trame $\mathbf{M}(X)$ auquel nous avons ajouter $n$ zéros.

## Exemple

Nous pouvons voir 0110101001 comme le polynôme :

$$0 \cdot X^9 + X^8 + X^7 + 0 \cdot X^6 + X^5 + 0 \cdot X^4 + X^3 + 0 \cdot X^2 + 0 \cdot X^1 + X^0.$$
$$X^8 + X^7 + X^5 + X^3 + 1.$$

# CRC32 – Principe

- $\mathrm{CRC}_{\mathbf{M}'}$ est le **reste de la division Euclidienne** de $\mathbf{M}'(X)$ par un polynôme $\mathbf{G}(X)$ **fixé** :

$$\mathbf{M}'(X) = \mathbf{Q}(X)\mathbf{G}(X) \oplus \mathrm{CRC}_{\mathbf{M}'}(X),$$

avec $\mathrm{CRC}_{\mathbf{M}'}(X) = 0$ ou $\deg\left(\mathrm{CRC}_{\mathbf{M}'}(X)\right) < \deg(G(X))$.

  - Pour CRC32, $\mathbf{G}(X) =$

$$X^{32} \oplus X^{26} \oplus X^{23} \oplus X^{22} \oplus X^{16} \oplus X^{12} \oplus X^{11} \oplus X^{10} \oplus X^8 \oplus X^7 \oplus X^5 \oplus X^4 \oplus X^2 \oplus X \oplus 1.$$

- On transmet :

$$\mathbf{M} \,||\, \mathrm{CRC}_{\mathbf{M}'}.$$

# CRC32 – Linéarité

$$\begin{aligned} \mathbf{M}'(X) &= \mathbf{Q}'(X)\mathbf{G}(X) \oplus \mathrm{CRC}_{\mathbf{M}'}(X), \\ \mathbf{M}''(X) &= \mathbf{Q}''(X)\mathbf{G}(X) \oplus \mathrm{CRC}_{\mathbf{M}''}(X). \end{aligned}$$

- Nous avons $\mathbf{M}'(X) \oplus \mathbf{M}''(X)$ :

$$\big(\mathbf{Q}'(X) \oplus \mathbf{Q}''(X)\big)\mathbf{G}(X) \oplus \big(\mathrm{CRC}_{\mathbf{M}'}(X) \oplus \mathrm{CRC}_{\mathbf{M}''}(X)\big),$$

avec $\mathrm{CRC}_{\mathbf{M}'}(X) \oplus \mathrm{CRC}_{\mathbf{M}''}(X) = 0$ ou
$\deg\big(\mathrm{CRC}_{\mathbf{M}'}(X) \oplus \mathrm{CRC}_{\mathbf{M}''}(X)\big) < \deg(G(X))$.

$$\boxed{\mathrm{CRC}_{\mathbf{M}' \oplus \mathbf{M}''} = \mathrm{CRC}_{\mathbf{M}'} \oplus \mathrm{CRC}_{\mathbf{M}''}}$$

# Plan du cours

# Caractéristiques

- Une même clef **K** (root key) partagée par les entités
  - 40 bits (5 octets)
  - 104 bits (13 octets)
  - 232 bits (29 octets)
- On ajoute (concatène) à cette clef un IV de 24 bit.

$$\mathbf{K} \,||\, \mathrm{IV}, \text{per packet key}.$$

- **Pas de précision** (initialement) dans 802.11 sur la gestion de l'IV
  - Génération aléatoire ?
  - On incrémente l'IV ?

# Chiffrement WEP

## Principe

Soit **K** la clef partagée (root key).

- **Formatage.** Soit **M** le message (trame) à chiffrer.

$$\mathbf{M}' = \mathbf{M} \,||\, \mathrm{ICV}(\mathbf{M}), \text{ avec } \mathrm{ICV}(\mathbf{M}) = \mathrm{CRC32}(\mathbf{M}),$$

  ICV étant Integrity Check Value.
- **Chiffrement.** On génère un $\mathrm{IV} \in \mathbb{F}_2^{24}$ (i.e. 24 bits) et :

$$\mathbf{C} = \mathbf{M}' \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}).$$

# Rafraîchissement de IV

- On note :

$$\mathbf{C}_1 = \mathbf{M}'_1 \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}) \qquad \mathbf{C}_2 = \mathbf{M}'_2 \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}).$$

- Nous avons une *collision* :

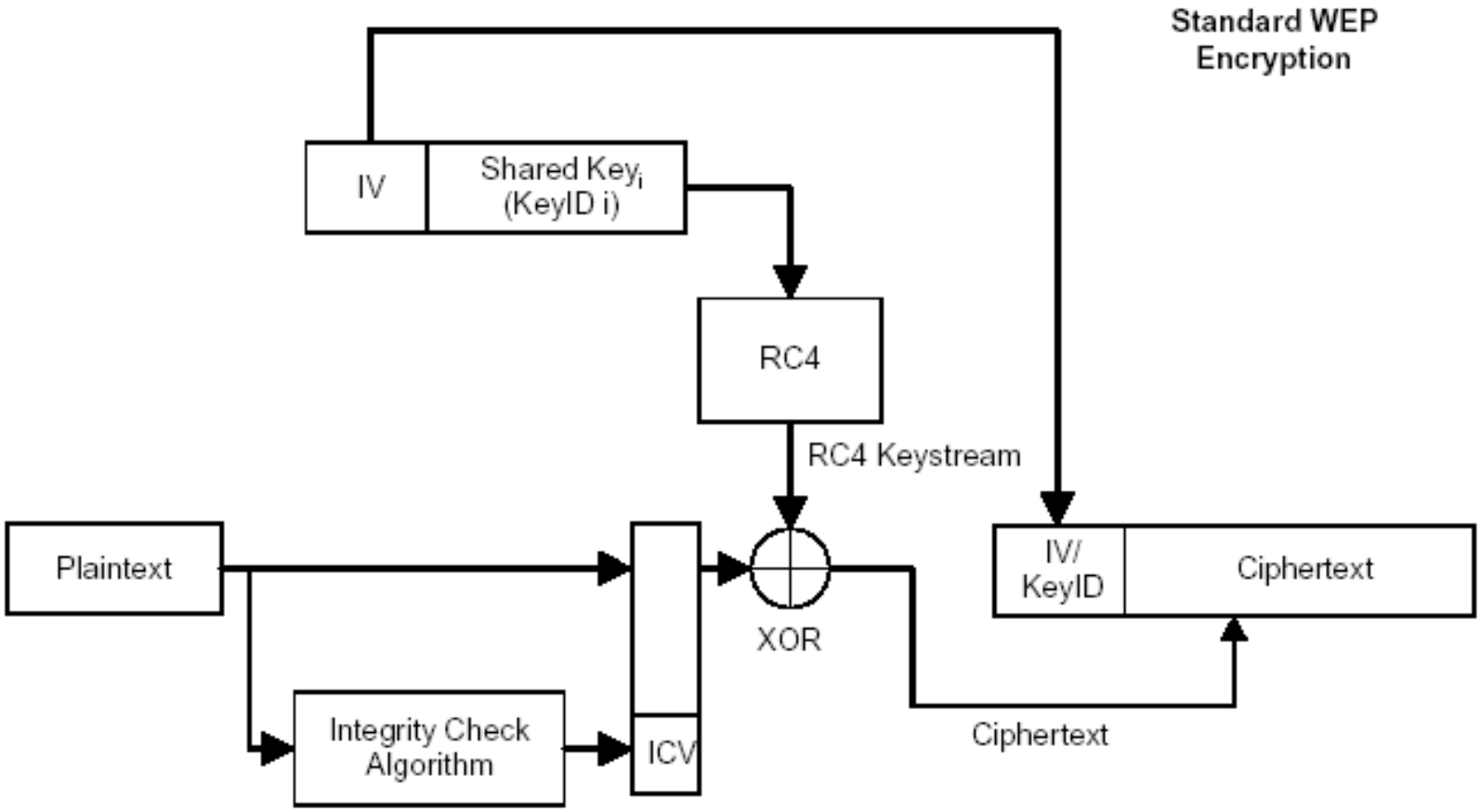
$$\mathbf{C}_1 \oplus \mathbf{C}_2 = \mathbf{M}'_1 \oplus \mathbf{M}'_2.$$

$\Rightarrow$ Changement de l'IV nécessaire.

# Trame 802.11


Vecteur d'initialisation IV (en clair)
N° de clé : 6 bits réservés, 2 bits pour le KeyID
Données Chiffrées
ICV Chiffré
3 octets (24 bits)
1 octet
jusqu'à 2304 octets
4 octets (32 bits)

# Chiffrement WEP – Résumé



# Open System Authentification – WEP

- authentification automatique

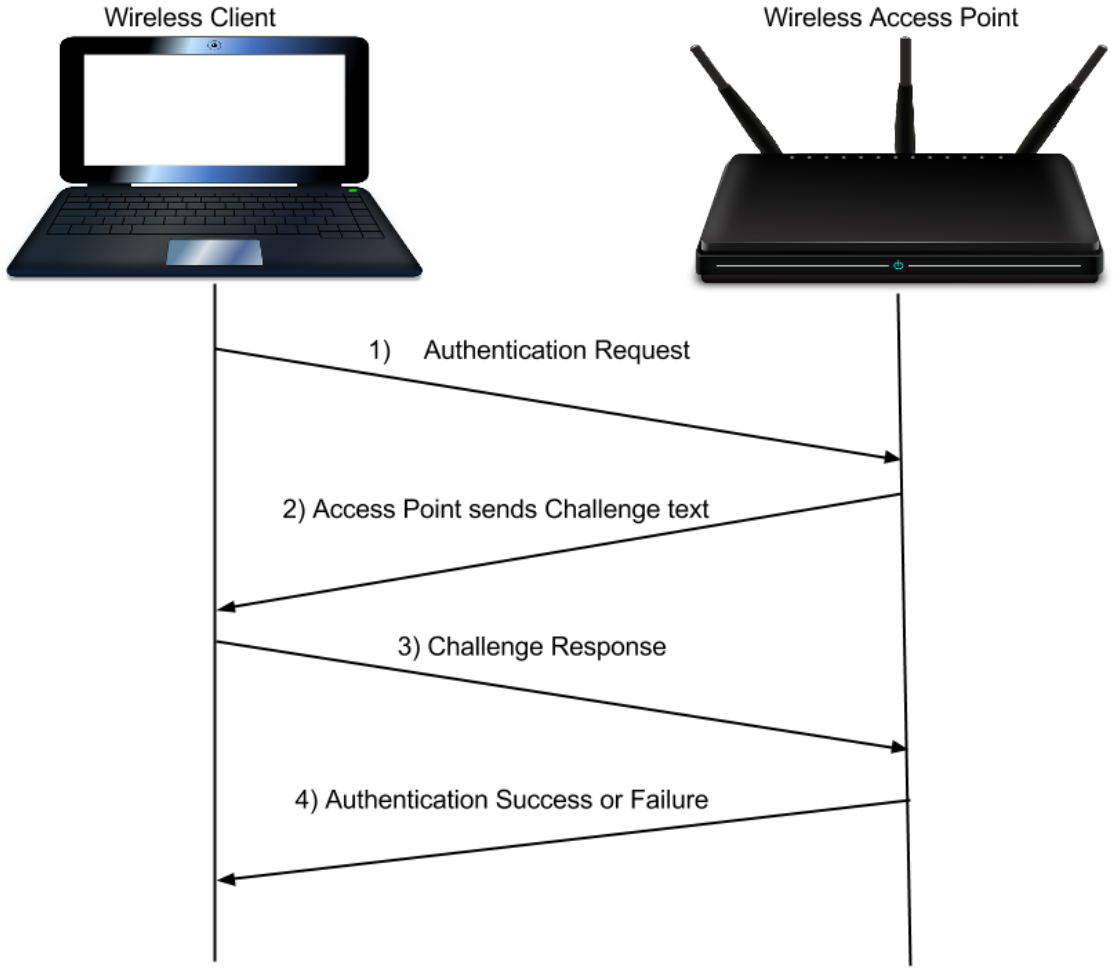
# Authentification – Principe

- Les entités partagent une clef secrète PA.

# Shared Key Authentification

- Défi (aléa) **R** d'une taille de 128 bit.
- On chiffre l'aléa avec la méthode précédente :

$$\mathbf{R} \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}).$$



# Faiblesses de RC4

- Attaque FMS de S. R. Fluhrer, I. Mantin, et A. Shamir (2001).
  - Recouvrement de clé avec 9.000.000 requêtes (probablité de succès $1/2$)
- Korek-FMS (2001)
  - Recouvrement de clé avec 700.000 requêtes (probablité de succès $1/2$ )
- Attaque de A. Klein (2005)
  - Recouvrement de clé avec 43.000 requêtes (probablité de succès $1/2$ )
- Attaque PTW de A. Pyshkin, E. Tews, et R.-P. Weinmann (2005)
  - Recouvrement de clé avec 35.000 requêtes (probablité de succès $1/2$ )
- Tornado attaque de P. Sepehrdad, S. Vaudenay, et M. Vuagnoux (2011)
- Ensemble des biais sur RC4 par N. J. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, Jacob C. N. Schuldt (2013).
- ...

# Faiblesses du protocole

- Attaque de W. A. Arbaugh (2001)
- Attaque Chochop de Korek (2004)
- Attaque par fragmentation de A. Bittau, M. Handley, et J. Lackey (2006).
- ...


https://www.aircrack-ng.org/

# Injection de paquet

## Rejeu

Possible de **réinjecter** une trame chiffrée dans le réseau.

- Aucun mécanisme prévu dans `WEP` contre le rejeu.

# Fausse authentification – SKA

## Authentification sans clé

- Dans la phase d'authentification :

$$\mathbf{R} \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}),$$

  avec un aléa **R** **publique** d'une taille de 128 bit.
- Nous pouvons **extraire** $\mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV})$.
- Pour tout challenge $\mathbf{R}'$, nous pouvons **construire un authentifiant valide** :

$$\mathbf{R}' \oplus \mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}).$$

# Modification de paquet

## Linéarité du CRC

$$\mathrm{ICV}(\mathbf{M}' \oplus \mathbf{M}) = \mathrm{ICV}(\mathbf{M}) \oplus \mathrm{ICV}(\mathbf{M}').$$

$$\begin{aligned}
&\mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}) \oplus \left(\mathbf{M} \oplus \mathbf{M}' \,||\, \mathrm{ICV}(\mathbf{M} \oplus \mathbf{M}')\right) \\
&\mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}) \oplus \left(\mathbf{M} \oplus \mathbf{M}' \,||\, \mathrm{ICV}(\mathbf{M}) \oplus \mathrm{ICV}(\mathbf{M}')\right) \\
&\mathrm{RC4}(\mathbf{K} \,||\, \mathrm{IV}) \oplus \left(\mathbf{M} \,||\, \mathrm{ICV}(\mathbf{M})\right) \oplus \left(\mathbf{M}' \,||\, \mathrm{ICV}(\mathbf{M}')\right)
\end{aligned}$$

$\Rightarrow$ Possibilité d'injecter des paquets modifiés dans le réseau.

# Attaque Chochop – (I)

- Dans $\mathrm{CRC32}, \mathbf{G}(X) =$

$$X^{32} \oplus X^{26} \oplus X^{23} \oplus X^{22} \oplus X^{16} \oplus X^{12} \oplus X^{11} \oplus X^{10} \oplus X^{8} \oplus X^{7} \oplus X^{5} \oplus X^{4} \oplus X^{2} \oplus X \oplus 1.$$

- $$\mathbf{M}' = \mathbf{M} \,||\, \mathrm{ICV}(\mathbf{M}), \text{ avec } \mathrm{ICV}(\mathbf{M}) = \mathrm{CRC32}(\mathbf{M}).$$
- Soit $P_{\mathrm{CHECK}}$ un polynôme de degré $< 32$ tel que :

$$\mathbf{M}'(X) \equiv P_{\mathrm{CHECK}} \bmod \mathbf{G}(X).$$

# Attaque Chochop – (II)

- Soit $\mathbf{P_7}(X)$ un polynôme de degré $< 8$ tel que :

$$\mathbf{M}'(X) = \mathbf{Q}(X) \cdot X^8 \oplus \mathbf{P_7}(X).$$

- Ainsi :

$$\mathbf{Q}(X) \cdot X^8 \equiv \mathbf{P_7}(X) \oplus P_{\mathrm{CHECK}} \bmod \mathbf{G}(X).$$

- Nous avons $\mathrm{pgcd}\left(X^8, \mathbf{G}(X)\right) = 1$.
- Il existe donc $\mathbf{R}_{\mathrm{Inv}} \equiv (X^8)^{-1} \bmod \mathbf{G}(X) =$

$$X^{31} \oplus X^{29} \oplus X^{27} \oplus X^{24} \oplus X^{23} \oplus X^{22} \oplus X^{20} \oplus X^{17} \oplus X^{16} \oplus X^{15} \oplus X^{14} \oplus X^{13} \oplus X^{10} \oplus X^9 \oplus X^7 \oplus X^5 \oplus X^2 \oplus X.$$

# Attaque Chochop – (III)

- Comme $\mathbf{Q}(X) \cdot X^8 \equiv \mathbf{P_7}(X) \oplus P_{\text{CHECK}} \bmod \mathbf{G}(X)$ :

$$\mathbf{Q} \equiv \mathbf{R}_{\text{Inv}} \cdot (\mathbf{P_7} \oplus P_{\text{CHECK}}) \bmod \mathbf{G}(X).$$

- Soit $\mathbf{P}_{\text{Cor}} = P_{\text{CHECK}} \oplus \mathbf{R}_{\text{Inv}} \cdot (\mathbf{P_7} \oplus P_{\text{CHECK}})$, alors :

$$\mathbf{Q}' = \mathbf{Q} \oplus \mathbf{P}_{\text{Cor}} \equiv P_{\text{CHECK}} \bmod \mathbf{G}(X).$$

- On suppose l'existence d'un oracle $\mathcal{O}_{\text{CRC}}$ qui retourne 1 si une trame est valide, et 0 sinon.
  - $\mathcal{O}_{\text{CRC}}(\mathbf{Q}') = 1$, nous avons retrouvé le dernier octet de $\mathbf{M}'$

# Comment construire $\mathcal{O}_{\mathrm{CRC}}$ ?

- L'attaquant possède deux stations $A$ et $B$ dans le réseau. Il transmet à $B$ via $A$ une trame $\mathbf{Q}'$. La trame est relayée à $B$ par l'AP ssi $\mathbf{Q}'$ est valide.
- L'attaquant possède une machine dans le réseau. Il transmet trame $\mathbf{Q}'$ à une adresse aléatoire. Un message d'erreur est retourné par l'AP si la trame est valide.

# Plan du cours

# Evolutions

Wi-Fi Protected Access (WPA et WPA2) [802.11i, 2004]

## Authentification

- Serveur d'identification 802.1X (Extensible Authentication Protocol, EAP) pour distribuer les clefs.
  - Mode dégradé pre-shared key (PSK)

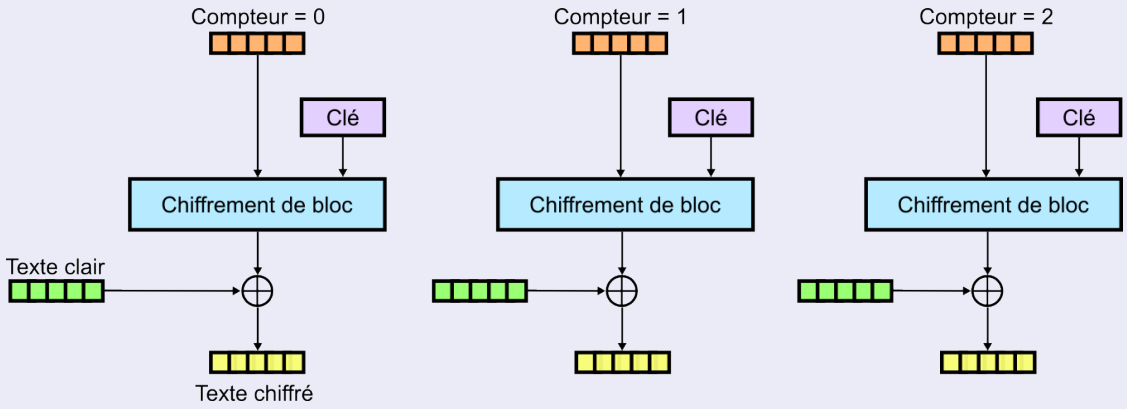
## Confidentialité

- Temporal Key Integrity Protocol (TKIP)
  - code d'authentification de message : message integrity code (MIC)
  - un compteur pour les vecteurs d'initialisation
  - gestion dynamique des clefs de chiffrement
- Utilisation de RC4 WPA dans (clef de 128 bits, IV de 48 bits)
- Utilisation de AES dans WPA2

# AES-CCMP

## Chiffrement et Authentification

- Mode CTR
- CBC-MAC




*Source :* `https://fr.wikipedia.org/`