Chap. VI : Systèmes de chiffrement par blocs itérés

Laurent Poinsot

1er octobre 2009

Plan

Plan

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Introduction

On a vu que l'on peut utiliser les statistiques sur les occurrences des lettres dans un message pour cryptanalyser. Si on veut éviter cela, il faut être capable de "masquer", de faire disparaître ces statistiques. Pour cela en pratique on emploie deux techniques :

1. On chiffre un **bloc** de lettres plutôt que lettre après lettre de façon à masquer les statistiques individuelles de chaque lettre à l'intérieur ;
2. On mélange les lettres dans le message afin de le rendre de plus en plus incompréhensible.

Pour mettre en œuvre ces techniques, on utilise les cryptosystèmes par blocs itérés.

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs

Un **procédé de chiffrement par blocs** est un cryptosystème à clef secrète avec lequel on chiffre des messages M qui sont des **blocs** de n bits (où n est un entier fixé tel que 128, 256, ... par exemple). Un tel cryptosystème produit des messages chiffrés C qui sont aussi des blocs de n bits. La clef secrète K quant à elle est un bloc de ℓ bits. Il est parfois possible que $\ell \neq n$. En conclusion les messages clairs M et chiffrés C sont des éléments de $\mathbb{Z}_2^n$ et les clefs secrètes sont des éléments de $\mathbb{Z}_2^\ell$. (Rappelons que $\mathbb{Z}_2 = \{0, 1\}$.)

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\ldots \\ C_r &:= E(C_{r-1})\ . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\dots \\ C_r &:= E(C_{r-1}) . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\dots \\ C_r &:= E(C_{r-1}) . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$
\begin{aligned}
C_1 &:= E(M);\\
C_2 &:= E(C_1);\\
&\ldots\\
C_r &:= E(C_{r-1})\ .
\end{aligned}
$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{array}{l} C_1 := E(M); \\ C_2 := E(C_1); \\ \dots \\ C_r := E(C_{r-1}) \, . \end{array}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\dots \\ C_r &:= E(C_{r-1})\ . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\dots \\ C_r &:= E(C_{r-1}) . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$
\begin{aligned}
C_1 &:= E(M); \\
C_2 &:= E(C_1); \\
&\dots \\
C_r &:= E(C_{r-1}).
\end{aligned}
$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\dots \\ C_r &:= E(C_{r-1}). \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{array}{l} C_1 := E(M); \\ C_2 := E(C_1); \\ \dots \\ C_r := E(C_{r-1}). \end{array}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.

De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{array}{l} C_1 := E(M); \\ C_2 := E(C_1); \\ \dots \\ C_r := E(C_{r-1})\ . \end{array}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{array}{l} C_1 := E(M); \\ C_2 := E(C_1); \\ \dots \\ C_r := E(C_{r-1}) . \end{array}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde**. Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{array}{l} C_1 := E(M); \\ C_2 := E(C_1); \\ \ldots \\ C_r := E(C_{r-1})\ . \end{array}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde.** Enfin C_r est le message chiffré correspondant à M.

Procédé de chiffrement par blocs itérés

Le principe de fonctionnement d'un procédé de chiffrement par blocs **itéré** est d'appliquer plusieurs fois (c'est-à-dire d'itérer) une même fonction sur le message clair afin d'obtenir le chiffré.
De façon schématique, si E est une certaine fonction et M le message clair, l'idée est de calculer successivement

$$\begin{aligned} C_1 &:= E(M); \\ C_2 &:= E(C_1); \\ &\ldots \\ C_r &:= E(C_{r-1}) . \end{aligned}$$

Dans un tel cryptosystème, r est le **nombre de tours** ou **nombre de rondes** et E s'appelle la **fonction de tour** ou **fonction de ronde.** Enfin C_r est le message chiffré correspondant à M.

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r) . \end{aligned}$$

k_i s'appelle la **sous-clef de la $i$ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$
\begin{aligned}
C_1 &:= E(M, k_1);\\
C_2 &:= E(C_1, k_2);\\
&\dots\\
C_r &:= E(C_{r-1}, k_r)\ .
\end{aligned}
$$

k_i s'appelle la **sous-clef de la $i$ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r). \end{aligned}$$

k_i s'appelle la **sous-clef de la $i$ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$
\begin{array}{l}
C_1 := E(M, k_1); \\
C_2 := E(C_1, k_2); \\
\dots \\
C_r := E(C_{r-1}, k_r) .
\end{array}
$$

k_i s'appelle la **sous-clef de la $i$ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{array}{l} C_1 := E(M, k_1); \\ C_2 := E(C_1, k_2); \\ \ldots \\ C_r := E(C_{r-1}, k_r)\ . \end{array}$$

k_i s'appelle la **sous-clef de la** i**ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r)\ . \end{aligned}$$

k_i s'appelle la **sous-clef de la** i**ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r)\ . \end{aligned}$$

k_i s'appelle la **sous-clef de la *i*ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r) . \end{aligned}$$

k_i s'appelle la **sous-clef de la $i$ème ronde.**

En pratique, la fonction de tour E dépend d'un autre paramètre k de telle sorte que l'on calcule successivement

$$\begin{aligned} C_1 &:= E(M, k_1); \\ C_2 &:= E(C_1, k_2); \\ &\dots \\ C_r &:= E(C_{r-1}, k_r)\ . \end{aligned}$$

k_i s'appelle la **sous-clef de la** i**ème ronde.**

En règle générale, chaque sous-clef de tour k_i est produite à partir de la clef secrète K, laquelle est également appelée **clef principale.** La procédure permettant de générer les sous-clefs à partir de K s'appelle un **algorithme de dérivation de sous-clef**.

En règle générale, chaque sous-clef de tour k_i est produite à partir de la clef secrète K, laquelle est également appelée **clef principale**. La procédure permettant de générer les sous-clefs à partir de K s'appelle un **algorithme de dérivation de sous-clef**.

En règle générale, chaque sous-clef de tour k_i est produite à partir de la clef secrète K, laquelle est également appelée **clef principale**. La procédure permettant de générer les sous-clefs à partir de K s'appelle un **algorithme de dérivation de sous-clef**.

En règle générale, chaque sous-clef de tour k_i est produite à partir de la clef secrète K, laquelle est également appelée **clef principale.** La procédure permettant de générer les sous-clefs à partir de K s'appelle un **algorithme de dérivation de sous-clef.**

En règle générale, chaque sous-clef de tour k_i est produite à partir de la clef secrète K, laquelle est également appelée **clef principale**. La procédure permettant de générer les sous-clefs à partir de K s'appelle un **algorithme de dérivation de sous-clef**.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Un **système de chiffrement par blocs itéré** est constitué de trois algorithmes :

1. L'**algorithme de chiffrement** proprement dit qui consiste principalement à itérer la fonction de tour ;
2. L'**algorithme de déchiffrement** ;
3. L'**algorithme de dérivation de sous-clef** qui produit la sous-clef du tour correspondant à partir de la clef principale.

Lorsque l'on décrit un tel procédé, il est indispensable de détailler les trois algorithmes qui le composent.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

	$K \in \mathbb{Z}_2^\ell$		$M \in \mathbb{Z}_2^n$
	$\downarrow$		$\downarrow$
ronde 1	*Der*	$\xrightarrow{k_1}$	$C_1 := E(M, k_1)$
	$\downarrow K$		$\downarrow$
ronde 2	*Der*	$\xrightarrow{k_2}$	$C_2 := E(C_1, k_2)$
	$\downarrow K$		$\downarrow$
	$\vdots$		$\vdots$
	$\downarrow K$		$\downarrow$
ronde r	*Der*	$\xrightarrow{k_r}$	$C_r := E(C_{r-1}, k_r)$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & \mathit{Der} & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & \mathit{Der} & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & \mathit{Der} & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & & C_1 := E(M, k_1) \\
 & \downarrow K & & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & & C_2 := E(C_1, k_2) \\
 & \downarrow K & & & \downarrow \\
 & \vdots & & & \vdots \\
 & \downarrow K & & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^\ell & & M \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde 1} & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde 2} & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & & C_1 := E(M, k_1) \\
 & \downarrow K & & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & & C_2 := E(C_1, k_2) \\
 & \downarrow K & & & \downarrow \\
 & \vdots & & & \vdots \\
 & \downarrow K & & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^\ell & & M \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$\begin{array}{llcl}
 & K \in \mathbb{Z}_2^\ell & & M \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \overset{k_1}{\longrightarrow} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \overset{k_2}{\longrightarrow} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \overset{k_r}{\longrightarrow} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.

Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Schéma de l'algorithme de chiffrement

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & M \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der & \xrightarrow{k_1} & C_1 := E(M, k_1) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der & \xrightarrow{k_2} & C_2 := E(C_1, k_2) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der & \xrightarrow{k_r} & C_r := E(C_{r-1}, k_r)
\end{array}
$$

$C := C_r$ est le message chiffré correspondant à M.
Légende : *Der* désigne l'algorithme de dérivation de sous-clef.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rccl} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rccc} E_{k_i} : & \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{cccc} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rccl} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rcccl} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n & \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \qquad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rcccl} E_{k_i} : & \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rcll} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \tag{1}$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.
Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rccl} E_{k_i} : & \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \tag{1}$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rccl} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$E_{k_i} : \begin{array}{ccc} \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \tag{1}$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$E_{k_i} : \begin{array}{ccc} \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \\ M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.
Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Algorithme de déchiffrement

Pour qu'il soit possible de déchiffrer, il faut que pour chaque sous-clef k_i fixée, la fonction suivante

$$\begin{array}{rcll} E_{k_i} : & \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \\ & M & \mapsto & E_{k_i}(M) := E(M, k_i) \end{array} \quad (1)$$

doit être inversible. Si tel est le cas, on note $E_{k_i}^{-1}$ sa fonction inverse.

Pour déchiffrer il suffit alors de faire tourner l'algorithme de chiffrement en **sens inverse** : on utilise le message chiffré C à la place du clair M, on remplace la fonction $E(., k_i)$ par $E_{k_i}^{-1}$ et on utilise les mêmes sous-clefs de ronde mais dans le sens inverse c'est-à-dire en partant de k_r jusqu'à k_1.

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E^{-1}_{k_r}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E^{-1}_{k_{r-1}}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E^{-1}_{k_1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{llcc}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & & \downarrow \\
 & \vdots & & & \vdots \\
 & \downarrow K & & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{lccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde 1} & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde 2} & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llll}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{llcl}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcl}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^{\ell} & & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & & \downarrow \\
 & \vdots & & & \vdots \\
 & \downarrow K & & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{lccc}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcr}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lcccc}
 & K \in \mathbb{Z}_2^\ell & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcr}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^n \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{llcc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

Schéma de l'algorithme de déchiffrement

Légende : $C := C_r$ est le message chiffré, avec la clef K, produit à partir du message clair M. Der^{-1} est l'algorithme de dérivation de sous-clef **en sens inverse**, c'est-à-dire que l'on commence par générer la sous-clef k_r, puis k_{r-1}, et ainsi de suite jusqu'à k_2 et enfin k_1.

$$
\begin{array}{lccc}
 & K \in \mathbb{Z}_2^{\ell} & & C \in \mathbb{Z}_2^{n} \\
 & \downarrow & & \downarrow \\
\text{ronde } 1 & Der^{-1} & \xrightarrow{k_r} & C_{r-1} := E_{k_r}^{-1}(C) \\
 & \downarrow K & & \downarrow \\
\text{ronde } 2 & Der^{-1} & \xrightarrow{k_{r-1}} & C_{r-2} := E_{k_{r-1}}^{-1}(C_{r-1}) \\
 & \downarrow K & & \downarrow \\
 & \vdots & & \vdots \\
 & \downarrow K & & \downarrow \\
\text{ronde } r & Der^{-1} & \xrightarrow{k_1} & M := E_{k_1}^{-1}(C_1)
\end{array}
$$

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

L'analyse de l'algorithme de déchiffrement nous permet d'affirmer qu'un procédé de chiffrement par blocs itérés est en fait caractérisé par essentiellement deux algorithmes : l'algorithme de chiffrement et l'algorithme de dérivation des sous-clefs de tour. En effet, à partir de ce dernier, on retrouve facilement Der^{-1} qui nous permet de définir l'algorithme de déchiffrement. En bref, l'algorithme de déchiffrement n'est rien d'autre que "*l'algorithme de chiffrement que l'on fait tourner en sens inverse*".

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Les **caractéristiques** d'un cryptosystème par blocs itéré sont

1. La **longueur** n des messages clairs et chiffrés ;
2. La **longueur** de la clef secrète principale ;
3. La **longueur** ℓ des sous-clefs de tour employées ;
4. Le **nombre de rondes** r ;

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) =$ $(1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2.
Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1,0,0,1,1) \oplus (1,1,0,0,1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0,1,0,1,0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) =$ $(1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1,0,0,1,1) \oplus (1,1,0,0,1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0,1,0,1,0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2.

Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) =$
$(1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2.
Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2.
Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) =$
$(1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2.
Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) =$
$(1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (1)

Si $(x_1, \ldots, x_n) \in \mathbb{Z}_2^n$ et $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$ sont deux blocs de n bits, on définit l'**addition modulo 2 composante par composante** ou **XOR** ou **ou-exclusif**, notée $\oplus$, par

$$(x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n) := (x_1 \oplus y_1, \ldots, x_n \oplus y_n)$$

où le $\oplus$ entre des bits est donné par la table suivante :

$\oplus$	0	1
0	0	1
1	1	0

$\oplus$ entre les bits n'est rien d'autre que l'addition modulo 2. Par exemple, $(1, 0, 0, 1, 1) \oplus (1, 1, 0, 0, 1) = (1 \oplus 1, 0 \oplus 1, 0 \oplus 0, 1 \oplus 0, 1 \oplus 1) = (0, 1, 0, 1, 0)$.

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1,\ldots,x_n)\oplus(x_1,\ldots,x_n)=\underbrace{(0,0,\ldots,0)}_{n\ \text{termes}}\ .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x\in\{0,1\}$, $x\oplus x=0$. Or on a

$$(x_1,\ldots,x_n)\oplus(x_1,\ldots,x_n)=(x_1\oplus x_1,\ldots,x_n\oplus x_n)\ .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1,\ldots,x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1,\ldots,y_n)\in\mathbb{Z}_2^n$,

$$((x_1,\ldots,x_n)\oplus(y_1,\ldots,y_n))\oplus(x_1,\ldots,x_n)=(y_1,\ldots,y_n)\ .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.

Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.

Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \ \textit{termes}} \ .$$

Démontrer-le.

Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) \ .$$

D'où le résultat.

Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) \ .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \ \textit{termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Rappels sur le ou-exclusif (2)

Le ou-exclusif composante par composante possède la propriété intéressante suivante :

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = \underbrace{(0, 0, \ldots, 0)}_{n \text{ termes}} .$$

Démontrer-le.
Il est clair (d'après la table de $\oplus$) que quel que soit $x \in \{0, 1\}$, $x \oplus x = 0$. Or on a

$$(x_1, \ldots, x_n) \oplus (x_1, \ldots, x_n) = (x_1 \oplus x_1, \ldots, x_n \oplus x_n) .$$

D'où le résultat.
Cette propriété implique d'une part que $(x_1, \ldots, x_n)$ est son propre opposé, et d'autre part, quel que soit $(y_1, \ldots, y_n) \in \mathbb{Z}_2^n$,

$$((x_1, \ldots, x_n) \oplus (y_1, \ldots, y_n)) \oplus (x_1, \ldots, x_n) = (y_1, \ldots, y_n) .$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccc} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \,. \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccl} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \, . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{llll} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccc} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \,. \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccl} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccc} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{llll} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccc} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) . \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccl} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \,. \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.

Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{rccl} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \,. \end{array}$$

Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ une application quelconque (en particulier il n'est pas nécessaire qu'elle soit inversible... mais en fait le pourrait-elle ?). Cette fonction prend en arguments deux blocs de n bits chacun et produit un résultat codé sur n bits également.
Soient G et D deux blocs de n bits chacun (de sorte que $(G, D) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$). Soit $K \in \mathbb{Z}_2^n$. On définit la **structure de Feistel** par $\mathcal{F}((G, D), K) := (D, G \oplus f(D, K))$. C'est donc la fonction suivante

$$\begin{array}{llll} \mathcal{F} : & (\mathbb{Z}_2^n \times \mathbb{Z}_2^n) \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & ((G, D), K) & \mapsto & (D, G \oplus f(D, K)) \ . \end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc}
G & & & & D \\
& \searrow & & \swarrow & \downarrow \\
& & \times & & f \leftarrow K \\
& \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{tikzcd}
G \arrow[dr] & & D \arrow[dl] \arrow[d] & \\
& \times \arrow[ddl] \arrow[dr] & f & K \arrow[l] \\
& & \oplus \arrow[d] & \\
D & & G \oplus f(D,K) &
\end{tikzcd}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D, K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc} G & & & & D \\ & \searrow & & \swarrow & \downarrow \\ & & \times & & f \leftarrow K \\ & \swarrow & & \searrow & \downarrow \\ \downarrow & & & & \oplus \\ \downarrow & & & & \downarrow \\ D & & & & G \oplus f(D,K) \end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
& \searrow & & \swarrow & \downarrow \\
& & \times & & f \leftarrow K \\
& \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
& \searrow & & \swarrow & \downarrow \\
& & \times & & f \leftarrow K \\
& \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D, K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D & & \\
 & \searrow & & \swarrow & \downarrow & & \\
 & & \times & & f & \leftarrow & K \\
 & \swarrow & & \searrow & \downarrow & & \\
\downarrow & & & & \oplus & & \\
\downarrow & & & & \downarrow & & \\
D & & & & G \oplus f(D, K) & &
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccccc}
G & & & & D & & \\
 & \searrow & & \swarrow & \downarrow & & \\
 & & \times & & f & \leftarrow & K \\
 & \swarrow & & \searrow & \downarrow & & \\
\downarrow & & & & \oplus & & \\
\downarrow & & & & \downarrow & & \\
D & & & & G \oplus f(D,K) & &
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{tikzcd}
G \arrow[dr] & & D \arrow[dl] \arrow[d] & \\
& \times \arrow[dl] \arrow[dr] & f \arrow[d] & K \arrow[l] \\
{} \arrow[d] & & \oplus \arrow[d] & \\
D & & G \oplus f(D,K) &
\end{tikzcd}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccccc}
G & & & & D & & \\
 & \searrow & & \swarrow & \downarrow & & \\
 & & \times & & f & \leftarrow & K \\
 & \swarrow & & \searrow & \downarrow & & \\
\downarrow & & & & \oplus & & \\
\downarrow & & & & \downarrow & & \\
D & & & & G \oplus f(D,K) & &
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc} G & & & & D \\ & \searrow & & \swarrow & \downarrow \\ & & \times & & f \leftarrow K \\ & \swarrow & & \searrow & \downarrow \\ \downarrow & & & & \oplus \\ \downarrow & & & & \downarrow \\ D & & & & G \oplus f(D,K) \end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccccc}
G & & & & D & & \\
& \searrow & & \swarrow & \downarrow & & \\
& & \times & & f & \leftarrow & K \\
& \swarrow & & \searrow & \downarrow & & \\
\downarrow & & & & \oplus & & \\
\downarrow & & & & \downarrow & & \\
D & & & & G \oplus f(D,K) & &
\end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{ccccc}
G & & & & D \\
 & \searrow & & \swarrow & \downarrow \\
 & & \times & & f \leftarrow K \\
 & \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}
$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$\begin{array}{ccccc}
G & & & & D \\
& \searrow & & \swarrow & \downarrow \\
& & \times & & f \leftarrow K \\
& \swarrow & & \searrow & \downarrow \\
\downarrow & & & & \oplus \\
\downarrow & & & & \downarrow \\
D & & & & G \oplus f(D,K)
\end{array}$$

On peut représenter schématiquement $\mathcal{F}$ par :

$$
\begin{array}{cccccc}
G & & & & D & \\
& \searrow & & \swarrow & \downarrow & \\
& & \times & & f & \leftarrow K \\
& \swarrow & & \searrow & \downarrow & \\
\downarrow & & & & \oplus & \\
\downarrow & & & & \downarrow & \\
D & & & & G \oplus f(D,K) &
\end{array}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$\begin{array}{l}(G_1, D_1) := \mathcal{F}((G, D), k_1);\\(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2);\\\ldots\\(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .\end{array}$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$\begin{array}{l}(G_1, D_1) := \mathcal{F}((G, D), k_1);\\ (G_2, D_2) := \mathcal{F}((G_1, D_1), k_2);\\ \ldots\\ (G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .\end{array}$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{array}{l}
(G_1, D_1) := \mathcal{F}((G, D), k_1); \\
(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2); \\
\ldots \\
(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .
\end{array}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{array}{l}
(G_1, D_1) := \mathcal{F}((G, D), k_1);\\
(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2);\\
\ldots\\
(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .
\end{array}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$\begin{array}{l}(G_1, D_1) := \mathcal{F}((G, D), k_1);\\ (G_2, D_2) := \mathcal{F}((G_1, D_1), k_2);\\ \ldots\\ (G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .\end{array}$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{array}{l}
(G_1, D_1) := \mathcal{F}((G, D), k_1); \\
(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2); \\
\dots \\
(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r) .
\end{array}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{aligned}
(G_1, D_1) &:= \mathcal{F}((G, D), k_1);\\
(G_2, D_2) &:= \mathcal{F}((G_1, D_1), k_2);\\
&\dots\\
(G_r, D_r) &:= \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .
\end{aligned}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{aligned}
(G_1, D_1) &:= \mathcal{F}((G, D), k_1);\\
(G_2, D_2) &:= \mathcal{F}((G_1, D_1), k_2);\\
&\ldots\\
(G_r, D_r) &:= \mathcal{F}((G_{r-1}, D_{r-1}), k_r)\ .
\end{aligned}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{array}{l}
(G_1, D_1) := \mathcal{F}((G, D), k_1); \\
(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2); \\
\ldots \\
(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r) .
\end{array}
$$

On souhaite utiliser $\mathcal{F}$ comme une fonction de tour dans un algorithme de chiffrement par blocs itéré. On suppose que la paire (G, D) va jouer le rôle du message M et K va jouer le rôle d'une sous-clef de tour. En d'autres termes, si on effectue r itérations, on devra successivement calculer :

$$
\begin{array}{l}
(G_1, D_1) := \mathcal{F}((G, D), k_1); \\
(G_2, D_2) := \mathcal{F}((G_1, D_1), k_2); \\
\ldots \\
(G_r, D_r) := \mathcal{F}((G_{r-1}, D_{r-1}), k_r) \; .
\end{array}
$$

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccc} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rrcl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?

Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !

Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rcll} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rcccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \rightarrow & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccl} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Mais pour que la structure de Feistel $\mathcal{F}$ soit une fonction de tour, que doit-elle vérifier ?
Il faut que quelle que soit la sous-clef de tour k_i, la fonction

$$\begin{array}{rccc} \mathcal{F}_{k_i} : & \mathbb{Z}_2^n \times \mathbb{Z}_2^n & \to & \mathbb{Z}_2^n \times \mathbb{Z}_2^n \\ & (G, D) & \mapsto & \mathcal{F}_{k_i}(G, D) := \mathcal{F}((G, D), k_i) \end{array}$$

soit inversible. Sans cela on n'a aucune chance de déchiffrer les messages !
Démontrer que $\mathcal{F}_{k_i}$ est inversible. (Astuce : Étant donnés k_i, $(G', D') := (D, G \oplus f(D, k_i))$ et, bien entendu, la fonction f, montrer que l'on peut reconstruire "algorithmiquement" (G, D).)

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Réponse : Supposons donnés la sous-clef k_i et $(G', D') := (D, G \oplus f(D, k_i))$. Alors il est clair que l'on retrouve facilement D, puisque $G' = D$. Il reste à reconstruire G à partir de la connaissance de D, de k_i et de $G \oplus f(D, k_i)$. Puisque l'on connaît D et k_i, on est en mesure de calculer $f(D, k_i)$. Puis on calcule $D' \oplus f(D, k_i) = (G \oplus f(D, k_i)) \oplus f(D, k_i) = G$. On a donc récupérer D et G, ce qui nous permet de reconstituer le couple (G, D). On en conclue que $\mathcal{F}$ est inversible, et donc peut être utiliser comme fonction de tour dans un algorithme de chiffrement par blocs itéré.

Question : Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. Quelles sont les caractéristiques d'un procédé de chiffrement utilisant comme fonction de tour une structure de Feistel ?

Question : Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$. Quelles sont les caractéristiques d'un procédé de chiffrement utilisant comme fonction de tour une structure de Feistel ?

Question : Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$. Quelles sont les caractéristiques d'un procédé de chiffrement utilisant comme fonction de tour une structure de Feistel ?

Question : Soit $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$. Quelles sont les caractéristiques d'un procédé de chiffrement utilisant comme fonction de tour une structure de Feistel ?

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

Réponse :

- (G, D) joue le rôle d'un message. Comme f prend (D, k_i) comme arguments, on en déduit que la taille de D est égale à n. Pour la même raison, la taille des sous-clefs est n également ;
- On effectue un ou-exclusif entre G et $f(D, k_i)$. Pour que cela soit possible il est nécessaire que G et $f(D, k_i)$ aient la même taille. Or par définition de f, $f(D, k_i)$ est de taille n. Donc G lui-même est de taille n. Au final la taille des messages clairs est la taille de (G, D) qui est égale à $2n$;
- Le résultat de $\mathcal{F}((G, D), k_i) = (D, G \oplus f(D, k_i))$. Or on sait que la taille de D est n et la taille de $G \oplus f(D, k_i)$ est n aussi. Les messages chiffrés sont donc de taille $2n$.

- Taille de la clef principale ? Impossible de le savoir car on n'a pas spécifié l'algorithme de dérivation des sous-clefs ;
- Nombre de rondes ? Impossible de le savoir.

- Taille de la clef principale ? Impossible de le savoir car on n'a pas spécifié l'algorithme de dérivation des sous-clefs ;
- Nombre de rondes ? Impossible de le savoir.

- Taille de la clef principale ? Impossible de le savoir car on n'a pas spécifié l'algorithme de dérivation des sous-clefs ;
- Nombre de rondes ? Impossible de le savoir.

- Taille de la clef principale ? Impossible de le savoir car on n'a pas spécifié l'algorithme de dérivation des sous-clefs ;
- Nombre de rondes ? Impossible de le savoir.

- Taille de la clef principale ? Impossible de le savoir car on n'a pas spécifié l'algorithme de dérivation des sous-clefs ;
- Nombre de rondes ? Impossible de le savoir.