

Can Symmetric Cryptography Be Liberated from the Von Neumann Style?

Jean-Baptiste Menant¹ Yves Ndiaye²

Pierre-Évariste Dagand³

¹ENS de Lyon

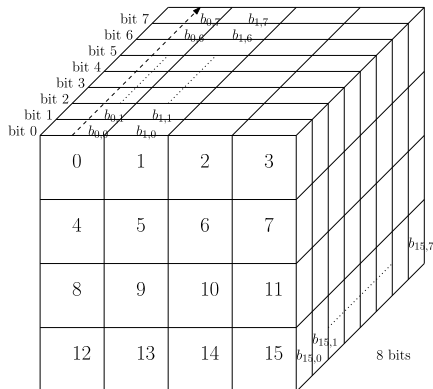
²Université Paris Cité, IRIF

³CNRS

Structures fonctorielles

Class Functor F :=

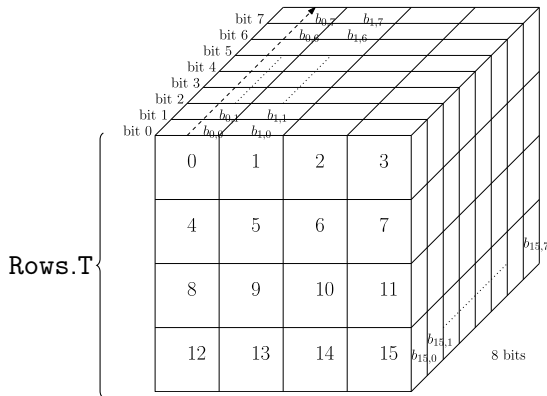
{ map: $\forall A B, (A \rightarrow B) \rightarrow F A \rightarrow F B$ }.



Structures fonctorielles

Class Functor $F :=$

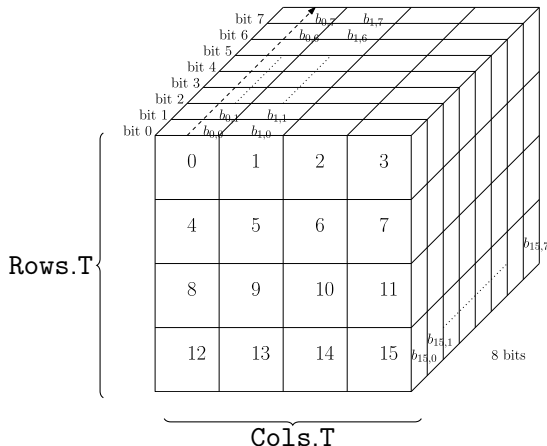
$\{ \text{map}: \forall A B, (A \rightarrow B) \rightarrow F A \rightarrow F B \}.$



Structures fonctorielles

Class Functor $F :=$

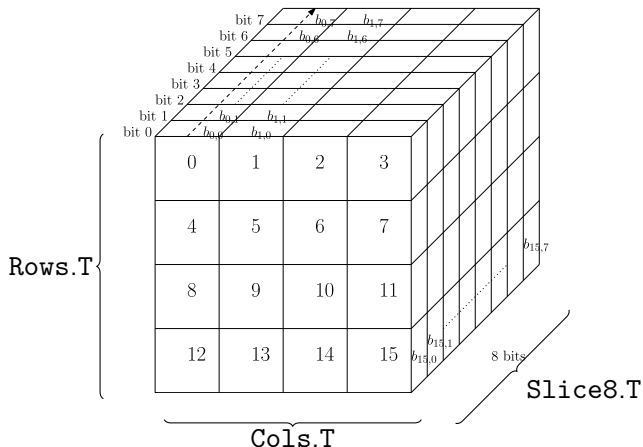
$\{ \text{map}: \forall A B, (A \rightarrow B) \rightarrow F A \rightarrow F B \}.$



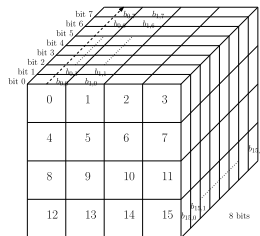
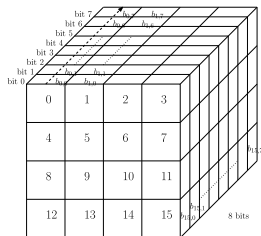
Structures fonctorielles

Class Functor F :=

{ map: $\forall A B, (A \rightarrow B) \rightarrow F A \rightarrow F B$ }.

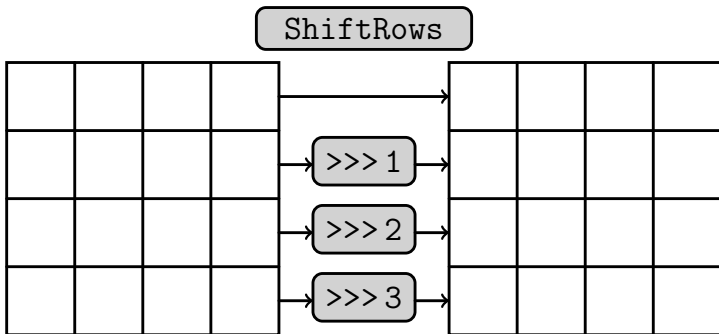


Add round key

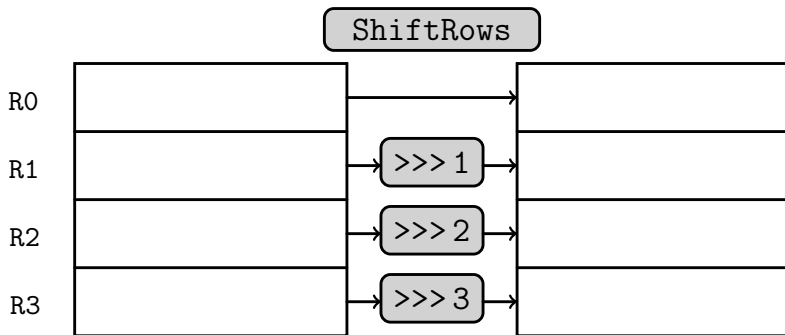


```
Class Applicative {F} '(Functor F) :=  
  { pure: ∀ A, A → F A  
    ; app: ∀ A B, F (A → B) → F A → F B }.
```

Shiftrows



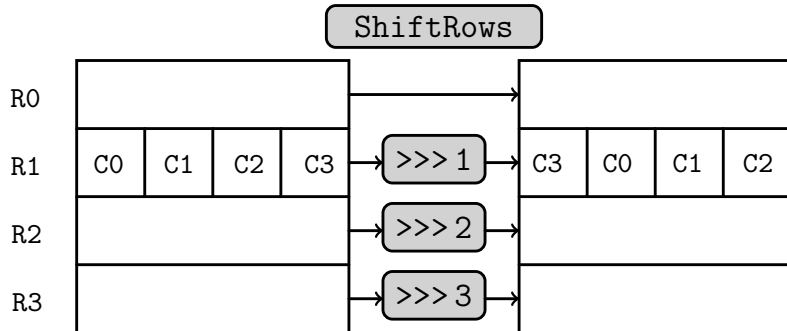
Shiftrows



Néperien

```
Class Naperian {F} '(Functor F) Ix :=  
  { lookup:  $\forall A, F A \rightarrow Ix \rightarrow A$   
    ; init:  $\forall A, (Ix \rightarrow A) \rightarrow F A$  }.
```

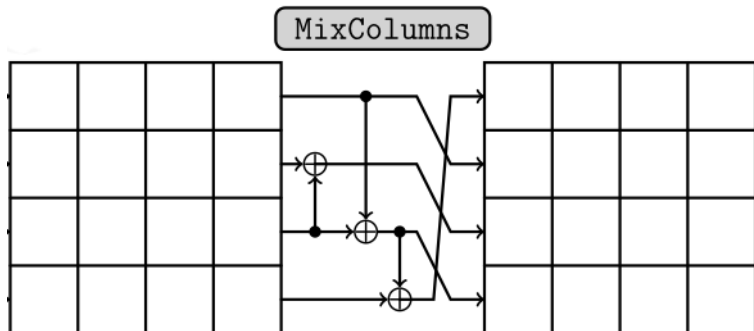
Shiftrows



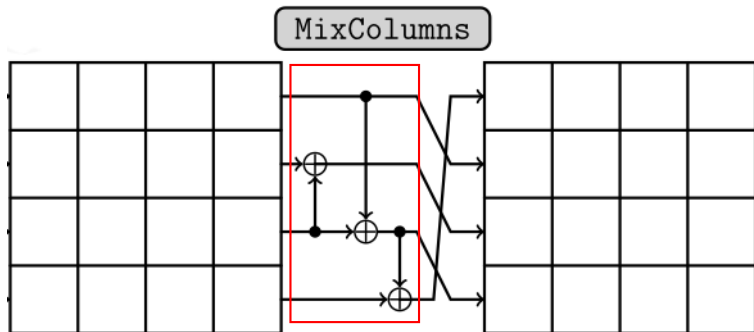
Circulant

```
Class Circulant {F} '(Functor F) :=  
  { circulant:  $\forall A, F A \rightarrow F (F A)$   
    ; anticirculant:  $\forall A, F A \rightarrow F (F A)$  }.
```

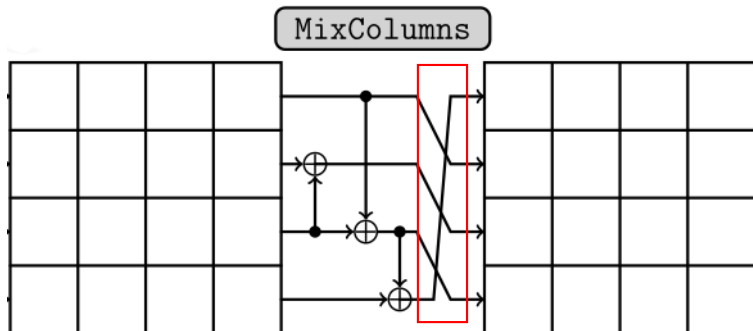
Mixcolumns



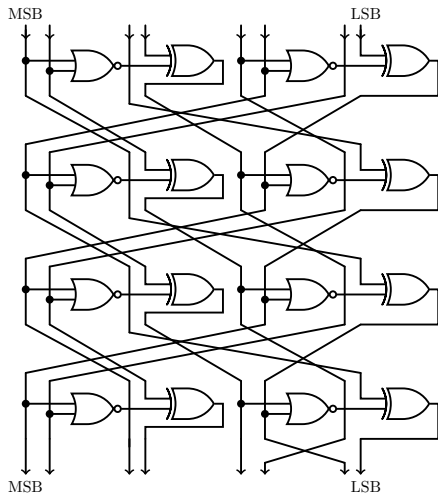
Mixcolumns



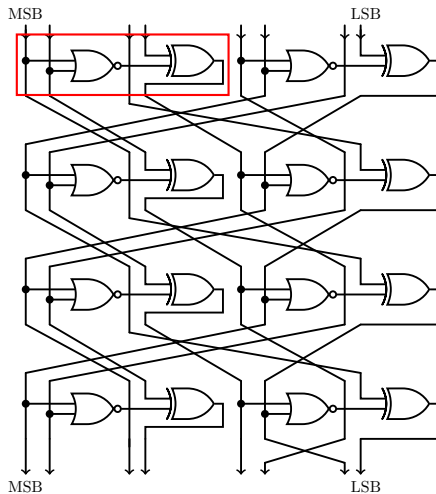
Mixcolumns



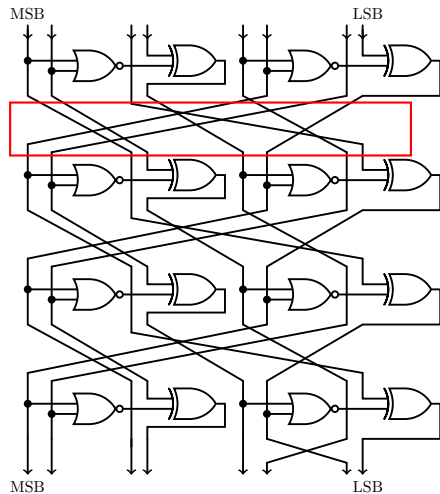
Subcells



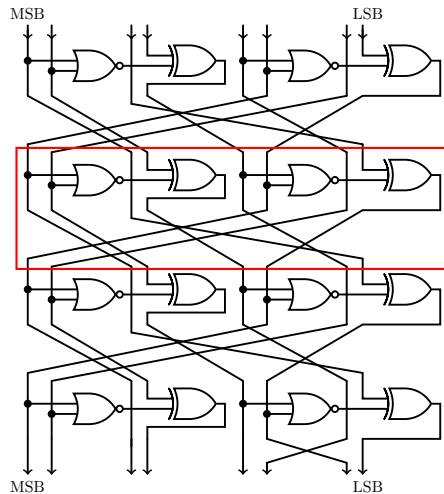
Subcells



Subcells



Subcells



Round complet

```
Class Boolean (T: Type) := { tt: T ; ff: T; ...}
```

Etapas

- `subcells :  $\forall B: \text{Boolean}, \text{Slice8.T } B \rightarrow \text{Slice8.T } B$`
- `add_round_key :  $\forall B: \text{Boolean}, B \rightarrow B \rightarrow B$`
- `shiftrows :  $\forall A, \text{Rows.T } (\text{Cols.T } A \rightarrow \text{Cols.T } A)$`
- `mixcolumns :  $\forall B: \text{Boolean}, \text{Rows.T } B \rightarrow \text{Rows.T } B$`

Composition

```
state := ?
```

Round complet

```
Class Boolean (T: Type) := { tt: T ; ff: T; ...}
```

Etapas

- `subcells :  $\forall B: \text{Boolean}, \text{Slice8.T } B \rightarrow \text{Slice8.T } B$`
- `add_round_key :  $\forall B: \text{Boolean}, B \rightarrow B \rightarrow B$`
- `shiftrows :  $\forall A, \text{Rows.T } (\text{Cols.T } A \rightarrow \text{Cols.T } A)$`
- `mixcolumns :  $\forall B: \text{Boolean}, \text{Rows.T } B \rightarrow \text{Rows.T } B$`

Composition

```
state := Rows.T (Cols.T (
```

Round complet

```
Class Boolean (T: Type) := { tt: T ; ff: T; ...}
```

Etapas

- `subcells :  $\forall B: \text{Boolean}, \text{Slice8.T } B \rightarrow \text{Slice8.T } B$`
- `add_round_key :  $\forall B: \text{Boolean}, B \rightarrow B \rightarrow B$`
- `shiftrows :  $\forall A, \text{Rows.T } (\text{Cols.T } A \rightarrow \text{Cols.T } A)$`
- `mixcolumns :  $\forall B: \text{Boolean}, \text{Rows.T } B \rightarrow \text{Rows.T } B$`

Composition

```
state := Rows.T (Cols.T ( Slice8.T
```

Round complet

```
Class Boolean (T: Type) := { tt: T ; ff: T; ...}
```

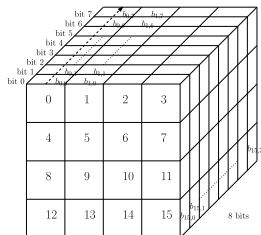
Etapas

- `subcells :  $\forall B: \text{Boolean}, \text{Slice8.T } B \rightarrow \text{Slice8.T } B$`
- `add_round_key :  $\forall B: \text{Boolean}, B \rightarrow B \rightarrow B$`
- `shiftrows :  $\forall A, \text{Rows.T } (\text{Cols.T } A \rightarrow \text{Cols.T } A)$`
- `mixcolumns :  $\forall B: \text{Boolean}, \text{Rows.T } B \rightarrow \text{Rows.T } B$`

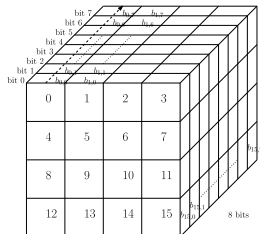
Composition

```
state := Rows.T (Cols.T ( Slice8.T bool))
```

Etat en mémoire



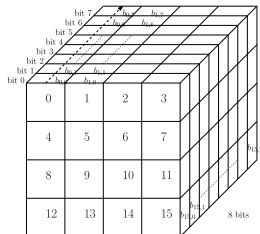
Etat en mémoire



Solution : Bitslicing

```
state := Rows.T( Cols.T (Slice8.T bool))
```

Etat en mémoire



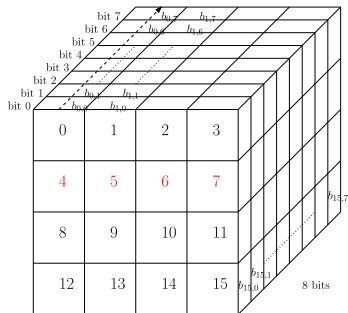
Solution : Bitslicing

```
state := Rows.T( Cols.T (Slice8.T bool))  
state2 := Slice8.T( Rows.T (Cols.T bool))  
           8 registres      16 bits
```

Remplissage optimal

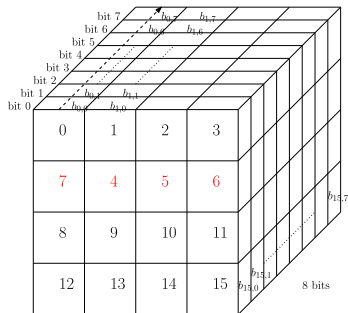
```
state * state
:= Slice8.T (Rows.T (Cols.T bool)) *
   Slice8.T (Rows.T (Cols.T bool))
= Slice8.T (Rows.T (Cols.T bool) *
              Rows.T (Cols.T bool))
= Slice8.T (Rows.T (Cols.T (bool*bool)))
```

Bitslicing : le surcoût



3	7	11	15	2	6	10	14	1	5	9	13	0	4	8	12
---	---	----	----	---	---	----	----	---	---	---	----	---	---	---	----

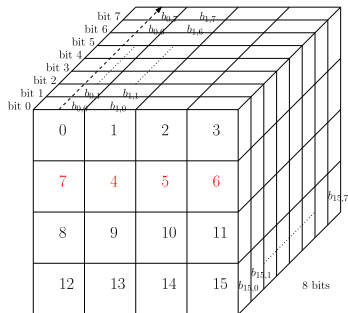
Bitslicing : le surcoût



3	7	11	15	2	6	10	14	1	5	9	13	0	4	8	12
---	---	----	----	---	---	----	----	---	---	---	----	---	---	---	----

	7				6				5				4		
--	---	--	--	--	---	--	--	--	---	--	--	--	---	--	--

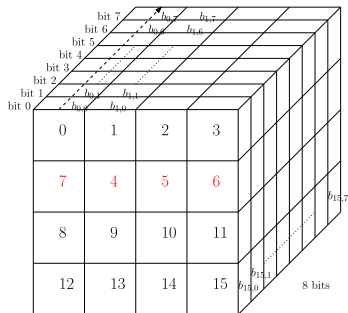
Bitslicing : le surcoût



3	0	11	15	2	0	10	14	1	0	9	13	0	0	8	12
---	---	----	----	---	---	----	----	---	---	---	----	---	---	---	----

	7				6				5				4		
--	---	--	--	--	---	--	--	--	---	--	--	--	---	--	--

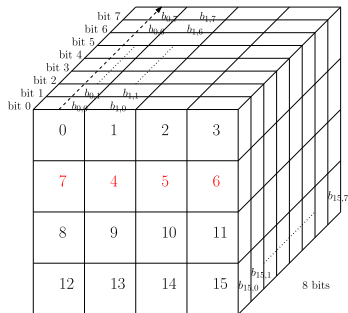
Bitslicing : le surcoût



3	0	11	15	2	0	10	14	1	0	9	13	0	0	8	12
---	---	----	----	---	---	----	----	---	---	---	----	---	---	---	----

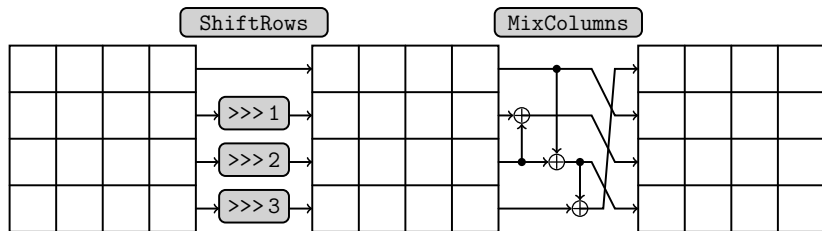
	6				5				4				7		
--	---	--	--	--	---	--	--	--	---	--	--	--	---	--	--

Bitslicing : le surcoût

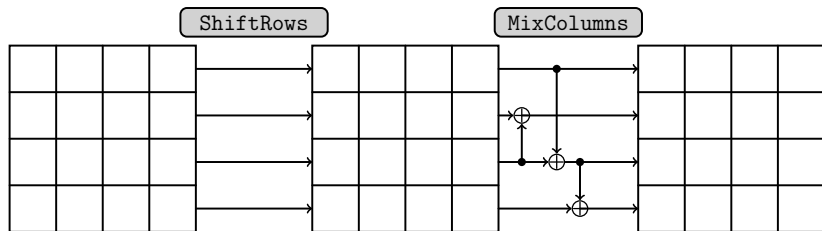


3	6	11	15	2	5	10	14	1	4	9	13	0	7	8	12
---	---	----	----	---	---	----	----	---	---	---	----	---	---	---	----

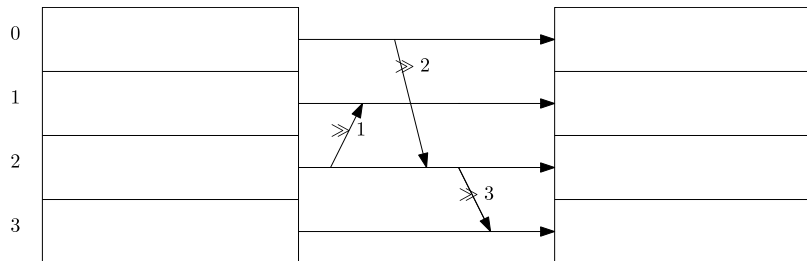
Fixslicing



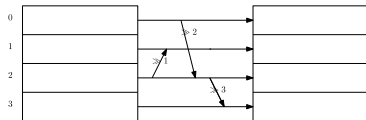
Fixslicing



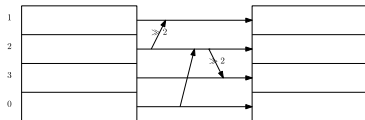
Fixslicing



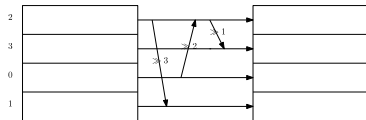
Fixslicing



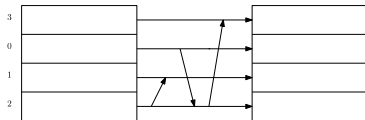
(a) mod 0



(b) mod 1



(c) mod 2



(d) mod 3

Dans ce travail :

- Expressivité des structures fonctorielles
- Une description pour trois implémentations
- Autre cas d'étude : GIFT (cf Yves)

Dans le futur :

- Structure ad-hoc : foncteur Circulant
- Compiler vers une vraie implémentation

Rocq : <https://github.com/pedagand/bitfix>