

Design Philosophy of Rijndael

Vincent Rijmen

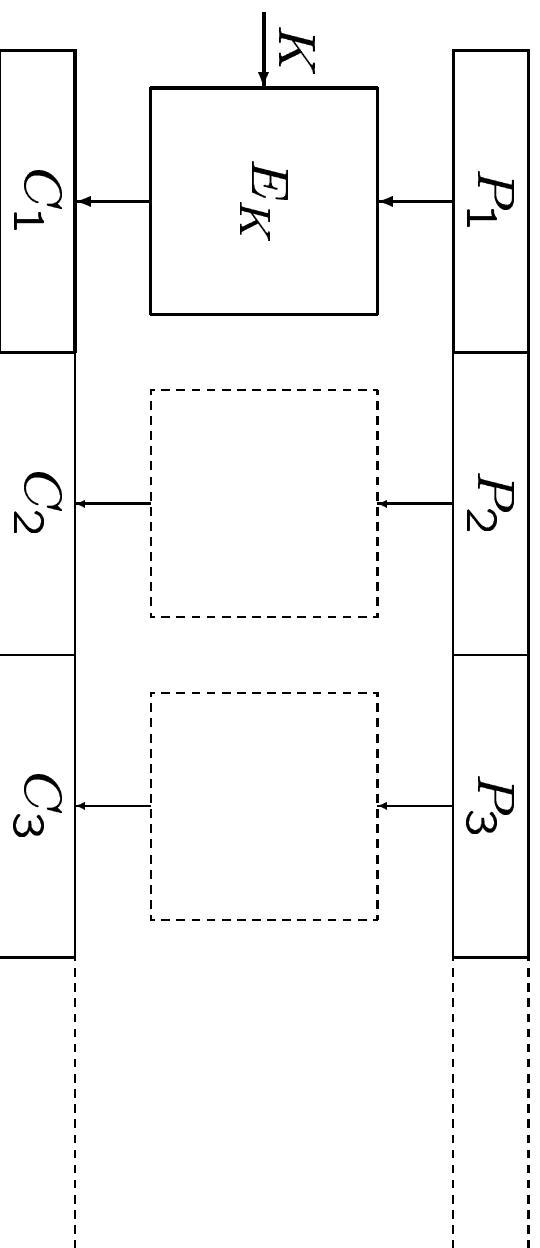
Cryptomathic, Belgium

Graz University of Technology, Austria

Motivations

1. Resistance against attacks:
differential, linear cryptanalysis
2. Efficiently implementable
3. Simplicity

Block Cipher Model



Transformation E_K takes one block at a time,
is identical for all blocks

Every K selects a different transformation

Design Trade-off

Security



cost

definition

resources

attack model

bandwidth

mathematical requirements

practical requirements

on transformations

on algorithm

Taxonomy of Security Definitions

If I use a word, it means what I choose it to mean.

(The queen in *Alice in Wonderland*)

1. Information theoretic: unconditional security
The Vernam scheme (1917) is the best we can do.
2. Complexity theoretic: reduce security to hard problems
3. Probabilistic: unconditional, probability of failure
4. Cryptanalytical: secure against state-of-the-art crypt-analysis

Practical Security: the Theory

Given $(P_1, C_1), (P_2, C_2), (P_3, C_3), \dots$, it should be 'difficult'

1. to recover the used key
2. to predict other (P_i, C_i) pairs
3. to characterize the family of 2^k permutations, or distinguish them from the other permutations

Important design criteria (Shannon)

1. Nonlinearity (in plaintext and in key input) ('confusion')
2. Spreading of statistics over all bits ('diffusion')

Practical Constraints

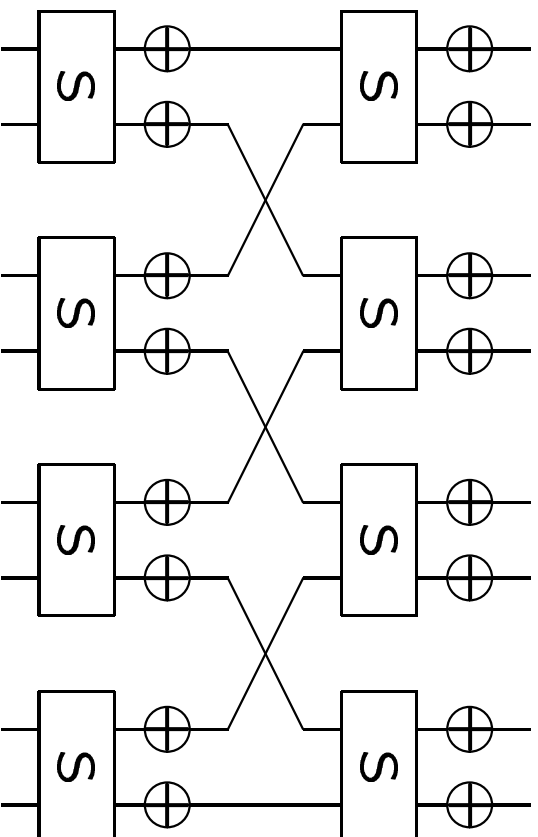
Block cipher should be realizable
in hardware and/or software

It should be possible to change the key 'easily'

⇒ a designer has to define an *algorithm*

- that implements the transformation,
- using relatively small components,
- with a limited number of input bits, output bits

Iterated Block Ciphers



Small code size (?), hardware area (!)

Iteration of a (weak) round transformation can make a strong cipher

Modern Cryptanalysis

Differential cryptanalysis: Propagation of differences

$$\begin{array}{l} P_1 \quad \rightarrow \quad C_1 \\ P_1 + \delta \quad \rightarrow \quad C_1 + \epsilon \end{array}$$

Choose texts with small δ

Try to collect samples with small ϵ

Probabilistic attack

Linear cryptanalysis: exploit input-output correlations

Estimation of data complexity

Propagation of Differences

Linear transformations: $L(X + \delta) = L(X) + L(\delta)$

$$\begin{aligned}\epsilon &= Y_2 - Y_1 = L(X_2) - L(X_1) = L(X_2 - X_1) \\ &= L(\delta)\end{aligned}$$

Example:

$$\begin{cases} y_1 = x_1 + x_3 \\ y_2 = x_2 + x_3 \end{cases}$$

$$\delta = (0, 0, 1) \rightarrow \epsilon = (1, 1);$$

$$\delta = (1, 1, 1) \rightarrow \epsilon = (0, 0)$$

Propagation of Differences (2)

Linear/affine components: δ determines ϵ

The pattern, or *trail* is fixed.

Probability ($\delta \rightarrow \epsilon$) = 1 or 0

$P(\delta = (0, 0, 1) \rightarrow \epsilon = (1, 1)) = 1$

$P(\delta = (0, 0, 1) \rightarrow \epsilon = (1, 0)) = 0$

...

Propagation of Differences (3)

nonlinear operators: $\epsilon = S(X_2) - S(X_1)$ depends on the values of X_1 and X_2

Several (δ, ϵ) tuples with $0 < P(\delta \rightarrow \epsilon) < 1$

Example:

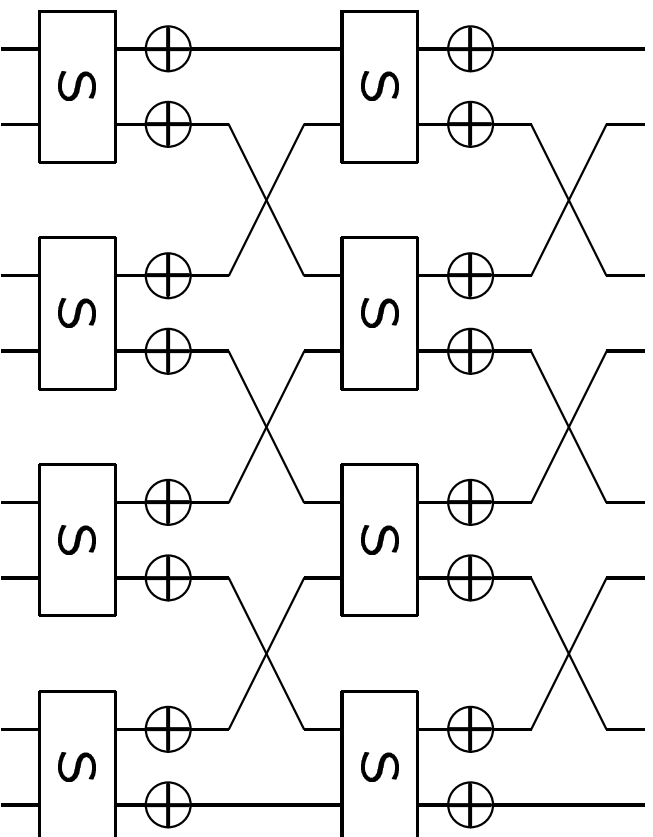
$$\begin{cases} y_1 = x_1 \cdot x_3 \\ y_2 = x_2 \cdot x_3 \end{cases}$$

$$\delta = (0, 0, 1) \Rightarrow \epsilon = (1, 1), (0, 1), (1, 0), (0, 0)$$

Freedom in the *trail*

(except if $\delta = 0$, then $\epsilon = 0$)

An example



Nonlinear components: *active* or *passive*

Bounds for Linear and Differential Crypt-analysis

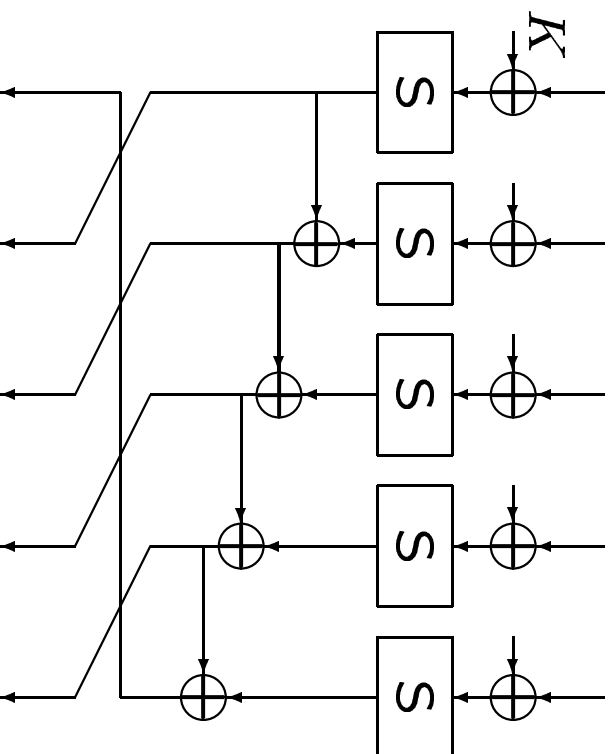
(Under a set of assumptions)

$$P_{\text{cipher}} = \prod_{r=1}^R P_{\text{round } r}$$

$$\begin{aligned} P_{\text{round}} &= \prod_c P_{\text{component } c} \\ &= \prod_c P_{\text{active non-linear component } c} \end{aligned}$$

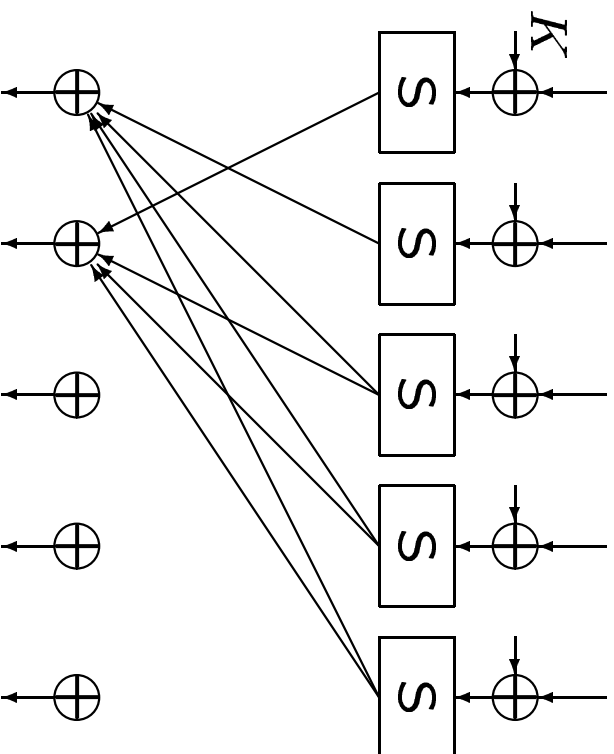
- Design goal: low upper bounds
- Lower bound # of active components

A Naive Solution (SEAL)



some outputs correlate with a small set of inputs

Another Naive Solution (Stafford-Tavares)



the smallest input changes have wide propagation, but there are still *narrow trails*

The Wide Trail Design Strategy

The round function of a block cipher consists of different components, with different functionality:

1. Non-linear substitution: low probability when active
2. Linear mixing transformations: make many non-linear components active
3. Key addition

Components can be selected and tested separately

Keep it as simple as possible

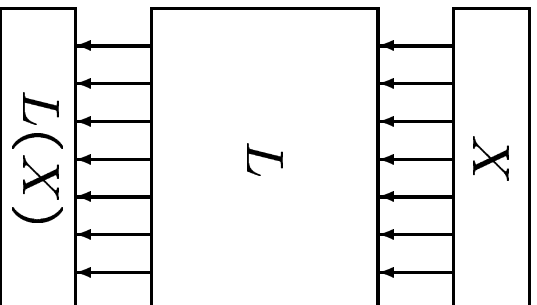
Linear transformation: Branch Number

of active components = $\sum_i \text{Hwt}(\text{difference}_i)$

consider 2 rounds: worst case is given by

$$\mathcal{B}(\mathcal{L}) = \min\{\text{Hwt}(\delta) + \text{Hwt}(\epsilon)\}$$

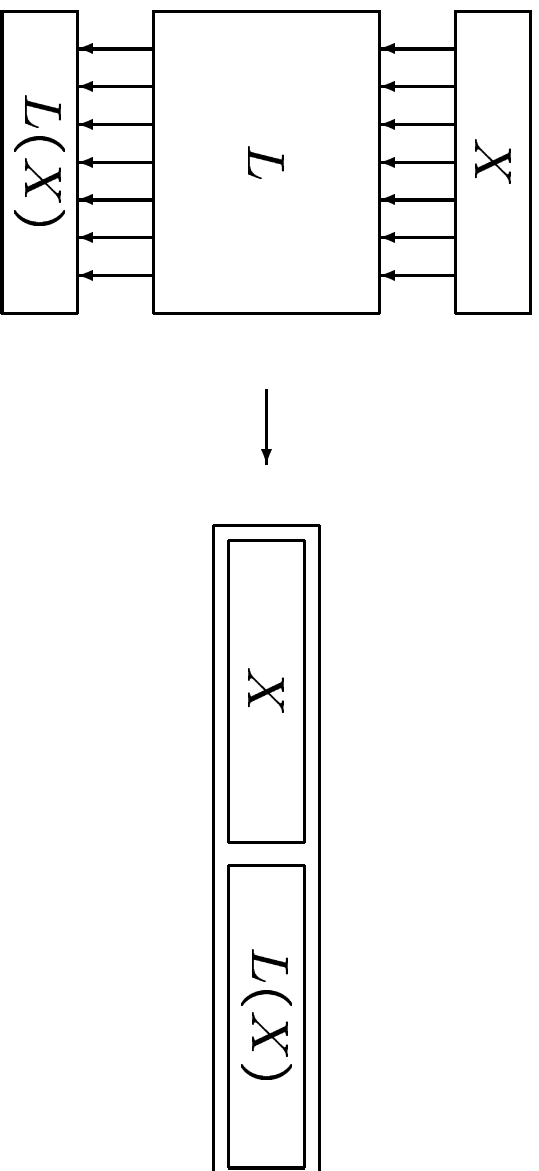
Optimal Diffusion



Maximise \mathcal{B}

\equiv the minimum of $\text{Hwt}(X_1 - X_2) + \text{Hwt}(L(X_1) - L(X_2))$

Idea: linear codes



Linear code with codewords $(X, L(X))$

dimension n

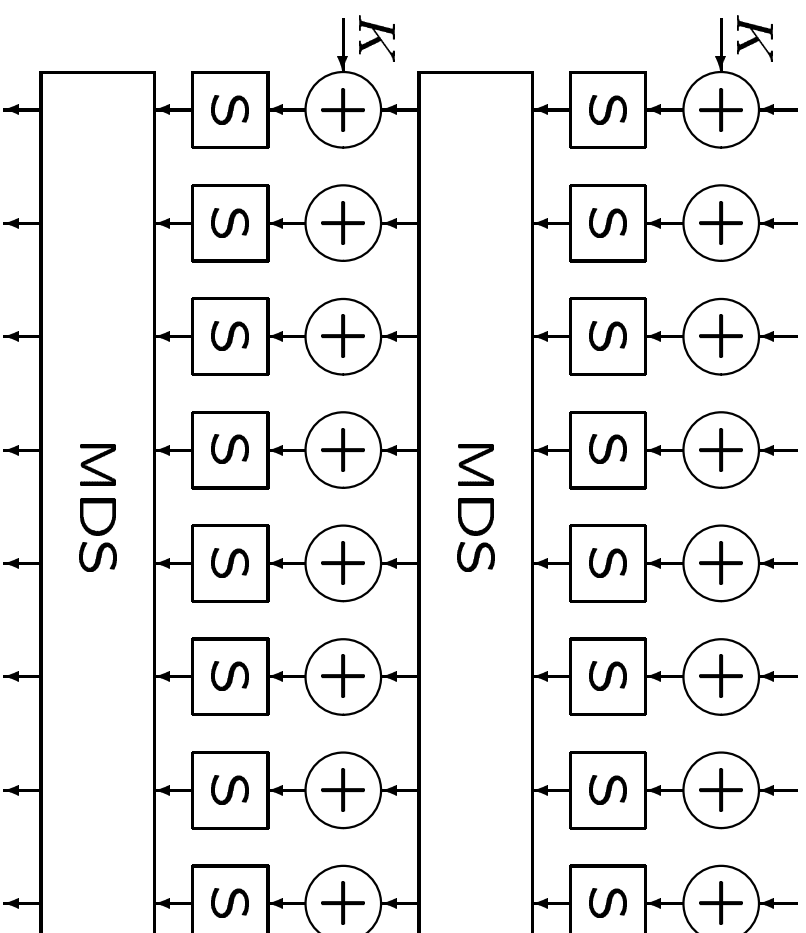
length $2n$

\mathcal{B} = minimal distance between two code words

Maximal distance: $n + 1$

MDS codes

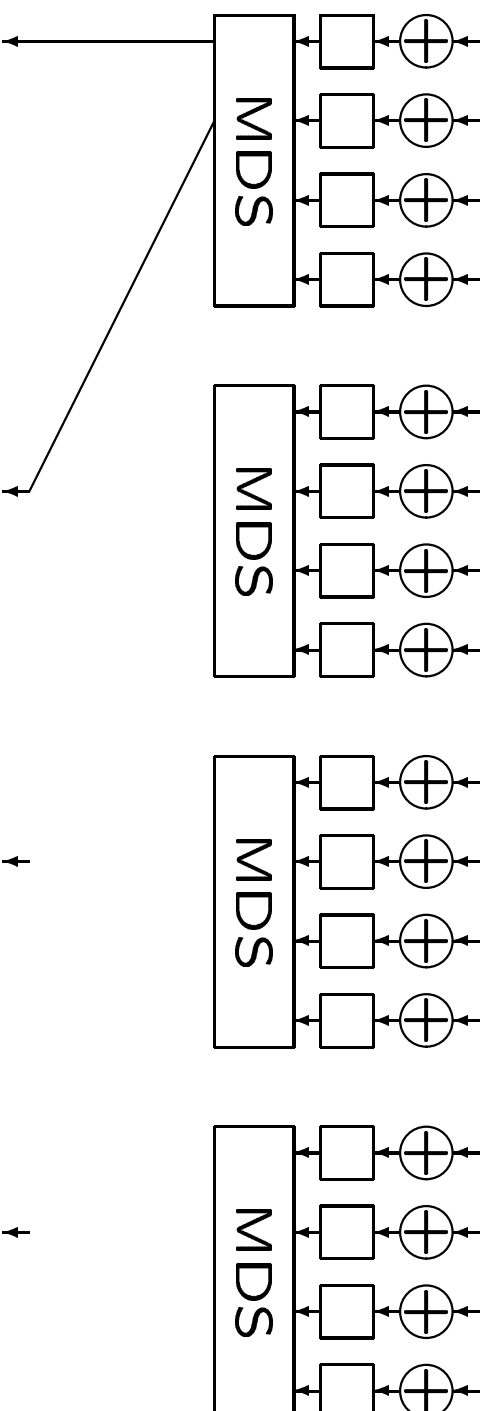
Shark



Number of active S-boxes per 2 rounds $\geq \mathcal{B} = 9$

Problem: efficient implementation

Square



Number of active S-boxes per 4 rounds $\geq B^2 = 25$

Efficient on 8-bit processors and 32-bit processors
Parallelism

Rijndael structure

10/12/14 iterations of the round transformation

uniform and parallel round transformation, composed of:

- byte substitution
- shift rows
- mix columns
- round key addition

Data Representation

Input: byte string

$p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13} p_{14} p_{15}$

Key: byte string

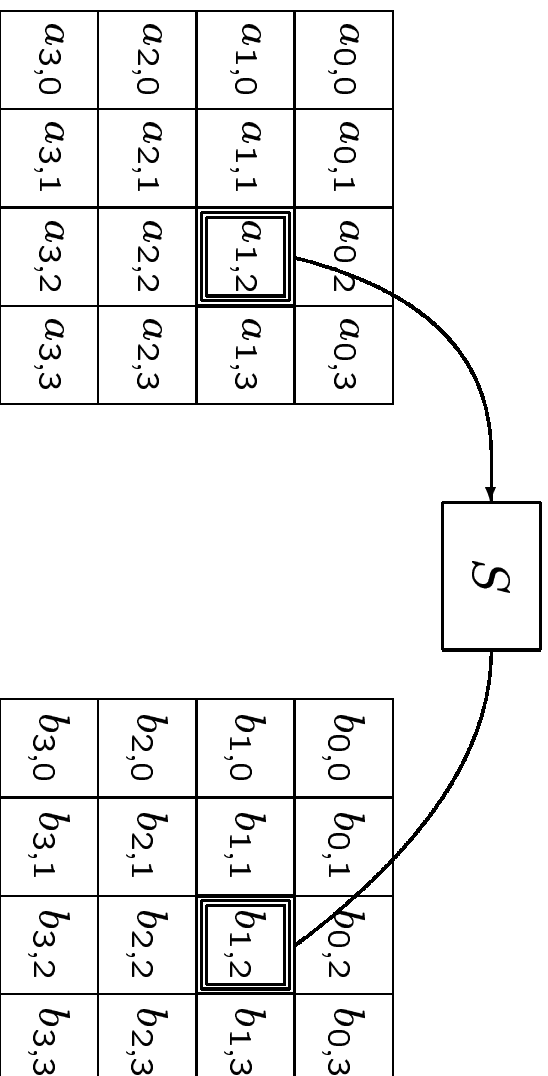
$k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} k_{12} k_{13} k_{14} k_{15} k_{16} k_{17} k_{18} k_{19} k_{20} k_{21} k_{22} k_{23}$

Representation: rectangular byte arrays

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

k_0	k_4	k_8	k_{12}	k_{16}	k_{20}
k_1	k_5	k_9	k_{13}	k_{17}	k_{21}
k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}
k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}

Byte Substitution

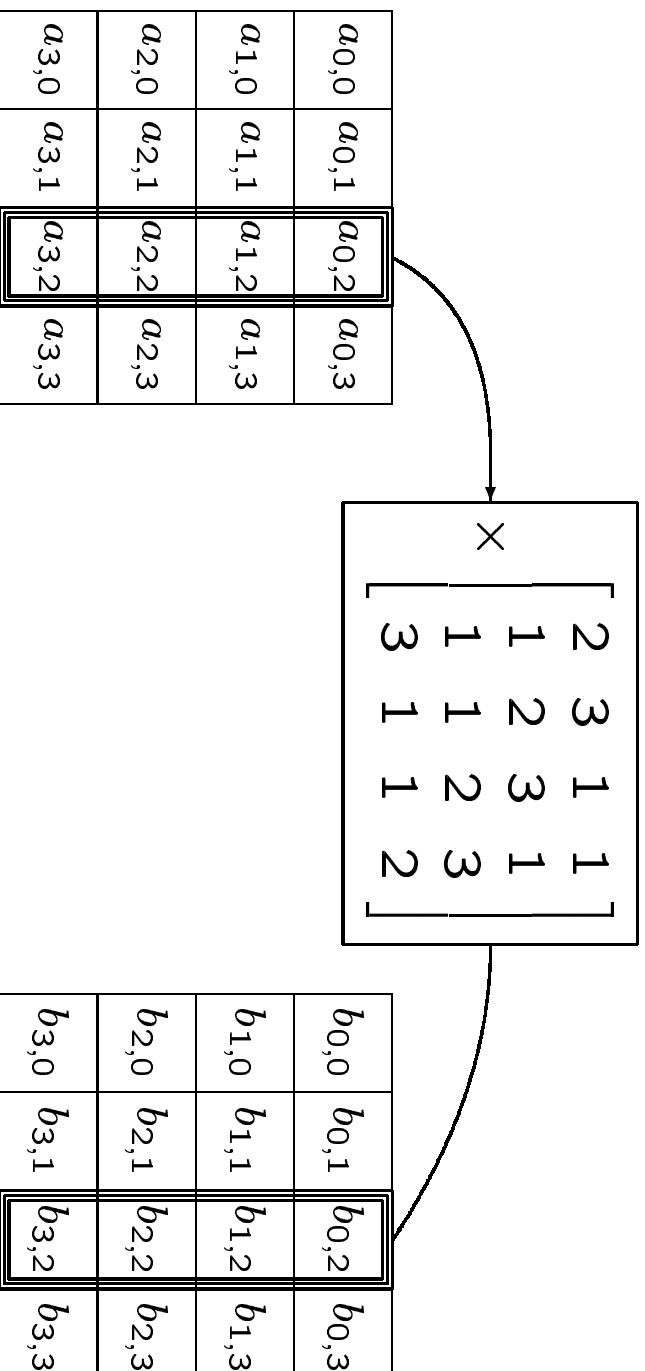


Transformation is byte by byte

Invertible S-box

1 S-box for the whole cipher (simplicity)

Mix Columns



Bytes in columns are combined linearly
Good diffusion properties
Based on Maximal Distance codes

Shift Rows

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>
<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>
<i>m</i>	<i>n</i>	<i>o</i>	<i>p</i>

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
<i>f</i>	<i>g</i>	<i>h</i>	<i>e</i>
<i>k</i>	<i>l</i>	<i>i</i>	<i>j</i>
<i>p</i>	<i>m</i>	<i>n</i>	<i>o</i>

Rows shifted over different offsets
Diffusion over the columns

Round key addition

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} + \begin{array}{|c|c|c|c|} \hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Bit-wise XOR

Rijndael Key Schedule

Different key scheduling for different key lengths (extendible for non-AES key lengths)

Fast and light, because in many applications, the key scheduling is a bottleneck: short messages, hashing, ...

Rijndael Key Schedule (2)

k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}	...
Round key 0			Round key 1			Round key 2			...							

$$k_{6n} = k_{6n-6} + f(k_{6n-1})$$

$$k_i = k_{i-6} + k_{i-1}$$

Cipher key expansion can be done just-in-time

No extra storage required

Difference propagation (I)

SubBytes: differences remain contained

*	*		
	*		

SubBytes
→

*	*		
	*		

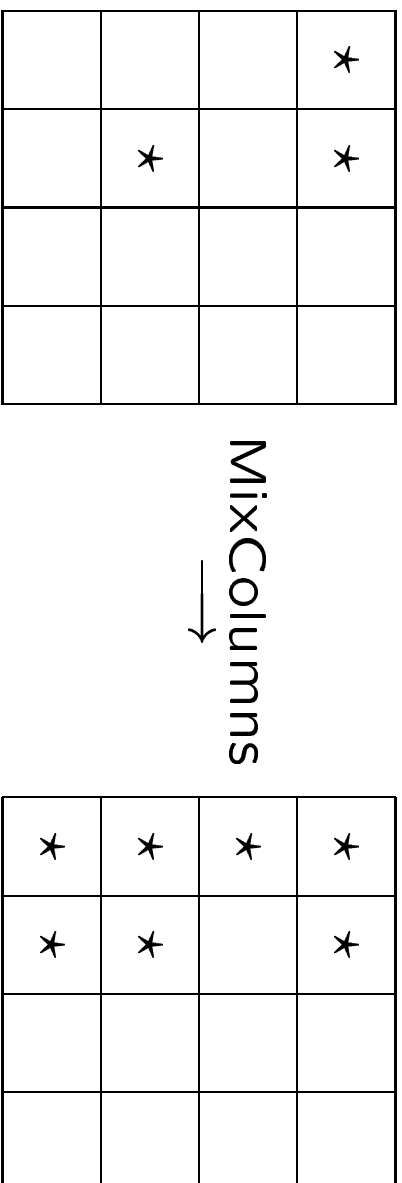
Difference byte values: unpredictable

Difference pattern: fixed

Difference propagation (II)

MixColumns: differences spread in the columns

ShiftRows: spreading over columns



Difference byte values: predictable

Difference pattern: use of MDS codes guarantees wide trails

The S-box

K. Nyberg ('94):

Use $x \rightarrow x^{-1}$ over $\text{GF}(2^8)$

1. optimal spreading of (δ, ϵ)
2. minimal input-output bit correlations
3. maximal nonlinear order of the output
4. 'easy' description

→ Add affine mapping to remediate (4)

The Square Attack

Invented by L.R. Knudsen in 1996: 5 rounds

Improved by:

1996: Daemen, Rijmen: 6 rounds

2000: Gilbert, Minier: 7 rounds (very high complexity)

2000: Ferguson et al.: 7 rounds (using the full codebook),
8 rounds for 256-bit keys

(Rijndael has 10/12/14 rounds)

Has been reinvented as: Saturation attack, Multi-set attack, Integral cryptanalysis

A-sets

A-set:

- 256 texts
- constant bytes
- bytes taking all 256 values exactly once

Example

$\{A_\lambda | \lambda = 0, 1, \dots, 255\}$

- $a_{0,0|\lambda} = 3\lambda + 5$
- $a_{1,j|\lambda} = 0$
- $a_{2,j|\lambda} = j$
- $a_{3,j|\lambda} = 7$

A	C	C	C
C	C	C	C
C	C	C	C
C	C	C	C
C	C	C	C

A-sets Through Rounds

SubBytes: set goes through unchanged

Key Addition: set goes through unchanged

ShiftRows: A/C values are shifted

MixColumns: depends # of A's in column:

0: $\rightarrow 0$

1: $\rightarrow 4$

2,3,4: \rightarrow unknown, but $\sum_{\lambda} a_{i,j|\lambda} = 0$

3-Rounds of Rijndael

1 A \rightarrow 4 A's \rightarrow 16 A's \rightarrow 16 zero-sums

Attack:

1. guess parts of the key in round 1, 5 and 6
2. ask to encrypt texts such that input of round 2 is a A-set
3. verify output of round 4
4. if the sums are not zero then the guessed key must have been wrong

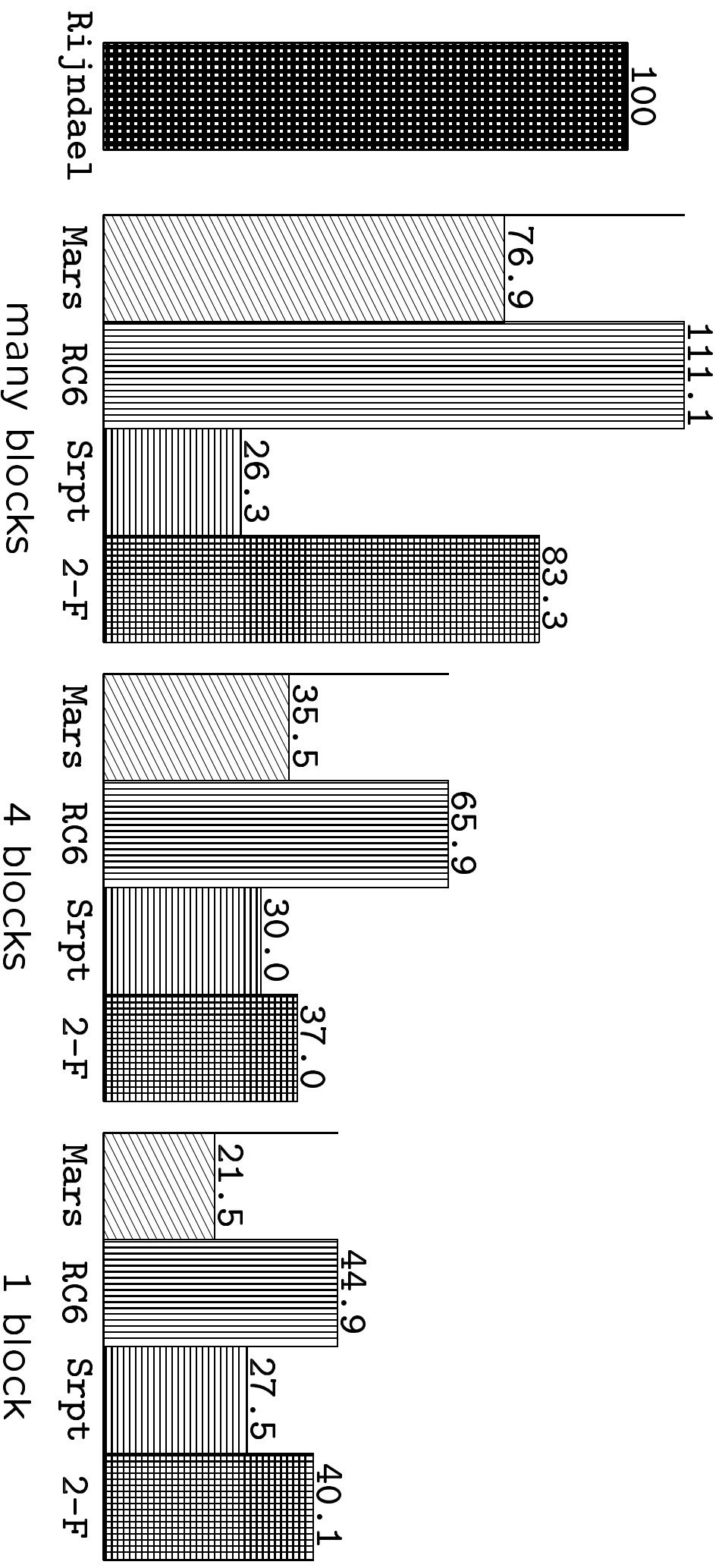
Performance-Security Trade-off

(Serpent designers:) Security weighted performance

→ Performance-weighted Security

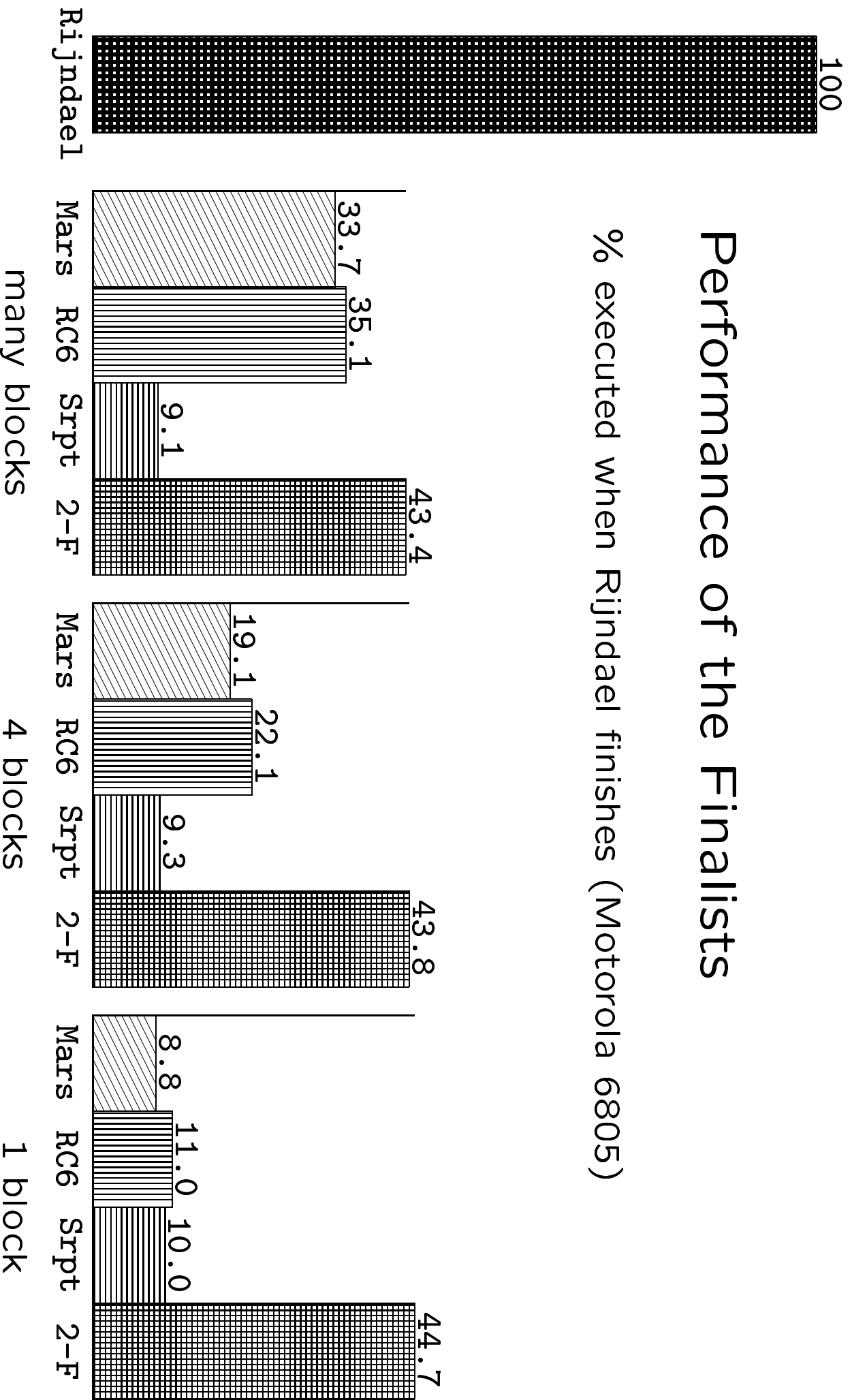
Performance of the Finalists

% executed when Rijndael finishes (Pentium II/Pro)



Performance of the Finalists

% executed when Rijndael finishes (Motorola 6805)



Performance figures of Rijndael

- Pentium III @800 MHz. (assembly):
232 cycles/encryption, or 412 Mbit/s
- IA-64: 124 cycles/encryption
- Motorola 6805 @4 MHz.:
9464 cycles/encryption (+ key setup), or 54 kbit/s
- Hardware (estimated) 5 Gbit/s

Conclusion

- Rijndael/AES is secure against *known* attacks (linear cryptanalysis, differential cryptanalysis, ...)
- Secure against 1 new attack
- Nice security/performance trade-off
- Designers' claim: an attack that breaks Rijndael, will also break other, more complicated designs