Cluster Search and MILP Modeling for Differential Attacks

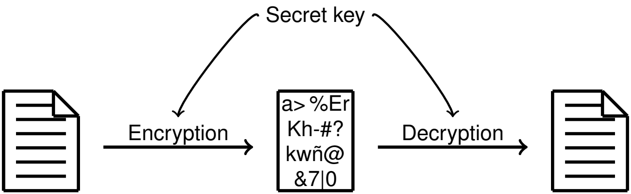
RODRÍGUEZ CORDERO Ana Margarita

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

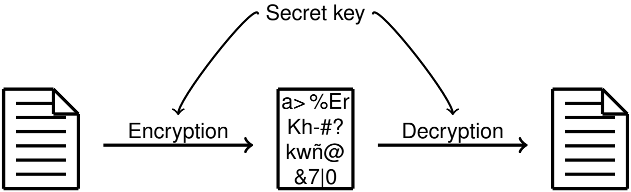
October 27th, 2022

Index

Symmetric Cryptography

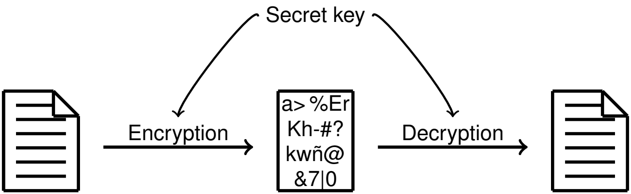


Symmetric Cryptography



- Stream ciphers
- Block ciphers

Symmetric Cryptography



- Block ciphers

Block Cipher

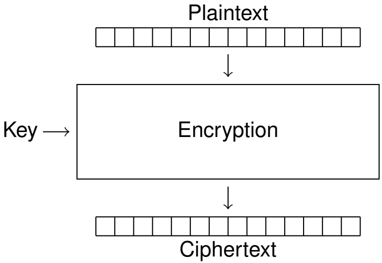
Block Cipher

Given a key $K \in \mathbb{F}_2^m$ and a message $M \in \mathbb{F}_2^N$, a block cipher of block size n is an **invertible** function E_K that encrypts the message M in blocks of size n

Block Cipher

Block Cipher

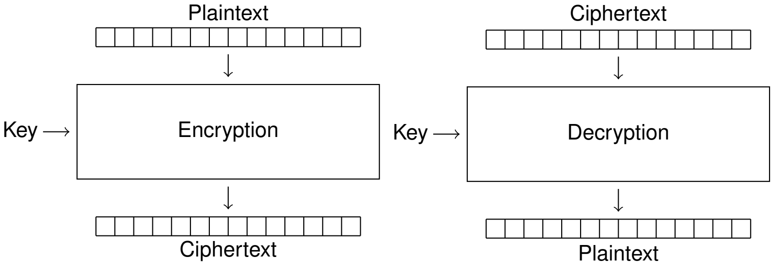
Given a key $K \in \mathbb{F}_2^m$ and a message $M \in \mathbb{F}_2^N$, a block cipher of block size n is an **invertible** function E_K that encrypts the message M in blocks of size n



Block Cipher

Block Cipher

Given a key $K \in \mathbb{F}_2^m$ and a message $M \in \mathbb{F}_2^N$, a block cipher of block size n is an **invertible** function E_K that encrypts the message M in blocks of size n



Block Cipher Constructions

Block Cipher Constructions

- Iterative cipher:

Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ ... \circ f_{K_1}$,

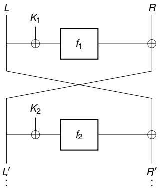
Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, $\quad f_{K_i}$ named round function

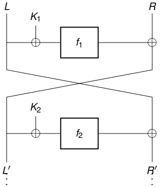
Feistel Network



Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network

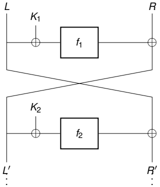


- IBM construction by Horst Feistel

Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network

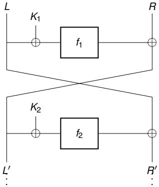


- IBM construction by Horst Feistel
- Identical encryption and decryption

Block Cipher Constructions

- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network

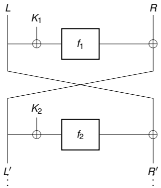


- IBM construction by Horst Feistel
- Identical encryption and decryption
- Used in Data Encryption Standard (DES)

Block Cipher Constructions

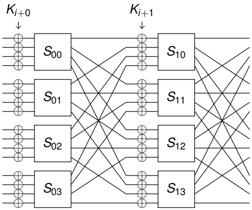
- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, $\quad f_{K_i}$ named round function

Feistel Network



- IBM construction by Horst Feistel
- Identical encryption and decryption
- Used in Data Encryption Standard (DES)

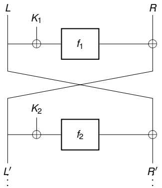
Substitution Permutation Network



Block Cipher Constructions

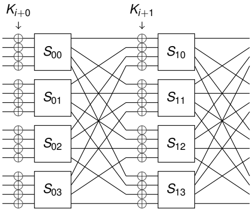
- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network



- IBM construction by Horst Feistel
- Identical encryption and decryption
- Used in Data Encryption Standard (DES)

Substitution Permutation Network

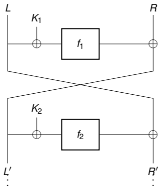


- Non-linear layer is a Substitution box (S-box)

Block Cipher Constructions

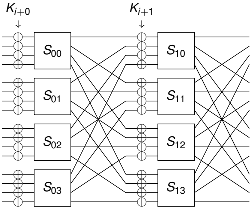
- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network



- IBM construction by Horst Feistel
- Identical encryption and decryption
- Used in Data Encryption Standard (DES)

Substitution Permutation Network

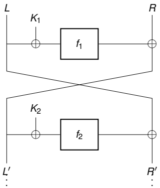


- Non-linear layer is a Substitution box (S-box)
- Linear layer includes a bit, nibble or byte permutation

Block Cipher Constructions

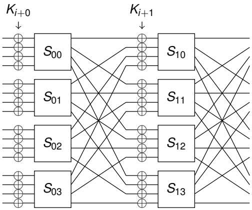
- Iterative cipher: $E_k = f_{K_r} \circ \ldots \circ f_{K_1}$, f_{K_i} named round function

Feistel Network



- IBM construction by Horst Feistel
- Identical encryption and decryption
- Used in Data Encryption Standard (DES)

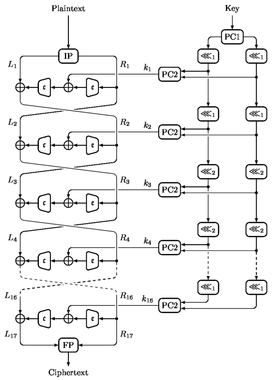
Substitution Permutation Network



- Non-linear layer is a Substitution box (S-box)
- Linear layer includes a bit, nibble or byte permutation
- Used in Advance Encryption Standard (AES)

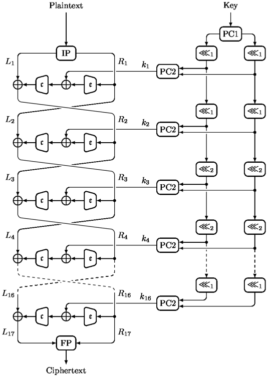
DES

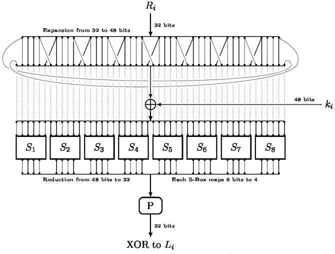
[Jérémy Jean, TikZ for Cryptographers]



DES

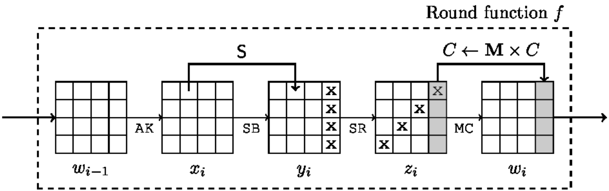
[Jérémy Jean, TikZ for Cryptographers]





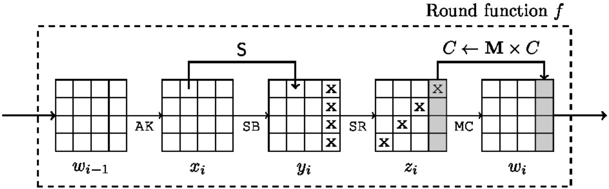
AES

[Daemen and Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography), 2002]



AES

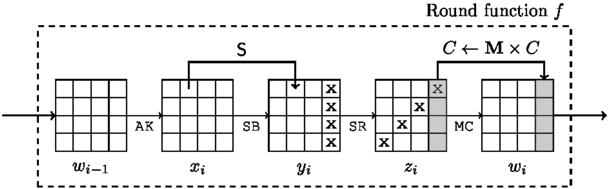
[Daemen and Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography), 2002]



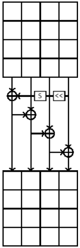
$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

AES

[Daemen and Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography), 2002]

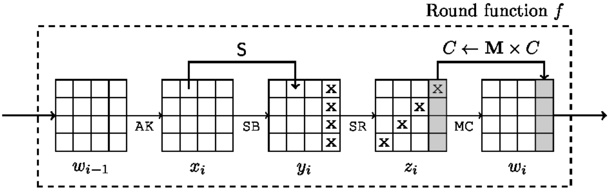


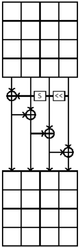
$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$



AES

[Daemen and Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography), 2002]



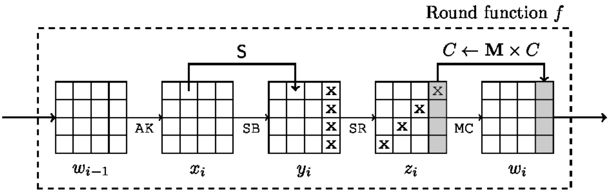


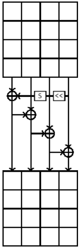
$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

- 128-bit version of Rijndael

AES

[Daemen and Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography), 2002]





$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

- 128-bit version of Rijndael
- Proposed in 1998 by Daemen and Rijmen for the 1997 NIST competition

Block Cipher Operations

Linear operations

Block Cipher Operations

Linear operations

- Constant additions

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Nonlinear operations

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Nonlinear operations

- Bit AND operation

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Nonlinear operations

- Bit AND operation
- Exponentiation

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- . . .

Nonlinear operations

- Bit AND operation
- Exponentiation
- Inverse operation

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Nonlinear operations

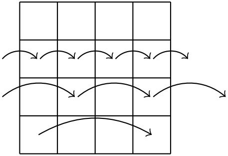
- Bit AND operation
- Exponentiation
- Inverse operation
- ...

Block Cipher Operations

Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- . . .

ShiftRows



Nonlinear operations

- Bit AND operation
- Exponentiation
- Inverse operation
- . . .

Block Cipher Operations

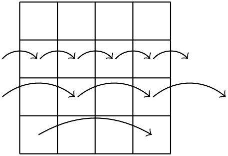
Linear operations

- Constant additions
- Bit XOR
- Mix columns matrices
- ...

Nonlinear operations

- Bit AND operation
- Exponentiation
- Inverse operation
- ...

ShiftRows



S-box

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S (**x**)	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

Substitution Box

A *substitution box* (*S-box*), is a non-linear operation usually represented as a look-up table:

$$\begin{array}{rccc} \mathbf{S}: & \mathbb{F}_2^{m_1} & \longrightarrow & \mathbb{F}_2^{m_2} \\ & \mathbf{x} & \mapsto & \mathbf{S}(\mathbf{x}) \end{array}$$

Substitution Box

A *substitution box* (*S-box*), is a non-linear operation usually represented as a look-up table:

$$\begin{array}{rccc} \mathbf{S}: & \mathbb{F}_2^{m_1} & \longrightarrow & \mathbb{F}_2^{m_2} \\ & \mathbf{x} & \mapsto & \mathbf{S}(\mathbf{x}) \end{array}$$

AES S-box:

$\mathbb{F}_{256}$

1. If $x \neq 0$, replace x by its inverse, $x = x^{-1}$ in $\mathbb{F}_2^{8*}$

* Generated by the polynomial $m(X) = X^8 + X^4 + X^3 + X + 1$

Substitution Box

A *substitution box* (*S-box*), is a non-linear operation usually represented as a look-up table:

$$\begin{array}{rccc} \mathbf{S}: & \mathbb{F}_2^{m_1} & \longrightarrow & \mathbb{F}_2^{m_2} \\ & \mathbf{x} & \mapsto & \mathbf{S}(\mathbf{x}) \end{array}$$

AES S-box:

$\mathbb{F}_{256}$

1. If $x \neq 0$, replace x by its inverse, $x = x^{-1}$ in $\mathbb{F}_2^{8*}$
2. $x = Ax + b$, where A is a fix 8×8 binary matrix and b is a fix 8 binary vector

* Generated by the polynomial $m(X) = X^8 + X^4 + X^3 + X + 1$

Differential Distinguisher

- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$

Differential Distinguisher

- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$
- For a given $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$,

Differential Distinguisher

- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$
- For a given $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$, $x \in \mathbb{F}_2^n$,

Differential Distinguisher

- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$
- For a given $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$, $x \in \mathbb{F}_2^n$, $p(E_K(x \oplus \Delta) = \nabla) \simeq 2^{-n}$

Differential Distinguisher

- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$
- For a given $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$, $x \in \mathbb{F}_2^n$, $p(E_K(x \oplus \Delta) = \nabla) \simeq 2^{-n}$

Differential Distinguisher

Find a pair $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$ such that $p(E_K(x \oplus \Delta) = \nabla) \gg 2^{-n}$

Differential Distinguisher

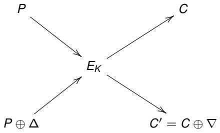
- An ideal block cipher should behave like a random permutation in $\mathbb{F}_2^n$
- For a given $(\Delta, \nabla) \in \mathbb{F}_2^{2n},\ x \in \mathbb{F}_2^n,\ p(E_K(x \oplus \Delta) = \nabla) \simeq 2^{-n}$

Differential Distinguisher

Find a pair $(\Delta, \nabla) \in \mathbb{F}_2^{2n}$ such that $p(E_K(x \oplus \Delta) = \nabla) \gg 2^{-n}$

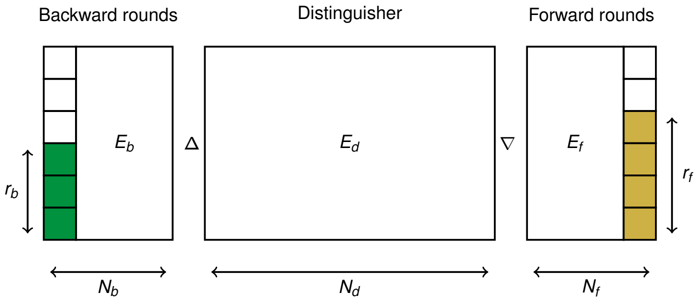
Study the propagation of input differences throughout the cipher:

$$\nabla = E_K(P) \oplus E_K(P \oplus \Delta)$$



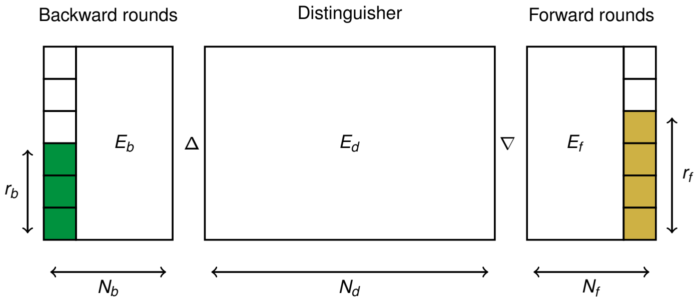
Differential Attack

[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



Differential Attack

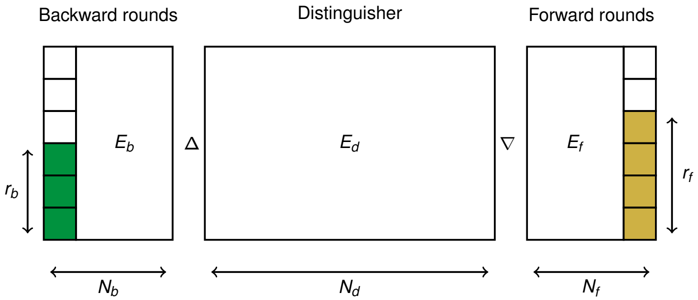
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991

Differential Attack

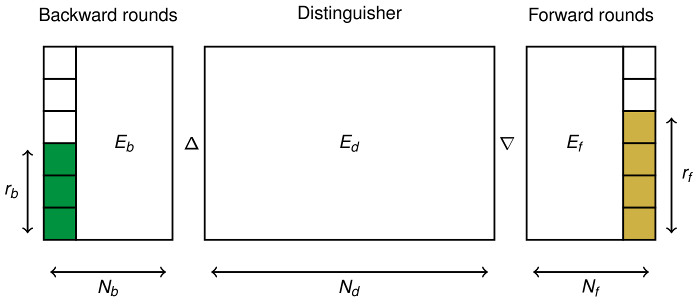
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA

Differential Attack

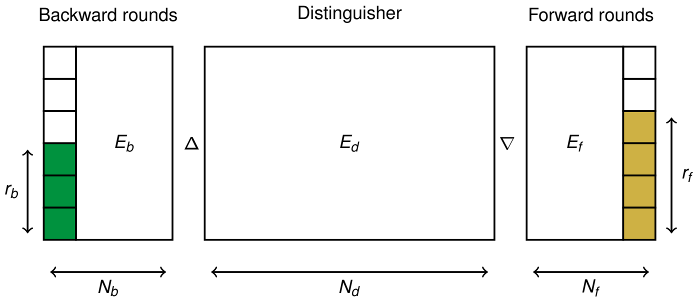
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential

Differential Attack

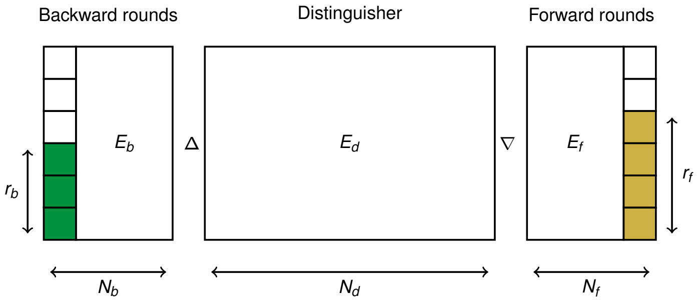
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find

Differential Attack

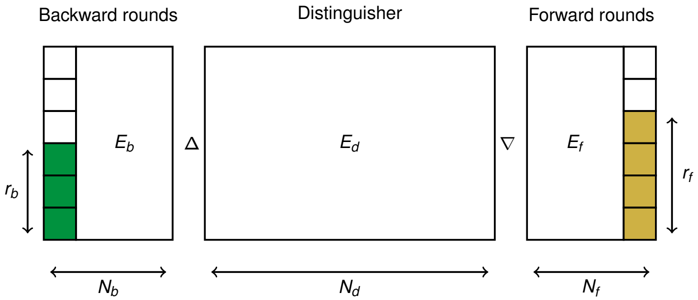
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find
- Differential characteristic: $\Delta = \delta_0 \rightarrow$

Differential Attack

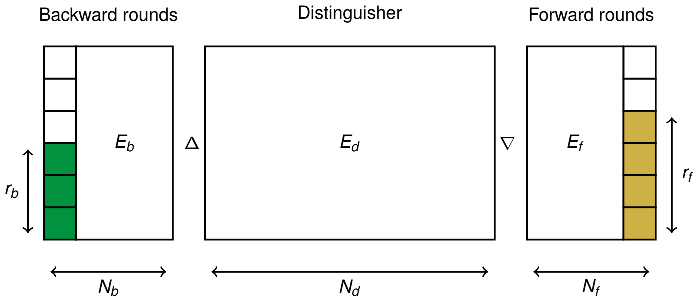
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find
- Differential characteristic: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow$

Differential Attack

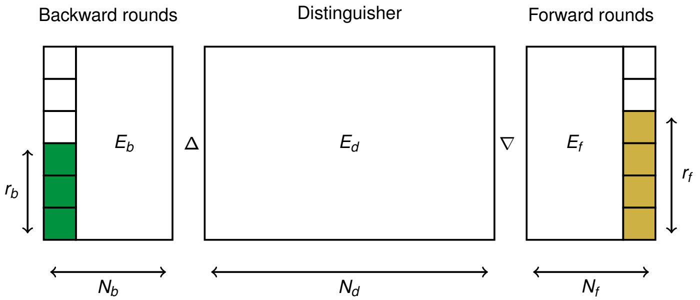
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find
- Differential characteristic: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow$

Differential Attack

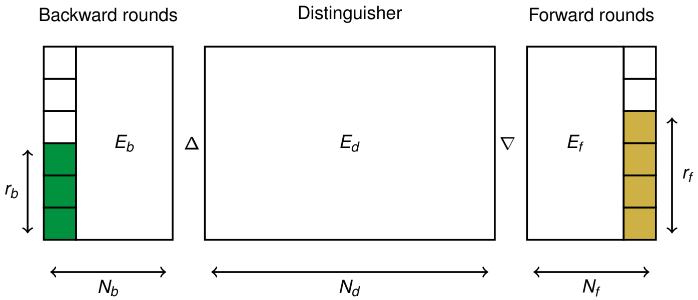
[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find
- Differential characteristic: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow \delta_r = \nabla$

Differential Attack

[Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology 4, 3–72 (1991)]



- Eli Biham and Adi Shamir 1991
- Already known by IBM and some security agencies like NSA
- The pair (Δ, ∇) is referred to as a differential usually hard to find
- Differential characteristic: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow \delta_r = \nabla$
- Analyze the differential behavior of cipher operations

Computing the Probability

$$p(\delta_0 \to \delta_1 \cdots \to \delta_r)$$

Computing the Probability

$$p(\delta_0 \to \delta_1 \cdots \to \delta_r)$$

$\to$ On the board

Difference Distribution Table

Difference Distribution Table

Difference Distribution Table:

$$DDT(\Delta_i, \nabla_o) = \#\left\{\mathbf{x} \in \mathbb{F}_2^n : S(\mathbf{x}) \oplus S(\mathbf{x} \oplus \Delta_i) = \nabla_o\right\}$$

Difference Distribution Table

Difference Distribution Table:

$$DDT(\Delta_i, \nabla_o) = \#\left\{\mathbf{x} \in \mathbb{F}_2^n : S(\mathbf{x}) \oplus S(\mathbf{x} \oplus \Delta_i) = \nabla_o\right\}$$

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	8	0	0	0	0	0	0	0
0x1	0	2	2	0	0	2	2	0
0x2	0	2	2	0	0	2	2	0
0x3	0	0	0	4	0	0	0	4
0x4	0	0	0	0	4	0	0	4
0x5	0	2	2	0	0	2	2	0
0x6	0	2	2	0	0	2	2	0
0x7	0	0	0	4	4	0	0	0

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1
- We obtain: *a Truncated differential characteristics*

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1
- We obtain: *a Truncated differential characteristics*

- Step2: Find a differential characteristic from the truncated differential

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1
- We obtain: *a Truncated differential characteristics*

- Step2: Find a differential characteristic from the truncated differential
- Follow the differential behaviour of the nonlinear operation

Abstraction Approach

- Step1: Minimize the number of non-linear *active* operations
- Step1: Any non-zero difference is represented as 1
- We obtain: *a Truncated differential characteristics*

- Step2: Find a differential characteristic from the truncated differential
- Follow the differential behaviour of the nonlinear operation

-> Obtain differential characteristics

Clusters

- We obtain differential characteristics from the abstraction method: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow \delta_r = \nabla$.

Clusters

- We obtain differential characteristics from the abstraction method: $\Delta = \delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow \delta_r = \nabla$.
- Easy to compute $p(\delta_0 \rightarrow \delta_1 \rightarrow \cdots \rightarrow \delta_r) = \Pi_{i=0}^{r-1} P(\delta_i \rightarrow \delta_{i+1})$.

Clusters

- We obtain differential characteristics from the abstraction method: $\Delta = \delta_0 \to \delta_1 \to \cdots \to \delta_r = \nabla$.
- Easy to compute $p(\delta_0 \to \delta_1 \to \cdots \to \delta_r) = \Pi_{i=0}^{r-1} P(\delta_i \to \delta_{i+1})$.
- What is the exact probability of the differential (Δ, ∇)?

Clusters

- We obtain differential characteristics from the abstraction method: $\Delta = \delta_0 \to \delta_1 \to \cdots \to \delta_r = \nabla$.
- Easy to compute $p(\delta_0 \to \delta_1 \to \cdots \to \delta_r) = \Pi_{i=0}^{r-1} P(\delta_i \to \delta_{i+1})$.
- What is the exact probability of the differential (Δ, ∇)?

Cluster

A cluster is a set of differential characteristics, for a given number of rounds, that have the same input and output difference

$$\Delta = \delta_0^j \to \delta_1^j \to \cdots \to \delta_r^j = \nabla$$

Clusters

- We obtain differential characteristics from the abstraction method: $\Delta = \delta_0 \to \delta_1 \to \cdots \to \delta_r = \nabla$.
- Easy to compute $p(\delta_0 \to \delta_1 \to \cdots \to \delta_r) = \Pi_{i=0}^{r-1} P(\delta_i \to \delta_{i+1})$.
- What is the exact probability of the differential (Δ, ∇)?

Cluster

A cluster is a set of differential characteristics, for a given number of rounds, that have the same input and output difference

$$\Delta = \delta_0^j \to \delta_1^j \to \cdots \to \delta_r^j = \nabla$$

$$p(\Delta \to \nabla) \approx \sum_j p(\delta_0^j \to \delta_1^j \to \cdots \to \delta_r^j)$$

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle
- Search forward and backward to the middle

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle
- Search forward and backward to the middle
- Stores the middle values in a table

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle
- Search forward and backward to the middle
- Stores the middle values in a table
- Performs a crossed search

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
  - Too much computation time

Naive approach improved:

- Fix also probability
  - Improves computation time
  - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle
- Search forward and backward to the middle
- Stores the middle values in a table
- Performs a crossed search
  - Finds the whole cluster (for small number of rounds)

Implementing Cluster Search

[Chen et all., Improved Differential Characteristic Searching Methods, 2015]

Naive approach

- Fix input and output differences, change in the middle:
    - Too much computation time

Naive approach improved:

- Fix also probability
    - Improves computation time
    - Obtains first the highest probabilities

Meet in the middle method

- From the truncated path: selects a round with few values in the middle
- Search forward and backward to the middle
- Stores the middle values in a table
- Performs a crossed search
    - Finds the whole cluster (for small number of rounds)
    - Uses too much memory

MILP Modeling

MILP: Mixed-Integer Linear Programming

MILP Modeling

ILP: Integer Linear Programming

- Minimize or maximize an objective function

$$\sum_i a_i X_i$$

MILP Modeling

ILP: Integer Linear Programming

- Minimize or maximize an objective function

$$\sum_i a_i X_i$$

- Constraints $\sum b_i X_i \geq b$ $\sum c_i X_i \leq c$ $\sum d_i X_i == d$

MILP Modeling

ILP: Integer Linear Programming

- Minimize or maximize an objective function

$$\sum_i a_i X_i$$

- Constraints $\sum b_i X_i \geq b$ $\sum c_i X_i \leq c$ $\sum d_i X_i == d$

XOR Truth Table

$a, b, c \in \mathbb{F}_2$

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

MILP Modeling

ILP: Integer Linear Programming

- Minimize or maximize an objective function

$$\sum_i a_i X_i$$

- Constraints $\sum b_i X_i \geq b$ $\sum c_i X_i \leq c$ $\sum d_i X_i == d$

XOR Truth Table

$a, b, c \in \mathbb{F}_2$

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Non-valid transitions:
$(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)$

MILP Modeling

ILP: Integer Linear Programming

- Minimize or maximize an objective function

$$\sum_i a_i X_i$$

- Constraints $\sum b_i X_i \geq b \qquad \sum c_i X_i \leq c \qquad \sum d_i X_i == d$

XOR Truth Table

$a, b, c \in \mathbb{F}_2$

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Non-valid transitions:

$(0,0,1), (0,1,0), (1,0,0), (1,1,1)$

$a + b \geq c \qquad a + c \geq b \qquad b + c \geq a$

$a + b + c \leq 2$

Step1 MILP Modeling

Working with truncated characteristics:

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Non-valid transitions:
$(0, 0, 1), (0, 1, 0), (1, 0, 0)$

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Non-valid transitions:
$(0,0,1),(0,1,0),(1,0,0)$
$a + b + c \neq 1 \Rightarrow$

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Non-valid transitions:

$$(0,0,1),(0,1,0),(1,0,0)$$

$$a+b+c \neq 1 \Rightarrow$$

$$a+b \geq c \qquad a+c \geq b \qquad b+c \geq a$$

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Non-valid transitions:

$$(0,0,1),(0,1,0),(1,0,0)$$

$$a+b+c \neq 1 \Rightarrow$$

$$a+b \geq c \qquad a+c \geq b \qquad b+c \geq a$$

Step1 objective function:

Step1 MILP Modeling

Working with truncated characteristics:

Abstracted XOR Truth Table

a	b	$c = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0
1	1	1

Non-valid transitions:

$$(0,0,1),(0,1,0),(1,0,0)$$

$$a+b+c \neq 1 \Rightarrow$$

$$a+b \geq c \qquad a+c \geq b \qquad b+c \geq a$$

Step1 objective function:

$$\sum_{i,r} X_{i,r}$$

where i word position, r round, $X_{i,r} = 1$ if there is a non-zero value at the S-box, zero otherwise.

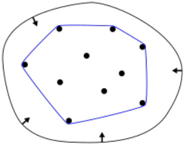
Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

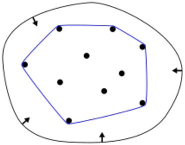
- H-representation of the convex-hull



Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

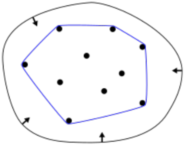
- H-representation of the convex-hull



Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

- H-representation of the convex-hull

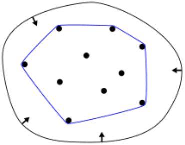


- Product-of-Sum Representation of Boolean Functions

Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

- H-representation of the convex-hull

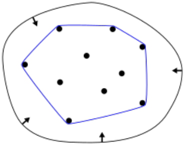


- Product-of-Sum Representation of Boolean Functions Quine-McCluskey (QM) algorithm

Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

- H-representation of the convex-hull

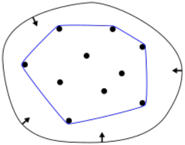


- Product-of-Sum Representation of Boolean Functions Quine-McCluskey (QM) algorithm
- Logical condition techniques for 8-bit S-boxes

Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

- H-representation of the convex-hull



- Product-of-Sum Representation of Boolean Functions Quine-McCluskey (QM) algorithm
- Logical condition techniques for 8-bit S-boxes

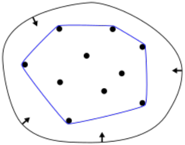
Minimization

- Greedy algorithm
- MILP minimization
- Prime implicants table

Step2 MILP Modeling

How do we model a non-linear function only with linear constraints?

- H-representation of the convex-hull



- Product-of-Sum Representation of Boolean Functions Quine-McCluskey (QM) algorithm
- Logical condition techniques for 8-bit S-boxes

Minimization

- Greedy algorithm
- MILP minimization
- Prime implicants table

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: DDT

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	8	0	0	0	0	0	0	0
0x1	0	2	2	0	0	2	2	0
0x2	0	2	2	0	0	2	2	0
0x3	0	0	0	4	0	0	0	4
0x4	0	0	0	0	4	0	0	4
0x5	0	2	2	0	0	2	2	0
0x6	0	2	2	0	0	2	2	0
0x7	0	0	0	4	4	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	8	0	0	0	0	0	0	0
0x1	0	2	2	0	0	2	2	0
0x2	0	2	2	0	0	2	2	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	2	2	0	0	2	2	0
0x6	0	2	2	0	0	2	2	0
0x7	0	0	0	0	0	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	8	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	4	0	0	0	4
0x4	0	0	0	0	4	0	0	4
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	4	4	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

    $\{(0,0),(1,1),(1,2),(1,5),\ldots,(7,4)\}$

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(7,4)\}$$

  - Set of valid transitions 2-DDT and 4-DDT

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(7,4)\}$$

  - Set of valid transitions 2-DDT and 4-DDT

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(6,6)\}$$

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(7,4)\}$$

  - Set of valid transitions 2-DDT and 4-DDT

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(6,6)\}$$

    $$\{(0,0),(3,3),(3,7),(4,4),\ldots,(7,4)\}$$

DDT Modeling with MILP

[Abdelkhalek et al., MILP Modeling for (Large) S-boxes to OptimizeProbability of Differential Characteristics, 2017]

- Establish valid transitions: *-DDT

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	1	1	0	0	0

- Use 2-DDT and 4-DDT:

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	1	1	0	0	1	1	0
0x2	0	1	1	0	0	1	1	0
0x3	0	0	0	0	0	0	0	0
0x4	0	0	0	0	0	0	0	0
0x5	0	1	1	0	0	1	1	0
0x6	0	1	1	0	0	1	1	0
0x7	0	0	0	0	0	0	0	0

Δ: Input difference	∇: output difference							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x0	1	0	0	0	0	0	0	0
0x1	0	0	0	0	0	0	0	0
0x2	0	0	0	0	0	0	0	0
0x3	0	0	0	1	0	0	0	1
0x4	0	0	0	0	1	0	0	1
0x5	0	0	0	0	0	0	0	0
0x6	0	0	0	0	0	0	0	0
0x7	0	0	0	1	1	0	0	0

- H-representation of convex-hull:
  - Set of valid transitions DDT:

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(7,4)\}$$

  - Set of valid transitions 2-DDT and 4-DDT

    $$\{(0,0),(1,1),(1,2),(1,5),\ldots,(6,6)\}$$

    $$\{(0,0),(3,3),(3,7),(4,4),\ldots,(7,4)\}$$

$$\sum_{i=0}^{7} a_i^j x_i + a^j \geq 0$$

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)
  - 1 if it is used

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)
  - 1 if it is used
  - 0 otherwise

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)
  - 1 if it is used
  - 0 otherwise
- Relate each inequality to the points that satisfy it

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)
  - 1 if it is used
  - 0 otherwise
- Relate each inequality to the points that satisfy it
- Minimize the number of inequalities constrained to: all points must be included

MILP Minimization of the H-representation

[Sasaki and Todo, New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search, 2017]

- Assign a variable to each inequality (Value 0 or 1)
  - 1 if it is used
  - 0 otherwise
- Relate each inequality to the points that satisfy it
- Minimize the number of inequalities constrained to: all points must be included

- Step2 objective: Maximize the probability of the transitions throughout 2-DDT and 4-DDT

Lightweight Cryptography

Lightweight Cryptography

Need for encryption and authentication on constrained devices

Lightweight Cryptography

Need for encryption and authentication on constrained devices

- Small hardware footprint

Lightweight Cryptography

Need for encryption and authentication on constrained devices

- Small hardware footprint
- Small block size

Lightweight Cryptography

Need for encryption and authentication on constrained devices

- Small hardware footprint
- Small block size
- Low-latency

Lightweight Cryptography

Need for encryption and authentication on constrained devices

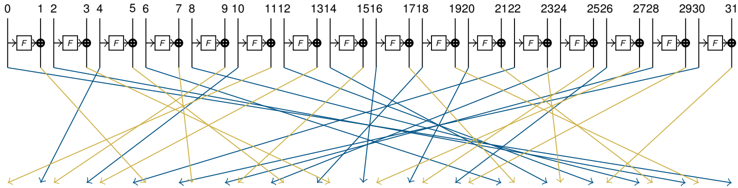
- Small hardware footprint
- Small block size
- Low-latency
- Low-energy consumption

Warp

[Banik et all., WARP : Revisiting GFN for Lightweight 128-bit Block Cipher, 2020]

Warp

[Banik et all., WARP : Revisiting GFN for Lightweight 128-bit Block Cipher, 2020]

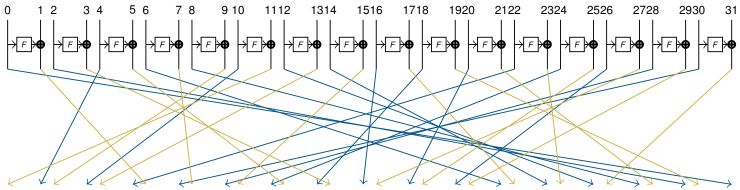


$F = ARK \circ S$

- 128-bit Generalize Feistel cipher

Warp

[Banik et all., WARP : Revisiting GFN for Lightweight 128-bit Block Cipher, 2020]

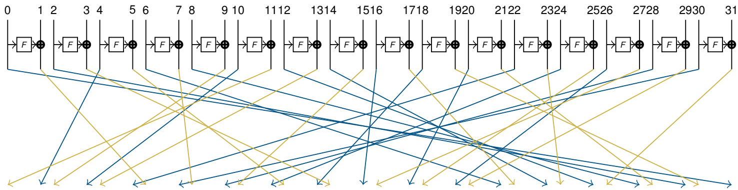


$F = ARK \circ S$

- 128-bit Generalize Feistel cipher
- 128-bit key size

Warp

[Banik et all., WARP : Revisiting GFN for Lightweight 128-bit Block Cipher, 2020]

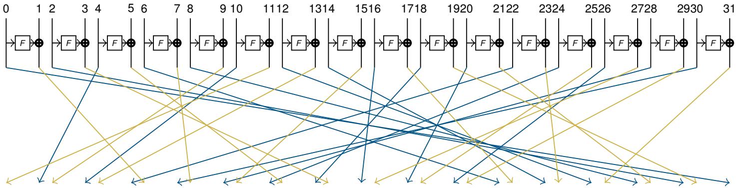


$F = ARK \circ S$

- 128-bit Generalize Feistel cipher
- 128-bit key size
- Linear key schedule

Warp

[Banik et all., WARP : Revisiting GFN for Lightweight 128-bit Block Cipher, 2020]



$F = ARK \circ S$

- 128-bit Generalize Feistel cipher
- 128-bit key size
- Linear key schedule
- 41 round function iterations

Clusters for Warp

Rounds	S-boxes	n_sol	Step2 -log(prob)	Cluster size	Cluster prob
10	17	2	34	4	32
11	22	2	44	4	42
12	28	4	56	16	53
13	34	2	68	512	59

Clusters on Warp

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**a**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**a**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

Clusters on Warp

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**a**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**a**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**5**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**5**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

Clusters on Warp

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**a**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**a**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**5**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**5**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

0,	0,	0,	0,	0,	0,	7,	*d*,	*a*,	*d*,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	*d*,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*d*,	7,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,
0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*d*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**d**,	0,
0,	0,	0,	0,	*a*,	0,	*a*,	**d**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	*a*,
0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	*a*,	0,	0,	*a*,	0,	*a*,	0,	0,	0,	0,	*a*,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	*a*,	0,	0,	0,	0,	0,	0,	0,	0,	*a*,	*a*,	0,	0,	*a*,	0,	0,	*d*,	0,	0,	5,	0,	0,	*a*,	0,	*a*,	0,

Clusters on Warp

0,	0,	0,	0,	0,	0,	7,	d,	a,	d,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	d,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	d,	7,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,
0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**a**,	0,
0,	0,	0,	0,	a,	0,	a,	**a**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	a,
0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	a,	0,	0,	a,	0,	a,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	a,	0,	0,	d,	0,	0,	5,	0,	0,	a,	0,	a,	0,

0,	0,	0,	0,	0,	0,	7,	d,	a,	d,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	d,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	d,	7,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,
0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**5**,	0,
0,	0,	0,	0,	a,	0,	a,	**5**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	a,
0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	a,	0,	0,	a,	0,	a,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	a,	0,	0,	d,	0,	0,	5,	0,	0,	a,	0,	a,	0,

0,	0,	0,	0,	0,	0,	7,	d,	a,	d,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	d,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	d,	7,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,
0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**d**,	0,
0,	0,	0,	0,	a,	0,	a,	**d**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	a,
0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	a,	0,	0,	a,	0,	a,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	a,	0,	0,	d,	0,	0,	5,	0,	0,	a,	0,	a,	0,

0,	0,	0,	0,	0,	0,	7,	d,	a,	d,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	d,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	d,	7,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,
0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	d,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,
0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	**f**,	0,
0,	0,	0,	0,	a,	0,	a,	**f**,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	a,	0,	0,	0,	0,	a,
0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	5,	0,	0,	0,	a,	0,	0,	a,	0,	a,	0,	0,	0,	0,	a,	0,	0,	0,	0,	0,
0,	0,	0,	0,	7,	0,	a,	0,	0,	0,	0,	0,	0,	0,	0,	a,	a,	0,	0,	a,	0,	0,	d,	0,	0,	5,	0,	0,	a,	0,	a,	0,

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

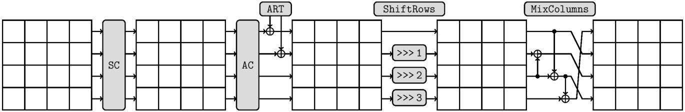
- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:

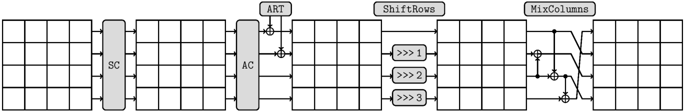


Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:



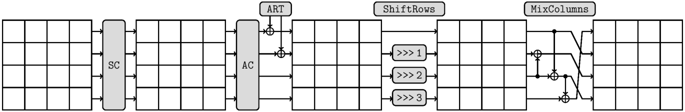
$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

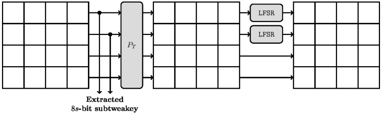
- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:



Tweakey update

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

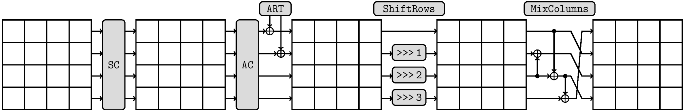


Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

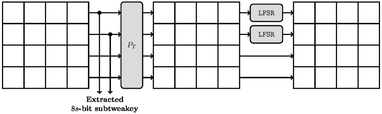
- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:



Tweakey update

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$



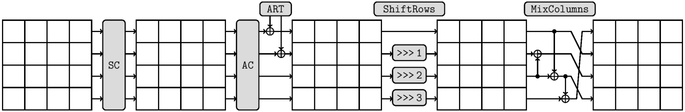
- S-boxes can be implemented with a few XOR and NOR operations

Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

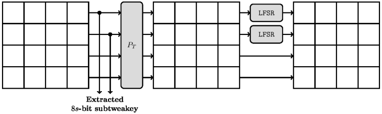
- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:

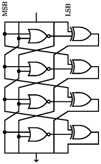


$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Tweakey update



- S-boxes can be implemented with a few XOR and NOR operations

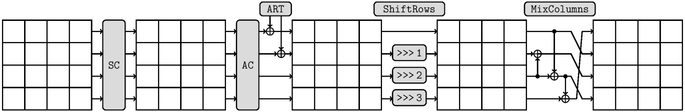


Skinny

[Beierle et all., The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS, 2016]

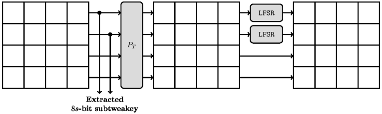
- n-bit SPN block cipher (n=64,128), seen as a 4×4 matrix
- Tweakey size of $t = kn$, $k = 1, 2, 3$
- Number of rounds between 32 and 56

Round function:

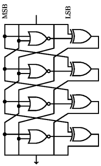


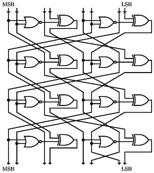
$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Tweakey update



- S-boxes can be implemented with a few XOR and NOR operations





Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem → Many solutions don't have a Step2 solution

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem → Many solutions don't have a Step2 solution
- Increase the objective of Step1

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem $\rightarrow$ Many solutions don't have a Step2 solution
- Increase the objective of Step1 $\rightarrow$ Too many solutions

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem → Many solutions don't have a Step2 solution
- Increase the objective of Step1 → Too many solutions
- Proposed solution:

Step1 Skinny

[Delaune et all., SKINNY with Scalpel - Comparing Tools for Differential Analysis, 2020]

Rounds	SK	n_sol	TK1	n_sol	TK2	n_sol	TK3	n_sol
11	52	2	32	2	16	1	10	3
12	55	2	38	7	21	1	13	2
13	58	6	41	2	25	2	16	2
14	61	2	45	3	31	1	19	1

- Problem $\rightarrow$ Many solutions don't have a Step2 solution
- Increase the objective of Step1 $\rightarrow$ Too many solutions
- Proposed solution: Add extra variables to control instantiation probability

Conclusions

Conclusions

- Differential search can be simplified throughout differential characteristics search

Conclusions

- Differential search can be simplified throughout differential characteristics search
- Differential characteristic search can be improved with abstraction approach in two steps

Conclusions

- Differential search can be simplified throughout differential characteristics search
- Differential characteristic search can be improved with abstraction approach in two steps
- Both steps can be model as a MILP problem

Conclusions

- Differential search can be simplified throughout differential characteristics search
- Differential characteristic search can be improved with abstraction approach in two steps
- Both steps can be model as a MILP problem
- Clusters can make a considerable improvement in the approximation of the probability of a differential

-> Future Work

Conclusions

- Differential search can be simplified throughout differential characteristics search
- Differential characteristic search can be improved with abstraction approach in two steps
- Both steps can be model as a MILP problem
- Clusters can make a considerable improvement in the approximation of the probability of a differential

-> Future Work

- Need to improve Cluster Search

Conclusions

- Differential search can be simplified throughout differential characteristics search
- Differential characteristic search can be improved with abstraction approach in two steps
- Both steps can be model as a MILP problem
- Clusters can make a considerable improvement in the approximation of the probability of a differential

-> Future Work

- Need to improve Cluster Search
- Need fo better Step1 model for Skinny

Cluster Search and MILP Modeling for Differential Attacks

RODRÍGUEZ CORDERO Ana Margarita

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

October 27th, 2022