Radboud University

On round functions of permutations

Joan Daemen, Radboud University NL

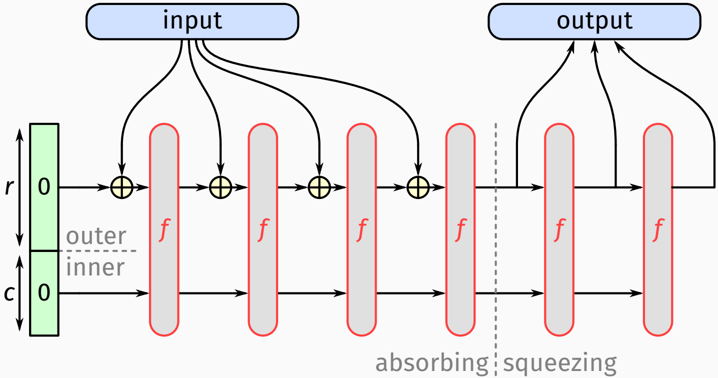
BFA, September 7, 2023

Voss, Norway

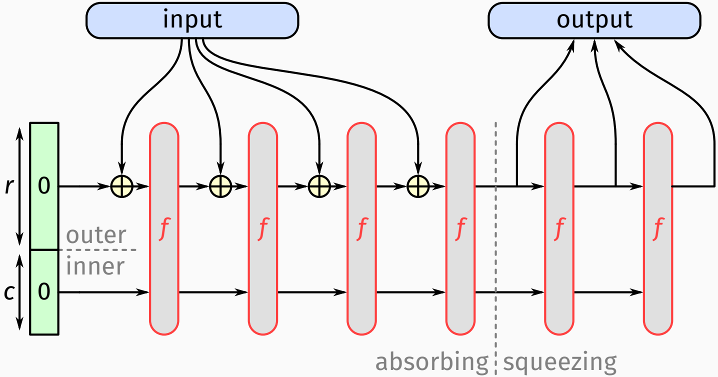
ESCADA

Permutation-based cryptography

Keccak (SHA-3) [Bertoni et al. 2007]

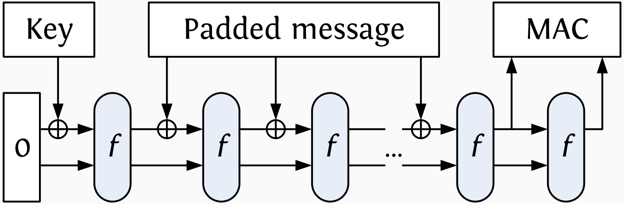


Keccak (SHA-3) [Bertoni et al. 2007]

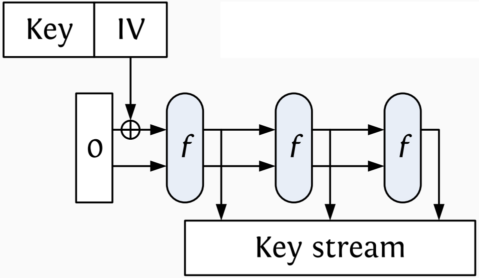


A priori for unkeyed hashing

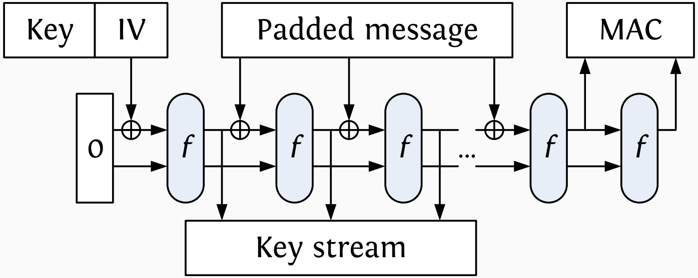
Mac computation with sponge



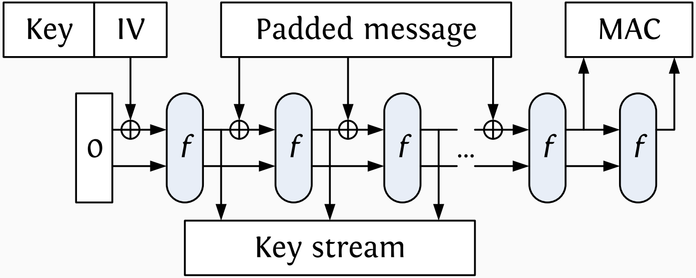
Stream encryption with sponge



Authenticated encryption with sponge/duplex [Bertoni et al., SAC 2011]

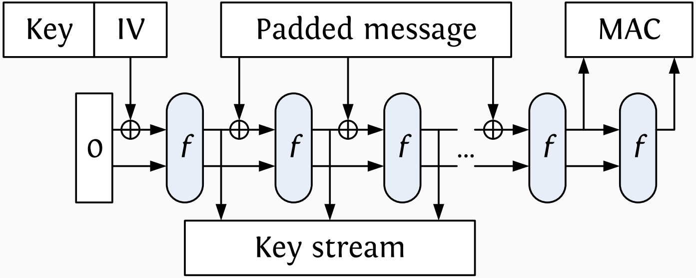


Authenticated encryption with sponge/duplex [Bertoni et al., SAC 2011]



Ideal for lightweight!

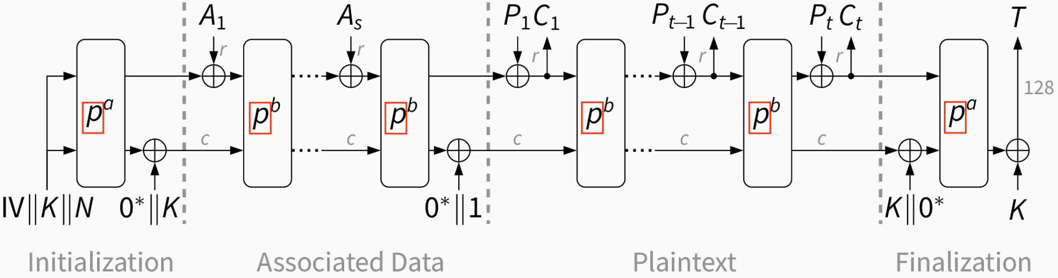
Authenticated encryption with sponge/duplex [Bertoni et al., SAC 2011]



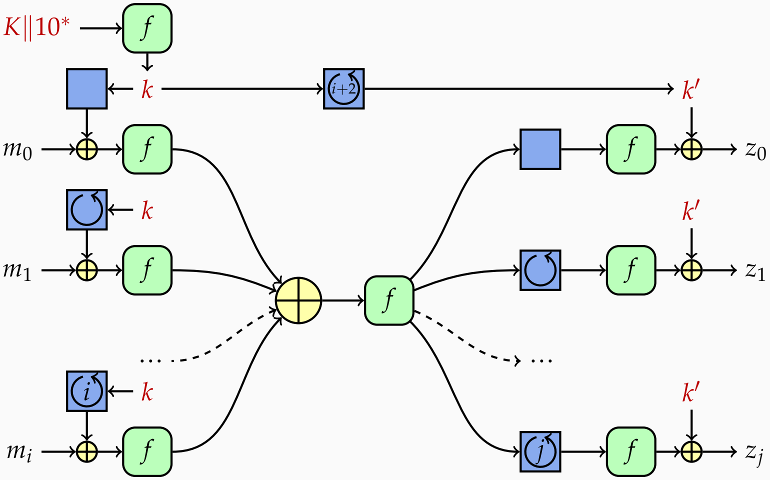
Ideal for lightweight!

Especially the variant MonkeyDuplex that we proposed in [Bertoni et al., DIAC 2012]

Ascon [Dobraunig et al. 2016]

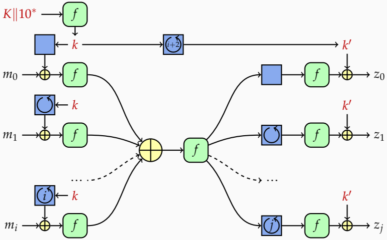


NIST's new standard for lightweight authenticated encryption!

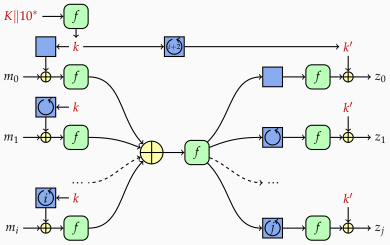


$K\|10^*$
f
k
$i+2$
k'
m_0
f
f
z_0
k
k'
m_1
f
f
f
z_1
$\cdots$
$\cdots$
i
k
k'
m_i
f
j
f
z_j

Farfalle construction [Bertoni et al., 2017]

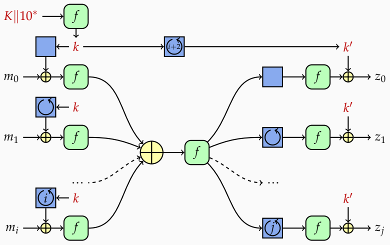


Farfalle construction [Bertoni et al., 2017]



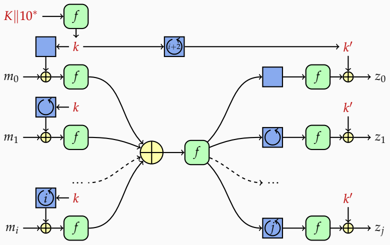
- Farfalle builds a deck function
  - A keyed primitive more versatile than a block cipher

Farfalle construction [Bertoni et al., 2017]



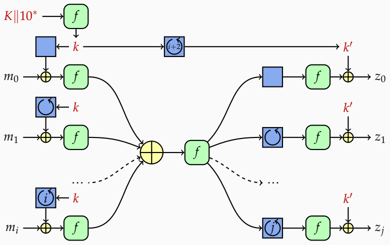
- Farfalle builds a deck function
  - A keyed primitive more versatile than a block cipher
  - For everything keyed, see "Jammin on the deck" [Băcuieți et al., Asiacrypt 2022]

Farfalle construction [Bertoni et al., 2017]



- Farfalle builds a deck function
  - A keyed primitive more versatile than a block cipher
  - For everything keyed, see "Jammin on the deck" [Băcuieți et al., Asiacrypt 2022]
- XOOFFF [Bertoni et al., 2018]
  - Farfalle with XOODOO permutation

Farfalle construction [Bertoni et al., 2017]



- Farfalle builds a deck function
  - A keyed primitive more versatile than a block cipher
  - For everything keyed, see "Jammin on the deck" [Băcuieți et al., Asiacrypt 2022]
- Xoofff [Bertoni et al., 2018]
  - Farfalle with Xoodoo permutation
  - Competitive with AES even on CPUs with AES-NI instruction

Ingredients that make permutation-based cryptography efficient

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
    - adversary cannot make *inverse* queries

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
    - adversary cannot make *inverse* queries
    - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
    - it feeds the output of a keyed compression straight into a stream cipher

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
    - adversary cannot make *inverse* queries
    - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
    - it feeds the output of a keyed compression straight into a stream cipher
    - input and output separated by 3 permutation layers

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
  - it feeds the output of a keyed compression straight into a stream cipher
  - input and output separated by 3 permutation layers
- Duplex-based authenticated encryption is *compact*

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
  - it feeds the output of a keyed compression straight into a stream cipher
  - input and output separated by 3 permutation layers
- Duplex-based authenticated encryption is *compact*
  - during operation no need for key storage

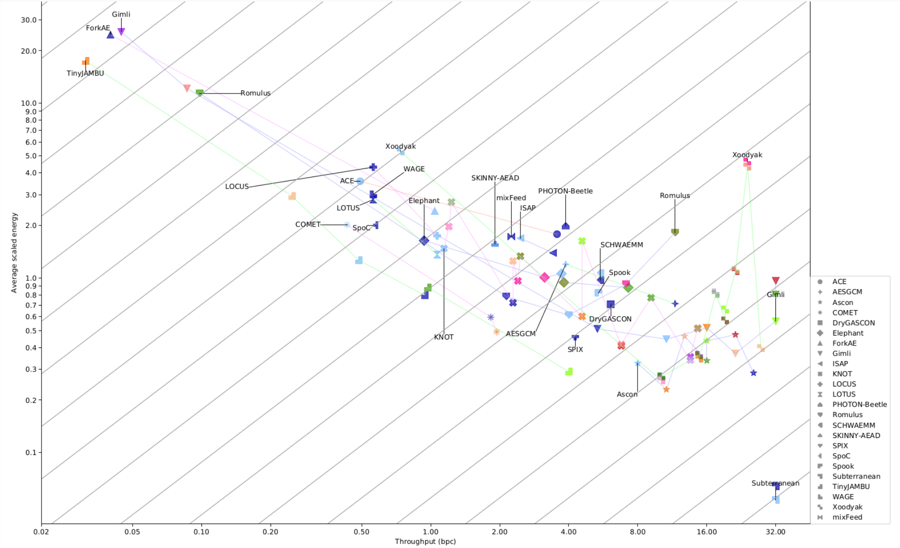
Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
  - it feeds the output of a keyed compression straight into a stream cipher
  - input and output separated by 3 permutation layers
- Duplex-based authenticated encryption is *compact*
  - during operation no need for key storage
  - in monkeyDuplex presence of nonce allows reducing # rounds after init

Ingredients that make permutation-based cryptography efficient

- Inverse permutation is not used
  - adversary cannot make *inverse* queries
  - more liberty in designing round function
- Farfalle is *computationally efficient* thanks to limiting exposure of permutation
  - it feeds the output of a keyed compression straight into a stream cipher
  - input and output separated by 3 permutation layers
- Duplex-based authenticated encryption is *compact*
  - during operation no need for key storage
  - in monkeyDuplex presence of nonce allows reducing # rounds after init
- Interesting hardware benchmarks related to lightweight:
  - https://eprint.iacr.org/2020/1207
  - https://eprint.iacr.org/2020/1459
  - https://eprint.iacr.org/2021/049

Taken from https://eprint.iacr.org/2021/049



Focus on the permutation

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area
- In software
  - main criterion: speed
  - ...on a wide range of CPUs
  - lightweight: RAM, code size, etc.

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area
- In software
  - main criterion: speed
  - ... on a wide range of CPUs
  - lightweight: RAM, code size, etc.
- When side-channel attacks are a threat

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area
- In software
  - main criterion: speed
  - ...on a wide range of CPUs
  - lightweight: RAM, code size, etc.
- When side-channel attacks are a threat
  - permutation should run in constant time

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area
- In software
  - main criterion: speed
  - ...on a wide range of CPUs
  - lightweight: RAM, code size, etc.
- When side-channel attacks are a threat
  - permutation should run in constant time
  - suitability for masking: low algebraic degree building blocks

Implementation properties required from the permutations

- In dedicated hardware
  - main criterion: energy-efficiency
  - achievable speed
  - lightweight: power-efficiency, area
- In software
  - main criterion: speed
  - ...on a wide range of CPUs
  - lightweight: RAM, code size, etc.
- When side-channel attacks are a threat
  - permutation should run in constant time
  - suitability for masking: low algebraic degree building blocks
- As opposed to block ciphers: no need for efficient inverse

Propagation properties required from a permutation (of $\mathbb{F}_2^n$)

Propagation properties required from a permutation (of $\mathbb{F}_2^n$)

Differential probability (DP) of a differential (a, b)

$$\mathrm{DP}(a, b) = \frac{\#\{x \in \mathbb{F}_2^n \mid f(x + a) + f(x) = b\}}{2^n}$$

Propagation properties required from a permutation (of $\mathbb{F}_2^n$)

Differential probability (DP) of a differential (a, b)

$$\mathrm{DP}(a, b) = \frac{\#\{x \in \mathbb{F}_2^n \mid f(x+a) + f(x) = b\}}{2^n}$$

Correlation and ***linear potential*** **(LP) of a linear approximation** (a, b)

$$\mathrm{C}(a, b) = \frac{\sum_{x \in \mathbb{F}_2^n} (-1)^{a^{\mathrm{T}} x + b^{\mathrm{T}} f(x)}}{2^n} \text{ and } \mathrm{LP}(a, b) = \mathrm{C}^2(a, b)$$

Propagation properties required from a permutation (of $\mathbb{F}_2^n$)

Differential probability (DP) of a differential (a, b)

$$\mathrm{DP}(a, b) = \frac{\#\{x \in \mathbb{F}_2^n \mid f(x + a) + f(x) = b\}}{2^n}$$

Correlation and ***linear potential*** **(LP) of a linear approximation** (a, b)

$$\mathrm{C}(a, b) = \frac{\sum_{x \in \mathbb{F}_2^n} (-1)^{a^{\mathrm{T}} x + b^{\mathrm{T}} f(x)}}{2^n} \text{ and } \mathrm{LP}(a, b) = \mathrm{C}^2(a, b)$$

LC DC requirements are of the following type:

$$\forall (a, b) \neq (0, 0) : \mathrm{DP}(a, b) < \text{ limit}$$
$$\forall (a, b) \neq (0, 0) : \mathrm{LP}(a, b) < \text{ limit}$$

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*
  - AKA higher order differentials, cube attacks, division property, . . .

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*
  - AKA higher order differentials, cube attacks, division property, ...
  - principle: summing the outputs corresponding to inputs in a large set V

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*
  - AKA higher order differentials, cube attacks, division property, ...
  - principle: summing the outputs corresponding to inputs in a large set V
  - Often V is an affine space

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*
  - AKA higher order differentials, cube attacks, division property, ...
  - principle: summing the outputs corresponding to inputs in a large set V
  - Often V is an affine space
  - used as a distinguisher

Propagation properties required from a permutation (cont'd)

- There are other propagation properties that play a role in certain attacks
- The most powerful in many scenario's are *summing attacks*
  - AKA higher order differentials, cube attacks, division property, ...
  - principle: summing the outputs corresponding to inputs in a large set V
  - Often V is an affine space
  - used as a distinguisher
  - or to harvest (linear) equations in unknown state bits

Requirements related to summing attacks are of the following type:

$$\forall V \subset \mathbb{F}_2^n \text{ such that } \forall x \in \mathbb{F}_2^n : \sum_{v \in V} f(x+v) = 0, |V| > \text{ limit}$$

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software
- There are different kinds of round functions
  - Feistel: function applied to one half, result added to other half and swap
  - generalized Feistel: multiple parts
  - Addition, Rotation, XOR (ARX), ...

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software
- There are different kinds of round functions
  - Feistel: function applied to one half, result added to other half and swap
  - generalized Feistel: multiple parts
  - Addition, Rotation, XOR (ARX), ...
  - symmetric, consisting of a non-linear layer and a linear layer

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software
- There are different kinds of round functions
  - Feistel: function applied to one half, result added to other half and swap
  - generalized Feistel: multiple parts
  - Addition, Rotation, XOR (ARX), ...
  - symmetric, consisting of a non-linear layer and a linear layer
- We assume the latter with $\mathrm{R} = \gamma \circ \lambda$

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software
- There are different kinds of round functions
  - Feistel: function applied to one half, result added to other half and swap
  - generalized Feistel: multiple parts
  - Addition, Rotation, XOR (ARX), ...
  - symmetric, consisting of a non-linear layer and a linear layer
- We assume the latter with $\mathrm{R} = \gamma \circ \lambda$
  - non-linear layer γ of identical S-boxes (we'll assume)

Iterated permutations

- Permutation as the repetition of a relatively simple round function
  - similar to block ciphers, e.g., DES, Rijndael
  - efficient in hardware but also software
- There are different kinds of round functions
  - Feistel: function applied to one half, result added to other half and swap
  - generalized Feistel: multiple parts
  - Addition, Rotation, XOR (ARX), ...
  - symmetric, consisting of a non-linear layer and a linear layer
- We assume the latter with $\mathrm{R} = \gamma \circ \lambda$
  - non-linear layer γ of identical S-boxes (we'll assume)
  - linear layer λ where $y = \lambda(x) = \mathrm{M}x + c$ (affine really)

Propagation over an iterated permutation

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathsf{DP}(a, b)$ or $\mathsf{LP}(a, b)$ directly is hard
- But over a single round computing $\mathsf{DP}(a, b)$ or $\mathsf{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathsf{DP}(Q)$ by $\mathsf{EDP}(Q) = \prod_i \mathsf{DP}(a^{i-1}, a^i)$

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)
  - then verify $\mathrm{DP}(Q) \approx \mathrm{EDP}(Q)$: hypothesis of stochastic equivalence

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)
  - then verify $\mathrm{DP}(Q) \approx \mathrm{EDP}(Q)$: hypothesis of stochastic equivalence
  - and check clustering of trails in differentials as $\mathrm{DP}(a, b) = \mathrm{DP}_{Q \in (a,b)}(Q)$

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)
  - then verify $\mathrm{DP}(Q) \approx \mathrm{EDP}(Q)$: hypothesis of stochastic equivalence
  - and check clustering of trails in differentials as $\mathrm{DP}(a, b) = \mathrm{DP}_{Q \in (a,b)}(Q)$
- General approach for correlation
  - correlation contribution of a trail is $\mathrm{C}(Q) = \prod_i \mathrm{C}(a^{i-1}, a^i)$

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)
  - then verify $\mathrm{DP}(Q) \approx \mathrm{EDP}(Q)$: hypothesis of stochastic equivalence
  - and check clustering of trails in differentials as $\mathrm{DP}(a, b) = \mathrm{DP}_{Q \in (a,b)}(Q)$
- General approach for correlation
  - correlation contribution of a trail is $\mathrm{C}(Q) = \prod_i \mathrm{C}(a^{i-1}, a^i)$
  - bound $\mathrm{LP}(Q)$ $(= \mathrm{C}^2(Q))$ (during the design effort)

Propagation over an iterated permutation

- For multi-round permutation f bounding $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ directly is hard
- But over a single round computing $\mathrm{DP}(a, b)$ or $\mathrm{LP}(a, b)$ is easy
- Round diff/approx $(a^0, a^1), (a^1, a^2), \ldots$ chain to *trails* $Q = (a^0, a^1, a^2, \ldots)$
- General approach for differential propagation
  - approximate trail $\mathrm{DP}(Q)$ by $\mathrm{EDP}(Q) = \prod_i \mathrm{DP}(a^{i-1}, a^i)$
  - bound $\mathrm{EDP}(Q)$ (often during the design effort)
  - then verify $\mathrm{DP}(Q) \approx \mathrm{EDP}(Q)$: hypothesis of stochastic equivalence
  - and check clustering of trails in differentials as $\mathrm{DP}(a, b) = \mathrm{DP}_{Q \in (a,b)}(Q)$
- General approach for correlation
  - correlation contribution of a trail is $\mathrm{C}(Q) = \prod_i \mathrm{C}(a^{i-1}, a^i)$
  - bound $\mathrm{LP}(Q)$ $(= \mathrm{C}^2(Q))$ (during the design effort)
  - then check clustering of trails as $\mathrm{C}(a, b) = \mathrm{C}_{Q \in (a,b)}(Q)$

Differential over a round function

For a differential (a^0, a^1) over R we have

$$\mathsf{DP}_{\mathrm{R}}(a^0, a^1) = \mathsf{DP}_{\gamma}(b^0, a^1) \text{ with } b^0 = \mathrm{M}a^0$$

Differential over a round function

For a differential (a^0, a^1) over R we have

$$\mathsf{DP}_{\mathrm{R}}(a^0, a^1) = \mathsf{DP}_{\gamma}(b^0, a^1) \text{ with } b^0 = \mathrm{M}a^0$$

We can further split $\mathsf{DP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \ldots b_{m-1}), a^1 = (a_0, a_1, \ldots a_{m-1})$

$$\mathsf{DP}_{\mathrm{R}}(a^0, a^1) = \mathsf{DP}_{\gamma}(b^0, a^1) = \prod_i \mathsf{DP}_S(b_i, a_i)$$

Differential over a round function

For a differential (a^0, a^1) over R we have

$$\mathrm{DP}_{\mathrm{R}}(a^0, a^1) = \mathrm{DP}_{\gamma}(b^0, a^1) \text{ with } b^0 = \mathrm{M}a^0$$

We can further split $\mathrm{DP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \ldots b_{m-1}), a^1 = (a_0, a_1, \ldots a_{m-1})$

$$\mathrm{DP}_{\mathrm{R}}(a^0, a^1) = \mathrm{DP}_{\gamma}(b^0, a^1) = \prod_i \mathrm{DP}_S(b_i, a_i)$$

Switching from DP to weight with: $2^{-w(a,b)} = \mathrm{DP}(a, b)$ makes it additive

$$w_{\mathrm{R}}(a^0, a^1) = \sum_i w(b_i, a_i) \text{ with } b^0 = \mathrm{M}a^0$$

Differential over a round function

For a differential (a^0, a^1) over R we have

$$\mathrm{DP}_{\mathrm{R}}(a^0, a^1) = \mathrm{DP}_{\gamma}(b^0, a^1) \text{ with } b^0 = \mathrm{M}a^0$$

We can further split $\mathrm{DP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \ldots b_{m-1}), a^1 = (a_0, a_1, \ldots a_{m-1})$

$$\mathrm{DP}_{\mathrm{R}}(a^0, a^1) = \mathrm{DP}_{\gamma}(b^0, a^1) = \prod_i \mathrm{DP}_S(b_i, a_i)$$

Switching from DP to weight with: $2^{-w(a,b)} = \mathrm{DP}(a, b)$ makes it additive

$$w_{\mathrm{R}}(a^0, a^1) = \sum_i w(b_i, a_i) \text{ with } b^0 = \mathrm{M}a^0$$

An S-box with zero input difference contributes 0 to the weight: it is *passive*.

Linear approximation over a round function

For a linear approximation (a^0, a^1) over R we have

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

Linear approximation over a round function

For a linear approximation (a^0, a^1) over R we have

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

We can further split $\mathsf{LP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \ldots b_{m-1}), a^1 = (a_0, a_1, \ldots a_{m-1})$

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) = \prod_i \mathsf{LP}_S(b_i, a_i)$$

Linear approximation over a round function

For a linear approximation (a^0, a^1) over R we have

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

We can further split $\mathsf{LP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \dots b_{m-1}), a^1 = (a_0, a_1, \dots a_{m-1})$

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) = \prod_i \mathsf{LP}_S(b_i, a_i)$$

Switching from LP to weight with: $2^{-w(a,b)} = \mathsf{LP}(a, b)$ makes it additive

$$w_{\mathrm{R}}(a^0, a^1) = \sum_i w(\Delta_{b_i}, \Delta_{a_i}) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

Linear approximation over a round function

For a linear approximation (a^0, a^1) over R we have

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

We can further split $\mathsf{LP}_{\gamma}(b^0, a^1)$ with $b^0 = (b_0, b_1, \ldots b_{m-1}), a^1 = (a_0, a_1, \ldots a_{m-1})$

$$\mathsf{LP}_{\mathrm{R}}(a^0, a^1) = \mathsf{LP}_{\gamma}(b^0, a^1) = \prod_i \mathsf{LP}_S(b_i, a_i)$$

Switching from LP to weight with: $2^{-w(a,b)} = \mathsf{LP}(a, b)$ makes it additive

$$w_{\mathrm{R}}(a^0, a^1) = \sum_i w(\Delta_{b_i}, \Delta_{a_i}) \text{ with } a^0 = \mathrm{M}^{\mathrm{T}} b^0$$

An S-box with zero output mask contributes 0 to the weight: it is *passive*.

Weight of trails

The weight of a differential trail is the sum of the weights of its active S-boxes

$$w(Q) = \sum_{i,r} w(b_i^{r-1}, a_i^r) \text{ with } b^i = \mathrm{M} a^i \quad \text{and} \quad \mathsf{DP}(Q) \approx \mathsf{EDP}(Q) = 2^{-w(Q)}$$

The weight of a linear trail is the sum of the weights of its active S-boxes

$$w(Q) = \sum_{i,r} w(b_i^{r-1}, a_i^r) \text{ with } a^i = \mathrm{M}^{\mathrm{T}} b^i \quad \text{and} \quad \mathsf{LP}(Q) = 2^{-w(Q)}$$

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes
  - wide trail: no, we need **more** active S-boxes (or non-linear operations)

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes
  - wide trail: no, we need **more** active S-boxes (or non-linear operations)
- Idea: round composed of three layers
  - non-linear layer operating locally
  - mixing layer operating locally
  - shuffle layer(s): moving nearby bits/cells away from each other
- Two flavours: aligned (or cell-oriented) and non-aligned (or bit-oriented)

Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes
  - wide trail: no, we need **more** active S-boxes (or non-linear operations)
- Idea: round composed of three layers
  - non-linear layer operating locally
  - mixing layer operating locally
  - shuffle layer(s): moving nearby bits/cells away from each other
- Two flavours: aligned (or cell-oriented) and non-aligned (or bit-oriented)
- Symmetry plays an important role

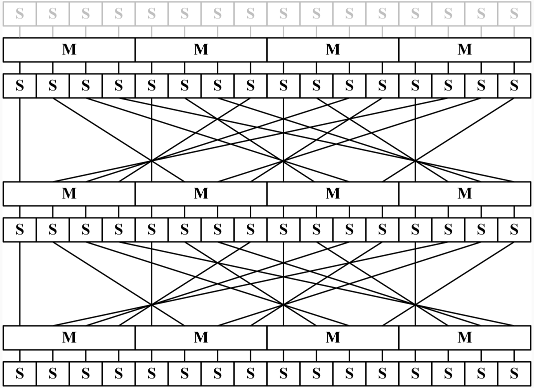
Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes
  - wide trail: no, we need **more** active S-boxes (or non-linear operations)
- Idea: round composed of three layers
  - non-linear layer operating locally
  - mixing layer operating locally
  - shuffle layer(s): moving nearby bits/cells away from each other
- Two flavours: aligned (or cell-oriented) and non-aligned (or bit-oriented)
- Symmetry plays an important role
  - leads to simple specification

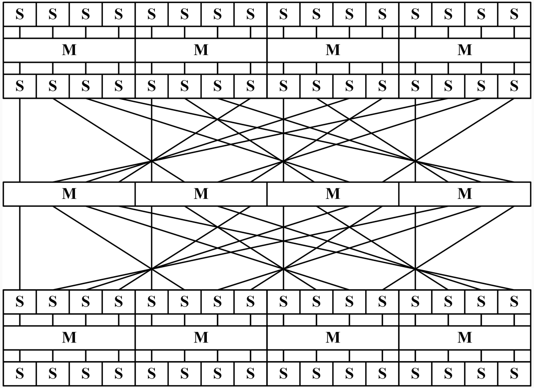
Wide trail strategy [Daemen, 1991-1993]

- Design strategy addressing resistance against LC and DC
  - reaction to DC/LC attacks on DES that made use of light trails
  - quasi consensus among cryptographers: we need **wider** S-boxes
  - wide trail: no, we need **more** active S-boxes (or non-linear operations)
- Idea: round composed of three layers
  - non-linear layer operating locally
  - mixing layer operating locally
  - shuffle layer(s): moving nearby bits/cells away from each other
- Two flavours: aligned (or cell-oriented) and non-aligned (or bit-oriented)
- Symmetry plays an important role
  - leads to simple specification
  - less *corners where weaknesses can hide*

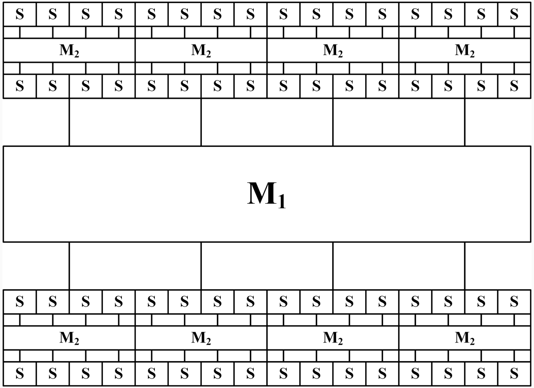
Example of cell-oriented wide trail design: Rijndael-128 [Daemen & Rijmen, 1998]



Example of cell-oriented wide trail design: Rijndael-128 [Daemen & Rijmen, 1998]

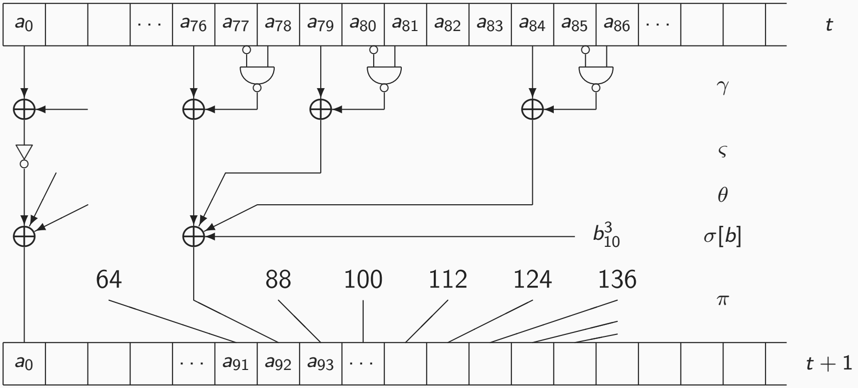


Example of cell-oriented wide trail design: Rijndael-128 [Daemen & Rijmen, 1998]

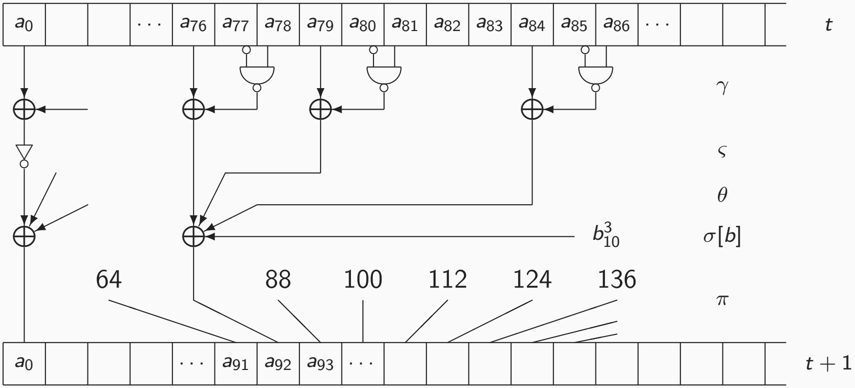


Thanks to *superboxes* proving any 4-round trail has at least 25 active S-boxes is easy!

Example of bit-oriented wide trail design: Subterranean [Daemen 1992]



Example of bit-oriented wide trail design: Subterranean [Daemen 1992]



Proving trail bounds requires computer-assisted search

Choice of the S-box

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)
  - low computational complexity

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)
  - low computational complexity
  - symmetry: as much as we can get

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)
  - low computational complexity
  - symmetry: as much as we can get
- Computational complexity

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)
  - low computational complexity
  - symmetry: as much as we can get
- Computational complexity
  - in hardware: # gate equivalent, circuit depth

S-box

- Permutation operating on $\mathbb{F}_2^n$ for some small n typically $\in \{3, 4, 5, 6, 8\}$
  - For block ciphers n was quasi always a power of 2
  - For permutations this is no longer required
- Wish list:
  - no differentials with high DP
  - no linear approximations with high LP
  - low degree (for protection against masking)
  - low computational complexity
  - symmetry: as much as we can get
- Computational complexity
  - in hardware: # gate equivalent, circuit depth
  - in bit-sliced software: number of bitwise Boolean operations

Let's generalize

We can generalize to transformations of $\mathbb{F}_{p^n}$ with p a prime

Let's generalize

We can generalize to transformations of $\mathbb{F}_{p^n}$ with p a prime

DP of differentials (a, b) of a transformation of $\mathbb{F}_{p^n}$

$$\mathrm{DP}(a, b) = \frac{\#\{x \mid f(x + a) - f(x) = b\}}{p^n}$$

Let's generalize

We can generalize to transformations of $\mathbb{F}_{p^n}$ with p a prime

DP of differentials (a, b) of a transformation of $\mathbb{F}_{p^n}$

$$\mathrm{DP}(a, b) = \frac{\#\{x \mid f(x+a) - f(x) = b\}}{p^n}$$

For correlation we need the trace function that maps $\mathbb{F}_{p^n}$ to $\mathbb{F}_p$: $\mathrm{Tr}(x) = \sum_{0 \leq i < n} x^{p^i}$

Let's generalize

We can generalize to transformations of $\mathbb{F}_{p^n}$ with p a prime

DP of differentials (a, b) of a transformation of $\mathbb{F}_{p^n}$

$$\mathrm{DP}(a, b) = \frac{\#\{x \mid f(x+a) - f(x) = b\}}{p^n}$$

For correlation we need the trace function that maps $\mathbb{F}_{p^n}$ to $\mathbb{F}_p$: $\mathrm{Tr}(x) = \sum_{0 \leq i < n} x^{p^i}$

Correlation and LP of linear approximations (a, b) of a transformation of $\mathbb{F}_{p^n}$

$$\mathrm{C}(a, b) = \frac{\sum_x \omega^{\mathrm{Tr}(ax - bf(x))}}{p^n} \text{ with } \omega = e^{\frac{2\pi i}{p}}$$

$$\mathrm{LP}(a, b) = \mathrm{C}(a, b)\overline{\mathrm{C}}(a, b)$$

Transformations in $\mathbb{F}_{p^n}$ with much symmetry:

Transformations in $\mathbb{F}_{p^n}$ with much symmetry: power functions

- Functions of the form $y \leftarrow x^e$
- Invertible if e is coprime to $p^n - 1$
- Invertible power functions form a group
  - isomorphic to $(\mathbb{Z}/(p^n - 1)\mathbb{Z})^*$
  - order is $\varphi(p^n - 1)$
- Inverse of $y \leftarrow x^e$ is $y \leftarrow x^d$ with $d = e^{-1} \bmod (p^n - 1)$

Symmetry in power functions

- Differentials and correlation:
  - $\mathrm{DP}(a, b) = p^{-n} \# \{x \mid (x+a)^e - x^e = b\}$
  - $\mathrm{C}(a, b) = p^{-n} \sum_x \omega^{\mathrm{Tr}(ax - bx^e)}$

Symmetry in power functions

- Differentials and correlation:
  - $\mathrm{DP}(a, b) = p^{-n} \#\{x \mid (x + a)^e - x^e = b\}$
  - $\mathrm{C}(a, b) = p^{-n} \sum_x \omega^{\mathrm{Tr}(ax - bx^e)}$
- Symmetry in propagation

Symmetry in power functions

- Differentials and correlation:
  - $\mathrm{DP}(a, b) = p^{-n} \#\{x \mid (x+a)^e - x^e = b\}$
  - $\mathrm{C}(a, b) = p^{-n} \sum_x \omega^{\mathrm{Tr}(ax - bx^e)}$
- Symmetry in propagation
  - $\mathrm{DP}(a, b) = \mathrm{DP}(1, ba^{-e}) = \mathrm{DP}(ab^{-d}, 1)$

- Differentials and correlation:
  - $\mathrm{DP}(a,b) = p^{-n}\#\{x \mid (x+a)^e - x^e = b\}$
  - $\mathrm{C}(a,b) = p^{-n}\sum_x \omega^{\mathrm{Tr}(ax-bx^e)}$
- Symmetry in propagation
  - $\mathrm{DP}(a,b) = \mathrm{DP}(1, ba^{-e}) = \mathrm{DP}(ab^{-d}, 1)$
  - $\mathrm{C}(a,b) = \mathrm{C}(1, ba^{-e}) = \mathrm{C}(ab^{-d}, 1)$

Symmetry in power functions

- Differentials and correlation:
  - $\mathrm{DP}(a, b) = p^{-n} \#\{x \mid (x+a)^e - x^e = b\}$
  - $\mathrm{C}(a, b) = p^{-n} \sum_x \omega^{\mathrm{Tr}(ax - bx^e)}$
- Symmetry in propagation
  - $\mathrm{DP}(a, b) = \mathrm{DP}(1, ba^{-e}) = \mathrm{DP}(ab^{-d}, 1)$
  - $\mathrm{C}(a, b) = \mathrm{C}(1, ba^{-e}) = \mathrm{C}(ab^{-d}, 1)$
- Power functions with $e = p^i$ are *linear*, giving additional symmetry

Symmetry in power functions

- Differentials and correlation:
    - $\mathrm{DP}(a,b) = p^{-n}\#\{x \mid (x+a)^e - x^e = b\}$
    - $\mathrm{C}(a,b) = p^{-n}\sum_x \omega^{\mathrm{Tr}(ax - bx^e)}$
- Symmetry in propagation
    - $\mathrm{DP}(a,b) = \mathrm{DP}(1, ba^{-e}) = \mathrm{DP}(ab^{-d}, 1)$
    - $\mathrm{C}(a,b) = \mathrm{C}(1, ba^{-e}) = \mathrm{C}(ab^{-d}, 1)$
- Power functions with $e = p^i$ are *linear*, giving additional symmetry
    - $\forall i < n : \mathrm{DP}(1, p^i b) = \mathrm{DP}(1, b)$
    - $\forall i < n : \mathrm{C}(1, p^i b) = \mathrm{C}(1, b)$

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \dots, \alpha^{p^{n-1}}\}$

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$
  - $(x^e)^p = x^{pe} = (x^p)^e$: power function gives a *shift-invariant S-box*

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$
  - $(x^e)^p = x^{pe} = (x^p)^e$: power function gives a *shift-invariant S-box*
- It also implies a partitioning of exponents in classes

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \dots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$
  - $(x^e)^p = x^{pe} = (x^p)^e$: power function gives a *shift-invariant S-box*
- It also implies a partitioning of exponents in classes
  - so $x \leftarrow x^{pe}$ is just $x \leftarrow x^e$ followed by a cyclic shift

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$
  - $(x^e)^p = x^{pe} = (x^p)^e$: power function gives a *shift-invariant S-box*
- It also implies a partitioning of exponents in classes
  - so $x \leftarrow x^{pe}$ is just $x \leftarrow x^e$ followed by a cyclic shift
  - cyclic shift can be absorbed in linear layer

Converting power functions in $\mathbb{F}_{p^n}$ to S-boxes in $\mathbb{F}_p^n$

- Converting from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p^n$ requires choice of a basis
- Normal basis: $\{\alpha, \alpha^p, \ldots, \alpha^{p^{n-1}}\}$
  - Raising to a power p^i in $\mathbb{F}_{p^n}$ corresponds to a cyclic coordinate shift in $\mathbb{F}_p^n$
  - $(x^e)^p = x^{pe} = (x^p)^e$: power function gives a *shift-invariant S-box*
- It also implies a partitioning of exponents in classes
  - so $x \leftarrow x^{pe}$ is just $x \leftarrow x^e$ followed by a cyclic shift
  - cyclic shift can be absorbed in linear layer
  - exponents $e, pe, p^2e \ldots$ are equivalent with respect to our analysis

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$
- exponents in $\{1, 2, 4, 8\}$ give linear power functions

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$
- exponents in $\{1, 2, 4, 8\}$ give linear power functions
- Let us take 14

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$
- exponents in $\{1, 2, 4, 8\}$ give linear power functions
- Let us take 14
  - additional symmetry: involution because $14^2 \bmod 15 = 1$ so $14 = -1$

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$
- exponents in $\{1, 2, 4, 8\}$ give linear power functions
- Let us take 14
  - additional symmetry: involution because $14^2 \bmod 15 = 1$ so $14 = -1$
  - represents the mapping that takes the multiplicative inverse and maps 0 to 0

Let's try building an invertible 4-bit S-box from a power function

- We take $p = 2, n = 4$
- (algebraic degree in $\mathbb{F}_2^n$ is Hamming weight of binary representation of e)
- $\varphi(2^4 - 1) = 8$ candidate exponents in two classes: $\{1, 2, 4, 8\}$ and $\{7, 14, 13, 11\}$
- exponents in $\{1, 2, 4, 8\}$ give linear power functions
- Let us take 14
  - additional symmetry: involution because $14^2 \bmod 15 = 1$ so $14 = -1$
  - represents the mapping that takes the multiplicative inverse and maps 0 to 0
- Multiplicative inverse mapping is often called the Kaisa S-box [Nyberg, EC '93]

DP-table (aka scaled DDT) of $y \leftarrow x^{-1}$ in $\mathbb{F}_2^4$

1/8	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	2					1	1		1		1		1	1	
2			1				1		2	1		1	1		1
3		1					1	1		1	1	1		2	
4					1	1		1		1			2	1	1
5				1		1		1	1		2	1			1
6	1			1	1	1	2			1		1			
7	1	1	1			2	1	1							1
8			1	1	1		1				1			1	2
9	1	2			1				1			1		1	1
a		1	1	1		1						2	1	1	
b	1		1		2			1			1	1	1		
c		1	1		1	1			1	2	1				
d	1	1		2						1	1		1		1
e	1		2	1				1	1	1				1	
f		1		1	1		1	2	1				1		

DP-table of $x \leftarrow x^{-1}$ in $\mathbb{F}_2^4$, reordered

1/8	1	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
1	2					1		1			1	1		1	1
α					1		1			1	1		1	1	2
α^2				1		1			1	1		1	1	2	
α^3			1		1			1	1		1	1	2		
α^4		1		1			1	1		1	1	2			
α^5	1		1			1	1		1	1	2				
α^6		1			1	1		1	1	2					1
α^7	1			1	1		1	1	2					1	
α^8			1	1		1	1	2					1		1
α^9		1	1		1	1	2					1		1	
α^{10}	1	1		1	1	2					1		1		
α^{11}	1		1	1	2					1		1			1
α^{12}		1	1	2					1		1			1	1
α^{13}	1	1	2					1		1			1	1	
α^{14}	1	2					1		1			1	1		1

Correlation matrix (aka scaled LAT) of $x \leftarrow x^{-1}$ in $\mathbb{F}_2^4$

1/4	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	-1		1		1		-1	-1		-1	2	1		1	2
2				1	1	-1	-1		2	2		1	-1	1	-1
3	1		-1	-1	2	1	2	-1		1			-1		1
4		1	-1			1	-1			1	-1	2	2	-1	1
5	1	1	2		-1	1		-1	2		-1	-1			1
6		-1	1	1	1	2		2		-1	-1	1	-1		
7	-1	-1	2	-1			1	1		2	1		1	-1	
8	-1		-1		-1	2	1		1		1		1	2	-1
9		2			2			1	1	-1	1	-1	1	-1	-1
a	-1	2	1	1		-1	2		-1		-1	1		1	
b	2			-1	-1	-1	1	1	1	-1	1	2			
c	1	1		2	-1	1			-1	1	2		-1	-1	
d		-1	-1	2		-1	1	1	1			-1	1		2
e	1	1		-1			-1	2	-1	1		-1		2	1
f	2	-1	1	1	1			-1	-1				2	1	-1

Correlation matrix of $x \leftarrow x^{-1}$ in $\mathbb{F}_2^4$, reordered

1/4	1	β	β^2	β^3	β^4	β^5	β^6	β^7	β^8	β^9	β^{10}	β^{11}	β^{12}	β^{13}	β^{14}
1		-1	-1	1	-1	2	1		-1	1	2		1		
β	-1	-1	1	-1	2	1		-1	1	2		1			
β^2	-1	1	-1	2	1		-1	1	2		1				-1
β^3	1	-1	2	1		-1	1	2		1				-1	-1
β^4	-1	2	1		-1	1	2		1				-1	-1	1
β^5	2	1		-1	1	2		1				-1	-1	1	-1
β^6	1		-1	1	2		1				-1	-1	1	-1	2
β^7		-1	1	2		1				-1	-1	1	-1	2	1
β^8	-1	1	2		1				-1	-1	1	-1	2	1	
β^9	1	2		1				-1	-1	1	-1	2	1		-1
β^{10}	2		1				-1	-1	1	-1	2	1		-1	1
β^{11}		1				-1	-1	1	-1	2	1		-1	1	2
β^{12}	1				-1	-1	1	-1	2	1		-1	1	2	
β^{13}				-1	-1	1	-1	2	1		-1	1	2		1
β^{14}			-1	-1	1	-1	2	1		-1	1	2		1	

Now let's try building a 5-bit S-box

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
    - 1 is linear

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
  - 1 is linear
  - $30 = -1$: multiplicative inverse

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
    - 1 is linear
    - $30 = -1$: multiplicative inverse
    - 3 and 21 are each other's inverses

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
  - 1 is linear
  - $30 = -1$: multiplicative inverse
  - 3 and 21 are each other's inverses
  - 5 and 25 are each other's inverses
- Multiplicative inverse:

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
  - 1 is linear
  - $30 = -1$: multiplicative inverse
  - 3 and 21 are each other's inverses
  - 5 and 25 are each other's inverses
- Multiplicative inverse:
  - $\mathrm{DP}(a, b) = 2^{-4}$ if $\mathrm{Tr}((ab)^{-1}) = 0$ or $b = a^{-1}$ and 0 otherwise

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
  - 1 is linear
  - $30 = -1$: multiplicative inverse
  - 3 and 21 are each other's inverses
  - 5 and 25 are each other's inverses
- Multiplicative inverse:
  - $\mathrm{DP}(a, b) = 2^{-4}$ if $\mathrm{Tr}((ab)^{-1}) = 0$ or $b = a^{-1}$ and 0 otherwise
  - correlation matrix: $\mathrm{C}(a, b) = 2^{-3}x$ with $x \in \{-2, -1, 0, 1, 2, 3\}$ (as found in [Carlet et al., 2010])

Now let's try building a 5-bit S-box

- $\varphi(2^5 - 1) = 30$ candidate exponents in six classes
- Classes represented by $\{1, 3, 21, 5, 25, 30\}$
    - 1 is linear
    - $30 = -1$: multiplicative inverse
    - 3 and 21 are each other's inverses
    - 5 and 25 are each other's inverses
- Multiplicative inverse:
    - $\mathrm{DP}(a, b) = 2^{-4}$ if $\mathrm{Tr}((ab)^{-1}) = 0$ or $b = a^{-1}$ and 0 otherwise
    - correlation matrix: $\mathrm{C}(a, b) = 2^{-3}x$ with $x \in \{-2, -1, 0, 1, 2, 3\}$ (as found in [Carlet et al., 2010])
    - DP table has 16 non-zero entries per row, correlation matrix 26

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ab^{-d}) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ab^{-d}) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise
- Due to the linearity of the trace function:
  - Output diff b compatible with input diff a form an affine space
  - Input masks a compatible with output mask b form an affine space

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ab^{-d}) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise
- Due to the linearity of the trace function:
  - Output diff b compatible with input diff a form an affine space
  - Input masks a compatible with output mask b form an affine space
  - **All** valid differentials and approximations (a, b) have weight $n - 1$

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ab^{-d}) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise
- Due to the linearity of the trace function:
  - Output diff b compatible with input diff a form an affine space
  - Input masks a compatible with output mask b form an affine space
  - **All** valid differentials and approximations (a, b) have weight $n - 1$
- Still, in general
  - Input diff a compatible with output diff b form no affine space
  - Output masks b compatible with input mask a form no affine space

Gold functions

- Exponents $e = 3$ and $e = 5$ give S-boxes with algebraic degree 2 in $\mathbb{F}_2^n$
- Power functions with exponents $e = 2^i + 1$ are called Gold functions [Gold '68]
- For odd n they have (as found in [Carlet et al., 2010])
  - $\mathrm{DP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ba^{-e}) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{1-n}$ if $\mathrm{Tr}(ab^{-d}) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise
- Due to the linearity of the trace function:
  - Output diff b compatible with input diff a form an affine space
  - Input masks a compatible with output mask b form an affine space
  - **All** valid differentials and approximations (a, b) have weight $n - 1$
- Still, in general
  - Input diff a compatible with output diff b form no affine space
  - Output masks b compatible with input mask a form no affine space
  - Propagation of masks follows a different rule than propagation of differences

Can we do better than Gold?

- It would be great if
  - propagation of masks follows the same rule than propagation of differences
  - forward and backward propagation would be the same or at least similar

Can we do better than Gold?

- It would be great if
  - propagation of masks follows the same rule than propagation of differences
  - forward and backward propagation would be the same or at least similar
- Taking $n = 3$ and $d = e = 6 = -1$ results in a Golden Kaisa function
  - $\mathrm{DP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise

Can we do better than Gold?

- It would be great if
  - propagation of masks follows the same rule than propagation of differences
  - forward and backward propagation would be the same or at least similar
- Taking $n = 3$ and $d = e = 6 = -1$ results in a Golden Kaisa function
  - $\mathsf{DP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathsf{DP}(a, b) = 0$ otherwise
  - $\mathsf{LP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathsf{LP}(a, b) = 0$ otherwise
- $y \leftarrow x^6$ in $\mathbb{F}_{2^3}$ has following properties
  - forward, backward, differential and linear propagation are all the same
  - compatible masks/differences form affine spaces

Can we do better than Gold?

- It would be great if
  - propagation of masks follows the same rule than propagation of differences
  - forward and backward propagation would be the same or at least similar
- Taking $n = 3$ and $d = e = 6 = -1$ results in a Golden Kaisa function
  - $\mathsf{DP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathsf{DP}(a, b) = 0$ otherwise
  - $\mathsf{LP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathsf{LP}(a, b) = 0$ otherwise
- $y \leftarrow x^6$ in $\mathbb{F}_{2^3}$ has following properties
  - forward, backward, differential and linear propagation are all the same
  - compatible masks/differences form affine spaces
- This works for no other size n or exponent e!

Can we do better than Gold?

- It would be great if
  - propagation of masks follows the same rule than propagation of differences
  - forward and backward propagation would be the same or at least similar
- Taking $n = 3$ and $d = e = 6 = -1$ results in a Golden Kaisa function
  - $\mathrm{DP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathrm{DP}(a, b) = 0$ otherwise
  - $\mathrm{LP}(a, b) = 2^{-2}$ if $\mathrm{Tr}(ab) = 1$ and $\mathrm{LP}(a, b) = 0$ otherwise
- $y \leftarrow x^6$ in $\mathbb{F}_{2^3}$ has following properties
  - forward, backward, differential and linear propagation are all the same
  - compatible masks/differences form affine spaces
- This works for no other size n or exponent e!
- When choosing the normal basis, $\mathrm{Tr}(ab) = 1$ translates to $a_0 b_0 + a_1 b_1 + a_2 b_2 = 1$

On the Golden Kaisa S-box

- Limitation is mostly its width of 3

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width
- Rijndael S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width
- Rijndael S-box
  - $w \geq 6$

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width
- Rijndael S-box
  - $w \geq 6$
  - 10 xor and 4 and/or per bit [Boyar & Peralta 2010][Stoffelen & Schwabe 2016]

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width
- Rijndael S-box
  - $w \geq 6$
  - 10 xor and 4 and/or per bit [Boyar & Peralta 2010][Stoffelen & Schwabe 2016]
- Golden Kaisa S-box
  - $w = 2$

On the Golden Kaisa S-box

- Limitation is mostly its width of 3
  - due to small size its diff/approx have weight of only $w = 2$
  - implies permutation width has to be a multiple of 3
- What if we want to use larger S-boxes?
  - for odd size there is *more choice* than for even
  - computational cost of power functions increases sharply with width
- Rijndael S-box
  - $w \geq 6$
  - 10 xor and 4 and/or per bit [Boyar & Peralta 2010][Stoffelen & Schwabe 2016]
- Golden Kaisa S-box
  - $w = 2$
  - $y_i \leftarrow x_i + (x_{i+1 \bmod 3} + 1)x_{i+2 \bmod 3}$: costs 1 xor and 1 and per bit

A family of S-boxes χ_n [Daemen, WIC 1990]

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1) a_{i+2 \bmod n}$$

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1) a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1) a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$
  - backwards propagation does not give affine spaces

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$
  - backwards propagation does not give affine spaces
  - weight increases with Hamming weight of differences/masks

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$
  - backwards propagation does not give affine spaces
  - weight increases with Hamming weight of differences/masks
  - correlation matrix has fewer zeroes than DP table

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$
  - backwards propagation does not give affine spaces
  - weight increases with Hamming weight of differences/masks
  - correlation matrix has fewer zeroes than DP table
- Popular choice is $n = 5$: Keccak and Ascon

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1)a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
    - inverse is more complex and has higher degree: $(n+1)/2$
    - backwards propagation does not give affine spaces
    - weight increases with Hamming weight of differences/masks
    - correlation matrix has fewer zeroes than DP table
- Popular choice is $n = 5$: Keccak and Ascon
- Extreme: in Subterranean $n = 257$

A family of S-boxes χ_n [Daemen, WIC 1990]

- The Golden Kaisa S-box can be generalized to any width n and is called χ_n

$$\forall i : b_i = a_i + (a_{i+1 \bmod n} + 1) a_{i+2 \bmod n}$$

- Invertible if n is odd
- For $n > 3$ a lot of symmetry is lost
  - inverse is more complex and has higher degree: $(n+1)/2$
  - backwards propagation does not give affine spaces
  - weight increases with Hamming weight of differences/masks
  - correlation matrix has fewer zeroes than DP table
- Popular choice is $n = 5$: Keccak and Ascon
- Extreme: in Subterranean $n = 257$
- Excellent trade-off between implementation cost and non-linearity

The linear layer

Wide trail strategy

- linear layer split in mixing layer and shuffle

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells
  - shuffle moves cells to different super-cells

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells
  - shuffle moves cells to different super-cells
  - analysis and specification is natural at the cell level

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells
  - shuffle moves cells to different super-cells
  - analysis and specification is natural at the cell level
- Bit-oriented (non-aligned)

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells
  - shuffle moves cells to different super-cells
  - analysis and specification is natural at the cell level
- Bit-oriented (non-aligned)
  - three layers partition the statebits in different ways

Wide trail strategy

- linear layer split in mixing layer and shuffle
- Cell-oriented (aligned)
  - state is an array of cells, defined by the S-box layer: typically bytes or nibbles
  - mixlayer operates on super-cells: sub-arrays of cells
  - shuffle moves cells to different super-cells
  - analysis and specification is natural at the cell level
- Bit-oriented (non-aligned)
  - three layers partition the statebits in different ways
  - analysis and specification is natural at the bit level

Square approach [Daemen & Rijmen 1997]

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
  - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
    - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix
    - Matrix is MDS, has branch number 5: at least 5 active bytes before and after

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
  - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix
  - Matrix is MDS, has branch number 5: at least 5 active bytes before and after
  - Matrix symmetry: it is circulant

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
  - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix
  - Matrix is MDS, has branch number 5: at least 5 active bytes before and after
  - Matrix symmetry: it is circulant
  - Transformation symmetry: multi-permutation where any 4 entries out of 8 can be chosen

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
  - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix
  - Matrix is MDS, has branch number 5: at least 5 active bytes before and after
  - Matrix symmetry: it is circulant
  - Transformation symmetry: multi-permutation where any 4 entries out of 8 can be chosen
- Shuffle moving bytes of a column to different columns: here transposing the array

Square approach [Daemen & Rijmen 1997]

- State as a 4 by 4 array of bytes
- Powerful S-box operating on individual bytes
- Mixing layer operating in parallel on each 4-byte column
  - Multiplication in $\mathbb{F}_{2^8}$ with a 4×4 matrix
  - Matrix is MDS, has branch number 5: at least 5 active bytes before and after
  - Matrix symmetry: it is circulant
  - Transformation symmetry: multi-permutation where any 4 entries out of 8 can be chosen
- Shuffle moving bytes of a column to different columns: here transposing the array
- Easy to prove that any 4-round trail has at least 25 active S-boxes

Generalizations of and advances in the square approach

Generalizations of and advances in the square approach

Function	shape	cells	width	type
Rijndael [Daemen & Rijmen, 1998]	$4 \times (4 \text{ to } 8)$	bytes	128 to 256	block
Whirlpool [Rijmen & Barreto, 2000]	8^2	bytes	512	block
Groestl [Rechberger et al., 2008]	8^2	bytes	512	perm
ECHO [Benadjila et al., 2008]	4^4	bytes	2048	perm
JH [Wu, 2008]	2^8	nibbles	1024	perm
Primates [Andreeva et al., 2014]	$(5 \text{ or } 7) \times 8$	5-bit	200 or 280	perm
Saturnin [Canteaut et al., 2019]	4^3	nibbles	256	block

Generalizations of and advances in the square approach

Function	shape	cells	width	type
Rijndael [Daemen & Rijmen, 1998]	$4 \times (4 \text{ to } 8)$	bytes	128 to 256	block
Whirlpool [Rijmen & Barreto, 2000]	8^2	bytes	512	block
Groestl [Rechberger et al., 2008]	8^2	bytes	512	perm
ECHO [Benadjila et al., 2008]	4^4	bytes	2048	perm
JH [Wu, 2008]	2^8	nibbles	1024	perm
Primates [Andreeva et al., 2014]	$(5 \text{ or } 7) \times 8$	5-bit	200 or 280	perm
Saturnin [Canteaut et al., 2019]	4^3	nibbles	256	block

Advances in building efficient MDS matrices: +60 publications at crypto venues

Generalizations of and advances in the square approach

Function	shape	cells	width	type
Rijndael [Daemen & Rijmen, 1998]	$4 \times (4 \text{ to } 8)$	bytes	128 to 256	block
Whirlpool [Rijmen & Barreto, 2000]	8^2	bytes	512	block
Groestl [Rechberger et al., 2008]	8^2	bytes	512	perm
ECHO [Benadjila et al., 2008]	4^4	bytes	2048	perm
JH [Wu, 2008]	2^8	nibbles	1024	perm
Primates [Andreeva et al., 2014]	$(5 \text{ or } 7) \times 8$	5-bit	200 or 280	perm
Saturnin [Canteaut et al., 2019]	4^3	nibbles	256	block

Advances in building efficient MDS matrices: +60 publications at crypto venues

- mostly focusing on 4×4 matrices operating on bytes or nibbles

Generalizations of and advances in the square approach

Function	shape	cells	width	type
Rijndael [Daemen & Rijmen, 1998]	$4 \times (4 \text{ to } 8)$	bytes	128 to 256	block
Whirlpool [Rijmen & Barreto, 2000]	8^2	bytes	512	block
Groestl [Rechberger et al., 2008]	8^2	bytes	512	perm
ECHO [Benadjila et al., 2008]	4^4	bytes	2048	perm
JH [Wu, 2008]	2^8	nibbles	1024	perm
Primates [Andreeva et al., 2014]	$(5 \text{ or } 7) \times 8$	5-bit	200 or 280	perm
Saturnin [Canteaut et al., 2019]	4^3	nibbles	256	block

Advances in building efficient MDS matrices: +60 publications at crypto venues

- mostly focusing on 4×4 matrices operating on bytes or nibbles
- goal: reduce the total xor count or xor depth

Generalizations of and advances in the square approach

Function	shape	cells	width	type
Rijndael [Daemen & Rijmen, 1998]	$4 \times (4 \text{ to } 8)$	bytes	128 to 256	block
Whirlpool [Rijmen & Barreto, 2000]	8^2	bytes	512	block
Groestl [Rechberger et al., 2008]	8^2	bytes	512	perm
ECHO [Benadjila et al., 2008]	4^4	bytes	2048	perm
JH [Wu, 2008]	2^8	nibbles	1024	perm
Primates [Andreeva et al., 2014]	$(5 \text{ or } 7) \times 8$	5-bit	200 or 280	perm
Saturnin [Canteaut et al., 2019]	4^3	nibbles	256	block

Advances in building efficient MDS matrices: +60 publications at crypto venues

- mostly focusing on 4×4 matrices operating on bytes or nibbles
- goal: reduce the total xor count or xor depth
- insight: cost increases sharply with MDS matrix dimension

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$
  - Cost $2,25$ xor per bit instead of 3 xor per bit for AES

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$
  - Cost $2,25$ xor per bit instead of 3 xor per bit for AES
- Global structure

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$
  - Cost $2,25$ xor per bit instead of 3 xor per bit for AES
- Global structure
  - Cube with side 4 of elements of $\mathbb{F}_{2^4}$

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$
  - Cost $2,25$ xor per bit instead of 3 xor per bit for AES
- Global structure
  - Cube with side 4 of elements of $\mathbb{F}_{2^4}$
  - Two different shuffles in rounds with index $1 \bmod 4$ and $3 \bmod 4$

The best square-approach design: the block cipher Saturnin [Canteaut et al., 2019]

- S-box
  - 4-bit S-box instead of 8-bit one
  - no power function but $w \geq 2$
  - cost $1,5$ xor plus $1,5$ and/or per bit instead of 11 xor plus 4 and/or for AES
- MDS matrix in mixlayer
  - 4×4 operating on elements of $\mathbb{F}_{2^4}$
  - Cost $2,25$ xor per bit instead of 3 xor per bit for AES
- Global structure
  - Cube with side 4 of elements of $\mathbb{F}_{2^4}$
  - Two different shuffles in rounds with index $1 \bmod 4$ and $3 \bmod 4$
  - Any 8-round trail has at least $5^3 = 125$ active S-boxes

Showdown Saturnin vs AES (not counting round key addition)

Showdown Saturnin vs AES (not counting round key addition)

AES

# rounds	cost xor	cost and/or	min. trail weight
1	14	4	6
2	28	8	30
3	42	12	56
4	56	16	150

Saturnin

# rounds	cost xor	cost and/or	min. trail weight
1	3,75	1,5	2
2	7,5	3	10
3	11,25	4,5	18
4	15	6	50
5	18,75	7,5	82
6	22,5	9	90
7	26,25	10,5	122
8	30	12	250

Limitations of the Square approach

Limitations of the Square approach

Hourglass trail profile effect:

Limitations of the Square approach

Hourglass trail profile effect:

- You can have light S-boxes or light MDS matrices or few rounds

Limitations of the Square approach

Hourglass trail profile effect:

- You can have light S-boxes or light MDS matrices or few rounds
- . . . but not all at the same time

Limitations of the Square approach

Hourglass trail profile effect:

- You can have light S-boxes or light MDS matrices or few rounds
- . . . but not all at the same time
- This is because sparse states propagate to sparse in both directions

Limitations of the Square approach

Hourglass trail profile effect:

- You can have light S-boxes or light MDS matrices or few rounds
- ...but not all at the same time
- This is because sparse states propagate to sparse in both directions
- Example: $16 - 4 - 1 - 4 - 16$ profile

Limitations of the Square approach

Hourglass trail profile effect:

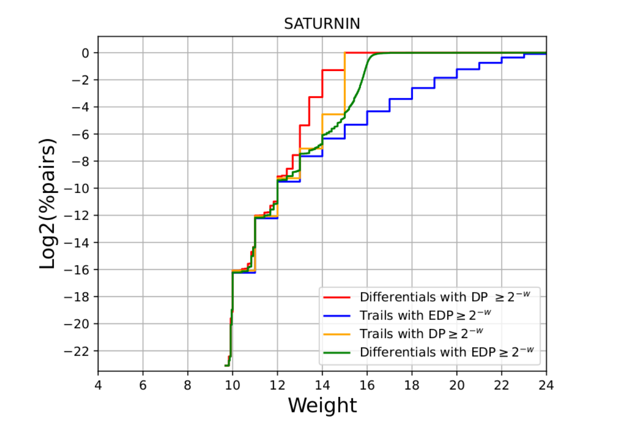
- You can have light S-boxes or light MDS matrices or few rounds
- . . . but not all at the same time
- This is because sparse states propagate to sparse in both directions
- Example: $16 - 4 - 1 - 4 - 16$ profile

Clustering and *clipping* in the AES superbox:

- massive clustering of trails in differentials [Daemen & Rijmen, SCN 2006]
- clipping: DP(Q) strongly deviates from EDP(Q) for most trails [Daemen & Rijmen, IET 2007]

Clustering and clipping in the Saturnin superbox, illustrated

Much less clustering and clipping than in AES thanks to smaller S-box, still significant



graph courtesy of Giovanni Uchua de Assis

Bit-oriented mixing

- In general it is just a binary matrix $\mathbf{M}$
  - operating on the full state, or
  - operating in parallel on parts of the state

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M circulant

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M circulant
- Polynomial representation of input, output and matrix

$$b(X) \leftarrow \theta(X)a(X) \bmod 1 + X^m$$

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M circulant
- Polynomial representation of input, output and matrix

$$b(X) \leftarrow \theta(X)a(X) \bmod 1 + X^m$$

- Invertible if $\theta(X)$ is coprime to $1 + X^m$

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M circulant
- Polynomial representation of input, output and matrix

$$b(X) \leftarrow \theta(X)a(X) \bmod 1 + X^m$$

- Invertible if $\theta(X)$ is coprime to $1 + X^m$
- Often one takes a multiplication polynomial that is a trinomial

Bit-oriented mixing

- In general it is just a binary matrix M
  - operating on the full state, or
  - operating in parallel on parts of the state
- Symmetry: make M circulant
- Polynomial representation of input, output and matrix

$$b(X) \leftarrow \theta(X)a(X) \bmod 1 + X^m$$

- Invertible if $\theta(X)$ is coprime to $1 + X^m$
- Often one takes a multiplication polynomial that is a trinomial
- Unless carefully chosen, inverse of $\theta(X)$ is dense
  - no problem if the inverse of the permutation is not needed
  - has an advantage for trail bounds

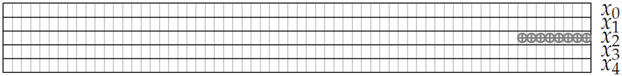
Some primitives using bit-oriented circulant mixing

Function	length	# t	non-lin.	b	shuffle
Cellhash [Daemen, AC 1991]	257	3	χ_{257}	257	multiplicative
3Way [Daemen, 1993]	12	7	χ_3	96	2 row shift steps
BaseKing [Daemen, 1994]	12	7	χ_3	192	2 row shift steps
Panama [Daemen & Clapp, 1997]	17	3	χ_{17}	544	1 row shift step
SHA-256 [NIST, 2001]	32	3	ARX	256	-
SHA-512 [NIST. 2001]	64	3	ARX	512	-
RadioGatun [Bertoni et al., 2006]	19	3	χ_{19}	608	1 row shift step
Ascon [Dobraunig et al., 2019]	64	3	χ_5+	320	different $m(x)$

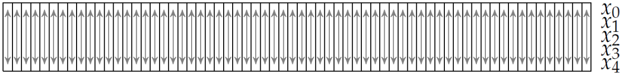
Ascon-p Round function

Ascon-p Round function

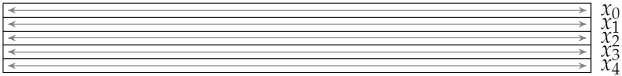
- 320-bit state: 5 rows $x_0, \ldots, x_4$ and 64 columns
- Round function $\mathrm{R} = p_L \circ p_S \circ p_C$



(a) Round constant addition p_C



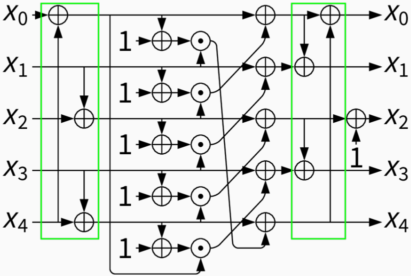
(b) Substitution layer p_S with 5-bit S-box $\mathcal{S}(x)$



(c) Linear layer with 64-bit diffusion functions $\Sigma_i(x_i)$

figure by Ascon team

Operations dedicated to mixing in Ascon-p



6 bitwise XOR

$$
\begin{aligned}
x_0 &\leftarrow x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
x_1 &\leftarrow x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
x_2 &\leftarrow x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6) \\
x_3 &\leftarrow x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
x_4 &\leftarrow x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}
$$

10 bitwise XOR + 10 cyclic shifts

Showdown Ascon-p vs Saturnin (not counting round key/constant addition)

Showdown Ascon-p vs Saturnin (not counting round key/constant addition)

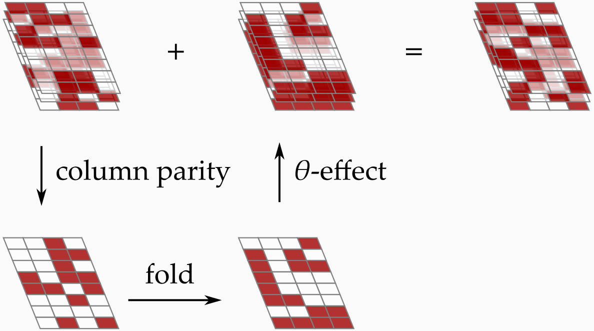
Saturnin

# rounds	cost xor	cost and/or	min. trail weight
1	3,75	1,5	2
2	7,5	3	10
3	11,25	4,5	18
4	15	6	50
5	18,75	7,5	82
6	22,5	9	90
7	26,25	10,5	122
8	30	12	250

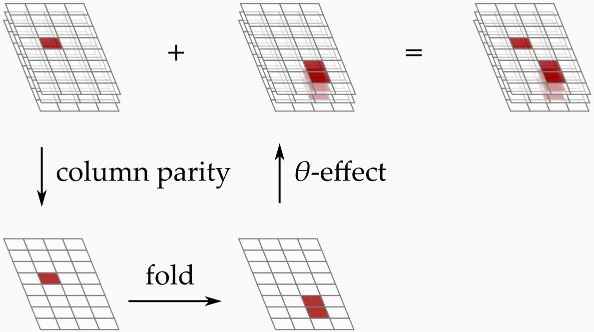
Ascon

# rounds	cost xor	cost and/or	min. trail weight diff	min. trail weight lin
1	4,2	1	2	2
2	8,4	2	8	8
3	12,6	3	40	28
4	16,8	4	≥ 86	≥ 88
5	21	5	≥ 100	≥ 96
6	25,2	6	≥ 129	≥ 132

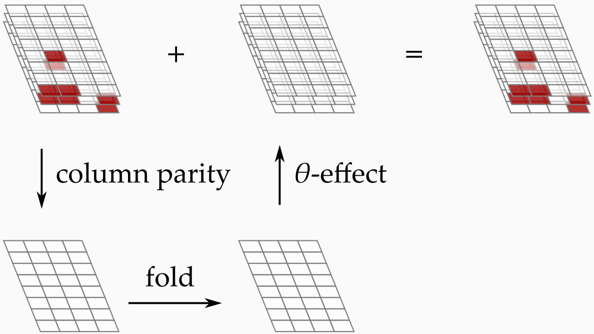
Another type of mixing layer: column parity mixers (Keccak-f and Xoodoo)



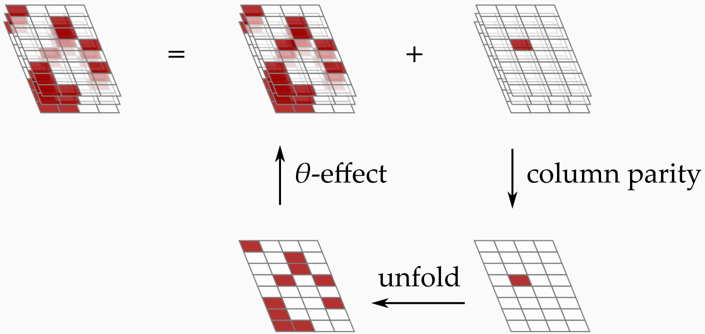
Another type of mixing layer: column parity mixers (Keccak-f and Xoodoo)



Another type of mixing layer: column parity mixers (Keccak-f and Xoodoo)



Another type of mixing layer: column parity mixers (Keccak-f and Xoodoo)



- Good average diffusion, identity for states in *kernel*
- Cost: 2 xor per bit

Showdown Xoodoo vs Ascon-p

Showdown Xoodoo vs Ascon-p

Ascon

# rounds	cost xor	cost and/or	min. trail weight diff	min. trail weight lin
1	4,2	1	2	2
2	8,4	2	8	8
3	12,6	3	40	28
4	16,8	4	≥ 86	≥ 88
5	21	5	≥ 100	≥ 96
6	25,2	6	≥ 129	≥ 132

Xoodoo

# rounds	cost xor	cost and/or	min. trail weight
1	3	1	2
2	6	2	8
3	9	3	36
4	12	4	80
5	15	5	≥ 98
6	18	6	≥ 132

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo
- 3-round trails [Bordes et al., CRYPTO 2021]

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo
- 3-round trails [Bordes et al., CRYPTO 2021]
    - we have checked all differential and linear trails with weight up to 50

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo
- 3-round trails [Bordes et al., CRYPTO 2021]
    - we have checked all differential and linear trails with weight up to 50
    - each of them is alone in its differential/linear approximation

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo
- 3-round trails [Bordes et al., CRYPTO 2021]
  - we have checked all differential and linear trails with weight up to 50
  - each of them is alone in its differential/linear approximation
  - for each differential trails we have: $\mathrm{DP}(Q) = \mathrm{EDP}(Q)$

Trail clustering and independence of round differentials in Xoodoo

- We investigated clipping and clustering in Xoodoo
- 3-round trails [Bordes et al., CRYPTO 2021]
  - we have checked all differential and linear trails with weight up to 50
  - each of them is alone in its differential/linear approximation
  - for each differential trails we have: $\mathrm{DP}(Q) = \mathrm{EDP}(Q)$
- 4-round trails: work in progress
  - 4 trails of weight 80
  - 2 of these cluster into differential with $\mathrm{EDP}(a, b) = 2^{-79}$
  - dependence of round differentials: we're starting

Ambition

- Develop a family of permutations with varying widths

Ambition

- Develop a family of permutations with varying widths
- Goals:

Ambition

- Develop a family of permutations with varying widths
- Goals:
  - no trails with weight below 128 at cost 12 xor plus 4 and per bit

Ambition

- Develop a family of permutations with varying widths
- Goals:
  - no trails with weight below 128 at cost 12 xor plus 4 and per bit
  - no noticeable clustering
  - no noticeable dependencies between round differentials
  - attention for resistance against summation attacks

Ambition

- Develop a family of permutations with varying widths
- Goals:
  - no trails with weight below 128 at cost 12 xor plus 4 and per bit
  - no noticeable clustering
  - no noticeable dependencies between round differentials
  - attention for resistance against summation attacks
- Using:
  - χ_3 or χ_5

Ambition

- Develop a family of permutations with varying widths
- Goals:
  - no trails with weight below 128 at cost 12 xor plus 4 and per bit
  - no noticeable clustering
  - no noticeable dependencies between round differentials
  - attention for resistance against summation attacks
- Using:
  - χ_3 or χ_5
  - mixing layer with cost 2 xor per bit

Ambition

- Develop a family of permutations with varying widths
- Goals:
  - no trails with weight below 128 at cost 12 xor plus 4 and per bit
  - no noticeable clustering
  - no noticeable dependencies between round differentials
  - attention for resistance against summation attacks
- Using:
  - χ_3 or χ_5
  - mixing layer with cost 2 xor per bit
  - shuffle with *as few shifts as we can afford*

Thanks for your attention!