RUHR-UNIVERSITÄT BOCHUM

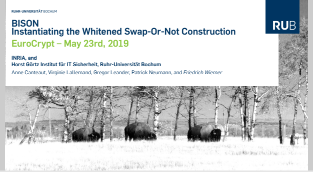
# BISON
# Instantiating the Whitened Swap-Or-Not Construction

## EuroCrypt – May 23rd, 2019

**INRIA, and**
**Horst Görtz Institut für IT Sicherheit, Ruhr-Universität Bochum**
Anne Canteaut, Virginie Lallemand, Gregor Leander, Patrick Neumann, and *Friedrich Wiemer*








- Whitened Swap-Or-Not Construction developed by Hoang et al. and Tessaro
- Way of building block ciphers
- As this is one of the few talks here at EuroCrypt about block ciphers, we will start simple

# Overview

Topics covered

## Two Parts

- Why do we need so many encryption rounds?
- Security argument for differential cryptanalysis.

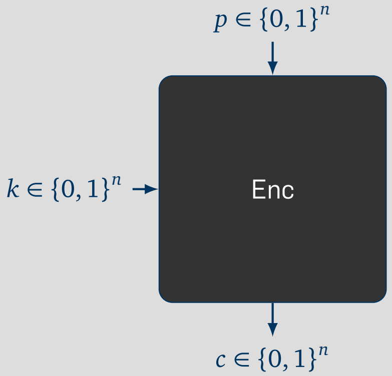




- Talk mainly about two parts of the paper:
  - Why do we need so many rounds
    (easy to understand argument)
  - Security against differential cryptanalysis
    (again relative simple argument that gives strong security here)

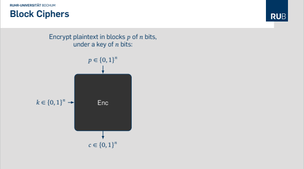
# Block Ciphers

Encrypt plaintext in blocks $p$ of $n$ bits,
under a key of $n$ bits:







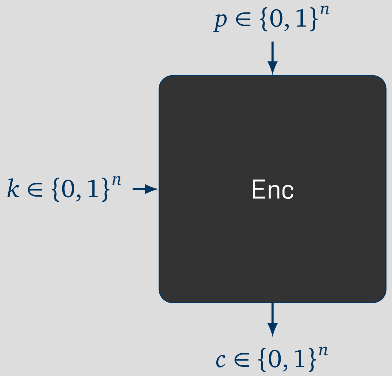



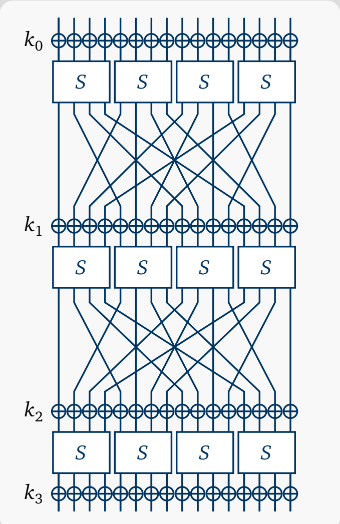
- Block ciphers encrypt *blocks* of $n$-bit inputs under an $n$-bit master key
- As a basic cryptographic primitive, we need special modes of operations, if the data to be encrypted is not of exactly $n$-bit length.
- This we do not consider here, instead we want to look at how to build this black box.

- Typicall approach is an SPN structure, where key-addition, S-box layer and a linear layer are iterated over several rounds.
- Relatively well understood
- Good security arguments against known attacks
- There are some problems: differentials and linear hull effects

# Block Ciphers

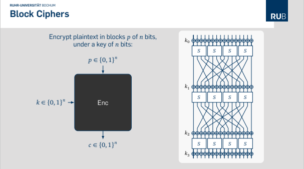
Encrypt plaintext in blocks $p$ of $n$ bits,
under a key of $n$ bits:







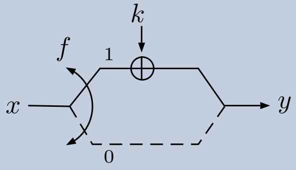
- Block ciphers encrypt *blocks* of $n$-bit inputs under an $n$-bit master key
- As a basic cryptographic primitive, we need special modes of operations, if the data to be encrypted is not of exactly $n$-bit length.
- This we do not consider here, instead we want to look at how to build this black box.

- Typicall approach is an SPN structure, where key-addition, S-box layer and a linear layer are iterated over several rounds.
- Relatively well understood
- Good security arguments against known attacks
- There are some problems: differentials and linear hull effects

# The WSN construction

Published by Tessaro at AsiaCrypt 2015 [`ia.cr/2015/868`].

## Overview round, iterated $r$ times



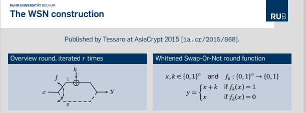

## Whitened Swap-Or-Not round function

$$x, k \in \{0,1\}^n \quad \text{and} \quad f_k : \{0,1\}^n \to \{0,1\}$$

$$y = \begin{cases} x + k & \text{if } f_k(x) = 1 \\ x & \text{if } f_k(x) = 0 \end{cases}$$





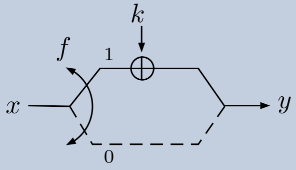


- Lets take a look at the WSN construction (simplified).
- Again, an iterated round function, where the input is fed into from the left.
- Next, a Boolean function decides if either the round key $k$ is xored onto the input, or nothing happens.
- The result is the updated state, respective the output of the round.
- In other words, $x$, and $k$ are both $n$-bit strings and $f$ is an $n$-bit Boolean function.
- The round output $y$ is either $x + k$ if $f_k(x) = 1$ or just $x$ in the other case.
- So why is this nice?

# The WSN construction

Published by Tessaro at AsiaCrypt 2015 [`ia.cr/2015/868`].

## Overview round, iterated $r$ times




## Whitened Swap-Or-Not round function

$$x, k \in \{0,1\}^n \quad \text{and} \quad f_k : \{0,1\}^n \to \{0,1\}$$

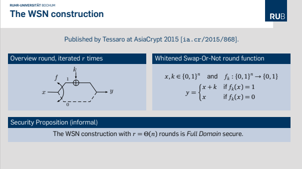
$$y = \begin{cases} x + k & \text{if } f_k(x) = 1 \\ x & \text{if } f_k(x) = 0 \end{cases}$$

## Security Proposition (informal)

The WSN construction with $r = \Theta(n)$ rounds is *Full Domain* secure.







- Lets take a look at the WSN construction (simplified).
- Again, an iterated round function, where the input is fed into from the left.
- Next, a Boolean function decides if either the round key $k$ is xored onto the input, or nothing happens.
- The result is the updated state, respective the output of the round.
- In other words, $x$, and $k$ are both $n$-bit strings and $f$ is an $n$-bit Boolean function.
- The round output $y$ is either $x+k$ if $f_k(x) = 1$ or just $x$ in the other case.
- So why is this nice?

- Tessaro was able to show that this construction, when iterated over $\Theta(n)$ rounds, achieves *Full Domain* security (what ever that means).

# An Implementation





- Sounds all very great.
- So from a practitioners point of view the natural next point is: lets implement it.

# An Implementation

RUB

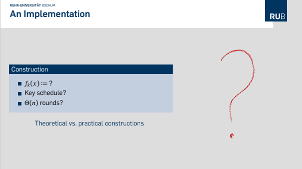
## Construction

- $f_k(x) := ?$
- Key schedule?
- $\Theta(n)$ rounds?

Theoretical vs. practical constructions







- Sounds all very great.
- So from a practitioners point of view the natural next point is: lets implement it.

- But uggh…
- How does this Boolean function $f_k$ actually looks like?
- What about a key schedule? How do we derive the round keys?
- And how many are $\Theta(n)$ rounds?

- So, from a theoretical point of view we have a nice construction.
- But from a practical point of view it is basically useless.

- OK, let us fix this.

# The WSN construction

## Encryption

Input

$x$

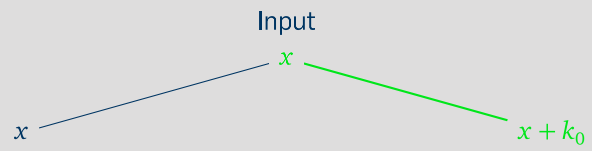




- We can observe an interesting first property, when looking at the encryption procedure round by round
- Starting with the plaintext $x$...

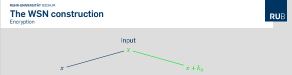
# The WSN construction

## Encryption





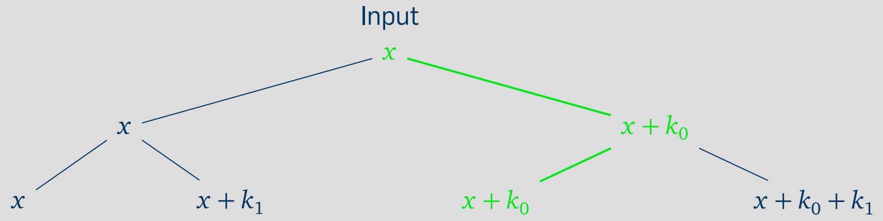


- We can observe an interesting first property, when looking at the encryption procedure round by round
- Starting with the plaintext $x$...
- ...in each round, we either add the round key $k_i$, ...

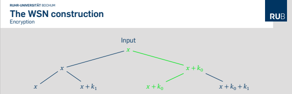
# The WSN construction

## Encryption





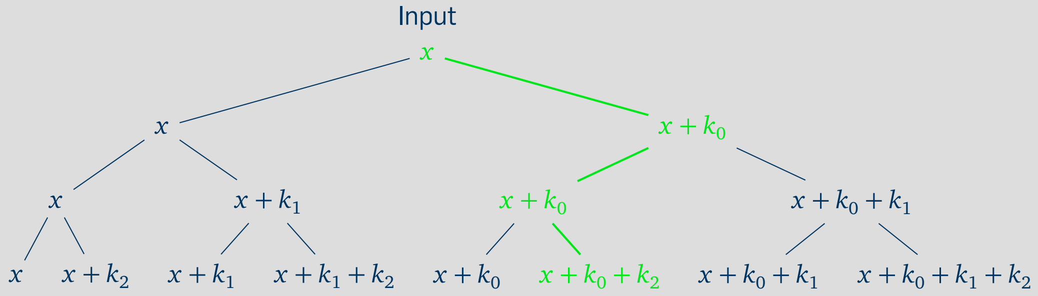


- We can observe an interesting first property, when looking at the encryption procedure round by round
- Starting with the plaintext $x$…
- …in each round, we either add the round key $k_i$, …
- …or not.

# The WSN construction

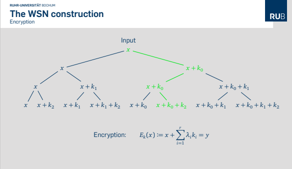
Encryption




Encryption: $$E_k(x) := x + \sum_{i=1}^{r} \lambda_i k_i = y$$







- We can observe an interesting first property, when looking at the encryption procedure round by round
- Starting with the plaintext $x$…
- …in each round, we either add the round key $k_i$, …
- …or not.
- Thus we end up with a binary tree of possible states.
- Furthermore, the encryption can also be written as the plaintext plus the sum of some round keys, chosen by the $\lambda_i$'s here.

# Generic Analysis

## On the number of rounds

RUB

### Observation

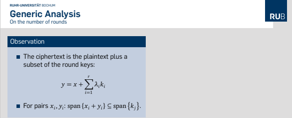
- The ciphertext is the plaintext plus a subset of the round keys:

$$y = x + \sum_{i=1}^{r} \lambda_i k_i$$

- For pairs $x_i, y_i$: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$.






Generic Analysis
On the number of rounds
Observation
The ciphertext is the plaintext plus a subset of the round keys:
$y = x + \sum_{i=1}^{r} \lambda_i k_i$
For pairs $x_i, y_i$: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$.

- First observation: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$
- Leads to a simple distinguishing attack, *if number of rounds* $r < n$.

# Generic Analysis

On the number of rounds

RUB

## Observation

- The ciphertext is the plaintext plus a subset of the round keys:

$$y = x + \sum_{i=1}^{r} \lambda_i k_i$$

- For pairs $x_i, y_i$: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$.

## Distinguishing Attack for $r < n$ rounds

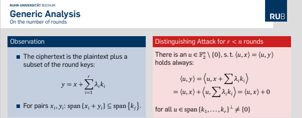
There is an $u \in \mathbb{F}_2^n \setminus \{0\}$, s. t. $\langle u, x\rangle = \langle u, y\rangle$ holds always:

$$\begin{aligned}\langle u, y\rangle &= \left\langle u, x + \sum \lambda_i k_i \right\rangle \\ &= \langle u, x\rangle + \left\langle u, \sum \lambda_i k_i \right\rangle = \langle u, x\rangle + 0\end{aligned}$$

for all $u \in \operatorname{span}\{k_1, \dots, k_r\}^{\perp} \neq \{0\}$







- First observation: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$
- Leads to a simple distinguishing attack, *if number of rounds* $r < n$.

- It is easy to find a $u$, s. t. $\langle u, y\rangle = \langle u, x\rangle = 0$ for all $x, y = x, E(x)$.
- Simply use the bilinearity of the scalar product.
- Then any $u$ from the dual space spanned by the round keys fullfills the above equation.
- As long as there are less then $n$ round keys, this dual space has dimension greater or equal then one.

# Generic Analysis

## On the number of rounds

RUB

### Observation

- The ciphertext is the plaintext plus a subset of the round keys:

$$y = x + \sum_{i=1}^{r} \lambda_i k_i$$

- For pairs $x_i, y_i$: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$.

### Distinguishing Attack for $r < n$ rounds

There is an $u \in \mathbb{F}_2^n \setminus \{0\}$, s. t. $\langle u, x \rangle = \langle u, y \rangle$ holds always:

$$\begin{aligned} \langle u, y \rangle &= \left\langle u, x + \sum \lambda_i k_i \right\rangle \\ &= \langle u, x \rangle + \left\langle u, \sum \lambda_i k_i \right\rangle = \langle u, x \rangle + 0 \end{aligned}$$

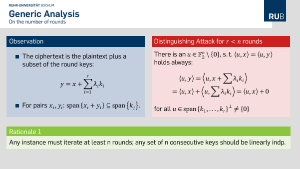
for all $u \in \operatorname{span}\{k_1, \dots, k_r\}^{\perp} \neq \{0\}$

### Rationale 1

Any instance must iterate at least n rounds; any set of n consecutive keys should be linearly indp.







- First observation: $\operatorname{span}\{x_i + y_i\} \subseteq \operatorname{span}\{k_j\}$
- Leads to a simple distinguishing attack, *if number of rounds $r < n$.*

- It is easy to find a $u$, s. t. $\langle u, y \rangle = \langle u, x \rangle = 0$ for all $x, y = x, E(x)$.
- Simply use the bilinearity of the scalar product.
- Then any $u$ from the dual space spanned by the round keys fullfills the above equation.
- As long as there are less then $n$ round keys, this dual space has dimension greater or equal then one.

- A first design rational is thus…

# Generic Analysis

On the Boolean functions $f$

A bit out of the clear blue sky, but:

**Rationale 2**

For any instance, $f_k$ has to depend on all bits, and for any $\delta \in \mathbb{F}_2^n$ : $\Pr[f_k(x) = f_k(x+\delta)] \approx \frac{1}{2}$.





- We also need this second rationale.
- Its not so easy explainable without going into more depth.
- So you have to believe me on this one.
- It basically says that for any input difference $\delta \neq k$:

$$\Pr[f_k(x) = f_k(x+\delta)] \approx \frac{1}{2}$$

# A genus of the WSN family: BISON

**Rationale 1**

Any instance must iterate at least n rounds; any set of n consecutive keys should be linearly indp.

**Rationale 2**

For any instance, $f_k$ has to depend on all bits, and for any $\delta \in \mathbb{F}_2^n: \Pr[f_k(x) = f_k(x+\delta)] \approx \frac{1}{2}$.

**Generic properties of Bent whItened Swap Or Not (BISON)**

- At least $n$ iterations of the round function
- Consecutive round keys linearly independent
- The round function depends on all bits
- $\forall \delta: \Pr[f_k(x) = f_k(x+\delta)] = \frac{1}{2}$ (*bent*)





- A quick recap and implications for any WSN instance.
- Rationale 2 basically tells us, we have to use *bent* functions.
- Thats nice, as those functions are quite well understood and already well scrutinised.
- Also, this is the reason for the name: Bent Whitened Swap-Or-Not
- But, and thats not so nice...

# A genus of the WSN family: BISON

RUB

**Rationale 1**

Any instance must iterate at least n rounds; any set of n consecutive keys should be linearly indp.

**Rationale 2**

For any instance, $f_k$ has to depend on all bits, and for any $\delta \in \mathbb{F}_2^n : \Pr[f_k(x) = f_k(x+\delta)] \approx \frac{1}{2}$.

**Generic properties of Bent whItened Swap Or Not (BISON)**

- At least $n$ iterations of the round function
- Consecutive round keys linearly independent
- The round function depends on all bits
- $\forall \delta : \Pr[f_k(x) = f_k(x+\delta)] = \frac{1}{2}$ (*bent*)

Rational 1 & 2: WSN is *slow* in practice!

The advantage?

Differential Cryptanalysis!



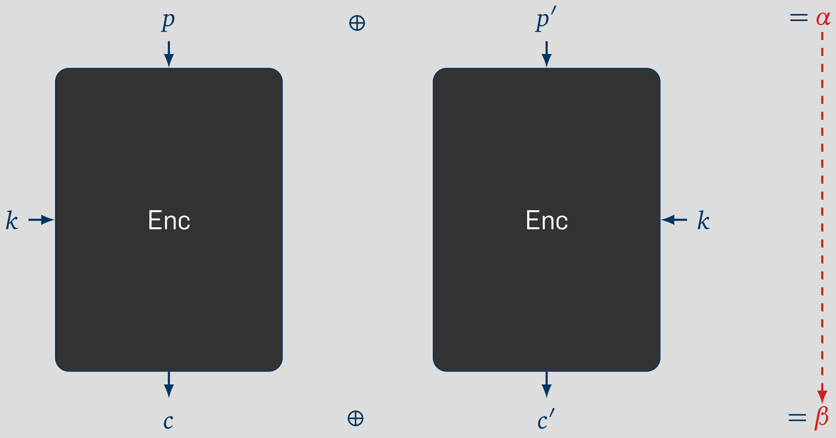


- A quick recap and implications for any WSN instance.
- Rationale 2 basically tells us, we have to use *bent* functions.
- Thats nice, as those functions are quite well understood and already well scrutinised.
- Also, this is the reason for the name: Bent Whitened Swap-Or-Not
- But, and thats not so nice…

- $n$ iterations of a round function that depends on *all* bits will be slow
- Let me repeat this (Reviewer 2 argued that we should optimise more): No matter how good we will optimise this: *it will be slow*
- For example, assume you can do one round in one clock cycle, this is still an order of magnitude slower than AES.
- So, why should we care about any instance?
- All hope is not lost, let's have a look at differential cryptanalysis!

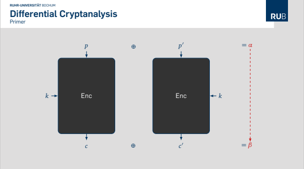
# Differential Cryptanalysis

Primer





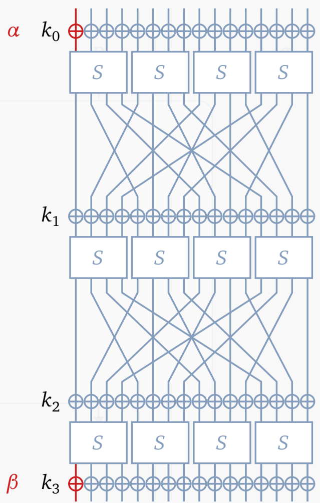


- For differential cryptanalysis, interested in propagation of input difference $\alpha$ to output difference $\beta$.
- Doing this in general at this abstraction level is a very hard problem.

# Differential Cryptanalysis

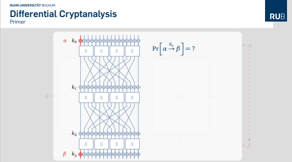
Primer




$$\Pr\left[\alpha \xrightarrow{E_k} \beta\right] = ?$$



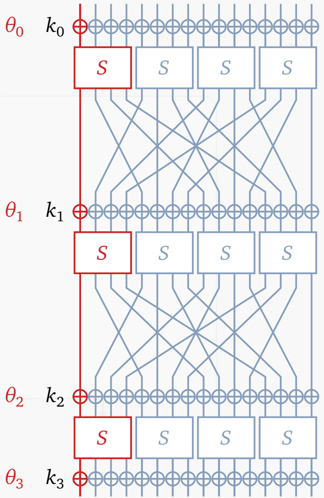




- For differential cryptanalysis, interested in propagation of input difference $\alpha$ to output difference $\beta$.
- Doing this in general at this abstraction level is a very hard problem.
- To say anything, we usually look for single so called *trails* through the inner building blocks.

# Differential Cryptanalysis

Primer

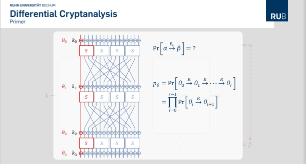



$$\Pr\left[\alpha \xrightarrow{E_k} \beta\right] = ?$$

$$p_\theta = \Pr\left[\theta_0 \xrightarrow{R} \theta_1 \xrightarrow{R} \cdots \xrightarrow{R} \theta_r\right]$$
$$= \prod_{i=0}^{r-1} \Pr\left[\theta_i \xrightarrow{R} \theta_{i+1}\right]$$



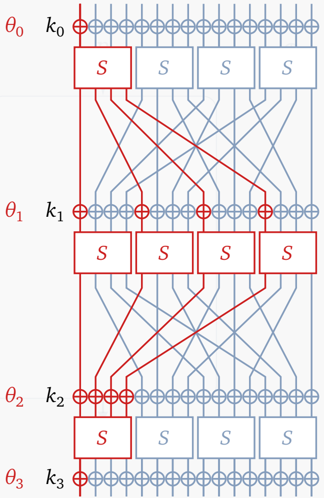




- For differential cryptanalysis, interested in propagation of input difference $\alpha$ to output difference $\beta$.
- Doing this in general at this abstraction level is a very hard problem.
- To say anything, we usually look for single so called *trails* through the inner building blocks.
- Now, computing the probability of one such trail is actually doable.
- But, trails can go several alternative ways through non-linear parts, thus we have to cope with a branching effect…

# Differential Cryptanalysis

Primer



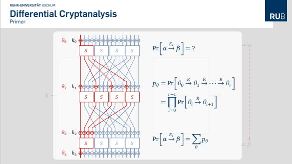

$$\Pr\left[\alpha \xrightarrow{E_k} \beta\right] = ?$$

$$p_\theta = \Pr\left[\theta_0 \xrightarrow{R} \theta_1 \xrightarrow{R} \cdots \xrightarrow{R} \theta_r\right] = \prod_{i=0}^{r-1} \Pr\left[\theta_i \xrightarrow{R} \theta_{i+1}\right]$$

$$\Pr\left[\alpha \xrightarrow{E_k} \beta\right] = \sum_\theta p_\theta$$







- For differential cryptanalysis, interested in propagation of input difference $\alpha$ to output difference $\beta$.
- Doing this in general at this abstraction level is a very hard problem.
- To say anything, we usually look for single so called *trails* through the inner building blocks.
- Now, computing the probability of one such trail is actually doable.
- But, trails can go several alternative ways through non-linear parts, thus we have to cope with a branching effect. . .
- And eventually, several of these trails cluster in a so called *differential*.
- While in this example it is still feasible, computing the *exact* probability in a real cipher is not.
- We thus have to restrain on bounding or approximating this probability.
- In the end, tight bounds for differentials over several rounds remain an open (but important!) problem.
- For BISON our aim is to give exactly this: a tight bound for any differential over several rounds.

# Differential Cryptanalysis

One round

**Theorem**

*For one round of BISON the probabilities are:*

$$\Pr_x[\alpha \to \beta] = \begin{cases} 1 & \textit{if } \alpha = \beta = k \textit{ or } \alpha = \beta = 0 \\ \frac{1}{2} & \textit{else if } \beta \in \{\alpha, \alpha + k\} \\ 0 & \textit{else} \end{cases}$$





- We start by understanding the differential one round behaviour.
- For the three possible cases, let us look at what differences are actually possible.

# Differential Cryptanalysis

One round

### Possible differences

$$\begin{array}{lll} & x & + f_k(x) & \cdot k \\ \oplus & x + \alpha & + f_k(x+\alpha) & \cdot k \\ = & \alpha & + (f_k(x) + f_k(x+\alpha)) & \cdot k \end{array}$$

### Theorem

*For one round of BISON the probabilities are:*

$$\Pr_x[\alpha \to \beta] = \begin{cases} 1 & \text{if } \alpha = \beta = k \text{ or } \alpha = \beta = 0 \\ \frac{1}{2} & \text{else if } \beta \in \{\alpha, \alpha + k\} \\ 0 & \text{else} \end{cases}$$

### Properties of $f_k$

$$f_k(x) = f_k(x+k) \tag{1}$$





RUHR-UNIVERSITÄT BOCHUM
Differential Cryptanalysis
One round
Theorem
For one round of BISON the probabilities are:
Possible differences
Properties of $f_k$

- We start by understanding the differential one round behaviour.
- For the three possible cases, let us look at what differences are actually possible.

- Remember that one round computes the output as $x + f_k(x) \cdot k$.
- With the input difference $\alpha$ we get as possible output differences $\beta \in \{0, \alpha, k, \alpha + k\}$.
- For decryption we need that $x$ and $x + k$ are mapped to the same value by $f_k$
- Thus, $\beta = \alpha$ with probability one, if and only if $\alpha = k$ or $\alpha = 0$
- If $\beta$ is not one of the above four values, such an input/output pair cannot occur, thus the probability is zero.

# Differential Cryptanalysis

One round

RUB

## Theorem

*For one round of BISON the probabilities are:*

$$\Pr_x[\alpha \to \beta] = \begin{cases} 1 & \text{if } \alpha = \beta = k \text{ or } \alpha = \beta = 0 \\ \frac{1}{2} & \text{else if } \beta \in \{\alpha, \alpha + k\} \\ 0 & \text{else} \end{cases}$$

## Possible differences

$$\begin{array}{lll} & x & + f_k(x) & \cdot k \\ \oplus & x + \alpha & + f_k(x+\alpha) & \cdot k \\ = & \alpha & + (f_k(x) + f_k(x+\alpha)) & \cdot k \end{array}$$

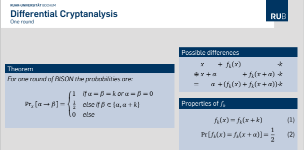
## Properties of $f_k$

$$f_k(x) = f_k(x+k) \tag{1}$$

$$\Pr[f_k(x) = f_k(x+\alpha)] = \frac{1}{2} \tag{2}$$







- We start by understanding the differential one round behaviour.
- For the three possible cases, let us look at what differences are actually possible.

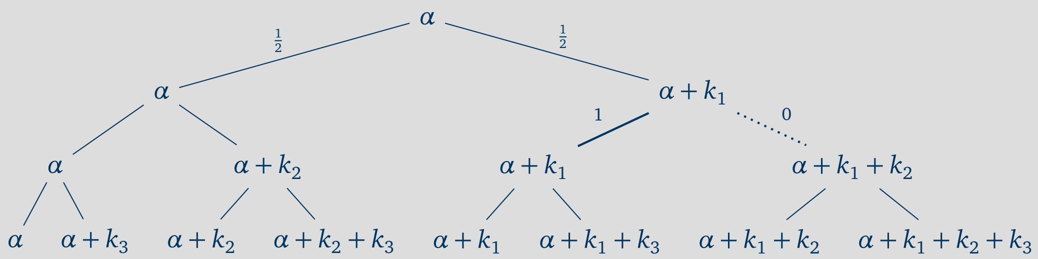
- Remember that one round computes the output as $x + f_k(x) \cdot k$.
- With the input difference $\alpha$ we get as possible output differences $\beta \in \{0, \alpha, k, \alpha + k\}$.
- For decryption we need that $x$ and $x + k$ are mapped to the same value by $f_k$
- Thus, $\beta = \alpha$ with probability one, if and only if $\alpha = k$ or $\alpha = 0$
- If $\beta$ is not one of the above four values, such an input/output pair cannot occur, thus the probability is zero.

- For the last case, remember that for any input difference, we required that $f_k$ collides with probability one half.

# Differential Cryptanalysis

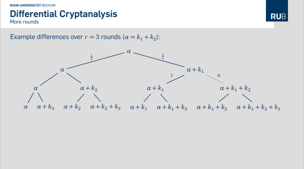
## More rounds

Example differences over $r = 3$ rounds ($\alpha = k_1 + k_2$):





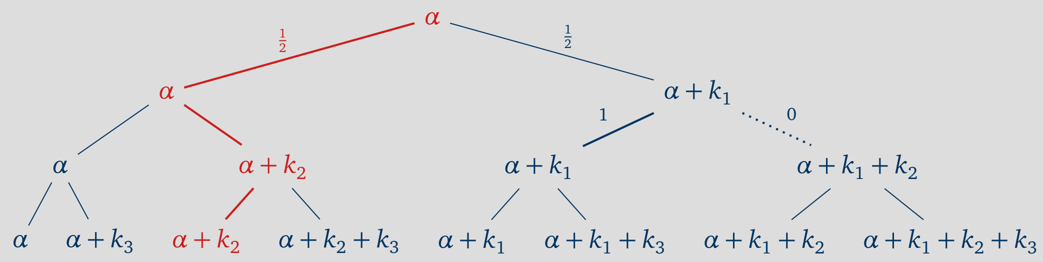


- When we look at more rounds, we can depict these cases again in this tree structure.
- Starting with the input difference $\alpha$, choosing an input value $x$ determines the path we take through the tree.
- Now assuming that $\alpha = k_1 + k_2$ it can not happen that the differential characteristic takes twice the right branch.
- This would result in a zero difference, which is not possible for permutations as long as the input difference is non-zero.

# Differential Cryptanalysis
More rounds

Example differences over $r = 3$ rounds ($\alpha = k_1 + k_2$):

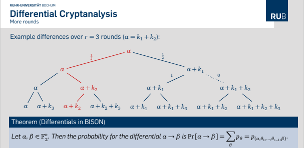



**Theorem (Differentials in BISON)**

*Let $\alpha, \beta \in \mathbb{F}_2^n$. Then the probability for the differential $\alpha \to \beta$ is* $\Pr[\alpha \to \beta] = \sum_{\theta} p_\theta = p_{(\alpha,\theta_1,\ldots,\theta_{r-1},\beta)}$.







- When we look at more rounds, we can depict these cases again in this tree structure.
- Starting with the input difference $\alpha$, choosing an input value $x$ determines the path we take through the tree.
- Now assuming that $\alpha = k_1 + k_2$ it can not happen that the differential characteristic takes twice the right branch.
- This would result in a zero difference, which is not possible for permutations as long as the input difference is non-zero.

- Regarding differentials, the important observation is:
- For any input/output pair $(\alpha, \beta)$, there is only one path.
- In other words: no branching occurs and the differential consists of a single trail only.

## A concrete species





- Up to now we do not have specified much more then the initial WSN construction had.
- For a concrete implementation, we still need to define
  - Number of rounds
  - Key Schedule
  - Boolean function $f_k$
- So let us look at a concrete BISON species
- In particular, we discuss how to tackle Rationales 1 and 2.

# Addressing Rationale 1

The Key Schedule

RUB

## Rationale 1

Any instance must iterate at least n rounds; any set of n consecutive keys should be linearly indp.

### Design Decisions

- Choose number of rounds as $3 \cdot n$
- Round keys derived from the state of LFSRs
- Add round constants $c_i$ to $w_i$ round keys

### Implications

- Clocking an LFSR is cheap
- For an LFSR with irreducible feedback polynomial of degree $n$, every $n$ consecutive states are linearly independent
- Round constants avoid structural weaknesses





- Due to analysis of the alg. deg., we chose $3n$ rounds, see last slide.
- Deriving round keys from the states of LFSRs is efficiently implementable and fullfils our requirements for linear independent round keys.
- For those of you how attended Gregors talk at FSE:
  - While generating test vectors for the implementation we again noted some unwanted symmetries for encryptions with low hamming weight.
  - Thus we added round constants, to avoid these structural weaknesses.
  - (Sorry for fixing another cipher)

# Addressing Rationale 2

## The Round Function

RUB

### Rationale 2

For any instance, the $f_k$ should depend on all bits, and for any $\delta \in \mathbb{F}_2^n : \Pr[f_k(x) = f_k(x+\delta)] \approx \frac{1}{2}$.

### Design Decisions

- Choose $f_k : \mathbb{F}_2^n \to \mathbb{F}_2$ s. t.

$$\delta \in \mathbb{F}_2^n : \Pr[f_k(x) = f_k(x+\delta)] = \frac{1}{2},$$

  that is, $f_k$ is a bent function.
- Choose the simplest bent function known:

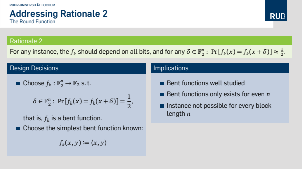
$$f_k(x, y) := \langle x, y \rangle$$

### Implications

- Bent functions well studied
- Bent functions only exists for even $n$
- Instance not possible for every block length $n$







- Just chose the simplest bent function, the scalar product.
- This is also efficiently implementable.

- But, another drawback:
- Bent functions exists only for even $n$.
- Thus BISON cannot be instantiated for every block length $n$.
- In particular, due to reasons not covered here, we can actually only instantiate it for *odd* block lengths.

# An Implementation

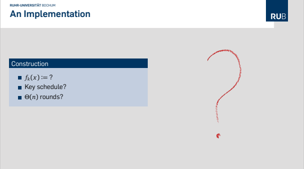
RUB

**Construction**

- $f_k(x) := ?$
- Key schedule?
- $\Theta(n)$ rounds?







- Coming back to our initial question.
- And basically only for the sake of completeness, as we already saw this is going to be slow.

# An Implementation

RUB

## Construction

- $f_k(x,y) := \langle x,y \rangle$
- Key schedule: LFSRs.
- $\Theta(n) = 3n$ rounds.





RUHR-UNIVERSITÄT BOCHUM
An Implementation
RUB
Construction
$f_k(x,y) := \langle x,y \rangle$
Key schedule: LFSRs.
$\Theta(n) = 3n$ rounds.

- Coming back to our initial question.
- And basically only for the sake of completeness, as we already saw this is going to be slow.
- We have specified everything, so let's benchmark against AES (what else).

# An Implementation

RUB

**Construction**

- $f_k(x,y) := \langle x,y \rangle$
- Key schedule: LFSRs.
- $\Theta(n) = 3n$ rounds.

| Cipher | Block size (bit) | Cycles/Byte mean |
|---|---|---|
| AES* | 128 | 0.65 |
| BISON† | 129 | 3 064.08 |

* AES-128 on Skylake Intel® Core i7-7800X @ 3.5GHz, see Daemen et al. [The design of Xoodoo and Xoofff, Table 5].
† BISON on CoffeeLake Intel® Core i7-8700 @ 3.7 GHz.





- Coming back to our initial question.
- And basically only for the sake of completeness, as we already saw this is going to be slow.
- We have specified everything, so let's benchmark against AES (what else).
- OK, told you so, BISON is like 4 700 times slower than AES.
- Or: more than three orders of magnitude.
- Optimising this will not help enough.

# Further Cryptanalysis

RUB

## Linear Cryptanalysis

For $r \geqslant n$ rounds, the correlation of any non-trivial linear trail for BISON is upper bounded by $2^{-\frac{n+1}{2}}$.

## Invariant Attacks

For $r \geqslant n$ rounds, neither invariant subspaces nor nonlinear invariant attacks do exist for BISON.

## Zero Correlation

For $r > 2n - 2$ rounds, BISON does not exhibit any zero correlation linear hulls.

## Impossible Differentials

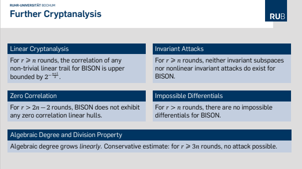
For $r > n$ rounds, there are no impossible differentials for BISON.

## Algebraic Degree and Division Property

Algebraic degree grows *linearly*. Conservative estimate: for $r \geqslant 3n$ rounds, no attack possible.







- We did more cryptanalysis, but our results are more of the “classical” kind.
- For linear cryptanalysis, we bound the correlation of any non-trivial trail.
- Current known security arguments for resistance against invariant attacks apply.
- Zero correlation and impossible differentials do not exist for $2n$ rounds or more.
- Best attacks seem to exploit the algebraic degree.
- We show that it grows only linearly – which is especially bad in comparison to SPN ciphers.
- The result on the algebraic degree also applies to NLFSRs or maximally unbalanced Feistel networks.
- Conservative estimation: might work for more than $2n$ rounds, but not for $3n$ or more.

# Conclusion/Questions

Thank you for your attention!

RUB

## BISON

- A first instance of the WSN construction
- Good results for differential cryptanalysis

## Open Problems

- Construction for linear cryptanalysis?
- Similar args. for Unbalanced Feistel?

Thank you!

Questions?

2018/1011





Conclusion/Questions
Thank you for your attention!
BISON
A first instance of the WSN construction
Good results for differential cryptanalysis
Open Problems
Construction for linear cryptanalysis?
Similar args. for Unbalanced Feistel?
Thank you!
Questions?
2018/1011

# Details

# BISON

## Round Function

RUB

### BISON's round function

For round keys $k_i \in \mathbb{F}_2^n$ and $w_i \in \mathbb{F}_2^{n-1}$ the round function computes

$$R_{k_i,w_i}(x) := x + f_{b(i)}\big(w_i + \Phi_{k_i}(x)\big) \cdot k_i.$$

where

- $\Phi_{k_i}$ and $f_{b(i)}$ are defined as

$$\Phi_k(x) : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$$
$$\Phi_k(x) := (x + x[i(k)] \cdot k)[j]_{\substack{1 \leqslant j \leqslant n \\ j \neq i(k)}}$$

$$f_{b(i)} : \mathbb{F}_2^{\frac{n-1}{2}} \times \mathbb{F}_2^{\frac{n-1}{2}} \to \mathbb{F}_2$$
$$f_{b(i)}(x, y) := \langle x, y \rangle + b(i),$$

- and $b(i)$ is 0 if $i \leqslant \frac{r}{2}$ and 1 else.

## BISON's key schedule

Given

- primitive $p_k, p_w \in \mathbb{F}_2[x]$ with degrees $n, n-1$ and companion matrices $C_k, C_w$.
- master key $K = (k, w) \in \left(\mathbb{F}_2^n \times \mathbb{F}_2^{n-1}\right) \setminus \{0, 0\}$

The $i$th round keys are computed by

$$\begin{aligned} \mathrm{KS}_i : \mathbb{F}_2^n \times \mathbb{F}_2^{n-1} &\to \mathbb{F}_2^n \times \mathbb{F}_2^{n-1} \\ \mathrm{KS}_i(k, w) &:= (k_i, c_i + w_i) \end{aligned}$$

where

$$k_i = (C_k)^i k, \qquad c_i = (C_w)^{-i} e_1, \qquad w_i = (C_w)^i w.$$