Improving Key Recovery Linear Attacks with Walsh Spectrum Puncturing

Antonio Flórez-Gutiérrez, Yosuke Todo
antonio.florez@ntt.com, yosuke.todo@ntt.com

NTT Social Informatics Laboratories, Japan

EUROCRYPT 2024, Zurich
May 27, 2024

Introduction

Structure of the Presentation

1. Introduction
   - Linear Key Recovery Attacks
   - Walsh spectrum of a (pseudo)Boolean function
2. Previous Work
   - Walsh Transform Technique (Collard, Standaert, Quisquater, 2007)
   - Walsh Transform Pruning (Flórez-Gutiérrez, 2022)
3. Linear Attacks with Walsh Spectrum Puncturing
   - Motivation for Puncturing
   - Puncturing Strategies
4. Impact of Puncturing on the Data Complexity
5. Conclusion and Applications

Linear Cryptanalysis

Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be a block cipher, $E(x, K) = E_K(x) = y$

Linear Cryptanalysis

Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be a block cipher, $E(x, K) = E_K(x) = y$

A linear approximation (Matsui, 1993) is any linear combination of bits of the plaintext and the ciphertext (and sometimes also the key):

$$\langle \alpha, x \rangle \oplus \langle \beta, y \rangle$$

α and β are called input and output masks

Linear Cryptanalysis

Let $E : \mathbb{F}_2^n \times \mathbb{F}_2^\kappa \longrightarrow \mathbb{F}_2^n$ be a block cipher, $E(x, K) = E_K(x) = y$

A linear approximation (Matsui, 1993) is any linear combination of bits of the plaintext and the ciphertext (and sometimes also the key):

$$\langle \alpha, x \rangle \oplus \langle \beta, y \rangle$$

α and β are called input and output masks

The correlation measures the statistical imbalance of the approximation:

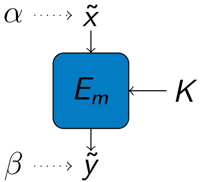
$$\mathrm{cor}_K(\alpha, \beta) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x \rangle \oplus \langle \beta, y \rangle}$$

Linear cryptanalysis exploits approximations with high correlation

Linear Key Recovery Attack

Linear approximation of (part of) a block cipher:

$$\langle \alpha, \tilde{x} \rangle \oplus \langle \beta, \tilde{y} \rangle \text{ with correlation } c$$



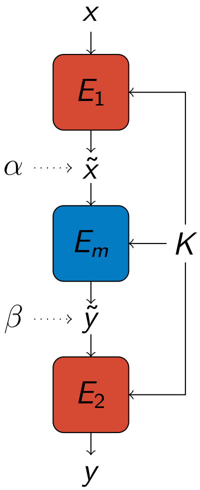
Linear Key Recovery Attack

Linear approximation of (part of) a block cipher:

$$\langle \alpha, \tilde{x} \rangle \oplus \langle \beta, \tilde{y} \rangle \text{ with correlation } c$$

We express the linear approximation as a function of the plaintext, ciphertext and key with the key recovery map, for example:

$$E_1^{\text{trunc}}(x, k) \oplus E_2^{\text{trunc}}(y, k), \text{ where } k \text{ is part of the key}$$



Linear Key Recovery Attack

Linear approximation of (part of) a block cipher:

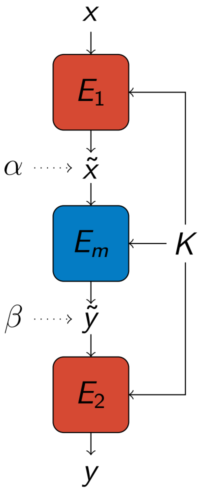
$$\langle \alpha, \tilde{x} \rangle \oplus \langle \beta, \tilde{y} \rangle \text{ with correlation } c$$

We express the linear approximation as a function of the plaintext, ciphertext and key with the key recovery map, for example:

$$E_1^{\text{trunc}}(x, k) \oplus E_2^{\text{trunc}}(y, k), \text{ where } k \text{ is part of the key}$$

We divide the relevant part of the plaintext/ciphertext into segments and consider key recovery maps of the form:

$$f_0(x) \oplus \underbrace{f_1(x_1 \oplus k_1^O, k_1^I) \oplus \ldots \oplus f_d(x_d \oplus k_d^O, k_d^I)}_{f(X \oplus K^O, K^I)}$$



Linear Key Recovery Attack *(cont.)*

The objective is to compute the experimental correlations for all key guesses k:

$$\widehat{\mathrm{cor}}(k) = \frac{1}{N}\sum_{x\in\mathcal{D}}(-1)^{E_1^{\mathsf{trunc}}(x,k)\oplus E_2^{\mathsf{trunc}}(y,k)},\ \mathcal{D} \text{ data sample of size } N \approx 1/c^2$$

as the correct key guess is expected to have a larger experimental correlation

Linear Key Recovery Attack *(cont.)*

The objective is to compute the experimental correlations for all key guesses k:

$$\widehat{\mathrm{cor}}(k) = \frac{1}{N} \sum_{x \in \mathcal{D}} (-1)^{E_1^{\text{trunc}}(x,k) \oplus E_2^{\text{trunc}}(y,k)}, \ \mathcal{D} \text{ data sample of size } N \approx 1/c^2$$

as the correct key guess is expected to have a larger experimental correlation

$$\widehat{cor}(K^O, K^I) = \frac{1}{N} \sum_{x \in \mathcal{D}} (-1)^{f_0(x)} (-1)^{f(X \oplus K^O, K^I)}$$

Linear Key Recovery Attack *(cont.)*

The objective is to compute the experimental correlations for all key guesses k:

$$\widehat{\mathrm{cor}}(k) = \frac{1}{N}\sum_{x\in\mathcal{D}}(-1)^{E_1^{\text{trunc}}(x,k)\oplus E_2^{\text{trunc}}(y,k)},\ \mathcal{D} \text{ data sample of size } N \approx 1/c^2$$

as the correct key guess is expected to have a larger experimental correlation

$$\widehat{cor}(K^O, K^I) = \frac{1}{N}\sum_{x\in\mathcal{D}}(-1)^{f_0(x)}(-1)^{f(X\oplus K^O,K^I)}$$

We can compute this vector either directly or with a distillation step, with costs

$$N \cdot 2^{|K^I|+|K^O|} \text{ (Matsui, 1993) and } N + 2^{|K^I|+2|K^O|} \text{ (Matsui, 1994)}$$

$|x|$ denotes the number of bits of the vector x.

Walsh Spectrum of a Boolean Function

Walsh Transform of a (pseudo)Boolean function

We can see $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$ as $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\} \subseteq \mathbb{R}$ by taking $(-1)^f$, and work in the larger space of pseudoboolean functions $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$

Walsh Spectrum of a Boolean Function

Walsh Transform of a (pseudo)Boolean function

We can see $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$ as $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\} \subseteq \mathbb{R}$ by taking $(-1)^f$, and work in the larger space of pseudoboolean functions $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$

The Walsh spectrum or Walsh transform of f is $\widehat{f} : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ is

$$\widehat{f}(u) = \frac{1}{2^\ell} \sum_{x \in \mathbb{F}_2^\ell} (-1)^{\langle x,u \rangle} f(x)$$

We note that $2^\ell \widehat{\widehat{f}} = f$

Walsh Spectrum of a Boolean Function

Walsh Transform of a (pseudo)Boolean function

We can see $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$ as $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\} \subseteq \mathbb{R}$ by taking $(-1)^f$, and work in the larger space of pseudoboolean functions $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$

The Walsh spectrum or Walsh transform of f is $\widehat{f} : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ is

$$\widehat{f}(u) = \frac{1}{2^\ell} \sum_{x \in \mathbb{F}_2^\ell} (-1)^{\langle x,u \rangle} f(x)$$

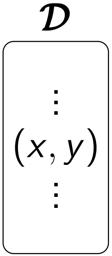
We note that $2^\ell \widehat{\widehat{f}} = f$

There is a fast algorithm to obtain $\widehat{f}$ from f requiring $\ell 2^\ell$ additions

Previous Work

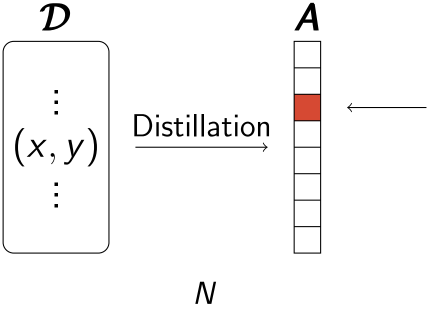
The Walsh Transform Technique

(Collard, Standaert, Quisquater, 2007), (Flórez-Gutiérrez, Naya-Plasencia, 2020)



The Walsh Transform Technique

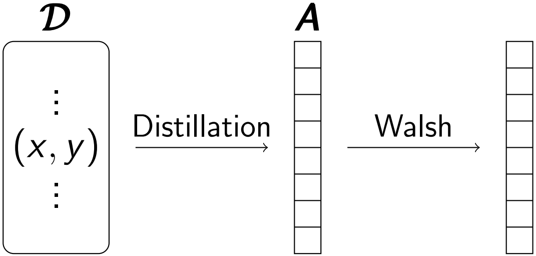
(Collard, Standaert, Quisquater, 2007), (Flórez-Gutiérrez, Naya-Plasencia, 2020)



$$A[X] = \sum_{\substack{x \in \mathcal{D} \\ x \mapsto X}} f_0(x)$$

The Walsh Transform Technique

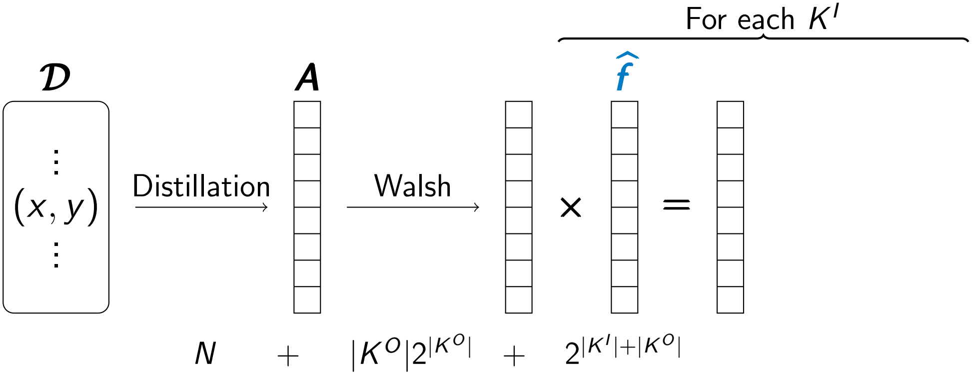
(Collard, Standaert, Quisquater, 2007), (Flórez-Gutiérrez, Naya-Plasencia, 2020)



$$N \quad + \quad |K^O| 2^{|K^O|}$$

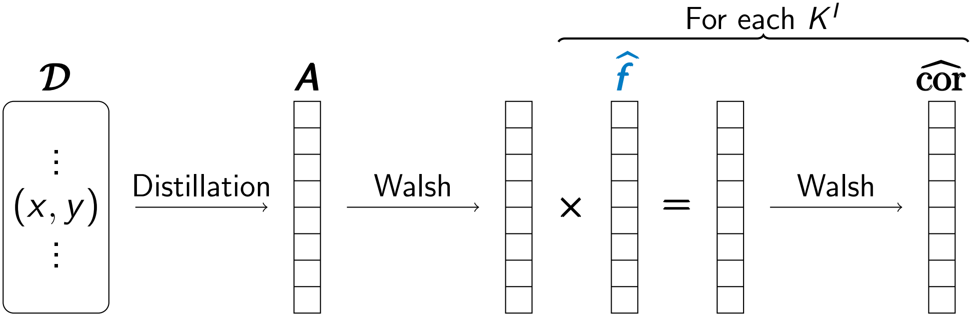
The Walsh Transform Technique

(Collard, Standaert, Quisquater, 2007), (Flórez-Gutiérrez, Naya-Plasencia, 2020)



The Walsh Transform Technique

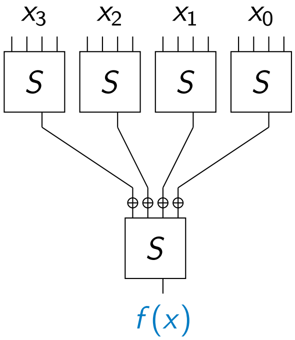
(Collard, Standaert, Quisquater, 2007), (Flórez-Gutiérrez, Naya-Plasencia, 2020)



$$N \quad + \quad |K^O|2^{|K^O|} \quad + \quad 2^{|K^I|+|K^O|} \quad + \quad |K^O|2^{|K^I|+|K^O|}$$

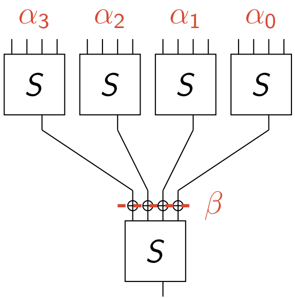
Walsh Transform Pruning

We assume S balanced and consider the following map:



Walsh Transform Pruning

We assume S balanced and consider the following map:



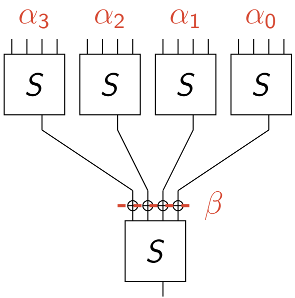
We have a nice formula for its Walsh coefficients:

$$\hat{f}(\alpha_3, \alpha_2, \alpha_1, \alpha_0) = \pm \frac{1}{4} \hat{S}(\alpha_3)\,\hat{S}(\alpha_2)\,\hat{S}(\alpha_1)\,\hat{S}(\alpha_0)\,\hat{S}(\beta),$$

where $\beta_i = 1 \Leftrightarrow \alpha_i \neq 0$ because S is balanced

Walsh Transform Pruning

We assume S balanced and consider the following map:



We have a nice formula for its Walsh coefficients:

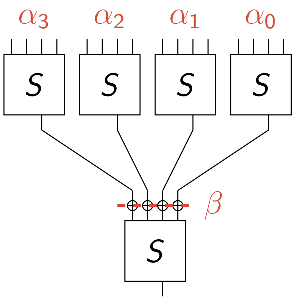
$$\hat{f}(\alpha_3, \alpha_2, \alpha_1, \alpha_0) = \pm \frac{1}{4} \hat{S}(\alpha_3) \hat{S}(\alpha_2) \hat{S}(\alpha_1) \hat{S}(\alpha_0) \hat{S}(\beta),$$

where $\beta_i = 1 \Leftrightarrow \alpha_i \neq 0$ because S is balanced

If $\hat{S}(\texttt{0xF}) = 0$, then $\hat{f}(\alpha_3, \alpha_2, \alpha_1, \alpha_0) \neq 0 \implies \alpha_i = 0$ for some i

The nonzero Walsh coefficients of f are contained in 4 vector subspaces of dimension 12 of $\mathbb{F}_2^{16}$, given by the conditions $\alpha_0 = 0$, $\alpha_1 = 0$, $\alpha_2 = 0$ and $\alpha_3 = 0$

Walsh Transform Pruning



We assume S balanced and consider the following map:

We have a nice formula for its Walsh coefficients:

$$\hat{f}(\alpha_3, \alpha_2, \alpha_1, \alpha_0) = \pm \frac{1}{4} \hat{S}(\alpha_3) \hat{S}(\alpha_2) \hat{S}(\alpha_1) \hat{S}(\alpha_0) \hat{S}(\beta),$$

where $\beta_i = 1 \Leftrightarrow \alpha_i \neq 0$ because S is balanced

If $\hat{S}(\texttt{0xF}) = 0$, then $\hat{f}(\alpha_3, \alpha_2, \alpha_1, \alpha_0) \neq 0 \implies \alpha_i = 0$ for some i

The nonzero Walsh coefficients of f are contained in 4 vector subspaces of dimension 12 of $\mathbb{F}_2^{16}$, given by the conditions $\alpha_0 = 0$, $\alpha_1 = 0$, $\alpha_2 = 0$ and $\alpha_3 = 0$

What if $\hat{S}(\texttt{0xF}) \neq 0$? ...we will discuss this situation later

Walsh Spectrum Pruning *(cont.)*

In (Flórez-Gutiérrez, 2022), a technique is introduced to exploit this and other redundancies, like those induced by the key schedule

These redundancies can be expressed as sparsity properties of the nonzero inputs and desired outputs of the Walsh transform steps

Walsh Spectrum Pruning *(cont.)*

In (Flórez-Gutiérrez, 2022), a technique is introduced to exploit this and other redundancies, like those induced by the key schedule

These redundancies can be expressed as sparsity properties of the nonzero inputs and desired outputs of the Walsh transform steps

A variant of the fast Walsh transform algorithm is introduced which has lower time complexity when the inputs and/or outputs lie in affine subspaces

Walsh Spectrum Pruning *(cont.)*

In (Flórez-Gutiérrez, 2022), a technique is introduced to exploit this and other redundancies, like those induced by the key schedule

These redundancies can be expressed as sparsity properties of the nonzero inputs and desired outputs of the Walsh transform steps

A variant of the fast Walsh transform algorithm is introduced which has lower time complexity when the inputs and/or outputs lie in affine subspaces

In particular, the importance of the structure of the support of f is shown

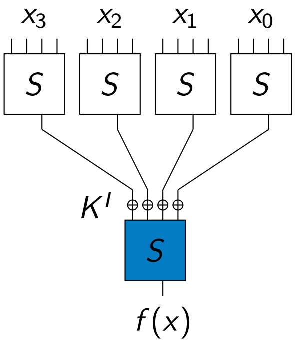
If the spectrum of f lies on a few affine subspaces of small dimension (like in the previous slide), the time complexity of the attack can be greatly reduced

Linear Attacks with Walsh Spectrum Puncturing

Walsh Spectrum Puncturing Example

Let's return to the example: we know that if $\hat{S}(\mathtt{0xF}) = 0$, there is an exploitable structure

But what happens when $\hat{S}(\mathtt{0xF}) \neq 0$? ...we make it so!

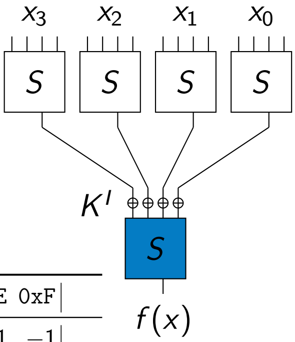


Walsh Spectrum Puncturing Example

Let's return to the example: we know that if $\hat{S}(\texttt{0xF}) = 0$, there is an exploitable structure

But what happens when $\hat{S}(\texttt{0xF}) \neq 0$? ...we make it so!

Idea 1: We reject some inputs of S so that $\widehat{S_{new}}(\texttt{0xF}) = 0$, we increase the data complexity to compensate



	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	1	−1	1	1	−1	1	1	0	1	−1	−1	1	0	0	0	−1	$\frac{4}{3}N$
$\widehat{S_{new}}$	2	2	−2	2	2	2	−2	10	4	0	−4	−4	0	4	0	0	

Walsh Spectrum Puncturing Example *(cont.)*

Idea 2: We just remove (puncture) the bad coefficient from the spectrum

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}																	
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

Walsh Spectrum Puncturing Example *(cont.)*

Idea 2: We just remove (puncture) the bad coefficient from the spectrum

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}																	
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

This has several advantages:

- Intuitively, the key recovery map is modified "as little as possible"
- We are able to remove more coefficients, for example 0x7, 0xB, 0xD, 0xE

Walsh Spectrum Puncturing Example *(cont.)*

Idea 2: We just remove (puncture) the bad coefficient from the spectrum

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	0.75	−0.75	1.25	0.75	−0.75	0.75	0.75	−0.75	1.25	−1.25	−1.25	1.25	0.75	−0.75	−0.75	−1.25	
$\widehat{S_{new}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

This has several advantages:

- Intuitively, the key recovery map is modified "as little as possible"
- We are able to remove more coefficients, for example 0x7, 0xB, 0xD, 0xE

But we have to resolve some issues:

- The key recovery map is no longer a Boolean function

Walsh Spectrum Puncturing Example *(cont.)*

Idea 2: We just remove (puncture) the bad coefficient from the spectrum

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	0.75	−0.75	1.25	0.75	−0.75	0.75	0.75	−0.75	1.25	−1.25	−1.25	1.25	0.75	−0.75	−0.75	−1.25	?
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

This has several advantages:

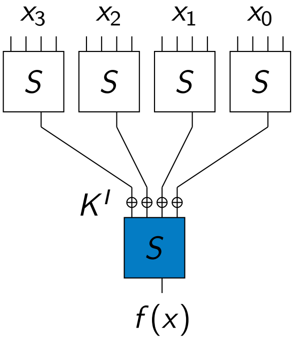
- Intuitively, the key recovery map is modified "as little as possible"
- We are able to remove more coefficients, for example 0x7, 0xB, 0xD, 0xE

But we have to resolve some issues:

- The key recovery map is no longer a Boolean function
- We don't know what the effect on the data complexity is

A Simpler Case

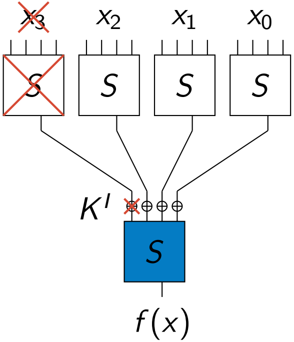
We can also consider simpler puncturing strategies



A Simpler Case

We can also consider simpler puncturing strategies

For example, we may want to force an Sbox to be inactive by puncturing the coefficients which make it active

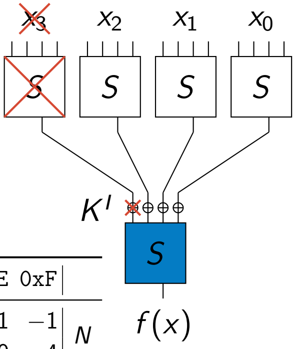


A Simpler Case

We can also consider simpler puncturing strategies

For example, we may want to force an Sbox to be inactive by puncturing the coefficients which make it active

Luckily, this time we don't have the same problems



	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	1	−1	0	1	0	0	0	−1	1	−1	0	1	0	0	0	−1	$2N$
$\widehat{S_{new}}$	0	4	0	4	4	0	−4	8	0	0	0	0	0	0	0	0	

The Puncturing and Approximation Problems

- "Classical" key recovery map: A Boolean function $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\}$
  Each plaintext is contributes by either 1 or -1 to the correlation

The Puncturing and Approximation Problems

- "Classical" key recovery map: A Boolean function $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\}$
  Each plaintext is contributes by either 1 or -1 to the correlation
- Key recovery map with plaintext rejection: $g : \mathbb{F}_2^\ell \longrightarrow \{1, -1, 0\}$
  Plaintexts can contribute 1 or -1 to the correlation, or be rejected
  The data complexity is increased by the proportion of rejected plaintexts

The Puncturing and Approximation Problems

- "Classical" key recovery map: A Boolean function $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\}$
  Each plaintext is contributes by either 1 or -1 to the correlation
- Key recovery map with plaintext rejection: $g : \mathbb{F}_2^\ell \longrightarrow \{1, -1, 0\}$
  Plaintexts can contribute 1 or -1 to the correlation, or be rejected
  The data complexity is increased by the proportion of rejected plaintexts
- Punctured key recovery map: $g g : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$
  g is obtained by changing spectrum coefficients of f to zero
  Correlation contributions are real numbers, we can consider them "weights"
  We don't know how to compute the new data complexity

The Puncturing and Approximation Problems

- "Classical" key recovery map: A Boolean function $f : \mathbb{F}_2^\ell \longrightarrow \{1, -1\}$
  Each plaintext is contributes by either 1 or -1 to the correlation
- Key recovery map with plaintext rejection: $g : \mathbb{F}_2^\ell \longrightarrow \{1, -1, 0\}$
  Plaintexts can contribute 1 or -1 to the correlation, or be rejected
  The data complexity is increased by the proportion of rejected plaintexts
- Punctured key recovery map: $gg : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$
  g is obtained by changing spectrum coefficients of f to zero
  Correlation contributions are real numbers, we can consider them "weights"
  We don't know how to compute the new data complexity
- Arbitrary real-valued approximation: $g : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$
  g is an arbitrary real function which "approximates" f

Impact of Puncturing on the Data Complexity

Data Complexity with Approximated Key Recovery

Theorem: Key Recovery Map Approximation

We can substitute the key recovery map $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ for the approximation $g : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ by increasing the data complexity by a factor $1/\rho^2$, where

$$\rho = \frac{|\langle f, g\rangle|}{\|f\|_2 \cdot \|g\|_2} = \frac{\frac{1}{2^\ell}\sum_{x\in\mathbb{F}_2^\ell} f(x)g(x)}{\sqrt{\frac{1}{2^\ell}\sum_{x\in\mathbb{F}_2^\ell} f(x)^2} \cdot \sqrt{\frac{1}{2^\ell}\sum_{x\in\mathbb{F}_2^\ell} g(x)^2}}$$

The Pearson correlation coefficient ρ can also be obtained from $\widehat{f}$ and $\widehat{g}$:

$$\rho = \frac{\sum_{u\in\mathbb{F}_2^\ell} \widehat{f}(u)\widehat{g}(u)}{\sqrt{\sum_{u\in\mathbb{F}_2^\ell} \widehat{f}(u)^2} \cdot \sqrt{\sum_{u\in\mathbb{F}_2^\ell} \widehat{g}(u)^2}}.$$

Data Complexity with Approximated Key Recovery *(cont.)*

Example: We can check this result in the specific case of plaintext rejection

We assume we reject a fraction ϵ of the inputs of $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$

Data Complexity with Approximated Key Recovery *(cont.)*

Example: We can check this result in the specific case of plaintext rejection

We assume we reject a fraction ϵ of the inputs of $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$

- $\|f\|_2 = \sqrt{\frac{1}{2^\ell} \sum_{x\in\mathbb{F}_2^\ell} (\pm 1)^2} = 1$ because f is a Boolean function
- $\|g\|_2 = \sqrt{\frac{1}{2^\ell} \left(\sum_{g(x)\neq 0} (\pm 1)^2 + \sum_{g(x)=0} 0^2 \right)} = \sqrt{1-\epsilon}$
- $\langle f, g\rangle = \frac{1}{2^\ell} \left(\sum_{g(x)\neq 0} (\pm 1)^2 + \sum_{g(x)=0} (\pm 1)\cdot 0 \right) = 1-\epsilon$

Data Complexity with Approximated Key Recovery *(cont.)*

Example: We can check this result in the specific case of plaintext rejection

We assume we reject a fraction ϵ of the inputs of $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$

- $\|f\|_2 = \sqrt{\frac{1}{2^\ell}\sum_{x\in\mathbb{F}_2^\ell}(\pm 1)^2} = 1$ because f is a Boolean function
- $\|g\|_2 = \sqrt{\frac{1}{2^\ell}\left(\sum_{g(x)\neq 0}(\pm 1)^2 + \sum_{g(x)=0} 0^2\right)} = \sqrt{1-\epsilon}$
- $\langle f, g\rangle = \frac{1}{2^\ell}\left(\sum_{g(x)\neq 0}(\pm 1)^2 + \sum_{g(x)=0}(\pm 1)\cdot 0\right) = 1-\epsilon$

So $\rho = \frac{|\langle f,g\rangle|}{\|f\|_2\cdot\|g\|_2} = \sqrt{1-\epsilon}$, and the data complexity is $\frac{1}{\rho^2}N = \frac{1}{1-\epsilon}N$ as expected

Data Complexity with Approximated Key Recovery *(cont.)*

Example: We can check this result in the specific case of plaintext rejection

We assume we reject a fraction ϵ of the inputs of $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$

- $\|f\|_2 = \sqrt{\frac{1}{2^\ell} \sum_{x \in \mathbb{F}_2^\ell} (\pm 1)^2} = 1$ because f is a Boolean function
- $\|g\|_2 = \sqrt{\frac{1}{2^\ell} \left(\sum_{g(x) \neq 0} (\pm 1)^2 + \sum_{g(x)=0} 0^2 \right)} = \sqrt{1 - \epsilon}$
- $\langle f, g \rangle = \frac{1}{2^\ell} \left(\sum_{g(x) \neq 0} (\pm 1)^2 + \sum_{g(x)=0} (\pm 1) \cdot 0 \right) = 1 - \epsilon$

So $\rho = \frac{|\langle f, g \rangle|}{\|f\|_2 \cdot \|g\|_2} = \sqrt{1 - \epsilon}$, and the data complexity is $\frac{1}{\rho^2} N = \frac{1}{1-\epsilon} N$ as expected

A more elaborate model for this case has been proposed (Wu, Li, Wang, 2024)

Puncturing Data Complexity

Key Recovery Map Walsh Spectrum Puncturing

Given the key recovery map $f : \mathbb{F}_2^\ell \longrightarrow \mathbb{F}_2$, a puncture set is any $\mathcal{P} \subseteq \mathbb{F}_2^\ell$. We define $g : \mathbb{F}_2^\ell \longrightarrow \mathbb{R}$ as a function whose Walsh spectrum is:

$$\widehat{g}(u) = \begin{cases} \widehat{f}(u) & \text{if } u \notin \mathcal{P} \\ 0 & \text{if } u \in \mathcal{P} \end{cases}$$

Theorem: Puncturing Data Complexity

The data complexity is increased by a factor of

$$\frac{1}{1-\epsilon}, \text{ where } \epsilon = \sum_{u \in \mathcal{P}} \widehat{f}(u)^2$$

Conclusion and Applications

Some Open Problems

- Further applications.

Some Open Problems

- Further applications.
- Optimization Strategies: Is there a general way to find good puncturing sets?

Some Open Problems

- Further applications.
- Optimization Strategies: Is there a general way to find good puncturing sets?
- Automatization: Developing software for key recovery attack design

Some Open Problems

- Further applications.
- Optimization Strategies: Is there a general way to find good puncturing sets?
- Automatization: Developing software for key recovery attack design
- Dependence: Can it be incorporated into the statistical model?

Some Open Problems

- Further applications.
- Optimization Strategies: Is there a general way to find good puncturing sets?
- Automatization: Developing software for key recovery attack design
- Dependence: Can it be incorporated into the statistical model?
- Key Recovery vs. Distinguishers: Both steps are becoming mixed: can we describe both under the same model?

Summary of Applications

Target	Rounds	Data	Complexity: Time	Memory	P_S	Comment
Serpent (192-bit)	**12**	$2^{127.5}$ KP	$2^{189.74}$	$2^{182.00}$	80%	Most rounds
Serpent (256-bit)	12	$2^{125.16}$ KP	$\mathbf{2^{214.36}}$	$2^{125.16}$	81%	Best time
	12	$2^{126.30}$ KP	$\mathbf{2^{210.36}}$	$2^{125.16}$	80%	Best time
GIFT-128 (General)	25	$\mathbf{2^{123.02}}$ KP	$\mathbf{2^{124.61}}$	$2^{112.00}$	80%	Best data/time (for LC)
GIFT-128 (COFB)	**17**	$2^{62.10}$ KP	$2^{125.09}$	$2^{62.10}$	80%	Most rounds
DES	Full	$\mathbf{2^{41.62}}$ KP	$2^{41.76}$	$\mathbf{2^{34.54}}$	70%	Almost matches best data with lower memory
NOEKEON	12	$\mathbf{2^{119.55}}$ KP	$\mathbf{2^{120.63}}$	$\mathbf{2^{115.00}}$	80%	Best data/time/memory

Data Complexity with Approximated Key Recovery *(cont.)*

Sketch of the proof:

The main idea is to separate g into two orthogonal components:

$$g = \frac{\langle f, g \rangle^2}{\|f\|_2} f + g^{\perp}, \text{ where } \langle f, g^{\perp} \rangle = 0$$

Data Complexity with Approximated Key Recovery *(cont.)*

Sketch of the proof:

The main idea is to separate g into two orthogonal components:

$$g = \frac{\langle f, g \rangle^2}{\|f\|_2} f + g^{\perp}, \text{ where } \langle f, g^{\perp} \rangle = 0$$

- The statistical behaviour of $\frac{\langle f, g \rangle^2}{\|f\|_2} f$ can be obtained by applying existing models for linear cryptanalysis (Blondeau, Nyberg, 2017)
- We assume random (and independent) behaviour for $g^{\perp}$

Data Complexity with Approximated Key Recovery *(cont.)*

Sketch of the proof:

The main idea is to separate g into two orthogonal components:

$$g = \frac{\langle f, g \rangle^2}{\|f\|_2} f + g^{\perp}, \text{ where } \langle f, g^{\perp} \rangle = 0$$

- The statistical behaviour of $\frac{\langle f, g \rangle^2}{\|f\|_2} f$ can be obtained by applying existing models for linear cryptanalysis (Blondeau, Nyberg, 2017)
- We assume random (and independent) behaviour for $g^{\perp}$

We deduce the mean and variance of the experimental correlation, and they coincide (up to scaling) with those for f under a data sample of size $N^* = N/\rho^2$

Some Further Examples

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	0.75	−0.75	1.25	0.75	−0.75	0.75	0.75	−0.75	1.25	−1.25	−1.25	1.25	0.75	−0.75	−0.75	−1.25	$\frac{16}{15}N$
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

Some Further Examples

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	0.75	−0.75	1.25	0.75	−0.75	0.75	0.75	−0.75	1.25	−1.25	−1.25	1.25	0.75	−0.75	−0.75	−1.25	$\frac{16}{15}N$
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

We can also puncture all the coefficients of Hamming weight 3 or 4:

S_{new}	1	−1	2	2	1	−1	0	0	1	−1	0	0	1	−1	−2	−2	$\frac{16}{6}N$
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	0	4	0	−4	0	0	0	0	0	

Some Further Examples

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S	1	−1	1	1	−1	1	1	−1	1	−1	−1	1	1	−1	−1	−1	N
$\hat{S}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	4	
S_{new}	0.75	−0.75	1.25	0.75	−0.75	0.75	0.75	−0.75	1.25	−1.25	−1.25	1.25	0.75	−0.75	−0.75	−1.25	$\frac{16}{15}N$
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	8	4	0	−4	−8	0	4	0	0	

We can also puncture all the coefficients of Hamming weight 3 or 4:

S_{new}	1	−1	2	2	1	−1	0	0	1	−1	0	0	1	−1	−2	−2	$\frac{16}{6}N$
$\widehat{S_{\text{new}}}$	0	4	0	4	4	0	−4	0	4	0	−4	0	0	0	0	0	

Or just keep the largest coefficients:

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF	
S_{new}	0	0	0	0	−1	1	1	−1	1	−1	−1	1	0	0	0	0	$2N$
$\widehat{S_{\text{new}}}$	0	0	0	0	0	0	0	8	0	0	0	−8	0	0	0	0	