RUHR-UNIVERSITÄT BOCHUM

Security Arguments and Tool-based Design of Block Ciphers

PhD Defense

December 13th, 2019

Arbeitsgruppe Symmetrische Kryptographie, Horst-Görtz-Institut für IT Sicherheit, Ruhr-Universität Bochum

Friedrich Wiemer

Topics of the Thesis

Based on mainly four papers:

Security Arguments

- A. Canteaut, L. Kölsch, and F. Wiemer. *Observations on the DLCT and Absolute Indicators*. 2019. iacr: 2019/848
- A. Canteaut, V. Lallemand, G. Leander, P. Neumann, and F. Wiemer. “Bison – Instantiating the Whitened Swap-Or-Not Construction”. In: *EUROCRYPT 2019, Part III*. 2019, pp. 585–616. iacr: 2018/1011

Tool-based Design

- T. Kranz, G. Leander, K. Stoffelen, and F. Wiemer. “Shorter Linear Straight-Line Programs for MDS Matrices”. In: *IACR Trans. Symm. Cryptol.* 2017.4 (2017), pp. 188–211. iacr: 2017/1151
- G. Leander, C. Tezcan, and F. Wiemer. “Searching for Subspace Trails and Truncated Differentials”. In: *IACR Trans. Symm. Cryptol.* 2018.1 (2018), pp. 74–100

Topics of the Thesis

RUB

Based on mainly four papers:

Security Arguments

- A. Canteaut, L. Kölsch, and F. Wiemer. *Observations on the DLCT and Absolute Indicators*. 2019. iacr: 2019/848
- A. Canteaut, V. Lallemand, G. Leander, P. Neumann, and F. Wiemer. "Bison – Instantiating the Whitened Swap-Or-Not Construction". In: *EUROCRYPT 2019, Part III*. 2019, pp. 585–616. iacr: 2018/1011

Tool-based Design

- T. Kranz, G. Leander, K. Stoffelen, and F. Wiemer. "Shorter Linear Straight-Line Programs for MDS Matrices". In: *IACR Trans. Symm. Cryptol.* 2017.4 (2017), pp. 188–211. iacr: 2017/1151
- G. Leander, C. Tezcan, and F. Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: *IACR Trans. Symm. Cryptol.* 2018.1 (2018), pp. 74–100

The General Setting

Block Ciphers and Security Notion

Block Cipher

- $\mathcal{E} = (E, D)$ tuple of permutation families.
- Here:
  block lengh $=$ *key length* $= n$
  encryption $E : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$
  decryption $D : \mathbb{F}_2^n \times \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$
- *Correctness*:
  $\forall k \in \mathbb{F}_2^n : D(k, E(k, x)) = x \quad \forall x \in \mathbb{F}_2^n$

The General Setting

Block Ciphers and Security Notion

RUB

Block Cipher

- $\mathcal{E} = (E, D)$ tuple of permutation families.
- Here:
  block lengh $=$ *key length* $= n$
  encryption $E : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$
  decryption $D : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$
- *Correctness*:
  $\forall k \in \mathbb{F}_2^n : D(k, E(k, x)) = x \quad \forall x \in \mathbb{F}_2^n$

Security

Given a block cipher $\mathcal{E} = (E, D)$.

- $F_k = E(k, \cdot)$
- $\mathrm{Perm}_n = \left\{ f : \mathbb{F}_2^n \to \mathbb{F}_2^n \;\middle|\; f \text{ is permutation} \right\}$

If all efficient adversaries $\mathcal{A}$ have a *negligible* distinguishing advantage

$$\left| \Pr_{k \in_R \mathbb{F}_2^n}[\mathcal{A}^{F_k}(1^n) = 1] - \Pr_{f \in_R \mathrm{Perm}_n}[\mathcal{A}^{f}(1^n) = 1] \right| ,$$

then $\mathcal{E}$ is *secure*.

The General Setting

Block Ciphers and Security Notion

Block Cipher

- $\mathcal{E} = (E, D)$ tuple of permutation families.
- Here:
  block lengh $=$ *key length* $= n$
  encryption $E : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$
  decryption $D : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n$
- *Correctness*:
  $\forall k \in \mathbb{F}_2^n : D(k, E(k, x)) = x \quad \forall x \in \mathbb{F}_2^n$

Security

Given a block cipher $\mathcal{E} = (E, D)$.

- $F_k = E(k, \cdot)$
- $\mathrm{Perm}_n = \{ f : \mathbb{F}_2^n \to \mathbb{F}_2^n \mid f$ is permutation$\}$

If all efficient adversaries $\mathcal{A}$ have a *negligible* distinguishing advantage

$$\left| \Pr_{k \in_R \mathbb{F}_2^n}[\mathcal{A}^{F_k}(1^n) = 1] - \Pr_{f \in_R \mathrm{Perm}_n}[\mathcal{A}^{f}(1^n) = 1] \right|,$$

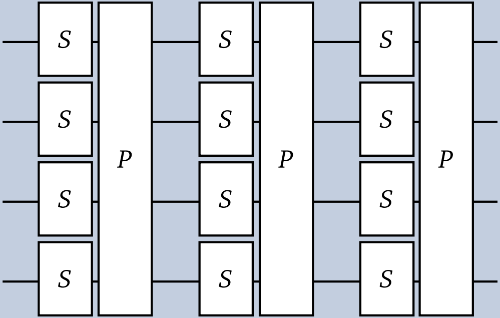
then $\mathcal{E}$ is *secure*.

In practice: security of a block cipher always security against known attacks.

Substitution Permutation Networks

The most common design structure for block ciphers

Exemplary SPN Structure

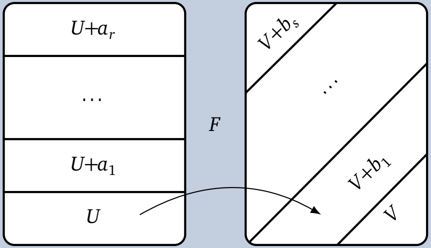


- S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ provides *confusion* and non-linearity on small blocks (typically $3 \leqslant n \leqslant 8$)
- Linear layer $P : \mathbb{F}_2^{kn} \to \mathbb{F}_2^{kn}$ provides *diffusion* and spreads the S-box influence over the whole state
- Key-alternating: the round keys are added in between the rounds

Overview

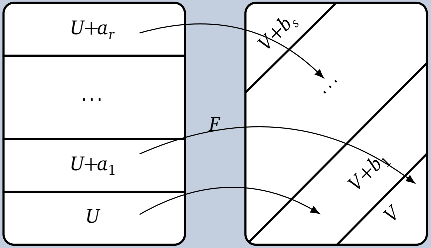
Subspace Trail Cryptanalysis

Main Idea of Subspace Trails



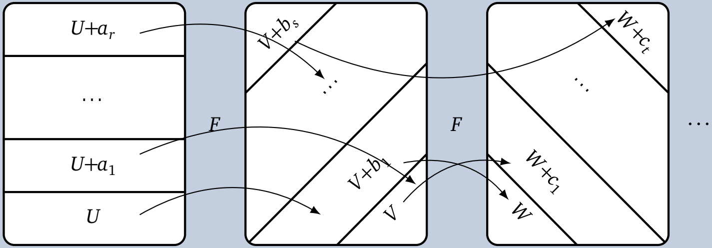
Subspace Trail Cryptanalysis

Main Idea of Subspace Trails



Subspace Trail Cryptanalysis

Main Idea of Subspace Trails



Subspace Trail Cryptanalysis [GRR16] (FSE'16)

Let $U_0, \dots, U_r \subseteq \mathbb{F}_2^n$, and $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Then these form a *subspace trail* (ST), $U_0 \xrightarrow{F} \cdots \xrightarrow{F} U_r$, iff

$$\forall a \in U_i^\perp : \exists b \in U_{i+1}^\perp : \qquad F(U_i + a) \subseteq U_{i+1} + b$$

Our Goal

Find a solution to

Problem: Security against Subspace Trails

Given an SPN with round function F, consisting of

- k parallel applications of an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and
- a linear layer $L : \mathbb{F}_2^{kn} \to \mathbb{F}_2^{kn}$.

Compute an upper bound on the length of any subspace trail through the cipher.

Subspace Propagation

Given a starting subspace U, we can efficiently compute the corresponding longest subspace trail.

Lemma

Let $U \xrightarrow{F} V$ be a ST. Then for all $u \in U$ and all x: $F(x) + F(x+u) \in V$.

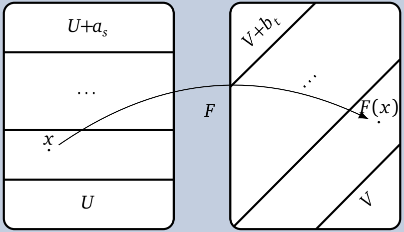
Subspace Propagation

Given a starting subspace U, we can efficiently compute the corresponding longest subspace trail.

Lemma

Let $U \xrightarrow{F} V$ *be a ST. Then for all* $u \in U$ *and all* x*:* $F(x) + F(x+u) \in V$.

Proof



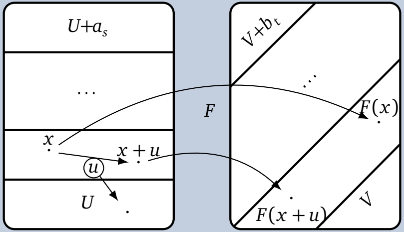
Subspace Propagation

Given a starting subspace U, we can efficiently compute the corresponding longest subspace trail.

Lemma

Let $U \xrightarrow{F} V$ *be a ST. Then for all* $u \in U$ *and all* x: $F(x) + F(x+u) \in V$.

Proof



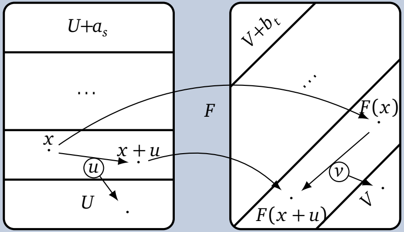
Subspace Propagation

Given a starting subspace U, we can efficiently compute the corresponding longest subspace trail.

Lemma

Let $U \xrightarrow{F} V$ *be a ST. Then for all* $u \in U$ *and all* x: $F(x) + F(x+u) \in V$.

Proof



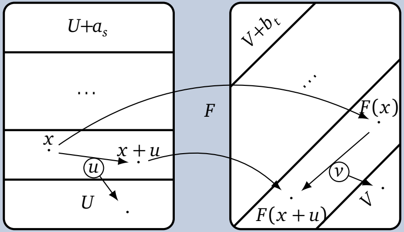
Subspace Propagation

Given a starting subspace U, we can efficiently compute the corresponding longest subspace trail.

Lemma

Let $U \xrightarrow{F} V$ *be a ST. Then for all* $u \in U$ *and all* x: $F(x) + F(x+u) \in V$.

Proof



Computing the subspace trail

- To compute the next subspace, we have to compute the image of the derivatives.

Propagate a Basis

Actually it is enough to compute only the image of the derivatives in direction of U's basis vectors.

Lemma

Given $U \subseteq \mathbb{F}_2^n$ *with basis* $\{b_1, \ldots, b_k\}$. *Then* $\mathrm{Span}\left\{\bigcup_{u \in U} \mathrm{Im}\, \Delta_u(F)\right\} = \mathrm{Span}\left\{\bigcup_{b_i} \mathrm{Im}\, \Delta_{b_i}(F)\right\}$.

Propagate a Basis

RUB

Actually it is enough to compute only the image of the derivatives in direction of U's basis vectors.

Lemma

Given $U \subseteq \mathbb{F}_2^n$ *with basis* $\{b_1, \ldots, b_k\}$. *Then* $\operatorname{Span}\left\{\bigcup_{u \in U} \operatorname{Im} \Delta_u(F)\right\} = \operatorname{Span}\left\{\bigcup_{b_i} \operatorname{Im} \Delta_{b_i}(F)\right\}$.

Proof: $\supseteq$ trivial, $\subseteq$ by induction over the dimension k of U

Let $u = \sum_{i=1}^{k} \lambda_i b_i$ and $v \in \operatorname{Im} \Delta_u(F)$, i.e. there exists an x s.t.

$$v = F(x) + F(x + \sum_{i=1}^{k} \lambda_i b_i) = F(y + \lambda_k b_k) + F(y + \sum_{i=1}^{k-1} \lambda_i b_i) = \lambda_k \Delta_{b_k}(F)(y) + \lambda' \Delta_{u'}(F)(y)\,.$$

Thus $v \in \operatorname{Span}\left\{\operatorname{Im} \Delta_{b_k}(F) \cup \operatorname{Im} \Delta_{u'}(F)\right\}$, where u' is contained in a $(k-1)$ dimensional subspace.

RUB

`ComputeTrail` Algorithm

Computation of Subspace Trails

Input: A nonlinear function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, a subspace U.
Output: A subspace trail $U \rightrightarrows^F \cdots \rightrightarrows^F V$.

1 **function** `ComputeTrail`(F, U)
2   **if** $\dim U = n$ **then return** U
3     $V \leftarrow \emptyset$
4   **for** u_i basis vectors of U **do**
5   **for** enough $x \in_{\mathrm{R}} \mathbb{F}_2^n$ **do**
6             $V \leftarrow V \cup \Delta_{u_i}(F)(x)$
7     $V \leftarrow \mathrm{Span}\{V\}$
8   **return** $U \rightrightarrows^F$ `ComputeTrail`(F, V)

Correctness: previous two lemmata

Runtime:

- Line 4: max. n iterations
- Line 5: $n + c$ random vectors are enough
- Overall: $\mathcal{O}(n^2)$ evaluations of F

Remaining Problem: cyclic STs

How to Bound the Length of Subspace Trails

RUB

Goal

Give an upper bound on the length of any subspace trail.

How to Bound the Length of Subspace Trails

Goal

Give an upper bound on the length of any subspace trail.

Naïve Approach I

$\forall U \subseteq \mathbb{F}_2^m$ run $\texttt{ComputeTrail}(F, U)$

Problem

Exponentially many starting subspaces.

How to Bound the Length of Subspace Trails

Goal

Give an upper bound on the length of any subspace trail.

Naïve Approach I

$\forall U \subseteq \mathbb{F}_2^m$ run $\texttt{ComputeTrail}(F, U)$

Naïve Approach II

$\forall u \subseteq \mathbb{F}_2^m \setminus \{0\}$ run $\texttt{ComputeTrail}(F, \mathrm{Span}\{u\})$

Problem

Exponentially many starting subspaces.

Problem

Still $2^m - 1$ starting subspaces.

How to Bound the Length of Subspace Trails

RUB

Goal

Give an upper bound on the length of any subspace trail.

Naïve Approach I

$\forall U \subseteq \mathbb{F}_2^m$ run $\texttt{ComputeTrail}(F, U)$

Naïve Approach II

$\forall u \subseteq \mathbb{F}_2^m \setminus \{0\}$ run $\texttt{ComputeTrail}(F, \mathrm{Span}\,\{u\})$

Problem

Exponentially many starting subspaces.

Problem

Still $2^m - 1$ starting subspaces.

Often used heuristic

Activate single S-boxes only. That is, for a round function with k S-boxes which are n-bit wide, choose $U = \{0\}^i \times V \times \{0\}^{k-i-1}$, where $V \subseteq \mathbb{F}_2^n$.

Activating a single S-box only

RUB

Problem

Heuristic not valid in general when we want to prove a bound on the subspace trail length. In particular one can construct examples where the best subspace trail does activate more than one S-box in the beginning.

The good case

However, we will see next a sufficient condition for the case when the heuristic is valid.

The Connection to Linear Structures

RUB

Let us observe how a single S-box S behaves regarding subspace trails:

Given a subpsace trail $U \overset{S}{\rightrightarrows} V$, this implies

$$\Delta_u(S)(x) \in V \qquad \text{for all } x \in \mathbb{F}_2^n \text{ and } u \in U.$$

By definition of the dual space $V^\perp$:

$$\langle \alpha, \Delta_u(S)(x) \rangle = 0 \qquad \text{for all } \alpha \in V^\perp,$$

which are exactly the *linear structures* of S:

$$\mathrm{LS}(S) := \{(\alpha, u) \mid \langle \alpha, \Delta_u(S)(x) \rangle \text{ is constant for all } x\}$$

S-boxes without Linear Structures

This observation implies that S-boxes without linear structures (e. g. the AES S-box) exhibit only two important subspace trails:

$$\{0\} \rightrightarrows \{0\} \qquad \text{and} \qquad \mathbb{F}_2^n \rightrightarrows \mathbb{F}_2^n$$

We can further show that subspace trails over an S-box layer without linear structures are direct products of the above two subspace trails.

S-boxes without Linear Structures

This observation implies that S-boxes without linear structures (e. g. the AES S-box) exhibit only two important subspace trails:

$$\{0\} \rightrightarrows \{0\} \qquad \text{and} \qquad \mathbb{F}_2^n \rightrightarrows \mathbb{F}_2^n$$

We can further show that subspace trails over an S-box layer without linear structures are direct products of the above two subspace trails.

Theorem

Let F be an S-box layer of k parallel S-boxes $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$. If S has no non-trivial linear structures, then for every subspace trail $U \rightrightarrows^F V$:

$$U = V = U_1 \times \cdots \times U_k\,,$$

with $U_i \in \left\{\{0\}, \mathbb{F}_2^n\right\}$.

S-boxes without Linear Structures

RUB

This observation implies that S-boxes without linear structures (e. g. the AES S-box) exhibit only two important subspace trails:

$$\{0\} \rightrightarrows \{0\} \qquad \text{and} \qquad \mathbb{F}_2^n \rightrightarrows \mathbb{F}_2^n$$

We can further show that subspace trails over an S-box layer without linear structures are direct products of the above two subspace trails.

Theorem

Let F be an S-box layer of k parallel S-boxes $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$. If S has no non-trivial linear structures, then for every subspace trail $U \rightrightarrows^F V$:

$$U = V = U_1 \times \cdots \times U_k \,,$$

with $U_i \in \left\{\{0\}, \mathbb{F}_2^n\right\}$.

Proof

For all $\alpha = (\alpha_1, \ldots, \alpha_k) \in V^\perp$: $\langle \alpha, \Delta_u(F)(x) \rangle = \left\langle \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_k \end{pmatrix}, \begin{pmatrix} \Delta_{u_1}(S)(x_1) \\ \vdots \\ \Delta_{u_k}(S)(x_k) \end{pmatrix} \right\rangle = \sum_{i=1}^{k} \left\langle \alpha_i, \Delta_{u_i}(S)(x_i) \right\rangle = 0$

S-boxes without Linear Structures

The length ℓ of any subspace trail is upper bounded by

$$\ell = \max_{U \in \left\{\{0\}, \mathbb{F}_2^n\right\}^k} \left| \texttt{ComputeTrail}(F, U) \right| ,$$

which needs 2^k evaluations of the `ComputeTrail` algorithm.

S-boxes with Linear Structures

RUB

Compared to the no-linear-structures-case, $V^\perp$ can now contain much more elements, namely all combinations of linear structures, such that their corresponding constants sum to zero.

Instead, we can show that (for any not-trivially-insecure S-box) the subspace after the first S-box layer contains at least one element of a specific structure:

$$W_{i,\alpha} = \{0\}^{i-1} \times \{0, \alpha\} \times \{0\}^{k-i} \ .$$

The length ℓ of any subspace trail is then upper bounded by

$$\ell = \max_{W_{i,\alpha}} \left| \mathtt{ComputeTrail}(F', W_{i,\alpha}) \right| + 1 \ ,$$

which needs $k \cdot 2^n$ evaluations of the `ComputeTrail` algorithm.

Note that F' first applies the linear layer, then the S-box layer (b/c of the skipped first S-box layer).

Conclusion

Thanks for your attention!

Applications of `ComputeTrail`

- Bound longest probability-one subspace trail
- Link to Truncated Differentials
- Finding key-recovery strategies