RUHR-UNIVERSITÄT BOCHUM

Cryptanalysis of Clyde and Shadow

July 3rd, 2019

Horst Görtz Institut für IT Sicherheit, Ruhr-Universität Bochum

Gregor Leander, and *Friedrich Wiemer*

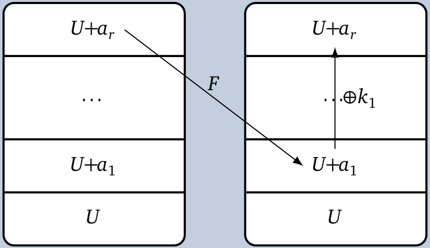
Overview

Section 1

Invariant Attacks – Round Constants

Invariant Attacks

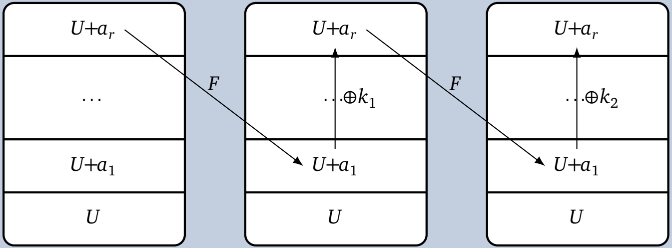
Main Idea: Invariant Subspaces



Invariant Attacks

RUB

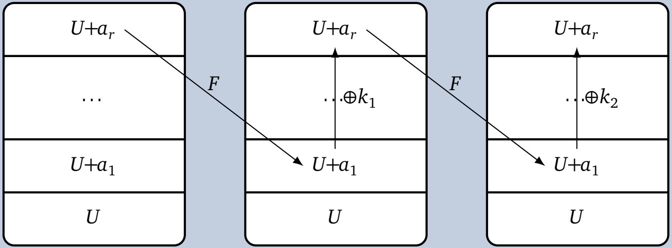
Main Idea: Invariant Subspaces



Invariant Attacks

RUB

Main Idea: Invariant Subspaces

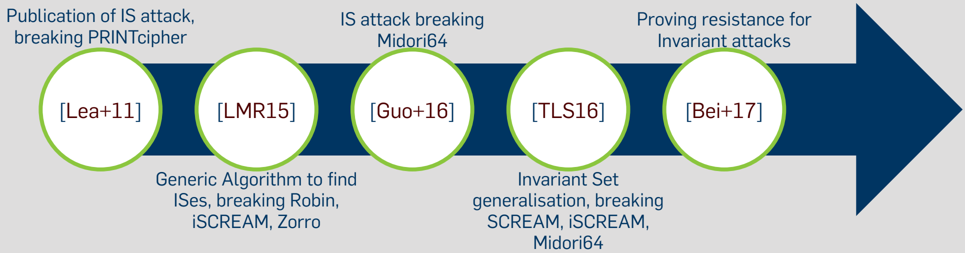


Invariant Subspace Attacks [Lea+11] (CRYPTO'11)

Let $U \subseteq \mathbb{F}_2^n$, $c, d \in U^\perp$, and $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Then U is an *invariant subspace* (IS) if and only if $F(U{+}c) = U{+}d$ and all round keys in $U{+}(c+d)$ are *weak keys*.

Invariant Attacks

A Short History



Invariant Attacks

Proving Resistance

RUB

Goal: Apply security argument from

C. Beierle, A. Canteaut, G. Leander, and Y. Rotella. "Proving Resistance Against Invariant Attacks: How to Choose the Round Constants". In: CRYPTO 2017, Part II. 2017. doi: 10.1007/978-3-319-63715-0_22. iacr: 2017/463.

What do we get from this?

- Non-existence of invariants for both parts of the ronud function (S-box and linear layer)

Issues

- Other partitionings of the round function might allow invariants (Christof B. found examples)
- Not clear how to prove the general absence of invariant attacks (best we can currently prove)
- All known attacks exploit exactly this structure (splitting in S-box and linear layer)

Invariant Attacks

Recap Security Argument (I)

Observation

- Invariants for the linear layer L and round key addition have to contain special linear structures.
- Denote by $c_1, \ldots, c_t$ the round constant differences for rounds with the same round key.
- Then the linear structures of any invariant have to contain $W_L(c_1, \ldots, c_t)$.

Invariant Attacks

Recap Security Argument (I)

RUB

Observation

- Invariants for the linear layer L and round key addition have to contain special linear structures.
- Denote by $c_1, \ldots, c_t$ the round constant differences for rounds with the same round key.
- Then the linear structures of any invariant have to contain $W_L(c_1, \ldots, c_t)$.

Linear Structures

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Then its *linear structures* are

$$\mathrm{LS} := \{a \mid f(x) + f(x+a) \text{ is constant}\} .$$

The smallest L-invariant subspace

$W_L(c_1, \ldots, c_t)$ is the *smallest L-invariant subspace* of $\mathbb{F}_2^n$ containing all c_i

$$\Leftrightarrow \forall x \in W_L(c_1, \ldots, c_t) : L(x) \in W_L(c_1, \ldots, c_t)$$

Invariant Attacks

Recap Security Argument (I)

RUB

Observation

- Invariants for the linear layer L and round key addition have to contain special linear structures.
- Denote by $c_1, \ldots, c_t$ the round constant differences for rounds with the same round key.
- Then the linear structures of any invariant have to contain $W_L(c_1, \ldots, c_t)$.

Linear Structures

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Then its *linear structures* are

$$\mathrm{LS} := \{a \mid f(x) + f(x+a) \text{ is constant}\} .$$

The smallest L-invariant subspace

$W_L(c_1, \ldots, c_t)$ is the *smallest L-invariant subspace* of $\mathbb{F}_2^n$ containing all c_i

$$\Leftrightarrow \forall x \in W_L(c_1, \ldots, c_t) : L(x) \in W_L(c_1, \ldots, c_t)$$

The simple case

If $W_L(c_1, \ldots, c_t) = \mathbb{F}_2^n$, only trivial invariants for L and key addition are possible (constant 0 and 1 function).

Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$

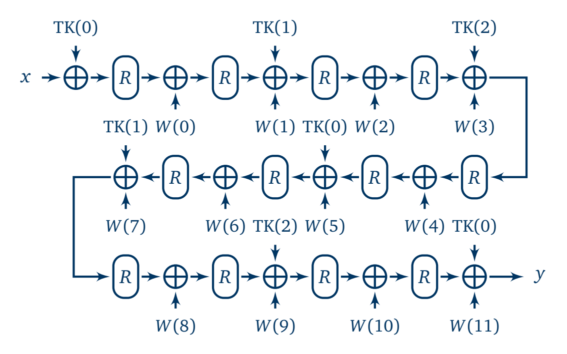
Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$



$$D = D_{\mathrm{TK}(0)} \cup D_{\mathrm{TK}(1)} \cup D_{\mathrm{TK}(2)} \cup D_0$$

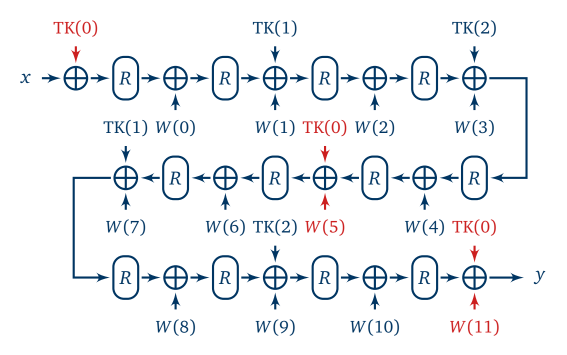
Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$



$$D = D_{\mathrm{TK}(0)} \cup D_{\mathrm{TK}(1)} \cup D_{\mathrm{TK}(2)} \cup D_0$$

$$D_{\mathrm{TK}(0)} = \{0 + W(5), 0 + W(11), W(5) + W(11)\}$$

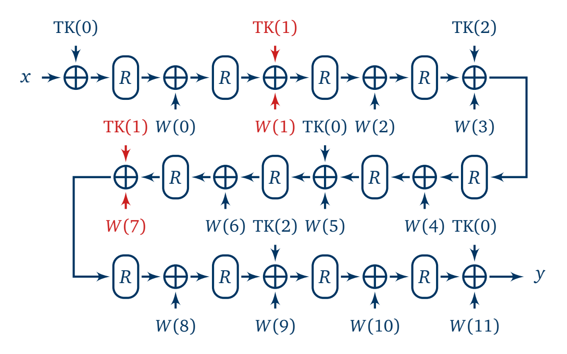
Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$



$$D = D_{\mathrm{TK}(0)} \cup D_{\mathrm{TK}(1)} \cup D_{\mathrm{TK}(2)} \cup D_0$$

$$D_{\mathrm{TK}(0)} = \{0 + W(5), 0 + W(11), W(5) + W(11)\}$$

$$D_{\mathrm{TK}(1)} = \{W(1) + W(7)\}$$

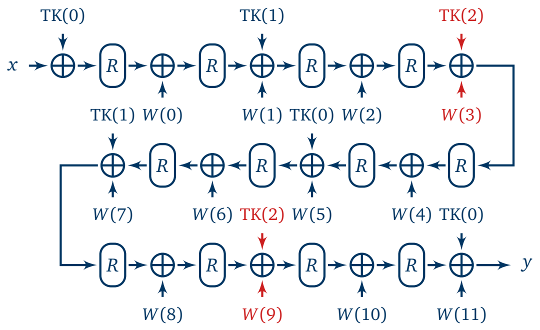
Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences: (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$



$$D = D_{\mathrm{TK}(0)} \cup D_{\mathrm{TK}(1)} \cup D_{\mathrm{TK}(2)} \cup D_0$$

$$D_{\mathrm{TK}(0)} = \{0 + W(5), 0 + W(11), W(5) + W(11)\}$$

$$D_{\mathrm{TK}(1)} = \{W(1) + W(7)\}$$

$$D_{\mathrm{TK}(2)} = \{W(3) + W(9)\}$$

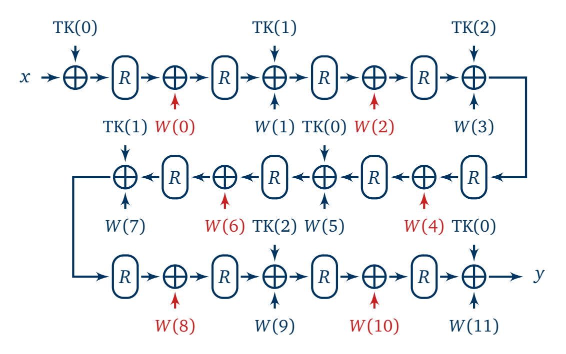
Invariant Attacks

Recap Security Argument (II)

Application to Clyde

- Find the important round constant differences:
  (the differences where the same tweakey is added)

Set of RC differences D below with $|D| = 20$



$$D = D_{\mathrm{TK}(0)} \cup D_{\mathrm{TK}(1)} \cup D_{\mathrm{TK}(2)} \cup D_0$$

$$D_{\mathrm{TK}(0)} = \{0 + W(5), 0 + W(11), W(5) + W(11)\}$$

$$D_{\mathrm{TK}(1)} = \{W(1) + W(7)\}$$

$$D_{\mathrm{TK}(2)} = \{W(3) + W(9)\}$$

$$D_0 = \left\{a + b \;\middle|\; a, b \in D', a \neq b\right\}$$

$$D' = \{W(0), W(2), W(4), W(6), W(8), W(10)\}$$

Invariant Attacks

Application to Clyde

- Computing W_L is efficiently doable (takes $\approx$ 10 seconds on my laptop).
- For the round constants chosen for Clyde, $\dim W_L(D) = 128 = n$.
- Thus, we can apply:

Proposition 2 [Bei+17]

Suppose that the dimension of $W_L(D)$ is n. Then any invariant g is constant (and thus trivial).

- We conclude that we cannot find any non-trivial g for Clyde which is at the same time invariant for the S-box layer and for the linear layer.

Invariant Attacks

Improvable?

RUB

Bounding the dimension of W_L, [Bei+17, Theorem 1]

Given a linear layer L. Denote by Q_i its *invariant factors*. Then

$$\max_{c_1,\ldots,c_t \in \mathbb{F}_2^n} \dim W_L(c_1,\ldots,c_t) = \sum_{i=1}^{t} \deg Q_i \; .$$

Invariant Attacks

Improvable?

RUB

Bounding the dimension of W_L, [Bei+17, Theorem 1]

Given a linear layer L. Denote by Q_i its *invariant factors*. Then

$$\max_{c_1,\dots,c_t \in \mathbb{F}_2^n} \dim W_L(c_1,\dots,c_t) = \sum_{i=1}^{t} \deg Q_i \; .$$

Application to Clyde

- Compute invariant factors of linear layer:
- This gives a lower bound on the number of rounds:

Invariant Attacks

Improvable?

RUB

Bounding the dimension of W_L, [Bei+17, Theorem 1]

Given a linear layer L. Denote by Q_i its *invariant factors*. Then

$$\max_{c_1,\ldots,c_t \in \mathbb{F}_2^n} \dim W_L(c_1,\ldots,c_t) = \sum_{i=1}^{t} \deg Q_i\ .$$

Application to Clyde

- Compute invariant factors of linear layer: $4 \times (x^{32}+1)$
- This gives a lower bound on the number of rounds: 3 steps/6 rounds

Invariant Attacks

Improvable?

RUB

Bounding the dimension of W_L, [Bei+17, Theorem 1]

Given a linear layer L. Denote by Q_i its *invariant factors*. Then

$$\max_{c_1,\ldots,c_t \in \mathbb{F}_2^n} \dim W_L(c_1,\ldots,c_t) = \sum_{i=1}^{t} \deg Q_i \,.$$

Application to Clyde

- Compute invariant factors of linear layer: $4 \times (x^{32}+1)$
- This gives a lower bound on the number of rounds: 3 steps/6 rounds

- 3 stps/6 rnds: $\dim W_L(c_1,\ldots,c_4) = 96$
- 4 stps/8 rnds: $\dim W_L(c_1,\ldots,c_8) = 128$
- 5 stps/10 rnds: $\dim W_L(c_1,\ldots,c_{13}) = 128$
- 6 stps/12 rnds: $\dim W_L(c_1,\ldots,c_{20}) = 128$

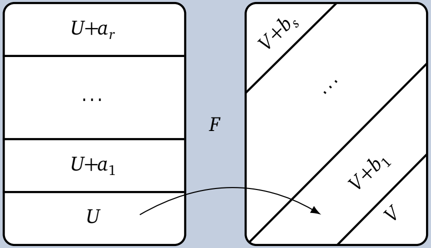
Section 2

Subspace Trails

Probability 1 Truncated Differentials

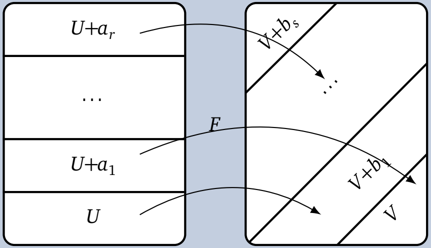
Subspace Trails

Main Idea: Subspace Trails



Subspace Trails

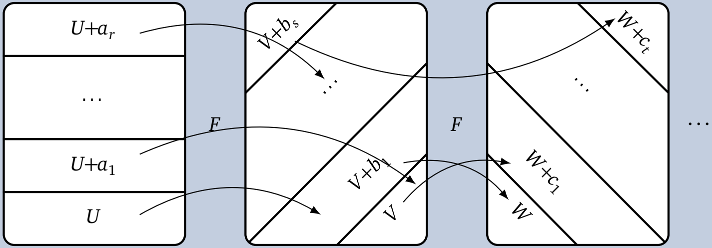
Main Idea: Subspace Trails



Subspace Trails

RUB

Main Idea: 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

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

Computing Subspace Trails

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*:* $F(x) + F(x+u) \in V$.

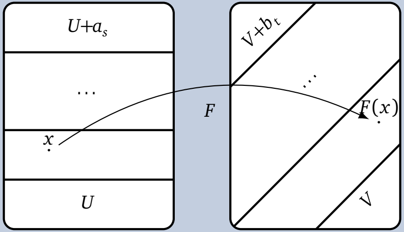
Computing Subspace Trails

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



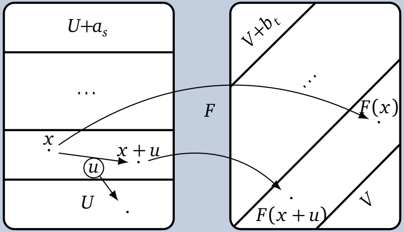
Computing Subspace Trails

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 Subspace Trails

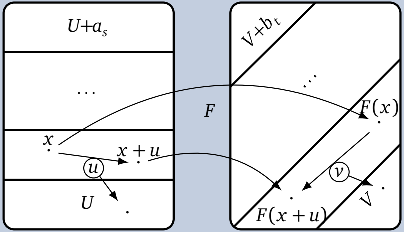
RUB

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 Subspace Trails

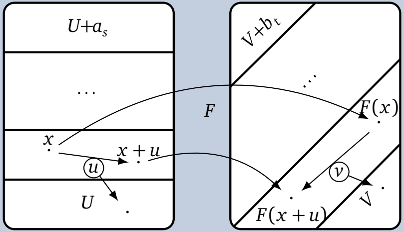
RUB

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.

Computing Subspace Trails

Algorithm

Compute Subspace Trails

Input: A nonlinear, bijective function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a subspace U.
Output: The longest ST starting in U over F.

function $\mathrm{Compute\ Trail}(F, U)$
  if $\dim(U) = n$ **then**
    return U
  $V \leftarrow \emptyset$
  for u_i basis vectors of U **do**
    for enough $x \in_{\mathrm{R}} \mathbb{F}_2^n$ **do** ▷ e. g. $n + 20$ x's are enough
      $V \leftarrow V \cup \Delta_{u_i}(F)(x)$ ▷ $\Delta_a(F)(x) := F(x) + F(x + a)$
  $V \leftarrow \mathrm{span}(V)$
  return the subspace trail $U \to \mathrm{Compute\ Trail}(F, V)$

Subspace Trails

Proving Resistance

Goal: Apply security argument from

G. Leander, C. Tezcan, and F. Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: ToSC 2018.1 (2018). doi: 10.13154/tosc.v2018.i1.74-100.

What do we get from this?

- (Tight) upper bound on the length of any ST for an SPN construction

Why is the Compute Trail algorithm not enough?

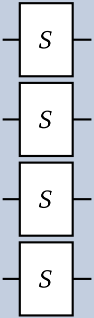
- Exhaustively checking all possible starting points is to costly.

Subspace Trails

How to bound the length of any subspace trail

Observation

U



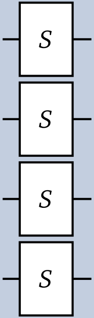
$$V \ni \begin{pmatrix} 0 \\ \alpha \\ 0 \\ 0 \end{pmatrix}$$

Subspace Trails

How to bound the length of any subspace trail

Observation

U



$$V \ni \begin{pmatrix} 0 \\ \alpha \\ 0 \\ 0 \end{pmatrix}$$

Algorithm Idea

Compute the subspace trails for any starting point $w_{i,\alpha} \in W$, with

$$w_{i,\alpha} := (\underbrace{0, \ldots, 0}_{i-1}, \alpha, 0, \ldots, 0)$$

Complexity (Size of W)

For an S-box layer $S : \mathbb{F}_2^{kn} \to \mathbb{F}_2^{kn}$ with k S-boxes, each n-bit:

$$|W| = k \cdot (2^n - 1)$$

Subspace Trails

Algorithm

RUB

Generic Subspace Trail Search

Input: A linear layer matrix $M : \mathbb{F}_2^{n\cdot k\times n\cdot k}$, and an S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$.
Output: A bound on the length of all STs over $F = M \circ S^k$.

```
function Generic Subspace Trail Length(M, S)
    empty list L
    for possible initial subspaces represented by w_{i,α} ∈ W do    ▷ Overall k·(2^n − 1) iterations
        L.append(Compute Trail(S^k ∘ M, {w_{i,α}}))                 ▷ S^k denotes the S-box layer
    return max {len(t) | t ∈ L}
```

Overall Complexity

Algorithm	Compute Trail	Generic Subspace Trail Length	Overall	Clyde	Shadow
Complexity	$\mathcal{O}(k^2n^2)$	$\mathcal{O}(k2^n)$	$\mathcal{O}(k^3n^22^n)$	2^{23}	2^{29}

Subspace Trails

Results

Clyde

- Generic Subspace Trail Length Bound:
  2 (+1) Rounds

Shadow

- Generic Subspace Trail Length Bound:
  4 (+1) Rounds

Section 3

Division Property

(Disclaimer)

Division Property

Main Idea: (Bit-based) Division Property

- Generalisation of Integral and Higher Order Differential attacks (Degree-based)
- Captures properties of bits in a set (e. g. linear combination of bits is balanced)
- For standard integral attacks: zero-sum, all or constant
- The Division Property allows to capture properties “in between” these (even if they do not have such a nice description as e. g. the zero-sum)

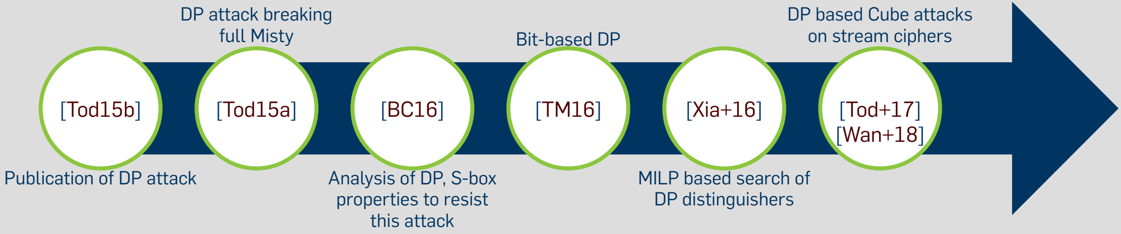
Division Property

A set $X \subseteq \mathbb{F}_2^n$ has Division Property (DP) $\mathcal{D}_K^n$ if for all $u \in \mathbb{F}_2^n$ with $\forall k \in K : u \prec k$

$$\sum_{x \in X} x^u = \sum_{x \in X} \prod_{i=1}^{n} x_i^{u_i} = 0\,.$$

Division Property

Related Work



Division Trails

RUB

Propagating (Bit-Based) Division Properties

$$\mathrm{copy}: x \mapsto (x, x)$$

$$\mathcal{D}^1_x \xrightarrow{\mathrm{copy}} \begin{cases} \mathcal{D}^2_{(0,0)} & \text{if } x = 0 \\ \mathcal{D}^2_{(0,1),(1,0)} & \text{if } x = 1 \end{cases}$$

$$\mathrm{xor}: (x, y) \mapsto x + y$$

$$\mathcal{D}^2_{(k_0,k_1)} \xrightarrow{\mathrm{xor}} \mathcal{D}^1_{k_0+k_1}$$

Division Trails

RUB

Propagating (Bit-Based) Division Properties

$$\mathrm{copy}: x \mapsto (x, x)$$

$$\mathcal{D}^1_x \xrightarrow{\mathrm{copy}} \begin{cases} \mathcal{D}^2_{(0,0)} & \text{if } x = 0 \\ \mathcal{D}^2_{(0,1),(1,0)} & \text{if } x = 1 \end{cases}$$

$$\mathrm{xor}: (x, y) \mapsto x + y$$

$$\mathcal{D}^2_{(k_0,k_1)} \xrightarrow{\mathrm{xor}} \mathcal{D}^1_{k_0+k_1}$$

S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$:
see [Xia+16, Algorithm 2],
computes for all $u \in \mathbb{F}_2^n$

$$\mathcal{D}^n_u \xrightarrow{S} \mathcal{D}^n_V$$

s. t. $u \to v$ is a DT $\forall v \in V$.

Division Trails

Propagating (Bit-Based) Division Properties

$$\text{copy} : x \mapsto (x, x)$$

$$\mathcal{D}^1_x \xrightarrow{\text{copy}} \begin{cases} \mathcal{D}^2_{(0,0)} & \text{if } x = 0 \\ \mathcal{D}^2_{(0,1),(1,0)} & \text{if } x = 1 \end{cases}$$

$$\text{xor} : (x, y) \mapsto x + y$$

$$\mathcal{D}^2_{(k_0,k_1)} \xrightarrow{\text{xor}} \mathcal{D}^1_{k_0+k_1}$$

S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$:
see [Xia+16, Algorithm 2],
computes for all $u \in \mathbb{F}_2^n$

$$\mathcal{D}^n_u \xrightarrow{S} \mathcal{D}^n_V$$

s. t. $u \to v$ is a DT $\forall v \in V$.

Division Trail

Given a round function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and $K_i \subseteq \mathbb{F}_2^n$. Assume that

$$\forall k_i \in K_i : \exists k_{i+1} \in K_{i+1} \text{ s. t. } \mathcal{D}^n_{k_i} \to \mathcal{D}^n_{k_{i+1}} .$$

We call such a $(k_0, k_1, \ldots, k_r)$ an r-round Division Trail (DT).

Division Property

RUB

Goal: Apply security argument from

Z. Xiang, W. Zhang, Z. Bao, and D. Lin. "Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers". In: ASIACRYPT 2016, Part I. 2016. doi: 10.1007/978-3-662-53887-6_24. iacr: 2016/857.

What do we get from this?

Number of rounds for which a division property/integral distinguisher exists.

Approach (similiar to Subspace Trails)

- Pick starting DPs in a way that covers all possibilities
- Model division trail propagations as MILP
- Find solutions for this over increasing number of rounds

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0,1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Division Property

Modeling Propagation Rules: copy

Based on eprint's 2016/392, 2016/811, and 2016/1101

Propagation Rule

$$\mathsf{copy} : x \mapsto (x, x)$$

$$\mathcal{D}^1_x \xrightarrow{\mathsf{copy}} \begin{cases} \mathcal{D}^2_{(0,0)} & \text{if } x = 0 \\ \mathcal{D}^2_{(0,1),(1,0)} & \text{if } x = 1 \end{cases}$$

Valid Transitions

1. $(0) \xrightarrow{\mathsf{copy}} (0, 0)$
2. $(1) \xrightarrow{\mathsf{copy}} (0, 1)$
3. $(1) \xrightarrow{\mathsf{copy}} (1, 0)$

Division Property

Modeling Propagation Rules: copy

RUB

Based on eprint's 2016/392, 2016/811, and 2016/1101

Propagation Rule

$$\text{copy} : x \mapsto (x, x)$$

$$\mathcal{D}^1_x \xrightarrow{\text{copy}} \begin{cases} \mathcal{D}^2_{(0,0)} & \text{if } x = 0 \\ \mathcal{D}^2_{(0,1),(1,0)} & \text{if } x = 1 \end{cases}$$

Valid Transitions

1. $(0) \xrightarrow{\text{copy}} (0, 0)$
2. $(1) \xrightarrow{\text{copy}} (0, 1)$
3. $(1) \xrightarrow{\text{copy}} (1, 0)$

MILP Model

- Given division trail $(x) \xrightarrow{\text{copy}} (y, z)$
- Propagation represented by the (in)equality

$$x - y - z = 0$$
$$x, y, z \in \{0, 1\}$$

Division Property

Modeling Propagation Rules: xor

Based on eprint's 2016/392, 2016/811, and 2016/1101

Propagation Rule

$$\mathrm{xor} : (x, y) \mapsto x + y$$

$$\mathcal{D}^2_{(k_0,k_1)} \xrightarrow{\mathrm{xor}} \mathcal{D}^1_{k_0+k_1}$$

Valid Transitions

1. $(0, 0) \xrightarrow{\mathrm{xor}} (0)$
2. $(1, 0) \xrightarrow{\mathrm{xor}} (1)$
3. $(0, 1) \xrightarrow{\mathrm{xor}} (1)$

Division Property

Modeling Propagation Rules: xor

Based on eprint's 2016/392, 2016/811, and 2016/1101

Propagation Rule

$$\mathrm{xor} : (x, y) \mapsto x + y$$

$$\mathcal{D}^2_{(k_0,k_1)} \xrightarrow{\mathrm{xor}} \mathcal{D}^1_{k_0+k_1}$$

Valid Transitions

1. $(0, 0) \xrightarrow{\mathrm{xor}} (0)$
2. $(1, 0) \xrightarrow{\mathrm{xor}} (1)$
3. $(0, 1) \xrightarrow{\mathrm{xor}} (1)$

MILP Model

- Given division trail $(x, y) \xrightarrow{\mathrm{xor}} (z)$
- Propagation represented by the (in)equality:

$$x + y - z = 0$$

$$x, y, z \in \{0, 1\}$$

Based on approach by Sun *et al.* [Sun+14] for differential case

Propagation Rule

S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$:
see [Xia+16, Algorithm 2],
computes for all $u \in \mathbb{F}_2^n$

$$\mathcal{D}_u^n \xrightarrow{S} \mathcal{D}_V^n$$

Valid Transitions

1. $u \xrightarrow{S} v_1$

⋮ $\dots$

k. $u \xrightarrow{S} v_k$

for $v_i \in V$

Division Property

Modeling Propagation Rules: S-box

Based on approach by Sun *et al.* [Sun+14] for differential case

Propagation Rule

S-box $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$:
see [Xia+16, Algorithm 2],
computes for all $u \in \mathbb{F}_2^n$

$$\mathcal{D}_u^n \xrightarrow{S} \mathcal{D}_V^n$$

Valid Transitions

1 $\quad u \xrightarrow{S} v_1$

$\vdots \quad \dots \quad$ for $v_i \in V$

k $\quad u \xrightarrow{S} v_k$

MILP Model

- Interpret set of all valid $(u, v) \in \mathbb{F}_2^{2n}$ as polyhedron
- Get inequalities from its H-representation
- Choose inequalities for model by
  - Greedy Approach [Sun+14]
  - MILP Approach [ST17] (seems to be slower)

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Division Property

Objective, Start, Stop

RUB

What are we looking for?

- Unit vectors in output division property correspond to unbalanced bits.
- We have to exclude these from our MILP model.
- When minimising the sum over the output variables, we find these unit vectors first.

Objective

$$\text{minimise} \quad x_0^r + x_1^r + \cdots + x_n^r$$

Division Property

Objective, Start, Stop

Possible Starting DPs

- Similar to subspace trail approach, we need to reduce the starting DPs needed to be checked.
- [SWW17, Proposition 2] showed that given a first initial DP k_0, for any initial DP k_1 which is element-wise smaller than k_0 the following holds:
  If DP starting in k_0 does not have a DP after r rounds, the same holds for DP starting in k_1.
- This reduces the initial DPs we have to check to n for an n-bit cipher.

Division Property

Objective, Start, Stop

RUB

Possible Starting DPs

- Similar to subspace trail approach, we need to reduce the starting DPs needed to be checked.
- [SWW17, Proposition 2] showed that given a first initial DP k_0, for any initial DP k_1 which is element-wise smaller than k_0 the following holds:
  If DP starting in k_0 does not have a DP after r rounds, the same holds for DP starting in k_1.
- This reduces the initial DPs we have to check to n for an n-bit cipher.

Initial DPs

All $k \in \mathbb{F}_2^n$ with hamming weight $n-1$ are possible initial DPs

Division Property

Objective, Start, Stop

Model Stopping Rule

Input: A Division Property MILP model $\mathcal{M}$
Output: A distinguisher exists or not

function $\mathrm{DP\ Distinguisher\ Search}(\mathcal{M})$
    while $\mathcal{M}$ has feasible solution **do**
        Solve $\mathcal{M}$

Stopping Rule

Division Property

Objective, Start, Stop

Model Stopping Rule

Input: A Division Property MILP model $\mathcal{M}$
Output: A distinguisher exists or not

```
function DP Distinguisher Search(M)
    while M has feasible solution do
        Solve M
        if objective value = 1 then
            Let solution = e_i
            Add constraint x_i^r = 0 to M
```

Stopping Rule

- Unit vectors in output division property correspond to unbalanced bits.
- We have to exclude these from our MILP model.

Division Property

Objective, Start, Stop

RUB

Model Stopping Rule

Input: A Division Property MILP model $\mathcal{M}$
Output: A distinguisher exists or not

```
function DP Distinguisher Search(M)
    while M has feasible solution do
        Solve M
        if objective value = 1 then
            Let solution = e_i
            Add constraint x_i^r = 0 to M
        else
            return Found distinguisher
    return No distinguisher exists
```

Stopping Rule

- Unit vectors in output division property correspond to unbalanced bits.
- We have to exclude these from our MILP model.
- If no more unit vectors where found, but MILP still has feasible solution, a distinguisher exists.

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Division Property

MILP model

RUB

Mixed Integer Linear Programs

Typical description of a MILP

Objective	max/min	$c^\top x$
linear inequalities	subject to	$Ax \leqslant b$

- A, b, c known coefficients
- x unknown variables ($\mathbb{R}$, $\mathbb{Z}$, or $\{0, 1\}$)

Applying MILPs to find Division Properties

Goal: Model Division Property as a MILP

We need:

- Objective function
- Starting DP
- Propagation Rules
- Stopping Rule

Similar approach

Using MILPs to find single differential trails and to estimate differentials basically same approach

We can now model the DP search for Clyde.

Division Property

Results

Division Property distinguisher for Clyde

- 8 Rounds

Conclusion

Conclusion

Thanks for your attention!

Future Work/Cryptanalysis

- Cryptagraph [HV18]
- Post cryptanalysis results on mailinglist?
- Eprint Write-Up?

pfasante.github.io/talk/spook_cryptanalysis

References I

[Lea+11] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. “A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack”. In: *CRYPTO 2011*. 2011. doi: 10.1007/978-3-642-22792-9_12.

[Sun+14] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. “Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers”. In: *ASIACRYPT 2014, Part I*. 2014. doi: 10.1007/978-3-662-45611-8_9.

[LMR15] G. Leander, B. Minaud, and S. Rønjom. “A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro”. In: *EUROCRYPT 2015, Part I*. 2015. doi: 10.1007/978-3-662-46800-5_11.

[Tod15a] Y. Todo. “Integral Cryptanalysis on Full MISTY1”. In: *CRYPTO 2015*. 2015. doi: 10.1007/978-3-662-47989-6_20. iacr: 2015/682.

[Tod15b] Y. Todo. “Structural Evaluation by Generalized Integral Property”. In: *EUROCRYPT 2015, Part I*. 2015. doi: 10.1007/978-3-662-46800-5_12. iacr: 2015/090.

[BC16] C. Boura and A. Canteaut. “Another View of the Division Property”. In: *CRYPTO 2016*. 2016. doi: 10.1007/978-3-662-53018-4_24. iacr: 2016/554.

[Guo+16] J. Guo, J. Jean, I. Nikolic, K. Qiao, Y. Sasaki, and S. M. Sim. “Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs”. In: *ToSC* 2016.1 (2016). doi: 10.13154/tosc.v2016.i1.33-56.

[TLS16] Y. Todo, G. Leander, and Y. Sasaki. “Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64”. In: *ASIACRYPT 2016, Part II*. 2016. doi: 10.1007/978-3-662-53890-6_1.

References II

[TM16] Y. Todo and M. Morii. "Bit-Based Division Property and Application to Simon Family". In: *FSE 2016*. 2016. doi: 10.1007/978-3-662-52993-5_18. iacr: 2016/285.

[Xia+16] Z. Xiang, W. Zhang, Z. Bao, and D. Lin. "Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers". In: *ASIACRYPT 2016, Part I*. 2016. doi: 10.1007/978-3-662-53887-6_24. iacr: 2016/857.

[Bei+17] C. Beierle, A. Canteaut, G. Leander, and Y. Rotella. "Proving Resistance Against Invariant Attacks: How to Choose the Round Constants". In: *CRYPTO 2017, Part II*. 2017. doi: 10.1007/978-3-319-63715-0_22. iacr: 2017/463.

[ST17] Y. Sasaki and Y. Todo. "New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search". In: *SecITC'17*. 2017. doi: 10.1007/978-3-319-69284-5_11.

[SWW17] L. Sun, W. Wang, and M. Wang. "Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property". In: *ASIACRYPT 2017, Part I*. 2017. doi: 10.1007/978-3-319-70694-8_5.

[Tod+17] Y. Todo, T. Isobe, Y. Hao, and W. Meier. "Cube Attacks on Non-Blackbox Polynomials Based on Division Property". In: *CRYPTO 2017, Part III*. 2017. doi: 10.1007/978-3-319-63697-9_9. iacr: 2018/306.

[HV18] M. Hall-Andersen and P. S. Vejre. "Generating Graphs Packed with Paths". In: *ToSC* 2018.3 (2018). doi: 10.13154/tosc.v2018.i3.265-289.

[LTW18] G. Leander, C. Tezcan, and F. Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: *ToSC* 2018.1 (2018). doi: 10.13154/tosc.v2018.i1.74-100.

[Wan+18] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, and W. Meier. "Improved Division Property Based Cube Attacks Exploiting Algebraic Properties of Superpoly". In: *CRYPTO 2018, Part I*. 2018. doi: 10.1007/978-3-319-96884-1_10. iacr: 2018/1063.