# SAT-aided Automatic Search of Boomerang Distinguishers for ARX Ciphers

Dachao Wang[1] Baocang Wang[1] Siwei Sun[2,3]

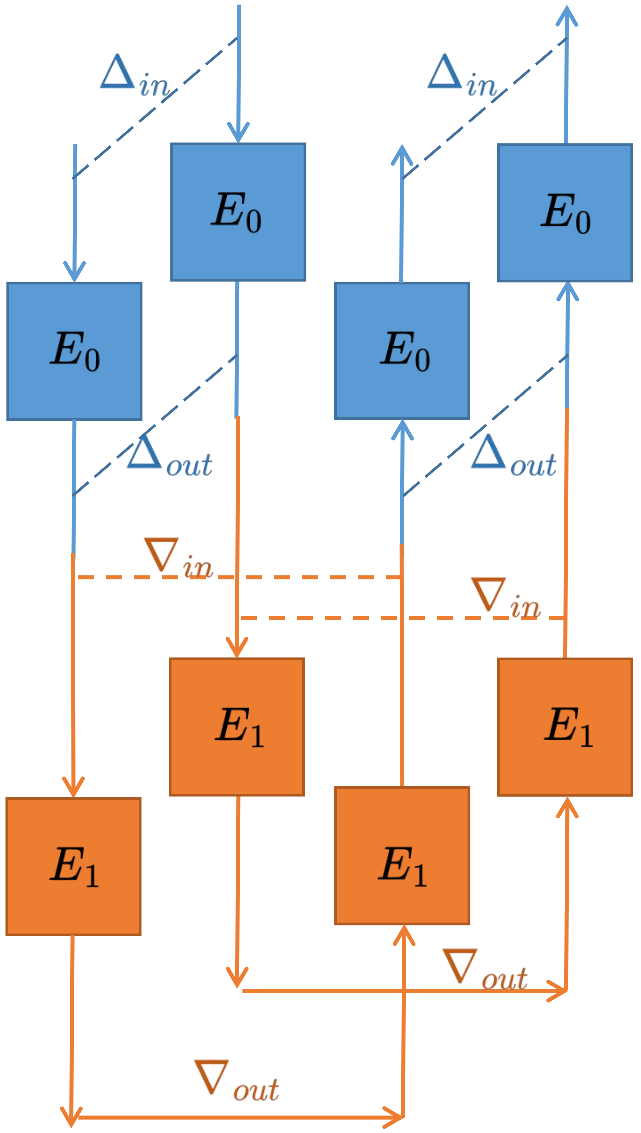
[1]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China

[2]School of Cryptology, University of Chinese Academy of Sciences, Beijing, China

[3]State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, China

FSE 2023, March 21, 2023

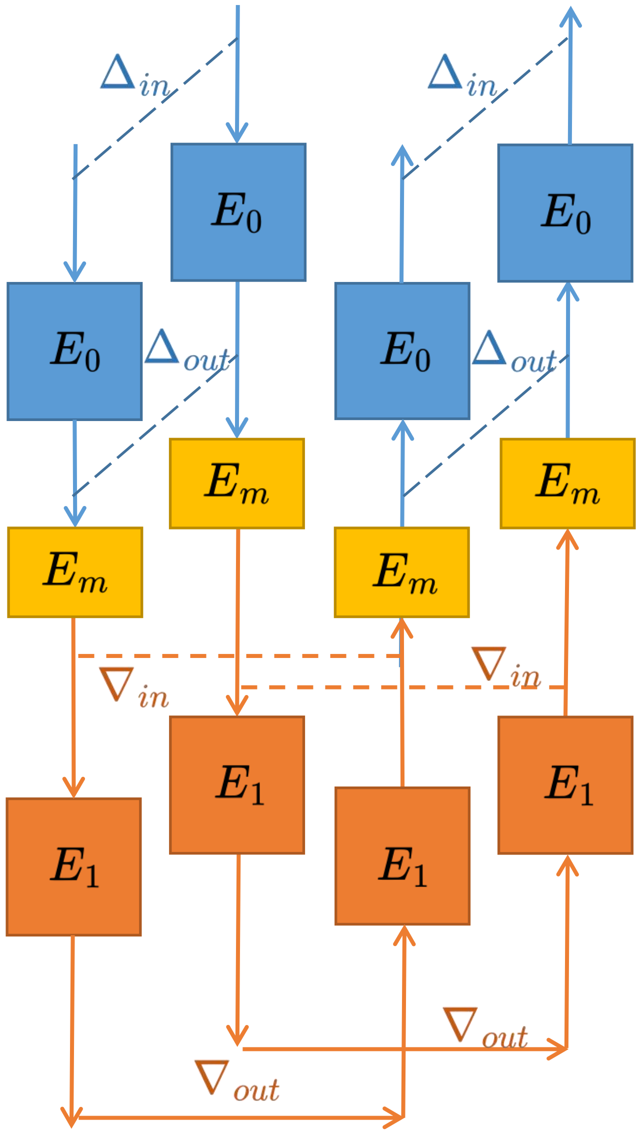
# Background: Boomerang Distinguishers




The whole cipher is seperated into two parts.

Under an independence assumption, these two differential characteristics can be connected.

# Background: Boomerang Distinguishers

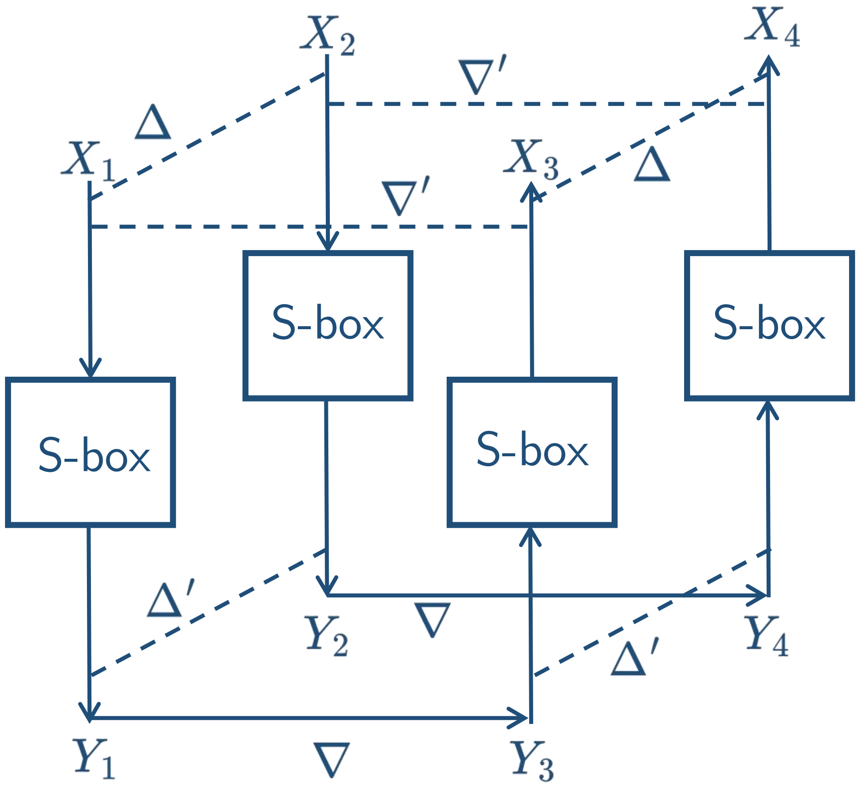



A new part is seperated out.

To analysis the connectivity of two characteristics, new cryptanalysis is put on the middle part.

Typically, cryptanalysis on $E_m$ is reduced to that on the non-linear operations.

# Background: Boomerang Tables




## Boomerang Connectivity Table (BCT)

$$\mathrm{BCT}(\Delta, \nabla) = \#\left\{x \in \mathbb{F}_2^n \mid S^{-1}(S(x) \oplus \nabla) \oplus S^{-1}(S(x \oplus \Delta) \oplus \nabla) = \Delta\right\}$$

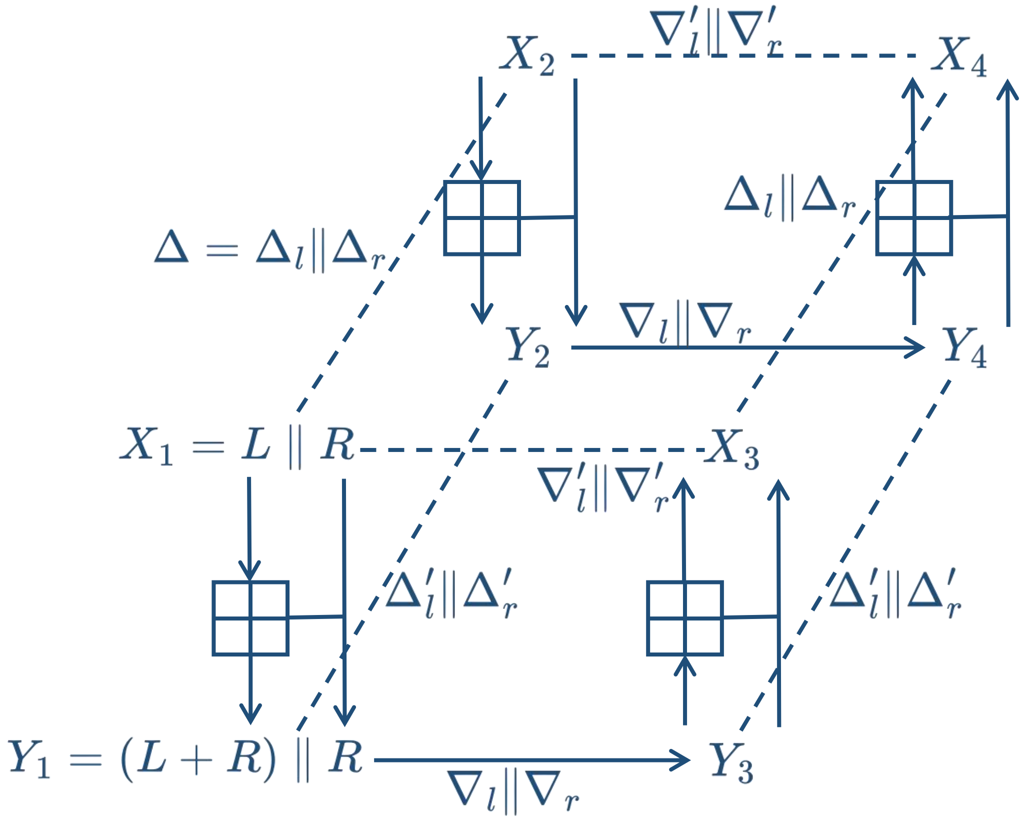
## Variants of BCT [DDV20]

$$\mathrm{UBCT}(\Delta, \Delta', \nabla) = \#\left\{x \in \mathbb{F}_2^n \mid \begin{array}{l} S(x) \oplus S(x \oplus \Delta) = \Delta' \\ S^{-1}(S(x) \oplus \nabla) \oplus S^{-1}(S(x \oplus \Delta) \oplus \nabla) = \Delta \end{array}\right\}$$

$$\mathrm{LBCT}(\Delta, \nabla', \nabla) = \#\left\{x \in \mathbb{F}_2^n \mid \begin{array}{l} S(x) \oplus S(x \oplus \nabla') = \nabla \\ S^{-1}(S(x) \oplus \nabla) \oplus S^{-1}(S(x \oplus \Delta) \oplus \nabla) = \Delta \end{array}\right\}$$

$$\mathrm{EBCT}(\Delta, \Delta', \nabla', \nabla) = \#\left\{x \in \mathbb{F}_2^n \mid \begin{array}{c} S(x) \oplus S(x \oplus \Delta) = \Delta' \\ S(x) \oplus S(x \oplus \nabla') = \nabla \\ S^{-1}(S(x) \oplus \nabla) \oplus S^{-1}(S(x \oplus \Delta) \oplus \nabla) = \Delta \end{array}\right\}$$

# Motivation



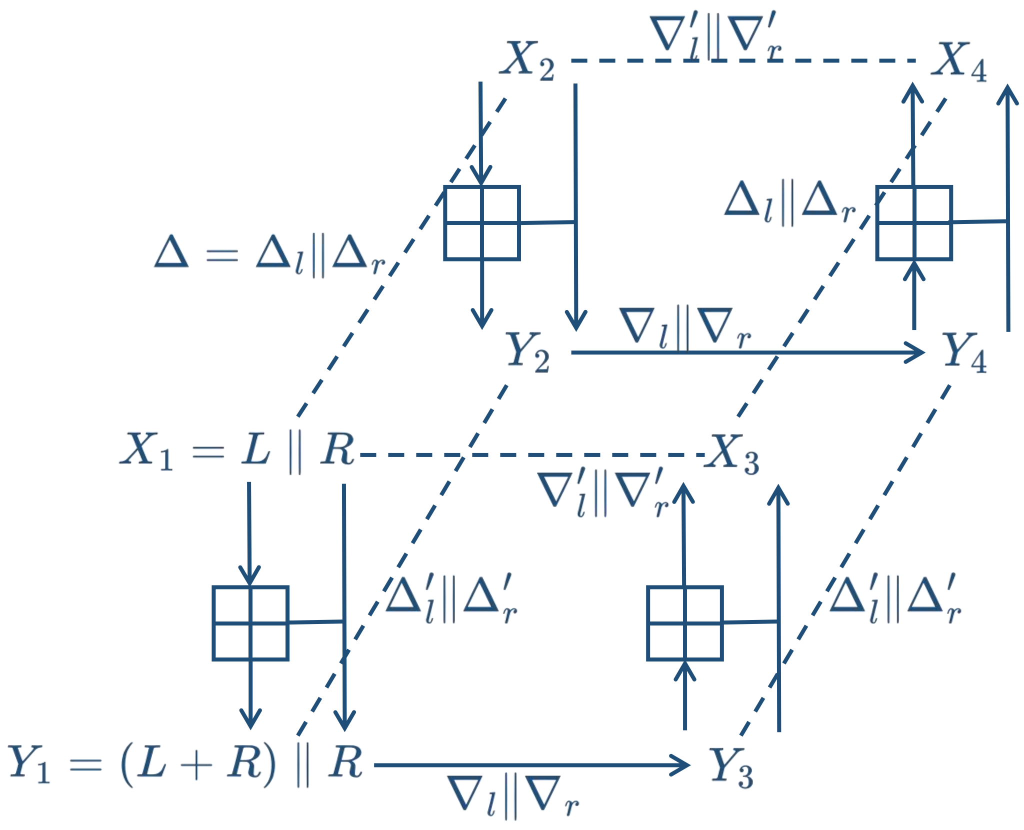

The previous result was given by Cid *et al.*.

In fact, the result can be derived from BCT on S-boxes. View the modular addition as a special S-box:

$$S(L\|R) = (L + R)\|R$$

$$\begin{aligned}&\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r)\\ =\ &\#\Big\{(L,R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \Big(\big((L \boxplus_n R) \oplus \nabla_l\big) \boxminus_n (R \oplus \nabla_r)\Big) \oplus \Big(\Big(\big((L \oplus \Delta_l) \boxplus_n (R \oplus \Delta_r)\big) \oplus \nabla_l\Big) \boxminus_n (R \oplus \Delta_r \oplus \nabla_r)\Big) = \Delta_l\Big\}\end{aligned}$$

# Motivation




The BCT is too large!

Is there a fast method to compute one entry?

Can we construct SAT/MILP models for it?

Any new results on ARX ciphers?

$$\begin{aligned}&\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r)\\ = \#&\Big\{(L,R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \Big(((L \boxplus_n R) \oplus \nabla_l) \boxminus_n (R \oplus \nabla_r)\Big) \oplus \Big(\Big(((L \oplus \Delta_l) \boxplus_n (R \oplus \Delta_r)) \oplus \nabla_l\Big) \boxminus_n (R \oplus \Delta_r \oplus \nabla_r)\Big) = \Delta_l\Big\}\end{aligned}$$

# Our Contributions

1. Design a dynamic programming algorithm to compute the BCT and its variants of the modular addition.
2. Construct SAT models for these tables.
3. Take Speck and LEA as examples, and improve the previous results.

# Compute $\mathsf{BCT_n}$

## Relations between modular addition (subtraction) and XOR

$$x \boxplus_n y = x \oplus y \oplus \mathrm{carry0}_n(x, y)$$

$$x \boxminus_n y = x \boxplus_n \bar{y} \boxplus_n 1 = x \oplus \bar{y} \oplus \mathrm{carry1}_n(x, \bar{y})$$

## Function $c = \mathrm{carry0}_n(x, y)$

$$c[i] = \begin{cases} 0 & i = 0 \\ (x[i-1] \wedge y[i-1]) \oplus (x[i-1] \wedge c[i-1]) \oplus (y[i-1] \wedge c[i-1]) & i > 0 \end{cases}$$

## Function $b = \mathrm{carry1}_n(x, y)$

$$b[i] = \begin{cases} 1 & i = 0 \\ (x[i-1] \wedge y[i-1]) \oplus (x[i-1] \wedge b[i-1]) \oplus (y[i-1] \wedge b[i-1]) & i > 0 \end{cases}$$

# Compute $\mathrm{BCT_n}$

$$\begin{aligned}
&\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r)\\
=\ &\#\Big\{(L,R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \Big(((L \boxplus_n R) \oplus \nabla_l) \boxminus_n (R \oplus \nabla_r)\Big) \oplus \Big(\Big(((L \oplus \Delta_l) \boxplus_n (R \oplus \Delta_r)) \oplus \nabla_l\Big) \boxminus_n (R \oplus \Delta_r \oplus \nabla_r)\Big) = \Delta_l\Big\}
\end{aligned}$$

$+$

$$\begin{aligned}
c_1 &= \mathrm{carry0}_n(L, R)\\
b_1 &= \mathrm{carry1}_n(L \oplus R \oplus c_1 \oplus \nabla_l, R \oplus \nabla_r)\\
c_2 &= \mathrm{carry0}_n(L \oplus \Delta_l, R \oplus \Delta_r)\\
b_2 &= \mathrm{carry1}_n(L \oplus \Delta_l \oplus R \oplus \Delta_r \oplus c_2 \oplus \nabla_l, R \oplus \Delta_r \oplus \nabla_r)
\end{aligned}$$

$\Downarrow$

$$\begin{aligned}
&\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r)\\
=\ &\#\Big\{(L,R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid c_1 \oplus b_1 \oplus c_2 \oplus b_2 = 0\Big\}\\
=\ &\#\Big\{(L,R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \forall i \in [0, n-1], c_1[i] \oplus b_1[i] \oplus c_2[i] \oplus b_2[i] = 0\Big\}
\end{aligned}$$

# Compute $\mathrm{BCT_n}$

$$\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r) = \#\Big\{(L, R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \forall i \in [0, n-1], c_1[i] \oplus b_1[i] \oplus c_2[i] \oplus b_2[i] = 0\Big\}$$

Every equation $c_1[i] \oplus b_1[i] \oplus c_2[i] \oplus b_2[i] = 0$ is a restriction on $L[i-1]$ and $R[i-1]$.
What about $L[n-1]$ and $R[n-1]$ ? **FREE!**

The four bit string $c_1[i] \| b_1[i] \| c_2[i] \| b_2[i]$ can only take **8** possible values:
$S_{\mathrm{BCT}} = \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$

# Compute $\mathrm{BCT_n}$

$$\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r) = \#\Big\{(L, R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \forall i \in [0, n-1], c_1[i] \oplus b_1[i] \oplus c_2[i] \oplus b_2[i] = 0\Big\}$$

$$\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r) \begin{cases} L[n-1] = 0, R[n-1] = 0, c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = 0000 \\ L[n-1] = 0, R[n-1] = 0, c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = 0011 \\ \cdots \\ L[n-1] = 0, R[n-1] = 1, c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = 0000 \\ \cdots \\ L[n-1] = 1, R[n-1] = 1, c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = 1111 \end{cases}$$

# Compute $\mathbf{BCT_n}$

$$\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r) = \#\Big\{(L, R) \in \mathbb{F}_2^n \times \mathbb{F}_2^n \mid \forall i \in [0, n-1], c_1[i] \oplus b_1[i] \oplus c_2[i] \oplus b_2[i] = 0\Big\}$$

$$= 4 \times \sum_{v \in S_{\mathrm{BCT}}} \# \left\{ \begin{array}{l} (L^{n-1}, R^{n-1}) \\ \in \mathbb{F}_2^{n-1} \times \mathbb{F}_2^{n-1} \end{array} \middle| \begin{array}{l} c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = v \\ \forall j \in [0, n-2], c_1[j] \oplus b_1[j] \oplus c_2[j] \oplus b_2[j] = 0 \end{array} \right\}$$

$$= 4 \times \sum_{v \in S_{\mathrm{BCT}}} f(n-1, v)$$

NEW GOAL: how to compute function $f(n-1, v)$ ?

# Compute $\mathrm{BCT_n}$

$$f(n-1, v) = \#\left\{(L^{n-1}, R^{n-1}) \in \mathbb{F}_2^{n-1} \times \mathbb{F}_2^{n-1} \;\middle|\; \begin{matrix} c_1[n-1] \| b_1[n-1] \| c_2[n-1] \| b_2[n-1] = v \\ \forall j \in [0, n-2], c_1[j] \oplus b_1[j] \oplus c_2[j] \oplus b_2[j] = 0 \end{matrix}\right\}$$

$$f(n-1, v) \begin{cases} L[n-2] = 0, R[n-2] = 0, c_1[n-2] \| b_1[n-2] \| c_2[n-2] \| b_2[n-2] = 0000 \\ L[n-2] = 0, R[n-2] = 0, c_1[n-2] \| b_1[n-2] \| c_2[n-2] \| b_2[n-2] = 0011 \\ \ldots \\ L[n-2] = 0, R[n-2] = 1, c_1[n-2] \| b_1[n-2] \| c_2[n-2] \| b_2[n-2] = 0000 \\ \ldots \\ L[n-2] = 1, R[n-2] = 1, c_1[n-2] \| b_1[n-2] \| c_2[n-2] \| b_2[n-2] = 1111 \end{cases}$$

# Compute $\mathrm{BCT_n}$

$$f(n-1,v)=\#\left\{(L^{n-1},R^{n-1})\in\mathbb{F}_2^{n-1}\times\mathbb{F}_2^{n-1}\ \middle|\ \begin{array}{l} c_1[n-1]\|b_1[n-1]\|c_2[n-1]\|b_2[n-1]=v \\ \forall j\in[0,n-2], c_1[j]\oplus b_1[j]\oplus c_2[j]\oplus b_2[j]=0 \end{array}\right\}$$

$$f(n-1,v)\left\{\begin{array}{l}
L[n-2]=0, R[n-2]=0, c_1[n-2]\|b_1[n-2]\|c_2[n-2]\|b_2[n-2]=0000 \\
L[n-2]=0, R[n-2]=0, c_1[n-2]\|b_1[n-2]\|c_2[n-2]\|b_2[n-2]=0011 \\
\cdots \\
L[n-2]=0, R[n-2]=1, c_1[n-2]\|b_1[n-2]\|c_2[n-2]\|b_2[n-2]=0000 \\
\cdots \\
L[n-2]=1, R[n-2]=1, c_1[n-2]\|b_1[n-2]\|c_2[n-2]\|b_2[n-2]=1111
\end{array}\right.$$

**$g(u,v,L[i],R[i],\Delta_l[i],\Delta_r[i],\nabla_l[i],\nabla_r[i])$**

Denote whether the chosen values of $L[n-2]$ and $R[n-2]$ are valid.

# Compute $\mathrm{BCT}_n$

$$f(n-1,v) = \#\left\{(L^{n-1},R^{n-1}) \in \mathbb{F}_2^{n-1} \times \mathbb{F}_2^{n-1} \mid \begin{array}{l} c_1[n-1]\|b_1[n-1]\|c_2[n-1]\|b_2[n-1] = v \\ \forall j \in [0,n-2], c_1[j] \oplus b_1[j] \oplus c_2[j] \oplus b_2[j] = 0 \end{array}\right\}$$

$$= \sum_{u \in S_{\mathrm{BCT}}} \sum_{L[n-2],R[n-2] \in \mathbb{F}_2} g(u,v,L[n-1],R[n-2],\Delta_l[n-2],\Delta_r[n-2],\nabla_l[n-2],\nabla_r[n-2])f(n-2,u)$$

$$= \sum_{u \in S_{\mathrm{BCT}}} f(n-2,u) \underline{\sum_{L[n-2],R[n-2] \in \mathbb{F}_2} g(u,v,L[n-2],R[n-2],\Delta_r[n-2],\nabla_l[n-2],\nabla_r[n-2])}$$

$$= \sum_{u \in S_{\mathrm{BCT}}} \underline{T(u,v,\Delta_l[n-2],\Delta_r[n-2],\nabla_l[n-2],\nabla_r[n-2])}f(n-2,u)$$

Recursive definition again!

# Compute $\mathrm{BCT_n}$

**Theorem 2**

Let $S'_{\mathrm{BCT}} = \{0000, 0011, 0101, 0110\}$. For $u, v \in S'_{\mathrm{BCT}}$, denote $f'(i,v) = f(i,v) + f(i,\bar{v})$ and

$$\begin{aligned} & T'(u,v,\Delta_l[i],\Delta_r[i],\nabla_l[i],\nabla_r[i]) \\ = \; & T(u,v,\Delta_l[i],\Delta_r[i],\nabla_l[i],\nabla_r[i]) + T(u,v,\Delta_l[i],\Delta_r[i],\nabla_l[i],\nabla_r[i]) \end{aligned}$$

Then

$$f'(i+1,v) = \sum_{u \in S'_{\mathrm{BCT}}} T'(u,v,\Delta_l[i],\Delta_r[i],\nabla_l[i],\nabla_r[i]) f'(i,u)$$

# Compute $\mathrm{BCT_n}$

**Algorithm 1** A dynamic programming algorithm to compute $f$.

```
procedure DP(n, v, Δ_l, Δ_r, ∇_l, ∇_r)                ▷ The value of f(n, v) where v ∈ S_BCT
    Initialize a hash table T_dp such that, for all u ∈ S_BCT, T_dp[u] = 0 except that
    T_dp[0101] = 1;
    for all i ∈ {0, 1, 2, . . . , n − 1} do
        Initialize a hash table T'_dp such that T'_dp[u] = 0 for all u ∈ S_BCT;
        for all u ∈ S_BCT do
            for all u' ∈ S_BCT do
                Look up the value T(u', u, Δ_l[i], Δ_r[i], ∇_l[i], ∇_r[i]). Assume that it is t;
                T'_dp[u] ← T'_dp[u] + t × T_dp[u'];
        T_dp ← T'_dp;
    return T_dp[v];
```

$$f'(n,v) = \sum_{u \in S'_{\mathrm{BCT}}} T'(u, v, \Delta_l[n-1], \Delta_r[n-1], \nabla_l[n-1], \nabla_r[n-1]) f'(n-1, u)$$

Booleanize (whether $f'(n,v)$ is non-zero or not)

$$f'_b(n,v) = \max_{u \in S'_{\mathrm{BCT}}} T'_b(u, v, \Delta_l[n-1], \Delta_r[n-1], \nabla_l[n-1], \nabla_r[n-1]) f'_b(n-1, u)$$

# Compute $\mathrm{BCT}_n$

**Algorithm 3** A dynamic programming algorithm to check an entry in BCT.

**procedure** IsZero$(n, \Delta_l, \Delta_r, \nabla_l, \nabla_r)$
  Initialize a hash table $T_{dp}$ such that, for all $u \in S_{\mathrm{BCT}}$, $T_{dp}[u] = False$ except that $T_{dp}[0101] = True$;
  **for all** $i \in \{0, 1, 2, \ldots, n-1\}$ **do**
    Initialize a hash table $T'_{dp}$ such that $T'_{dp}[u] = False$ for all $u \in S_{\mathrm{BCT}}$;
    **for all** $u \in S'_{\mathrm{BCT}}$ **do**
      **for all** $u' \in S'_{\mathrm{BCT}}$ **do**
        Look up the value $T'_b(u', u, \Delta_l[i], \Delta_r[i], \nabla_l[i], \nabla_r[i])$. Assume that it is $t$;
        $T'_{dp}[u] \leftarrow T'_{dp}[u] \vee (t \wedge T_{dp}[u'])$;
    $T_{dp} \leftarrow T'_{dp}$;
  $sum \leftarrow False$;
  **for all** $u \in S'_{\mathrm{BCT}}$ **do**
    $sum \leftarrow sum \vee T_{dp}[u]$;
  **return** $sum$; ▷ *$sum$ is True if the entry is non-zero and False if it is zero.*

check whether the BCT entry is non-zero or not.

$$f'_b(n, v) = \max_{u \in S'_{\mathrm{BCT}}} T'_b(u, v, \Delta_l[n-1], \Delta_r[n-1], \nabla_l[n-1], \nabla_r[n-1]) f'_b(n-1, u)$$

# Modelling: Partial BCT

If, for all $u \in S'_{\mathrm{BCT}}$, we have $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$, then

$$\sum_{v \in S'_{\mathrm{BCT}}} f'(i, v) = 4 \times \sum_{u \in S'_{\mathrm{BCT}}} f'(i-1, u)$$

**Note that:** $\mathrm{BCT}_n(\Delta_l, \Delta_r, \nabla_l, \nabla_r) = 4 \times \sum_{v \in S'_{\mathrm{BCT}}} f'(n-1, v)$.

# Modelling: Partial BCT

If, for all $u \in S'_{\mathrm{BCT}}$, we have $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$, then

$$\sum_{v \in S'_{\mathrm{BCT}}} f'(i, v) = 4 \times \sum_{u \in S'_{\mathrm{BCT}}} f'(i-1, u)$$

## A Heuristic Trick (Partial BCT)

If $k$ out of $n-1$ these equations are required to be satisfied, the resulting BCT entry would be $4^{k+1}q$. That is, just express $k$ restrictions

"for all $u \in S'_{\mathrm{BCT}}$, $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$"

as boolean expressions.

# Modelling: Partial BCT

## A Heuristic Trick (Partial BCT)

If $k$ out of $n-1$ these equations are required to be satisfied, the resulting BCT entry would be $4^{k+1}q$. That is, just express $k$ restrictions

"for all $u \in S'_{\mathrm{BCT}}$, $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$"

as boolean expressions.

| | | | |
|---|---|---|---|
| 8 | 8 | 8 | 8 |
| 8 | 0 | 2 | 6 |
| 8 | 4 | 2 | 4 |
| 8 | 2 | 6 | 2 |

expect: $4^{k+1}q$ with $q > 1$

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

# Modelling: Partial BCT

**A Heuristic Trick (Partial BCT)**

If $k$ out of $n-1$ these equations are required to be satisfied, the resulting BCT entry would be $4^{k+1}q$. That is, just express $k$ restrictions

"for all $u \in S'_{\mathrm{BCT}}$, $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$"

as boolean expressions.

Fact:

1. Currently, we lack enough knowledge about $q$.
2. The resulting partial BCT still contains values with $q < 1$.
3. Our experiments showed that the models always return BCT entries with values around $4^{k+1}$.

# Modelling: The Complete Model

**Algorithm 3** A dynamic programming algorithm to check an entry in BCT.

**procedure** ISZERO$(n, \Delta_l, \Delta_r, \nabla_l, \nabla_r)$
  Initialize a hash table $T_{dp}$ such that, for all $u \in S_{\mathrm{BCT}}$, $T_{dp}[u] = False$ except that $T_{dp}[0101] = True$;
  **for all** $i \in \{0, 1, 2, \ldots, n-1\}$ **do**
    Initialize a hash table $T'_{dp}$ such that $T'_{dp}[u] = False$ for all $u \in S_{\mathrm{BCT}}$;
    **for all** $u \in S'_{\mathrm{BCT}}$ **do**
      **for all** $u' \in S'_{\mathrm{BCT}}$ **do**
        Look up the value $T'_b(u', u, \Delta_l[i], \Delta_r[i], \nabla_l[i], \nabla_r[i])$. Assume that it is $t$;
        $T'_{dp}[u] \leftarrow T'_{dp}[u] \vee (t \wedge T_{dp}[u'])$;
    $T_{dp} \leftarrow T'_{dp}$;
  $sum \leftarrow False$;
  **for all** $u \in S'_{\mathrm{BCT}}$ **do**
    $sum \leftarrow sum \vee T_{dp}[u]$;
  **return** $sum$; ▷ *sum is True if the entry is non-zero and False if it is zero.*

(1) $\forall u \in S'_{\mathrm{BCT}}, T'_{dp}[u] = \bigvee_{u' \in S'_{\mathrm{BCT}}} \left(T'_b(u', u, \Delta_l[i], \Delta_r[i], \nabla_l[i], \nabla_r[i]) \wedge T_{dp}[u']\right)$

(2) $f'_b(n-2, 0000) \vee f'_b(n-2, 0011) \vee f'_b(n-2, 0101) \vee f'_b(n-2, 0110) = True$

# Modelling: The Complete Model

## A Heuristic Trick (Partial BCT)

If $k$ out of $n-1$ these equations are required to be satisfied, the resulting BCT entry would be $4^{k+1}q$. That is, just express $k$ restrictions

“for all $u \in S'_{\mathrm{BCT}}$, $T'(u, u, \Delta_l[i-1], \Delta_r[i-1], \nabla_l[i-1], \nabla_r[i-1]) \neq 0$“ (3)

as boolean expressions.

```
    T_dp ← T'_dp;
sum ← False;
for all u ∈ S'_BCT do
    sum ← sum ∨ T_dp[u];
return sum;        ▷ sum is True if the entry is non-zero and False if it is zero.
```

$$(1)\ \forall u \in S'_{\mathrm{BCT}}, T'_{dp}[u] = \bigvee_{u' \in S'_{\mathrm{BCT}}} \left(T'_b(u', u, \Delta_l[i], \Delta_r[i], \nabla_l[i], \nabla_r[i]) \wedge T_{dp}[u']\right)$$

$$(2)\ f'_b(n-2, 0000) \vee f'_b(n-2, 0011) \vee f'_b(n-2, 0101) \vee f'_b(n-2, 0110) = \mathit{True}$$

# Modelling: The Complete Model

**Framework 1**

1. Set the product of the probabilities of two differential characteristics as the objective function and maximize it.
2. Compute the probability of the switch and obtain the total probability.
3. Tweak the threshold and repeat step 1 and step 2 until a desired boomerang distinguisher is found.

It is fast but may miss better characteristics.

Suitable for long characteristics or complicated switches.

# Modelling: The Complete Model

**Framework 2**

1. Set the product of the probabilities of two differential characteristics as the objective function. Set the threshold to 0. Then, maximize it. This step returns the upper bound of the probability of boomerang distinguishers.
2. Set the threshold and the upper bound of the probability and enumerate all the charateristics. Take the largest cluster and enumerate all possible characteristics in it. Compute the probability of the cluster and return.
3. Tweak the threshold and upper bound. Repeat step 2 until a desired boomerang distinguisher is found.

It is slow but can take the advantage of clustering effect.

Suitable for short characteristics or simple switches.

# The Results

| Cipher | Rounds | $-\log_2$ Prob. EST. | $-\log_2$ Prob. EXP. | Input difference of $E_0$ (hex) | Output difference of $E_1$ (hex) |
|---|---|---|---|---|---|
| SPECK32/64 | 10 | 29.15 | 27.34 | 2800 ‖ 0010 | 8102 ‖ 8108 |
| | | 29.17 | 27.39 | 0014 ‖ 0800 | 8000 ‖ 840a |
| | | 29.78 | 28.43 | 2800 ‖ 0010 | 0040 ‖ 0542 |
| SPECK48/72 | 12 | 44.15 | - | 020082 ‖ 120200 | 0080a0 ‖ 2085a4 |
| | | 46.41 | - | 820200 ‖ 001202 | 800084 ‖ 8400a0 |
| | | 47.93 | - | 820200 ‖ 001202 | 008400 ‖ 00a084 |

Table 5: The Estimated Probabilities of the Switch in LEA

| Method | Prob. | Time* |
|---|---|---|
| [KKS20] | 0.661755 | 221 milliseconds |
| ARXtools | 0.710850 | 34 seconds |
| Our SAT-aided Method | 0.708521 | 17 hours |
| Experimental Evaluation | 0.71 | 17 hours |

# Our Technique VS ARXtools

1. The dynamic programming algorithm and ARXtools are based on the same property of modular additions, but they are constructed from two different perspectives.

2. From our perspective, more mathmatical properties behind the modular addition are revealed.

3. When estimating probabilities, ARXtools is faster and more precise.

4. Our technique is capable of searching characteristics, while ARXtools cannot.

# Conclusion

## Results:

1. A dynamic programming algorithm to compute the BCT entries of the modular addition.
2. SAT models for partial BCT, LBCT, UBCT, and EBCT.
3. New results on Speck and LEA.

## Limitations:

1. The current computations of BCT and its variants are not convenient to be modelled.
2. The models for switches are large.
3. Optimal distinguishers are still unknown.
4. Comparing with ARXtools, our computation of probabilities are slow.

All the codes are publicly available at https://github.com/0NG/boomerang_search

**Thank you very much for your attention!**