Cryptanalysis of ARX-based White-box Implementations

Alex Biryukov Baptiste Lambin Aleksei Udovenko

CHES 2023, September 14th

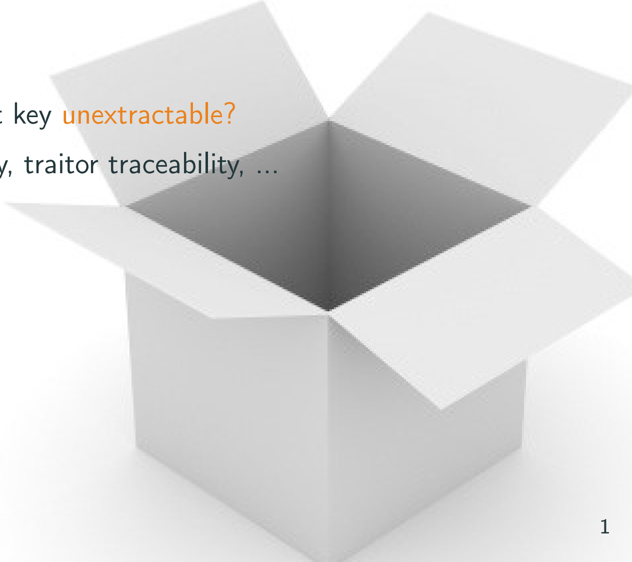
DCS and SnT, University of Luxembourg

Luxembourg's FNR and Germany's DFG joint project APLICA (C19/IS/13641232)

Plan

White-box cryptography

- Implementation fully available, secret key unextractable?
- **Extra**: one-wayness, incompressibility, traitor traceability, ...



White-box cryptography

- Implementation fully available, secret key unextractable?
- **Extra**: one-wayness, incompressibility, traitor traceability, ...

- The most **challenging**:
  existing symmetric primitives, e.g. the AES, Speck

Implicit computations (Ranea, Vandersmissen, and Preneel 2022)

Let $y = F(x)$

- usual method: write down *polynomials* or a *circuit* for F:

$$y_1 = F_1(x)$$
$$y_2 = F_2(x)$$
$$\dots$$

Implicit computations (Ranea, Vandersmissen, and Preneel 2022)

Let $y = F(x)$

- usual method: write down *polynomials* or a *circuit* for F:

$$\begin{aligned} y_1 &= F_1(x) \\ y_2 &= F_2(x) \\ &\dots \end{aligned}$$

- implicit function: write down polynomials relating x, y:

$$\begin{aligned} P_1(x, y) &= 0 \\ P_2(x, y) &= 0 \\ &\dots \end{aligned}$$

Implicit computations (Ranea, Vandersmissen, and Preneel 2022)

Let $y = F(x)$

- usual method: write down *polynomials* or a *circuit* for F:

$$y_1 = F_1(x)$$
$$y_2 = F_2(x)$$
$$\dots$$

- implicit function: write down polynomials relating x, y:

$$P_1(x, y) = 0$$
$$P_2(x, y) = 0$$
$$\dots$$

- how to compute y from x?
  1. require that $P(x, y)$ is linear in y
  2. plug in value for $x = \overline{x}$
  3. solve linear system $P(\overline{x}, y) = 0$ for y

Implicit ARX (Ranea, Vandersmissen, and Preneel 2022)

Modular addition

- let $\boxplus$ denote word addition (modulo 2^n)
- i-th output bit (from LSB) has degree i
- $\Rightarrow \boxplus$ has degree $n-1$
- however, there exists bilinear P such that $P(x, y, x \boxplus y) = 0$

Implicit ARX (Ranea, Vandersmissen, and Preneel 2022)

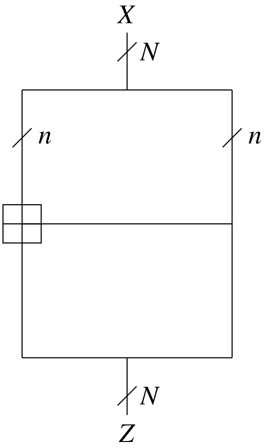
Modular addition

- let $\boxplus$ denote word addition (modulo 2^n)
- i-th output bit (from LSB) has degree i
- $\Rightarrow \boxplus$ has degree $n-1$
- however, there exists bilinear P such that $P(x, y, x \boxplus y) = 0$

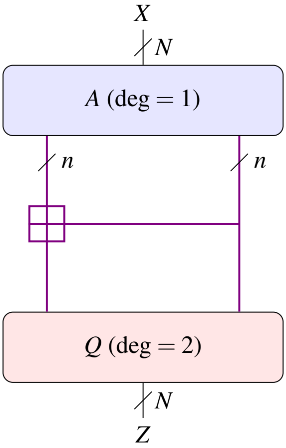
Why implicit?

- let I, O be input/output encodings, I low-degree, O linear
- polynomial $P(I(x), O(y))$ can be written compactly (representing $O \circ F \circ I$)
- obfuscate using graph automorphisms

Self-equivalences (Vandersmissen, Ranea, and Preneel 2022; Ranea, Vandersmissen, and Preneel 2022)

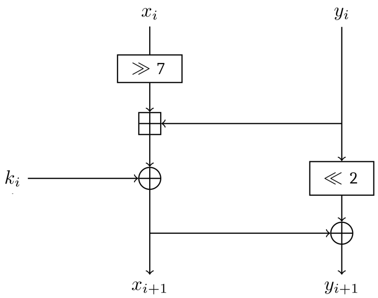


$=$

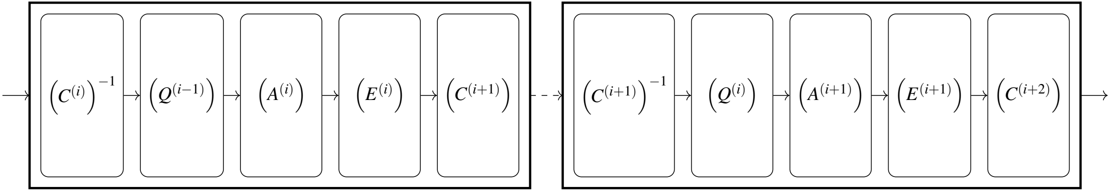


Block-cipher family Speck

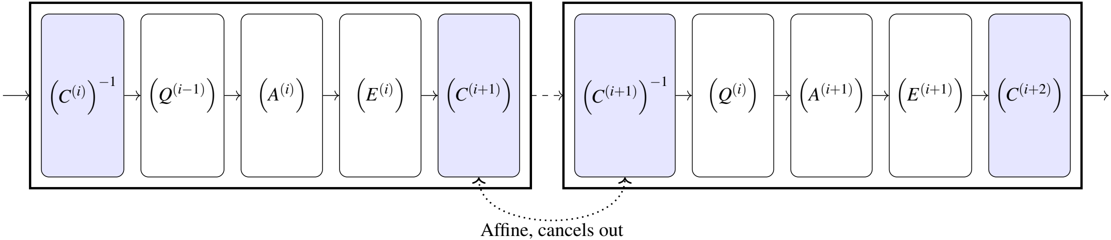
- Designed by NSA (2014)
- Simple ARX structure (1 round $\overset{\text{aff}}{\simeq} (x \boxplus y, y)$)
- Block size: **32**, 48, 64, ... (2 words)
- Key size: **64**, 72, 96, ... (2-4 words)



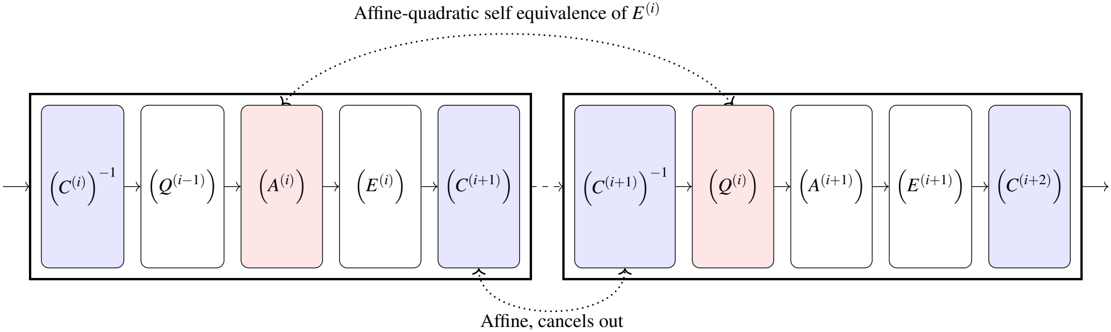
Speck-based implicit ARX white-box



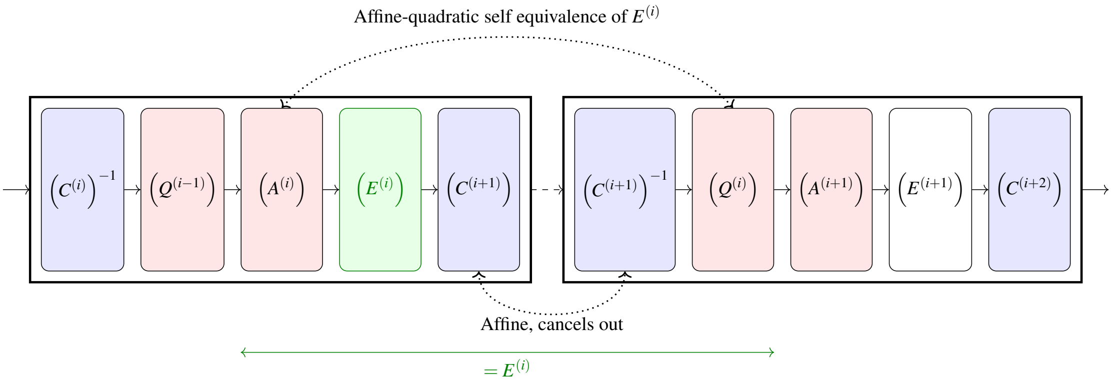
Speck-based implicit ARX white-box



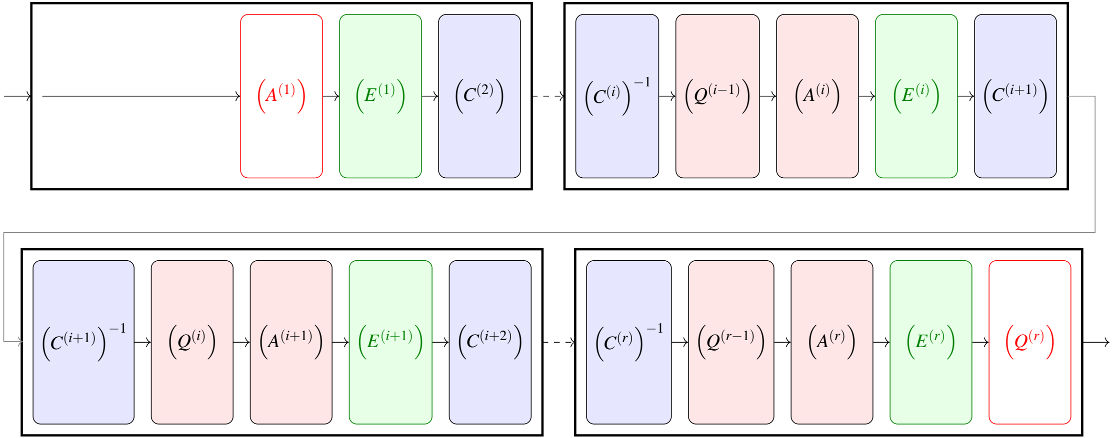
Speck-based implicit ARX white-box



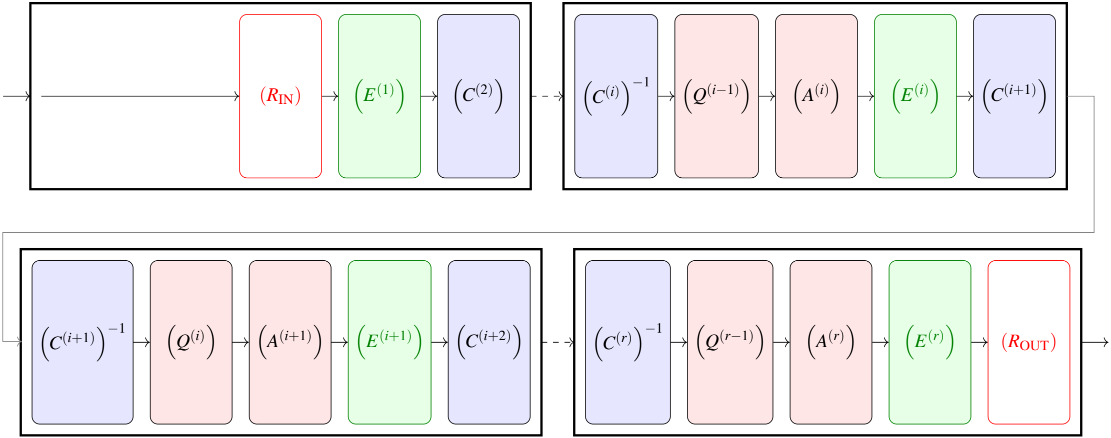
Speck-based implicit ARX white-box



External encodings: none



External encodings: random



Plan

Our attack summary

Attack	Target	Time	Data	Comment
Algebraic key recovery	**A**	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	No external encoding (any side)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	

Our attack summary

Attack	Target	Time	Data	Comment
Algebraic key recovery	**A**	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	No external encoding (any side)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	
Round oracle optimization	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Computes bilinear implicit function
	Q	$\mathcal{O}(n^9)$	$\mathcal{O}(n^3)$	Computes quadratic-affine implicit function

Our attack summary

Attack	Target	Time	Data	Comment
Algebraic key recovery	**A**	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	No external encoding (any side)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	
Round oracle optimization	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Computes bilinear implicit function
	Q	$\mathcal{O}(n^9)$	$\mathcal{O}(n^3)$	Computes quadratic-affine implicit function
Round oracle inversion	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Requires existence of bilinear implicit function
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	*Heuristic, several bits have to be guessed*

Our attack summary

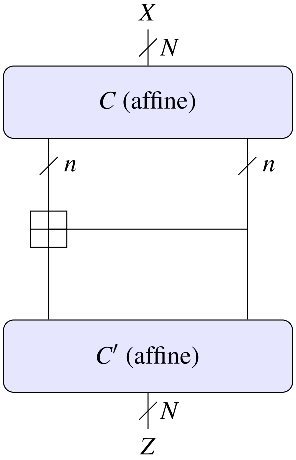
Attack	Target	Time	Data	Comment
Algebraic key recovery	**A**	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	No external encoding (any side)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	
Round oracle optimization	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Computes bilinear implicit function
	Q	$\mathcal{O}(n^9)$	$\mathcal{O}(n^3)$	Computes quadratic-affine implicit function
Round oracle inversion	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Requires existence of bilinear implicit function
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	*Heuristic, several bits have to be guessed*
Round decomposition	**A**	$\mathcal{O}(n^4)$	$\mathcal{O}(n^3)$	Total query time is at least $\mathcal{O}(n^6)$ (dominating)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Recovers quadratic encoding.

Our attack summary

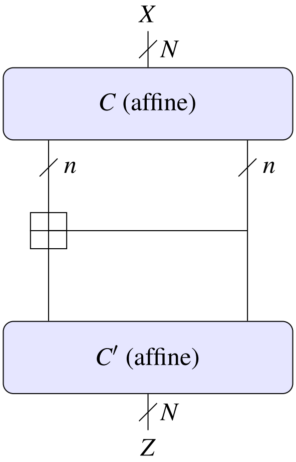
Attack	Target	Time	Data	Comment
Algebraic key recovery	**A**	$\mathcal{O}(n^3)$	$\mathcal{O}(n)$	No external encoding (any side)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	
Round oracle optimization	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Computes bilinear implicit function
	Q	$\mathcal{O}(n^9)$	$\mathcal{O}(n^3)$	Computes quadratic-affine implicit function
Round oracle inversion	**A**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Requires existence of bilinear implicit function
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	*Heuristic, several bits have to be guessed*
Round decomposition	**A**	$\mathcal{O}(n^4)$	$\mathcal{O}(n^3)$	Total query time is at least $\mathcal{O}(n^6)$ (dominating)
	Q	$\mathcal{O}(n^6)$	$\mathcal{O}(n^2)$	Recovers quadratic encoding.
Decomposition and key recovery	**A,Q**	$\mathcal{O}(n^6)$	$\mathcal{O}(n^3)$	Requires several consecutive decomposed rounds to recover the master key.

Plan

Step 1: locate linear bits

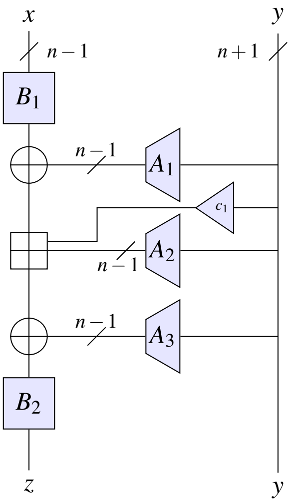


Step 1: locate linear bits

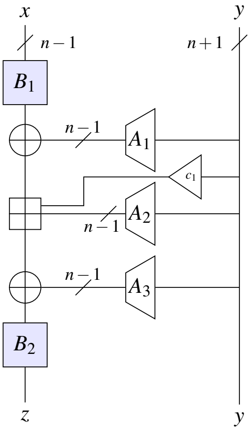


1. locate linear bits

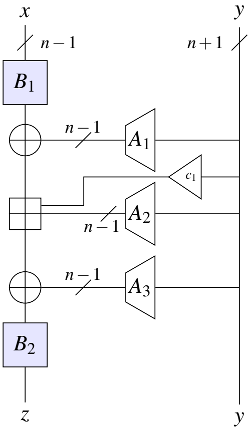
split unknown parts



Step 2: triangularize left branch encodings



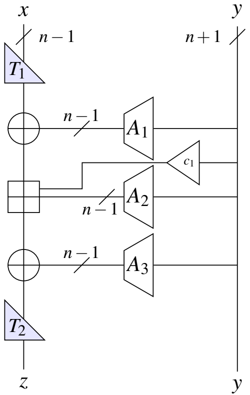
Step 2: triangularize left branch encodings



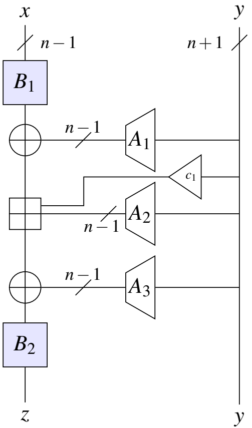
2. trinagularize B's

$\longrightarrow$

differential rank method

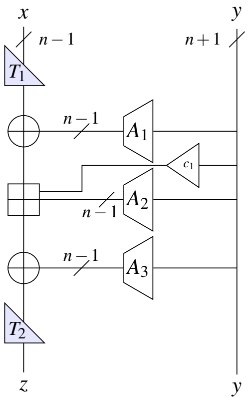


Step 2: triangularize left branch encodings

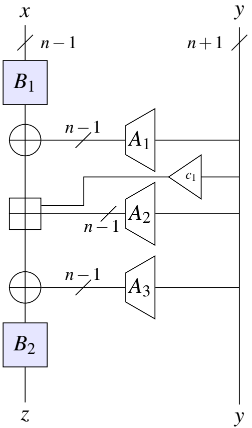


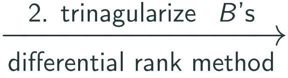
$$\xrightarrow[\text{differential rank method}]{\text{2. trinagularize } B\text{'s}}$$

- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$

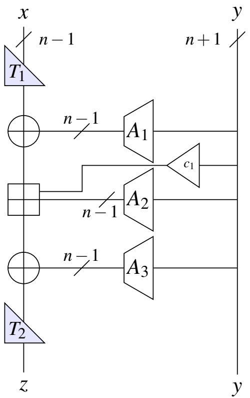


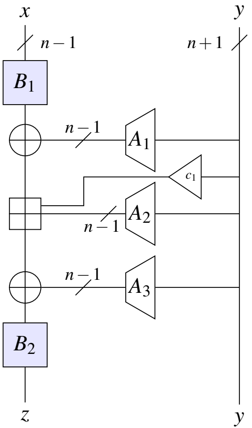
Step 2: triangularize left branch encodings





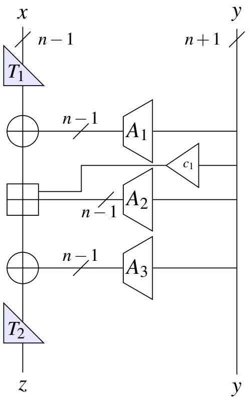
- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$
- $\Delta y = 0000000$



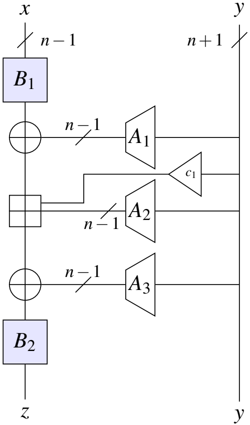


$\xrightarrow[\text{differential rank method}]{\text{2. trinagularize } B\text{'s}}$

- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$
- $\Delta y = 0000000$
- $\Delta z = ????100$

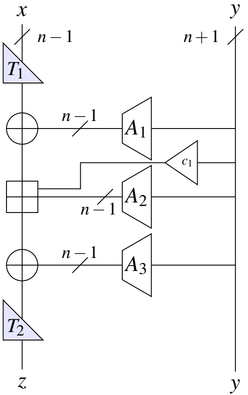


Step 2: triangularize left branch encodings

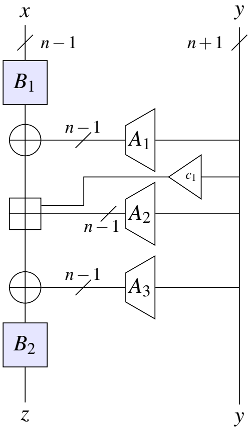


$$\xrightarrow[\text{differential rank method}]{\text{2. trinagularize } B\text{'s}}$$

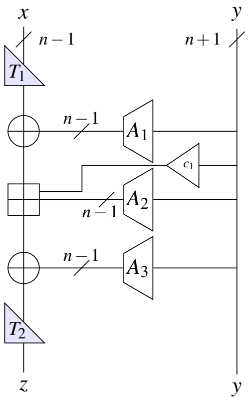
- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$
- $\Delta y = 0000000$
- $\Delta z = ????100$
- $\mathrm{rank}\,\{\Delta z \mid x \text{ rand.}\} \leq k$



Step 2: triangularize left branch encodings

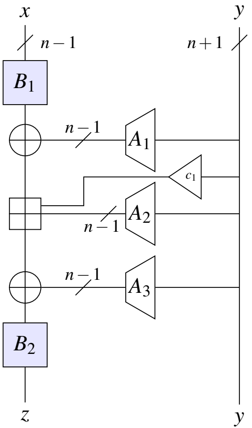


$$\xrightarrow[\text{differential rank method}]{\text{2. trinagularize } B\text{'s}}$$



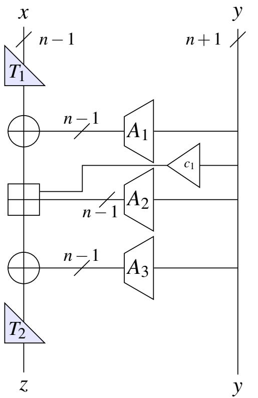
- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$
- $\Delta y = 0000000$
- $\Delta z = ????100$
- $\mathrm{rank}\,\{\Delta z \mid x \text{ rand.}\} \leq k$
- $\mathrm{rank}\,\{\Delta z \mid x \text{ rand.}\} = \max$
  $\Rightarrow$ the LSB is 1

Step 2: triangularize left branch encodings

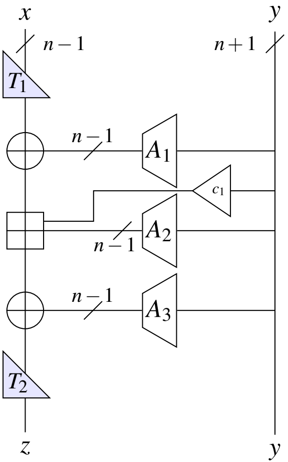


$\xrightarrow[\text{differential rank method}]{\text{2. trinagularize } B\text{'s}}$

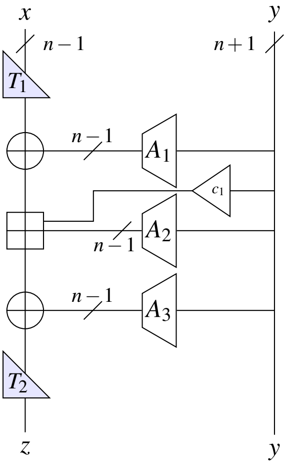
- $z = x \boxplus y$
- $\Delta x = \underbrace{????1}_{k\text{bits}}00$
- $\Delta y = 0000000$
- $\Delta z = ????100$
- $\mathrm{rank}\{\Delta z \mid x \text{ rand.}\} \leq k$
- $\mathrm{rank}\{\Delta z \mid x \text{ rand.}\} = \max$
  $\Rightarrow$ the LSB is 1
- repeat $\rightarrow$ get $B_1[0], B_2[0]$



Step 3: recover left-branch maps

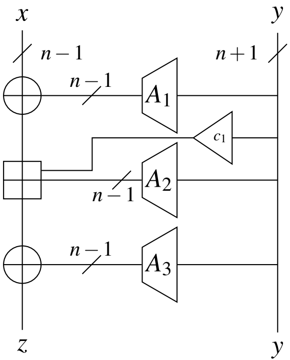


Step 3: recover left-branch maps



3. recover T_1, T_2

using differential probabilities



Step 3: recover left-branch maps

Proposition 4. *Let $z = x \boxplus y$ be an n-bit modular addition, $n \geq 3$. Set*

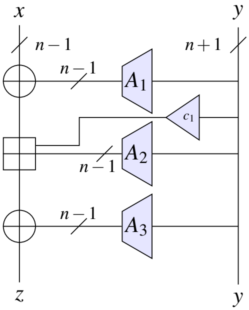
$$\Delta y = 0, \quad \Delta x_1 = e_0 = (0,0,0,\dots,0,1), \quad \Delta x_2 = e_0 \oplus e_{n-2} = (0,1,0,\dots,0,1).$$

Then, the most probable transitions with input differences $(\Delta x_1, \Delta y)$ and $(\Delta x_2, \Delta y)$ respectively are described by

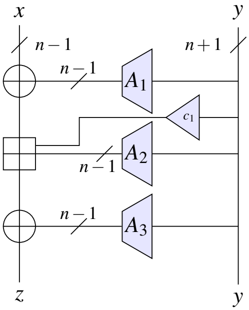
$$\Pr[(\Delta x_1, \Delta y) \xrightarrow{\boxplus} \Delta z] = \begin{cases} 1/2, & \Delta z = (0,\dots,0,0,1) = \Delta x_1, \\ 1/4, & \Delta z = (0,\dots,0,1,1), \\ \leq 1/4, & \textit{otherwise}\dots \end{cases} \tag{4}$$

$$\Pr[(\Delta x_2, \Delta y) \xrightarrow{\boxplus} \Delta z] = \begin{cases} 1/4, & \Delta z = (0,1,\dots,0,1) = \Delta x_2, \\ 1/4, & \Delta z = (1,1,\dots,0,1) = \Delta x_2, \\ \leq 1/4, & \textit{otherwise}\dots \end{cases} \tag{5}$$

Step 4: recover Feistel maps

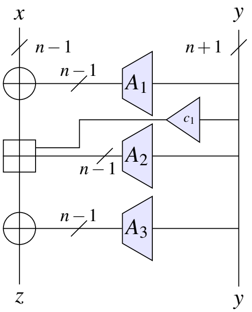


Step 4: recover Feistel maps



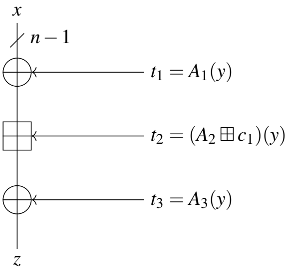
$$\xrightarrow[\text{by fixing righthand side}]{\text{4. recover } A_1, A_2 \boxplus c_1, A_3}$$

Step 4: recover Feistel maps

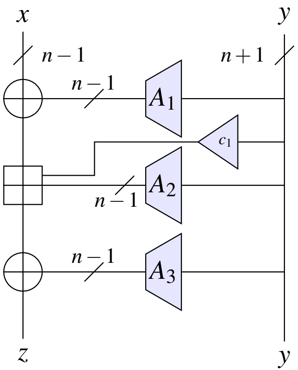


$$\xrightarrow[\text{by fixing righthand side}]{\text{4. recover } A_1, A_2 \boxplus c_1, A_3}$$

- 8 solutions per y
- choose arbitrarily
- combine for lin. indep. y's
- $\Rightarrow$ a solution
- (with some annoyances due to the carry c_1)

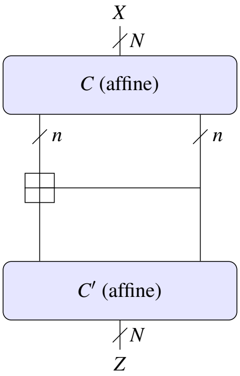


Step 5: finishing



1. recover affine equivalence of $c_1 \overset{\text{aff}}{\simeq} y_0 y_1$ (next slides)
2. move out the recovered Feistel maps A_1, A_2, A_3
3. collect all applied affine maps to get a decomposition of the original function

Step 5: finishing

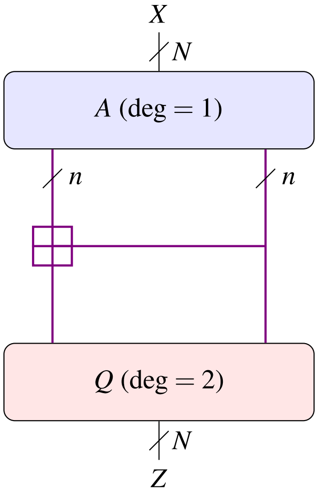


1. recover affine equivalence of $c_1 \overset{\text{aff}}{\simeq} y_0 y_1$ (next slides)
2. move out the recovered Feistel maps A_1, A_2, A_3
3. collect all applied affine maps to get a decomposition of the original function

Plan

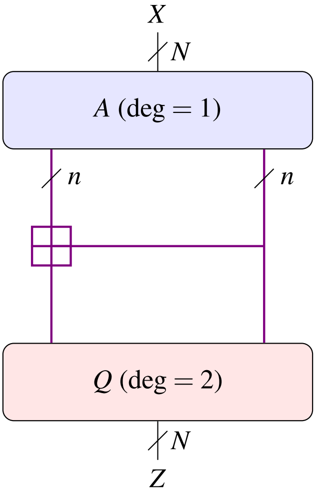
Sparsity observations

- let $S : (x, y) \mapsto (x \boxplus y, y)$
- let A, Q be affine-quadratic self-equiv. of S:
  $S = Q \circ S \circ A$
- theorem: half of outputs of Q has to be linear



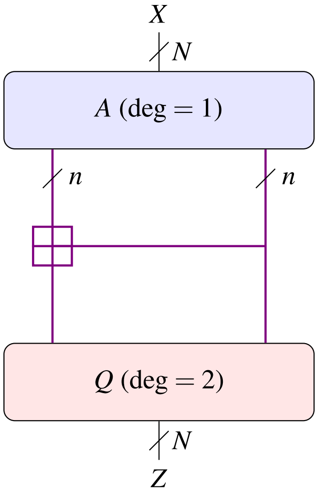
Sparsity observations

- let $S : (x, y) \mapsto (x \boxplus y, y)$
- let A, Q be affine-quadratic self-equiv. of S:
  $S = Q \circ S \circ A$
- theorem: half of outputs of Q has to be linear
- experimental: at most 3 quadratic outputs (linearly independent)



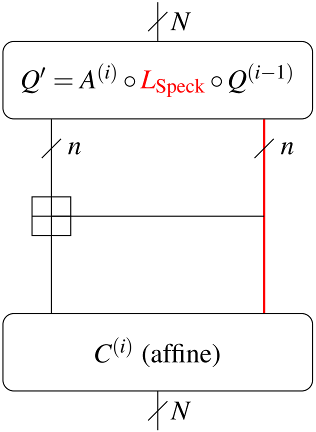
Sparsity observations

- let $S : (x, y) \mapsto (x \boxplus y, y)$
- let A, Q be affine-quadratic self-equiv. of S:
  $S = Q \circ S \circ A$
- theorem: half of outputs of Q has to be linear
- experimental: at most 3 quadratic outputs (linearly independent)
- experimental: each of them consists of 1-2 quadratic monomials (up to affine-equivalence)
- example:
  $x_0 y_0, (x_{n-1} + y_{n-1})(x_1 + x_5 + \ldots + y_1 + y_5 + \ldots)$



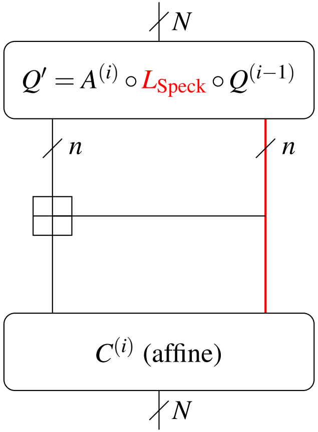
Step 1: locate exposed quadratic bits

- right branch leaks the quadratic monomials of B



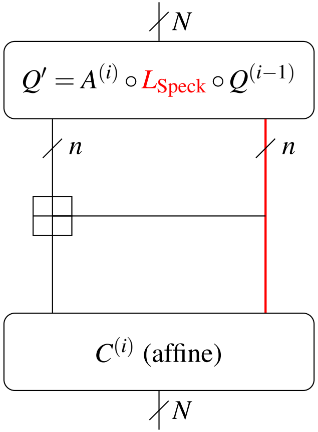
Step 1: locate exposed quadratic bits

- right branch leaks the quadratic monomials of B
- degree-2 outputs zero-sum on any 3-dimensional subspace
- but not on all 2-dimensional subspaces (separate from degree-1 output)



Step 1: locate exposed quadratic bits

- right branch leaks the quadratic monomials of B
- degree-2 outputs zero-sum on any 3-dimensional subspace
- but not on all 2-dimensional subspaces (separate from degree-1 output)
- linear algebra to find corresp. part of $C^{(i)}$



Step 2: decompose into monomials

Problem

Given quadratic Boolean polynomial f, find a linear map A such that $f(A(x))$ has smallest number of quadratic terms

Example

Instance: $f(x) = x_0x_2 + x_0x_5 + x_0x_6 + x_1x_3 + x_1 + x_2x_3 + x_2x_5 + x_2x_6 + x_2 + x_5 + x_6$

Step 2: decompose into monomials

Problem

Given quadratic Boolean polynomial f, find a linear map A such that $f(A(x))$ has smallest number of quadratic terms

Example

Instance: $f(x) = x_0x_2 + x_0x_5 + x_0x_6 + x_1x_3 + x_1 + x_2x_3 + x_2x_5 + x_2x_6 + x_2 + x_5 + x_6$

Answer: $f(x) = (x_0 + x_1 + x_3)(x_2 + x_5 + x_6) + (x_1 + x_5 + x_6)(x_2 + x_3 + x_5 + x_6) + x_3 + x_1$

Step 2: decompose into monomials

Problem

Given quadratic Boolean polynomial f, find a linear map A such that $f(A(x))$ has smallest number of quadratic terms

Example

Instance: $f(x) = x_0x_2 + x_0x_5 + x_0x_6 + x_1x_3 + x_1 + x_2x_3 + x_2x_5 + x_2x_6 + x_2 + x_5 + x_6$

Answer: $f(x) = (x_0 + x_1 + x_3)(x_2 + x_5 + x_6) + (x_1 + x_5 + x_6)(x_2 + x_3 + x_5 + x_6) + x_3 + x_1$

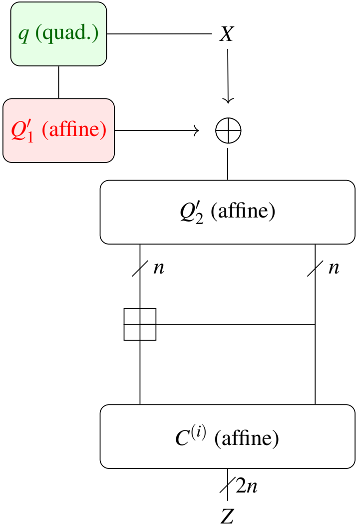
Definition (Linear Structures)

A linear structure δ of f is a probability-1 differential over f:

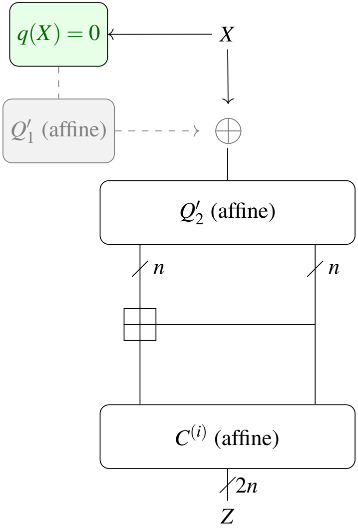
$$\exists c \; \forall x \quad f(x + \delta) = f(x) + c$$

Method: the dual space of LS is exactly the space of target linear combinations

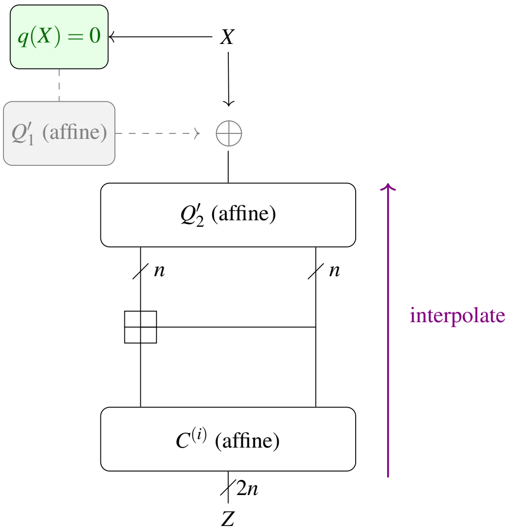
Step 3: algebraic recovery of Q'



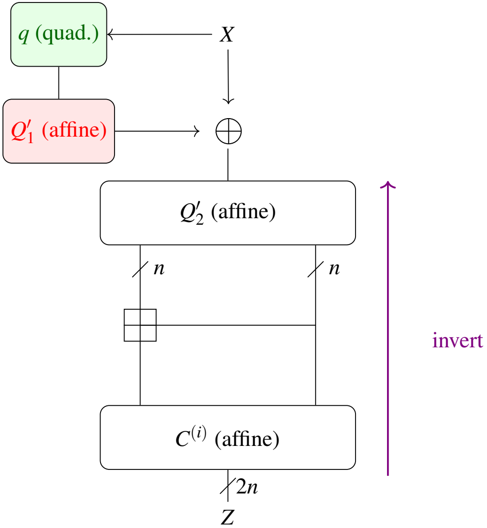
Step 3: algebraic recovery of Q'



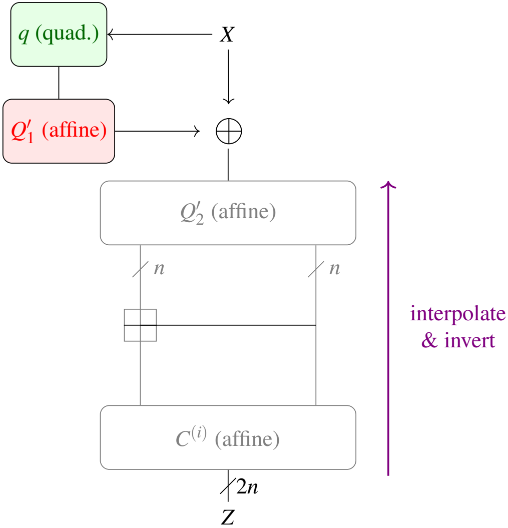
Step 3: algebraic recovery of Q'



Step 3: algebraic recovery of Q'



Step 3: algebraic recovery of Q'



Final Steps

1. inversion of Q - easy due to sparsity
2. "detach" Q from current round and "attach" to the previous round
3. run affine-encoded decomposition attack

Final Steps

1. inversion of Q - easy due to sparsity
2. "detach" Q from current round and "attach" to the previous round
3. run affine-encoded decomposition attack

4. combine round decompositions
5. extract subkeys (Vandersmissen, Ranea, and Preneel 2022)
6. recompute the **master** key (from 4 consecutive subkeys)

Plan

Conclusions

Implicit framework

- higher-degree graph obfuscation is not very useful
- still interesting tool, but need other targets (than 1 ARX round)

Conclusions

Implicit framework

- higher-degree graph obfuscation is not very useful
- still interesting tool, but need other targets (than 1 ARX round)

White-box ARX

- first algebraic attack for ARX designs (degrees 1,2)
- 1-round with affine/sparse-quadratic encodings is too weak
- need more rounds or stronger encodings

Conclusions

Implicit framework

- higher-degree graph obfuscation is not very useful
- still interesting tool, but need other targets (than 1 ARX round)

White-box ARX

- first algebraic attack for ARX designs (degrees 1,2)
- 1-round with affine/sparse-quadratic encodings is too weak
- need more rounds or stronger encodings

github.com/cryptolu/implicit_ARX_whitebox_cryptanalysis

tches.iacr.org/index.php/TCHES/article/view/10958