RSAC 2025 Conference

Many Voices.
One Community.

SESSION ID: CRYP-M01A

The Window Heuristic: Automating Differential Trail Search in ARX Ciphers with Partial Linearization Trade-offs

Emanuele Bellini, David Gerault, Juan Grados, Thomas Peyrin

TII, UAE, Nanyang Technological University, Singapore

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

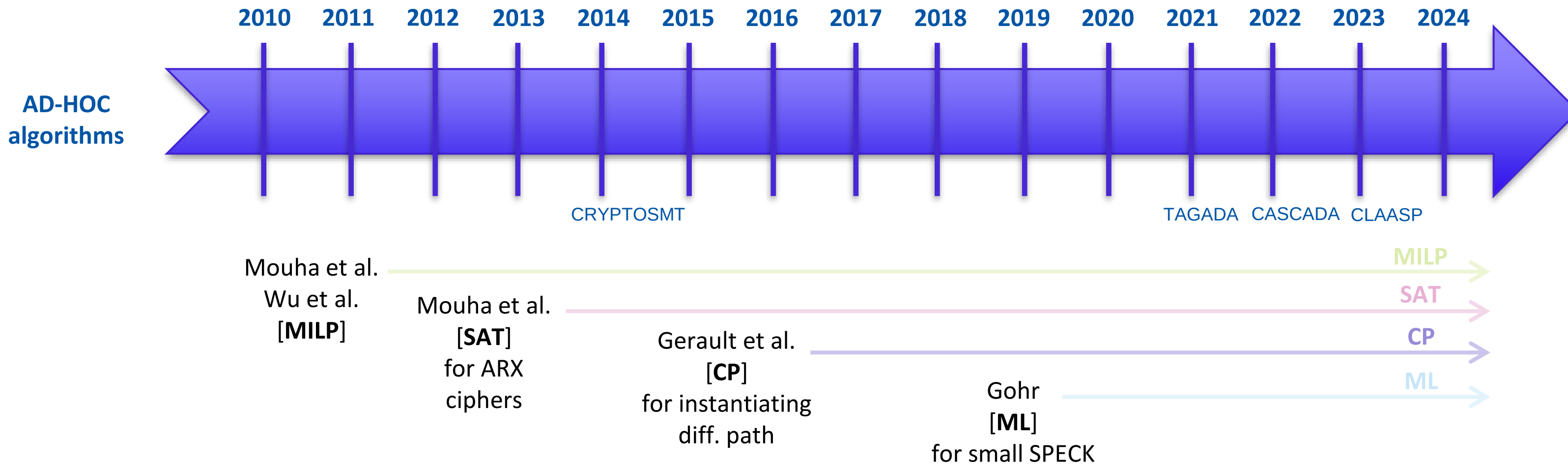
Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

RSAC 2025 Conference

Automated Cryptanalysis

Many Voices.
One Community.

Timeline of Automated Cryptanalysis



Today almost all cryptanalysis are assisted by solvers (FSE 2024: two entire sessions just for pure automated cryptanalysis)

Automated cryptanalysis using declarative frameworks (SAT/MILP/CP/etc.) is generally slower or at best same as ad-hoc tools, but so much **more convenient**

Mainly on **differential** and **linear cryptanalysis**, but now also on integral distinguishers, cube attacks, meet-in-the-middle attacks, etc.

Solving time is a crucial aspect and can be impacted by:

- the framework you use (SAT/MILP/CP/etc.)
- the strategy of modeling
- the solver
- the type of problem studied / scale

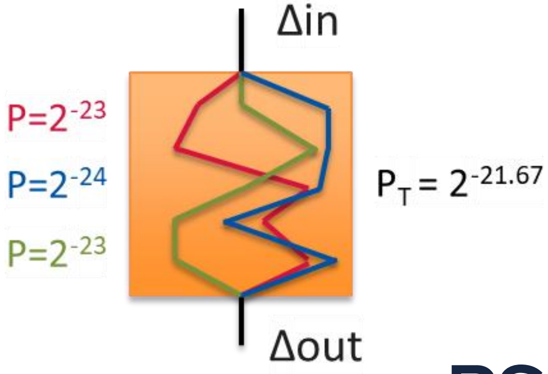
Automated Cryptanalysis for Differential Paths

Typically, for finding **differentials** or **differential trails**:

- Use **variables** to represent the various stages of the internal state bit differences during the round (and throughout the rounds)
- Use other variables to represent the **probability P** of the differential path (in -log2)
- Model a round of the cipher as a set of **declarative constraints** (Markov assumption !) to represent the difference propagation (either truncated or not). Use temporary variables if needed for certain components.
- Put all this into a system and use a **solver** on it.
- Can be combined with extra upper-level strategies (Matsui branch-and-bound, etc.)

One can:

- Find the best differential path / linear characteristic
- Enumerate the number of solutions
- Estimate the probability of a differential



Satisfiability Problem (SAT/SMT) for Cryptanalysis

All variables are **Boolean**, a constraint is a **Conjunctive Normal Form** (CNF – conjunction of disjunctions/clauses). Ex: $(a \vee b) \wedge (\neg a \vee c \vee d) \wedge (\neg b \vee c)$

- **Equality** is easy to model $x = y \Leftrightarrow (\neg x \vee y) \wedge (x \vee \neg y) = 1$
- **Linear layers** are easy to model, by combining the simple **XOR** model:
  $x = a \oplus b \Leftrightarrow (\neg a \vee b \vee x) \wedge (a \vee \neg b \vee x) \wedge (a \vee b \vee \neg x) \wedge (\neg a \vee \neg b \vee \neg x) = 1$
  (use dummy variables with combinations of XORs for multiple-inputs XORs)
- **Nonlinear layers (Sbox/AND/OR/Additions)** are more complex to model

Then, add a constraint to force the path to have a certain fixed probability P:

- If the solver returns SAT, we directly get a path with probability P.
- If the solver returns UNSAT, we run again this time with a probability P' < P (**drawback:** we need to iterate through decreasing target values).

SAT is very good **for all ciphers,** especially for **ARX ciphers**
SAT is very good for **finding optimal differential paths**, and good for **estimating the probability of a differential**.

Mixed Integer Linear Programming (MILP) for Cryptanalysis

All **variables** are **Boolean/integer/real**, a **constraint** is a linear **inequality**
Ex: $\{a + 4b + 2c \geq 3d + 7e\}$

- **Equality** is easy to model $x = y \Leftrightarrow \{x \geq y\}, \{y \geq x\}$
- **Linear layers** are easy to model, by using **4 inequalities**:
  $x = a \oplus b \Leftrightarrow \{a + b \geq x\}, \{a + x \geq b\}, \{b + x \geq a\}, \{a + b + x \leq 2\}$
  (use dummy variables with combinations of XORs for multiple-inputs XORs)
- **Nonlinear layers (Sbox/AND/OR/Additions)** are more complex to model

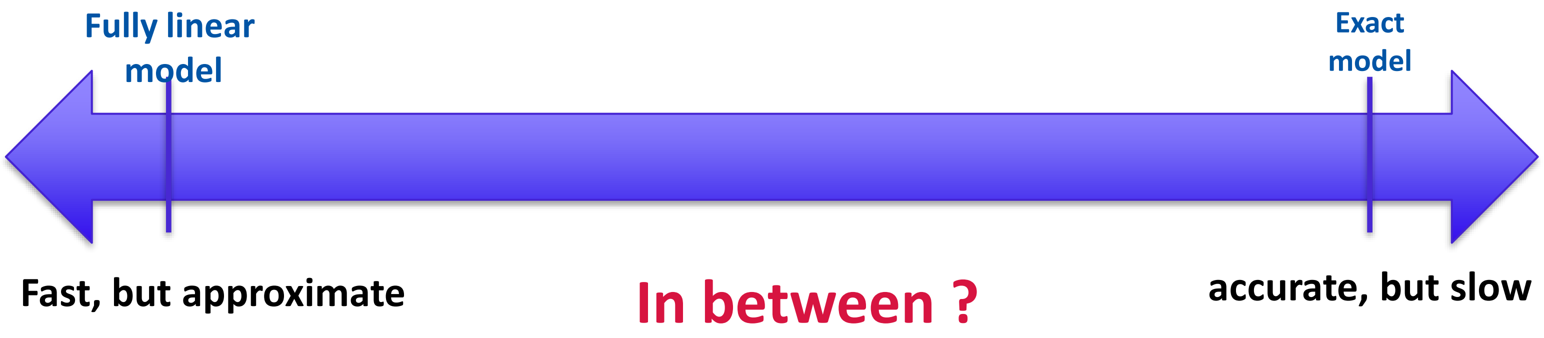
Then, add an **objective function** to minimize/maximize (encoding the path probability). Ex: $\{a + 3b + 2c + 5d\}$. The solver will directly return the best instance identified.

MILP is **good** on **most ciphers**.
MILP is **very good** for **finding optimal differential paths**, **less so** for **estimating the probability of a differential**.

The Problem with Modular Addition Modeling

- **Sboxes are relatively easy to model:** transform DDT into constraints
- **You can't use that strategy for ARX:** size is too large (32/64 bits)
- **To overcome this, researchers tried:**
  - to **fully linearise** mod. addition (Ex: MDx/SHA-x collision attacks): $x = a \boxplus b \approx a \oplus b$
  - to use Lipmaa-Moriai formula for XOR-differential propagation through addition:

    for $(\alpha, \beta \rightarrow \gamma)$: $eq(\alpha \ll 1, \beta \ll 1, \gamma \ll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \ll 1)) = 0$, with $eq(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z)$
  - to build the differential path iteratively, **only propagating the most likely differences** at each round (not representing the full cipher in a single model): partial DDT (pDDT)



RSAC 2025 Conference

The Window Heuristic

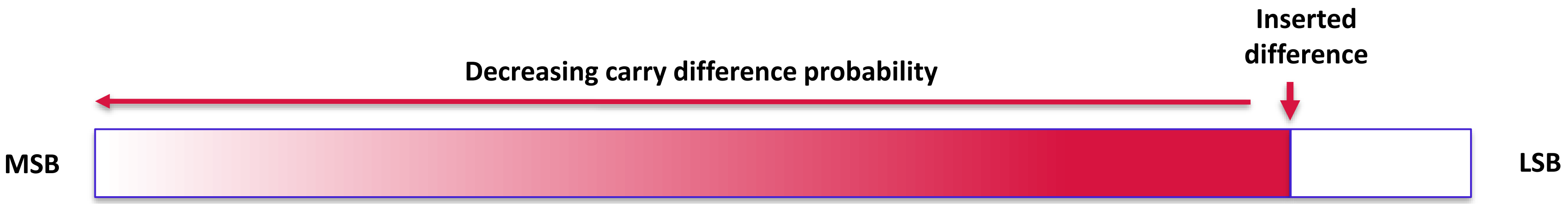
Many Voices.
One Community.

The Windows Heuristic Idea

A difference on bit i of a modular addition input can create a differential carry propagation from i to many later positions.

A linear model forces 0 carry difference propagation (but is fast), while exact model allows any propagation (but is slow).

- **Our observation:** good differential paths are (almost always) composed of modular addition diff. transitions that are propagating very little (if not at all), with only a few rounds with more propagation.

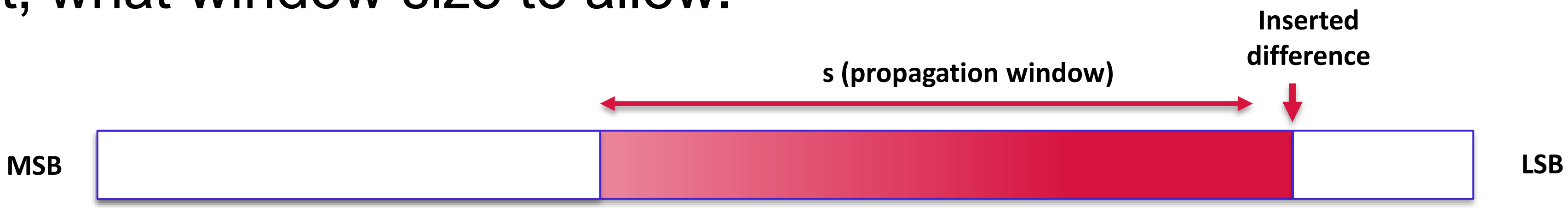


The Windows Heuristic Idea

A difference on bit i of a modular addition input can create a differential carry propagation from i to many later positions.

A linear model forces 0 carry difference propagation (but is fast), while exact model allows any propagation (but is slow).

- **Our observation:** good differential paths are (almost always) composed of modular addition diff. transitions that are propagating very little (if not at all), with only a few rounds with more propagation.
- **Our idea:** generalize the linear and exact models, by allowing a difference **in the carry** to propagate a little bit (over s bits, aka the "**window**" of propagation). We can control for each modular addition and each difference bit of it, what window size to allow.



The Windows Heuristic Idea

Example:

Difference 0x080000001e4a0848

0000100000000000000000000000000000011110010010100000100001001000

+

Difference 0x08000000000e0808

0000100000000000000000000000000000000000000011100000100000001000

=

Difference 0x00000000f2400040

0000000000000000000000000000000011110010010000000000000001000000

Carry Difference 0x00000000ec040000

0000000000000000000000000000000011101100000001000000000000000000

window

Encoding the Window in MILP and SAT

- In **MILP**, for each n-bit modular addition with s-bit window, we add the (n - 1 - s) following new constraints on the corresponding carry C:

$$\sum_{i=1}^{s+1} C[j-i] \leq s, \qquad for\ all\ j \in [s+1, \dots, n-1].$$

- In **SAT**, for each n-bit modular addition with s-bit window, we add the $2^{2(s+1)}$ (n - 1 - s) following new constraints on the corresponding carry C:

$$\bigwedge_{j=s+1}^{n-1} \bigvee_{i=1}^{s+1} \neg(a[j-i] \oplus b[j-i] \oplus c[j-i])$$

The Heuristics

Window Heuristic does not create invalid paths, but potentially miss good ones (good to find attacks, bad to prove the non-existence of diff. paths).

We observed that **fixing s for an entire round** generally works well in practice, but this remains dependent on the cipher and attack we study.

Other possible heuristics:

- Force each round to contain few consecutive carry difference clusters (in general small for good differential paths)
- Restrict the **total** number of carry differences for each modular additions (in general not too large for good differential paths)

Unfortunately, these other heuristics did not improve the results.

RSAC | 2025 Conference

Results

Many Voices.
One Community.

Application to SPECK Block Cipher

Version	Rounds	$\log_2(P_{DK})$	$\log_2(P_D)$	$\log_2(P_K)$	Ref.
SPECK-32/64	15	-94.0	-32.0	-62.0	[42]
	15	-85.0	-32.0	-53.0	[41]
	15	-73.0	-31.0	-42.0	$w_s = 0$
SPECK-48/96	14	-68.0	-43.0	-25.0	[42]
	14	-66.6	-43.0	-23.6	[41]
	14	-65.0	-34.0	-31.0	$w_s = 0$
	15	-89.0	-46.0	-43.0	[42]
	15	-83.5	-42.0	-41.5	[41]
	15	-75.0	-41.0	-34.0	$w_s = 0$
	16	-86.0	-43.0	-43.0	$w_s = 0$
SPECK-64/128	14	-88.0	-37.0	-51.0	[42]
	14	-72.0	-35.0	-37.0	[41]
	14	-66.0	-32.0	-34.0	$w_s = 0$
	15	-105.0	-45.0	-60.0	[42]
	15	-89.0	-42.0	-47.0	[41]
	15	-76.0	-33.0	-43.0	$w_s = 0$
	16	-103.0	-60.0	-43.0	[42]
	16	-85.0	-39.0	-46.0	$w_s = 0$
	17	-112.0	-62.0	-50.0	[42]
SPECK-128/256	16	-121.0	-76.0	-45.0	[42]
	16	-96.0	-52.0	-44.0	$w_s = 0$
	19	-190.0	-111.0	-79.0	[42]
	19	-171.0	-102.0	-69.0	$w_s = 0$

[41] Qin et al.
CoRR abs/2203.09741 - 2022

[42] Sadeghi et al.
Des. Codes Cryptography 2021

Application to ChaCha

Differential paths on ChaCha - comparison

Number of rounds	MILP + window heur.		SAT + window heuristic + "full window size"	MILP [7]	S-function [7]
	Pr	w_s	Pr	Pr	Pr
1	3	3	3	3	3
2	37	3	37	37	37
3	120	1	120	147	157
4	218	1	217	316	349

[7] Bellini et al. - IJACT 2023

Differential paths on ChaCha with MILP

r	Single-key			
	no-condition		conditions	
	Time	Pr	Time	Pr
1	2.17s	3	1.62s	3
2	13.2s	37	3.30s	37(0)
3	13.6s	147	319s	120(0)
4	36728s	316	48745s	218(1)

Boomerang Distinguishers on ChaCha

Number of rounds	Theoretical probability	Experimental probability
2	2^{-6} ($w_s = -1$, bottom part starting round 2)	$2^{-0.05}$
3	2^{-10} ($w_s = -1$, bottom part starting round 2)	$2^{-1.41}$
4	2^{-90} ($w_s = 0$, bottom part starting round 3)	-
5	2^{-154} ($w_s = 1$, bottom part starting round 3)	-
6	2^{-228} ($w_s = 1$, bottom part starting round 3)	-

Application to LEA Block Cipher

Differential paths on LEA

r	Window size		Others			
	Pr	w_s	Pr	Ref.	Pr	Ref.
12	107	9	112	[43]	107	[3]
13	123	1	134	[43]	127	[3]

Differential paths on LEA (time)

r	Single-key			
	no-condition		conditions	
	Time	Pr	Time	Pr
12	100709s	107/(-1)	73405s	107/(9)
13	609.53s	143/(-1)	7985s	123/(1)

[3] Bagherzadeh et al. - IET Information Sec. 2020

[43] Song et al. - ACISP 2016

Key recovery on LEA

Distinguisher			Key recovery		
Rounds	Pr	Ref.	Rounds	Complexity	Ref.
13	$2^{-123.79}$	[43]	14	$2^{124.79}$	[43]
13	$2^{-110.97}$	Ours ($w_s = 1$)	14	$2^{111.97}$	Ours ($w_s = 1$)

RSAC 2025 Conference

Conclusion
-
Future Works

Many Voices.
One Community.

Conclusion

- Maybe **better heuristics** exist with such window modeling trick
- Generalize the way we limit the carry, **maybe other good representations** are possible
- Generalize to **more complex operations** cases (multiplications, algebraic formula, ..)
- What about **linear cryptanalysis** ? Truncated differential cryptanalysis ?
- Incorporate the Window Heuristic into larger models with **key recovery**

RSAC 2025 Conference

Many Voices.
One Community.