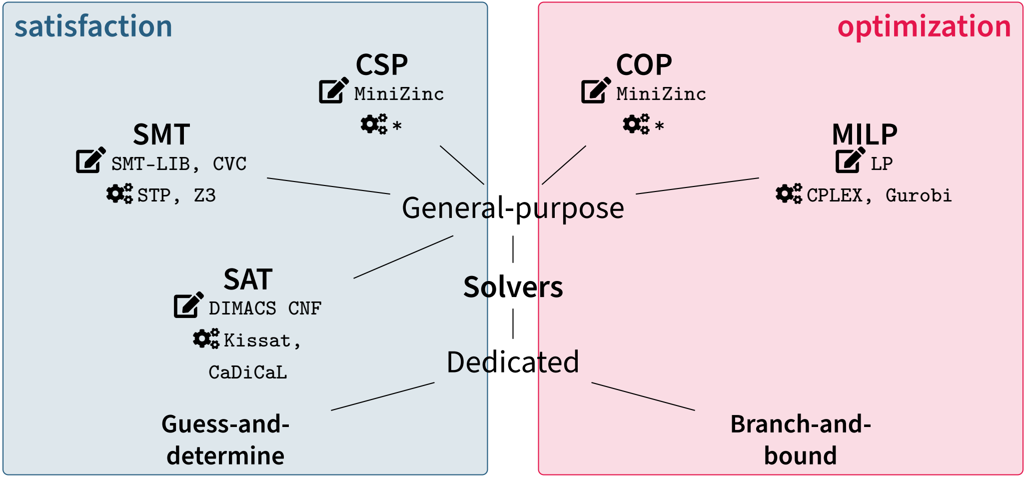
Challenges in Tool-based Cryptanalysis

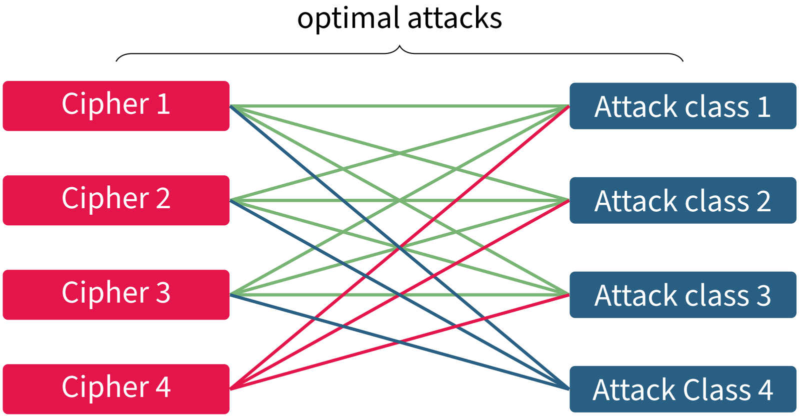
Maria Eichlseder

SKCAM 2025 Workshop on Symmetric-Key Cryptanalysis Automation and Modelling

Tools for Cryptanalysis



The Ideal Framework



Frameworks: The Challenges of Getting the Full Picture

Cipher representation

- What are the right abstraction layers? Which design paradigms to support?
- **Simplicity vs. versatility**

Precision of results

- What to assume, what to analyze?
- **Efficiency vs. quality**

Scope of results

- Building blocks or end-to-end attacks?
- **Optimality vs. feasibility**

Cipher Representation

Simplicity vs. Versatility

Direction 1: The Right Abstraction Level

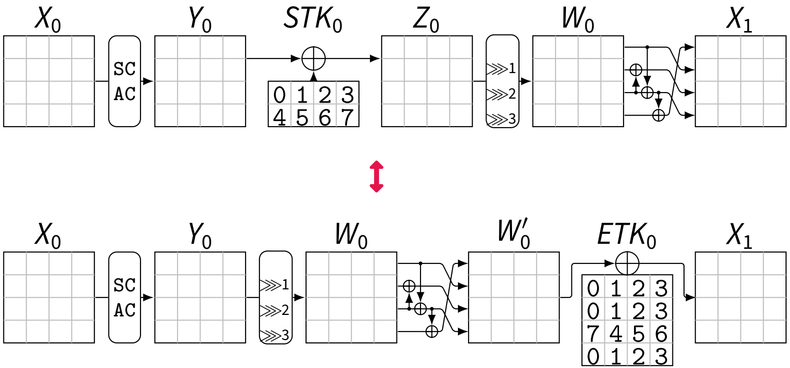
- **Construction**: Key-alternating SPN/LS, Feistel, Reflection, Alignment, …
- **Operations**: S-box, MDS matrix, modular addition, permutation, $\mathbb{F}_p$, …
- **Gates** and local operations: $\oplus$, $\odot$, …

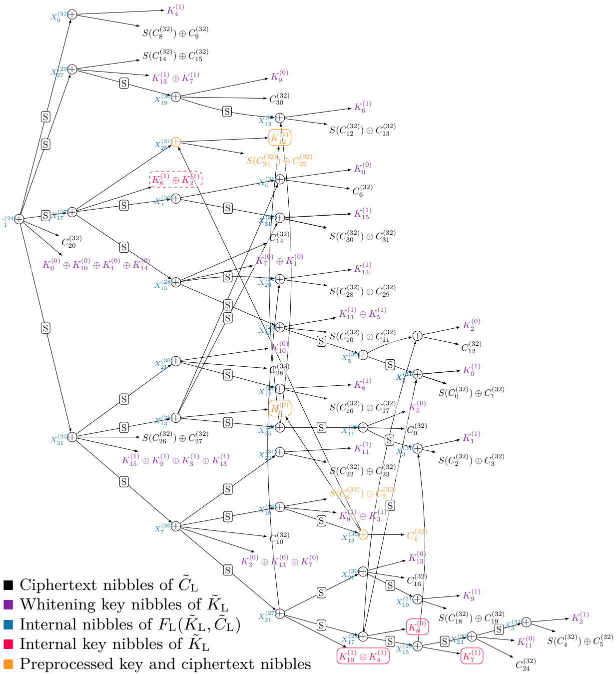
Decomposition into smaller functions:

- Necessary for efficient modelling
- Loses information (invertibility, MDS property, …), introduces inaccuracies

Direction 2: Rewiring the Cipher

- Restructure the cipher circuit representation for optimizations
- **Examples:** Equivalent subkeys, Round boundary, $f(C \oplus K)$ for FFT, ...





■ Ciphertext nibbles of $\tilde{C}_{\mathrm{L}}$

■ Whitening key nibbles of $\tilde{K}_{\mathrm{L}}$

■ Internal nibbles of $F_{\mathrm{L}}(\tilde{K}_{\mathrm{L}}, \tilde{C}_{\mathrm{L}})$

■ Internal key nibbles of $\tilde{K}_{\mathrm{L}}$

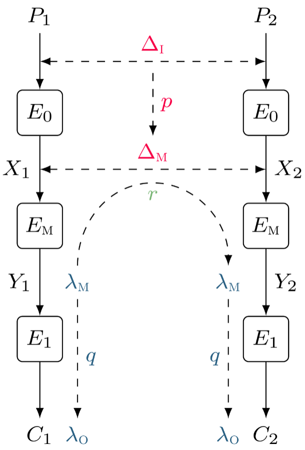
■ Preprocessed key and ciphertext nibbles

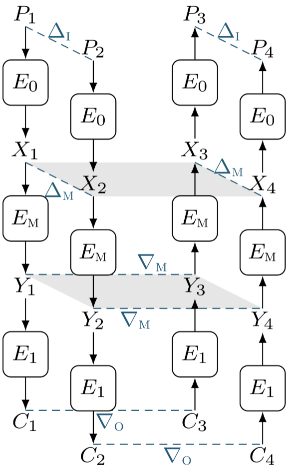
Precision of Results

Efficiency vs. Quality

Direction 3: Combined Trails

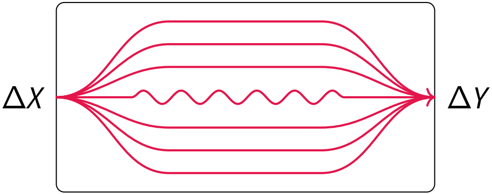
- **Examples:** Differential-linear, Boomerang, Miss-in-the-middle, …





Direction 4: Abundance of Trails

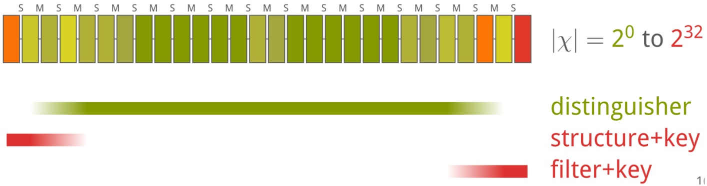
- More precise results by analyzing all relevant trails
- Leads to 2-level optimization problems
- **Examples:** Differential clustering, Linear hull, Quasidifferential trails



Direction 4: Abundance of Trails

- More precise results by analyzing all relevant trails
- Leads to 2-level optimization problems
- **Examples:** Differential clustering, Linear hull, Quasidifferential trails

- Representing partial knowledge and sets of trails
- **Examples:** (Partially) truncated propagation, Guess-and-determine



Direction 4: Abundance of Trails

- More precise results by analyzing all relevant trails
- Leads to 2-level optimization problems
- **Examples:** Differential clustering, Linear hull, Quasidifferential trails

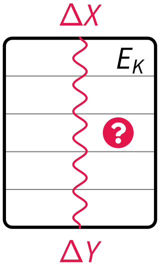
- Representing partial knowledge and sets of trails
- **Examples:** (Partially) truncated propagation, Guess-and-determine

$\nabla(z_j, z_j^*)$	$\nabla(z_j, z_j^*)$	$\nabla(z_j, z_j^*)$	$\nabla(z_j, z_j^*)$
0 = ○○○●	- = ●○○●	3 = ○○●●	7 = ○●●●
u = ○○●○	x = ○●●○	5 = ○●○●	B = ●○●●
n = ○●○○	# = ○○○○	A = ●○●○	D = ●●○●
1 = ●○○○	? = ●●●●	C = ●●○○	E = ●●●○

Direction 5: Absence of Trails

Some distinguishers are based on the **non-existence** of a valid trails

- **Differential** › Impossible differentials
- **Linear** › Zero-correlation linear approximations
- **Integral** › Division/monomial trail; ZC-based integrals



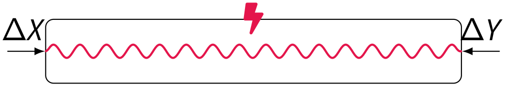
However, models for full attacks need **solution-based** distinguisher models
(or a quantified language like QSAT.)

Proving absence of absence-based distinguishers?

Two Ways of Modelling Impossibility

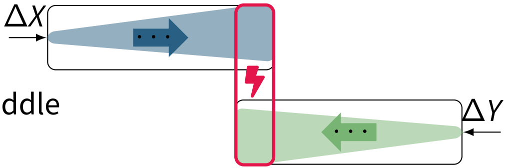
Unsatisfiability-based:

- First specify distinguisher, then check
- Precise, but potentially slow



Satisfiability-based:

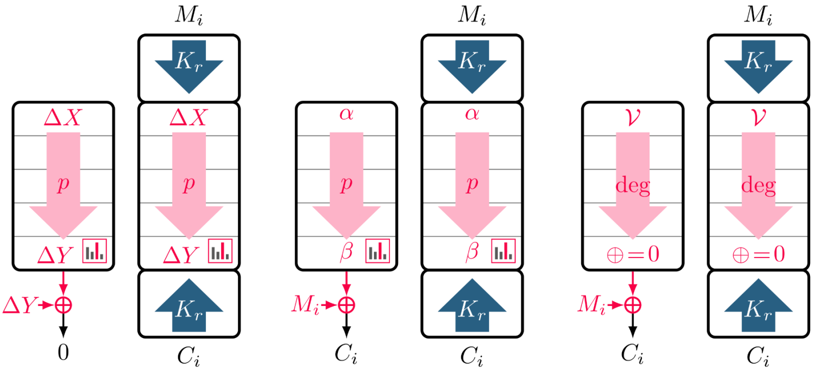
- Find distinguisher that misses in the middle
- Typically efficient, but less precise



Optimizing Full Attacks

Attack Architecture: What are we Looking for?

- **Block ciphers**:
  - Distinguisher (differential, linear, integral, combined trail?)
  - Key recovery (algorithm. . . ?)

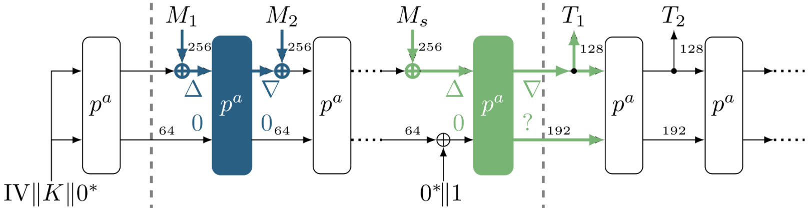


Attack Architecture: What are we Looking for?

- **Block ciphers**:
  - Distinguisher (differential, linear, integral, combined trail?)
  - Key recovery (algorithm…?)
- **Hash functions, compression functions**:
  - ???
- **Permutations**:
  - ???
- **Arithmetization-oriented primitives**:
  - ???

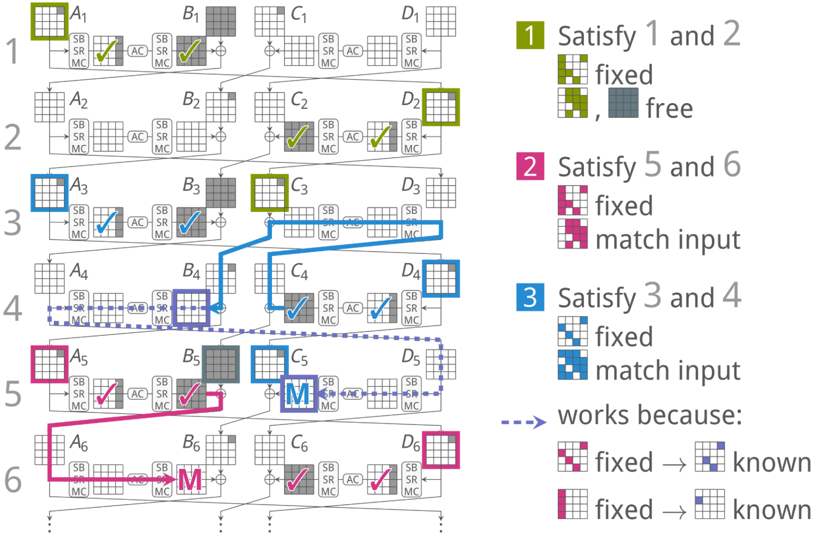
We need to know
- which space to search
- what we can vary
- what we can deduce

Attack Architecture: What are we Looking for?



(a) Forgery (■) or MAC forgery (■) via differential cryptanalysis

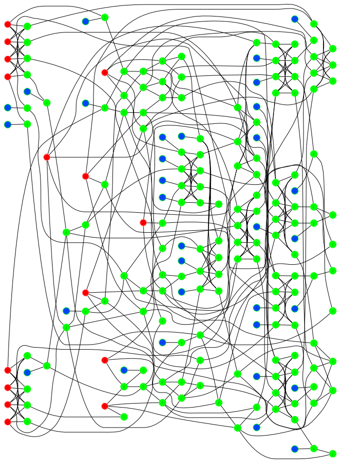
Attack Architecture: What are we Looking for?



Direction 7: Finding Multi-Step Algorithms

Optimizing Key-Recovery: Examples

- **Guess-and-determine** attacks:
  - </> Autoguess [HE22a] [a]

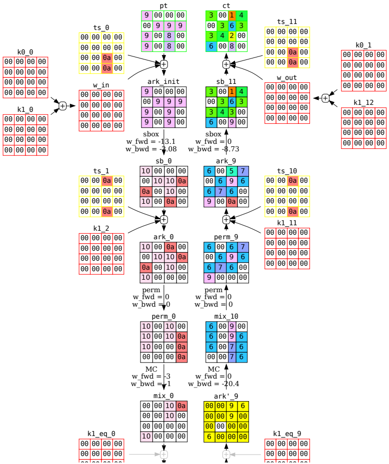


[a] https://github.com/hadipourh/autoguess
[b] https://extgit.isec.tugraz.at/castle/tool/keyrecoverytool
[c] https://github.com/hadipourh/mpt
[d] https://github.com/hadipourh/zero

Direction 7: Finding Multi-Step Algorithms

Optimizing Key-Recovery: Examples

- **Guess-and-determine** attacks:
  - </> Autoguess [HE22a] [a]
- **Differential** cryptanalysis:
  - </> keyrecoverytool [Nag22] [b]



[a] https://github.com/hadipourh/autoguess

[b] https://extgit.isec.tugraz.at/castle/tool/keyrecoverytool

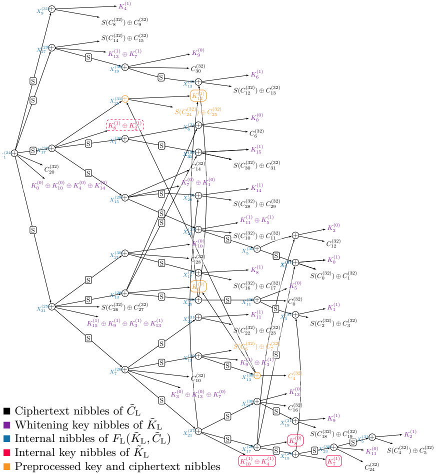
[c] https://github.com/hadipourh/mpt

[d] https://github.com/hadipourh/zero

Direction 7: Finding Multi-Step Algorithms

Optimizing Key-Recovery: Examples

- **Guess-and-determine** attacks:
  - </> Autoguess [HE22a] [a]
- **Differential** cryptanalysis:
  - </> keyrecoverytool [Nag22] [b]
- **Integral** cryptanalysis:
  - </> Graph-based [HE22b] [c]



[a] https://github.com/hadipourh/autoguess

[b] https://extgit.isec.tugraz.at/castle/tool/keyrecoverytool

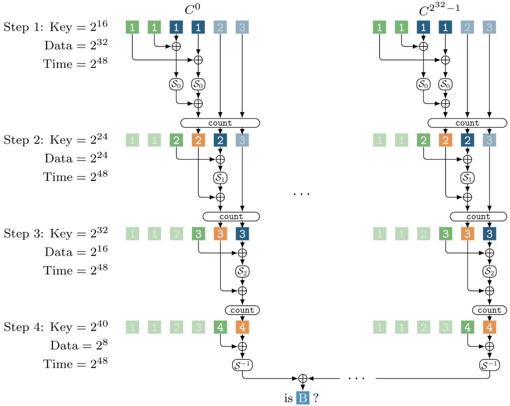
[c] https://github.com/hadipourh/mpt

[d] https://github.com/hadipourh/zero

Direction 7: Finding Multi-Step Algorithms

Optimizing Key-Recovery: Examples

- **Guess-and-determine** attacks:
  - </> Autoguess [HE22a] [a]
- **Differential** cryptanalysis:
  - </> keyrecoverytool [Nag22] [b]
- **Integral** cryptanalysis:
  - </> Graph-based [HE22b] [c]
  - </> AutoPSy [HSE23] [d]



[a] https://github.com/hadipourh/autoguess

[b] https://extgit.isec.tugraz.at/castle/tool/keyrecoverytool

[c] https://github.com/hadipourh/mpt

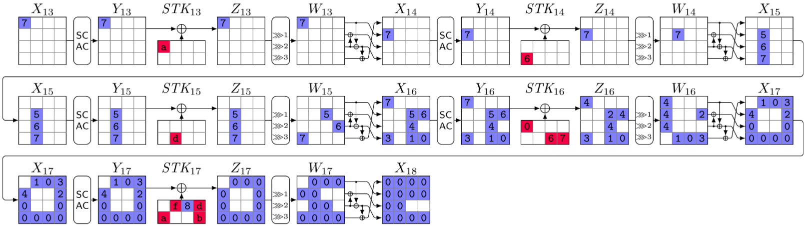
[d] https://github.com/hadipourh/zero

Direction 8: Optimizing Full Attacks

- Key recovery has long been ignored
- Fewer choices to make for the attacker

. . . but . . .

- Optimizations involve choices and tradeoffs
- Precise evaluation is tedious
- “Optimal” distinguisher doesn’t guarantee optimal attack



Step	Guessed	K × D =Mem	Time	Stored Texts
0	–	$2^0 \times 2^{40} = 2^{40}$	$2^{40-5.2}$	$Z_{17}[1,3,4,7]$; $X_{17}[8,11,12,13,15]$; $X_{16}[15]$
1	$STK_{17}[1]$	$2^4 \times 2^{36} = 2^{40}$	$2^{44-7.2}$	$Z_{17}[3,4,7]$; $X_{17}[8,11,12,15]$; $X_{16}[14,15]$
2	$STK_{17}[7]$	$2^8 \times 2^{32} = 2^{40}$	$2^{44-8.2}$	$Z_{17}[3,4]$; $X_{17}[8,12,15]$; $Z_{16}[6]$; $X_{16}[14,15]$
3	$STK_{17}[3]$	$2^{12} \times 2^{28} = 2^{40}$	$2^{44-7.2}$	$Z_{17}[4]$; $X_{17}[8,12]$; $Z_{16}[6]$; $X_{16}[12,14,15]$
4	$STK_{17}[4]$	$2^{16} \times 2^{28} = 2^{44}$	$2^{44-7.2}$	$Z_{16}[0,6,7]$; $X_{16}[10,12,14,15]$
5	$STK_{16}[6]$	$2^{20} \times 2^{20} = 2^{40}$	$2^{48-7.2}$	$Z_{16}[0,7]$; $X_{16}[12,15]$; $X_{15}[5]$
6	$STK_{16}[7]$	$2^{24} \times 2^{16} = 2^{40}$	$2^{44-7.2}$	$Z_{16}[0]$; $X_{16}[12]$; $X_{15}[5,9]$
7	$STK_{16}[0]$	$2^{28} \times 2^4 = 2^{32}$	$2^{44-6.2}$	$X_{13}[0]$
Σ		2^{44}	$2^{41.32}$	

Why Optimizing Full Attacks is Challenging

- Preferably use a **joint model** for distinguisher and key recovery
  - Only works for satisfiability-based distinguishers
- **Complexity formulas** are often complicated
  - Mix of polynomial/exponential terms; simplified assumptions
- **Multi-step** processes lead to heavy models
- Very different types of **key schedules**
- Many different **optimizations** and strategies

Conclusion

- **Cipher representation**
  - Simplicity vs. versatility
- **Precision of results**
  - Efficiency vs. quality
- **Scope of results**
  - Optimality vs. feasibility

Acknowledgements

European Research Council

Established by the European Commission

SPyCoDe

FWF

Der Wissenschaftsfonds.

This research was funded in part by the Austrian Science Fund (FWF) SFB project SPyCoDe (doi:10.55776/F85) and the European Research Council (ERC) starting grant KEYLESS (#101165216).

Bibliography

[HE22a] Hosein Hadipour and Maria Eichlseder. **Autoguess: A Tool for Finding Guess-and-Determine Attacks and Key Bridges.** ACNS 2022. Springer, 2022, pp. 230–250. DOI: 10.1007/978-3-031-09234-3_12.

[HE22b] Hosein Hadipour and Maria Eichlseder. **Integral Cryptanalysis of WARP based on Monomial Prediction.** IACR Trans. Symmetric Cryptol. 2 (2022), pp. 92–112. DOI: 10.46586/TOSC.V2022.I2.92-112.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. **Finding the Impossible: Automated Search for Full Impossible-Differential, Zero-Correlation, and Integral Attacks.** EUROCRYPT 2023. Springer, 2023, pp. 128–157. DOI: 10.1007/978-3-031-30634-1_5.

[Nag22] Marcel Nageler. **Automatic cryptanlysis of block ciphers: Finding efficient key-recovery attacks.** MA thesis. Graz University of Technology, 2022. DOI: 10.3217/n8ehm-dgj71.