Symmetric Cryptanalysis SotA and Future:

Towards a Generalization of Cryptanalysis Techniques

María Naya-Plasencia

ERC SoBaSyC

SKCAM

Symmetric-key Cryptanalysis Automation and Modelling

We have seen in the morning:

- Most popular existing tools for automated cryptanalysis.

We will see in the afternoon:

- Recent results and open problems contributing to cryptanalysis automation

SKCAM

Symmetric-key Cryptanalysis Automation and Modelling

We have seen in the morning:

- Most popular existing tools for automated cryptanalysis.

We will see in the afternoon:

- Recent results and open problems contributing to cryptanalysis automation

Now:

- We will see the context and the gap between both parts (not technical talk).

Symmetric Cryptography

Essential to secure communications efficiently: widely used.

Two main tasks:

- Design symmetric primitives (answering needs).
- Cryptanalysis, foundation of trust:
  applying known attacks to new primitives & discovering new attacks.

Symmetric Cryptography

Essential to secure communications efficiently: widely used.

Two main tasks:

- Design symmetric primitives (answering needs).
- Cryptanalysis, foundation of trust:
  applying known attacks to new primitives & discovering new attacks.

What is the problem?

In current cryptanalysis context:

- Many serious drawbacks
- Impossible to efficiently check if a primitive resists all known attack techniques

$\Longrightarrow$ Impacts confidence levels for symmetric primitives.

Symmetric Cryptanalysis Context

Many recent competitions for new standards motivated by new needs:

- 200+ new symmetric primitives designs.
- Huge amount of new ad hoc cryptanalysis results (known and new families).

Paradox: urgency of results does not allow to fully understand them – waste of time.

Main Drawbacks Generated by the Context

Drawback 1:
Low security understanding of new designs, sometimes broken by known techniques.

Drawback 2:
Some published attacks incorrect or non-optimal.

Drawback 3:
Many techniques re-discovered or/and renamed, and not reaching full potential.

Algorithmic Approach to Cryptanalysis

A bit chaotic, sometimes better to take a step back!

First objective

Generic algorithmic framework capturing all variants of known cryptanalysis techniques

Why is it hard?

Ad hoc attacks against specific primitives:

- relevant generalization and formalism?
- underlying algorithmic problems?

Final ideal objective

Include all this algorithms in an Open Cryptanalysis Platform (see Thomas's talk)
$\Longrightarrow$ the reference security benchmark for symmetric primitives.

About Existing Tools

Very nice and important effort! But,

- Often, tools dedicated to concrete attacks or even a concrete step of the attack, or on concrete ciphers. Most generalizations on isolated steps or include heuristics getting away from optimality.

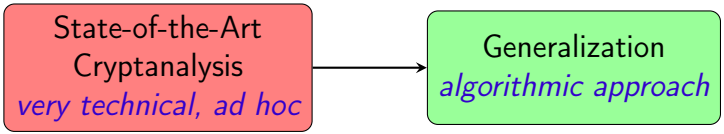
- Because of the chaotic situation, many do not include most existing improvements in the litterature (very difficult).

- Often hard to use: not systematically used by designers when proposing new designs in rationalle (see Maria E's talk for this).

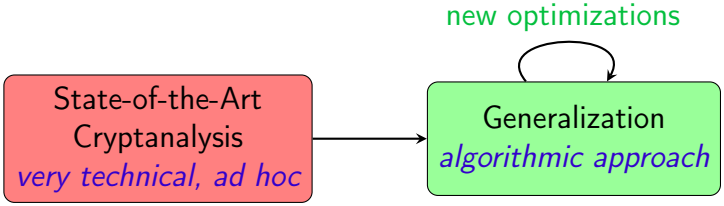
From Ad Hoc to Generic, from Approximative to Accurate

State-of-the-Art
Cryptanalysis
very technical, ad hoc

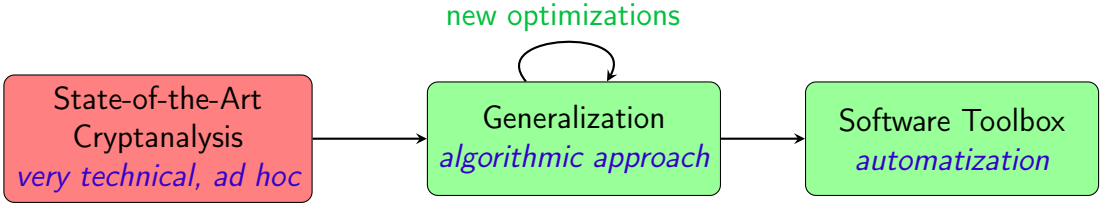
From Ad Hoc to Generic, from Approximative to Accurate



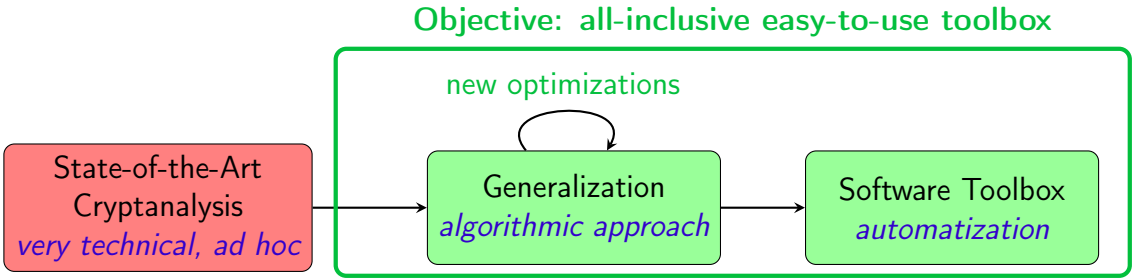
From Ad Hoc to Generic, from Approximative to Accurate



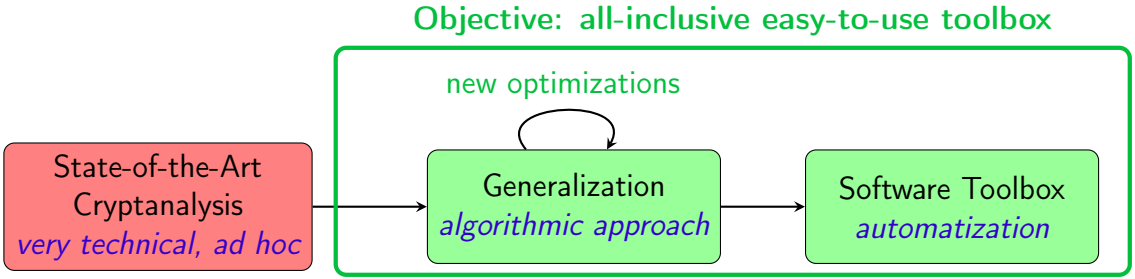
From Ad Hoc to Generic, from Approximative to Accurate



From Ad Hoc to Generic, from Approximative to Accurate

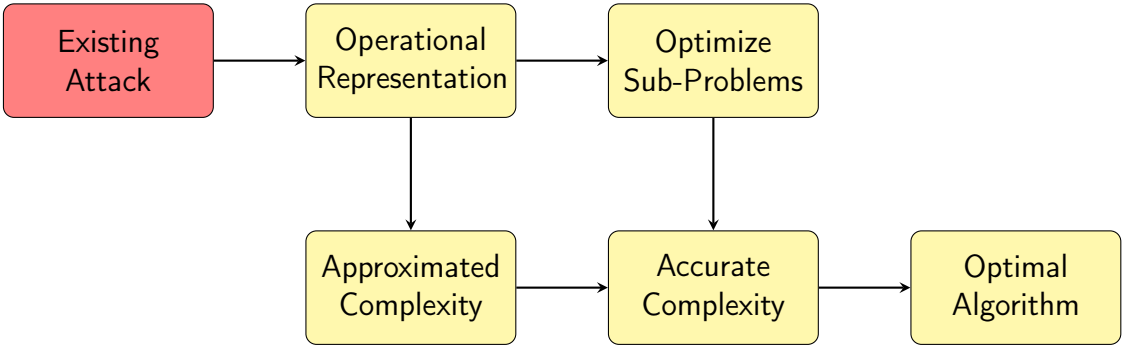


From Ad Hoc to Generic, from Approximative to Accurate

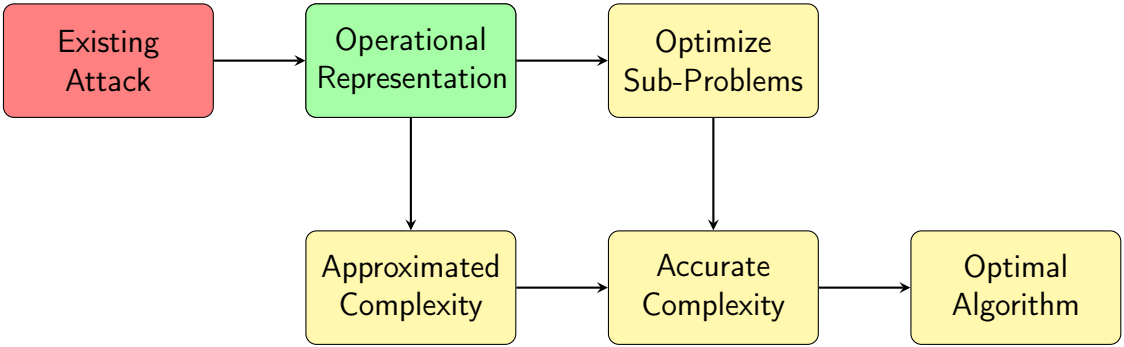


Many preliminary steps already done in these directions, but many left!

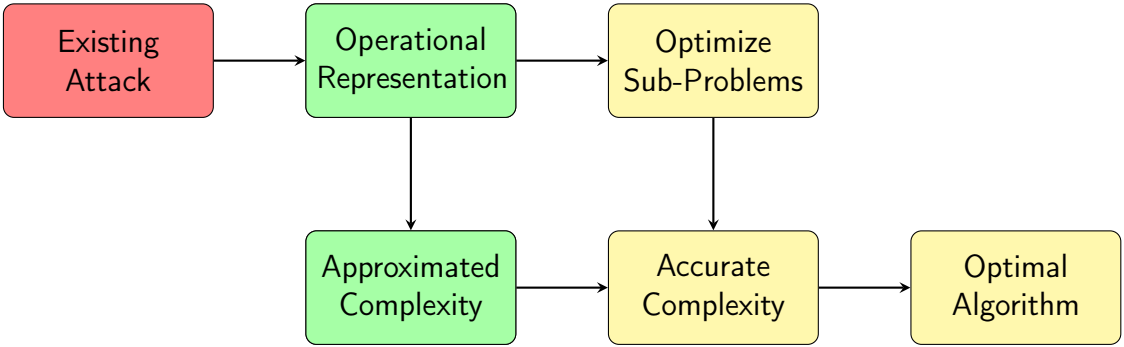
The Generalization Step



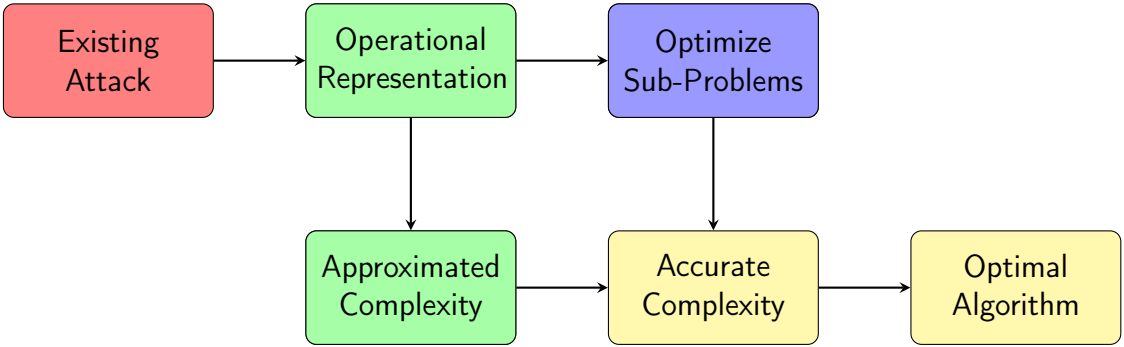
The Generalization Step



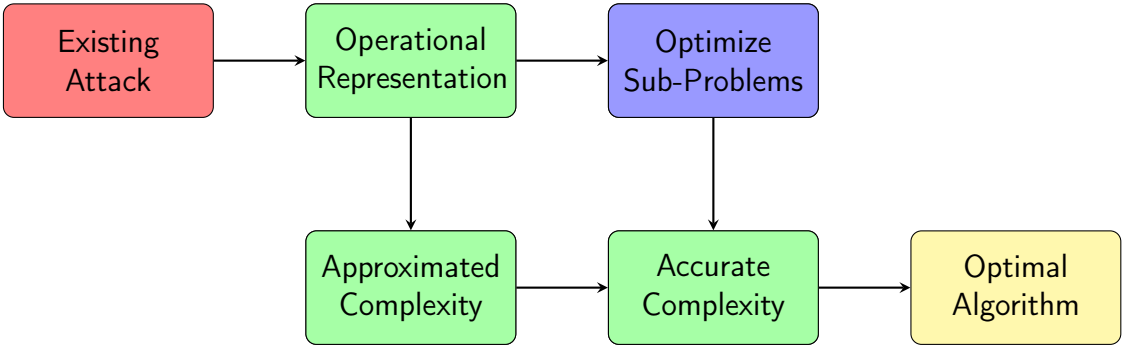
The Generalization Step



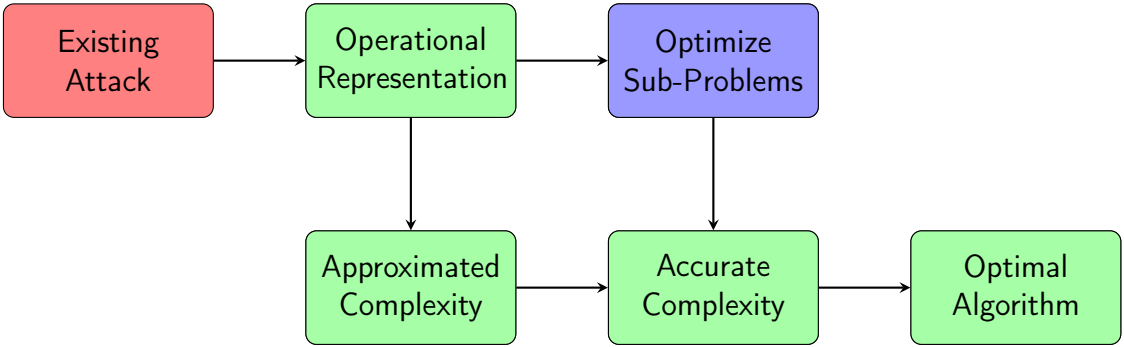
The Generalization Step



The Generalization Step

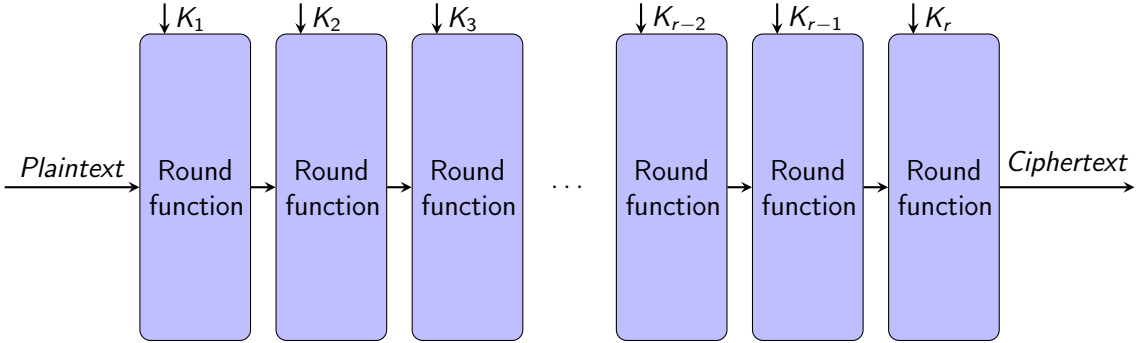


The Generalization Step



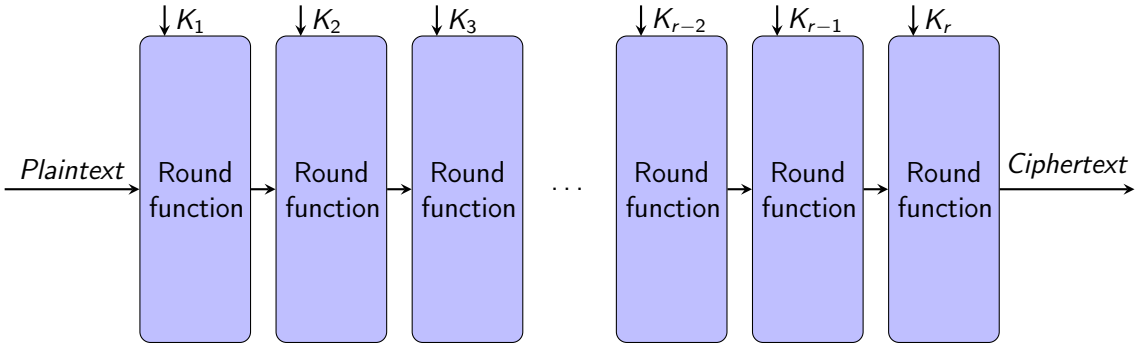
Example Case: Generalizing Differential Attacks [BS90]

Symmetric primitives are iterative. Attack: recover the secret faster than brute force.



Example Case: Generalizing Differential Attacks [BS90]

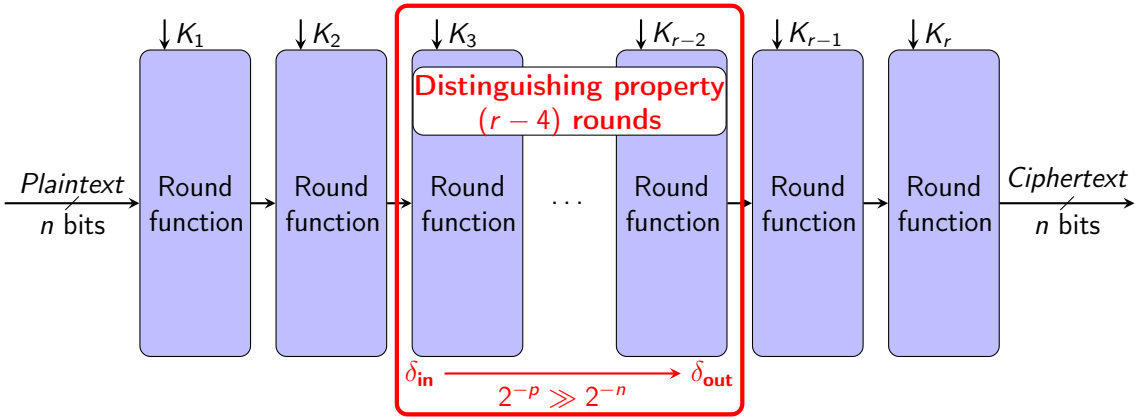
Finding attacks is a tedious procedure, often in 2 separate (but non-independent) steps:



Example Case: Generalizing Differential Attacks [BS90]

Finding attacks is a tedious procedure, often in 2 separate (but non-independent) steps:

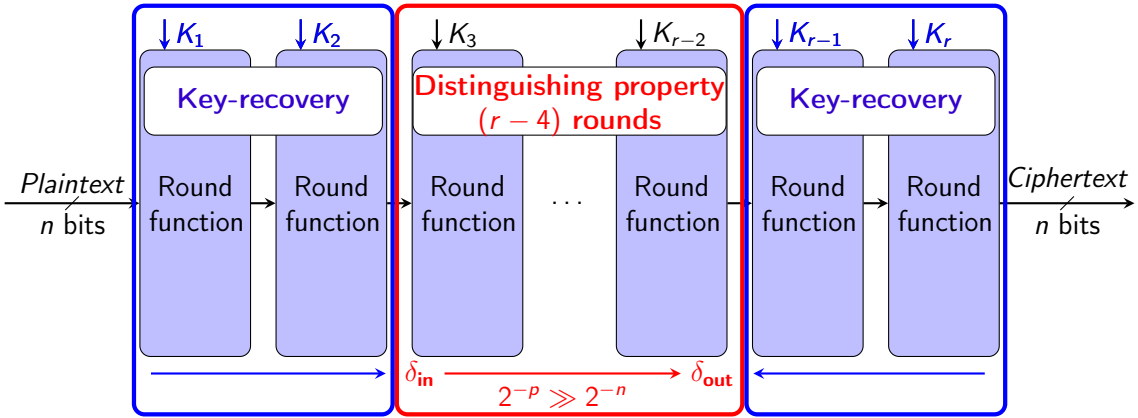
1. Find a distinguishing property.



Example Case: Generalizing Differential Attacks [BS90]

Finding attacks is a tedious procedure, often in 2 separate (but non-independent) steps:

1. Find a distinguishing property.
2. Find the optimal key-recovery.



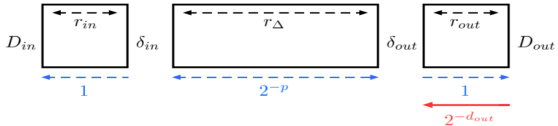
Existing Automation of (Differential) Attacks

- Finding a distinguisher:
  Some tools MILP, CP, SAT... find the best distinguisher for a given number of rounds (under some heuristics). See for instance Shaowei's talk.
- Finding the optimal key-recovery step:
  Tools for the best key-recovery step in concrete cases. See for instance Ling's talk.
- Finding the best overall attack (best distinguishers don't imply best attacks):
  Some tools partially include this with heuristics on the lower bound of the key-recovery (number of solutions). See for instance Hosein's talk this morning, or Patrick's talk on impossible differential attacks.

But an algorithm providing the best overall attacks? First generalization!

Ex: Generalize attacks? Start from a concrete attack.

1. Operational representation:



2.a Approximated complexity:

$$\mathcal{T} = \left(2^{p+1} + 2^{p+1}\frac{C_S}{C_E} + 2^{s+2d_{in}-1-n+d_{out}}\frac{C_{KR}}{C_E}\right)C_E.$$

2.b Optimize subproblems and include improvements.

Derive non-natural relevant parametters

3. More accurate complexity.
4. Repeat for other constructions: generalize and merge.
5. Put it all together -> towards an optimized and generalized algorithm to implement.

Interesting Tasks To Do

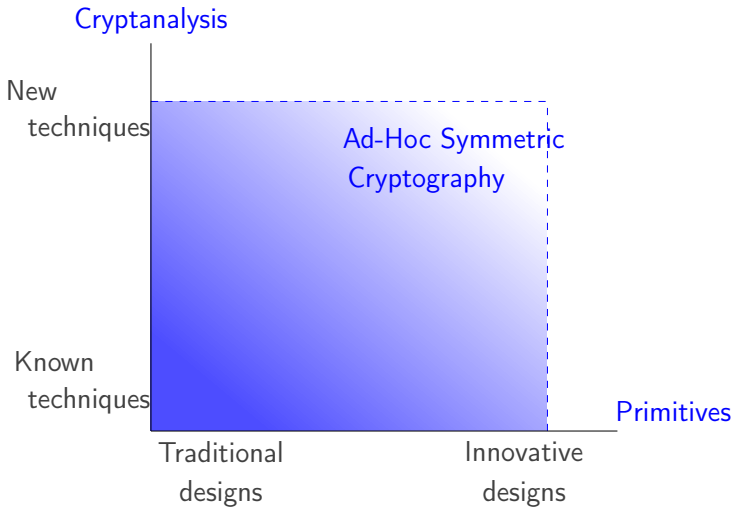
- Generalization steps representation and approximated complexity: have already been done for some families.
- Optimizing subproblems, including advanced variants, deriving more accurate complexities is more rare.

A lot of litterature to go through!

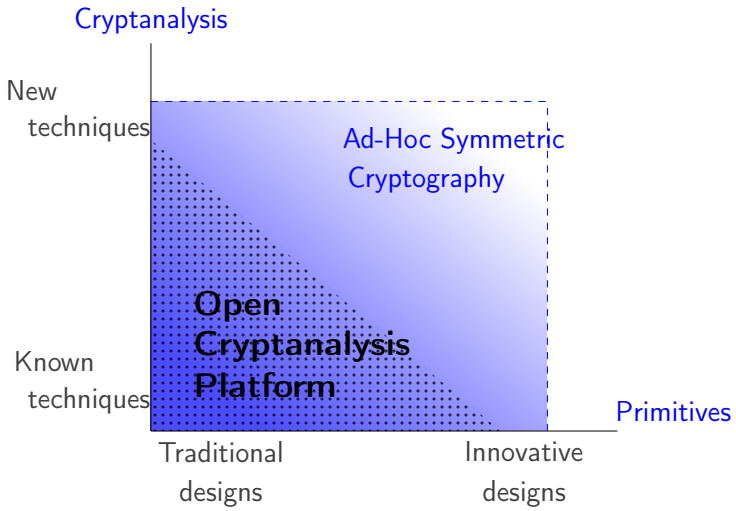
- In general, can be interesting to always take into account:
  - an "allowed precomputation parameter",
  - not to systematically cut steps at full rounds but consider partial rounds,
  - reusing more often gradual computations.

(Link to Hosein's talk.)

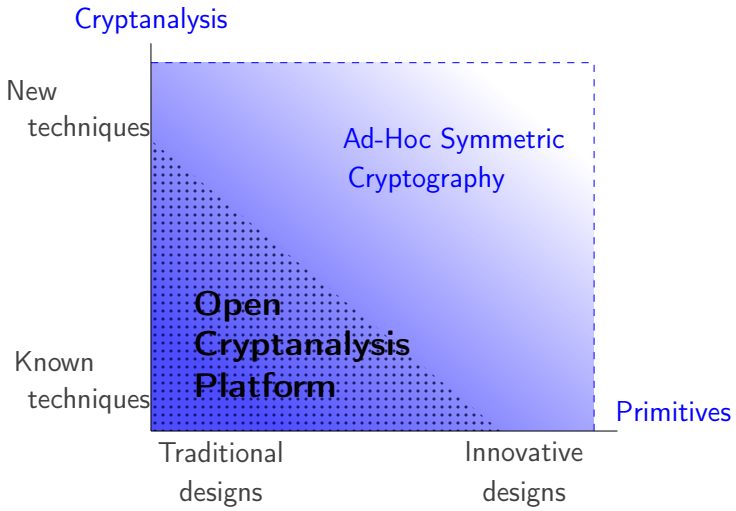
Different Scenarios



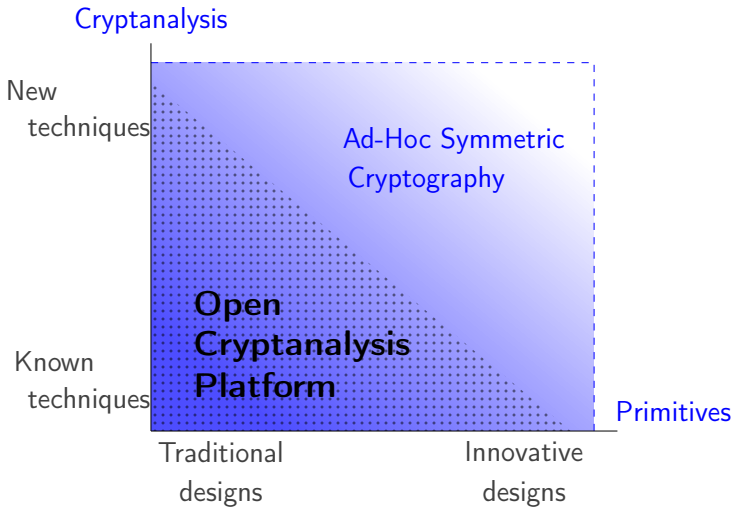
Different Scenarios



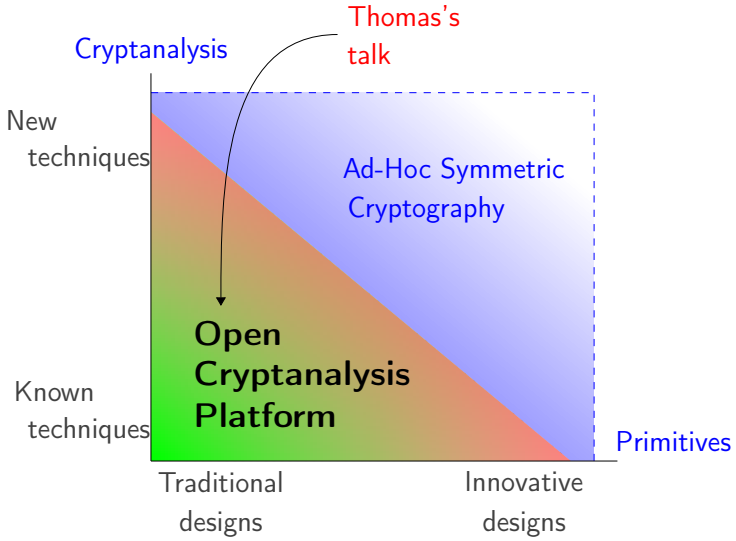
Different Scenarios



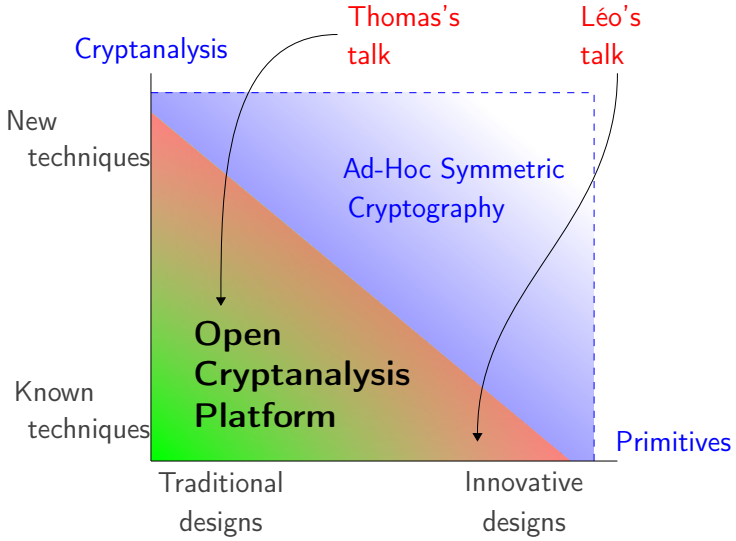
Different Scenarios



Different Scenarios



Different Scenarios



Conclusion and future

- We need to transform known symmetric cryptanalysis with an algorithmic formalism through generalisation.
- Long term goal (but we need to keep on working towards it now!): Reference benchmark for evaluating the resistance to known attacks.
- More time for new ad hoc results!
- Always new results, or more complicated to generalize: not everything will be included, but the toolbox will evolve.