Combining solvers to solve a Cryptanalytic Problem

David Gerault[1] Pascal Lafourcade[1] Marine Minier [2] Christine Solnon[3]

[1]LIMOS, Université Clermont Auvergne, [2]LORIA, Université de Lorraine [3]LIRIS, INSA Lyon

Doctoral Program of CP'17

UNION EUROPÉENNE
Fonds Européen de Développement Régional

LIMOS

UNIVERSITÉ Clermont Auvergne

Cryptography: Protecting messages

- Authenticity (Who sent the message?): Signature/MAC
- Integrity (Was the message modified?): Hash
- **Confidentiality** (Who can read the message?): Cipher

AES: TLS, SSH, secure messaging...

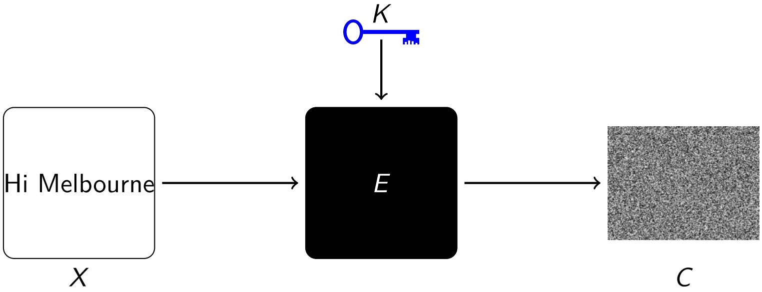
Automatic tools for Cryptography

Designing secure crypto is difficult

- Exhaustive search for attacks untractable
- Very hard to evaluate
- Iterative process: needs to be reasonably fast

Automatic tools are very popular in the community!

Block Ciphers

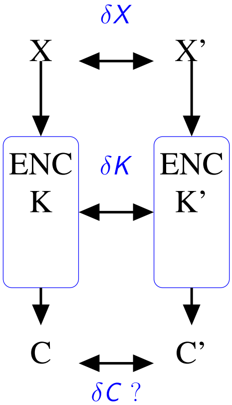


Keyed permutation $E\colon \{0,1\}^{\mathcal{K}} \times \{0,1\}^{\mathcal{P}} \to \{0,1\}^{\mathcal{P}}$**. Generally simple function iterated** n **times.**

Expected Property

Indistinguishable from a random permutation if K is unknown

Related Key Differential Cryptanalysis

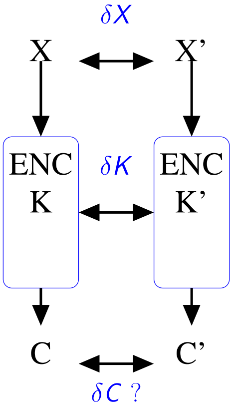


If for a random **secret** **key K,** δC **should be uniformly distributed**

Related Key Differential cryptanalysis

Changing the input (X,K) should not change the output (C) in a predictible way.

Related Key Differential Cryptanalysis



If for a random secret key K, δC should be uniformly distributed

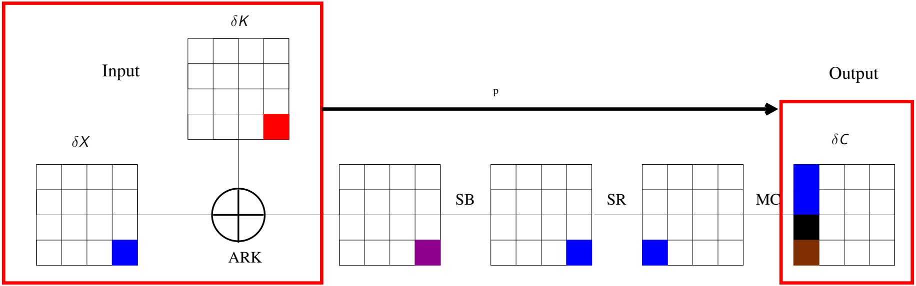
Related Key Differential cryptanalysis

Changing the input (X,K) should not change the output (C) in a predictible way.

But for real ciphers, δC is biased

Quantifying the bias

RK Differential characteristic: propagation pattern $(\delta X, \delta K) \rightarrow \delta C$

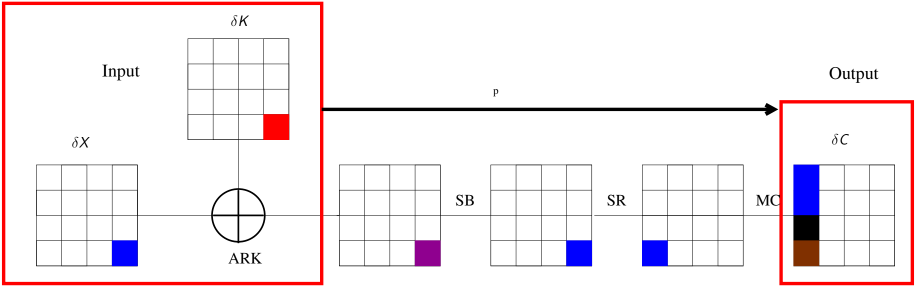


To evaluate a block ciphers, finding the best one is required

- Fix $\delta X, \delta K$
- Apply known propagation rules to obtain the most likely δC

The problem is solved for a given number of rounds

How difficult is it?



The SBoxes

Linearity is bad in a cipher, the SBoxes break it

- Linear operations: deterministic propagation
- SB: probabilistic propagation (127 possible output bytes for each input byte)

Size of the search space

128-bit message, {128,192,256}-bit key

2 steps solving

Step 1: boolean abstraction	**Step 2: actual byte values**
$\Delta = 0$	$\delta = 0$
$\Delta = 1$	$\delta \neq 0$
Find candidate solutions	Check their consistency

During Step 1, the SB operation is just identity

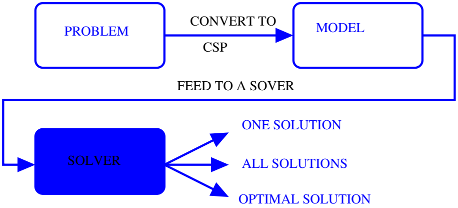
Step 1

Step1 returns outputs $\mathcal{O} = (\Delta X, \Delta K, \Delta C)$ and the corresponding difference propagation path, such that the number of Sboxes is minimal.

Step 2

For each solution to Step1, Step2($\mathcal{O}$) returns a fully instanciated RK differential characteristic with maximal probability if $\mathcal{O}$ is consistent, 0 otherwise.

CP



Our models

- One MiniZinc model for Step 1
- One Choco model for Step 2 (straightforward with table constraints)

Step 1

Very easy to model...

```
basicModelStep1(R) =>
  DX = new_array(R,4,4),       DX :: 0..1,
  DY = new_array(R-1,4,4),    DY :: 0..1,
  DK = new_array(R,4,4),       DK :: 0..1,
  foreach (I in 1..R-1, J in 1..4, K in 1..4)  % AddRoundKey
      sum([DY[I,J,K],DK[I+1,J,K],DX[I+1,J,K]]) #!= 1
  end,
  foreach(I in 1..R-1, K in 1..4)                  % MixColumns
      DX[I,1,K] + DX[I,2,(K mod 4)+1] + DX[I,3,((1+K) mod 4)+1]
    + DX[I,4,((2+K) mod 4)+1] + DY[I,1,K] + DY[I,2,K] + DY[I,3,K] + DY[I,4,K] #= S,
      S notin 1..4
  end,
  foreach(I in 2..R, J in 1..4)                    % KeySchedule
      sum([DK[I-1,J,1],DK[I-1,(J mod 4)+1,4],DK[I,J,1]]) #!= 1,
      foreach(K in 2..4)
          sum([DK[I-1,J,K],DK[I,J,K-1],DK[I,J,K]]) #!= 1
      end
  end.
```

...but too many inconsistent solutions

Step 1

Very easy to model...

```
basicModelStep1(R) =>
  DX = new_array(R,4,4),       DX :: 0..1,
  DY = new_array(R-1,4,4),    DY :: 0..1,
  DK = new_array(R,4,4),       DK :: 0..1,
  foreach (I in 1..R-1, J in 1..4, K in 1..4)  % AddRoundKey
      sum([DY[I,J,K],DK[I+1,J,K],DX[I+1,J,K]]) #!= 1
  end,
  foreach(I in 1..R-1, K in 1..4)                  % MixColumns
      DX[I,1,K] + DX[I,2,(K mod 4)+1] + DX[I,3,((1+K) mod 4)+1]
    + DX[I,4,((2+K) mod 4)+1] + DY[I,1,K] + DY[I,2,K] + DY[I,3,K] + DY[I,4,K] #= S,
      S notin 1..4
  end,
  foreach(I in 2..R, J in 1..4)                    % KeySchedule
      sum([DK[I-1,J,1],DK[I-1,(J mod 4)+1,4],DK[I,J,1]]) #!= 1,
      foreach(K in 2..4)
          sum([DK[I-1,J,K],DK[I,J,K-1],DK[I,J,K]]) #!= 1
      end
  end.
```

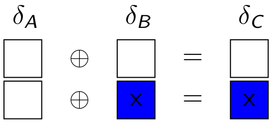
...but too many inconsistent solutions

We introduced byte level reasoning during Step 1 (See CP'16)

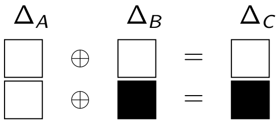
Example: XOR Constraint

(white = 0, colored ≠ 0)

Byte values



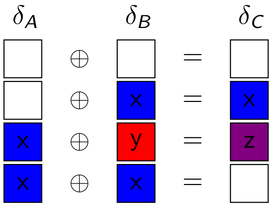
Boolean abstraction



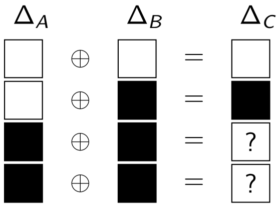
Example: XOR Constraint

(white $= 0$, colored $\neq 0$)

Byte values



Boolean abstraction



Δ_A	Δ_B	Δ_C
0	0	0
0	1	1
1	0	1
1	1	?

Inferring equalities from the result of a XOR helps filtering inconsistent solutions

Further Decompostion

Problem

Still not enough for larger key sizes (AES-192 and 256)

Combining solvers

Different solvers perform differently depending on the size of the search space

Minizinc challenge 2016: Picat_Sat is fast for finding one solution...

Further Decompostion

Problem

Still not enough for larger key sizes (AES-192 and 256)

Combining solvers

Different solvers perform differently depending on the size of the search space

Minizinc challenge 2016: Picat_Sat is fast for finding one solution...
...but slow for enumerating all solutions

Further Decompostion

Problem

Still not enough for larger key sizes (AES-192 and 256)

Combining solvers

Different solvers perform differently depending on the size of the search space

Minizinc challenge 2016: Picat_Sat is fast for finding one solution...
...but slow for enumerating all solutions

Our idea

Reducing the size of the search space with Picat_Sat, and then enumerating with Chuffed

Solving process

Usual representation

A solution:

- Δ X[i,j,k] for $i \in \{1..n\}, (j,k) \in \{0..3\}^2$
- Δ K[i,j,k] for $i \in \{1..n\}, (j,k) \in \{0..3\}^2$
- Δ Y[i,j,k] for $i \in \{1..n-1\}, (j,k) \in \{0..3\}^2$

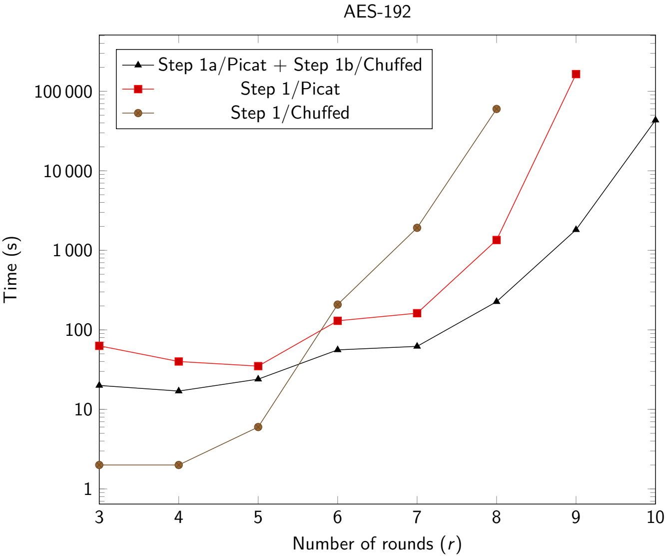
Class representation

- $\sum_{j=0}^{3} \sum_{k=0}^{3} \Delta X[i,j,k]$ for $i \in \{1..n\}$
- $\sum_{j=0}^{3} \sum_{k=0}^{3} \Delta K[i,j,k]$ for $i \in \{1..n\}$

Solution process

- List all solution classes with Picat
- For each class, list all solutions with Chuffed

Results



(timeout=48h)

Related Work & Contributions: AES

Standard since 2000

Problem

Finding optimal RK differential characteristics on AES-128, AES-192 and AES-256

Previous work

- Biryukov et al., 2010 : Branch & Bound
  → Several hours (AES-128), several weeks (AES-192)
- Fouque et al., 2013 : Graph traversal
  → 30 minutes, 60 Gb memory, 12 cores (AES-128)

Related Work & Contributions: AES

Standard since 2000

Problem

Finding optimal RK differential characteristics on AES-128, AES-192 and AES-256

Previous work

- Biryukov et al., 2010 : Branch & Bound
  → Several hours (AES-128), several weeks (AES-192)
- Fouque et al., 2013 : Graph traversal
  → 30 minutes, 60 Gb memory, 12 cores (AES-128)

Our results

- 25 minutes (AES-128), 24 hours (AES-192), 30 minutes (AES-256)
- New (better) RK differential characteristics on all versions
- Disproved incorrect one found in previous work

Conclusion and future challenges

Contributions

- CP models for cryptographic problem for the AES[a]
- New decomposition technique to combine solvers
- Faster than previous work

[a] Available on gerault.net, and part of the MiniZInc challenge

Future challenges

- Many more cryptographic problems (see FSE'17)
- Many more ciphers
- Better understanding the relations between solvers

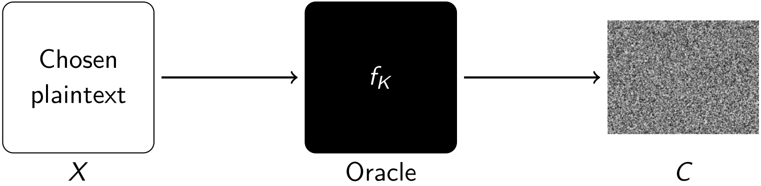
Take away message

The cryptography community is enthusiast about automatic tools, and has a lot of difficult problems to solve

Thank you for your attention

Questions?

Attacking a block cipher

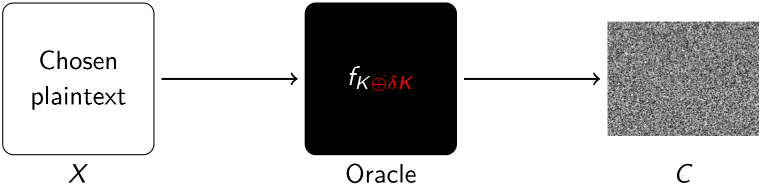


$$f \stackrel{?}{=} E \textbf{ or random permutation } \pi\textbf{?}$$

Distinguishing from $\pi \equiv$ recovering K

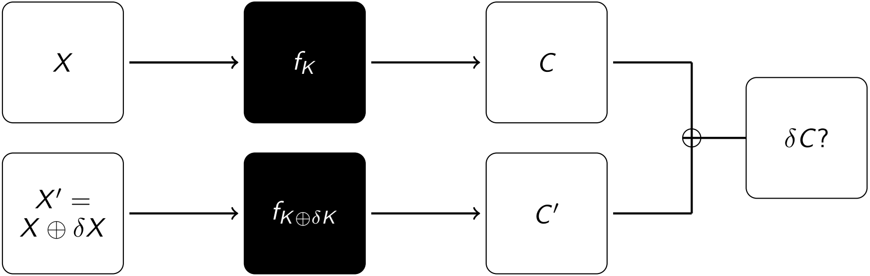
The attacker can encrypt messages of his choice and tries to recover the hidden key K.

Related Key Model



- The attacker choses δK (but K remains hidden)
- Allowed by certain protocol/real life applications
- A block cipher should be secure in the related key model
- **The best published attacks against AES are related key**

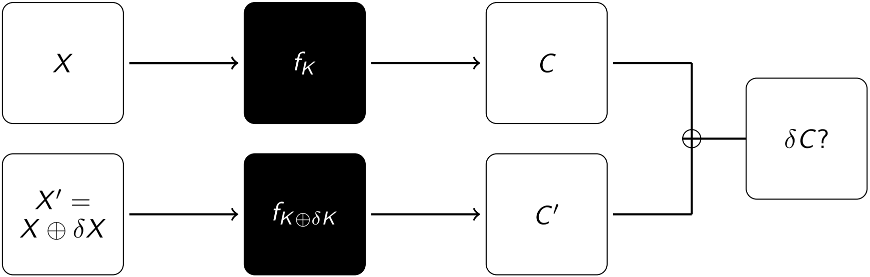
Related Key Attack



Distribution of δC for chosen $\delta X, \delta K$ and random X and K...

If $f = \pi$?
If $f = E$?

Related Key Attack



Distribution of δC for chosen $\delta X, \delta K$ and random X and K...

If $f = \pi$? Uniform
If $f = E$? Not uniform!

Distinguishing attack

The attacker requires many encryptions with input difference $\delta X, \delta K$ and observes whether there is a bias in the distribution of δC