Revisiting AES Related-Key Differential Attacks with Constraint Programming

D. Gerault[1], P. Lafourcade[1], **M. Minier**[2], C. Solnon[3]

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

Journées Méthodes Formelles - 30 May 2017

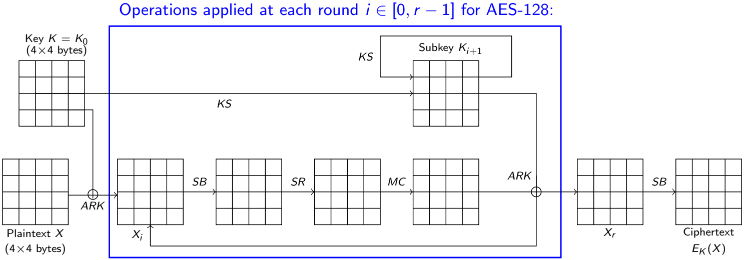
Revisiting AES RKD Attacks with CP

- **Differential cryptanalysis of the AES**
- First CP model for Step 1
- Second CP model for Step 1
- Third CP model for Step 1
- CP model for Step 2
- Results
- Conclusion

AES (Advanced Encryption Standard)

Block cipher standard since 2001

- Input:
  - A plaintext $X = 128$ bits $=$ 4x4 bytes
  - A key $K = 128$, 192, or 256 bits $=$ 4x4, 4x6, or 4x8 bytes
- Output: a ciphertext $E_K(X)$ such that $X = E_K^{-1}(E_K(X))$
- Iterative process of r rounds: $r = 10\ (12, 14)$ when $|K| = 128\ (192, 256)$

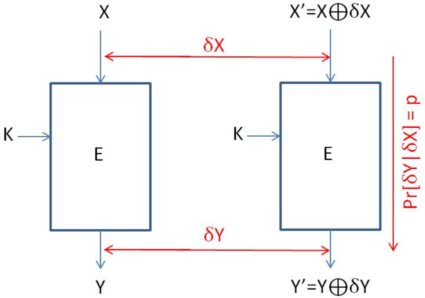


Cryptanalysis of the AES Block Cipher (1/2)

Differential Cryptanalysis [Biham and Shamir 1991]:

Track XOR differences through the ciphering process to recover the key:

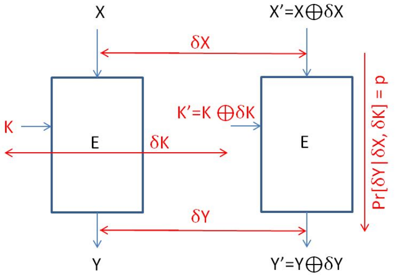
- Let $\delta X = X \oplus X'$ be an input plaintext difference
- Let $\delta Y = E_K(X) \oplus E_K(X')$ be the output difference
- The cipher is weak if $\exists\ \delta X$ and δY such that $Pr[\delta Y|\delta X] >> 2^{-|K|}$
  $\rightsquigarrow$ Key recovery in $\mathcal{O}(1/Pr[\delta Y|\delta X])$



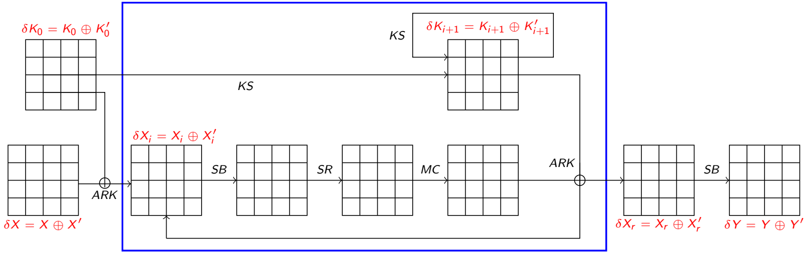
Cryptanalysis of the AES Block Cipher (2/2)

Related-Key Attack [Biham 1993]: Inject differences in texts and keys

- Let $\delta X = X \oplus X'$ be an input plaintext difference
- Let $\delta K = K \oplus K'$ be an input key difference
- Let $\delta Y = E_K(X) \oplus E_{K'}(X')$ be the output difference
- The cipher is weak if $\exists\ \delta X, \delta K$, and δY such that $Pr[\delta Y|\delta X, \delta K] >> 2^{-|K|}$
  $\rightsquigarrow$ Key recovery in $\mathcal{O}(1/Pr[\delta Y|\delta X, \delta K])$



Related-Key Differential of AES



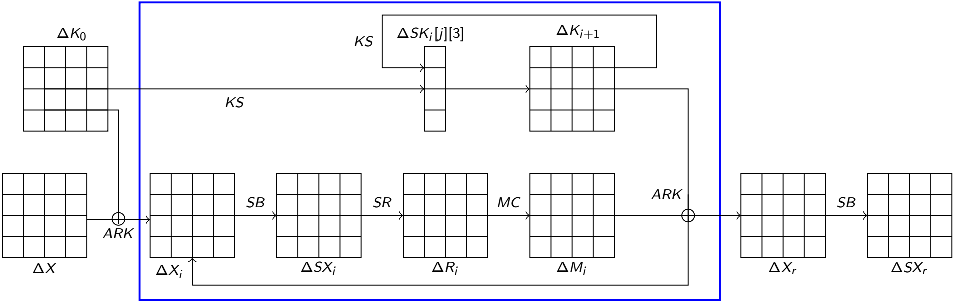
Goal: Find δX, δK_0, and δY that maximizes $Pr[\delta Y|\delta X, \delta K_0]$:

- ARK, SR, and MC are linear: $op(B_i) \oplus op(B_j) = op(B_i \oplus B_j)$
  $\leadsto$ Probabilities are equal to 1 (or 0) for these operators
- SB is not linear:
  - Let $Pr[\delta_o|\delta_i] = \frac{\#\{(B_1,B_2)\in[0,256]^2 \mid \delta_i = B_1 \oplus B_2 \text{ and } \delta_o = S(B_1)\oplus S(B_2)\}}{256}$
    $\leadsto$ Probability to have output difference δ_o given input difference δ_i
  - Perfect cipher: $\forall \delta_i, \delta_o, Pr[\delta_o|\delta_i] = \frac{1}{256}$... but this is impossible!
  - SB of AES: if $\delta_o = \delta_i = 0$ then $Pr[\delta_o|\delta_i] = 1$ else $Pr[\delta_o|\delta_i] \in \{0, \frac{2}{256}, \frac{4}{256}\}$

Two step solving process [Biryukov et al. 2010, Fouque et al. 2013]

Step 1: Asbtract differential bytes $\delta B = B \oplus B'$ to booleans ΔB

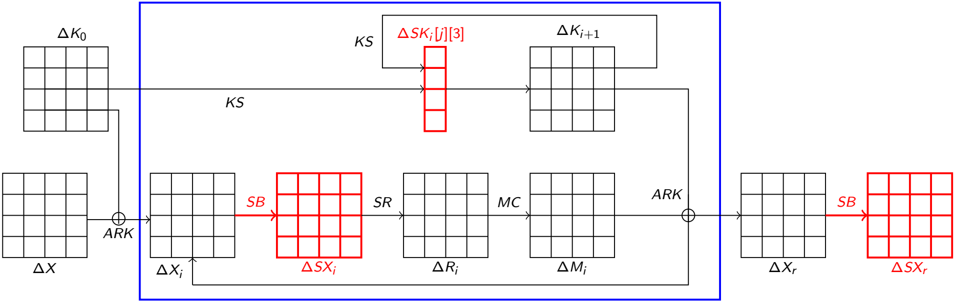
- For each differential byte δB: $\Delta B = 0$ if $\delta B = 0$; $\Delta B = 1$ if $\delta B \in [1, 255]$



Two step solving process [Biryukov et al. 2010, Fouque et al. 2013]

Step 1: Asbtract differential bytes $\delta B = B \oplus B'$ to booleans ΔB

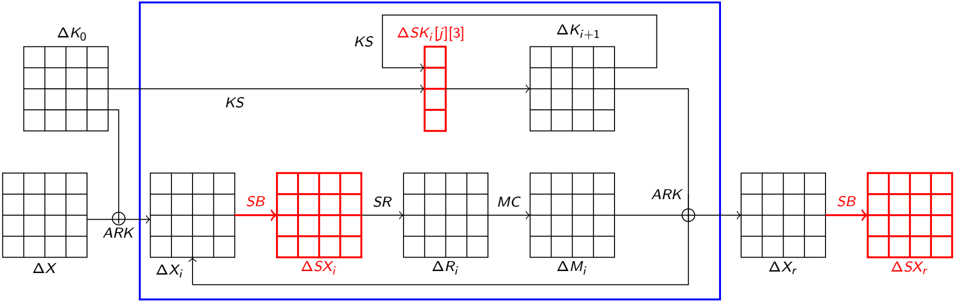
- For each differential byte δB: $\Delta B = 0$ if $\delta B = 0$; $\Delta B = 1$ if $\delta B \in [1, 255]$
- Minimize the nb of boolean variables $\Delta X_i[j][k]$ and $\Delta K_i[j][3]$ set to 1:
  - If $\delta X_i[j][k] = \delta SX_i[j][k] = 0$ then $Pr[\delta SX_i[j][k] | \delta X_i[j][k]] = 1$
  - Otherwise $Pr[\delta SX_i[j][k] | \delta X_i[j][k]] \in \{0, \frac{2}{256}, \frac{4}{256}\}$



Two step solving process [Biryukov et al. 2010, Fouque et al. 2013]

Step 2: Concretize booleans to differential bytes

- If $\Delta B = 0$ then set δB to 0; otherwise search for $\delta B \in [1, 255]$
  - If not possible: Solution byte-inconsistent
  - If possible: Solution byte-consistent
    $\rightsquigarrow$ Maximize the probability $Pr[\delta SX_r | \delta X, \delta K_0]$



Existing approaches

Biryukov et al. 2010:

$\rightsquigarrow$ Branch & Bound for Step 1

- $|K| = 128$: Several days of CPU time
- $|K| = 192$: Several weeks of CPU time

Fouque et al. 2013:

$\rightsquigarrow$ Graph traversal for Step 1

- $|K| = 128$: 30mn of CPU time (on 12 cores) but 60 GB of memory
- Not extended to $|K| = 192$ or 256

In both cases: Difficult and time-consuming programming work

$\rightsquigarrow$ Checking the correctness of the program is not straightforward...

What about Constraint Programming (CP)?

Solving a problem with CP:

- Define the problem with a declarative language:
  - Variables (unknowns) and their domains
  - Constraints (relations between variables)
  - Optionally: Objective function to optimize
- Use generic engines to search for solutions

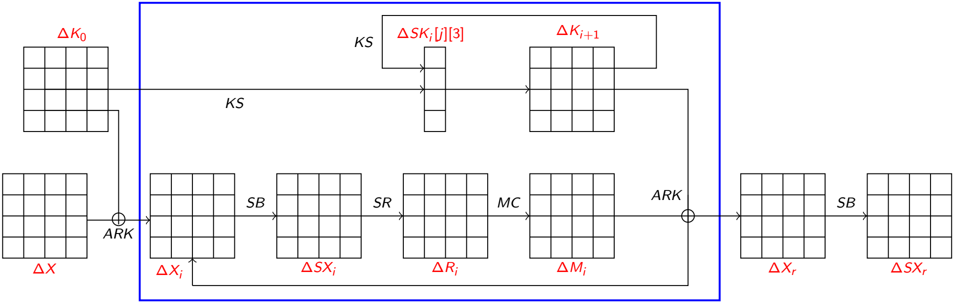
Using CP to compute related-key differentials:

- Step 1: Less than 35 hours for the hardest instance
- Step 2: Less than 6 minutes for the hardest instance
- Prove inconsistency of a solution proposed by Biryukov et al. 2010
- New related-key differentials:
  - $|K| = 128$: $p = 2^{-79}$ (instead of 2^{-81}) for 4 rounds
  - $|K| = 192$: $p = 2^{-176}$ for 10 rounds
  - $|K| = 256$: $p = 2^{-146}$ (instead of 2^{-154}) for 14 rounds

Revisiting AES RKD Attacks with CP

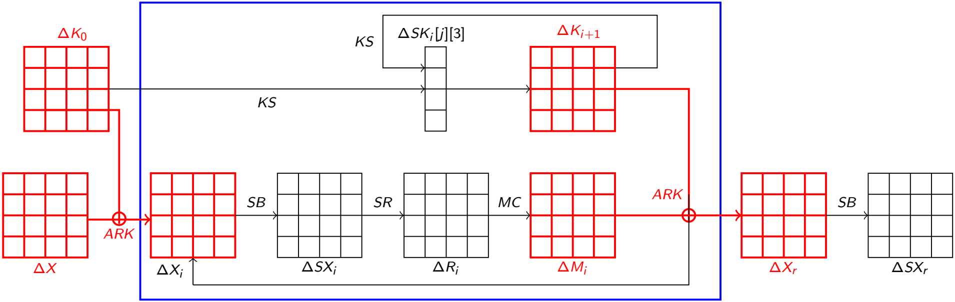
- Differential cryptanalysis of the AES
- **First CP model for Step 1**
- Second CP model for Step 1
- Third CP model for Step 1
- CP model for Step 2
- Results
- Conclusion

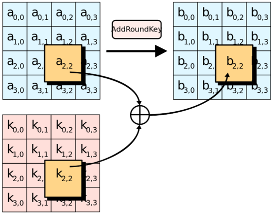
CP_{Basic}: First CP model for Step 1



- For each round i, for each row j and each column k: $\Delta X[j][k], \Delta X_i[j][k], \Delta SX_i[j][k], \Delta R_i[j][k], \Delta M_i[j][k], \Delta K_i[j][k], \Delta SK_i[j][3]$
- Boolean variables $\rightsquigarrow$ Domains $= \{0, 1\}$

CP_{Basic}: First CP model for Step 1





ARK performs XOR operations:

- $\forall j, k \in [0, 3] : XOR(\Delta X[j][k], \Delta K_0[j][k], \Delta X_0[j][k])$
- $\forall i \in [0, r-1], \forall j, k \in [0, 3] :$
  $XOR(\Delta M_i[j][k], \Delta K_{i+1}[j][k], \Delta X_{i+1}[j][k])$

CP_{Basic}: First CP model for Step 1

XOR at the byte level: $\delta B_1 \oplus \delta B_2 \oplus \delta B_3 = 0$

$$
\begin{array}{rlllll}
(\delta B_1, \delta B_2, \delta B_3) \in & \{(0, & 0, & 0)\} & & \\
\cup & \{(0, & x, & x) & \mid & x \in [1,255]\} \\
\cup & \{(x, & 0, & x) & \mid & x \in [1,255]\} \\
\cup & \{(x, & x, & 0) & \mid & x \in [1,255]\} \\
\cup & \{(x, & y, & z) & \mid & x, y, z \in [1,255], x \neq y \neq z\}
\end{array}
$$

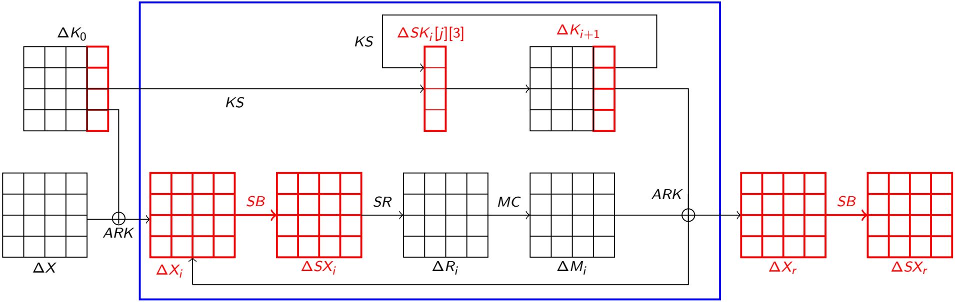
XOR at the boolean level:

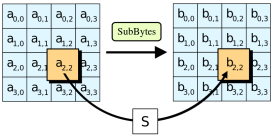
$$
\begin{array}{rlll}
(\Delta B_1, \Delta B_2, \Delta B_3) \in \{ & (0, & 0, & 0), \\
& (0, & 1, & 1), \\
& (1, & 0, & 1), \\
& (1, & 1, & 0), \\
& (1, & 1, & 1)\}
\end{array}
$$

Definition of the $XOR(\Delta B_1, \Delta B_2, \Delta B_3)$ constraint:

$$\Delta B_1 + \Delta B_2 + \Delta B_3 \neq 1$$

CP_{Basic}: First CP model for Step 1

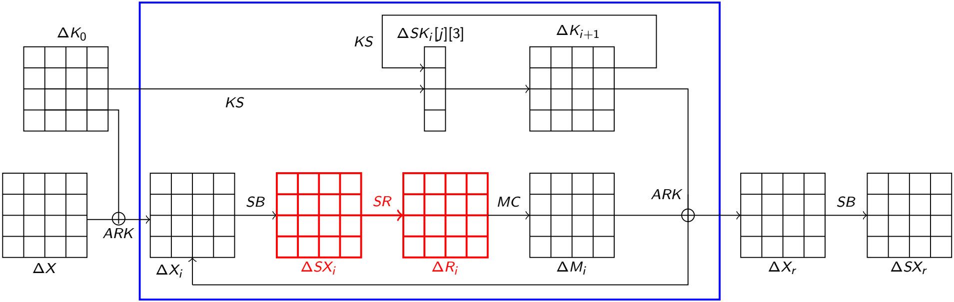


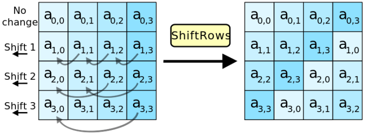


SubBytes does not introduce nor remove differences (because $B_i \oplus B_j = 0 \Leftrightarrow S(B_i) \oplus S(B_j) = 0$)

- $\forall i \in [0, r], \forall j, k \in [0, 3]$: $\Delta X_i[j][k] = \Delta SX_i[j][k]$
- $\forall i \in [0, r], \forall j \in [0, 3]$: $\Delta K_i[j][3] = \Delta SK_i[j][3]$

CP_{Basic}: First CP model for Step 1

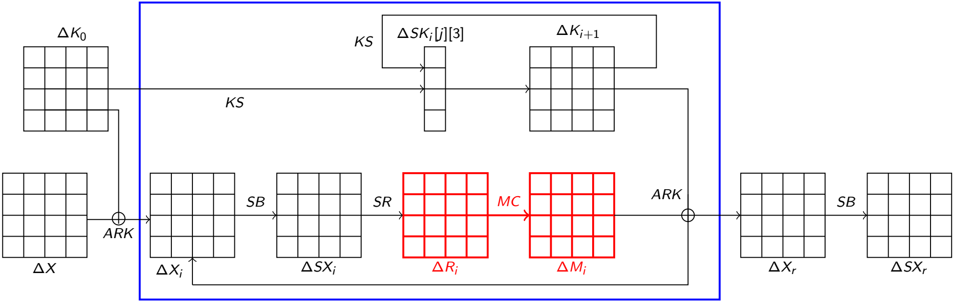


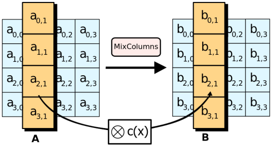


SR shifts bytes: $\forall i \in [0, r-1], \forall j, k \in [0, 3]$:

$$\Delta R_i[j][k] = \Delta SX_i[j][k + j\%4]$$

CP_{Basic}: First CP model for Step 1

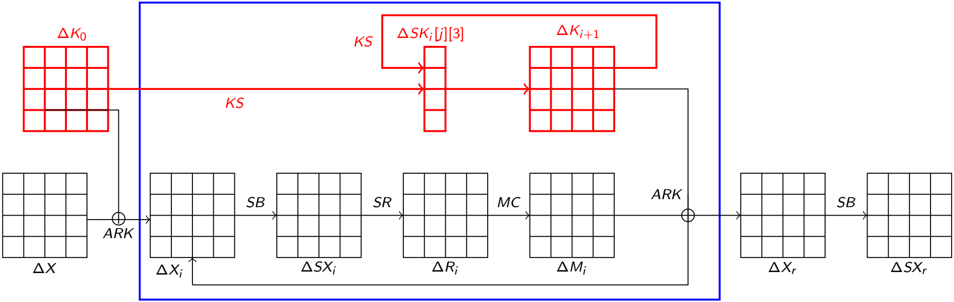


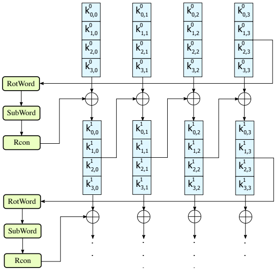


- MC multiplies each column by a fixed matrix
- Ensures the MDS property:
  $\forall i \in [0, r-1], \forall k \in [0, 3]$

$$\sum_{j=0}^{3} \Delta R_i[j][k] + \Delta M_i[j][k] \in \{0, 5, 6, 7, 8\}$$

CP_{Basic}: First CP model for Step 1

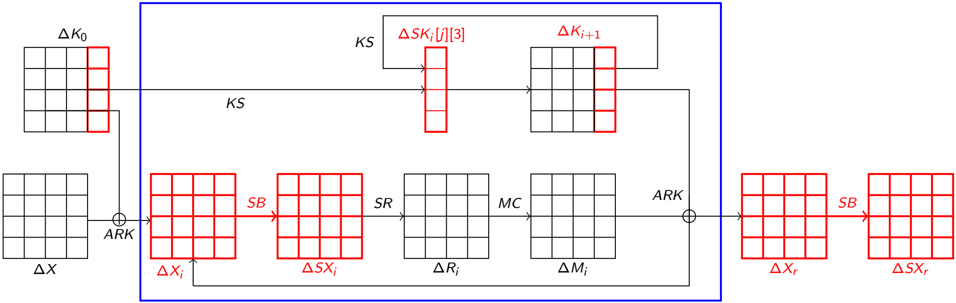




KS performs XOR, byte shifts, and SB operations
For AES-128: $\forall i \in [0, r-1], \forall j \in [0, 3]$:

- Column 0:
  $XOR(\Delta K_{i-1}[j][0], \Delta SK_{i-1}[(j+1)\%4][3], \Delta K_i[j][0])$
- Columns $k \in [1, 3]$:
  $XOR(\Delta K_{i-1}[j][k], \Delta K_i[j][k-1], \Delta K_i[j][k])$

CP_{Basic}: First CP model for Step 1



Goal: Minimize the number of differences that pass through SubBytes:

$$obj_{Step1} = \sum_{i=0}^{r-1}\sum_{j=0}^{3}(\Delta K_i[j][3] + \sum_{k=0}^{3}\Delta X_i[j][k])$$

Ordering heuristics:

- First choose variables that occur in the objective function

CP_{Basic}: First CP model for Step 1

r	obj_{Step1}	BCS	S	Gecode Time	Gecode CP	Choco 4 Time	Choco 4 CP	Chuffed Time	Chuffed CP
3	2	0	0	**0.0**	$9e^1$	**0.0**	$4e^1$	**0.0**	$5e^1$
3	3	0	$5e^2$	0.1	$2e^3$	0.4	$2e^3$	**0.0**	$7e^2$
3	4	0	$5e^3$	1.3	$2e^4$	1.8	$1e^4$	**0.2**	$5e^3$
3	5	2	$2e^4$	6.0	$6e^4$	5.1	$5e^4$	**0.9**	$2e^4$
4	8	0	0	**0.2**	$2e^4$	0.6	$1e^4$	0.3	$8e^3$
4	9	0	$2e^4$	7.1	$1e^5$	5.4	$7e^4$	**1.4**	$4e^4$
4	10	0	$6e^6$	-	-	1161.2	$2e^7$	**113.5**	$6e^6$
4	11	0	$9e^7$	-	-	-	-	**1974.5**	$9e^7$
4	12	2	-	-	-	-	-	-	-

- r = Nb rounds
- obj_{Step1} = Nb of differences that pass through SB (active S-boxes)
- BCS = Number of byte-consistent solutions
  $\leadsto$ Boolean solutions that can be concretized to byte solutions
- S = Number of solutions $\leadsto$ Most NOT byte-consistent
- CP = number of choice points in the search tree
  $\leadsto$ Chuffed explores less choice points and is faster

Revisiting AES RKD Attacks with CP

- Differential cryptanalysis of the AES
- First CP model for Step 1
- **Second CP model for Step 1**
- Third CP model for Step 1
- CP model for Step 2
- Results
- Conclusion

CP_{EQ}: Second CP model for Step 1

What's wrong with CP_{Basic}?

XOR constraints do not propagate equality relationships at the byte level

- For example, if $\delta a \oplus \delta b \oplus \delta c = 0$ and $\delta a \oplus \delta b \oplus \delta d = 0$ then $\delta c = \delta d$
- However, at the boolean level, we only propagate:
  $\Delta A + \Delta B + \Delta C \neq 1$ and $\Delta A + \Delta B + \Delta D \neq 1$

New variables and constraints to model byte equalities:

- For each couple of differential bytes $(\delta A, \delta B)$:
  - $EQ_{\delta A, \delta B} = 1$ if $\delta A = \delta B$
  - $EQ_{\delta A, \delta B} = 0$ if $\delta A \neq \delta B$
- Symmetry: $EQ_{\delta A, \delta B} = EQ_{\delta B, \delta A}$
- Transitivity: $EQ_{\delta A, \delta B} = EQ_{\delta B, \delta C} = 1 \Rightarrow EQ_{\delta A, \delta C} = 1$
- Relation with Δ variables:
  - $EQ_{\delta A, \delta B} = 1 \Rightarrow \Delta A = \Delta B$
  - $EQ_{\delta A, \delta B} = 0 \Rightarrow \Delta A + \Delta B \neq 0$

CP_{EQ}: Second CP model for Step 1

Definition of XOR in CP_{Basic}: $\Delta B_1 + \Delta B_2 + \Delta B_3 \neq 1$

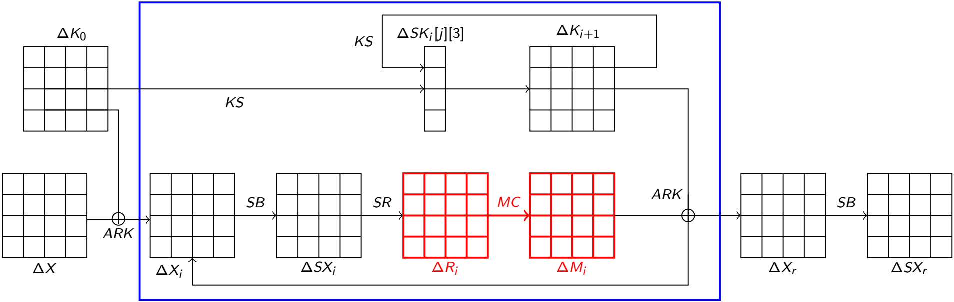
Can we strengthen it by exploiting byte equalities?
Yes, because:

- $\Delta B_1 = 0 \Leftrightarrow \delta B_2 = \delta B_3$
- $\Delta B_2 = 0 \Leftrightarrow \delta B_1 = \delta B_3$
- $\Delta B_3 = 0 \Leftrightarrow \delta B_1 = \delta B_2$

New definition of XOR:

$$
\begin{array}{rl}
\texttt{XOR}(\Delta B_1, \Delta B_2, \Delta B_3) \Leftrightarrow & ((\Delta B_1 + \Delta B_2 + \Delta B_3 \neq 1) \\
\wedge & (EQ_{\delta B_1, \delta B_2} = 1 - \Delta B_3) \\
\wedge & (EQ_{\delta B_1, \delta B_3} = 1 - \Delta B_2) \\
\wedge & (EQ_{\delta B_2, \delta B_3} = 1 - \Delta B_1))
\end{array}
$$

CP_{EQ}: Second CP model for Step 1



MDS also holds when XORing different columns of δR and δM:

$\forall i_1, i_2 \in [0, r-1], \forall k_1, k_2 \in [0,3]$, the number of bytes equal to 0 in

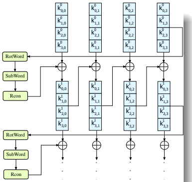
$$\delta R_{i_1}[j][k_1] \oplus \delta R_{i_2}[j][k_2] \text{ and } \delta M_{i_1}[j][k_1] \oplus \delta M_{i_2}[j][k_2] \in \{0,1,2,3,8\}$$

New constraints to ensure MDS: $\forall i_1, i_2 \in [0, r-1], \forall k_1, k_2 \in [0,3]$

$$\sum_{j=0}^{3} EQ_{\delta R_{i_1}[j][k_1], \delta R_{i_2}[j][k_2]} + EQ_{\delta M_{i_1}[j][k_1], \delta M_{i_2}[j][k_2]} \in \{0,1,2,3,8\}$$

CP_{EQ}: Second CP model for Step 1

KS (mainly) performs XOR operations:



- Column 0: $K_i[j][0] = K_{i-1}[j][0] \oplus SK_{i-1}[(j+1)\%4][3]$
- Columns $k \in [1,3]$: $K_i[j][k] = K_i[j][k-1] \oplus K_{i-1}[j][k]$

$\rightsquigarrow$ Each byte of K_i is eq. to a XOR of bytes of K_0 and SK_{i-1}

Ex: $K_2[1][1] = K_2[1][0] \oplus K_1[1][1]$
$= K_1[1][0] \oplus SK_1[2][3] \oplus K_1[1][0] \oplus K_0[1][1] = SK_1[2][3] \oplus K_0[1][1]$

New constraints:

- Pre-compute sets $V_{i,j,k}$ such that $\delta K_i[j][k] = \bigoplus_{\delta B \in V_{i,j,k}} \delta B$
- Introduce set variables $S_{i,j,k}$ and post the following constraints:
  - $S_{i,j,k} = \{\delta B \in V_{i,j,k} | \Delta B = 1\}$
  - If $S_{i,j,k} = \emptyset$ then $\Delta K_i[j][k] = 0$
  - If $S_{i,j,k} = \{\delta B\}$ then $EQ_{\delta K_i[j][k], \delta B} = 1$
  - If $S_{i,j,k} = \{\delta B_1, \delta B_2\}$ then $XOR(\Delta B_1, \Delta B_2, \Delta K_i[j][k])$
  - If $\exists i', j', k'$ s.t. $S_{i,j,k} = S_{i',j',k'}$ then $EQ_{\delta K_i[j][k], \delta K_{i'}[j'][k']} = 1$

Revisiting AES RKD Attacks with CP

- Differential cryptanalysis of the AES
- First CP model for Step 1
- Second CP model for Step 1
- **Third CP model for Step 1**
- CP model for Step 2
- Results
- Conclusion

CP_{Class}: Third CP model for Step 1

Alternative way to model equivalence classes:

- For each boolean variable ΔB, define an integer variable $Class_{\delta B}$
  $\rightsquigarrow D(Class_{\delta B}) = [0, 255]$
- Constraints:
  - $(\Delta B = 0) \Leftrightarrow (Class_{\delta B} = 0)$
  - Global *precede* constraint to break symmetries
  - Update all constraints:
    $\rightsquigarrow$ replace $EQ_{\delta B_1, \delta B_2} = 1$ by $Class_{\delta B_1} = Class_{\delta B_2}$
    $\rightsquigarrow$ replace $EQ_{\delta B_1, \delta B_2} = 0$ by $Class_{\delta B_1} \neq Class_{\delta B_2}$

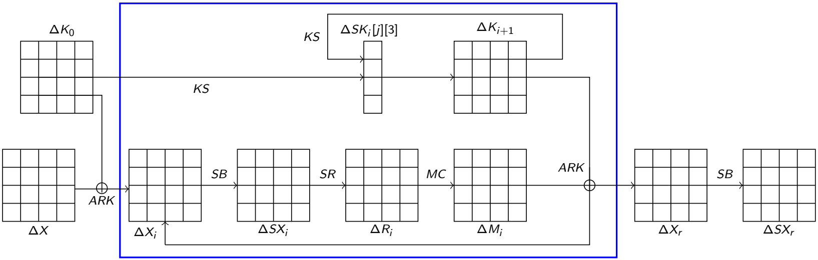
Revisiting AES RKD Attacks with CP

- Differential cryptanalysis of the AES
- First CP model for Step 1
- Second CP model for Step 1
- Third CP model for Step 1
- **CP model for Step 2**
- Results
- Conclusion

CP model for Step 2

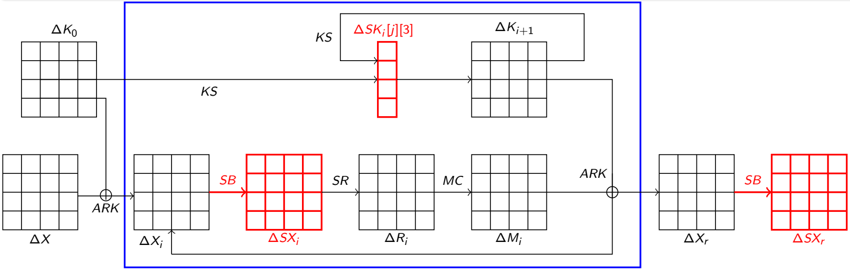
1. Initialize Obj_{Step1} to 1
2. Step 1: Search for all boolean solutions
3. For each boolean solution of Step 1:
   - Step 2: Search for byte values that maximize $Pr[\delta SX_r|\delta X, \delta K_0]$ (or detect inconsistency and set Pr to 0)

   $\leadsto$ Let Pr_{max} be the largest probability wrt all boolean solutions of Step 1
4. If $Pr_{max} < 2^{-6(Obj_{Step1}+1)}$ then increment Obj_{Step1} and go to (2)
   Otherwise, return Pr_{max}



CP model for Step 2

- For each boolean variable ΔB: Integer variable δB
  - If $\Delta B = 0$ in the Step 1 solution then: $D(\delta B) = \{0\}$
  - Otherwise: $D(\delta B) = [1, 255]$
- For each byte A on which SB is applied: Integer variable P_A
  $\rightsquigarrow$ Base 2 logarithm of $\Pr(\delta SA|\delta A)$
  - If $\Delta A = \Delta SA = 0$ then: $D(P_A) = \{0\}$ because $\Pr(0|0) = 1$
  - Otherwise: $D(P_A) = \{-7, -6\}$ because $Pr(\delta SA|\delta A) \in \{\frac{2}{256}, \frac{4}{256}\}$
- Objective function: Maximize $obj_{Step2} = \sum_{A \text{ on which SB is applied}} P_A$



CP model for Step 2

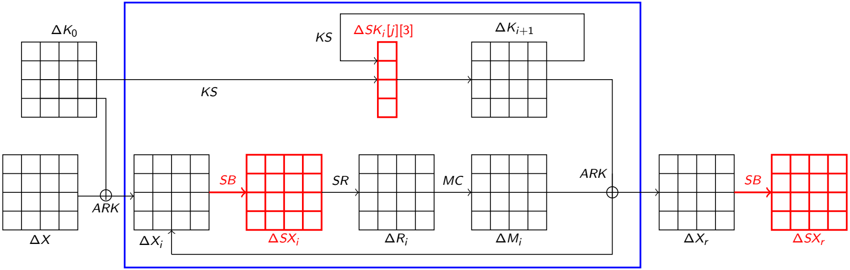
Table constraint related to SB:

For each byte A on which SB is applied:

$$(\delta A, \delta SA, P_A) \in \{(X, Y, P) | \quad \exists (B_1, B_2) \in [0, 255] \times [0, 255], X = B_1 \oplus B_2, \\ Y = S(B_1) \oplus S(B_2), P = \log_2(\Pr(Y|X))\}$$

Constraints related to KS, ARK, SR, and MC:

$\rightsquigarrow$ Straightforward definition with table constraints



Revisiting AES RKD Attacks with CP

- Differential cryptanalysis of the AES
- First CP model for Step 1
- Second CP model for Step 1
- Third CP model for Step 1
- CP model for Step 2
- **Results**
- Conclusion

Experimental setup

Languages and Solvers:

- CP models for Step 1 are defined in MiniZinc
  $\rightsquigarrow$ Benchmark for the 2016 MiniZinc Challenge
  $\rightsquigarrow$ Best results are obtained with Chuffed and Picat
- The CP model for Step 2 is defined in Choco 4 (Java CP library)

CPU time limit for Step 1:

- 24 hours on a 3.5 GHz i7-4710MQ processor with 8 gigabytes of memory
- If not solved in 24 hours: Decompose into independent subproblems

Scale-up properties: AES-128

	Step 1				Step 2		
r	obj_{Step1}	time CP_{EQ}	time CP_{Class}	# boolean solutions	# byte-cons. solutions	time	Pr
3	5	1	1	4	2	3	2^{-31}
4	12	114	81	8	2	5	2^{-79}
5	17	2799	2049	1236	97	248	2^{-105}
6	21	?	?	1542	20	?	2^{-131}

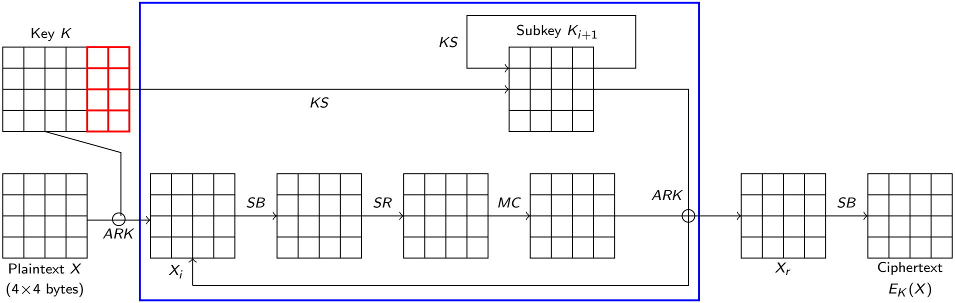
$\rightsquigarrow$ Useless to go on with $r > 6$ because $2^{-131} < 2^{-128}$

Recall of the results of CP_{Basic} for Step 1:

- $r = 3, obj_{Step1} = 5 : 2e^4$ boolean solutions
- $r = 4, obj_{Step1} = 11 : 9e^7$ boolean solutions

$\rightsquigarrow$ Most byte-inconsistent sol. are removed by new constraints at byte level

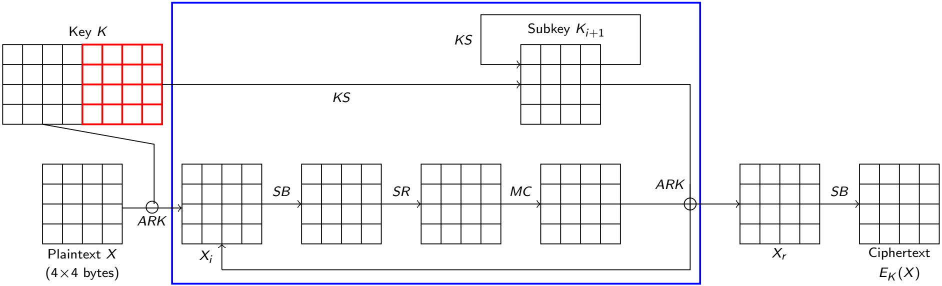
Extension to AES-192 and AES-256



Update constraints related to KeySchedule:

- Step 1: XOR constraints combined with byte shifts
- Step 2: XOR constraints combined with byte shifts
  + SubBytes on some columns

Extension to AES-192 and AES-256



Update constraints related to KeySchedule:

- Step 1: XOR constraints combined with byte shifts
- Step 2: XOR constraints combined with byte shifts
  + SubBytes on some columns

Scale-up properties: AES-192

	Step 1				Step 2		
r	obj_{Step1}	time CP_{EQ}	time CP_{Class}	# boolean	# byte	time	Pr
3	1	0	0	15	15	1	2^{-6}
4	4	2	1	4	4	2	2^{-24}
5	5	12	6	2	2	4	2^{-30}
6	10	573	298	6	6	4	2^{-60}
7	11	2108	988	2	0	8	
7	13	5833	2999	7	4	8	2^{-78}
8	16	44436	24619	4	0	19	
8	18	-	81073	19	9	19	2^{-108}
9	24	-	-	240	13	36	2^{-146}
10	29	-	-	29678	34	180	2^{-176}
11	30	-	-	92	0	93	
11	31	-	-	2436	0	110	

$\rightsquigarrow$ Useless to go on with $r > 11$ or $obj_{Step1} > 31$ because $6 * 32 = 192$

Scale-up properties: AES-256

	Step 1				Step 2		
r	obj_{Step1}	time CP_{EQ}	time CP_{Class}	# boolean solutions	# byte-cons. solutions	time	Pr
3	1	0	0	33	33	1	2^{-6}
4	3	0	0	14	14	3	2^{-18}
5	3	2	1	4	4	4	2^{-18}
6	5	30	16	3	3	10	2^{-30}
7	5	58	34	1	1	14	2^{-30}
8	10	2894	1722	3	1	25	2^{-60}
9	14	40064	21972	2	0	12	
9	15	85465	49138	16	16	60	2^{-92}
10	16	-	-	4	4	106	2^{-98}
11	20	-	-	4	4	115	2^{-122}
12	20	-	-	4	4	238	2^{-122}
13	24	-	-	4	4	326	2^{-146}
14	24	-	-	4	4	150	2^{-146}

Decomposition of hard instances into independent subproblems

For each round i: Integer variables $SumX_i$ and $SumK_i$

$\rightsquigarrow$ Number of differences in δX_i: $SumX_i = \sum_{j,k \in [0,3]} \Delta X_i[j][k]$
$\rightsquigarrow$ Number of differences in δK_i: $SumK_i = \sum_{j \in [0,3]} \Delta K_i[j][3]$

- One subproblem for each possible value of $SumX_i$ and $SumK_i$
- All subproblems are independent and may be solved in parallel

Combine Picat and Chuffed to solve the subproblems

- Use Picat to enumerate $SumX_i$ and $SumK_i$ with solutions (if any)
- Use Chuffed to enumerate all solutions, given $SumX_i$ and $SumK_i$ given by Picat

Time to solve the hardest instance

$\rightsquigarrow$ AES-192 with $r = 10$ and $obj_{Step1} = 29$

- Less than 24 hours for each subproblem and 35 hours for the complete sum

Revisiting AES RKD Attacks with CP

Conclusion

Better related-key differential Characteristic

- For AES-128, For 4 rounds, our solution (proved optimal): $\rightsquigarrow obj_{Step1} = 12$ and $Pr = 2^{-79}$ (before $obj_{Step1} = 13$ and $Pr = 2^{-81}$)
- For AES-192, For 10 rounds (not 11) best related-key differential trail: $\rightsquigarrow obj_{Step1} = 29$ and $Pr = 2^{-176}$
- For AES-256, For 14 rounds, our solution (proved optimal): $\rightsquigarrow obj_{Step1} = 24$ and $Pr = 2^{-146}$ (before $Pr = 2^{-154}$)

Declarative framework for Cryptanalysis?

CP models describe problems, not how to solve them:

- Easier to define and check than a full program
  $\rightsquigarrow$ Better solutions than [Biryukov et al 2009] and [Fouque et al 2013]
- Models are defined with the MiniZinc language:
  $\rightsquigarrow$ We can use different CP solvers to solve them

Thanks for Your Attention !

Questions ?