

Gaussian Cryptanalysis of Hash Functions: Collisions and Distinguishers

Dmitry Khovratovich, Alex Biryukov, Ivica Nikolic

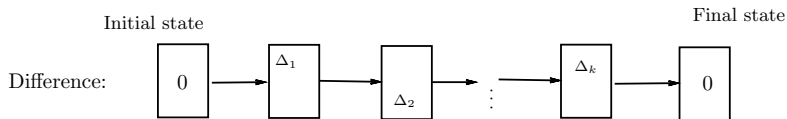
University of Luxembourg

16 January 2009

Problem

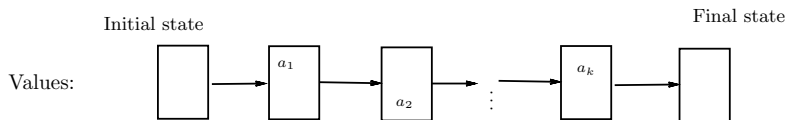
Differential trail for the collision search

$$H(M) = H(M').$$

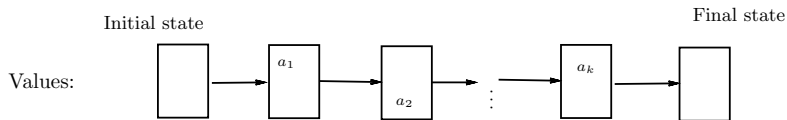


Sufficient condition for the difference propagation — fixed input:

$$f(a + \Delta_1) - f(a) = \Delta_2$$



System of equations



$$f(u, v, x) = y;$$

$$l(v, x, y) = z;$$

$$h(x, y, z) = a_1;$$

...

$$f(z, t, q) = t;$$

$$g(x, y, t) = a_2;$$

...

Solution

How to solve?

- Exhaustive search :);
- SAT-solvers;
- Groebner basis.

They may be too slow...

Idea: Gauss!

We know that the Gaussian elimination is fast.
But for nonlinear equations?

$$\begin{aligned}f(x \oplus y \oplus s) &= z; \\s \oplus z \oplus t \oplus x &= 0; \\g(t \oplus s) &= x.\end{aligned}$$

Idea: Gauss!

We know that the Gaussian elimination is fast.
 But for nonlinear equations?

$$\begin{array}{l}
 f(x \oplus y \oplus s) = z; \\
 s \oplus z \oplus t \oplus x = 0; \\
 g(t \oplus s) = x.
 \end{array}
 \quad \Leftrightarrow \quad
 \begin{array}{l}
 y \oplus f^{-1}(z) \oplus x \oplus s = 0; \\
 s \oplus z \oplus t \oplus x = 0; \\
 g(t \oplus s) = x.
 \end{array}$$

Idea: Gauss!

We know that the Gaussian elimination is fast.
But for nonlinear equations?

$$\begin{array}{l}
 f(x \oplus y \oplus s) = z; \\
 s \oplus z \oplus t \oplus x = 0; \\
 g(t \oplus s) = x.
 \end{array}
 \Leftrightarrow
 \begin{array}{l}
 y \oplus f^{-1}(z) \oplus x \oplus s = 0; \\
 s \oplus z \oplus t \oplus x = 0; \\
 g(t \oplus s) = x.
 \end{array}
 \Leftrightarrow
 \begin{array}{l}
 y \oplus f^{-1}(z) \oplus x \oplus s = 0; \\
 z \oplus x \oplus t \oplus s = 0; \\
 g(t \oplus s) = x.
 \end{array}$$

Idea: Gauss!

We know that the Gaussian elimination is fast.
But for nonlinear equations?

$$\begin{aligned}
 f(x \oplus y \oplus s) &= z; & y \oplus f^{-1}(z) \oplus x \oplus s &= 0; \\
 s \oplus z \oplus t \oplus x &= 0; & s \oplus z \oplus t \oplus x &= 0; \\
 g(t \oplus s) &= x. & g(t \oplus s) &= x.
 \end{aligned}
 \Leftrightarrow$$

$$\begin{aligned}
 & y \oplus f^{-1}(z) \oplus x \oplus s = 0; \\
 \Leftrightarrow & z \oplus x \oplus t \oplus s = 0; \\
 & g(t \oplus s) = x.
 \end{aligned}
 \Leftrightarrow$$

$$\begin{aligned}
 & y \oplus f^{-1}(z) \oplus x \oplus s = 0; \\
 \Leftrightarrow & z \oplus x \oplus t \oplus s = 0; \\
 & x \oplus g(t \oplus s) = 0.
 \end{aligned}$$

Free variables

$$\begin{aligned}
 f(x \oplus y \oplus s) &= z; & y \oplus f^{-1}(z) \oplus x & \oplus s = 0; \\
 s \oplus z \oplus t \oplus x &= 0; & z \oplus x \oplus t & \oplus s = 0; \\
 g(t \oplus s) &= x. & x \oplus g(t \oplus s) &= 0.
 \end{aligned}
 \Leftrightarrow$$

Before

$$\begin{pmatrix}
 x & y & z & s & t \\
 \hline
 1 & 1 & 1 & 1 & 0 \\
 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 0 & 1 & 1
 \end{pmatrix}$$

After

$$\begin{pmatrix}
 y & z & x & s & t \\
 \hline
 1 & 1 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 & 1 \\
 0 & 0 & 1 & \underbrace{1 \quad 1}_{\text{free variables}}
 \end{pmatrix}$$

- Assign t and s randomly;
- Get other variables.

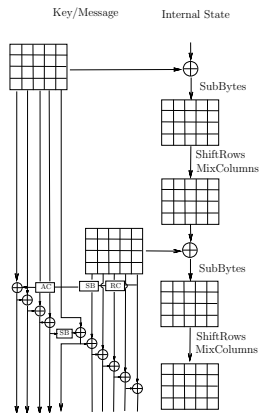
Bonus: You quickly get many solutions.

Applications

AES-based hash

Assume we want to make a compression function from pure AES.

- Use the Davies-Meyer mode;
- Extend the internal state to get a reasonable digest size;
- Extend the key size to get a reasonable speed;
- Example: 160-bit state, 320-bit message block.



AES-based hash

Equations:

Message additions and S-boxes:

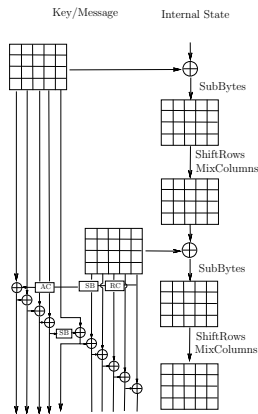
$$x_i \oplus m_i = S^{-1}(x_j).$$

MixColumns transformations:

$$MC^j(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}) = (y_{j_1}, y_{j_2}, y_{j_3}, y_{j_4}).$$

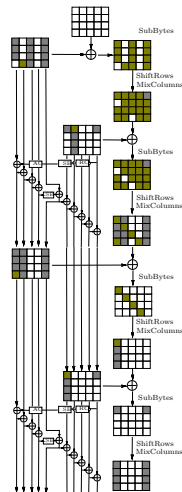
Message schedule:

$$m_i \oplus m_j = m_k.$$



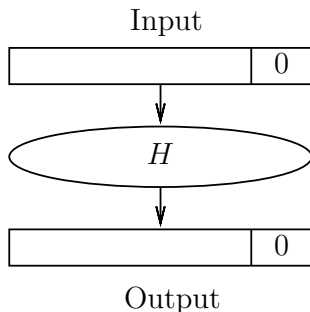
Semi-free-start collisions for an AES-based hash

- Differential collision trail for 12 rounds of AES-hash 320/160 (320-bit message, 160-bit internal state);
- Trail has 50 active S-boxes;
- We fix inputs of 41 of 50 active S-boxes and solve the resulting system;
- We get enough degrees of freedom to pass 9 remaining S-boxes.
- Will be presented at CT-RSA 2009.



Differentiability from the random oracle

Fix n bits of input and n bits of output.



- To find a solution for a random oracle one needs 2^n trials;
- For some compression functions we can generate solutions with cost 1.

MD6 compression function

f : 89 words \rightarrow 16 words.

Bit equations:

- Nonlinear: $x_{i-89}^j + x_{i-31}^j x_{i-67}^j + x_{i-18}^j x_{i-21}^j + x_{i-17}^j + x_i^j = 0$.
- Linear: $y_i^j = x_i^j + x_i^{j+l_i}$.

Number of equations = Number of steps \times Number of substep transformations \times Number of bits in the word.

$1664 \times 3 \times 64 \approx 300000$ equations for MD6-256.

Attack

f : 89 words \rightarrow 16 words.

We fix several bits in the input and the output of the compression function — and show how to derive the others.

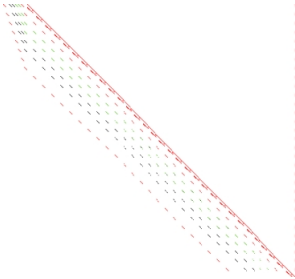
Rounds	Input bits fixed	Output bits fixed
22	9	9
26	6	6
30	2	2
32	2	2
33	1	1
80	MD6 – 160	
96	MD6 – 224	
104	MD6 – 256	

To be updated

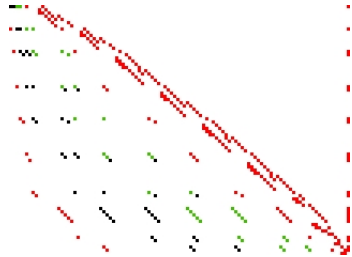
Real world

Reality is always worse than expected.

Starting matrix:



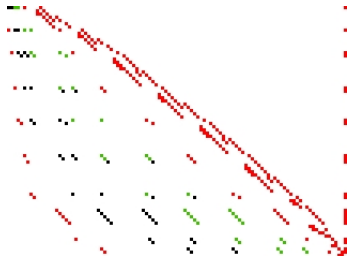
Matrix after triangulation:



The algorithm stops if every variable is involved in at least two equations.

What to do?

Matrix after triangulation:



Of course, heuristics!

- Guess one of variables;
- Do elimination on linear equations;
- Skip an equation and get the probability.

Memory requirement

- Store as a matrix is expensive after you have more than 4000 equations;
- We store as a graph: variables are nodes, equations are edges;
- Then $300 \cdot 10^3$ equations (full MD6) is not a problem.

Questions?