Practical Cryptanalysis of ARMADILLO-2

Thomas Peyrin
(joint work with María Naya-Plasencia)

Nanyang Technological University - Singapore

ASK 2012

Nagoya, Japan - August 29, 2012

UNIVERSITÉ DE VERSAILLES
SAINT-QUENTIN-EN-YVELINES

NANYANG
TECHNOLOGICAL
UNIVERSITY

Outline

Outline

The ARMADILLO-2 function

Free-start collision attack

Semi-free-start collision attack

Conclusion

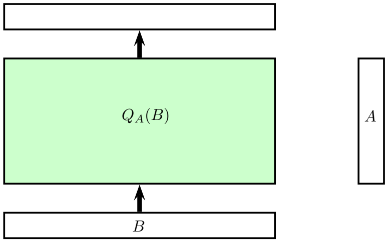
What is ARMADILLO-2 ?

- ARMADILLO-2 is a **lightweight**, **multi-purpose** cryptographic primitive published by Badel *et al.* at CHES 2010
- in the original article, ARMADILLO-1 is proposed but the authors identified a security issue and advised to use ARMADILLO-2
- ARMADILLO-2 is
  - a FIL-MAC
  - a stream-cipher
  - a hash function
- they are all based on an internal function that uses **data-dependent bit transpositions**
- 5 different parameters sizes defined

The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

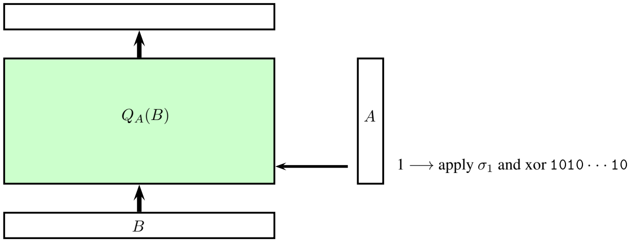
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010 ··· 10` to the internal state



The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

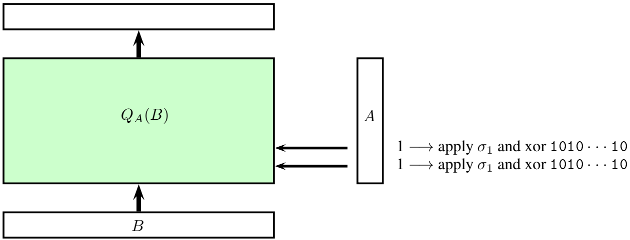
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010···10` to the internal state



The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

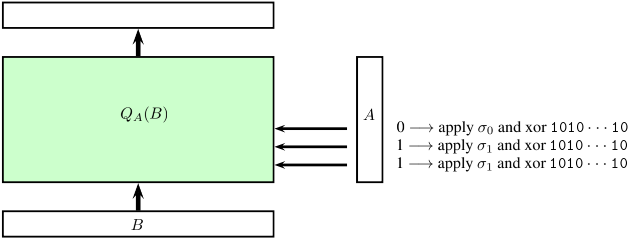
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** $1010\cdots10$ to the internal state



The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

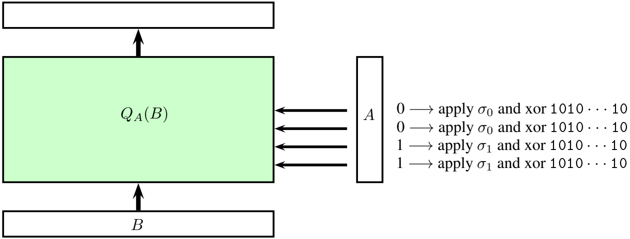
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010···10` to the internal state



The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

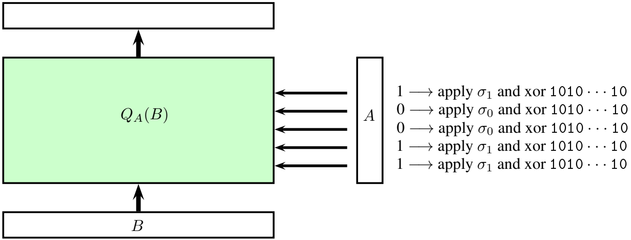
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010 ··· 10` to the internal state



The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

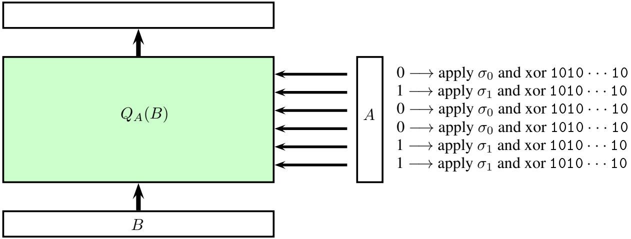
- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010 ··· 10` to the internal state



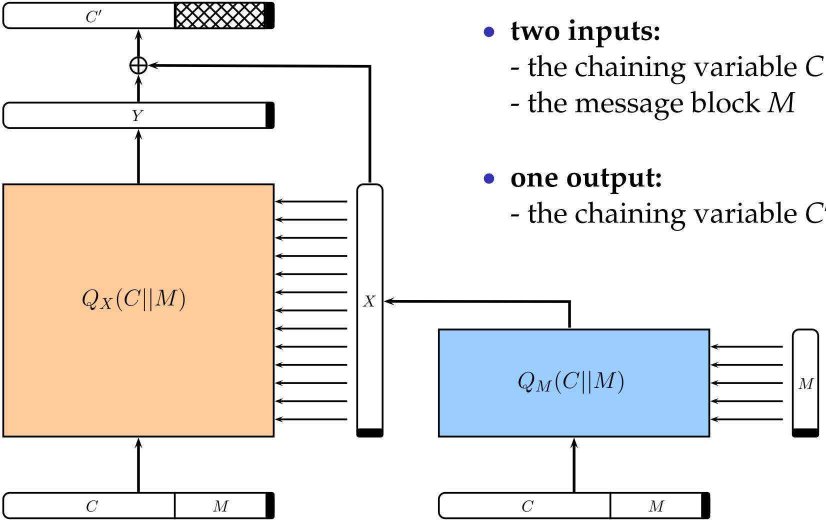
The basic building block: a parametrized permutation Q_X

ARMADILLO-2 uses a permutation $Q_A(B)$ as basic building block:

- the internal state is initialized with input B
  we apply a steps, where a is the bitsize of the input parameter A
- **for each step i:**
  - extract bit i from A
  - if A[i]=0, apply the **bitwise permutations** σ_0, otherwise σ_1
  - bitwise **XOR the constant** `1010 ··· 10` to the internal state

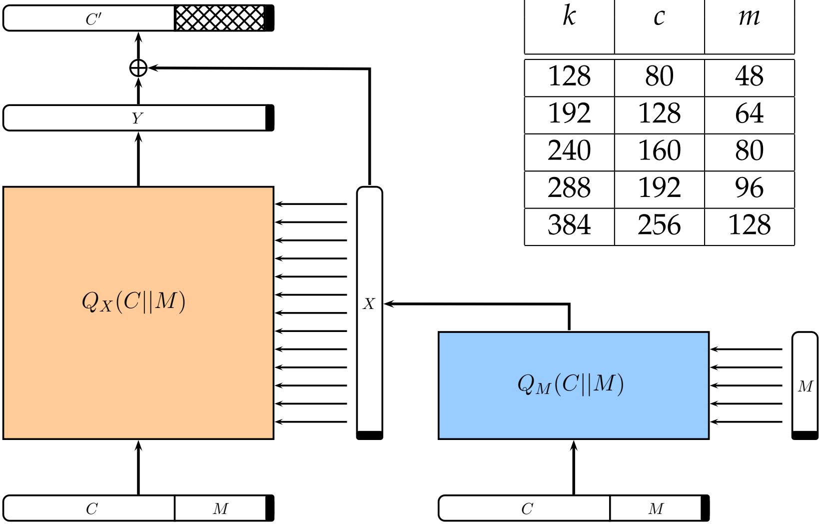


The ARMADILLO-2 compression function



- **two inputs:**
  - the chaining variable C
  - the message block M
- **one output:**
  - the chaining variable C'

The ARMADILLO−2 compression function



k	c	m
128	80	48
192	128	64
240	160	80
288	192	96
384	256	128

Cryptanalysis of ARMADILLO-2

Abdelraheem *et al.* (ASIACRYPT 2011):

- key recovery attack on the FIL-MAC
- key recovery attack on the stream cipher
- (second)-preimage attack on the hash function

... but **computation and memory complexity is very high**, often close to the generic complexity (example 256-bit preimage with 2^{208} computations and 2^{205} memory or 2^{249} computations and 2^{45} memory)

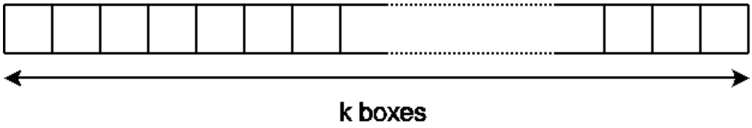
We provide **very practical attacks** (only a few operations):

- distinguisher and related-key recovery on the stream cipher
- free-start collision on the compression function (chosen-related IVs)
- semi-free-start collision on the compression/hash function (chosen IV)

First tools

For two random k-bit words A and B of Hamming weight a and b respectively, the probability that $\mathtt{HAM}(A \wedge B) = i$ is

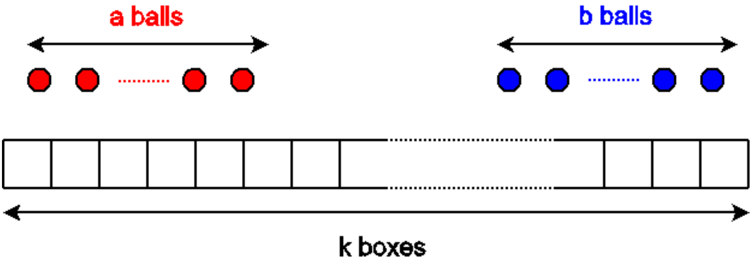
$$P_{\mathtt{and}}(k,a,b,i) = \frac{\binom{a}{i}\binom{k-a}{b-i}}{\binom{k}{b}} = \frac{\binom{b}{i}\binom{k-b}{a-i}}{\binom{k}{a}}.$$



First tools

For two random k-bit words A and B of Hamming weight a and b respectively, the probability that $\mathtt{HAM}(A \wedge B) = i$ is

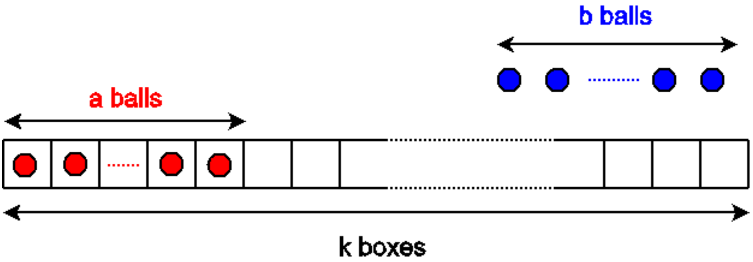
$$P_{\mathtt{and}}(k, a, b, i) = \frac{\binom{a}{i}\binom{k-a}{b-i}}{\binom{k}{b}} = \frac{\binom{b}{i}\binom{k-b}{a-i}}{\binom{k}{a}}.$$



First tools

For two random k-bit words A and B of Hamming weight a and b respectively, the probability that $\mathtt{HAM}(A \wedge B) = i$ is

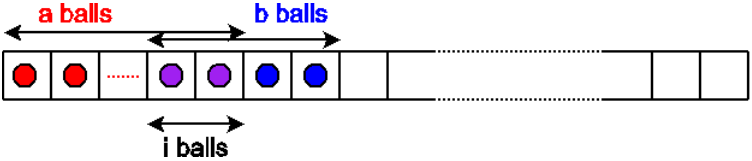
$$P_{\mathtt{and}}(k, a, b, i) = \frac{\binom{a}{i}\binom{k-a}{b-i}}{\binom{k}{b}} = \frac{\binom{b}{i}\binom{k-b}{a-i}}{\binom{k}{a}}.$$



First tools

For two random k-bit words A and B of Hamming weight a and b respectively, the probability that $\mathtt{HAM}(A \wedge B) = i$ is

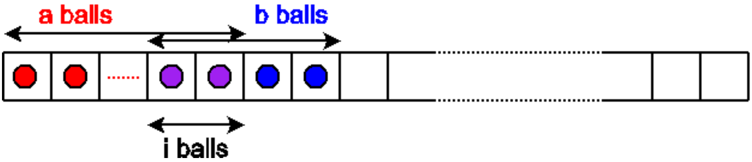
$$P_{\mathtt{and}}(k, a, b, i) = \frac{\binom{a}{i}\binom{k-a}{b-i}}{\binom{k}{b}} = \frac{\binom{b}{i}\binom{k-b}{a-i}}{\binom{k}{a}}.$$



First tools

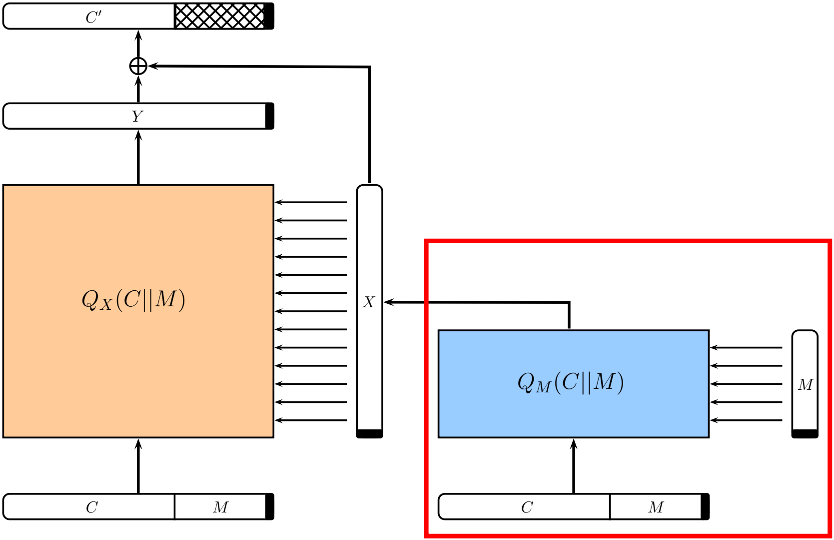
For two random k-bit words A and B of Hamming weight a and b respectively, the probability that $\texttt{HAM}(A \oplus B) = j$ is

$$P_{\texttt{xor}}(k,a,b,j) = \begin{cases} P_{\texttt{and}}(k,a,b,\frac{a+b-j}{2}) & \text{for } (a+b-j) \text{ even} \\ 0 & \text{for } (a+b-j) \text{ odd} \end{cases}$$

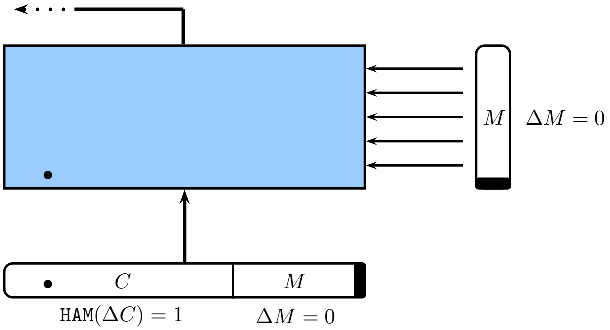


Outline

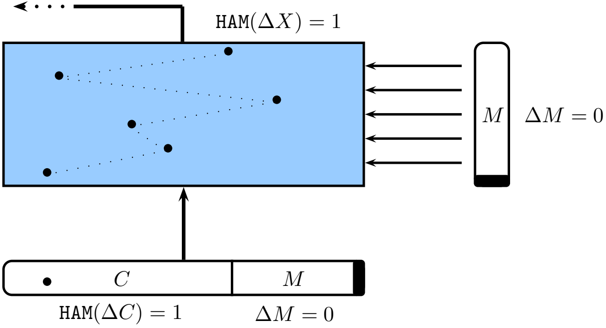
The differential path - right side



The differential path - right side

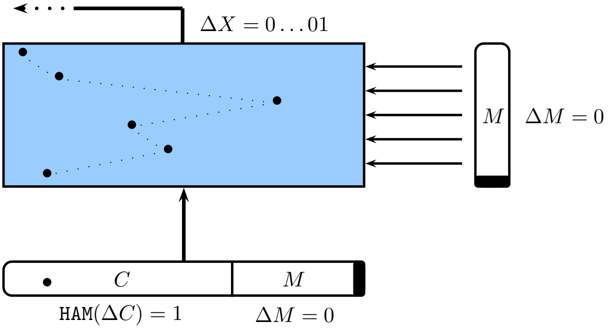


The differential path - right side



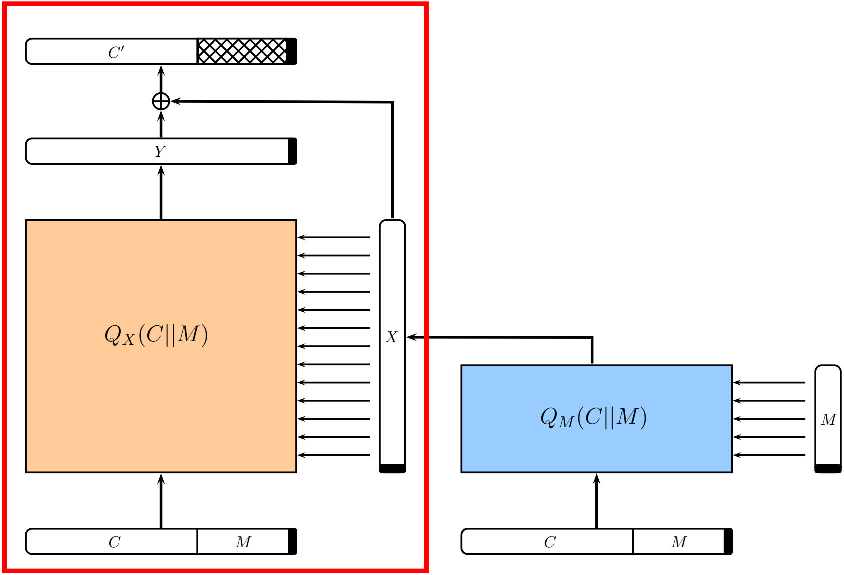
We have $\texttt{HAM}(\Delta X) = 1$ **with probability** 1

The differential path - right side

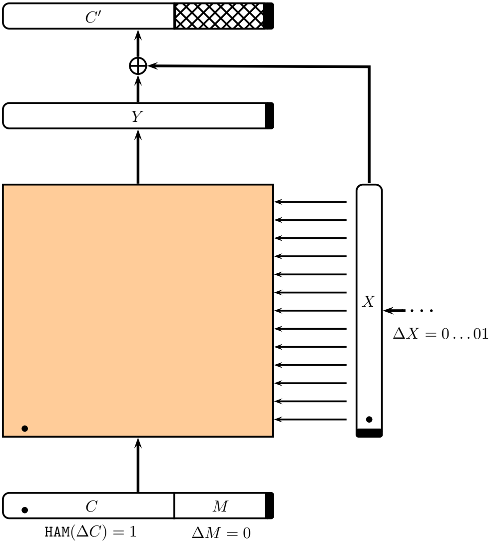


We have $\Delta X = 0\ldots01$ with probability $P_X = \frac{1}{k}$

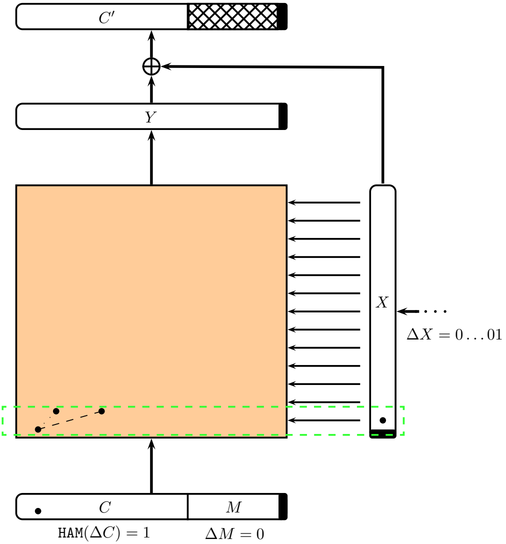
The differential path - left side



The differential path - left side



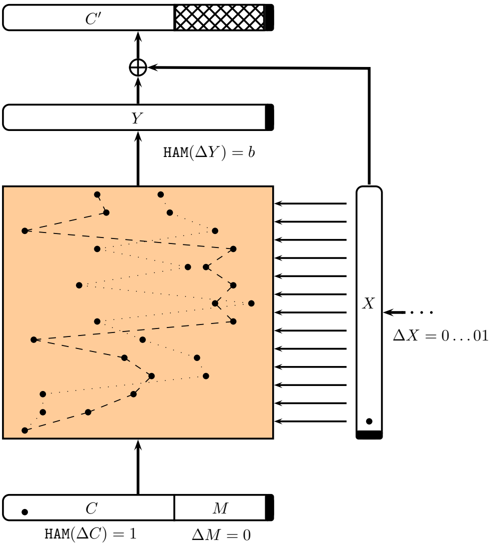
The differential path - left side



We have b active bits after first step with probability

$$P_{step}(b)$$

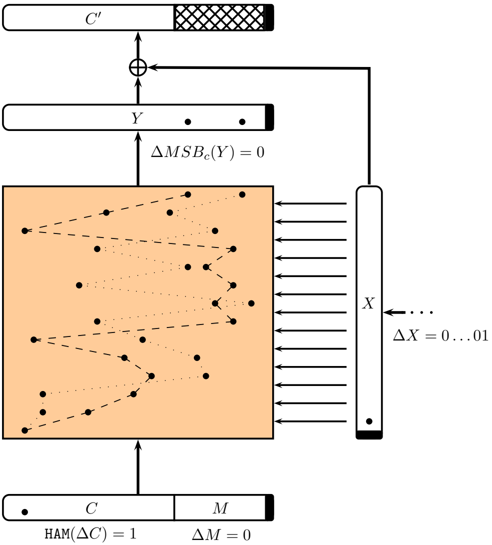
The differential path - left side



We have $\mathtt{HAM}(\Delta Y) = b$ with probability

$$P_{step}(b)$$

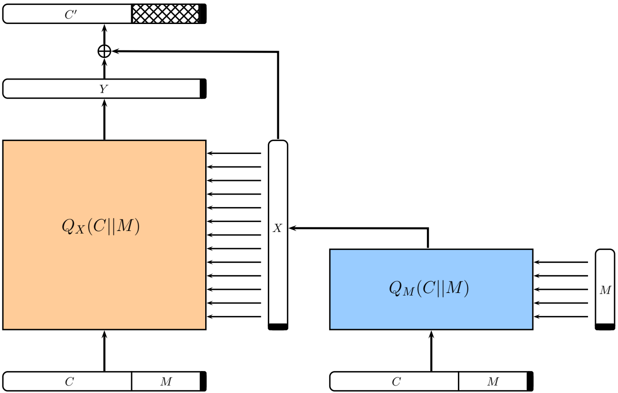
The differential path - left side



We have $\Delta MSB_c(Y) = 0$ with probability

$$
\begin{aligned}
& P_{step}(b) \cdot P_{out}(b) \\
= \quad & P_{step}(b) \cdot P_{\texttt{and}}(k, m, b, b) \\
= \quad & P_{step}(b) \cdot \prod_{i=0}^{i=b-1} \frac{m-i}{k-i}
\end{aligned}
$$

The differential path - overall differential probability



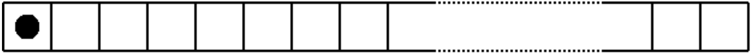
The overall collision probability is

$$P_X \cdot \sum_{i=1}^{i=m} P_{step}(i) \cdot P_{out}(i) = \frac{1}{k} \cdot \sum_{i=1}^{i=m} P_{step}(i) \cdot \prod_{i=0}^{i=b-1} \frac{m-i}{k-i}$$

The freedom degrees

For randomly chosen values of C and M, the collision probability will be too small:

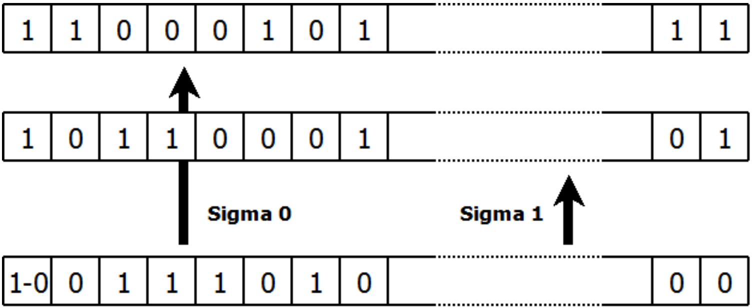
- we can choose b small, so that $P_{out}(b)$ is very high ...
- ... but $P_{step}(b)$ is very low anyway



The freedom degrees

For randomly chosen values of C and M, the collision probability will be too small:

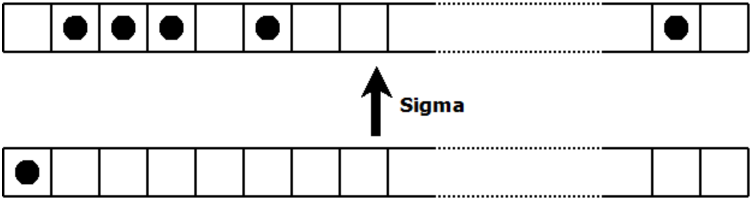
- we can choose b small, so that $P_{out}(b)$ is very high ...
- ... but $P_{step}(b)$ is very low anyway



The freedom degrees

For randomly chosen values of C and M, the collision probability will be too small:

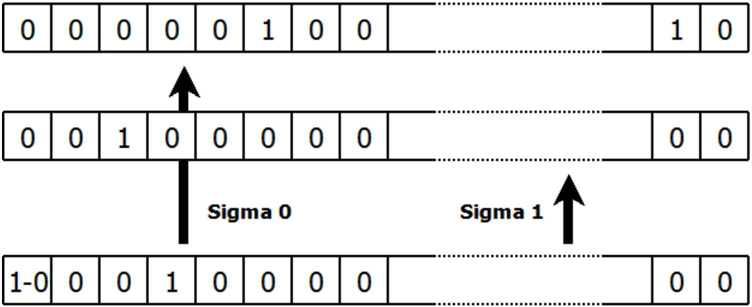
- we can choose b small, so that $P_{out}(b)$ is very high ...
- ... but $P_{step}(b)$ is very low anyway



The freedom degrees

However, we can use the freedom degrees:

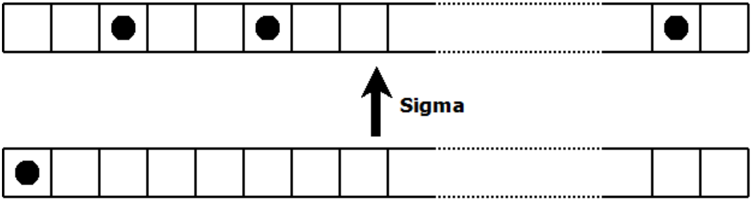
- by fixing the value of M and the difference position, one can first handle the right part of the differential path (Q_M)
- then by forcing the inputs value $(C||M)$ to have very low (or very high) Hamming weight *hw* it will be possible to have $P_{step}(b)$ high



The freedom degrees

However, we can use the freedom degrees:

- by fixing the value of M and the difference position, one can first handle the right part of the differential path (Q_M)
- then by forcing the inputs value $(C||M)$ to have very low (or very high) Hamming weight *hw* it will be possible to have $P_{step}(b)$ high



The freedom degrees

However, we can use the **freedom degrees**:

- by fixing the value of M and the difference position, one can first handle the right part of the differential path (Q_M)
- then by forcing the inputs value $(C||M)$ to have very low (or very high) Hamming weight hw it will be possible to have $P_{step}(b)$ high

$$P_{step}(b, hw) = \frac{hw}{c} \cdot P_{\mathtt{xor}}(k, hw, hw-1, b) + \frac{c - hw}{c} \cdot P_{\mathtt{xor}}(k, hw, hw+1, b)$$

Attack complexity and results

The total attack complexity is (probability P_X can be handled separately):

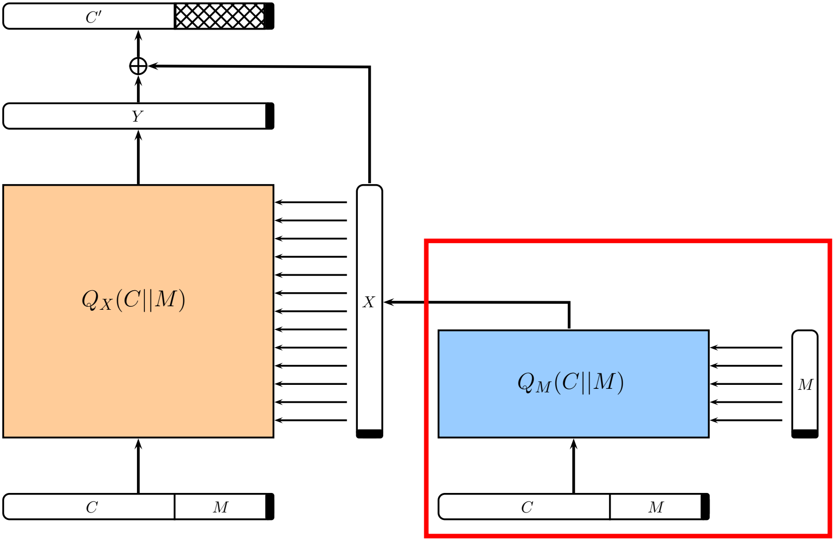
$$\frac{1}{\sum_{i=1}^{i=m} P_{step}(i, hw) \cdot P_{out}(i)}$$

scheme parameters			attack	
k	c	m	generic complexity	attack complexity
128	80	48	2^{40}	$2^{7.5}$
192	128	64	2^{64}	$2^{7.8}$
240	160	80	2^{80}	$2^{8.1}$
288	192	96	2^{96}	$2^{8.3}$
384	256	128	2^{128}	$2^{8.7}$

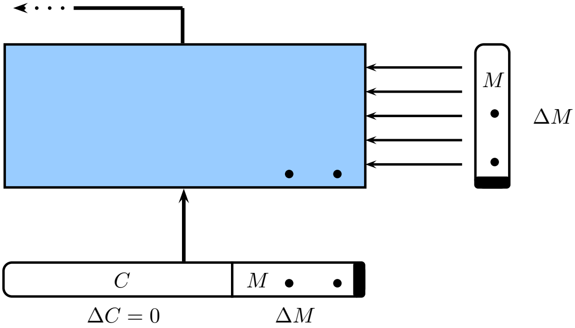
We implemented and verified the attack

Outline

The differential path - right side

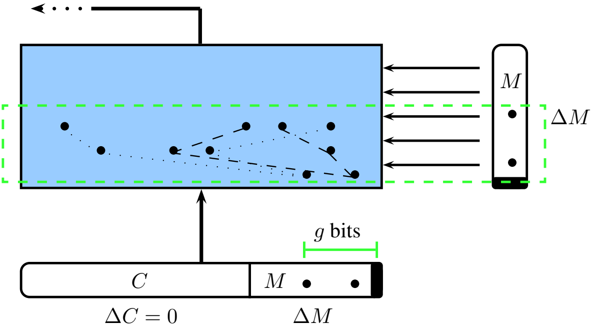


The differential path - right side



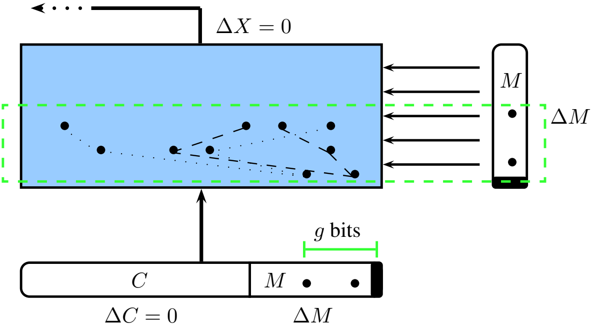
Assume we force the first g bits of M to a certain value (g being the most significant difference bit of M)

The differential path - right side



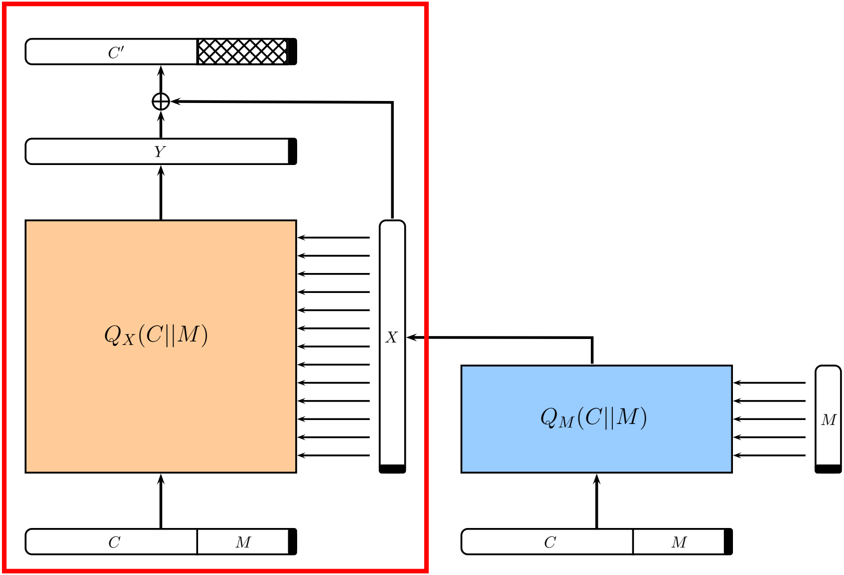
We would like a collision after step g, and this event can be obtained by solving a very particular system of linear equations since we know all first g steps

The differential path - right side

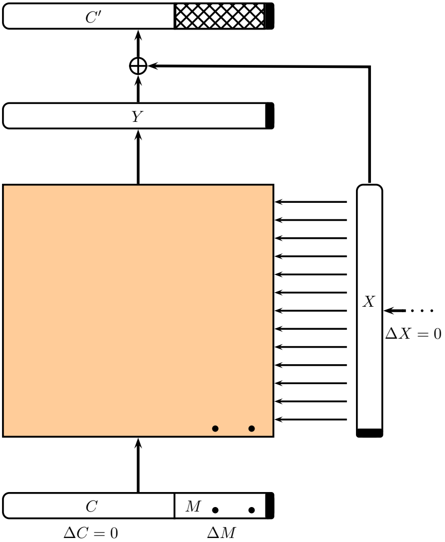


If the internal collision is obtained, we have $\Delta X = 0$ with probability 1

The differential path - left side

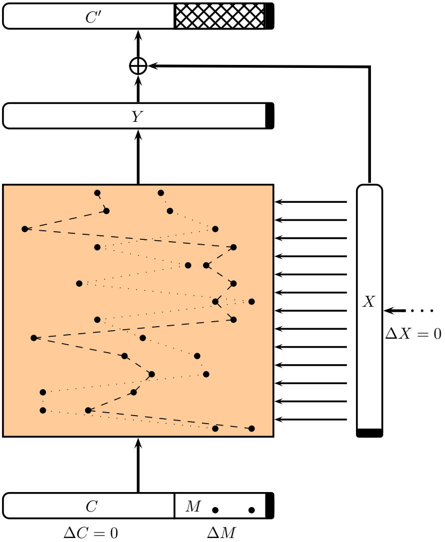


The differential path - left side



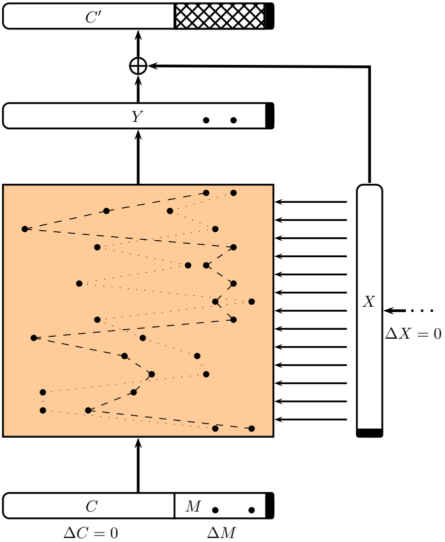
Assume we have b active bits on M

The differential path - left side



We have b active bits after applying Q_X with probability 1

The differential path - left side



We have $\Delta MSB_c(Y) = 0$ with probability

$$P_{out}(b) = P_{\mathtt{and}}(k, m, b, b) = \prod_{i=0}^{i=b-1} \frac{m-i}{k-i}$$

The system of linear equations

We know the value of the g first bit of M, therefore we know exactly the permutation applied to I and $I \oplus \Delta_I$ for the g first rounds of Q_M. For a collision after g rounds of Q_M, we want that

$$\begin{aligned} & \sigma_{M_1[g-1]}(\cdots(\sigma_{M_1[1]}(\sigma_{M_1[0]}(I) \oplus cst) \oplus cst)\cdots) \\ = \quad & \sigma_{M_2[g-1]}(\cdots(\sigma_{M_2[1]}(\sigma_{M_2[0]}(I \oplus \Delta_I) \oplus cst) \oplus cst)\cdots) \end{aligned}$$

and since **all operations are linear**, this can be rewritten as

$$\rho(I) \oplus A \quad = \quad \rho'(I \oplus \Delta_I) \oplus B = \rho'(I) \oplus \rho'(\Delta_I) \oplus B$$

where

$$\rho = \sigma_{M_1[g-1]} \circ \cdots \sigma_{M_1[1]} \circ \sigma_{M_1[0]} \qquad A = \sigma_{M_1[g-1]}(\cdots(\sigma_{M_1[1]}(cst) \oplus cst)\cdots)$$

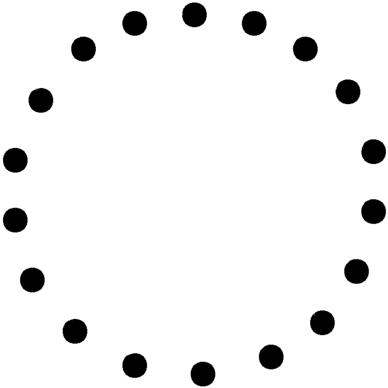
$$\rho' = \sigma_{M_2[g-1]} \circ \cdots \sigma_{M_2[1]} \circ \sigma_{M_2[0]} \qquad B = \sigma_{M_2[g-1]}(\cdots(\sigma_{M_2[1]}(cst) \oplus cst)\cdots).$$

The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

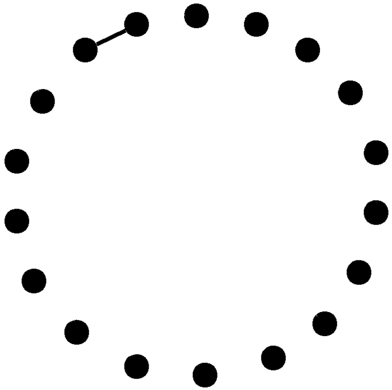


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

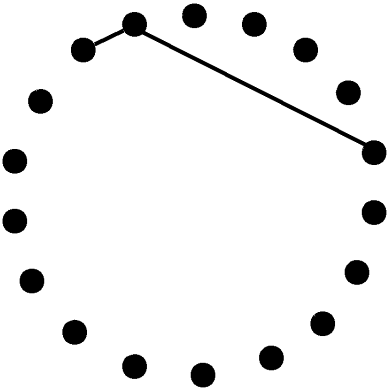


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

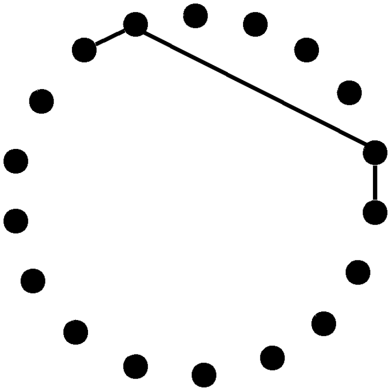


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

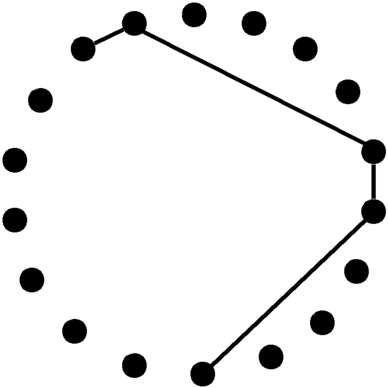


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

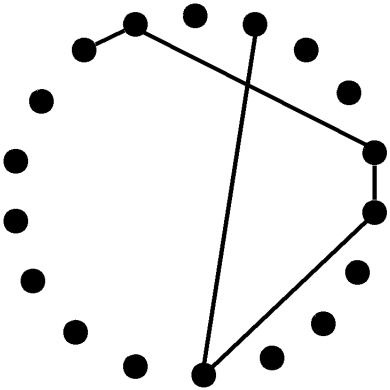


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

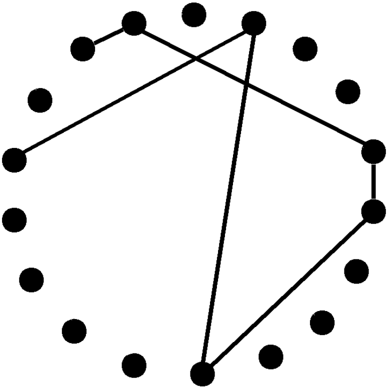


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

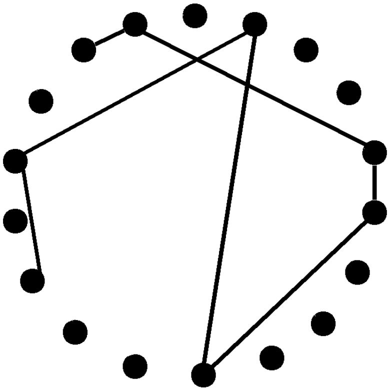


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

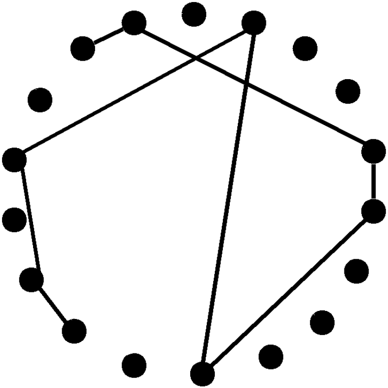


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

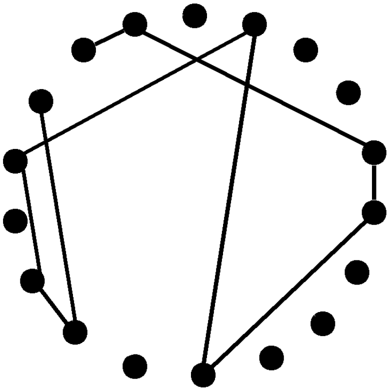


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

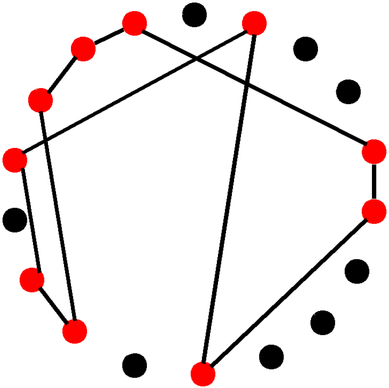


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

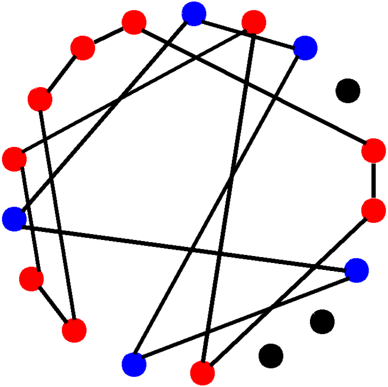


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)

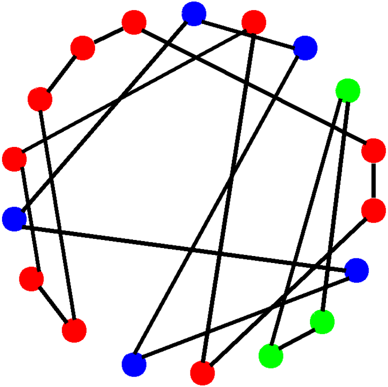


The system of linear equations

We have to solve $\rho(I) \oplus \rho'(I) = A \oplus B \oplus \rho'(\Delta_I)$ which can be rewritten

$$I \oplus \tau(I) = C$$

with C a constant and τ a bit permutation (we model as random)



The freedom degrees

The system of linear equations:

- admits at least a solution with a probability depending on the number of cycles of a complex composition of σ_0 and σ_1 (for random permutations σ_0 and σ_1, we have a probability of $2^{-\log(k)}$)
- **the average number of solutions is** 1

Thus, in order to find a collision, we need:

- that the guess of the g bits of M is valid (with probability 2^{-g})
- that the b active bits in M are truncated on the output of Q_X (with probability $P_{out}(b)$)

Minimizing g and b will provide better complexity, but we need enough randomization to eventually find a solution

Attack complexity and results

The total attack complexity is:

$$\frac{2^g}{P_{out}(b)}, \text{ with } \binom{g}{b} \geq 2 \cdot P_{out}^{-1}(b) \text{ so as to find a solution}$$

scheme parameters			attack	
k	c	m	generic complexity	attack complexity
128	80	48	2^{40}	$2^{8.9}$
192	128	64	2^{64}	$2^{10.2}$
240	160	80	2^{80}	$2^{10.2}$
288	192	96	2^{96}	$2^{10.2}$
384	256	128	2^{128}	$2^{10.2}$

We implemented and verified the attack

Outline

ARMADILLO-2 is not secure, attack complexities are very low:

- the diffusion can be controlled too easily
- local linearization allows to render linear the complex part of the differential paths
- the permutation $Q_A(B)$ preserves the parity of the input

Thank you for your attention !