# Practical Cryptanalysis of ARMADILLO-2 

María Naya-Plasencia and Thomas Peyrin

University of Versailles - France
Nanyang Technological University - Singapore

FSE 2012

Washington - March 19, 2012

## Outline

The ARMADILLO-2 function

Free-start collision attack

## Semi-free-start collision attack

Conclusion

## 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ 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 $\mathrm{A}[\mathrm{i}]=0$, apply the bitwise permutations $\sigma_{0}$, otherwise $\sigma_{1}$
- bitwise XOR the constant $1010 \cdots 10$ to the internal state



## The ARMADILLO-2 compression function



## The ARMADILLO-2 compression function



## 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 $\operatorname{HAM}(A \wedge B)=i$ is

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

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

$$
P_{\mathrm{xOr}}(k, a, b, i)= \begin{cases}P_{\text {and }}\left(k, a, b, \frac{a+b-i}{2}\right) & \text { for }(a+b-i) \text { even } \\ 0 & \text { for }(a+b-i) \text { odd }\end{cases}
$$

## Outline

The ARMADILLO-2 function

Free-start collision attack

## Semi-free-start collision attack

## Conclusion

The differential path - right side


The differential path - right side


## The differential path - right side



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

## The differential path - right side



We have $\Delta X=0 \ldots 01$ 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

$$
P_{\text {step }}(b)
$$

## The differential path - left side



We have $\operatorname{HAM}(\Delta Y)=b$ with probability
$P_{\text {step }}(b)$

## The differential path - left side



The differential path - overall differential probability


The overall collision probability is

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

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

- if $b$ is small, $P_{\text {step }}(b)$ is very low
- if $b$ is big, $P_{\text {out }}(b)$ is very low


## However, we can use the freedom degrees:

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

$$
P_{\text {step }}(b, h w)=\frac{h w}{c} \cdot P_{\mathrm{xOr}}(k, h w, h w-1, b)+\frac{c-h w}{c} \cdot P_{\mathrm{xor}}(k, h w, h w+1, b)
$$

## Attack complexity and results

The total attack complexity is (probability $P_{X}$ can be handled separately):

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

| scheme parameters |  |  | attack |  |
| :---: | :---: | :---: | :---: | :---: |
| $k$ | $c$ | $m$ | generic <br> complexity | attack <br> 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 ARMADILLO-2 function

Free-start collision attack

## Semi-free-start collision attack

## Conclusion

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 $\Delta M S B_{c}(Y)=0$ with probability

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

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 $\sigma_{0}$ and $\sigma_{1}$ (for random permutations $\sigma_{0}$ and $\sigma_{1}$, 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 $\left.P_{\text {out }}(b)\right)$

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:

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

| scheme parameters |  |  | attack |  |
| :---: | :---: | :---: | :---: | :---: |
| $k$ | $c$ | $m$ | generic <br> complexity | attack <br> 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

## The ARMADILLO-2 function

## Free-start collision attack

## Semi-free-start collision attack

Conclusion

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


## And Now for Something Completely Different

NTU is looking for

- a few PhD students (3 to 4 years)
- a few postdocs (1 to 2 years renewable)
on
- symmetric-key cryptography (cryptanalysis and/or design)
- lightweight cryptography
contact: thomas.peyrin@ntu.edu.sg


## Thank you for your attention !

