# Building power analysis resistant implementations of Keccak

Guido Bertoni[1] Joan Daemen[1]
Michaël Peeters[2] Gilles Van Assche[1]

[1]STMicroelectronics

[2]NXP Semiconductors

Second SHA-3 candidate conference, Santa Barbara, CA
August 23-24, 2010

# Outline

# Side-channel attacks

- Implementations may leak information through:
  - Computation time
  - Power consumption
  - Electromagnetic radiation
  - Actively generated faults
- Relevant in keyed modes or when processing secrets
  - MAC function, e.g. HMAC
  - Key derivation function
- KECCAK is more than just a hash function!
  - Stream encryption
  - Single-pass authenticated encryption

# Types of side-channel attacks

- Timing attacks
  - E.g., cache-miss attacks
- Power analysis
  - Simple (SPA): single trace suffices
  - Differential (DPA): multiple traces using statistical methods
- Electromagnetic analysis
  - Similar to power analysis
  - Simple (SEMA) or differential (DEMA)

# Differential power analysis

- First-order DPA:
  - Record traces for many computations
  - Partition traces based on a (partial) key value hypothesis
  - Detect correct key hypothesis from partition
- Flavours of DPA
  - CPA: exploits correlation with bit values
  - MIA: based on mutual information with bit values
- $m$-th order DPA
  - Considers joint distribution of $m$ time offsets
  - The higher the order, the more important trace alignment

# Countermeasures

- Different levels
  - Transistor-level: e.g. WDDL, SecLib, ...
  - Platform-level: redundancy, adding jitter, noise, ...
  - Program-level: dummy instructions, randomized order, ...
  - Algorithmic level: depends on algebraic operations
  - Protocol level: key usage limits, session keys, ...
- No such thing as 100 % security
- Robustness: combine countermeasures at different levels
- Cost: area and consumption increase, loss of speed, ...

# Secret sharing

- Countermeasure at algorithmic level:
  - Split variables in *random* shares: $x = a \oplus b \oplus \ldots$
  - Keep computed variables *independent* from *native* variables
  - Protection against $n$-th order DPA: at least $n + 1$ shares
- Implementation cost depends on the algebraic degree:
  - Linear: compute shares independently
  - Non-linear: higher degree $\Rightarrow$ more expensive
- KECCAK round function
  - Linear mapping $\lambda = \pi \circ \rho \circ \theta$ followed by nonlinear $\chi$:

$$x_i \leftarrow x_i + (x_{i+1} + 1)x_{i+2}$$

# Software: two-share masking

- Resistance against first-order DPA: two shares
- $\chi$ becomes:

$$\begin{aligned} a_i &\leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} \\ b_i &\leftarrow b_i + (b_{i+1} + 1)b_{i+2} + b_{i+1}a_{i+2} \end{aligned}$$

- Independence from native variables
  - Compute left-to-right
  - Avoid leakage in register or bus transitions
- Protection against higher-order DPA: noise and jitter
- Cost: roughly doubles RAM usage and computation time

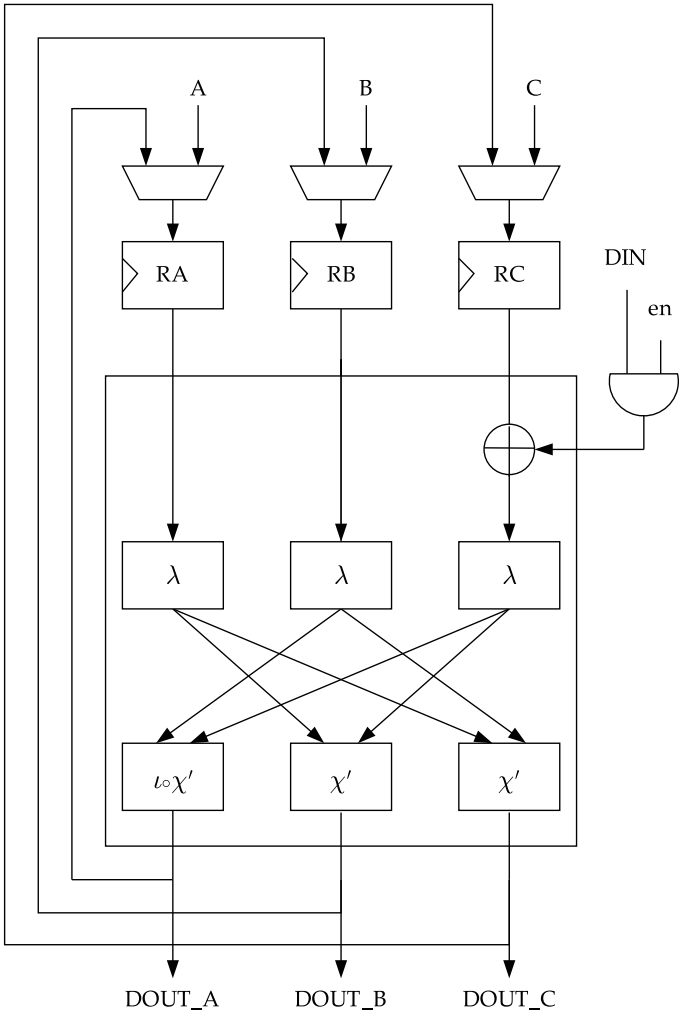
# Hardware: three-share masking

- [Nikova, Rijmen, Schläffer, Secure hardware implementations of nonlinear functions in the presence of glitches, ICISP 2008]:
  - Due to glitches computation order cannot be guaranteed
  - Idea: compute output share taking not all input shares
- Requires three shares for $\chi$:

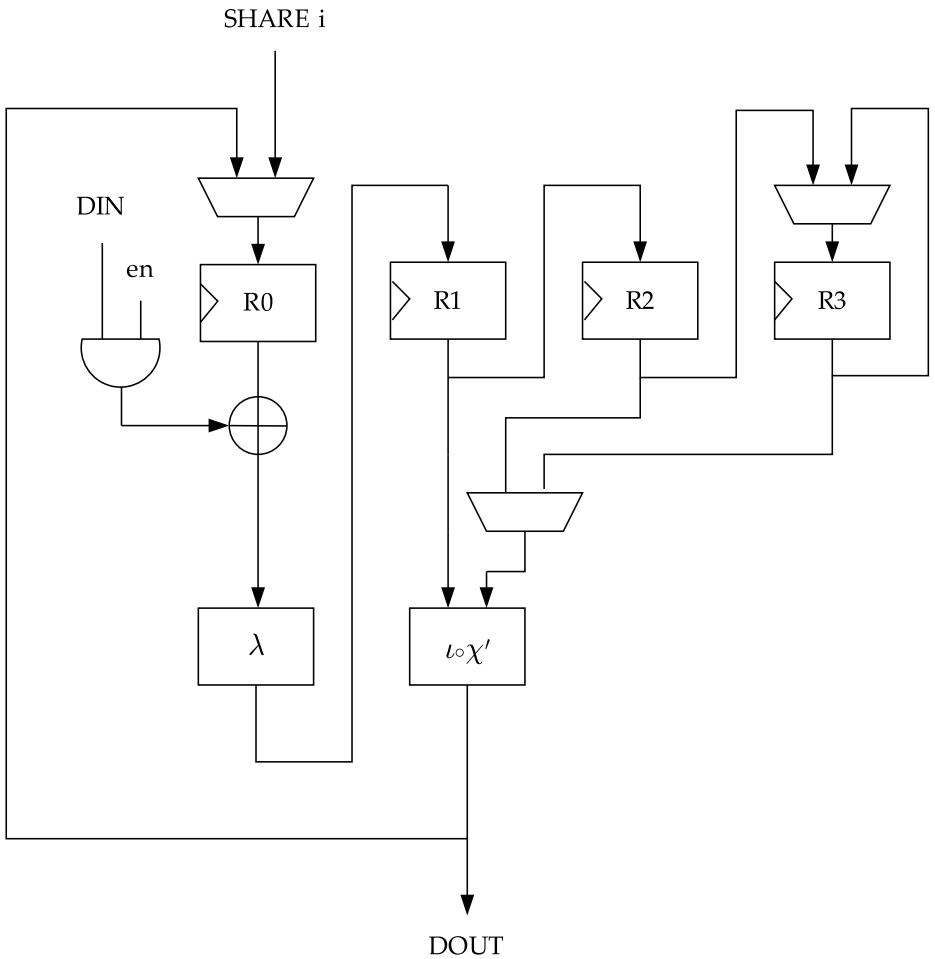
$$\begin{aligned} a_i &\leftarrow b_i + (b_{i+1} + 1)b_{i+2} + b_{i+1}c_{i+2} + c_{i+1}b_{i+2} \\ b_i &\leftarrow c_i + (c_{i+1} + 1)c_{i+2} + c_{i+1}a_{i+2} + a_{i+1}c_{i+2} \\ c_i &\leftarrow a_i + (a_{i+1} + 1)a_{i+2} + a_{i+1}b_{i+2} + b_{i+1}a_{i+2} \end{aligned}$$

- We present two architectures that implement this.

# One-cycle round architecture



# Three-cycle round architecture



# ASIC gate count and performance

| ASIC Core | Total size KGE | Frequency MHz | Throughput Gbit/s. |
|---|---|---|---|
| Unprotected one-cycle | 48 | 526 | 22.4 |
| One-cycle (fast) | 183 | 500 | 21.3 |
| One-cycle (compact) | 127 | 200 | 8.5 |
| Three-cycle (fast) | 115 | 714 | 10.1 |
| Three-cycle (medium) | 106 | 500 | 7.1 |
| Three-cycle (compact) | 95 | 200 | 2.8 |

rate: 1024 bits, technology: 130 nm STMicroelectronics

# Simulated power analysis

- Preliminary analysis based on simulated trace
  - Two architectures have been simulated at gate level: the plain fast core and the three-share one-cycle
  - 10,000 executions
  - Each execution performs 2 KECCAK-*f*
  - First absorbs the secret key
  - Second absorbs a known random message
- Correlation analysis
  - Highlighted leakage points on the plain architecture, like Hamming weights or Hamming distance
  - Applying the same analysis to the protected architecture results in no correlation
  - Work in progress...

# Conclusions

- Protection against side channel attacks is relevant
- KECCAK lends itself to implementations secure against DPA
  - Thanks to round function of algebraic degree 2
  - Software: speed divided by two
  - Hardware: excellent ratio performance/area
- Not easy for other architectures
  - See [Bertoni et al., Note on side channel attacks ...2009]
  - Large S-boxes: (very) expensive
  - ARX: particularly painful

http://keccak.noekeon.org/