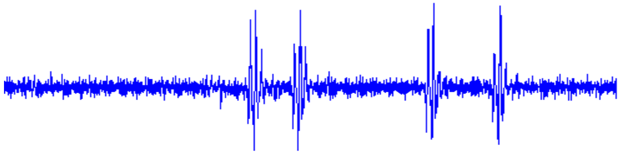
Masking Non-Linear Functions based on Secret Sharing

Martin Schläffer

Institute for Applied Information Processing and Communications (IAIK) - Krypto Group

Faculty of Computer Science
Graz University of Technology

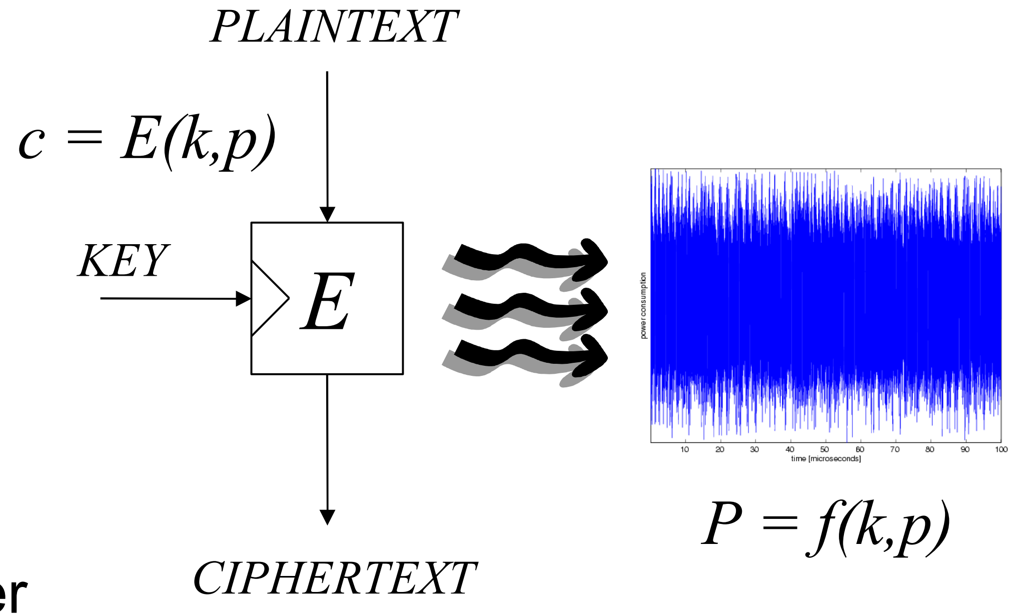
Overview



- **Introduction**
- Masking
- The Problem - Glitches
- From Masking to Secret Sharing
- Summary

Power and EM Analysis

- Power consumption depends on processed data
- Power consumption:
  $P = f(k,p)$
- Use different plaintexts to increase influence of key
- Attack:
  - Correlation between power consumption and key

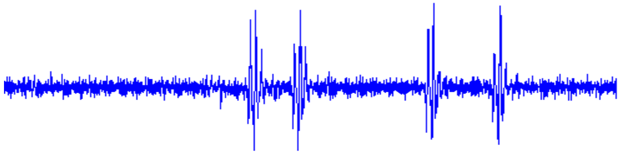


Countermeasures

→ Power consumption should be independent of processed data

- Hardware countermeasures
  - Gate level
  - Algorithm independent
  - Equal power consumption for different processed data
- Algorithmic countermeasures
  - Software level
  - Hardware technology independent
  - Mask (randomize) power consumption

Overview



- Introduction
- **Masking**
- The Problem - Glitches
- From Masking to Secret Sharing
- Summary

Masking

- Each value a is masked by an independent uniformly distributed mask m_a:

$$a_m = a \oplus m_a$$

  or equivalently: each value a is represented by a pair of values (a_m, m_a):

$$a = a_m \oplus m_a$$

- a_m needs to be independent uniformly distributed as well
  - The distribution of $a_m = a \oplus m_a$ is always the same, no matter which value a has

- Not the case with multiplicative masking: $a_m = a \cdot m_a \ (mod\ n)$
  - If $a=0$, then a_m is always zero, no matter which value m_a has

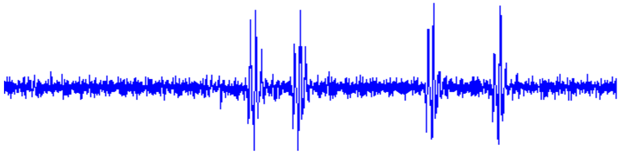
Security of Masking

- For all intermediate values a of the algorithm
  - m_a is independent uniformly distributed of a
  - a_m is independent uniformly distributed of a
  - each unmasked value a stays masked all the time

→ then power consumption is statistically independent of a

- There are security proofs on masking schemes
- But is masking therefore really secure?

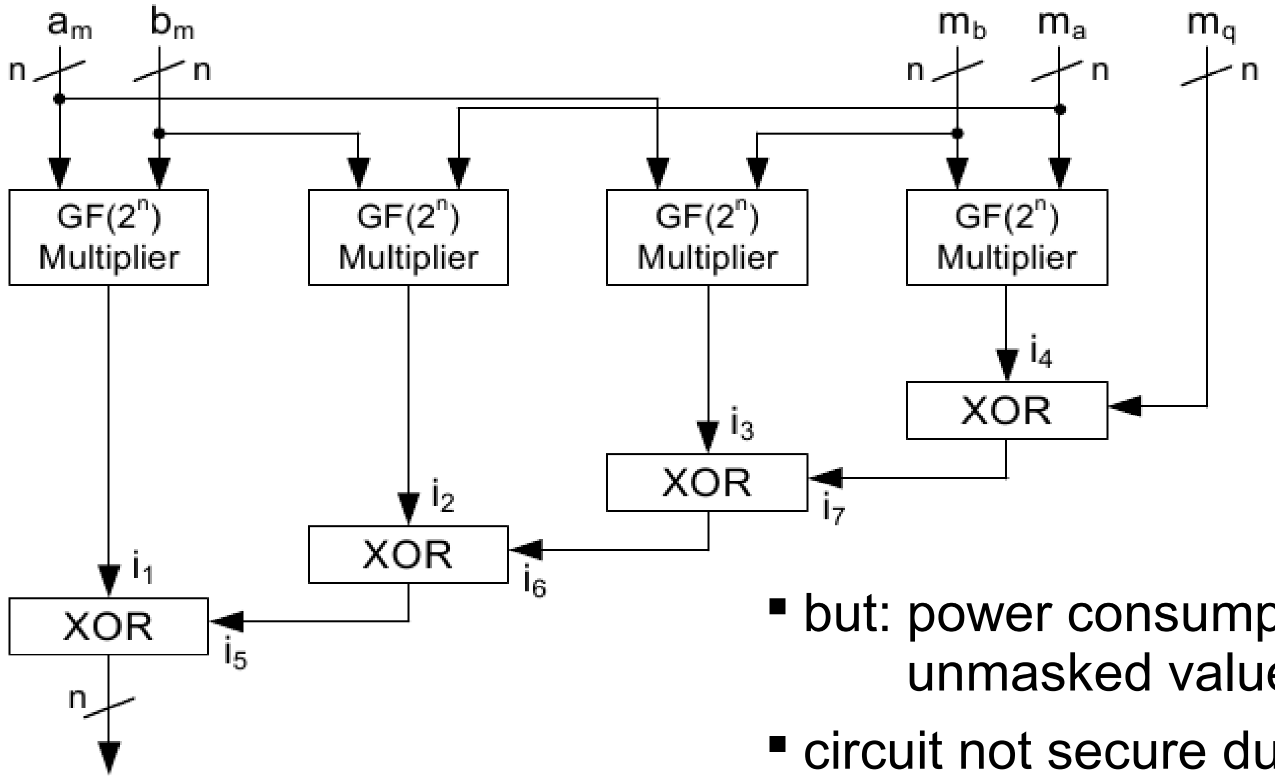
Overview



- Introduction
- Masking
- **The Problem - Glitches**
- From Masking to Secret Sharing
- Summary

Masked Multiplier

- All intermediate values are masked by independent uniformly distributed masks



- but: power consumption depends on unmasked values!
- circuit not secure due to *glitches*!

$$q_m = (a \cdot b) \oplus m_q = (a_m \cdot b_m) \oplus (b_m \cdot m_a) \oplus (a_m \cdot m_b) \oplus (m_b \cdot m_a) \oplus m_q$$

Glitches

- Glitches are unintended switching characteristics
  - delays on wires
  - different combinational paths
- The power consumption of a CMOS circuits strongly depends on the number of glitches (number of transitions)
- The number of glitches are a non-linear function of the inputs of a combinational logic
- In case of masked multiplier:

$$\#glitches = f(a_m, m_a, b_m, m_b, m_q)$$

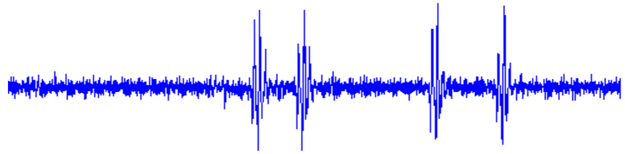
$$P = f(a_m, m_a, b_m, m_b, m_q)$$

- <u>Effect:</u> Provable secure masking schemes can be broken in practice!

Glitches

- Solutions for secure masking schemes:
- Hardware level:
  - Prevent glitches (by special logic styles)
- Algorithm level:
  - Find masking schemes resistant to glitches

Overview



- Introduction
- Masking
- The Problem - Glitches
- **From Masking to Secret Sharing**
- Summary

Basic Idea

- Nikova, Rechberger, and Rijmen [NRR06]
- Split computation into two parts:
  - Computation of mask
  - Computation of masked value
- Each combinational block is independent of unmasked value
  - Power consumption independent of unmasked value
  - Easy for linear components
  - Not possible for non-linear functions

- Example: Masked Multiplier
  - $a \cdot b = a_m \cdot b_m \oplus a_m \cdot m_b \oplus m_a \cdot b_m \oplus m_a \cdot m_b \ ?= f(a_m, b_m) \oplus f(m_a, m_b)$

Solution: use more than one mask! → secret sharing

Secret Sharing

- Split each input and output variable into n shares
  - (n,n) secret sharing scheme
  - n shares are needed to determine x uniquely

$$x = \bigoplus_{i=1}^{n} x_i$$

n-1
random values
required

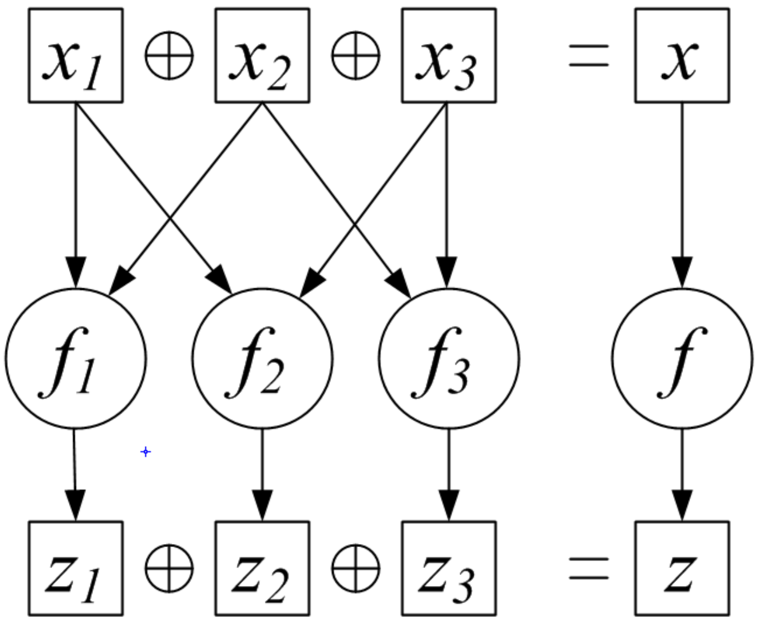
$$z = f(x) \quad \rightarrow \quad z_1 \oplus z_2 \oplus \ldots \oplus z_n = f(x_1 \oplus x_2 \oplus \ldots \oplus x_n)$$

$$z = f(x,y) \quad \rightarrow \quad z_1 \oplus z_2 \oplus \ldots \oplus z_n = f(x_1 \oplus x_2 \oplus \ldots \oplus x_n,\ y_1 \oplus y_2 \oplus \ldots \oplus y_n)$$

Secret Sharing

Split $f(x,y)$ into $f_i(x_1,x_2,\ldots,x_n,y_1,y_2,\ldots,y_n)$ such that

- each f_i is independent of x,y,z
- power consumption is independent of x,y,z



$$z = f(x)$$

$$z_1 = f_1(x_1, x_2)$$
$$z_2 = f_2(x_1, x_3)$$
$$z_3 = f_3(x_2, x_3)$$

Required Properties

- $f(x)$ is linear → easy
- $f(x)$ is non-linear → following properties for all f_i are needed

1. Correctness:
Sum of the output shares gives the desired function

2. Non-completeness:
Every f_i is independent of at least one share of each input variable.

3. Independent uniform distribution of input:
Any bias present in the joint distribution of the input shares is due to biases in the joint distribution of its unshared values.

Example: Shared Multiplier

- Secure masked AND gate
  - provable secure under transition count model (glitches)
  - with 3 shares

$$z = f(x, y) = x \cdot y$$

$$z_1 = f_1(x_2, x_3, y_2, y_3) = x_2 y_2 \oplus x_2 y_3 \oplus x_3 y_2$$
$$z_2 = f_2(x_1, x_3, y_1, y_3) = x_3 y_3 \oplus x_1 y_3 \oplus x_3 y_1$$
$$z_3 = f_3(x_1, x_2, y_1, y_2) = x_1 y_1 \oplus x_1 y_2 \oplus x_2 y_1$$

Apply to Arbitrary Functions

- Multiplication of n elements needs at least $n+1$ shares (see [NRR06])

- AES Sbox: inversion over GF(256)
  - $x^{-1} = x^{254} = x^{128} \cdot x^{64} \cdot x^{32} \cdot x^{16} \cdot x^{8} \cdot x^{4} \cdot x^{2}$
  - squaring is linear (characteristic 2)
  - 7 multiplications
  - we need 8 shares

- Hardware size increases about quadratic with the number of shares

- Can we reduce the number of shares?

Pipelining

- Use registers between combinational parts
- Registers are insensitive to glitches
- Split functions into parts with less non-linearity
  - Tower field approach of inversion

- Problem:
  - Property 3: the inputs of each step need to be independent uniformly distributed
  - Pipelining: output of each step is input of next step

→ We need Property 3 for output as well!

Uniform Shared Multiplier

- Cannot be fulfilled for shared multiplier with 3 shares
  - exhaustive search

- Uniform shared multiplier with 4 shares

$$z = x \cdot y = (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \cdot (y_1 \oplus y_2 \oplus y_3 \oplus y_4)$$

  - add correction terms
  - keep correctness and non-completeness!

$$\begin{aligned} z_1 &= (x_3 \oplus x_4)(y_2 \oplus y_3) \oplus y_2 \oplus y_3 \oplus y_4 \oplus x_2 \oplus x_3 \oplus x_4 \\ z_2 &= (x_1 \oplus x_3)(y_1 \oplus y_4) \oplus y_1 \oplus y_3 \oplus y_4 \oplus x_1 \oplus x_3 \oplus x_4 \\ z_3 &= (x_2 \oplus x_4)(y_1 \oplus y_4) \oplus y_2 \oplus x_2 \\ z_4 &= (x_1 \oplus x_2)(y_2 \oplus y_3) \oplus y_1 \oplus x_1 \end{aligned}$$

- Can we do better?

3-share Multiplication in GF(4)

$(e,f) = (a,b)\cdot(c,d) \quad \rightarrow$

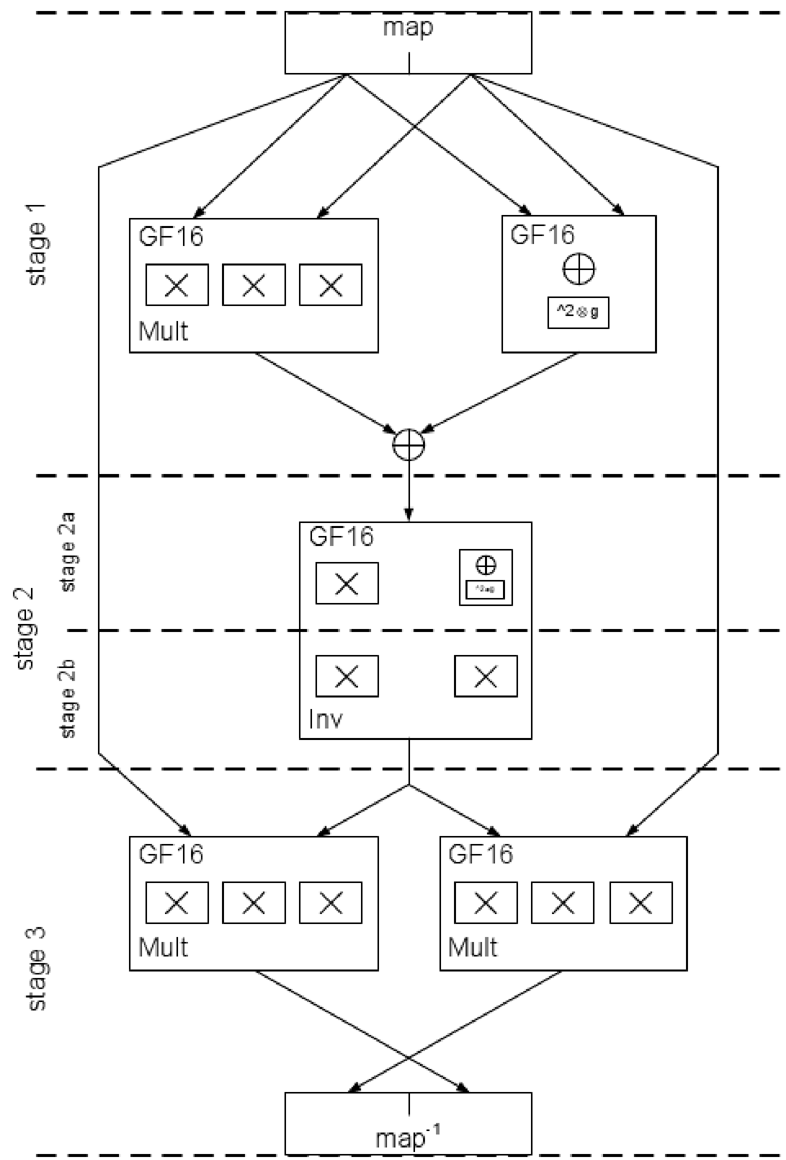
$(e_1 \oplus e_2 \oplus e_3,\ f_1 \oplus f_2 \oplus f_3) = (a_1 \oplus a_2 \oplus a_3, b_1 \oplus b_2 \oplus b_3)\cdot(c_1 \oplus c_2 \oplus c_3, d_1 \oplus d_2 \oplus d_3)$

- Split into 6 non-complete functions f_i
- Add correction terms (high degree of freedom)
  - 30 possible correction terms for each f_i
  - e_1: all combinations of $\{a_2,b_2,c_2,d_2\}$ and $\{a_3,b_3,c_3,d_3\}$
  - search space for one $f_i \sim 2^{30}$
  - can be reduced by combined search and reduced size
- Use solution as building block for inversion over GF(256)

Inversion over GF(256)

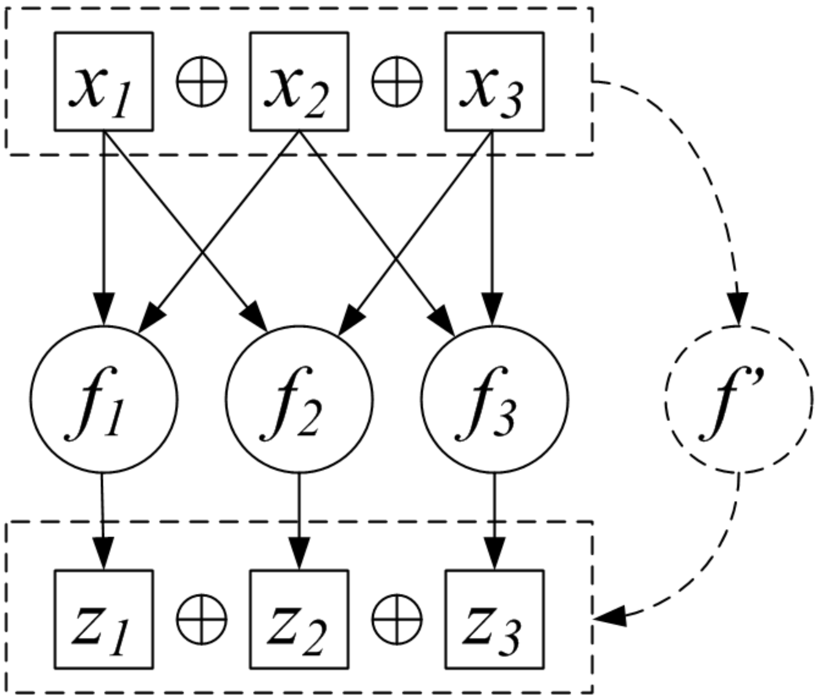
- Pipelined inversion
- Tower field approach
- Registers to avoid glitches

- Need to ensure properties in every step
  - Property 3 (uniformity) is difficult to achieve
  - High complexity of Boolean functions (8x3 = 24 inputs)

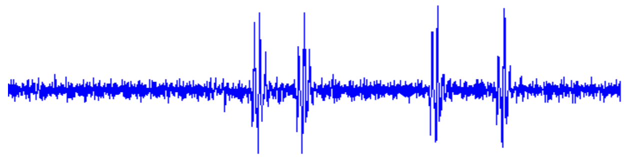


Vectorial Boolean Function

- View as vectorial Boolean function f' or (n,m)-function
- In case of $n = m$
  - we get Property 3 (uniformity) if f' is a permutation
- Other properties are difficult to fulfill
- What if $n \neq m$?
- Construction: Open problem



Overview



- Introduction
- Masking
- The Problem - Glitches
- From Masking to Secret Sharing
- **Summary**

Results

- Combinational parts independent of unmasked value
- Secure masked multiplier
  - 4-share GF(2)
  - 3-share GF(4)
- Security against first order attacks
- Security against higher order attacks
  - Increasing the number of shares

Open Problems

- Application to bigger (arbitrary) functions
  - Is there a solution?
  - How many shares?
  - How to find a solution?
  - Is the resulting circuit efficient?
- Is it secure in practice?
- Consider other attack scenarios and protection against them

Properties of New Method

+ Security against first-order attacks
  - even in the presence of glitches
+ Increase security by number of shares
+ Independent of combinational paths and wire lengths
+ Size comparable to hardware solutions (with 3-4 shares)

- Higher storage requirements
- Higher computational costs

Thank you for your Attention

Related Work

[NRR06] S. Nikova, C. Rechberger, V. Rijmen
Threshold Implementations Against Side-Channel Attacks and Glitches
LNCS 4307, Springer-Verlag, 2006, pp. 529–545