Multiplicative Masking and Power Analysis of AES

Jovan Golić

Rome CryptoDesign Center, Gemplus, Italy

Christophe Tymen

Gemplus and École Normale Supérieure, France

Presented by

Jean-François Dhem

Gemplus, France

GEMPLUS

OUTLINE

- Differential Power Analysis (DPA)
- Random Masking
- Higher-Order DPA
- Multiplicative Masking for AES
- Security Flaw of the Method
- DPA Attack
- Remedy Seems Impossible
- Embedded Multiplicative Masking
  - Overview of Countermeasure
  - Efficient Implementation
  - Security Analysis

Differential Power Analysis (DPA)

- **About power curves**
  - It is *critical* if they contain information about secret key in such a way that divide-and-conquer reconstruction attacks on parts of the secret key are feasible
- **Differential power analysis (DPA)** *[Kocher et al. 99]* is a powerful technique which
  - reconstructs the secret key in a divide-and-conquer manner
  - uses simple mathematical tools and
  - is practically independent on particular implementation

Fundamental hypothesis for DPA

- In the secret key algorithm, there exist intermediate variables that can be expressed as or are correlated to functions depending on a small number of key bits and on known input or output data

Basic idea

- Guess the involved key bits and partition measured power curves according to the computed value of the targeted intermediate variable
- Compare average curves and decide on the guess giving rise to significant differences, peaks, at one or more points in time
- Works if power consumption is not balanced, because of the same value being computed at the same time for the partitioned curves

Random Masking

Random masks

- Randomize computations and hence balance power consumption
- For block ciphers, first few and last few rounds are critical (fundamental hypothesis)
- When applied to affine operations, only additive constants have to be changed
- When applied to nonlinear operations, these operations typically have to be recomputed for each new mask; *for example, for an S-box*
  - ✔ a look-up table has to be stored in RAM instead of ROM
  - ✔ this is very costly for limited-space applications (smart cards)

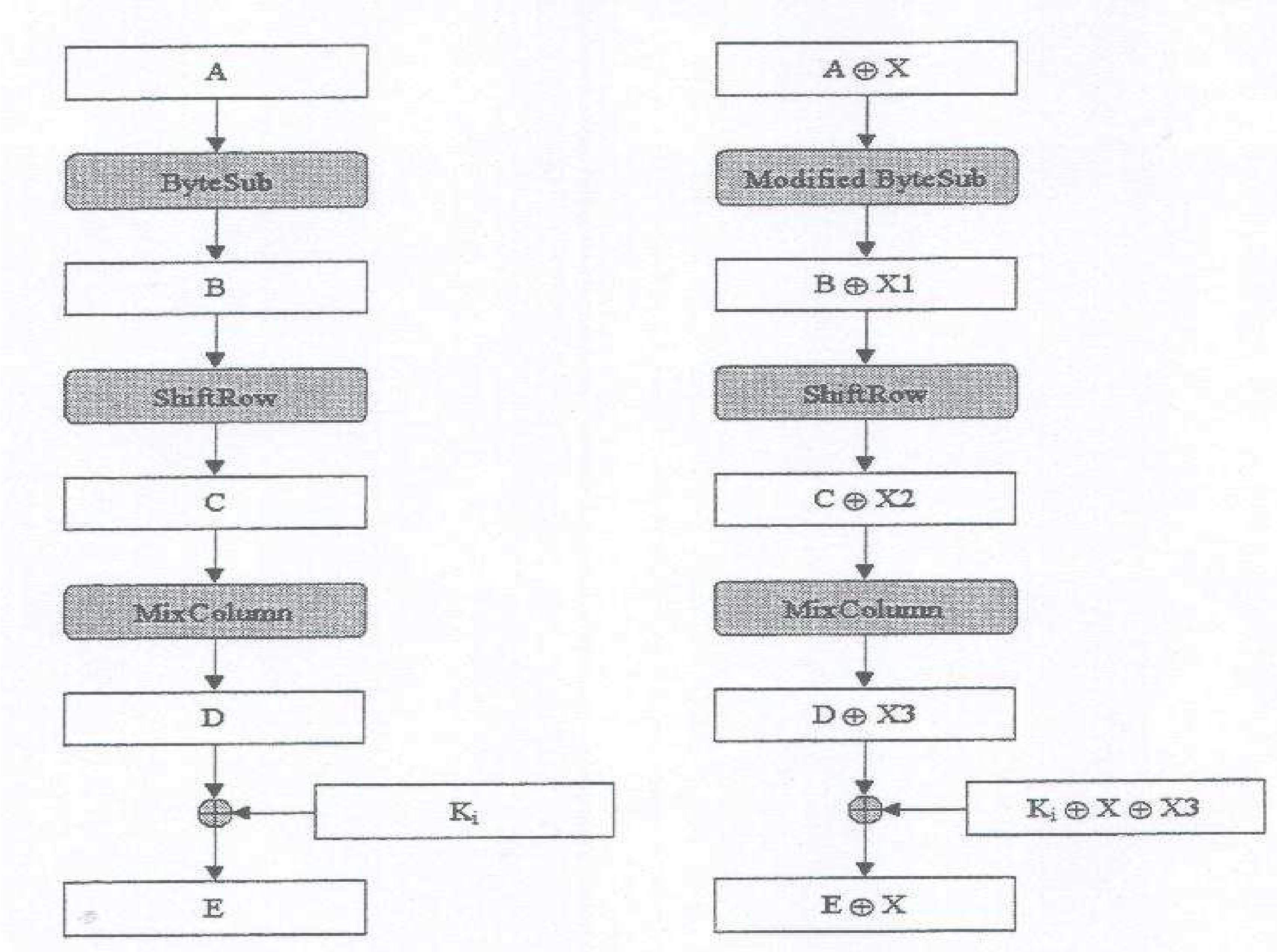
Higher-Order DPA

- Power curves are analyzed by using a joint statistic applied to a number of points in time
  - For example, for the second-order DPA, one can use variance of the difference *[Messerges 00]*
  - The attack is more complicated as suitable time points have to be identified

- Intermediate variables satisfying the fundamental hypothesis and being masked by the same mask are vulnerable to the second-order DPA

Multiplicative Masking for AES

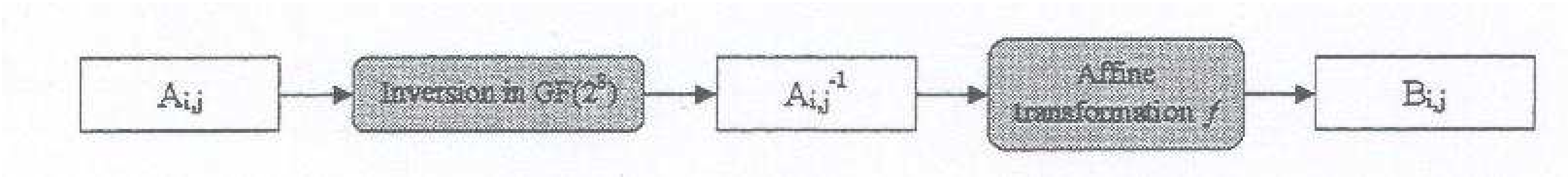
- *Akkar and Giraud at CHES 2001* observed that nonlinear parts of S-boxes in AES (Rijndael), performing the *multiplicative inversion in GF(256)*, can be randomized by multiplicative masking, without having to recompute and store them in RAM
- They also proposed to mask every round of AES and to use an extra binary additive mask, fixed for each round
- To this end, they proposed a secure method for the conversion between additive and multiplicative masks
- **Claim: method should be secure against DPA**



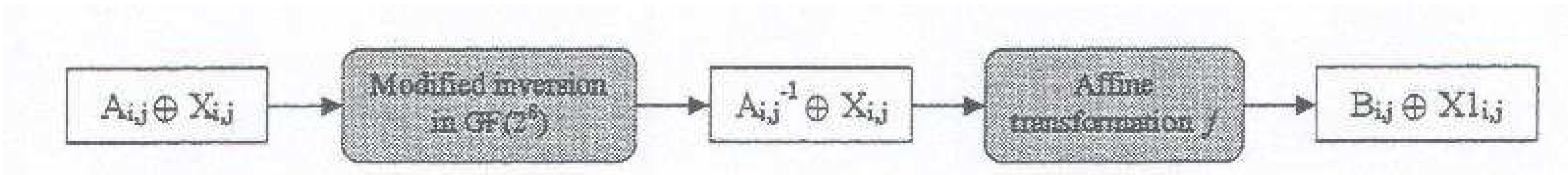
Slide #8

One round without and with masking

- ByteSub transformation without masking

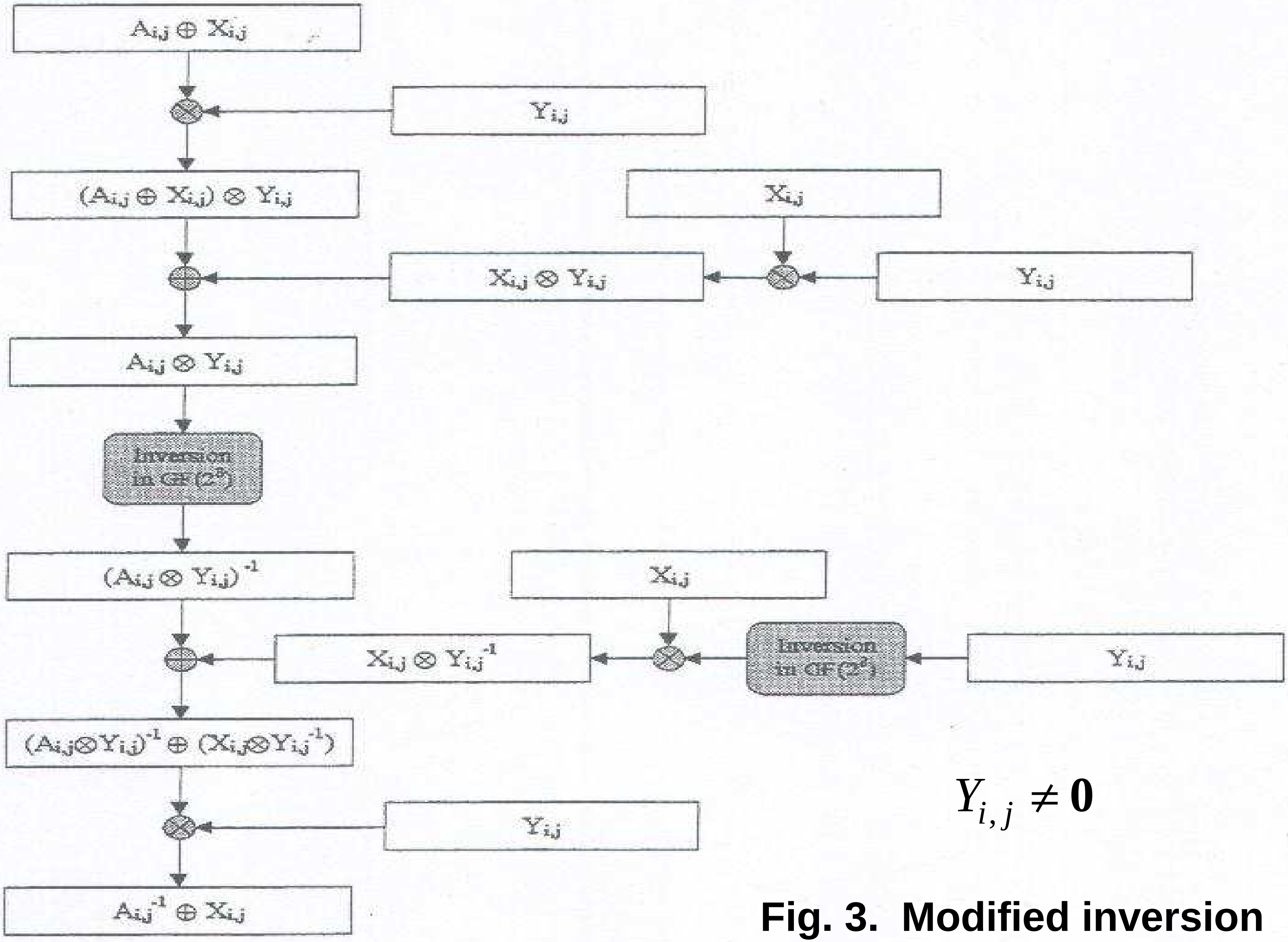


- ByteSub transformation with masking



(round index i, byte index j)

- Note that addition in GF(256) is the same as bitwise XOR



$$Y_{i,j} \neq \mathbf{0}$$

Fig. 3. Modified inversion

Security Flaw of the Method

- ***Problem*:**

  Multiplicative mask only masks non-zero data values, i.e., zero data value is not affected by masking, both at the input and the output of the multiplicative inversion

$$A_{i,j} = \mathbf{0} \;\Rightarrow\; A_{i,j} \otimes Y_{i,j} = \mathbf{0} \;\Rightarrow\; (A_{i,j} \otimes Y_{i,j})^{-1} = \mathbf{0}$$

- **As a consequence, the method is vulnerable to the 1st order DPA**

DPA Attack

- Note that $A_{i,j} = D_{i,j} \oplus K_{i,j}$ where $D_{i,j}$ is the data byte
- Objective is to recover the secret key used at the input to the 1st round, 8 bits at a time
- To perform the 1st order DPA , collect N power curves corresponding to the same secret key
- Guess 8 key bits and extract about $N/256$ power curves for which the corresponding data bits are equal to the key bits
  - for a correct guess, $A_{i,j} = \mathbf{0}$

- Compute the averages of these $N/256$ power curves and of all N power curves
- If the guess is correct and if $N/256$ is large enough, there will be peaks in the difference between the two average curves

- *In a sense, the DPA attack works better than without masking, because of randomization effect provided by multiplicative mask, when the guess is incorrect*
- However, without masking, one may also use partial output values of S-boxes for partitioning the power curves

Remedy Seems Impossible

- To remedy the weakness, one may try to replace the modified inversion computation on the zero input value by the computation on some other, random non-zero value $A_{i,j}$
- This will balance the computation if $A_{i,j} = \mathbf{0}$
- However, it is then necessary to
  - Perform computation to detect if $A_{i,j} = \mathbf{0}$
  - Replace the computed output value
- Both computations necessarily depend on input data and are hence vulnerable to the 1st order DPA
- **The point is that multiplicative masking does not cover the whole range of input values!**

- In conclusion, the multiplicative masking method is inherently vulnerable to the 1st order DPA
- ***It will be practically very important, especially for hardware implementations, to find a random masking method for AES that will not require recomputation and RAM storage of S-boxes***
- To this end, one should find (quasi)group operations for masking the input and output of an S-box that are compatible with the S-box nonlinear transformation, i.e., multiplicative inversion in GF(256)
- This does not appear to be likely

Embedded Multiplicative Masking

- ***Instead of an ideal solution to the problem, we provide an approximate solution, with a controllable security level***
- **The main idea is to randomly embed GF(256) into a larger algebraic structure so that**
  - the zero value is mapped into a set of values
  - the operations remain compatible with GF(256) so as to avoid recomputation and RAM storage of S-boxes
  - the multiplicative masks are used in essentially the same way as in *[Akkar, Giraud 01]*

Embedded Multiplicative Masking
Overview of Countermeasure

- Let P and Q be any two mutually coprime irreducible binary polynomials, where $\deg P = 8$ and $\deg Q = k$
- GF(256) can be represented as the ring of binary polynomials modulo P
- GF(256) is a subring of the ring of binary polynomials modulo PQ, $\mathcal{R} = \mathrm{GF}(256)[x]/(\mathrm{PQ})$, which itself is isomorphic to $\mathrm{GF}(256) \times \mathrm{GF}(2^k)$ with the isomorphism $U \mapsto (U_P, U_Q)$, where $U_P = U \bmod P$ and $U_Q = U \bmod Q$
- We will use the random mapping $\rho : \mathrm{GF}(256) \to \mathcal{R}$

$$U \mapsto \rho(U) = U + RP$$

where R is a random binary polynomial, $\deg R < k$

Embedded multiplicative masking method:

- Map the input $A_{i,j} \oplus X_j$ on Fig. 3 by ρ into $\mathcal{R}$ (here R acts as a k-bit embedding mask)
- Along the data path, the first multiplication and two additions are computed modulo PQ and the inversion is replaced by the mapping $F'(U) = U^{254}$ over $\mathcal{R}$ which on GF(256) coincides with the inversion
- All other operations remain the same as in the original multiplicative masking method

The zero value is mapped onto 2^k different values and should be more difficult to detect as k, the security parameter, increases

Embedded Multiplicative Masking
Efficient Implementation

- *Function* F' *should be implemented securely*
- The look-up table, in ROM, is impractical for $k \geq 8$
- **We propose the *'square-and-multiply'* method based on the specific choice of polynomials P and Q so that multiplication and squaring are easy**
- More precisely, we choose the polynomials satisfying

$$1 + x^{17} = (1 + x)P(x)Q(x)$$

and multiplication and squaring modulo $1 + x^{17}$ are very easy

(Check the paper for more details!)

- As P is different from that of AES, the conversion between the coordinates is performed by two 8×8 binary matrices, one of which is incorporated into ByteSub

- Complexity of computing F' is about 21 16-bit operations, as opposed to GCD-based algorithms which require at least about 100 16-bit operations

- Suitable for software implementations on 16-bit microprocessors and for hardware implementations

Embedded Multiplicative Masking
Security Analysis

- The average Hamming weight of 25 16-bit values involved in computing F' is obtained by computer simulations
- The maximum observed difference between the zero and nonzero cases is about 8.5% (8.596 versus 7.929)
  - without embedding, the difference is 0 versus 4 (for 8-bit values)
- To increase resistance against higher-order DPA,
  - *use mutually independent random masks (additive, multiplicative, and embedding), especially in the first and the last round of AES*
  - *use mutually independent random additive masks at the input and the output of one round (on Fig. 3)*
  - *randomize the order of S-box computations in a round*