DeKaRT: A New Paradigm for Key-Dependent Reversible Circuits

Jovan Goliæ

System on Chip, Telecom Italia Lab, Italy

CHES 2003, Cologne, Germany

Overview

1. Introduction
2. Probing Attacks and Data Scrambling
3. DeKaRT Method
4. Application for Block Ciphers
5. Power Analysis and Masking
6. DeKaRT Construction with Masking
7. Conclusions

1. Introduction

- **Main motivation:** resistance to side-channel attacks
- *Side-channel attacks* on microelectronic data-processing devices implementing cryptographic functions aim at recovering secret information through various measurements, without changing the functionality (*passive*)
  - Probing attacks (*physically invasive*)
  - Timing attacks
  - Power analysis attacks
  - Electromagnetic radiation attacks

- Countermeasures
  - *Algorithmic:* changes in hardware and/or software implementations of algorithm
  - *Physical:* passivation layers, shielding, detectors, sensors, filters, glue logic, ...
- Algorithmic countermeasures against probing attacks
  - Encryption or data scrambling of internal links and memories
- Algorithmic countermeasures against power analysis attacks
  - Masking with random masks: on software or hardware level

2. Probing Attacks and Data Scrambling

- *Probing attacks are invasive techniques consisting in introducing conductor microprobes into certain points of a tamper-resistant chip to measure electrical signals*
- Cryptographic function need not remain secure if intermediate data dependent on secret key is revealed
- Vulnerable points
  - Links and memories with regular structure: RAM, bus between CPU and RAM, bus between CPU and cryptoprocessor
- Countermeasures
  - Encryption or data scrambling (simplified encryption)

- **Encryption has to be done on hardware level, typically in only one CPU cycle** (*transparency*)
- <u>For memories</u>, encryption can depend on the address, and address can be encrypted too
- Code instructions can also be encrypted
- <u>For buses</u>, encryption can be achieved by bitwise XOR and a centralized (pseudo)random number generator implemented in hardware
- *<u>This solution is not satisfactory for memories</u>, as only one known pair of original and encrypted data yields the encryption key for a given memory location*

- **Usual block ciphers are too complex** (*gate count, speed, and power consumption*)
- **Restricted cryptanalytic scenario**
  - Partially known ciphertext
- Small and variable data block sizes (e.g., 8, 16, 32)
- For small block sizes,
  - one can simplify block ciphers, by reducing block size and number of rounds, and additionally use key-controlled bit permutations between rounds, in order to increase key size
  - however, security level is poor
- For very small block sizes,
  - key size is too small to resist meet-in-the-middle attacks

- <u>For small block sizes</u>,
  - inherent vulnerability to dictionary attack in known or chosen plaintext scenario; not realistic and does not recover secret key
  - *criterion:* secret key reconstruction attacks should be impractical, as the same key can be used for different scrambling functions
- Secret key for scrambling should be innovated for each new execution of cryptographic function
  - should be stored in a protected register, not RAM
  - can be generated by a (pseudo)random number generator

- **Main objective:**

  <u>In an iterated construction, each layer should</u>

  – implement a key-dependent reversible transformation

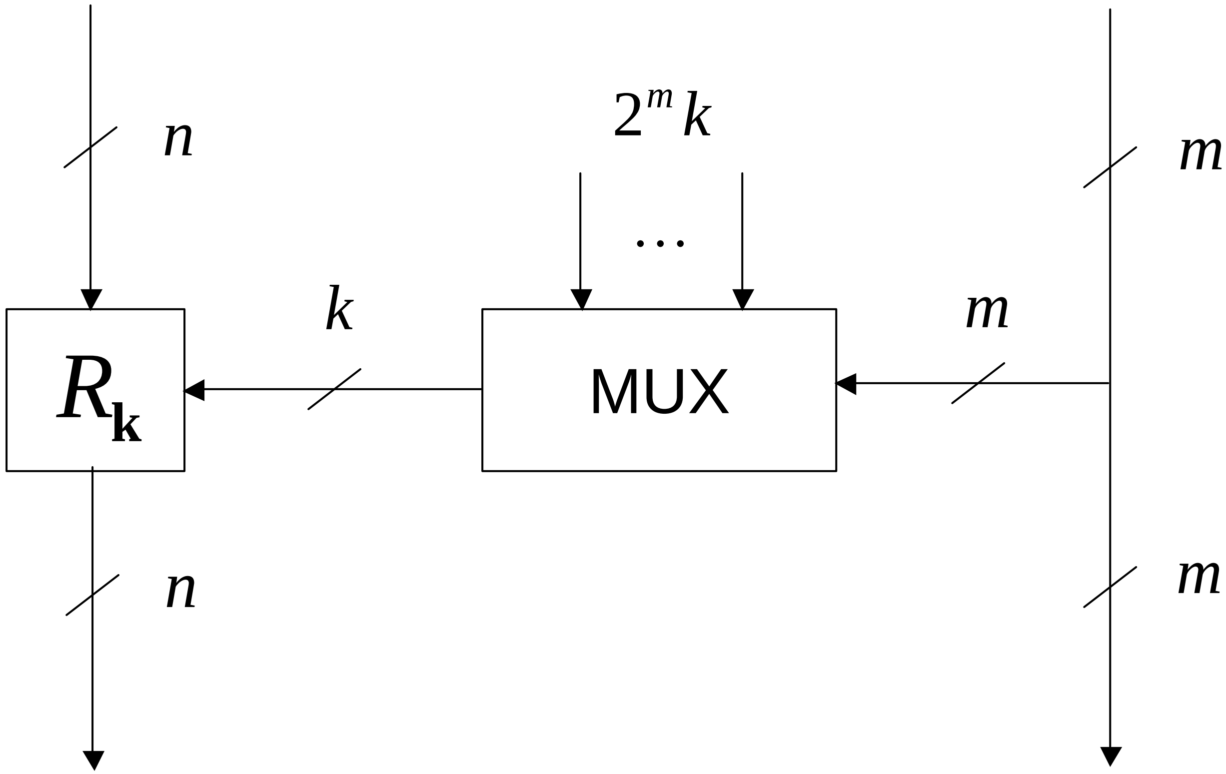
  – incorporate a relatively large number of key bits (impossible in usual costructions, for small block sizes)

  – have small logical depth and small size (gate count)

3. DeKaRT Method

- **Iterated and granular construction,** in which each layer consists of a number of elementary building blocks, each block implementing a key-dependent reversible transformation
- **A generic building block** acts on a small number of input data bits, divided into two groups of control and transformed bits
  – Control bits, which are taken intact to the output, choose key bits, which then choose a key-dependent reversible transformation acting on transformed bits which has to be easily implementable by a logical circuit

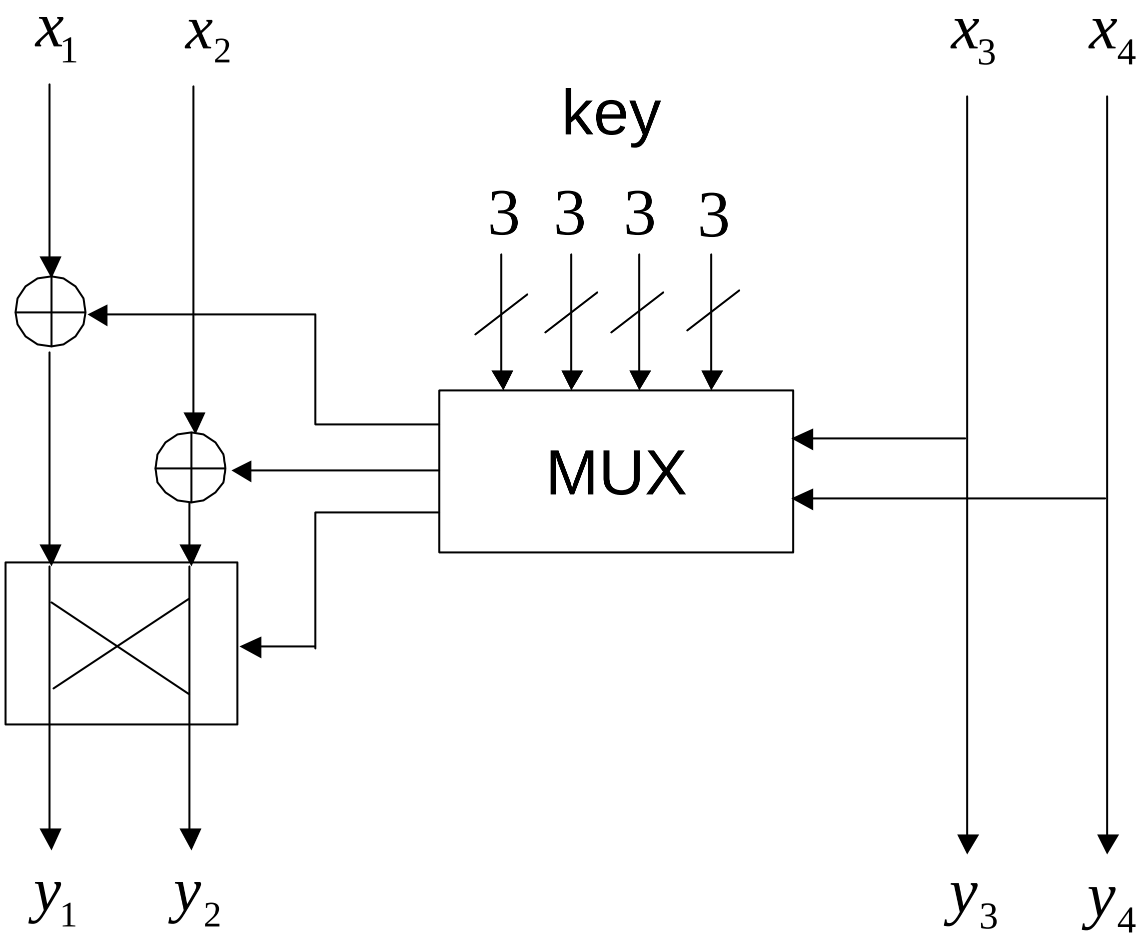
Generic DeKaRT Building Block



- **Underlying design paradigm:** D=>K=>RT
  - ***D****ata-chooses-****K****ey-chooses-****R****eversible_****T****ransformation*

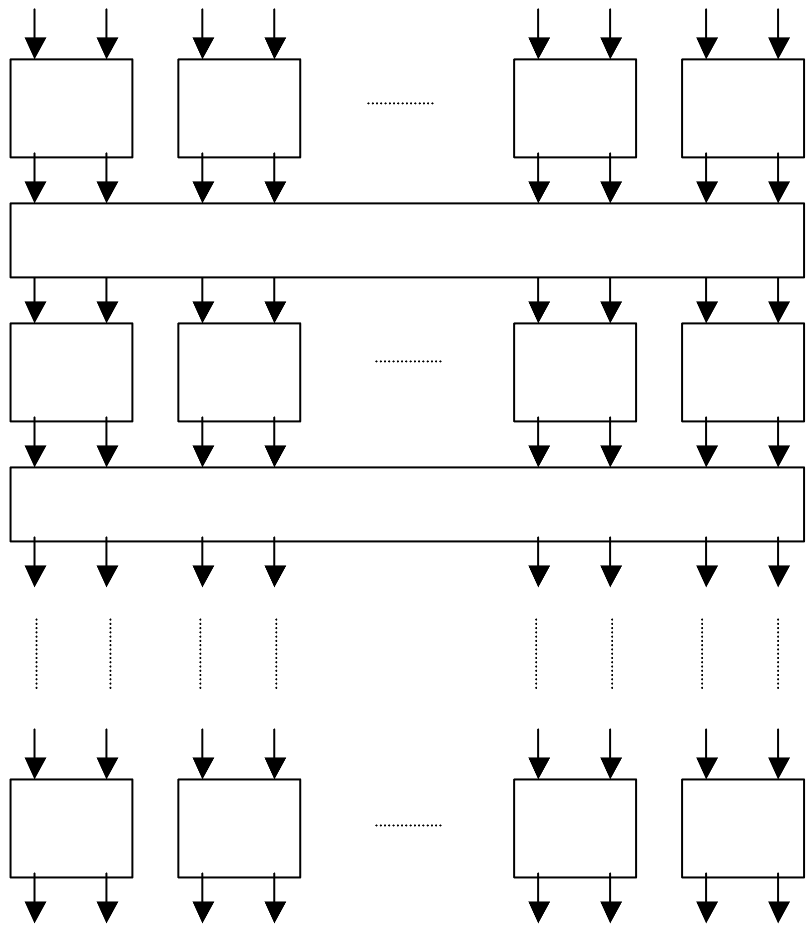
- **The problem is thus reduced** to designing key-dependent reversible transformations $R_{\mathbf{k}}$ acting on a smaller block size, with a difference that they do not have to depend on a relatively large number of key bits
- For example, such transformations can be implemented by a logical circuit composed of XORs and (controlled) SWITCHes
  - For each XOR, one input is a key bit
  - For each SWITCH, control bit is a key bit
- *For cryptographic security, certain additional properties regarding the choice of reversible transformations* $R_{\mathbf{k}}$ *can be imposed*

An Elementary DeKaRT Building Block



Size = 13 MUXes, Depth = 4 MUX levels, Key = 12 bits

Generic DeKaRT Network



- Layers are connected by fixed bit permutations satisfying the basic diffusion properties
  - Control bits in each layer are used as transformed bits in the next layer
  - In each building block, both control bits and transformed bits are extracted from maximal possible number of building blocks in preceding layer

- For data scrambling, a relatively small number of layers may suffice, e.g., 3 to 5

4. Application for Block Ciphers

- Block size is larger, e.g., 128
- To increase cryptographic security, a larger number of layers is needed, e.g., 32 or larger
- Between layers, in addition to bit permutations, use simple linear functions, e.g.,
  - XOR every transformed data bit at the input to each layer with a different transformed data bit from preceding layer
- XOR additional keys with input and output bits

- **DeKaRT construction can also be used for key expansion algorithm**
  - Linearly expand the secret key to fit the round key size (*avoid small subsets of expanded key bits to be linearly dependent*)
  - Use expanded key as input to another DeKaRT network, with possibly simplified building blocks, which is defined by a fixed key
  - Take intermediate outputs as round keys
  - *Desirable property:* round keys are connected together by reversible transformations

5. Power Analysis and Masking

- **Differential power analysis (DPA)** *Kocher et al. 99* is a powerful technique which
  - reconstructs the secret key in a divide-and-conquer manner, by partitioning the power curves measured in the known or chosen plaintext scenario
  - uses simple mathematical tools and
  - is practically independent of particular implementation
  - *works if power consumption depends on the values being computed*
- *Fundamental algorithmic hypothesis for DPA*
  - In the secret key algorithm, there exist intermediate variables correlated to functions of a small number of key bits and known input or output data

Masking on Hardware Level

- XOR input bits with random masking bits
- Modify logical circuit implementing cryptographic function: masked circuit acts on masked data bits and on (input, output, and auxiliary) masking bits
- Masking bits should preferably be produced by a random number generator, each time the cryptographic function is executed
- ***Secure computation assumption:*** *To prevent DPA, the output value of each logical gate in masked circuit should be statistically independent of secret key and input information*
- XOR output bits with random masking bits

- <u>**The whole logical circuit can be masked by masking individual logical gates**</u>
- Consider a logical gate implementing a Boolean function $f(X)$
- Let R be a binary vectorial input mask and r a binary output mask
- Masked gate is a logical circuit implementing the function $f'(X', R, r) = f(X' \oplus R) \oplus r$
- If $X' = X \oplus R$, then $f'(X', R, r) = f(X) \oplus r$

- ***Problem:*** how to compute $f'(X', R, r)$ securely?

- The computation is secure if each gate has an independent output masking bit; this requires a lot of memory to store the masking bits
- Alternatively, one can repeat the same masking bits
  - e.g., the output masking bit for a logical gate can be the same as one of the input masking bits, but *secure computation assumption* has to be satisfied
- More sophisticated power analysis attacks, such as higher-order DPA, may be applicable if the total number of masking bits is too small
- ***Masking on logical gate level is more secure than masking on software level, as ALL (elementary) computations are secure***

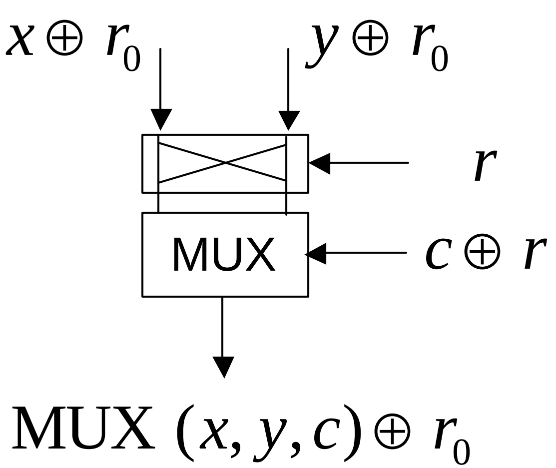
6. DeKaRT Construction with Masking

- The *m*-bit control MUX block can be masked by masking the constituent individual 1-bit control MUXes
- Logical circuit for implementing reversible transformations, composed of SWITCHes and XORs only, can be masked by masking the constituent SWITCHes
  - XORs are not masked; they change the masking bits only
- Masking bits should be assigned to individual gates so as to satisfy *secure computation assumption*

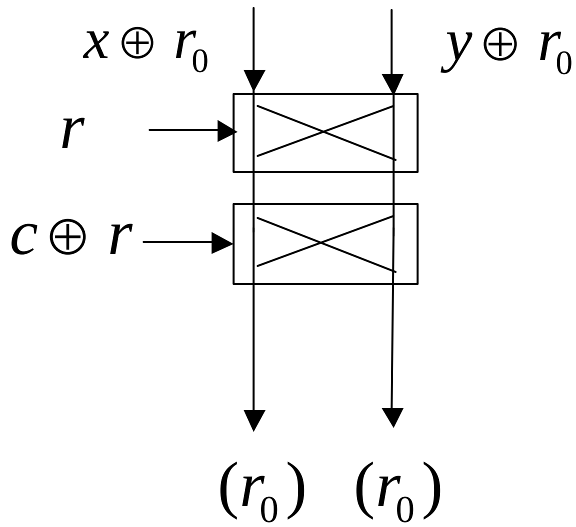
Masking MUX gate

- *Messerges et al.* US patent, Sept. 2001
- Mask a MUX by a cascade of a SWITCH and a MUX, where the SWITCH is controlled by the control masking bit and the MUX is controlled by the masked control bit

Masked MUX



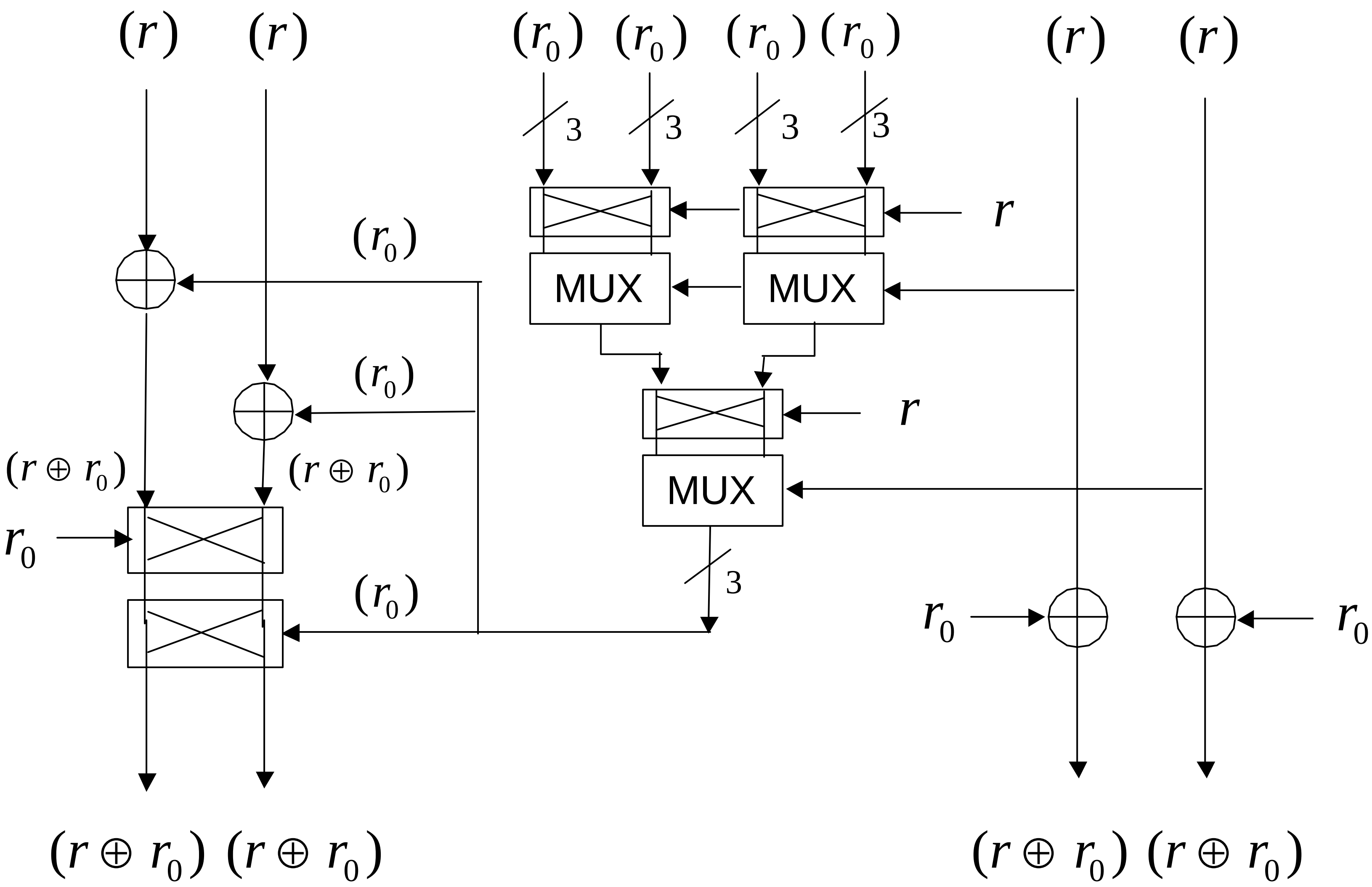
Masked SWITCH



- *Messerges et al.* US patent, Sept. 2001, can directly be applied to a tree of MUXes implementing a lookup table for a given Boolean function (in fact, one can show that only 2 masking bits suffice)

- Alternatively, it can be applied to any logical circuit consisting of MUXes, but the masking assignment should be such that for every MUX
  - 2 input masking bits are the same; this can be achieved by using additional XORs to adapt the masks
  - control masking bit is independent of input masking bit

A Masked DeKaRT Building Block



7. Conclusions

- **DeKaRT is a new general method** for
  - encryption of internal links and memories in data-processing devices against probing (and power analysis) attacks
  - hardware-oriented block ciphers
- **Masked DeKaRT construction** provides security against power analysis and other side-channel attacks on logical gate level
- ***Masking on logical gate level is more secure than masking on software level***
- *Analyzing cryptographic security of DeKaRT networks, in chosen plaintext and possibly partially known ciphertext scenario, is an interesting research problem*