Embedded Systems Security

Guy GOGNIAT
guy.gogniat@univ-ubs.fr

Journée 2007 de la section électronique du club EEA
SiP et SoC : nouvelles perspectives, nouveaux défis

Session Sécurité Informatique
Mercredi 28 mars 2007

Outline

- Cryptography principles
- Attacks on embedded systems
- Countermeasures
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

Outline

- **Cryptography principles**
- Attacks on embedded systems
- Countermeasures
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

Cryptography primitives

- Confidentiality
  - Data and messages

- Symmetric cryptography
  - AES, DES/3DES, RC5

- Integrity
  - Data and messages

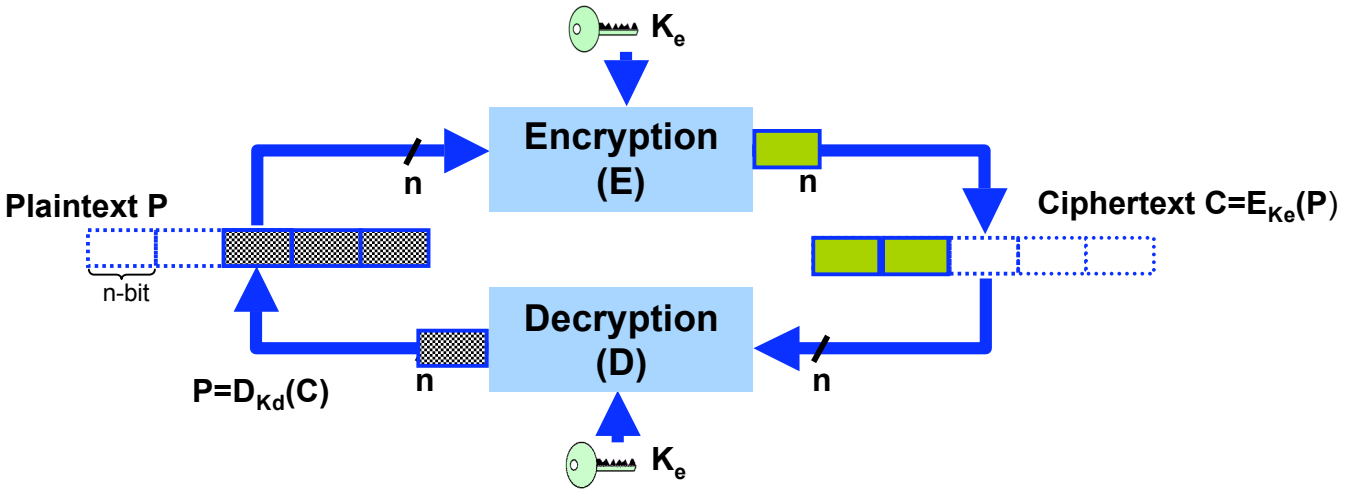
- Hashing function
  - MD5, SHA-1, SHA-2

- Authentication
  - Users and hosts

- Asymmetric cryptography
  - RSA, ECC

Symmetric encryption

- Block cipher



Asymmetric algorithm

- Each user has a Public Key (Published)
- And a Private Key (Secret)

$$E_{K1}(M) = C \qquad D_{K2}(C) = M \qquad D_{K2}(E_{K1}(M)) = M$$

RSA

1. Public key (size 1024 or 2048 bits)

$$n = p \times q$$

Compute **e** as "PGDC(n,e) = 1"

2. Private key

$$d = e^{-1} \bmod ((p-1)\times(q-1))$$

3. Ciphering requires **e** and **n**

$$c = m^e \bmod n$$

4. Deciphering requires **d** and **n**

$$m = c^d \bmod n$$

Three researchers from MIT, Ron **R**ivest, Adi **S**hamir and Len **A**dleman have patented in 1983 the **RSA** algorithm

Hashing function

Initial message

Prof. Robert this message to confirm our meeting tomorrow at 1 pm at my office

Digest of the initial message

215e781c0c3f7d1353518bd5f649805b

Received message

Prof. Robert this message to confirm our meeting tomorrow at 9 pm at my office

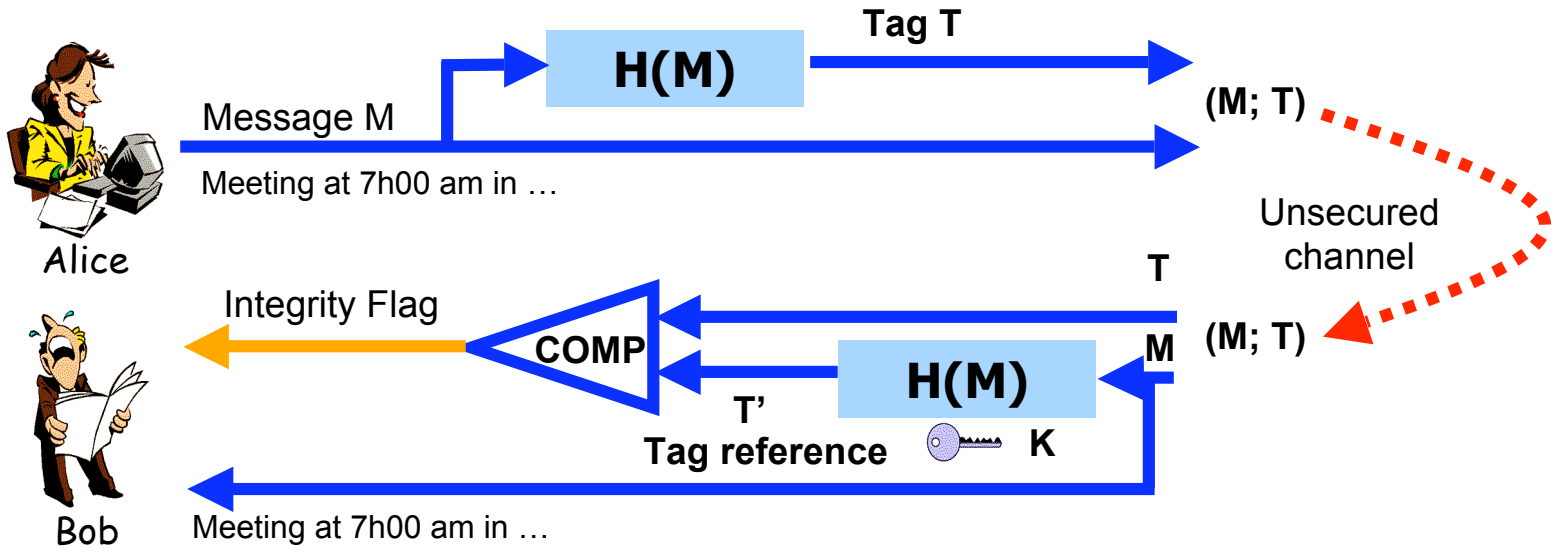
Digest of the received message

0601e38b93c1cc1c1a4b87dd8771b452

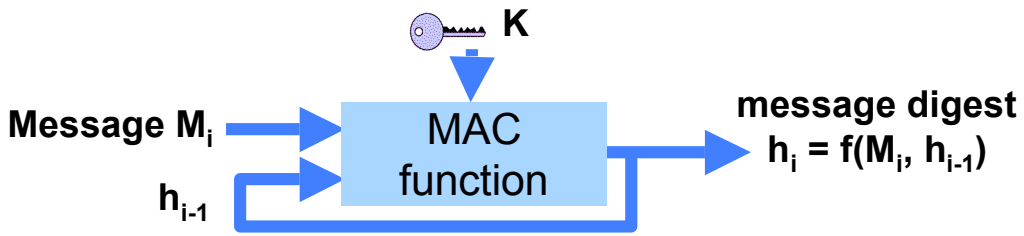
- Both digests are different
  - Someone has modified the message
  - There been an error during the communication

Integrity Checking

- **Principle:**



- **Hash functions:**
  - ✓ Compression function
  - ✓ One-way function
  - ✓ gives a compact **representative** image of the input



- **MAC(*) functions:** take a secret key as additional input **to authenticate** the source of the message.

(*) Message Authentication Code

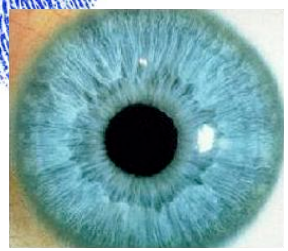
Outline

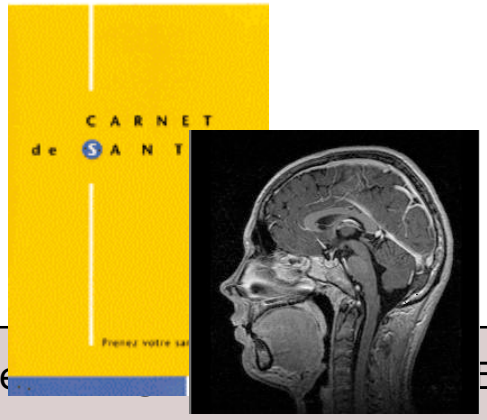
- Cryptography principles
- **Attacks on embedded systems**
- Countermeasures
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

Many sensitive data will be embedded

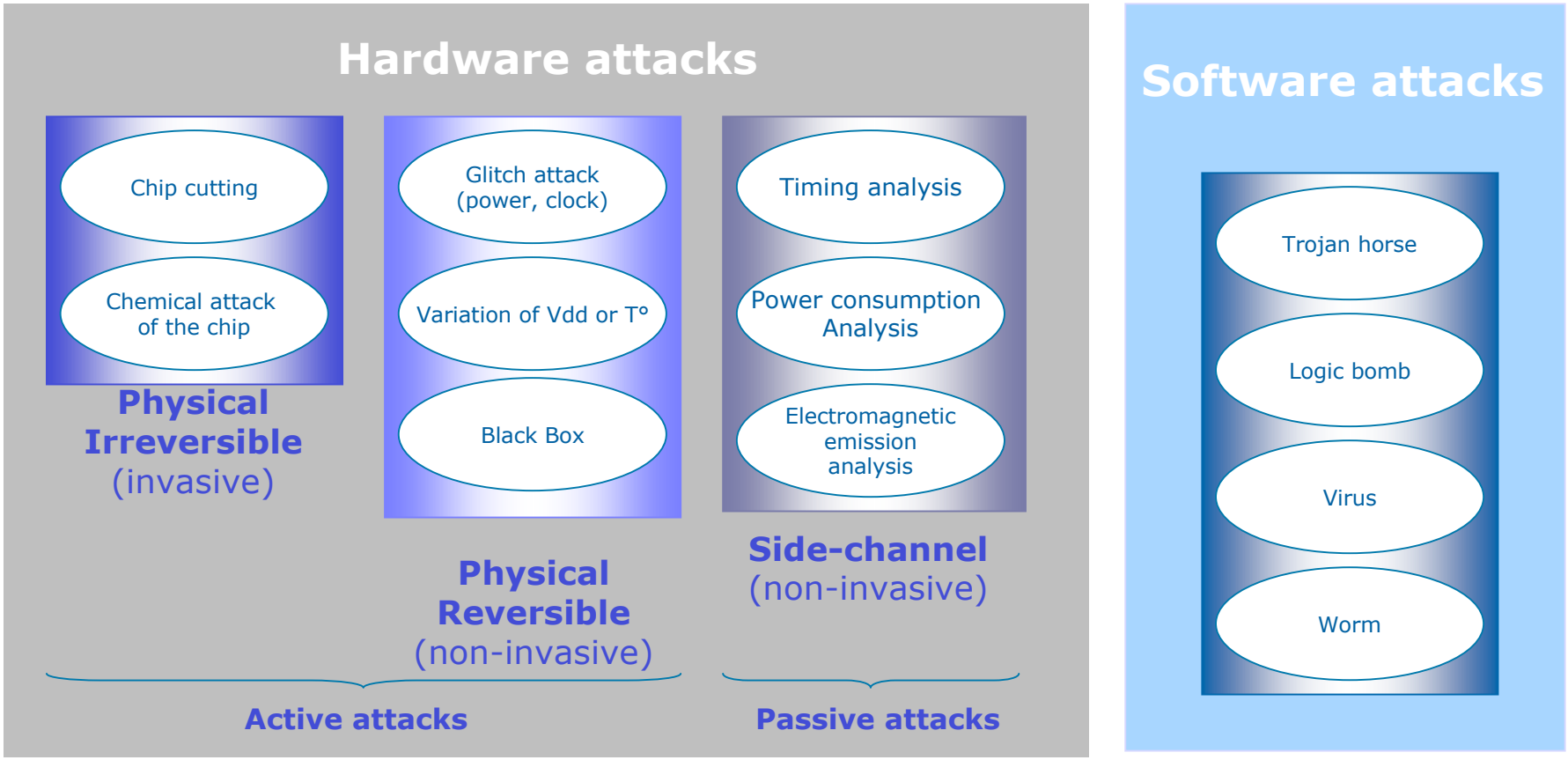
DALLET Dominique
Maître de conférences, département Télécommunications, responsable des stages industriels de 2e année au département Télécommunications
poste : 23 40 fax : 05 56 37 20 23 S 307
Dominique.Dallet@enseirb.fr

IXL Responsable de l'équipe Métrologie de la conversion de données
tél. 05 40 00 26 32 fax : 05 56 37 15 45
dallet@ixl.fr





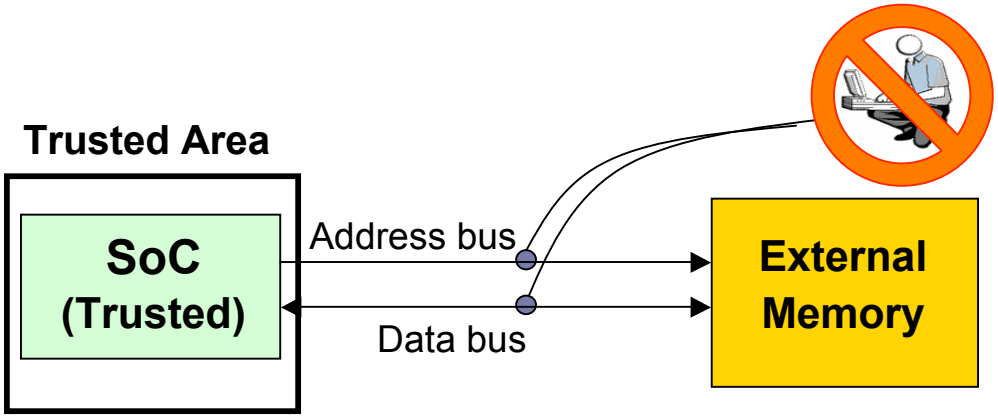
Classification of attacks



Processor-Memory Transactions Vulnerabilities

- Most embedded systems use off-chip memories
  - Data and instructions are exchanged in clear over the processor-memory bus

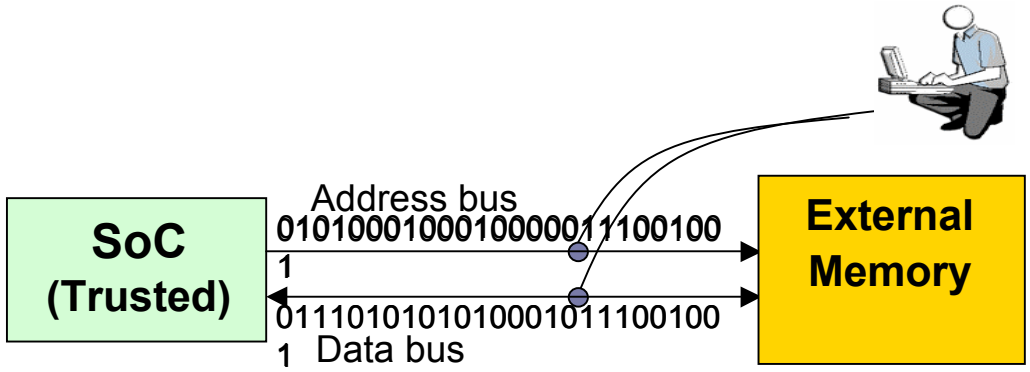
- Threats:
  - ✓ Unauthorized data reads
  - ✓ Code injection or data alteration
  - ✓ Memory tampering



- **Objectives:** Ensure the **confidentiality** and the **integrity** of data stored in off-chip memories and transferred on SoC memory interfaces

Passive Attacks

- Bus probing – eavesdropping [1]

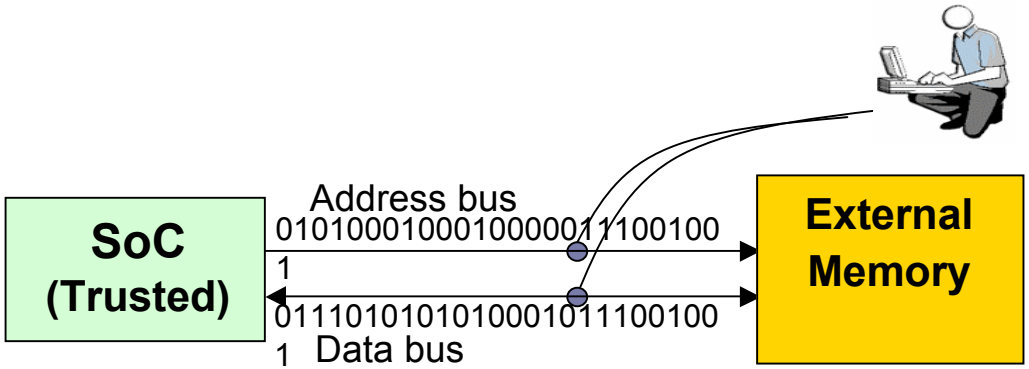


Add	Data / Instruction
0x00000010	0x080ff0fa

[1] M. G. Kuhn, “Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP” IEEE Trans. Comput., vol. 47, pp. 1153–1157, October. 1998.

Passive Attacks

- Bus probing – eavesdropping [1]



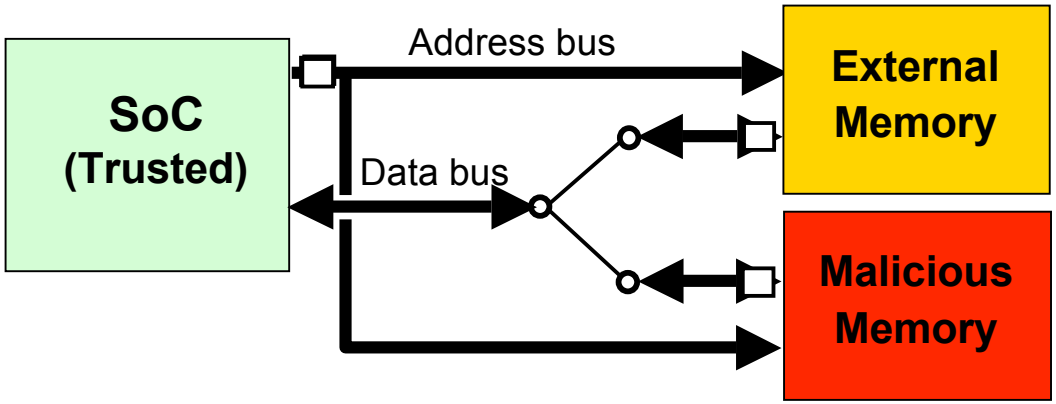
Add	Data / Instruction
0x00000010	0x080ff0fa
0x00000014	0x0ab820ff
0x00000018	0x080112f4
0x0000001C	0x102bcd0f
0x00000020	0x11ff11ab

- Attacker motivation:
  - ✓ Off-line analysis:
    - Key recovery
    - Message recovery
  - ✓ Raw materials for active attacks…

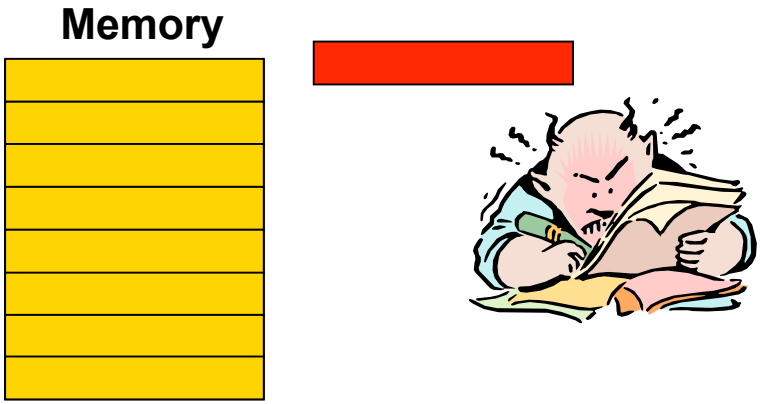
[1] M. G. Kuhn, “Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP” IEEE Trans. Comput., vol. 47, pp. 1153–1157, October. 1998.

Active Attacks

- Code and data injection

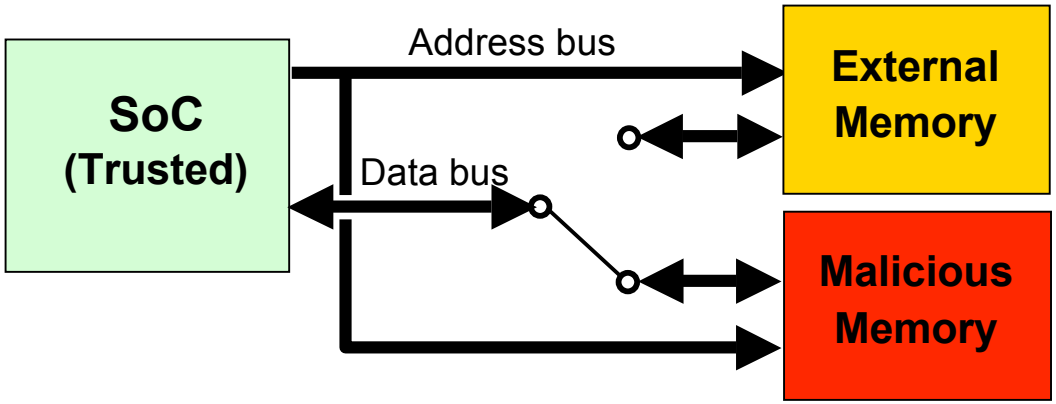


- Three kinds of active attacks are defined depending on the choice made by the adversary on the data to insert:
  - ✓ Spoofing: Random data injection

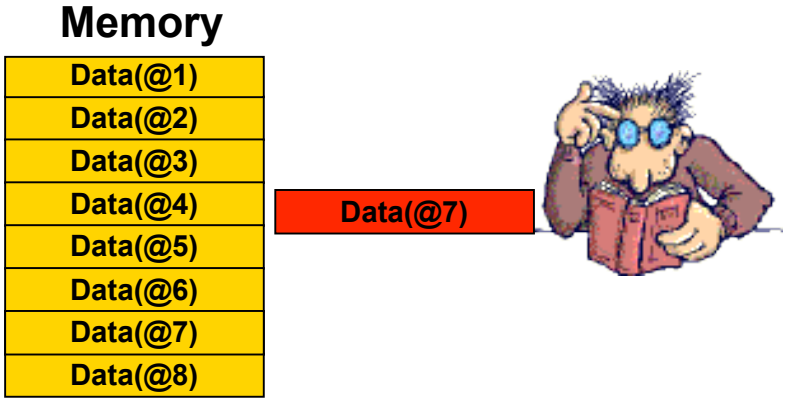


Active Attacks

- Code and data injection

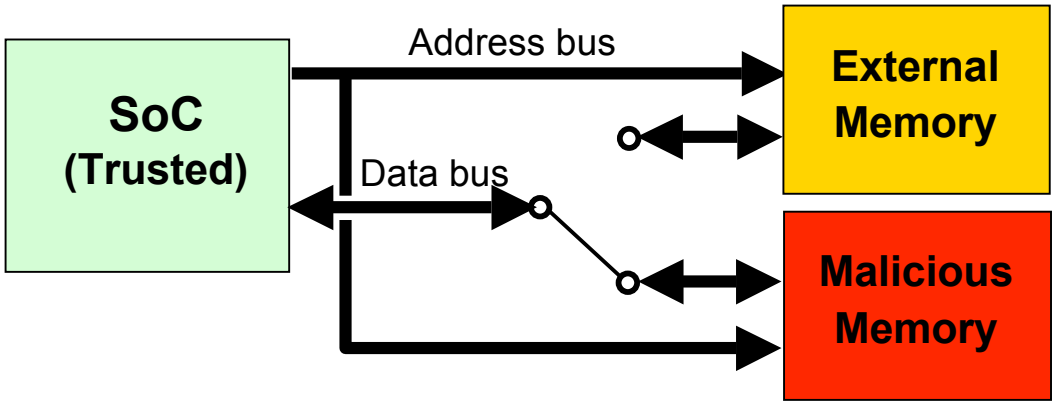


- Three kinds of active attacks are defined depending on the choice made by the adversary on the data to insert:
  - ✓ Spoofing: Random data injection
  - ✓ Splicing: Spatial permutation

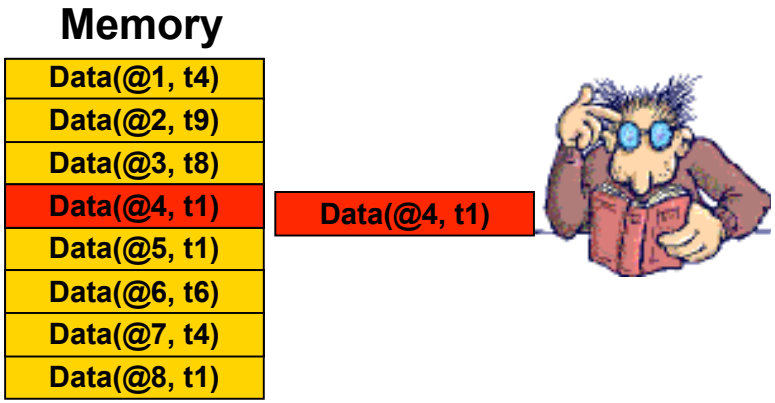


Active Attacks

- Code and data injection

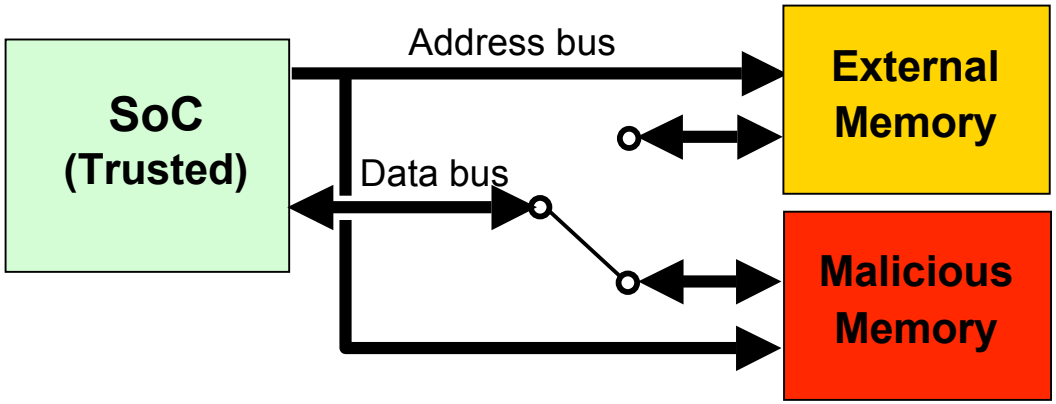


- Three kinds of active attacks are defined depending on the choice made by the adversary on the data to insert:
  - ✓ Spoofing: Random data injection
  - ✓ Splicing: Spatial permutation
  - ✓ Replay: Temporal permutation



Active Attacks

- Code and data injection



- Three kinds of active attacks are defined depending on the choice made by the adversary on the data to insert:

  - ✓ Spoofing: Random data injection
  - ✓ Splicing: Spatial permutation
  - ✓ Replay: Temporal permutation

- Attacker motivation:
  - ✓ Hijack the software execution
  - ✓ Reduce the search space for key recovery or message recovery

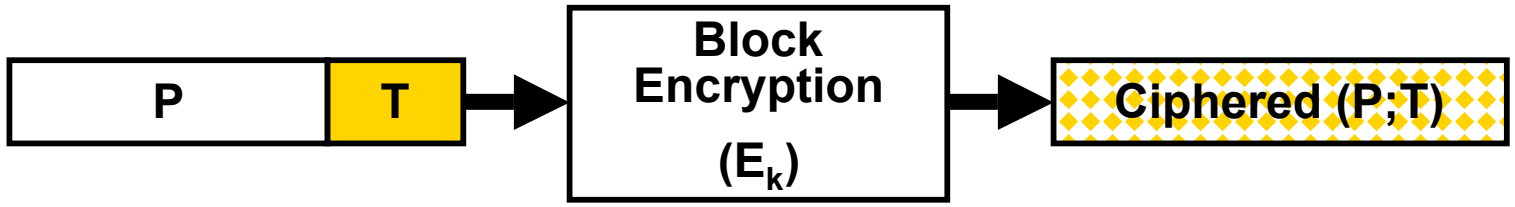
Outline

- Cryptography principles
- Attacks on embedded systems
- **Countermeasures**
  - **Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems**
    - **PE-ICE/ Extended OTP**
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

PE-ICE Principles[1]

- **PE-ICE: Parallelized Encryption & Integrity Checking Engine**
  - ✓ Only 1 pass over the data to provide both data confidentiality and integrity.
  - ✓ Tag are not computed over the data
- **Confidentiality** is ensured by **block encryption**
  - ✓ **Rijndael** (J.Daemen, V.Rijmen) **– AES** (NIST(*) standard)

- **Data integrity checking** relies on the **diffusion property** of block encryption:



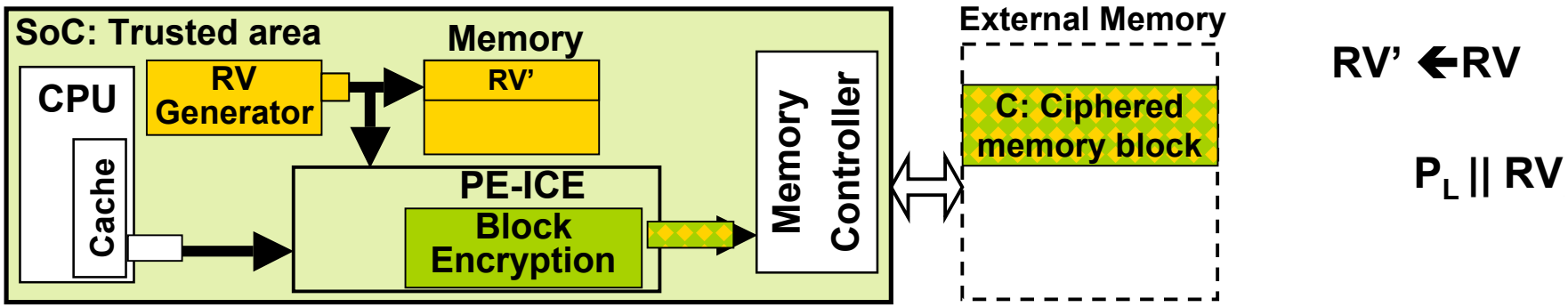
  - ✓ AREA (Added Redundancy Explicit Authentication) applied at the block level
    - ➔ Redundancy is inserted in each plaintext block **before encryption**
    - ➔ Redundancy is checked **after** each block **decryption**

(*) **NIST**: National Institute of Standard and Technology **AES**: Advanced Encryption Standard

[1]Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems, Reouven Elbaz December 2006

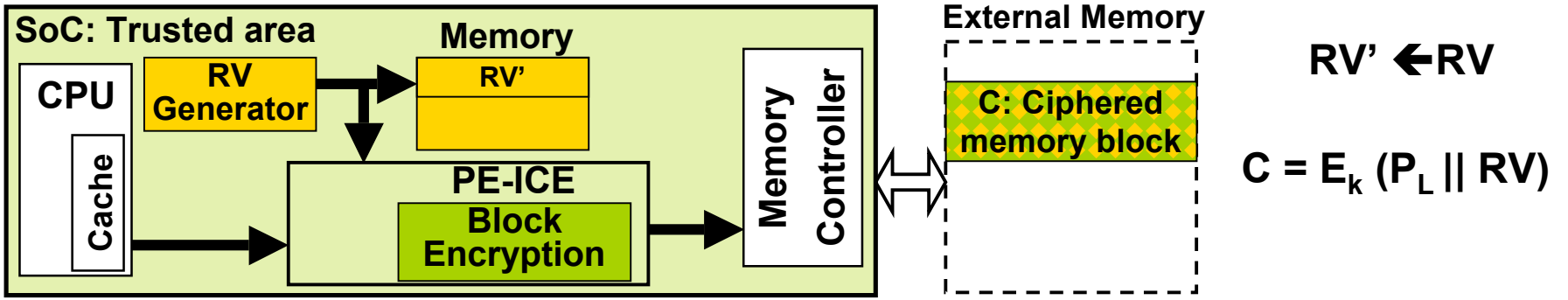
PE-ICE for Read Write Data

- **Write operations: The redundancy is added in each plaintext block**

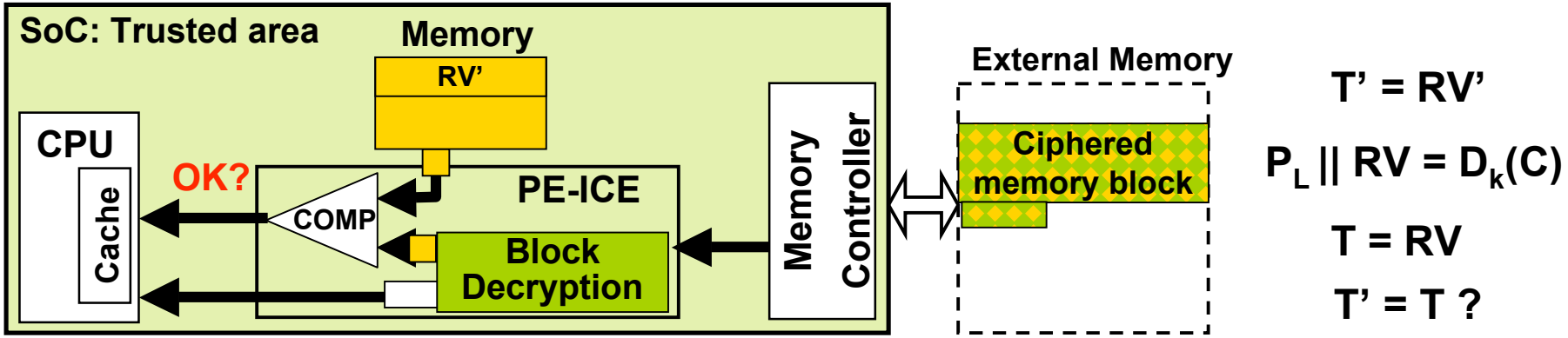


PE-ICE for Read Write Data

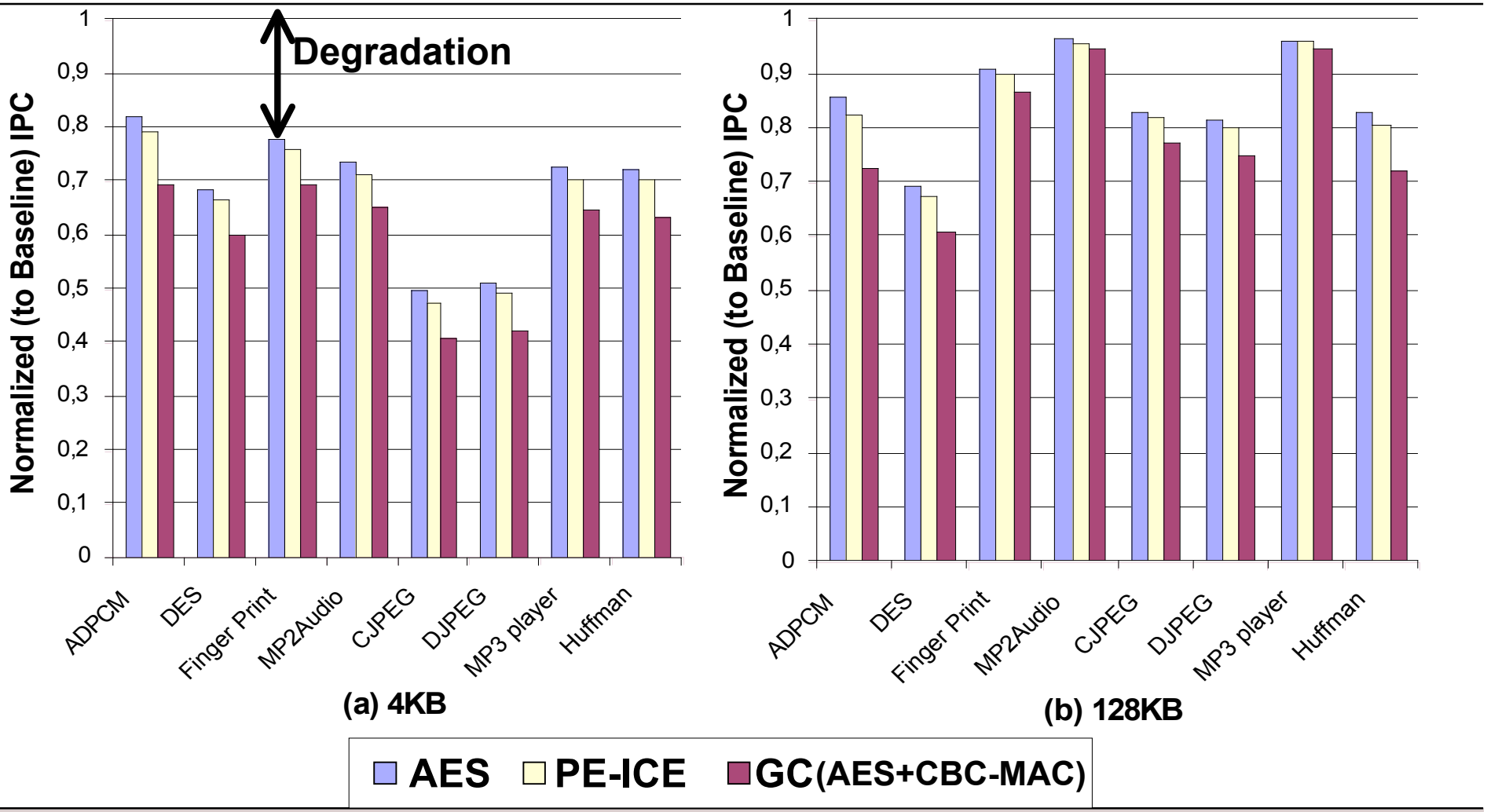
- **Write operations: The redundancy is added in each plaintext block**



- **Read operations: The redundancy is checked after decryption**



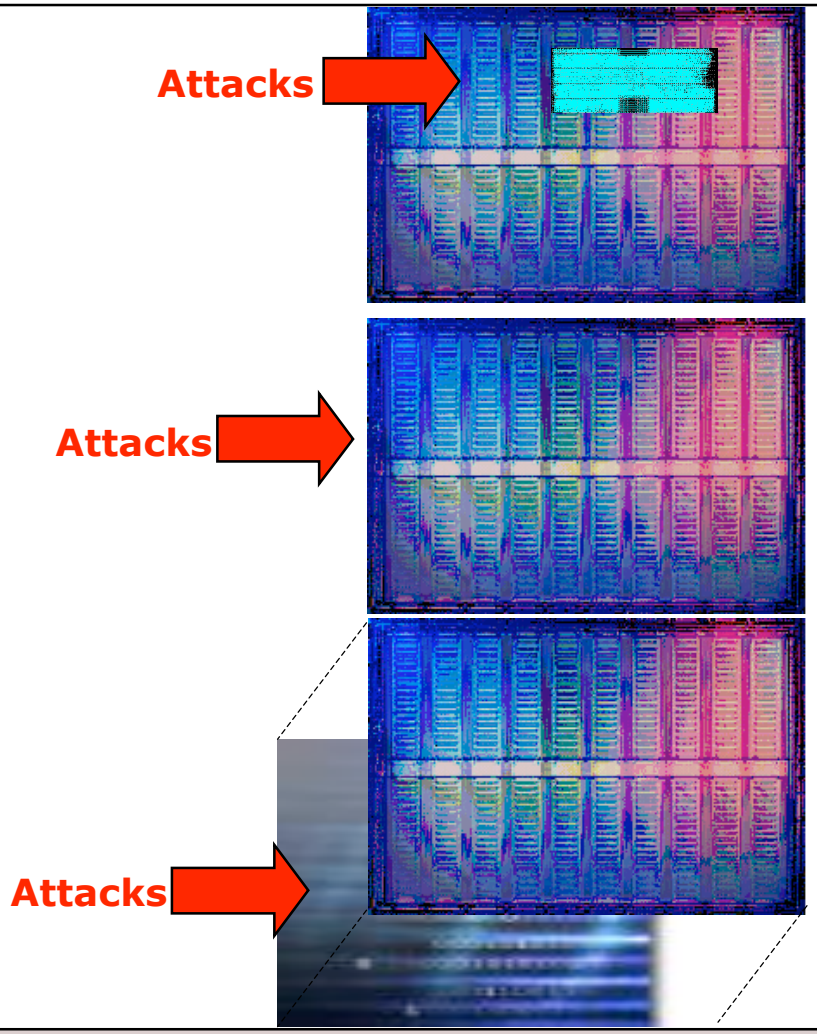
PE-ICE: Simulation Results



Outline

- Cryptography principles
- Attacks on embedded systems
- **Countermeasures**
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - **Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream**
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

Configurable Computing Security Space

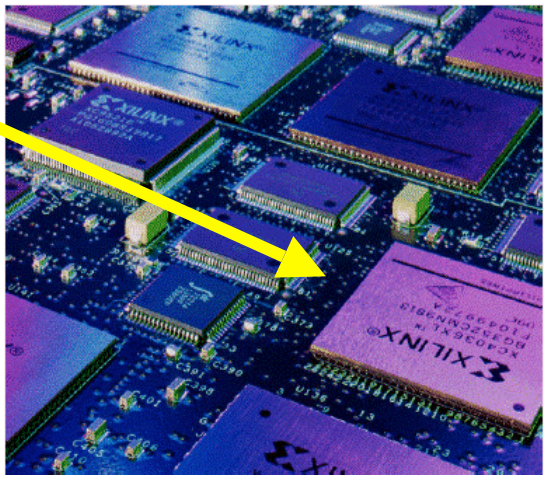


- **Configurable Security Primitive**
  - Use configurable computing primitive to protect a system, the module is seen as an agile hardware unit
- **Secure Configurable System**
  - The whole system is configurable. The security is provided by the agility of the whole system
- **Configurable Design Security**
  - Protect the configurable computing configuration

SRAM FPGA

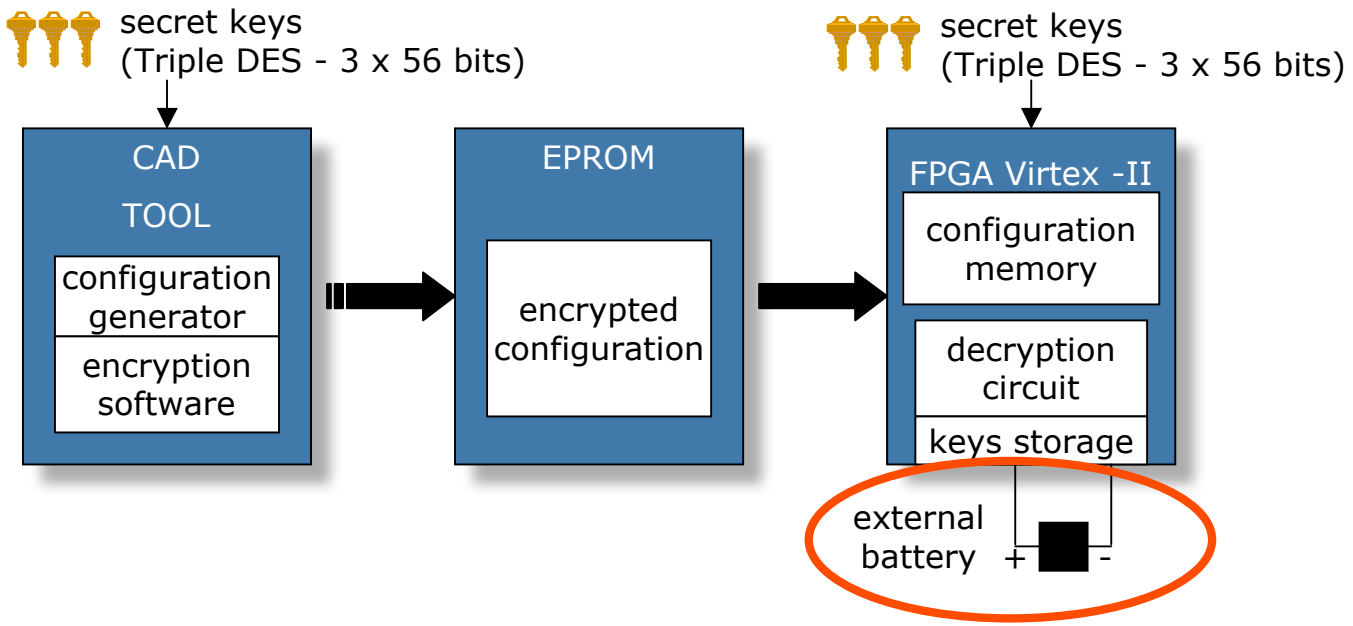
- Really reconfigurable !
- Need of a bitstream transfer upon power-on, security sensitive

The pirate can "read" the bitstream



Solution: Bitstream encryption ...

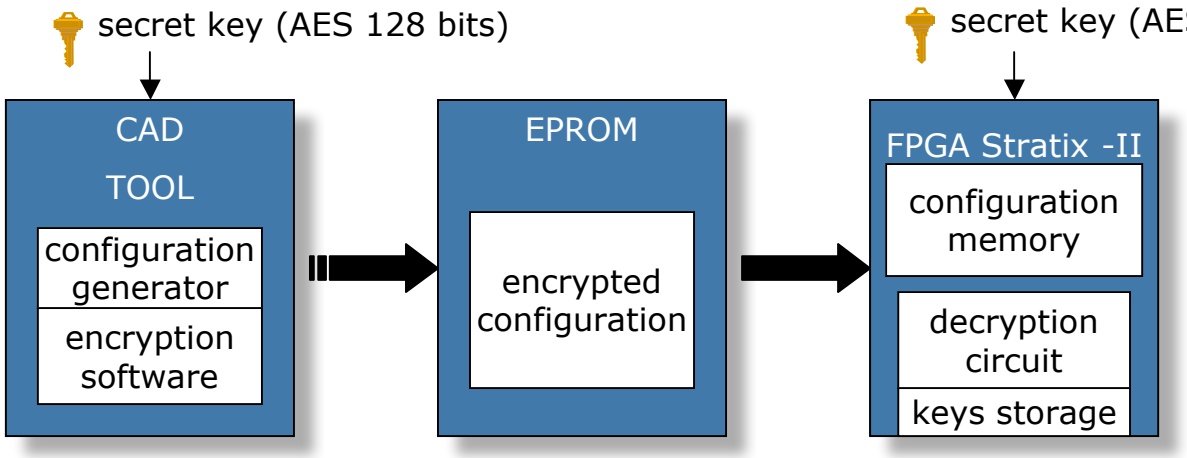
Xilinx Solution



Protection against cloning and reverse engineering

- Need of an external battery to save the keys
- The decryption circuit takes FPGA resources (silicon)…
- No flexibility for the decryption algorithm
- Partial reconfiguration is no more available

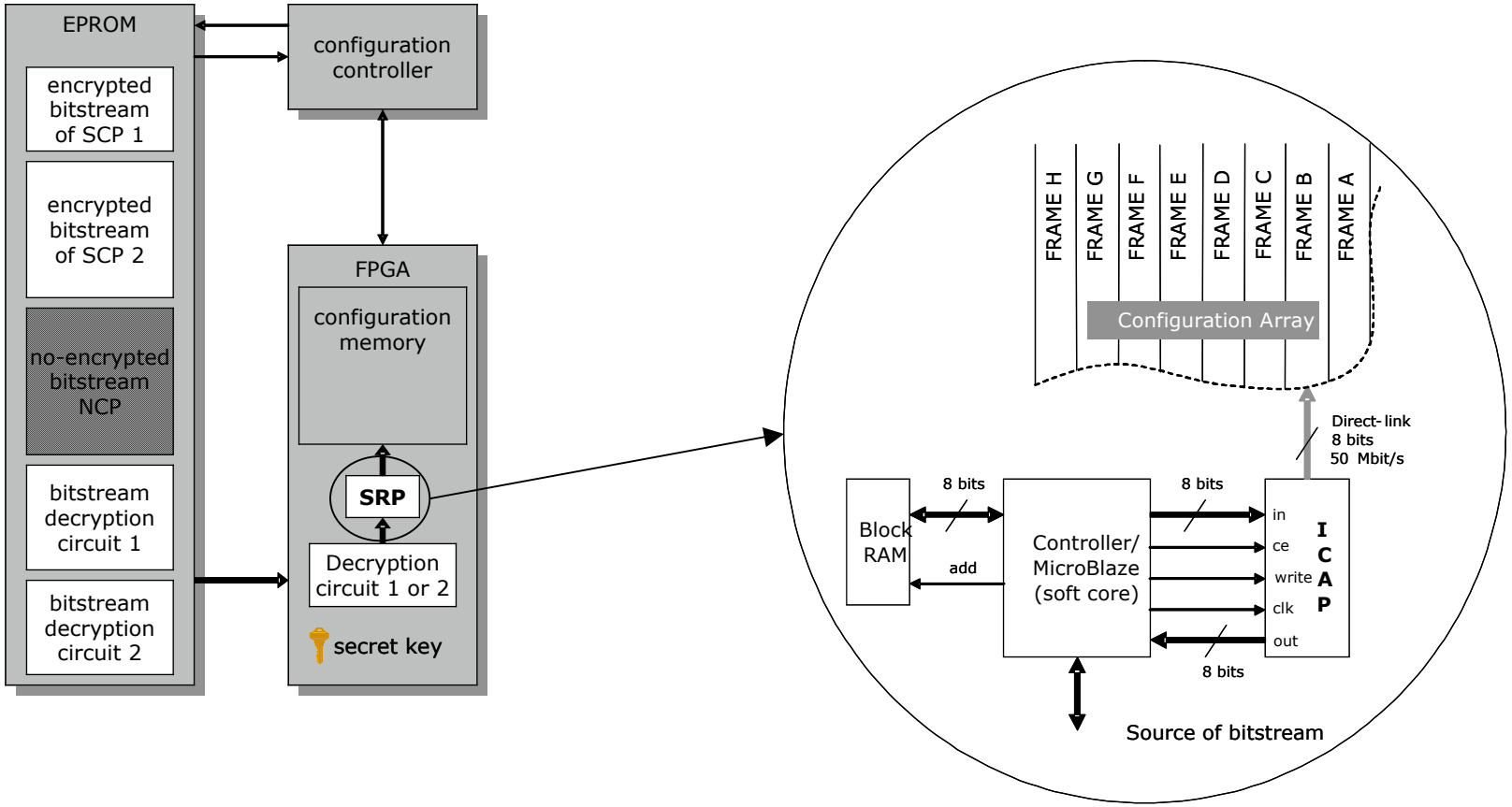
Altera Solution



- The decryption circuit takes FPGA resources (silicon)…
- No flexibility for the decryption algorithm

LESTER/UMASS Solution

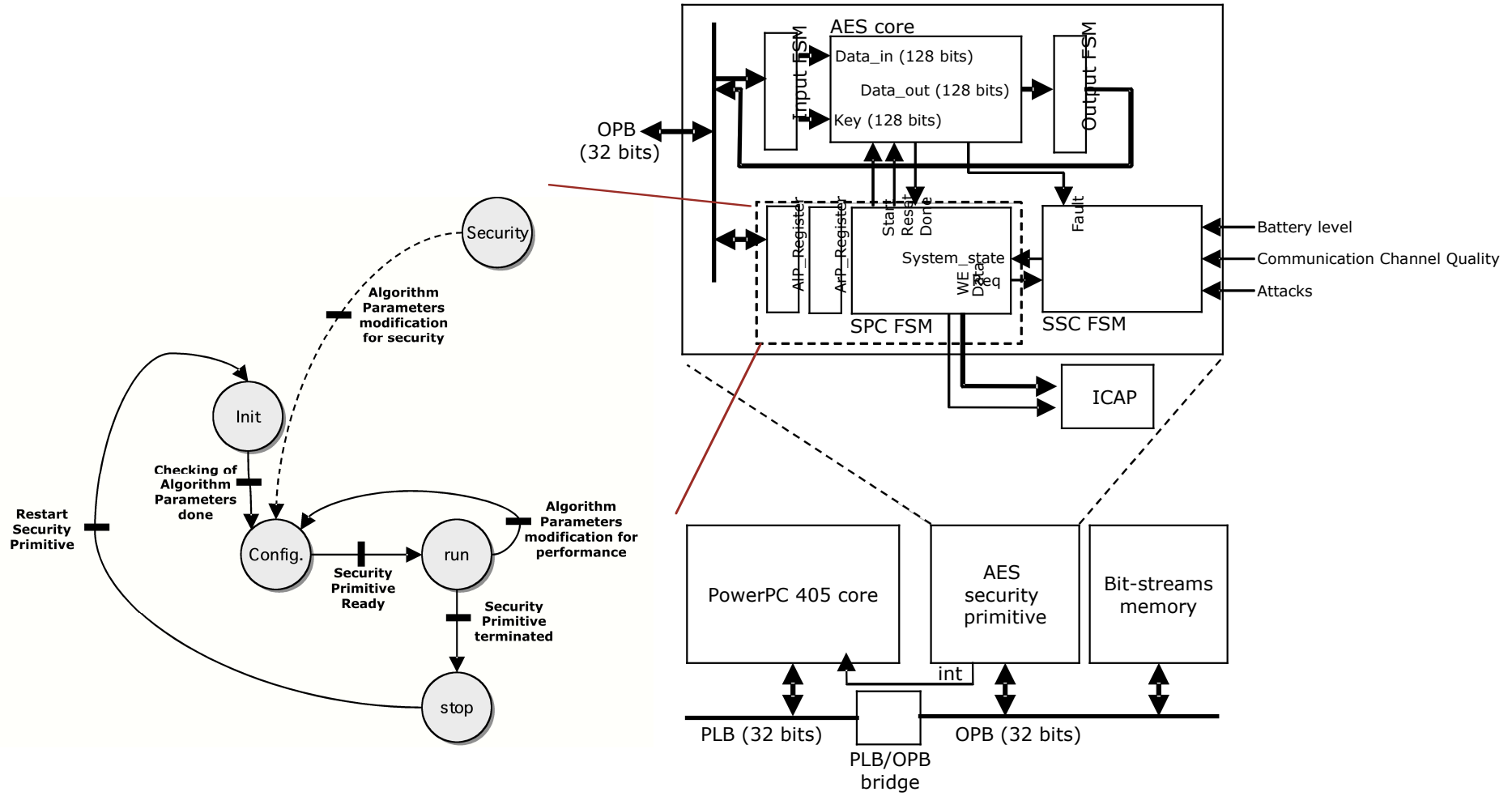
- Dynamic security of the bitstream for SRAM FPGA



Outline

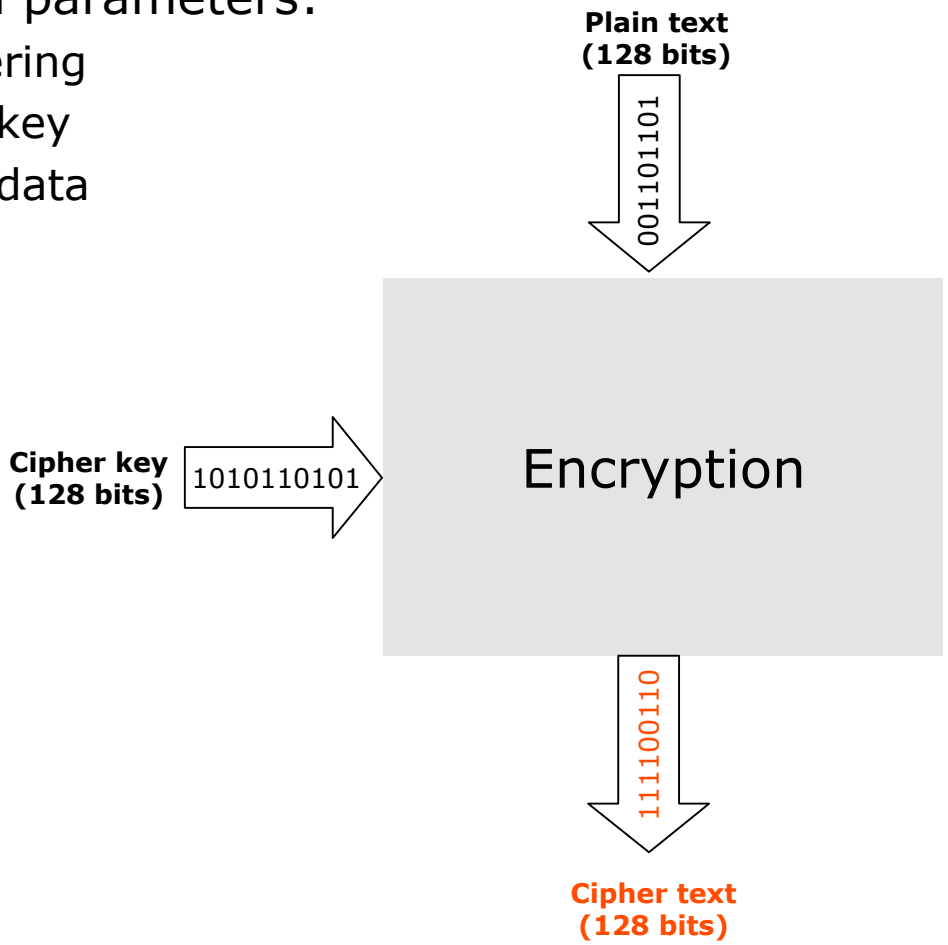
- Cryptography principles
- Attacks on embedded systems
- **Countermeasures**
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - **Security Architecture for Embedded Systems: SANES**
    - **Security primitive: AES case study on Virtex-II Pro**
  - Existing solutions: Secure Coprocessor/Microcontroller
- Conclusion

AES Platform: case study with a virtex-II Pro



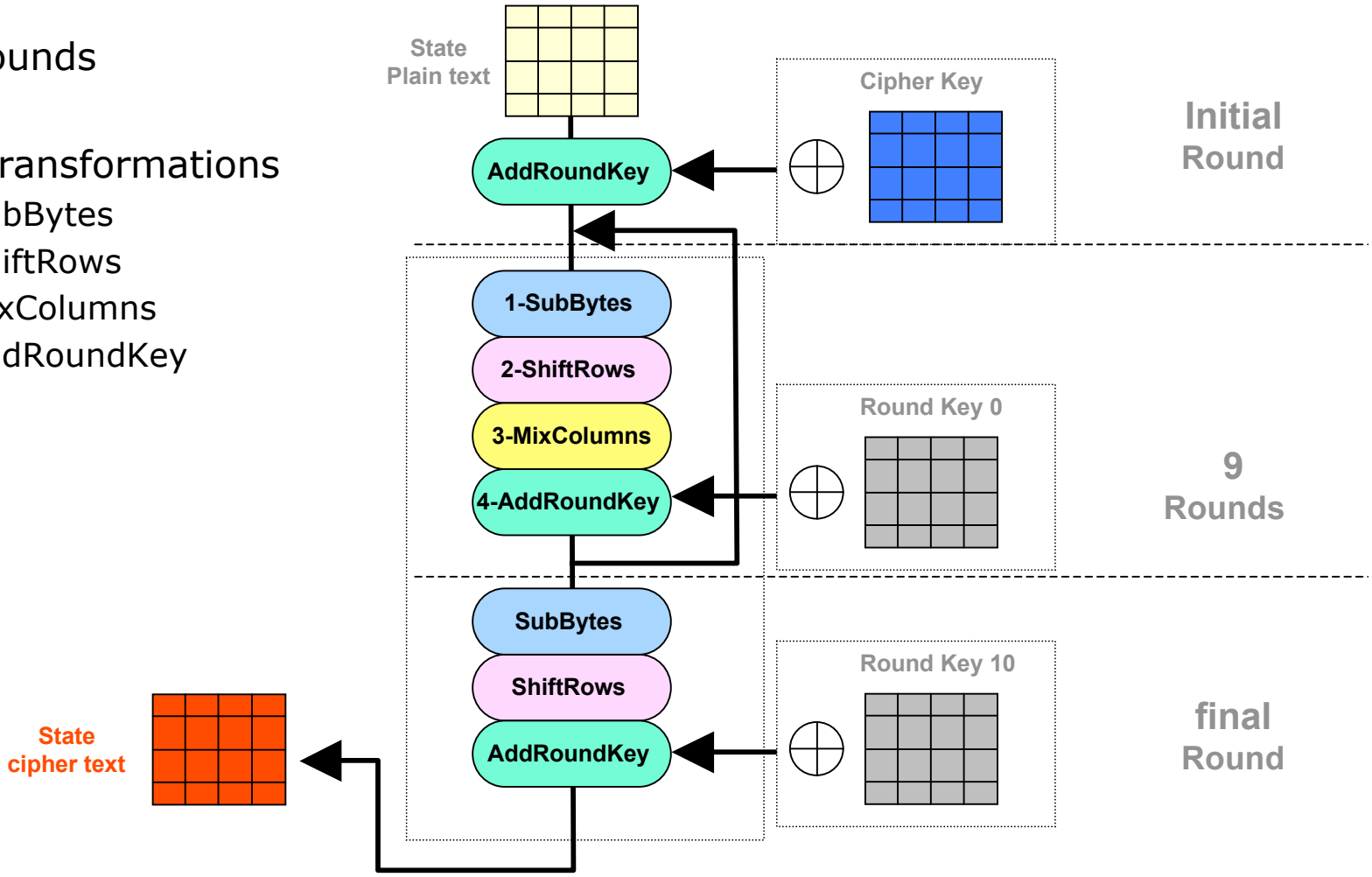
Case study: AES algorithm

- Rijndael algorithm parameters:
  - Data block ciphering
  - 128 bits for the key
  - 128 bits for the data



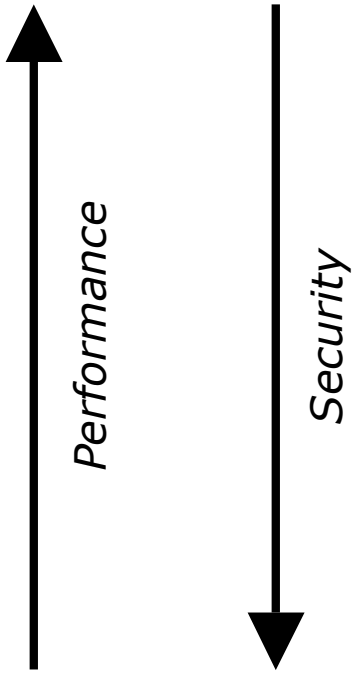
AES algorithm: encryption process

- Ten rounds
- Four transformations
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey

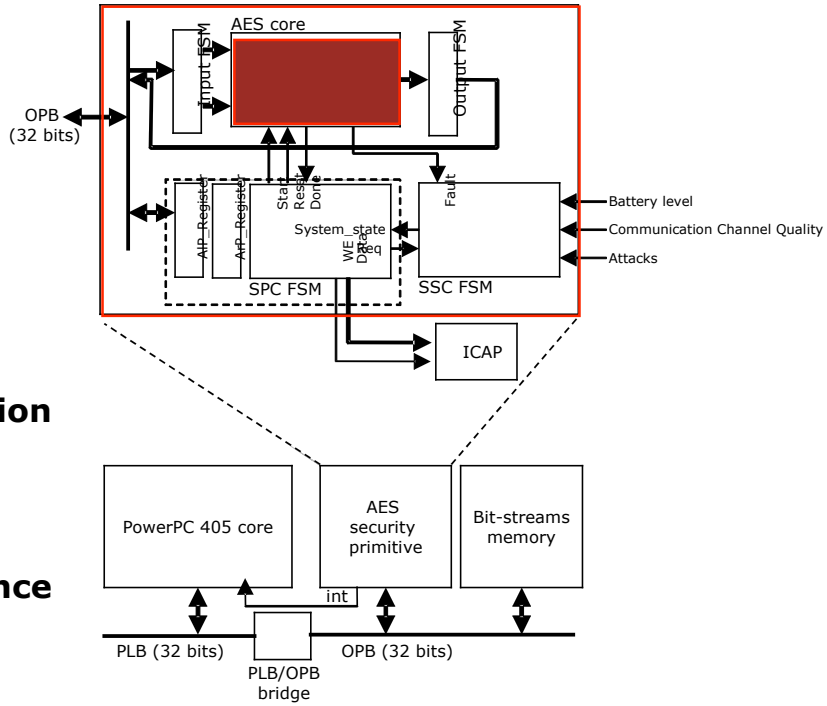


AES implementations – security primitive core

- Four implementations are considered for the Rijndael algorithm:

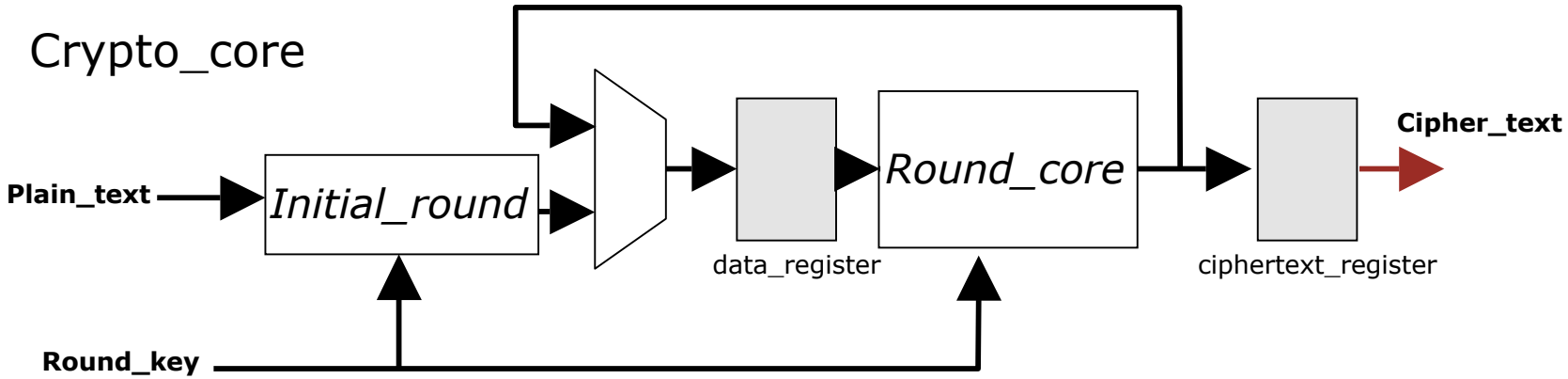


- **Non feedback mode without security** (N_FB)
  - *Pipeline*
- **Feedback mode without security** (FB)
  - *Iterative*
- **Feedback mode with fault detection** (FB_FD)
  - *Parity-based error detection*
- **Feedback mode with fault tolerance** (FB_FT)
  - *Triple module redundancy technique*

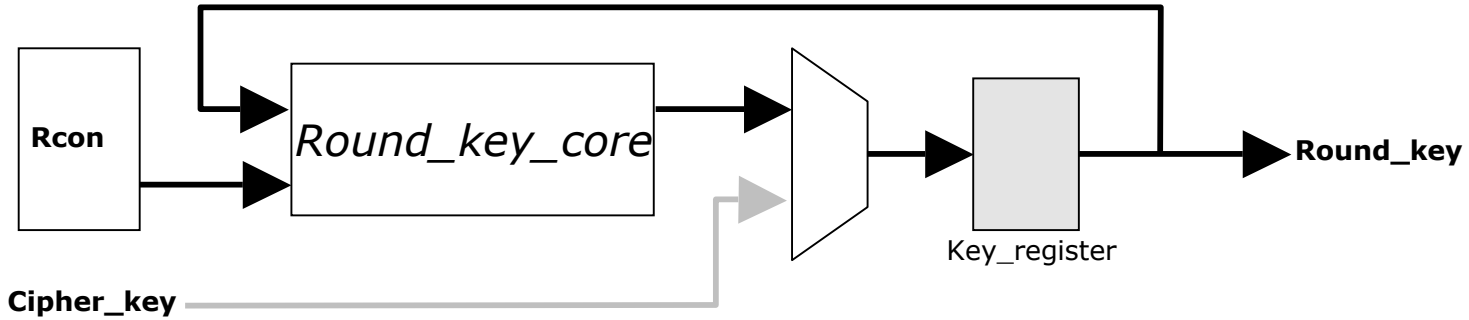


AES feedback mode without security (2/2)

- Crypto_core

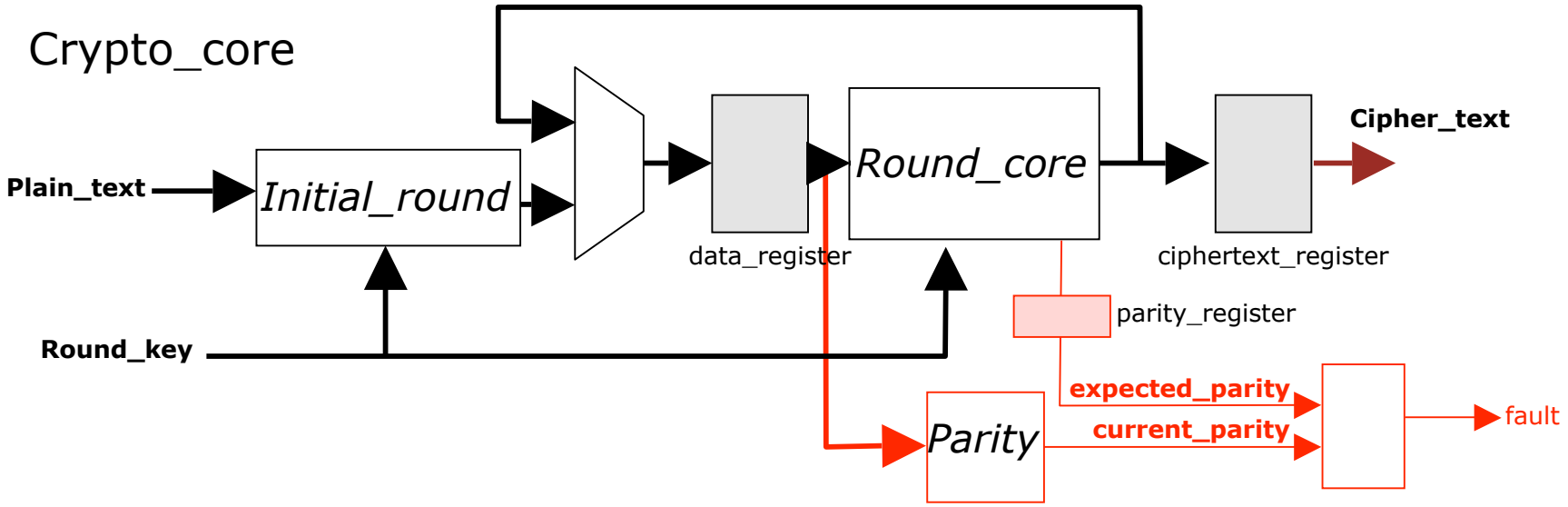


- Key_gene

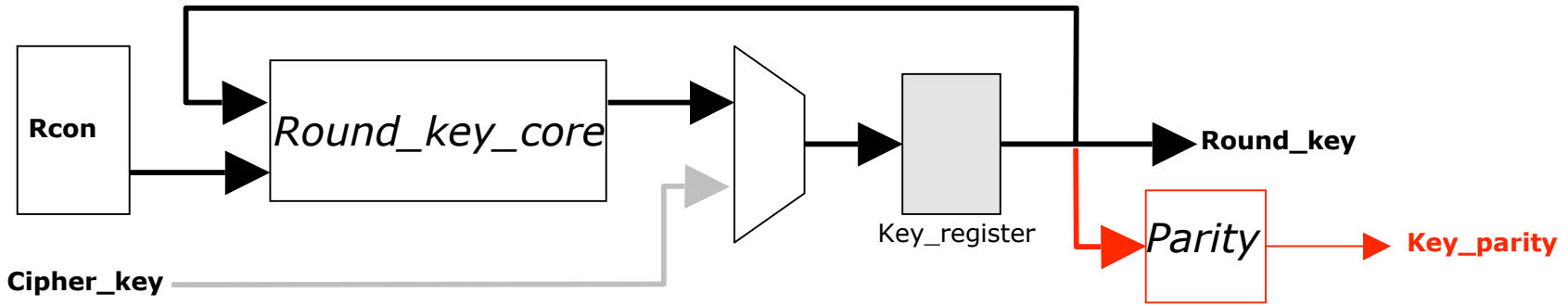


AES feedback mode with fault detection (2/3)

- Crypto_core

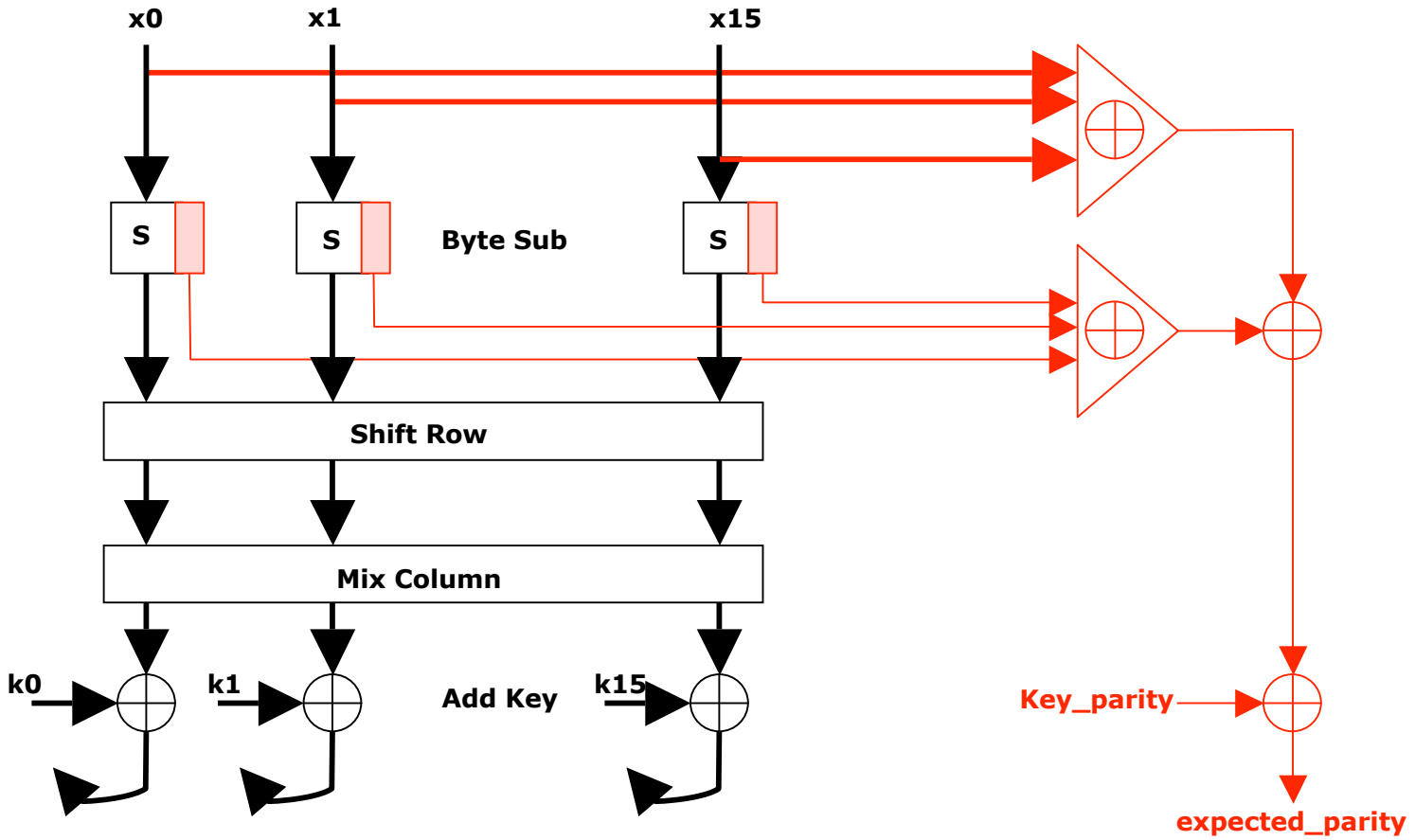


- Key_gene



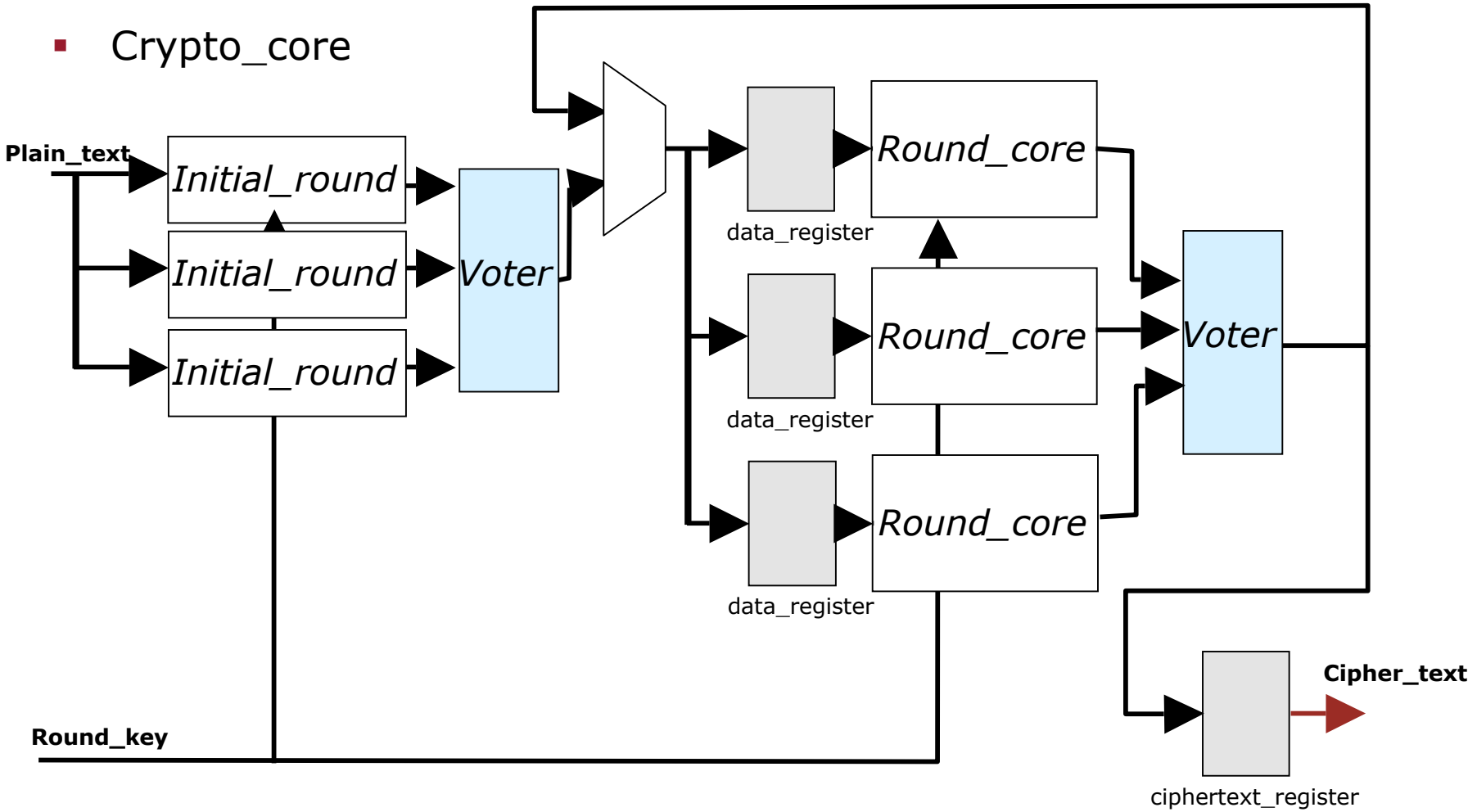
AES feedback mode with fault detection (3/3)

- Round_core

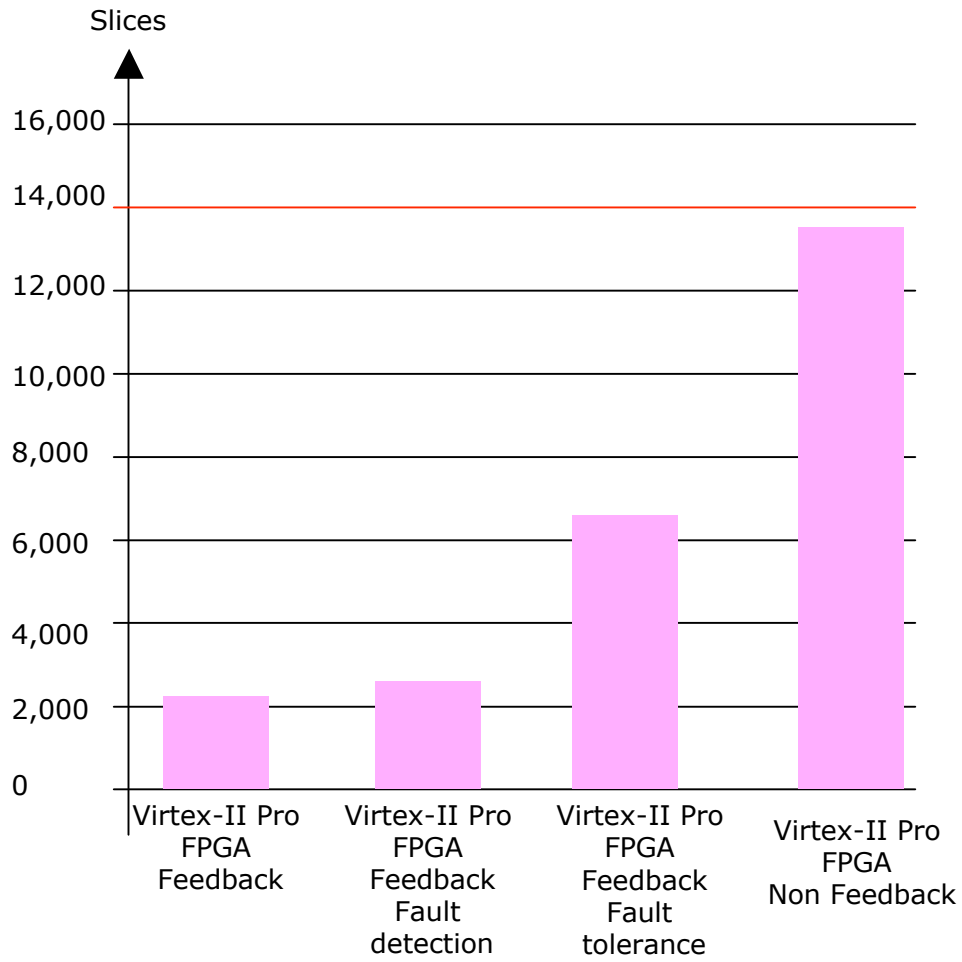


AES feedback mode with fault tolerance (2/3)

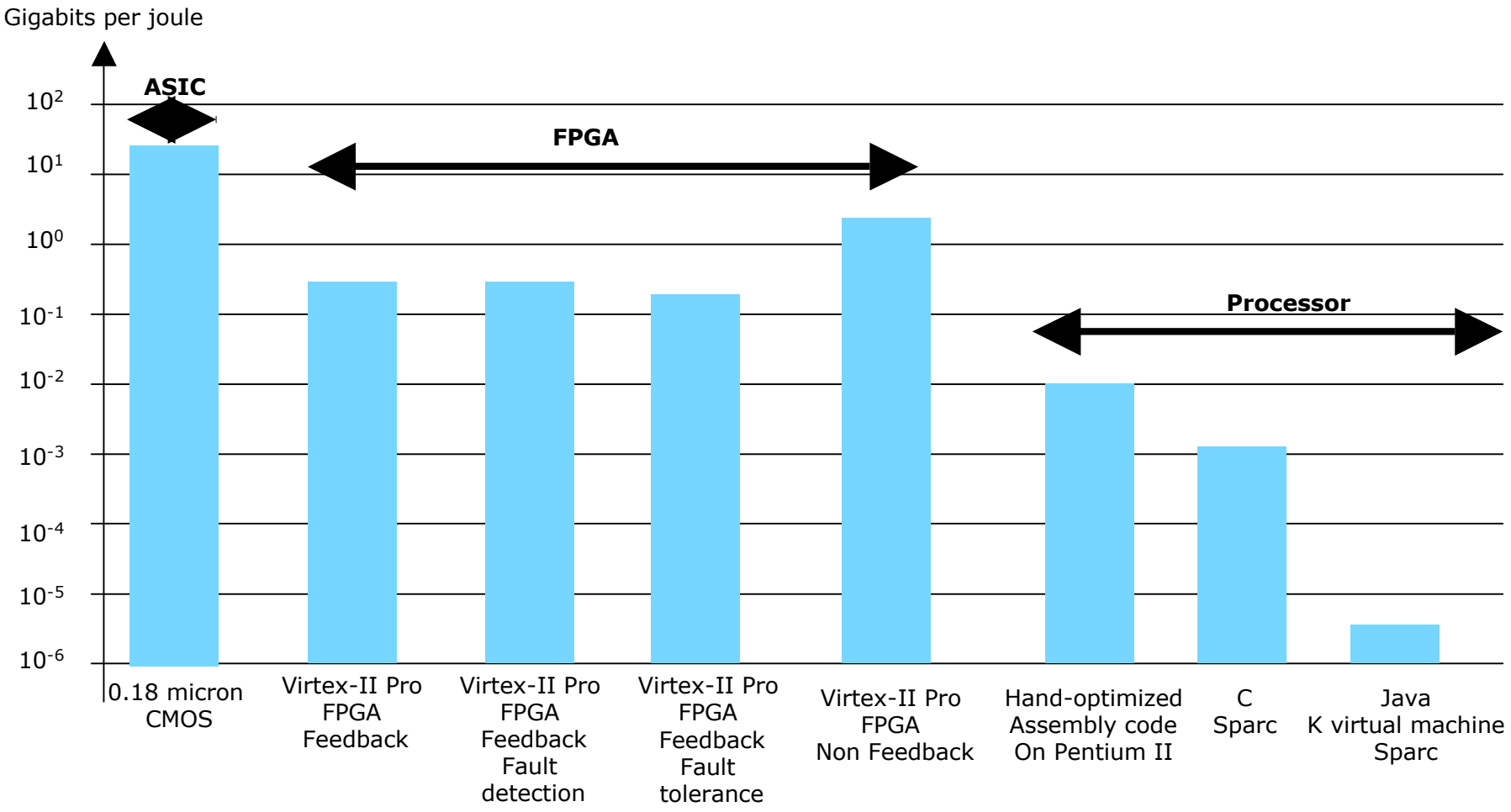
- Crypto_core



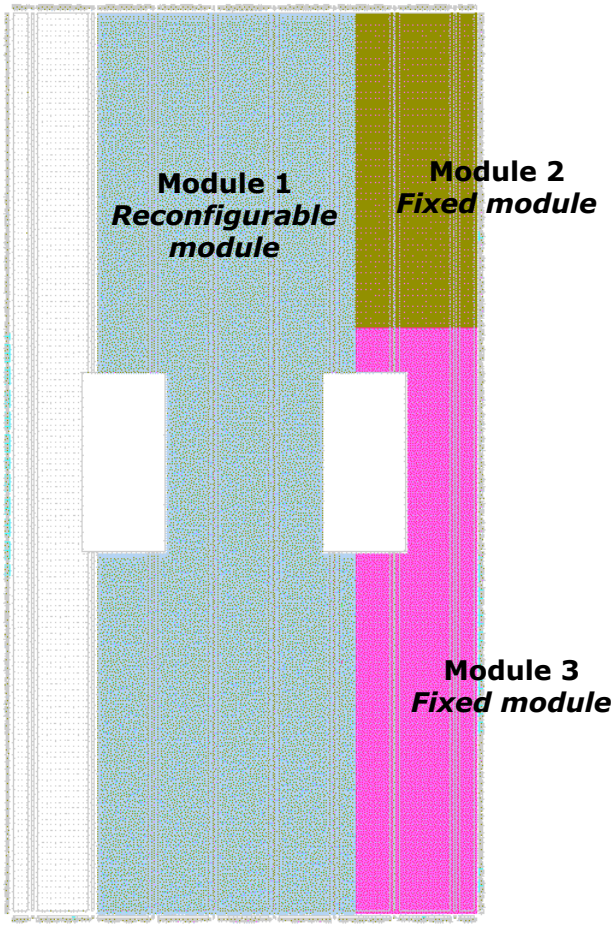
Area and power comparison of Rijndael implementations

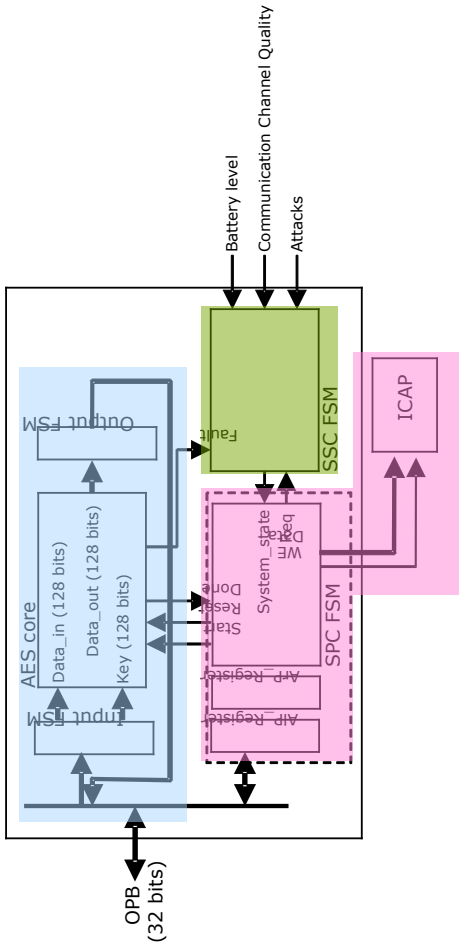


Energy efficiency of Rijndael implementations



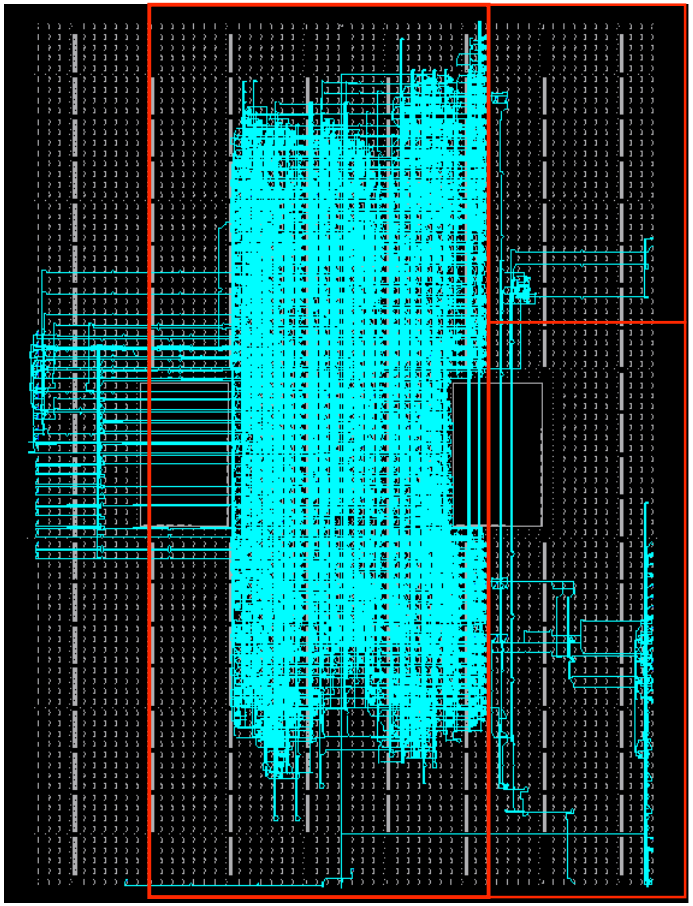
Security primitive: constraints





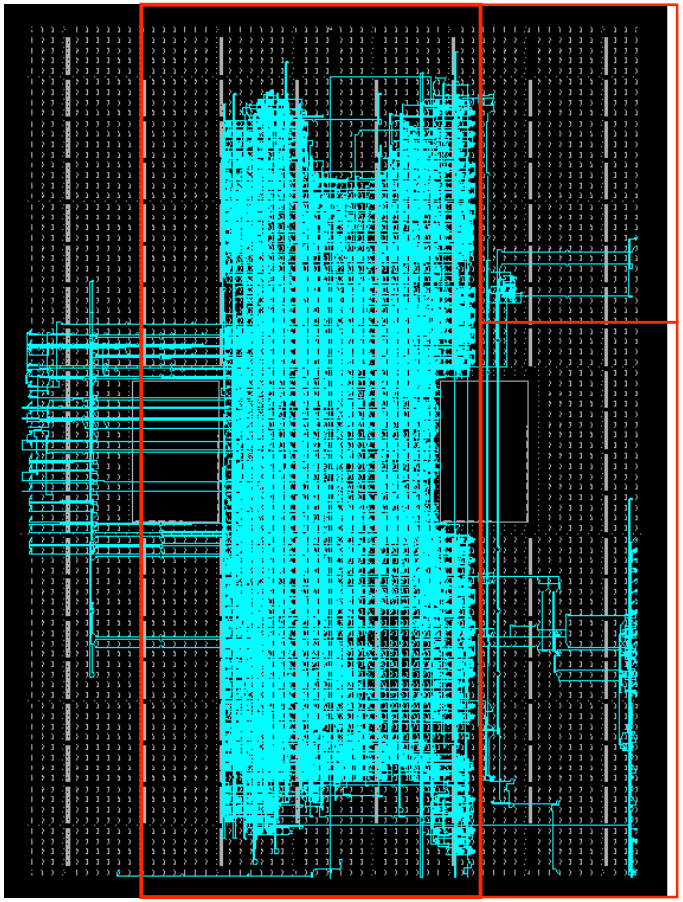
Virtex-II Pro xc2vp30-5ff896

Security primitive: feedback mode without security



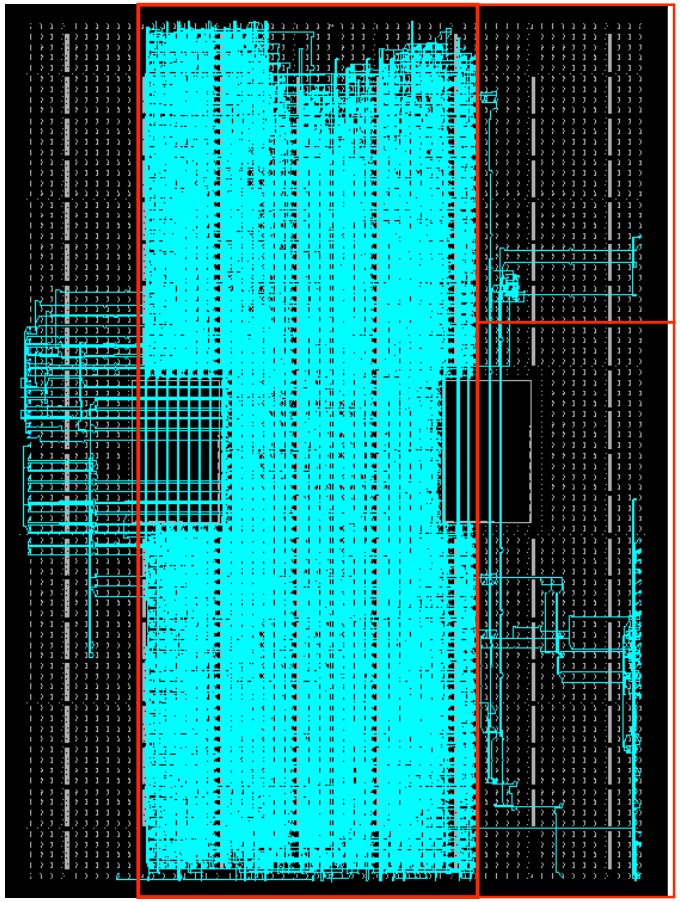
- Feedback mode without security
  - Security primitive core
    ~2000 slices (16%)
  - Security primitive controller
    ~50 slices (1 %)
  - System security controller
    ~50 slices (1 %)
- The core does not embed any security mechanisms
- "Low cost" solution

Security primitive: feedback mode with fault detection



- Feedback mode with fault detection
  - Security primitive core
    ~2000 slices (16%)
  - Security primitive controller
    ~50 slices (1 %)
  - System security controller
    ~50 slices (1 %)
- The core embeds fault detection mechanism
- "Low cost" solution
- Best tradeoff in term of security vs. performance
- Does not protect against denial of service attacks

Security primitive: feedback mode with fault tolerance



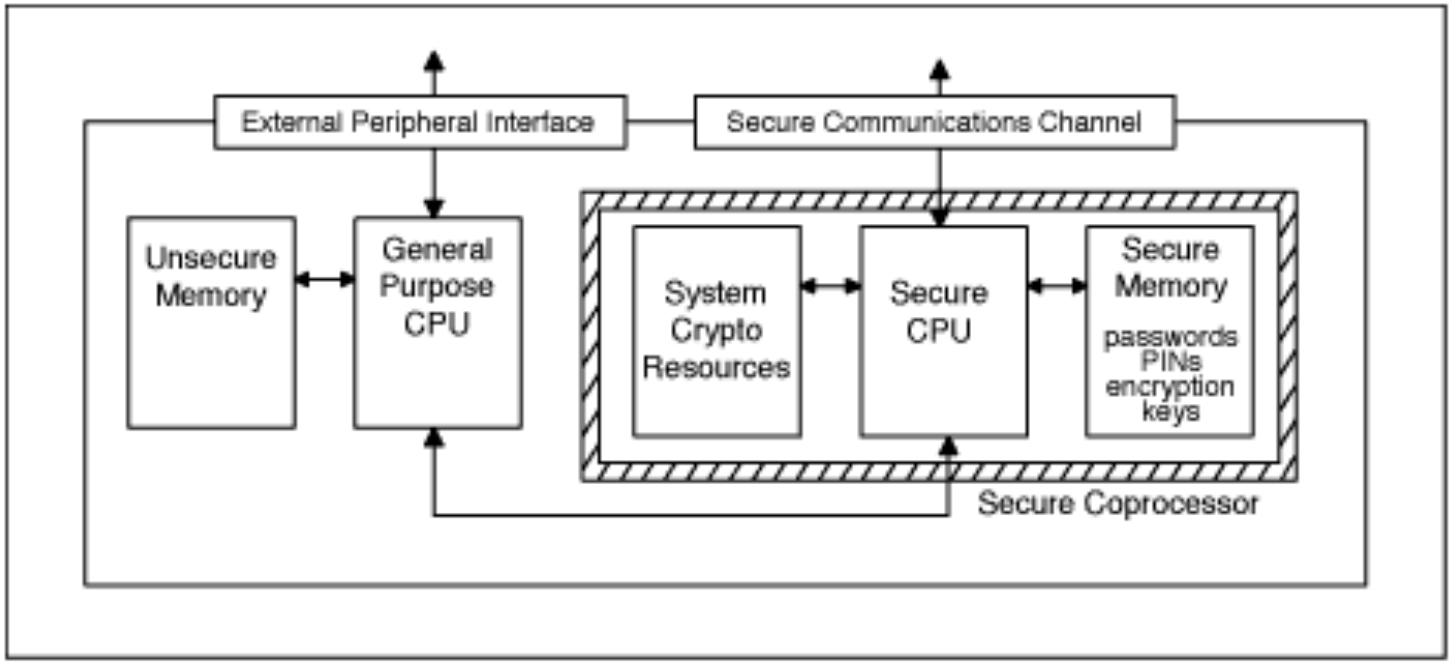
- Feedback mode with fault tolerance
  - Security primitive core
    ~6000 slices (46%)
  - Security primitive controller
    ~50 slices (1 %)
  - System security controller
    ~50 slices (1 %)
- The core embeds fault tolerance mechanism
- "high cost" solution
- Most reliable solution
- Provides the most efficient protection

Outline

- Cryptography principles
- Attacks on embedded systems
- **Countermeasures**
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - **Existing solutions: Secure Coprocessor/Microcontroller**
- Conclusion

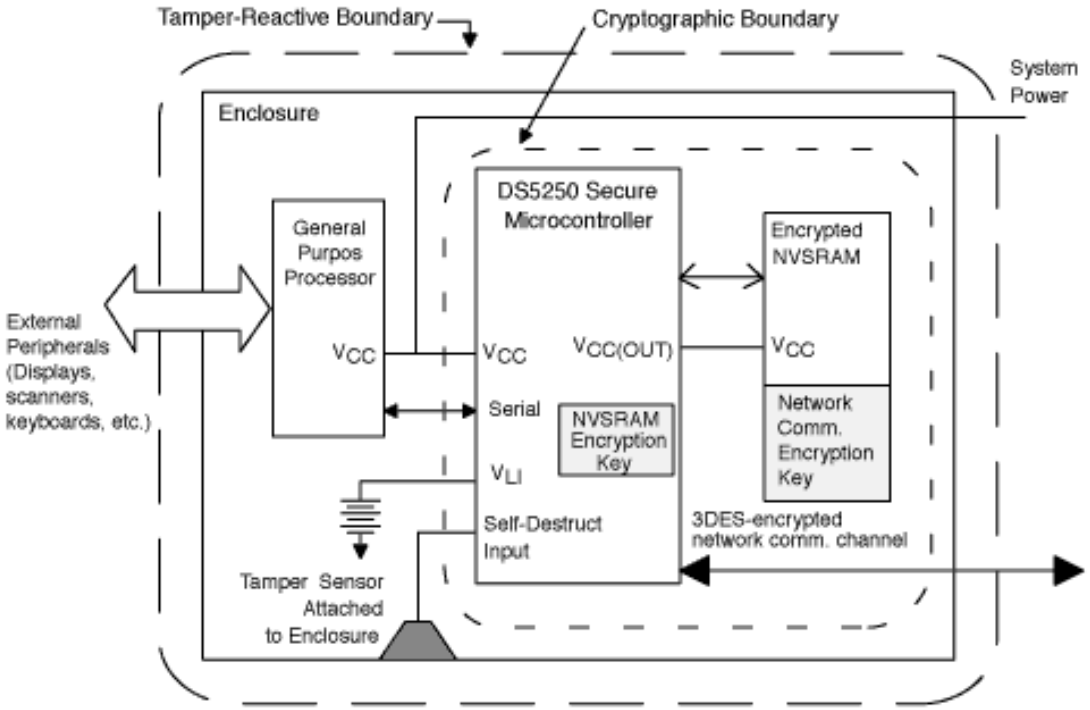
Secure Coprocessor/Microcontroller

- Ensure the security of the system
  - Chip resistant against attacks (invasive, non invasive, side channels)
  - Microcontroller embeds ciphering cores and keys generator
  - Secure memory with encrypted data



Secure microcontroller: example Dallas DS500 2FP

- Embedded RNG (generation of keys)
- NVSRAM (Non Volatile SRAM) for storage of ciphering
- DES (64 bit) ciphering for memory protection
- Self Destruct Input if an external device detect an attack and rises this input
- Two layers of metallization added on the top of the layout to increase reverse engineering difficulty

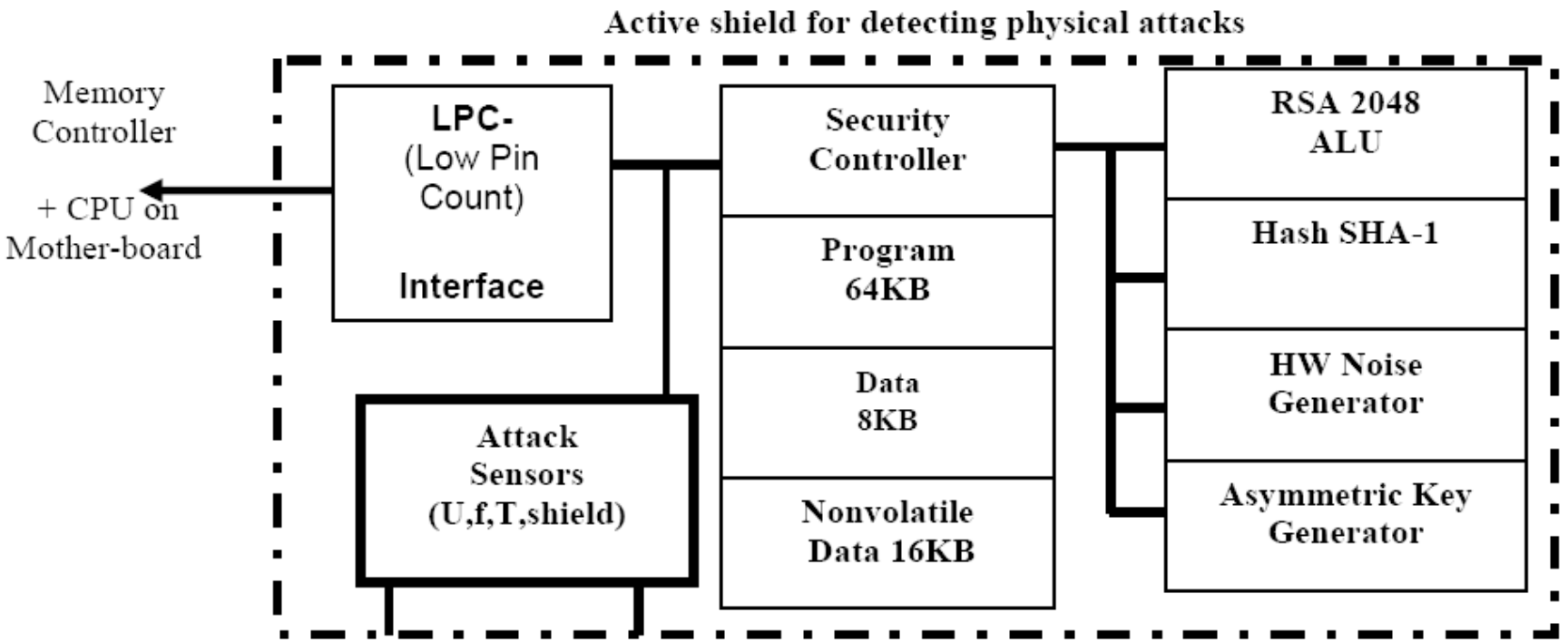


DALLAS SEMICONDUCTOR MAXIM

Trusted Platform Module

- Architecture
  - Processor dedicated for security
  - Asymmetric cryptography (2048 bits) RSA: data confidentiality
  - Hashing functions SHA-1/-2
  - RNG for keys generation
  - EEPROM non volatile: storage of secret keys
- Hardware security
  - Countermeasure against power and timing attacks
  - Sensors: frequency, voltage, temperature, light et glitch (clock)
  - Auto-tests functions

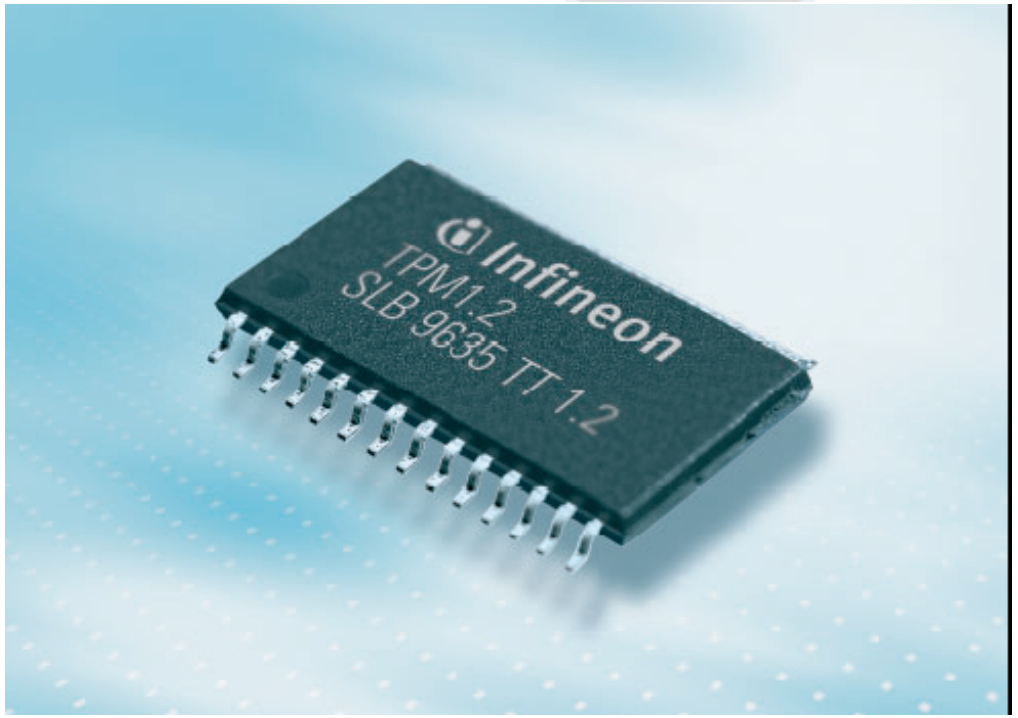
Trusted Platform Module - Architecture



Trusted Platform Module – Example for computer

- Infineon TPM 1.2 SLB 96 35 TT 1.2

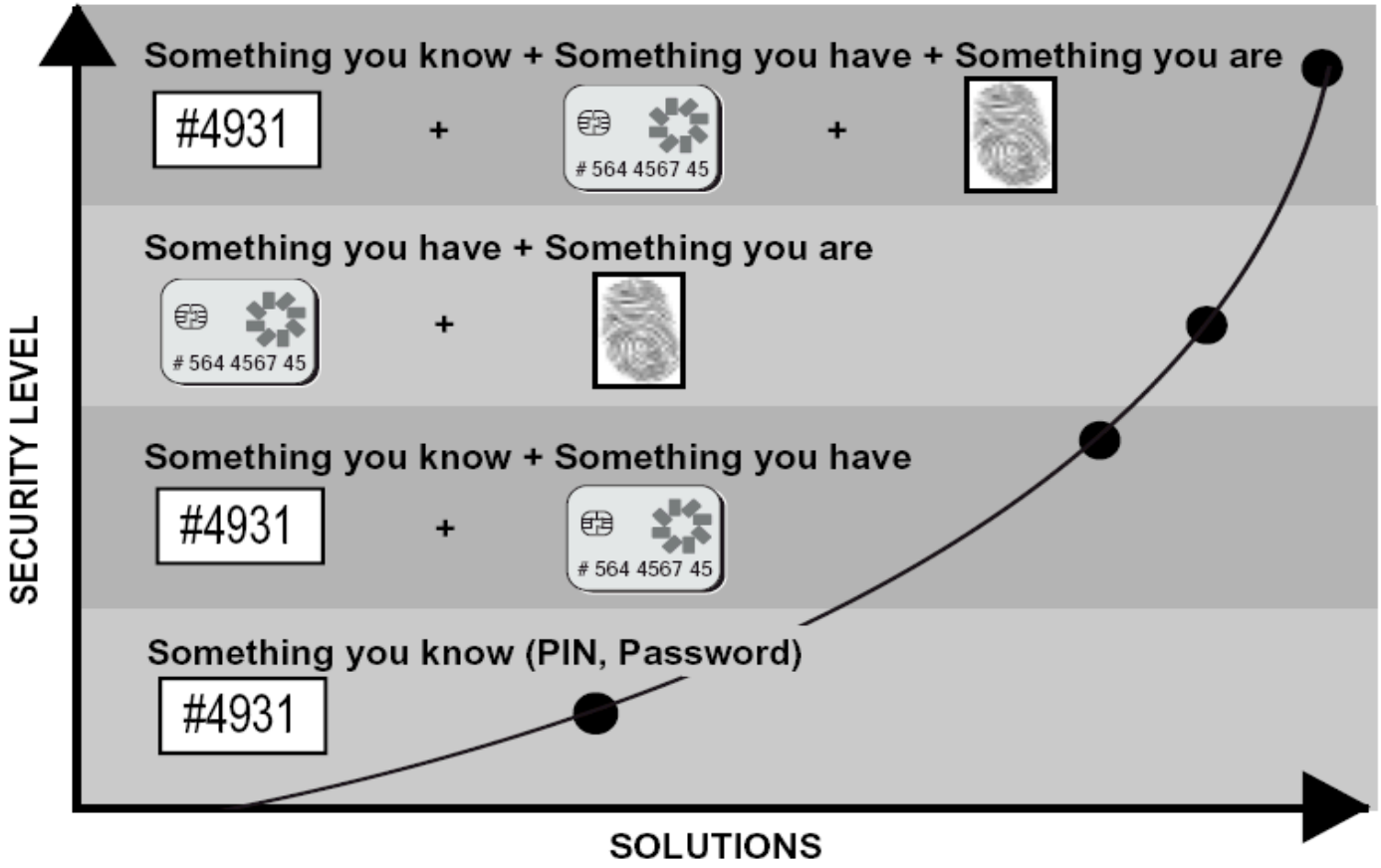
  - http://www.infineon.com/
  - Microcontroller 16 bits
  - Technology CMOS 0,22 µm



Outline

- Cryptography principles
- Attacks on embedded systems
- Countermeasures
  - Hardware Mechanisms for Secured Processor-Memory Transactions for Embedded Systems
    - PE-ICE/Extended OTP
  - Preventing Piracy and Reverse Engineering of SRAM FPGAs Bitstream
  - Security Architecture for Embedded Systems: SANES
    - Security primitive: AES case study on Virtex-II Pro
  - Existing solutions: Secure Coprocessor/Microcontroller
- **Conclusion**

Security and people, how it will evolve???



Security is a big and a complex issue

- Security deals with
  - Computer science
  - Electronic, computer engineering
  - Telecommunication (protocols)
  - People
  - Companies
  - Curriculum … (new curriculum should be considered)

- Security is a large domain that does not only focus on technology

  - Require a more global thinking on our society
  - Need to be considered by engineers…

Conclusion

- Security is becoming a critical problem in our society
  - Related cost is more and more important
  - Number of attacks keeps increasing (software but also hardware)
  - More and more embedded systems mobile and connected (embedding more and more personal data)
  - Strong threats at the hardware level
- Cryptography algorithms are the pillars of security… But security is a more complex problem
  - Software and hardware protections
  - Security policy: the right security level at the right time
  - Lot of work still to be done to provide some CAD tools to build secure architectures/platforms
  - Nothing should be neglected… the threat is where you are not expected to find it

Before ending the presentation…

- Thanks to Kris Gaj
  - http://ece.gmu.edu/faculty_info/gaj.html
- Lilian Bossuet
  - http://www.lilianbossuet.com/
- And Lionel Torres (Reouven Elbaz)
  - http://www.lirmm.fr/~torres

Thanks...

More data available at:
http://python.ecs.umass.edu/%7Eessg/home.html

guy.gogniat@univ-ubs.fr