LIRMM

Low-Cost Self-Test of Crypto Devices

G. DiNatale, M. Doulcier, M-L. Flottes, B. Rouzeyre

WDSN 2008

Motivation

- Secure circuits testing
  - ✓ Scan path
    - High fault coverage
    - Automatic generation of scan chains
    - Easy test sequence generation
  - ✓ Vulnerability
    - Control and observation of internal states of CUT
    - => secret data retrieval

Scan based attack DES [Yan et al., ITC 04]
AES [Yan et al., IEEE TCAD 06]

→ **BIST**

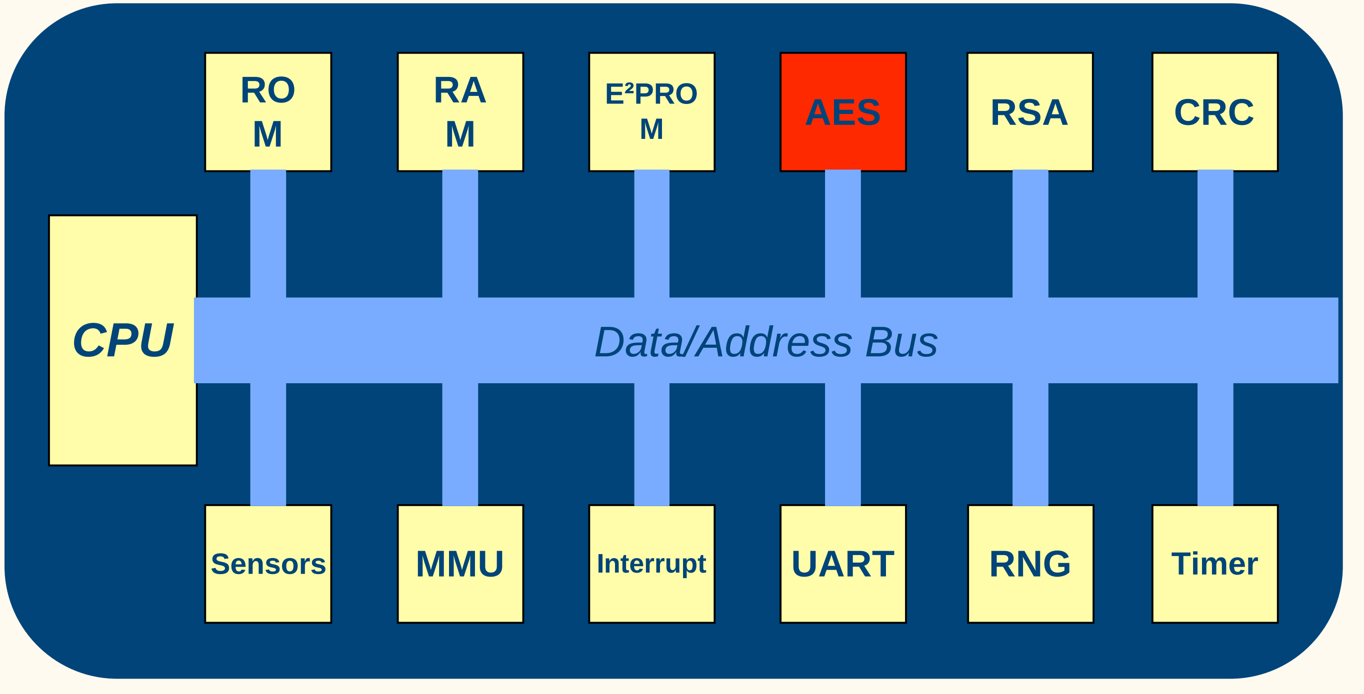
Motivation

- BIST
  - ✓ Reduced ATE cost
  - ✓ In-situ testing
  - ✓ Reduced external access

- But
  - ✓ Circuitry overhead
    - test controller
    - pattern generator
    - signature analyzer…

Motivation

- Secure circuits contain a crypto core
- E.g. Smart cards



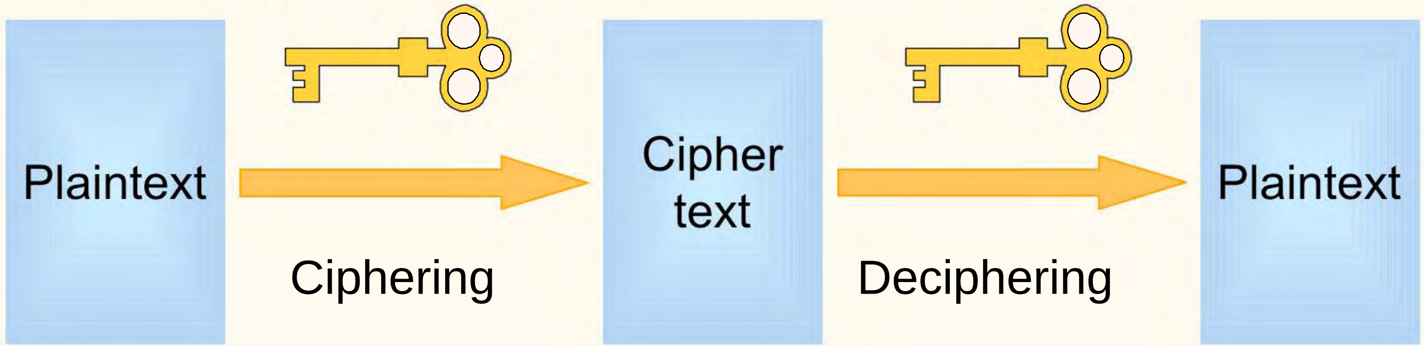
- Crypto core => Test resource

Outline

- AES & DES
  - ✓ Algorithm & architecture
  - ✓ Testability issues
- AES/DES as pattern generators
- AES/DES Self test
- Optimisations
- Conclusion

Introduction

- Symetric cryptography



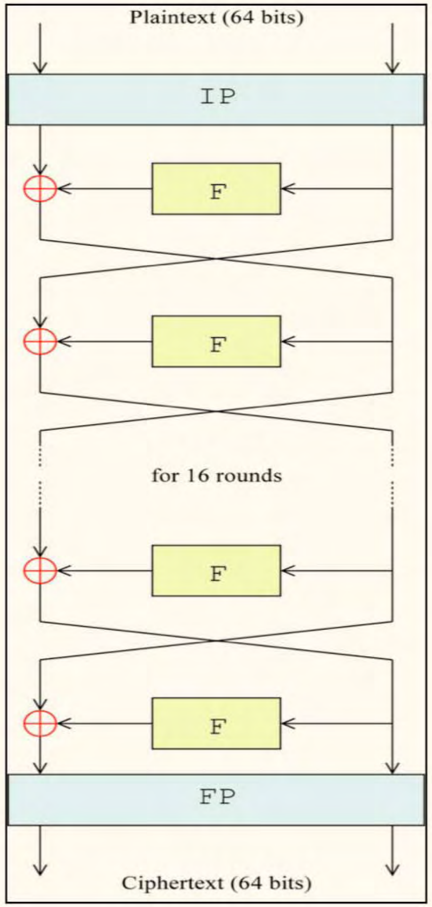
- DES
  - ✓ Adopted as standard in 1976
  - ✓ Data : 64 bits, Key : 56 bits
- AES : Advanced Encryption Standard
  - ✓ Adopted as standard in 2001
  - ✓ Data: 128 bits, Key: 128 bits (192, 256)

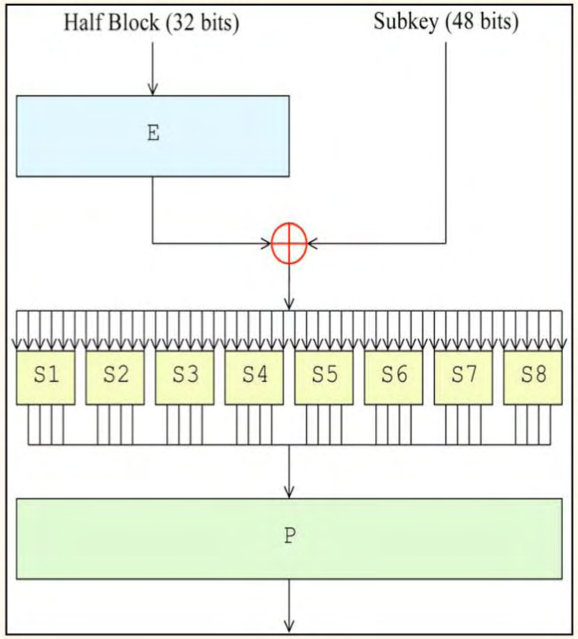
- Crypto algorithms basis: Diffusion & Confusion

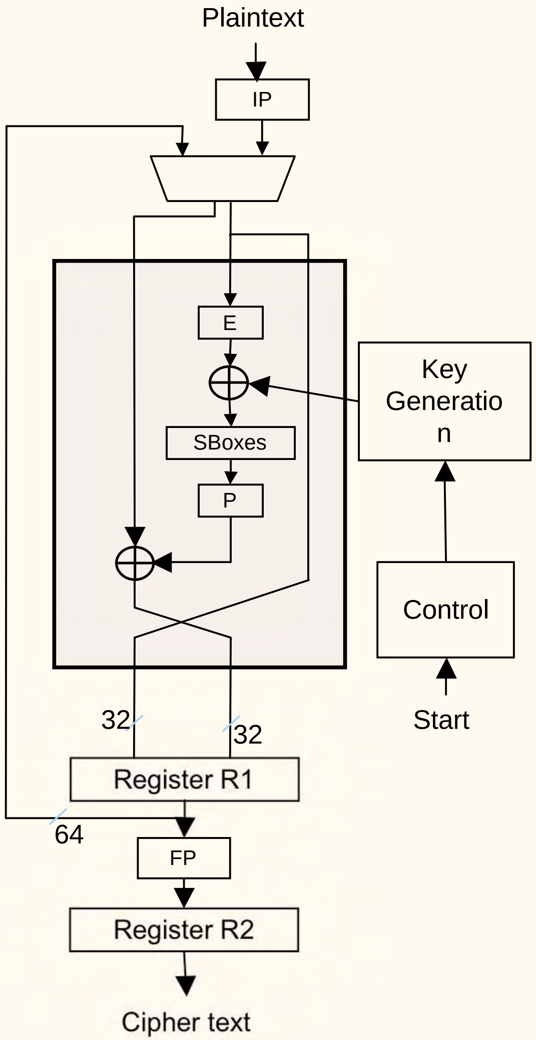
Characteristics

- Diffusion and confusion
  - ✓ *Confusion* refers to making the relationship between the key and the ciphertext as complex and involved as possible.
  - ✓ *Diffusion* refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. For diffusion to occur a change in a single bit of the plaintext should result in changing the value of many ciphertext bits.
- Iterative algorithms (rounds)
- Each round is a "bijective" operation

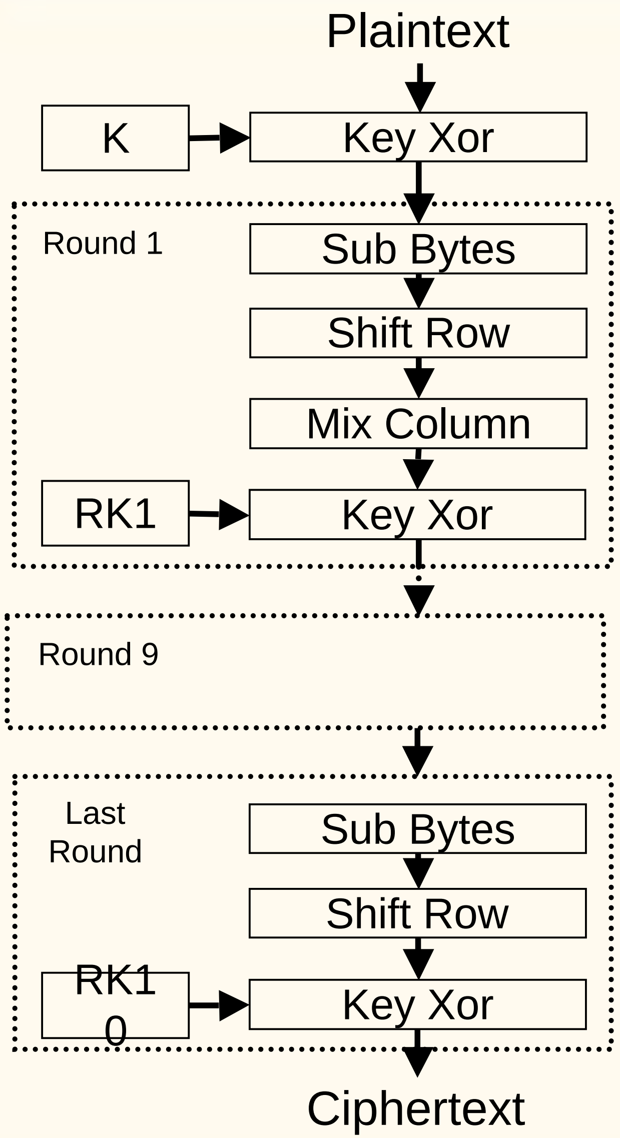
DES algorithm & architecture

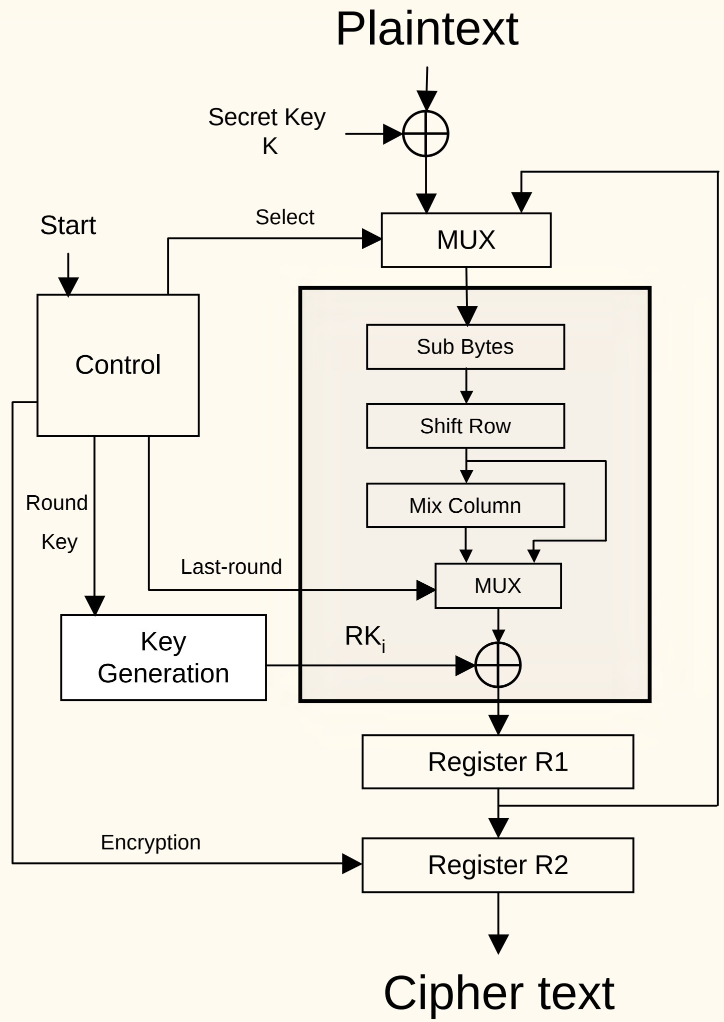






AES Algorithm & architecture



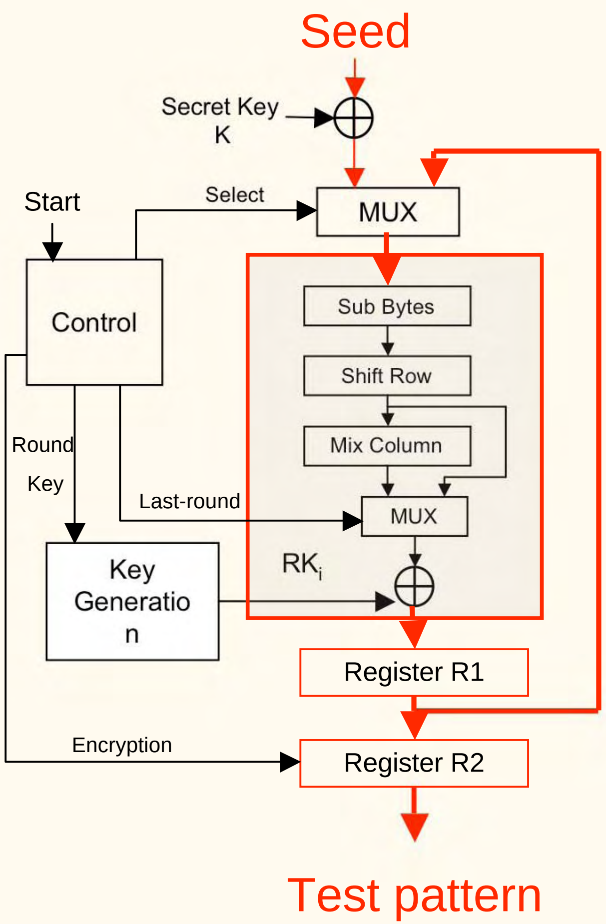


Cyphering & testability

- Diffusion
  - every input bit of a round influences many output bits, i.e. every input line of a round is in the logic cone of many output bits.
  - an error caused by a fault in the body of the round is very likely to propagate to the output.
  - observability
- Bijective
  - controllability
- Highly testable hardware implementations
  - => random testing

AES/DES as test pattern generator

One test pattern = Intermediate round result of encryption



AES/DES as TPG: randomness analysis

NIST Special Publication 800-22

[NIST 800-22]

Statistical package of 15 tests has been developed to test binary sequences randomness

1 : Monobit Test
2 : Block Frequency Test
3 : Cumulative Sums Forward (Reverse)
4 : Runs Test
5 : Long Runs of Ones Test
6 : Rank Test
7 : Discrete Fourier Transform (Spectral) Test
8 : Universal Statistical Test
9 : Approximate Entropy Test
10 : Serial Test
11 : Linear Complexity Test
12 : Aperiodic Templates
13 : Periodic Template Test
14 : Random Excursion Test 15 : Random Excursion Variant Test

1-round AES/DES : randomness

1.5 Mbit bitstream (leftmost bit)

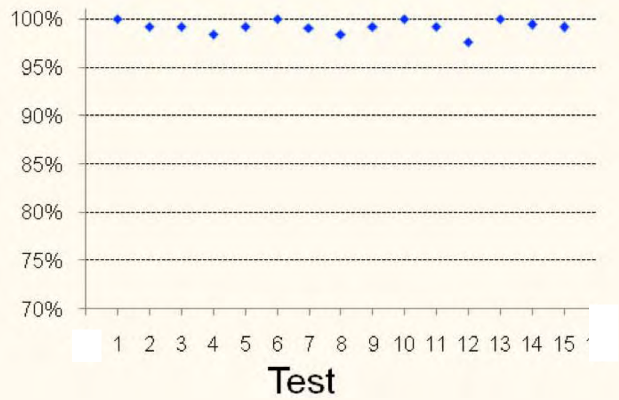
Test passes if x > 0.1

	1_round AES	1_round DES	LFSR
Frequency	0.71209	0.45847	0.00256
Blk-freq	0.47556	0.87065	0.44150
Runs	0.64156	0.18337	0.14362
Long Runs	0.28546	0.15829	0.96593
Rank	0.35722	0.24411	0.52660
DFT	0.03397	0.61040	0.81051
Aperiodic	0.50704	0.50541	0.49963
Periodic	0.08345	0.90055	0.39384
Univ.Maurer	0.44635	0.86625	0.24403
Lincomp	0.86761	0.88996	0
Serial	0.62350	0.42735	0.71383
Apen	0.44173	0.41358	0.63747
Cusum	0.73566	0.55751	0.00326
Random	0.41284	0.36790	0
Variant-R	0.49847	0.24177	0

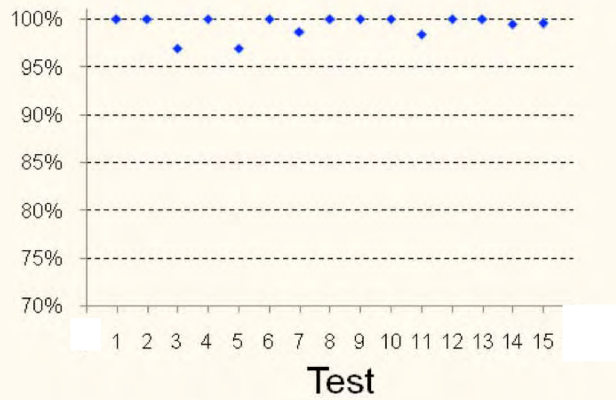
1-round AES/DES : randomness

Proportion of bitstreams passing each NIST test

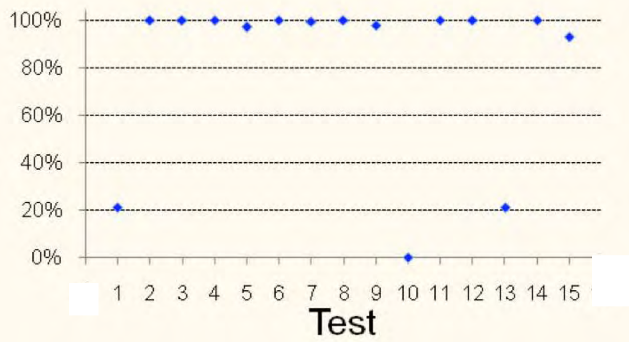
1-round AES



1-round DES



LFSR

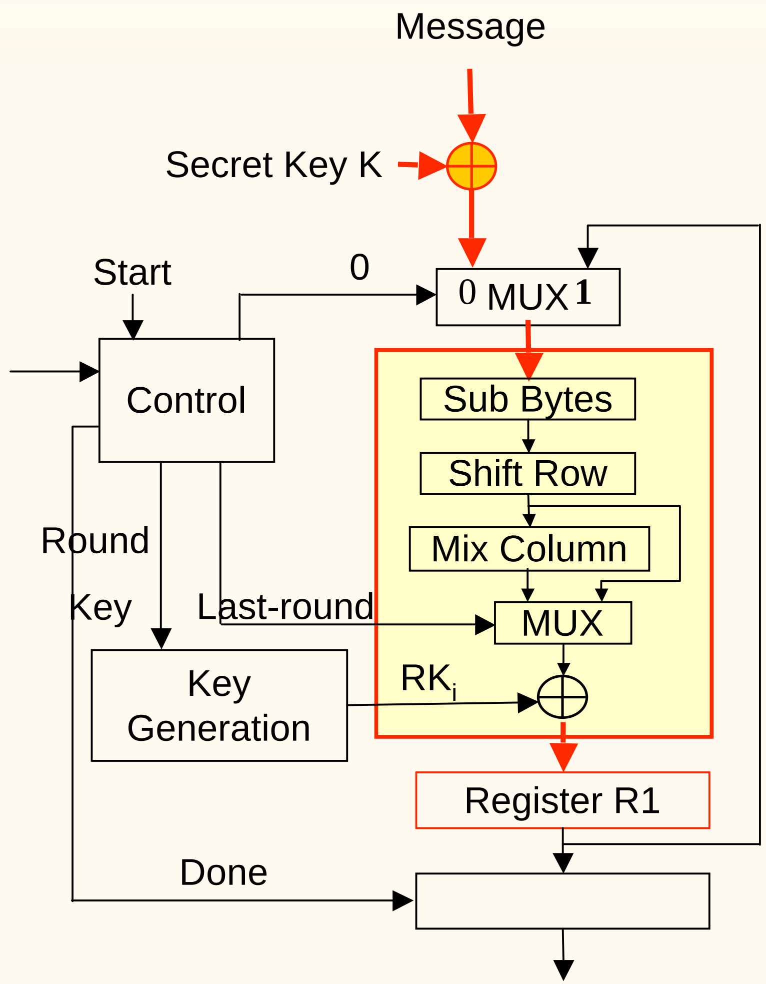


randomness:

“1-round AES” ≈ “1-round DES” ≈ LFSR

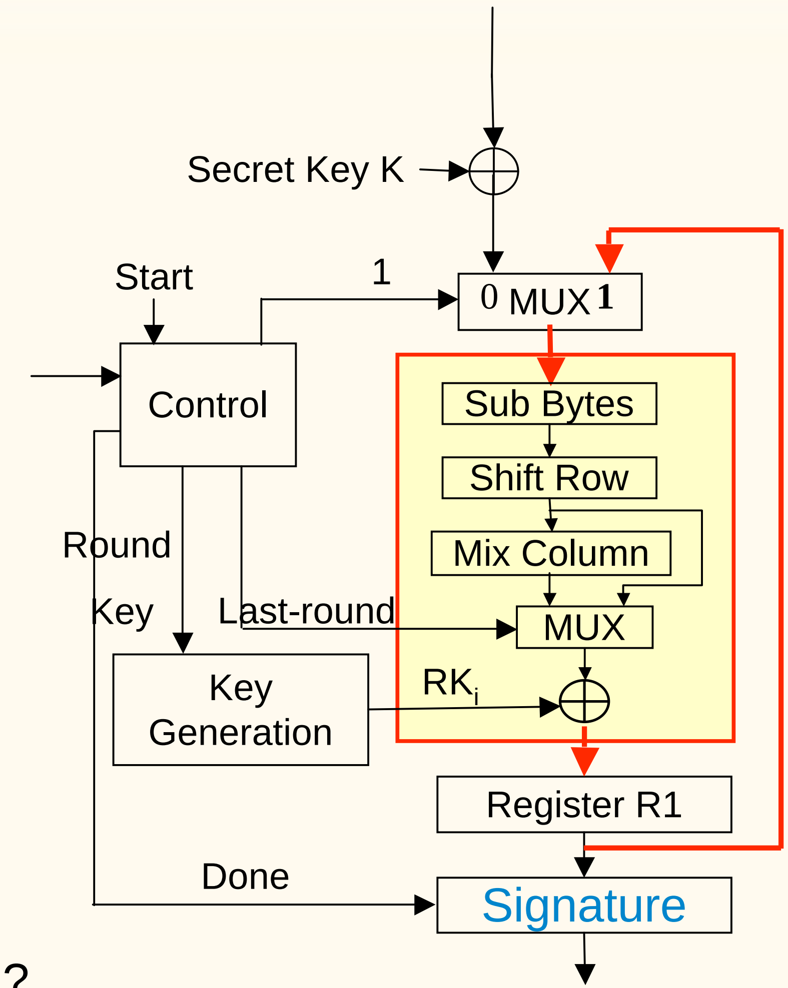
AES Self-test

Cycle 1



AES Self-test

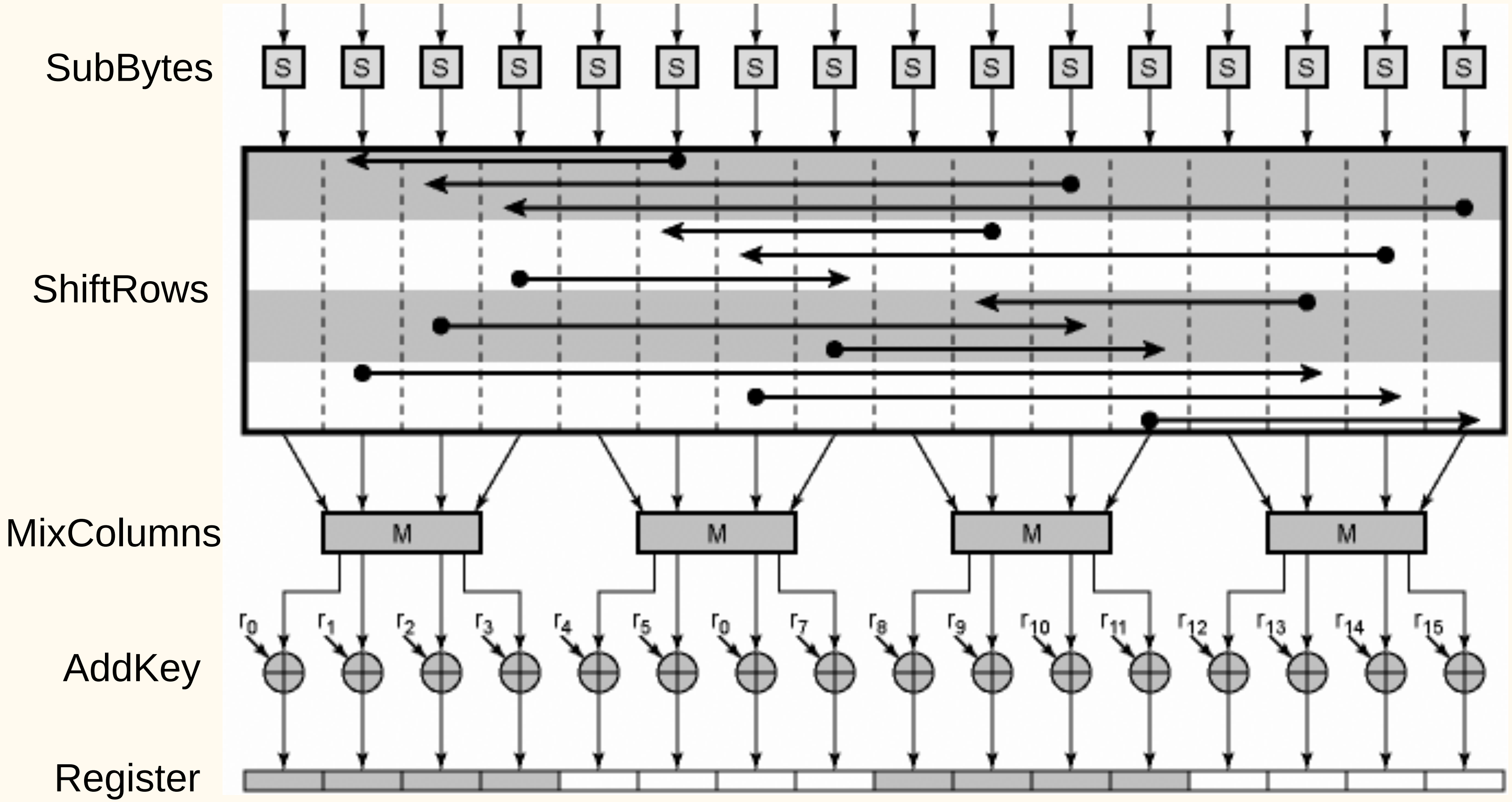
Cycle 2, 3, ……, T



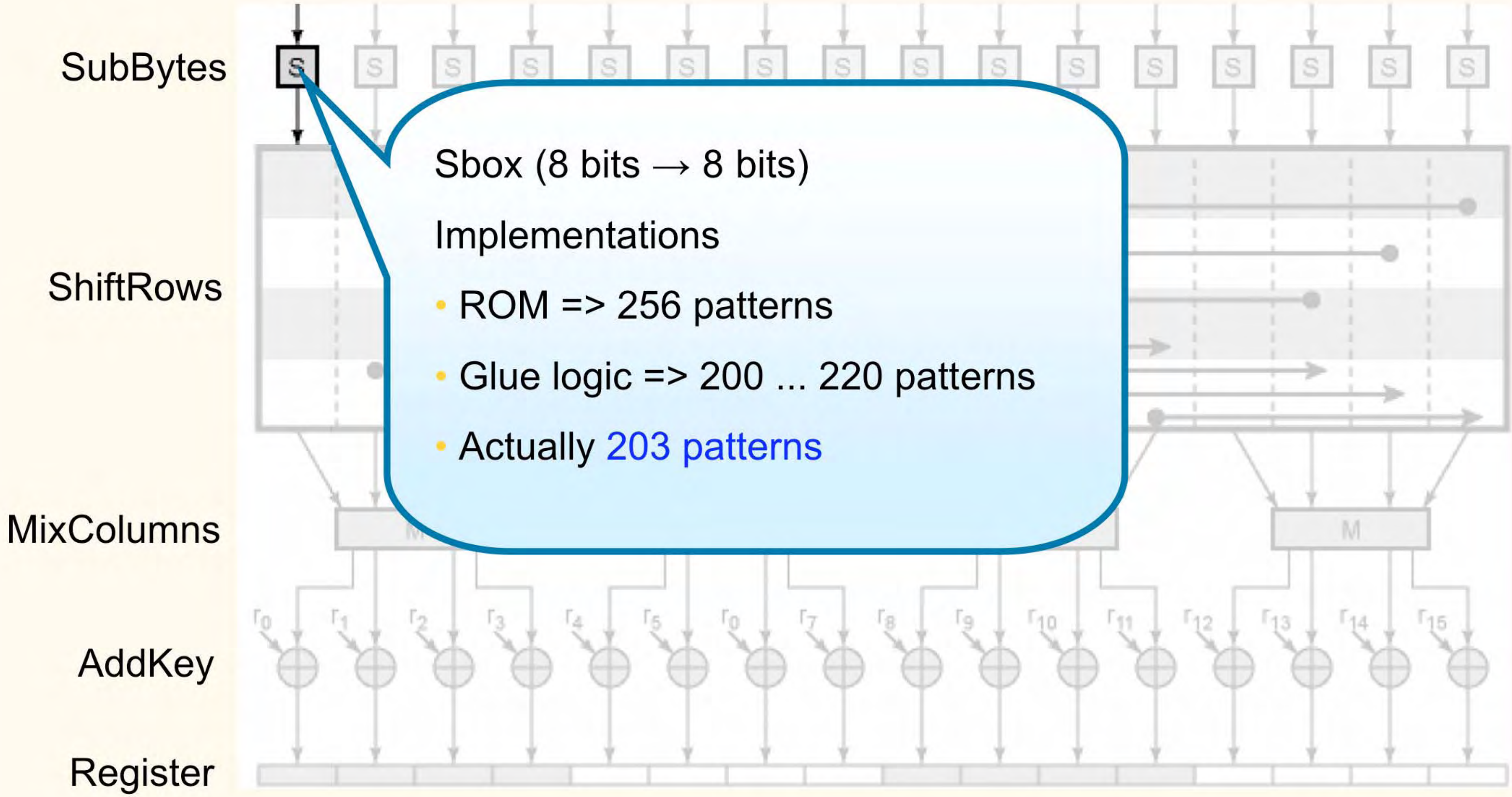
Is FC = 100% achievable ?

When ?

AES Self-test



AES Self-test

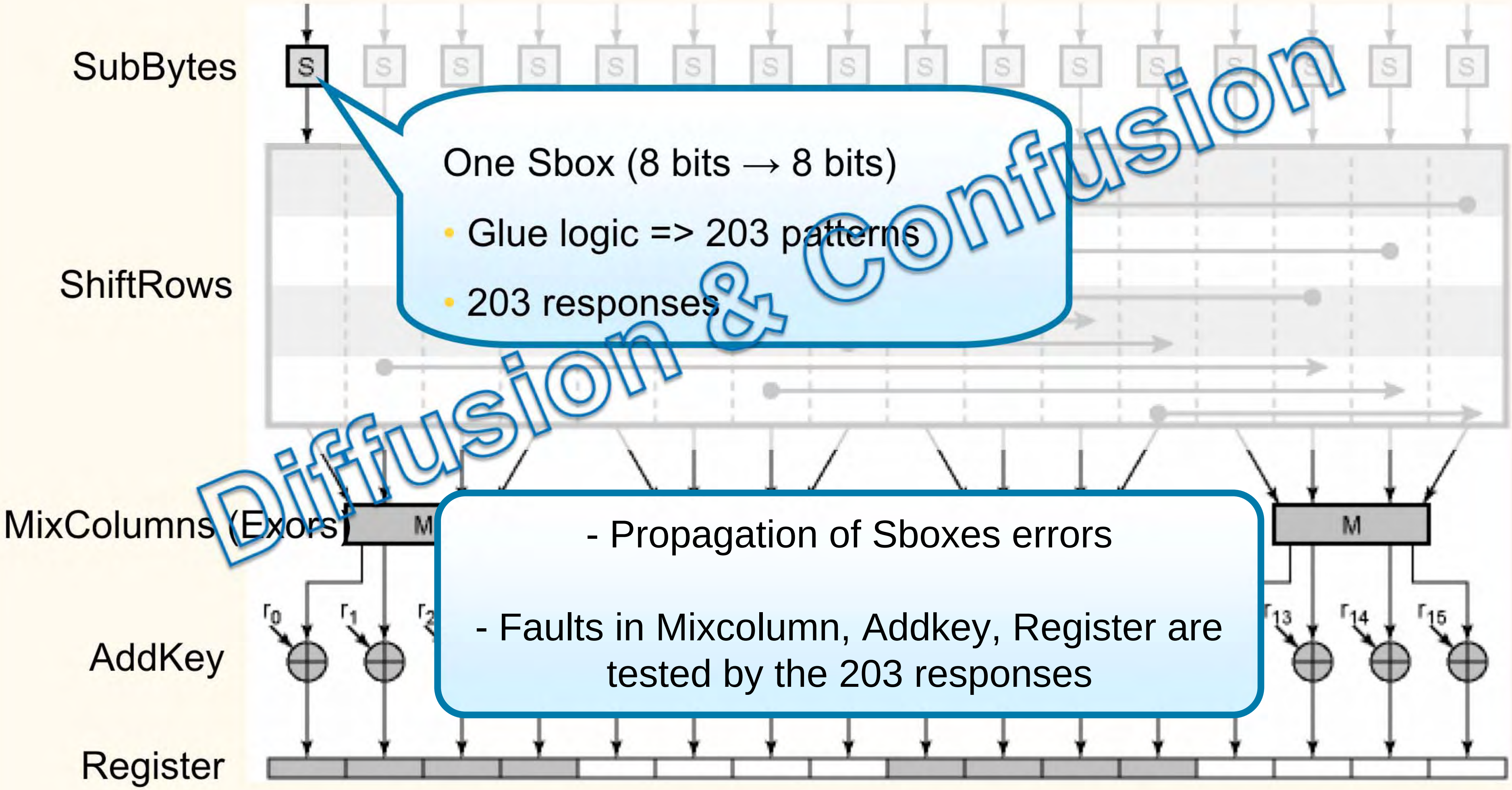


Sbox (8 bits → 8 bits)

Implementations

- ROM => 256 patterns
- Glue logic => 200 ... 220 patterns
- Actually 203 patterns

AES Self-test



AES Self-test

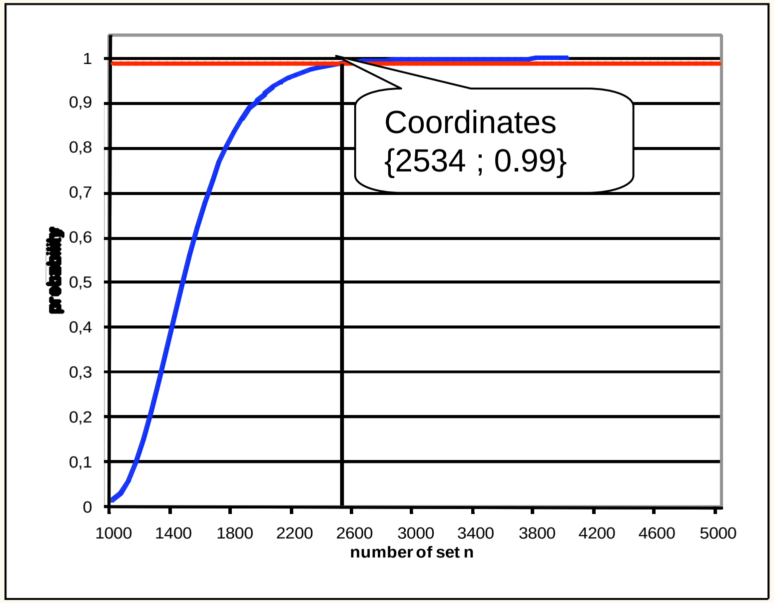
- How many random patterns are needed to get those 203 deterministic patterns? “The Coupon Collector Problem”

$$P\left(X_1 \cap X_2 \cap \ldots \cap X_k\right) = 1 - \sum_{j=1}^{k} (-1)^{j-1} C_k^j \left(\frac{m-j}{m}\right)^T$$

$$\underbrace{m = 2^{128} \qquad k = \#\text{vectors} = 203}$$

$P\left(X_1 \cap X_2 \cap \ldots \cap X_k\right) = 99\%$ ⇒ ***T = 2534* random patterns**

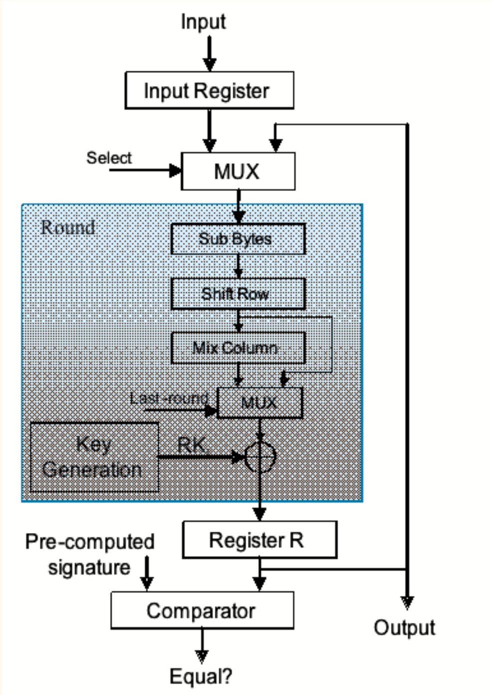
⇒ ***2534 AES rounds***



- Sbox implementation:
  - ✓ #test vectors ∈ {200,...,256} => T∈ {2520,...,2590}

AES Self-test

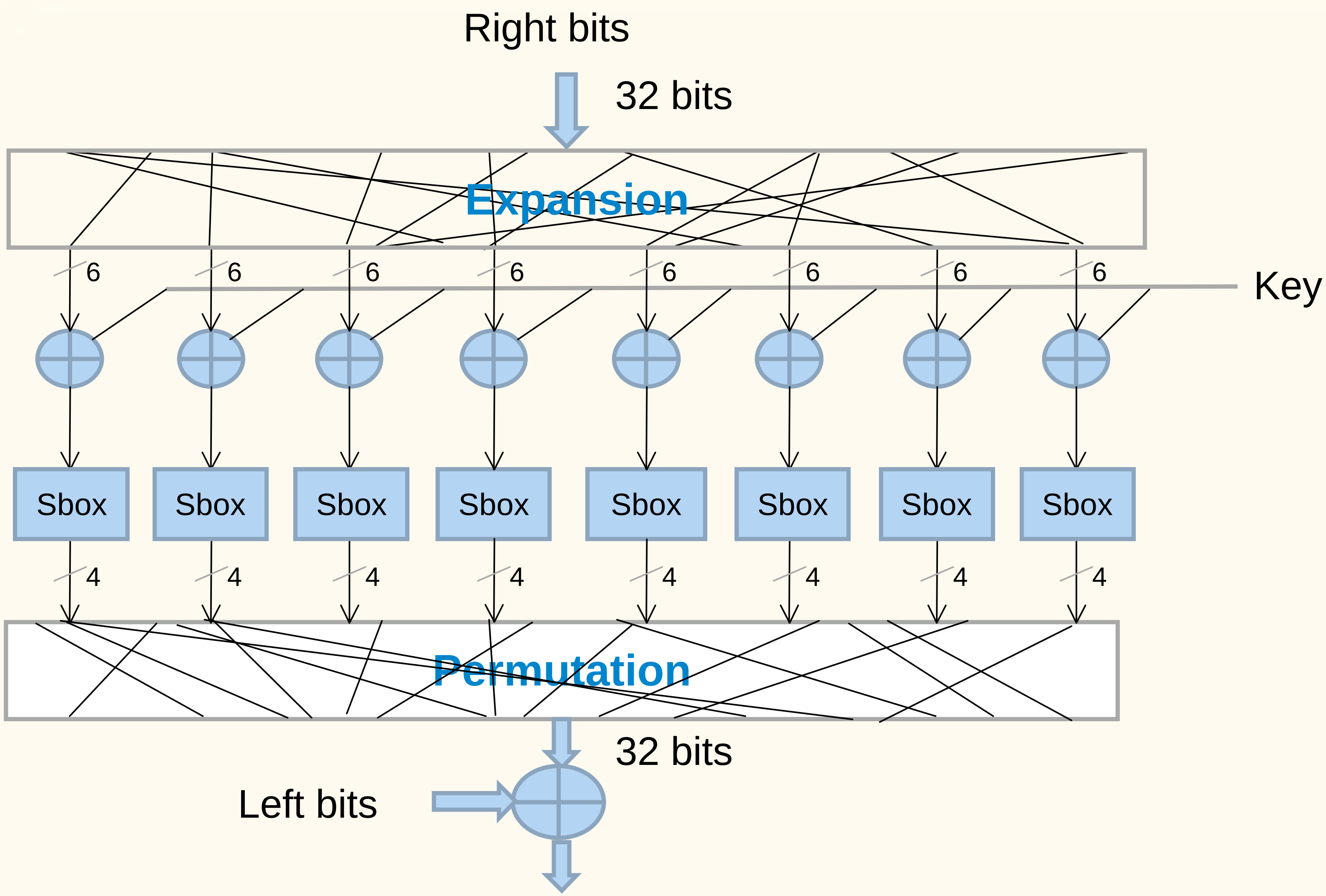
- "Pseudo" Fault Simulation



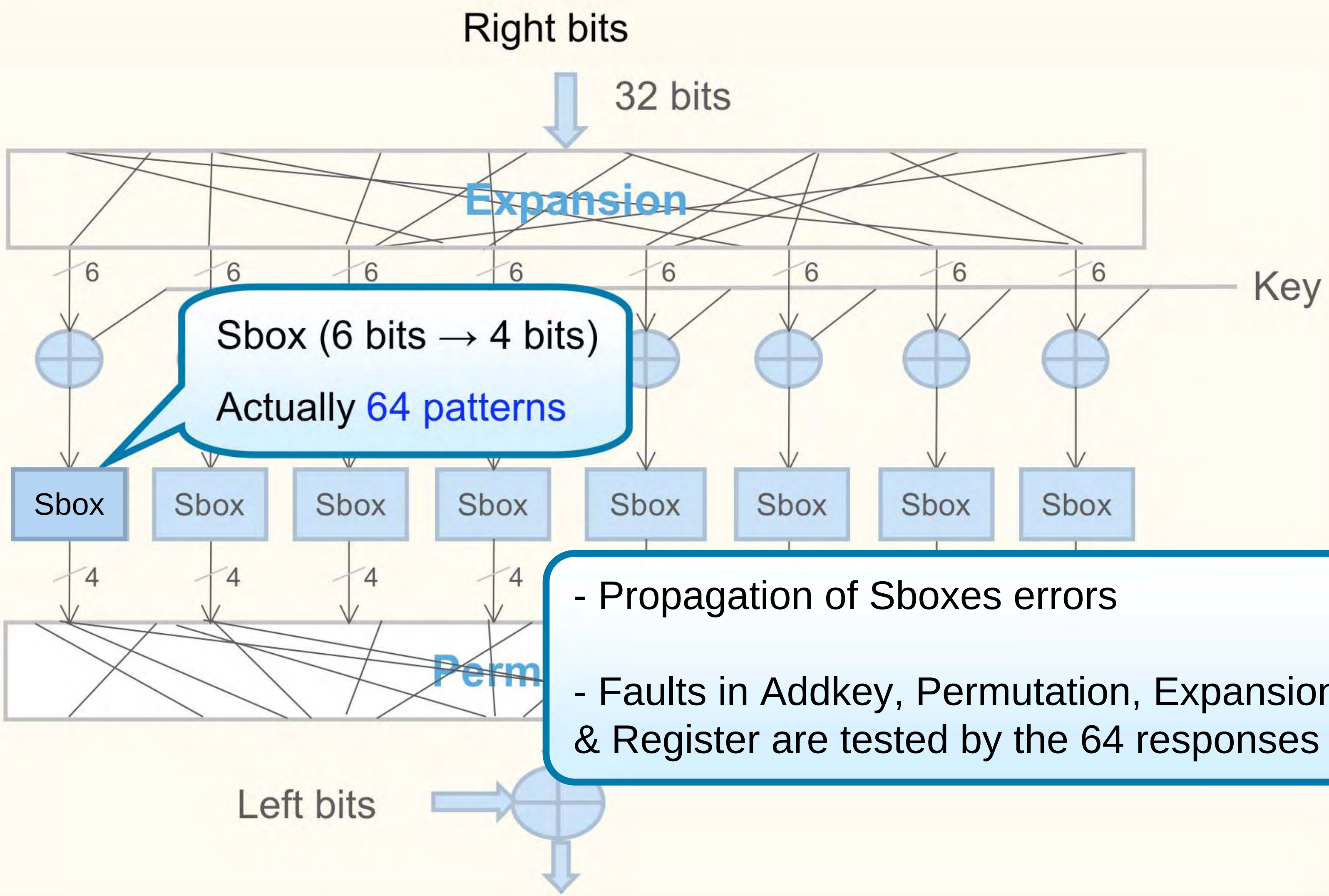
- Result :
  - ✓ Fault coverage: 100% after 2534 cycles
  - ✓ Test time reduction: 2400 cycles (with several keys, several plaintexts)

- Specific plaintext, specific key for minimal test time ?

DES Self-test



DES Self-test



Sbox (6 bits → 4 bits)

Actually 64 patterns

- Propagation of Sboxes errors
- Faults in Addkey, Permutation, Expansion & Register are tested by the 64 responses

DES random sequence length

$$P\left(X_1 \cap X_2 \cap \ldots \cap X_k\right) = 1 - \sum_{j=1}^{k} (-1)^{j-1} C_k^j \left(\frac{m-j}{m}\right)^T$$

$$\underbrace{m = 2^{64} \qquad k = \#vectors = 64}$$

$P\left(X_1 \cap X_2 \cap \ldots \cap X_k\right) = 99\%$ ⇨ ***T = 540***

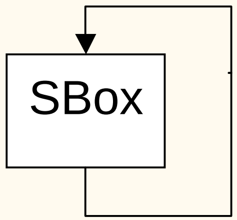
random patterns (540 rounds)

⇨ 34 encryptions

Results : 100% FC after 24 encryptions (Data path and control)

Optimisation

- Speeding up self-test of AES
  - ✓ 2500 cycles for 256 test patterns
  - ✓ Feed-back on Sbox

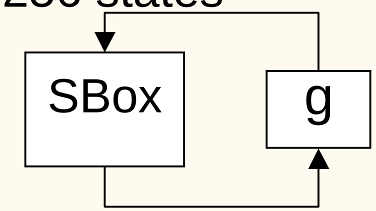


- 5 cycles in state graph =>

Length	States
59	63,FB,F,76,38,7,C5,A6,24,36,5,6B,7F,D2,B5,D5, 3,7B,21,FD,54,20,B7,A9,D3,66,33,C3,2E,31,C7, C6,B4,8D,5D,4C,29,A5,6,6F,A8,C2,25,3F,75,9D, 5E,58,6A,2,77,F5,E6,8E,19,D4,48,52,0
81	7C,10,CA,74,92,4F,84,5F,CF,8A,7E,F3,D,D7,E,AB, 62,AA,AC,91,81,C,FE,BB,EA,87,17,F0,8C,64,43,1A, A2,3A,80,CD,BD,7A,DA,57,5B,39,12,C9,DD,C1,78, BC,65,4D,E3,11,82,13,7D,FF,16,47,A0,E0,E1,F8,41, 83,EC,CE,8B,3D,27,CC,4B,B3,6D,3C,EB,E9,1E,72,40,9,1
87	F2,89,A7,5C,4A,D6,F6,42,2C,71,A3,A,67,85,97,88, C4,1C,9C,DE,1D,A4,49,3B,E2,98,46,5A,BE,AE,E4, 69,F9,99,EE,28,34,18,AD,95,2A,E5,D9,35,96,90,60,D0, 70,51,D1,3E,B2,37,9A,B8,6C,50,53,ED,55,FC,B0,E7, 94,22,93,DC,86,44,1B,AF,79,B6,4E,2F,15,59,CB,1F, C0,BA,F4,BF, 8,30,4
27	2B,F1,A1,32,23,26,F7,68,45,6E,9F,DB,B9,56,B1,C8, E8,9B,14,FA,2D,D8,61,EF,DF,9E, B
2	8F,73

- Add a (simple) feed-back function for traversing all 256 states

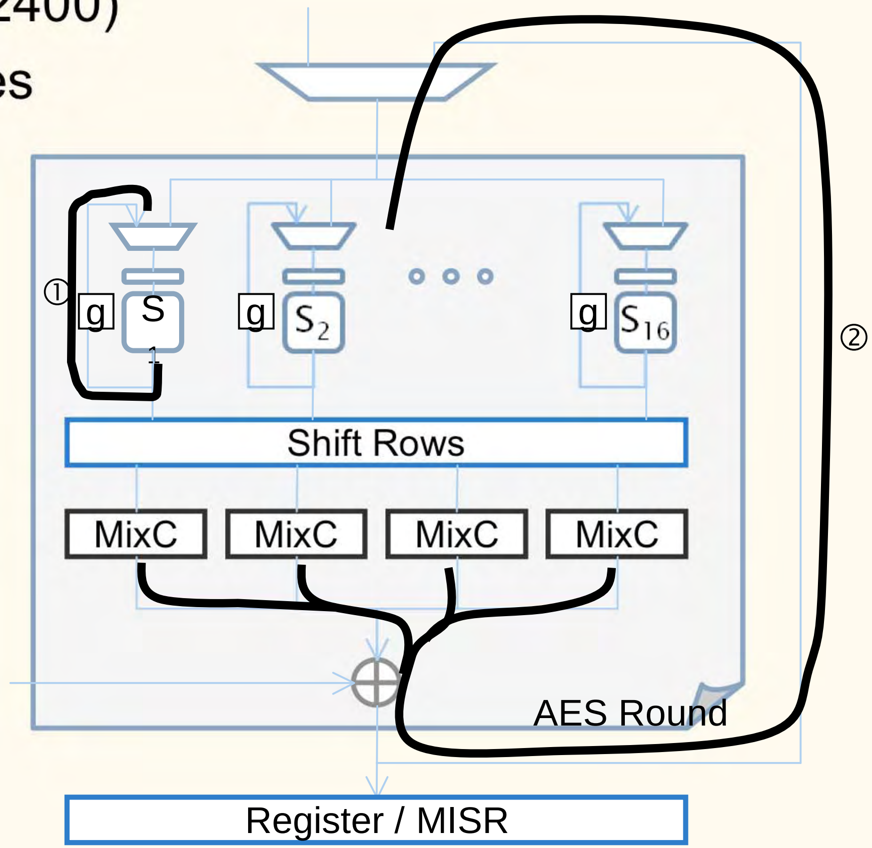
  g = exor (01110110) → 5 inverters



Optimisation

- 2 steps procedure
  - ✓ test of Sboxes: 256 cycles (vs 2400)
  - ✓ test of remaining logic: 16 cycles

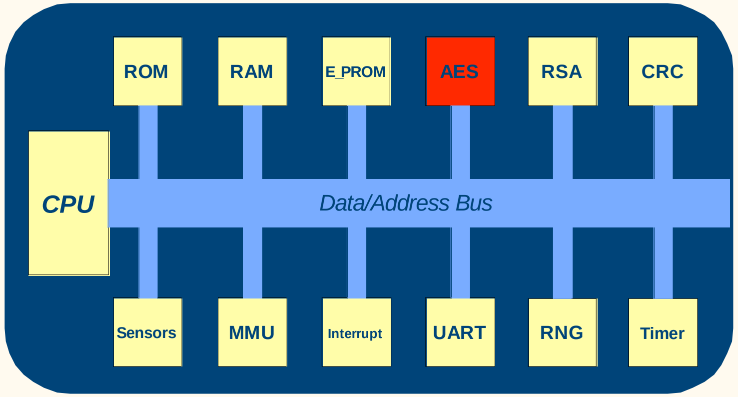
- Area overhead : 1%



Conclusion

- AES/DES as TPG
  - ✓ Randomness: better than LFSRs
- Self Testability
  - ✓ AES: 2400 encryption rounds (of a single message)
  - ✓ DES: 540 encryption rounds (of a single message)
  - ✓ Suitable technique for other ciphering circuits (IDEA, Fox, Blowfish, ...)

✓ No area overhead
✓ No impact on performance
✓ No impact on security



References

- **[FIPS PUB 46-3]:** DATA ENCRYPTION STANDARD (DES), 1999 October 25
- **[http://www.commentcamarche.net/crypto/des.php3]**
- **[FIPS PUB 197]:** Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001November 26
- **[Sch97]:** B. Schneier, Cryptographie appliquée : protocoles, algorithmes et codes sources en C, J. Wiley, 1997 (p491-499)
- **[http://www.securiteinfo.com/crypto/cracked.shtml]**
- **[Yan04]:** B. Yang, K. Wu, R. Karri, Polytechnic University, "Scan-based Side-Channel Attack on Dedicated Hardware Implementations on Data Encryption Standard", International Test Conference (ITC 2004), Charlottes, USA, October 26-28, pp 339-344
- [**Yan05]:** B. Yang, K. Wu and R. Karri, Secure Scan: A Design-for-Test Architecture for Crypto Chips, Design Automation Conference (DAC 2005), Anaheim, July 12-14 pp 135-140, 2005
- **[[Yan, FDTC 05]:** B. Yang & R. Karri, "Crypto BIST: A Built-In Self Test Architecture for Crypto Chips", 2nd Workshop on fault diagnosis and tolerance in cryptography (FDTC 2005), pp 95-108
- **[NIST 800-22]:** A statistical test suite for random and pseudorandom number generators for cryptographic applications NIST Special Publication 800-22 (with revisions dated May 15, 2001)

Statistical tests NIST

- -**Monobit Test**: determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.
- - **Block Frequency Test**: determine whether the number of ones and zeros in each of M non-overlapping blocks created from a sequence appear to have a random distribution.
- - **Cumulative Sums Forward (Reverse) Test**: determine whether the sum of the partial sequences occurring in the tested sequence is too large or too small.
- - **Runs Test**: determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.
- - **Long Runs of Ones Test**: determine whether the longest run of ones within the tested sequence is consistent with the longest run of ones that would be expected in a random sequence.
- - **Rank Test**: check for linear dependence among fixed length substrings of the original sequence.
- - **Discrete Fourier Transform (Spectral) Test**: detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.
- - **Aperiodic Templates Test**: reject sequences that exhibit too many occurrences of a given non-periodic (aperiodic) pattern.
- - **Periodic Template Test**: reject sequences that show deviations from the expected number of runs of ones of a given length.
- - **Universal Statistical Test**: detect whether or not the sequence can be significantly compressed without loss of information. A compressible sequence is considered to be nonrandom.
- - **Approximate Entropy Test**: compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a normally distributed sequence.
- - **Random Excursion Test**: determine if the number of visits to a state within a random walk exceeds what one would expect for a random sequence.
- - **Random Excursion Variant Test**: detect deviations from the distribution of the number of visits of a random walk to a certain state.
- - **Serial Test**: determine whether the number of occurrences of m-bit overlapping patterns is approximately the same as would be expected for a random sequence.
- - **Linear Complexity Test**: determine whether or not the sequence is complex enough to be considered random.