Scan chain encryption, a countermeasure against scan attacks

Mathieu da Silva, Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

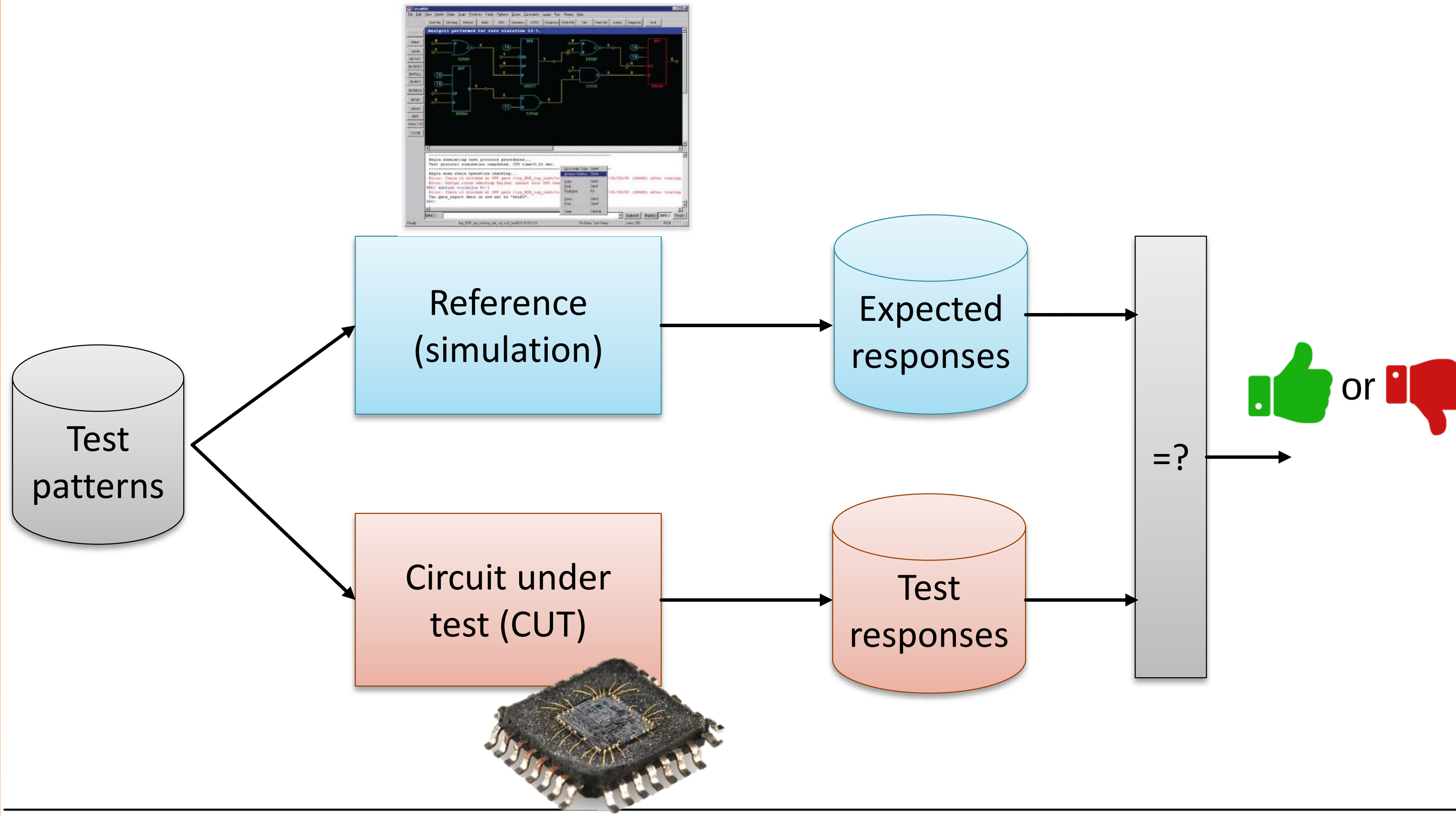
SCAN CHAIN ENCRYPTION, A COUNTERMEASURE AGAINST SCAN ATTACKS

Mathieu Da Silva, Marie-Lise Flottes, Giorgio Di Natale, Bruno Rouzeyre

PHISIC 2018

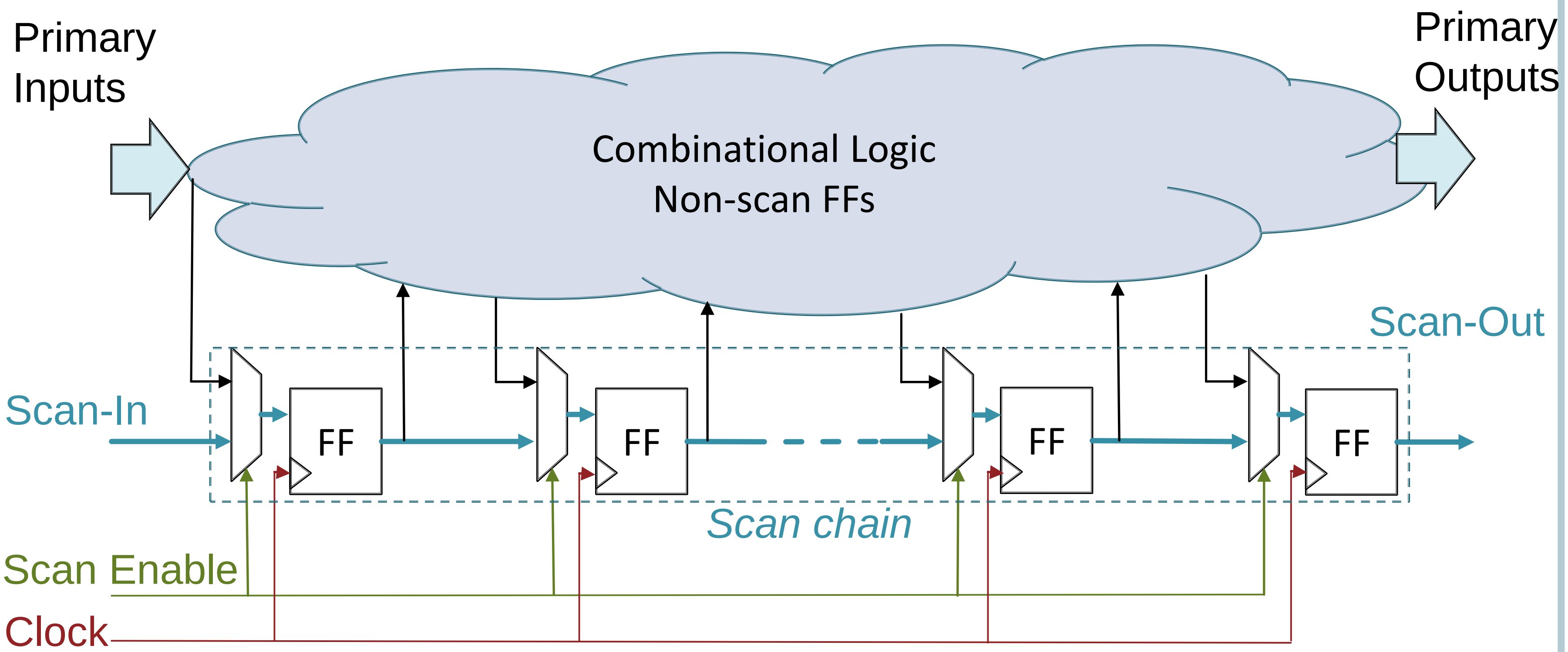
CONTEXT

- Test of circuit is a mandatory step in IC production



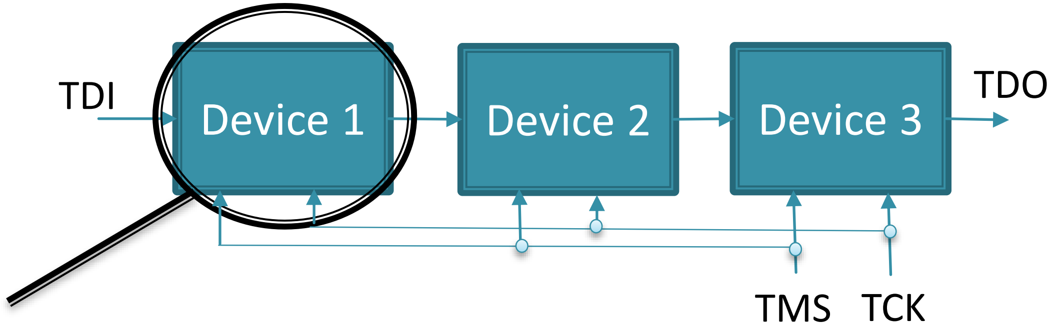
CONTEXT

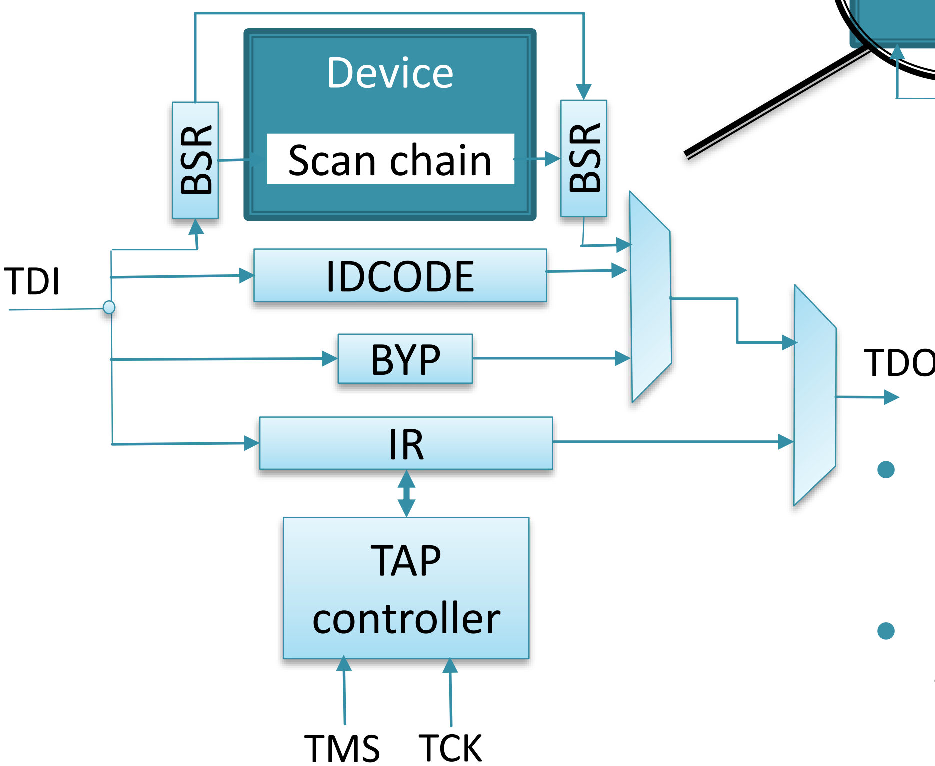
- Most popular method for Design-for-Test = Scan chains
  - Replace original FF by Scan FF connected serially together
  - Extra port « Scan-In » => controllability on internal states
  - Extra port « Scan-Out » => observability on internal states



CONTEXT

- Test standards
  - IEEE 1149 (JTAG) for board testing

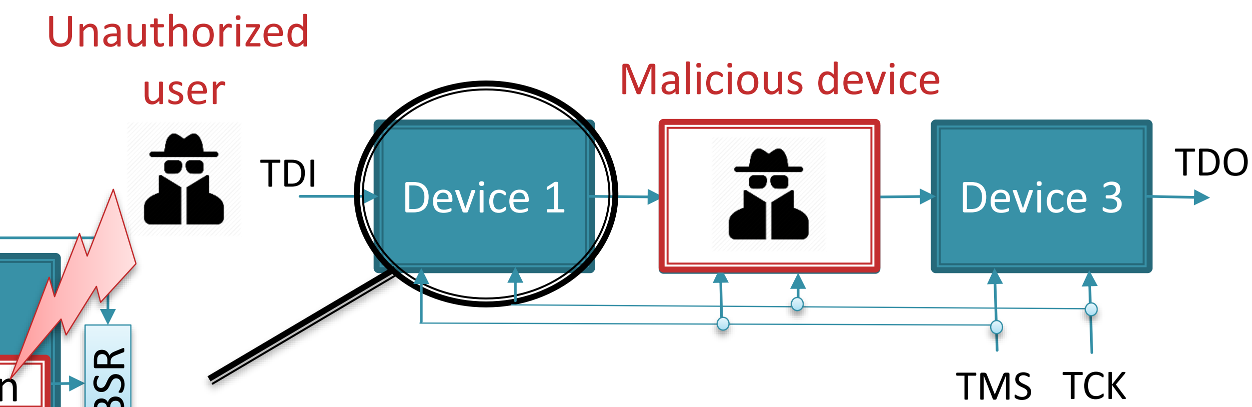


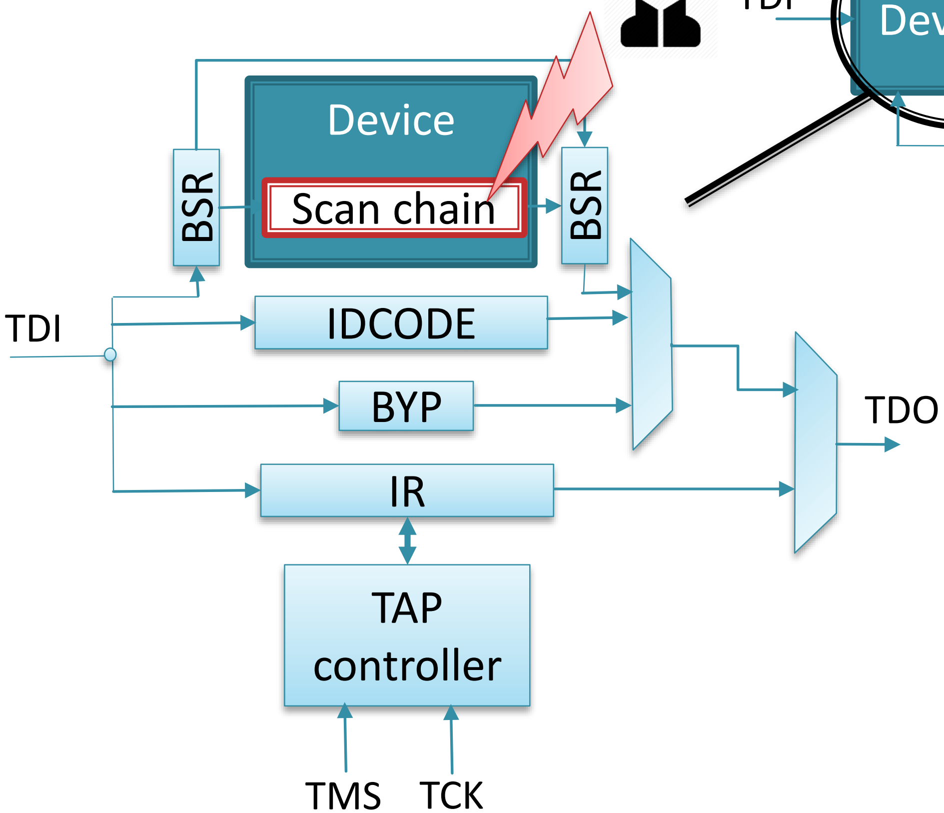


  - IEEE 1500 for cores testing in a SoC
  - IEEE 1687 (IJTAG) for embedded instruments

Context

- Threats





- Untrusted devices

  Rosenfeld et al., Attacks and Defenses for JTAG, IEEE Design & Test 2010

- Malicious users (example: scan attacks)

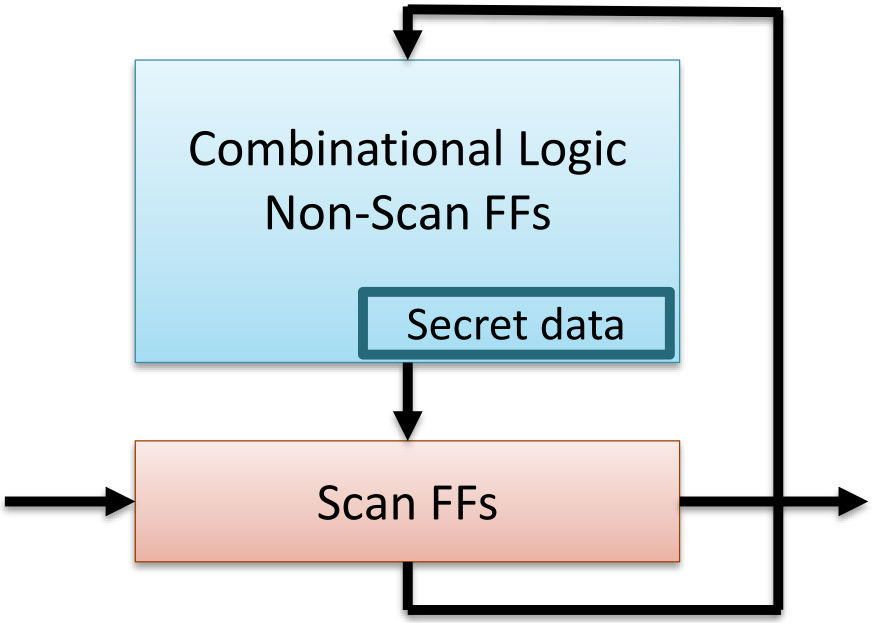
  Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD'06

SUMMARY

1) **Scan attacks**
2) A new countermeasure: Scan chain encryption
3) Implementation with block cipher
4) Implementation with stream cipher
5) Conclusion

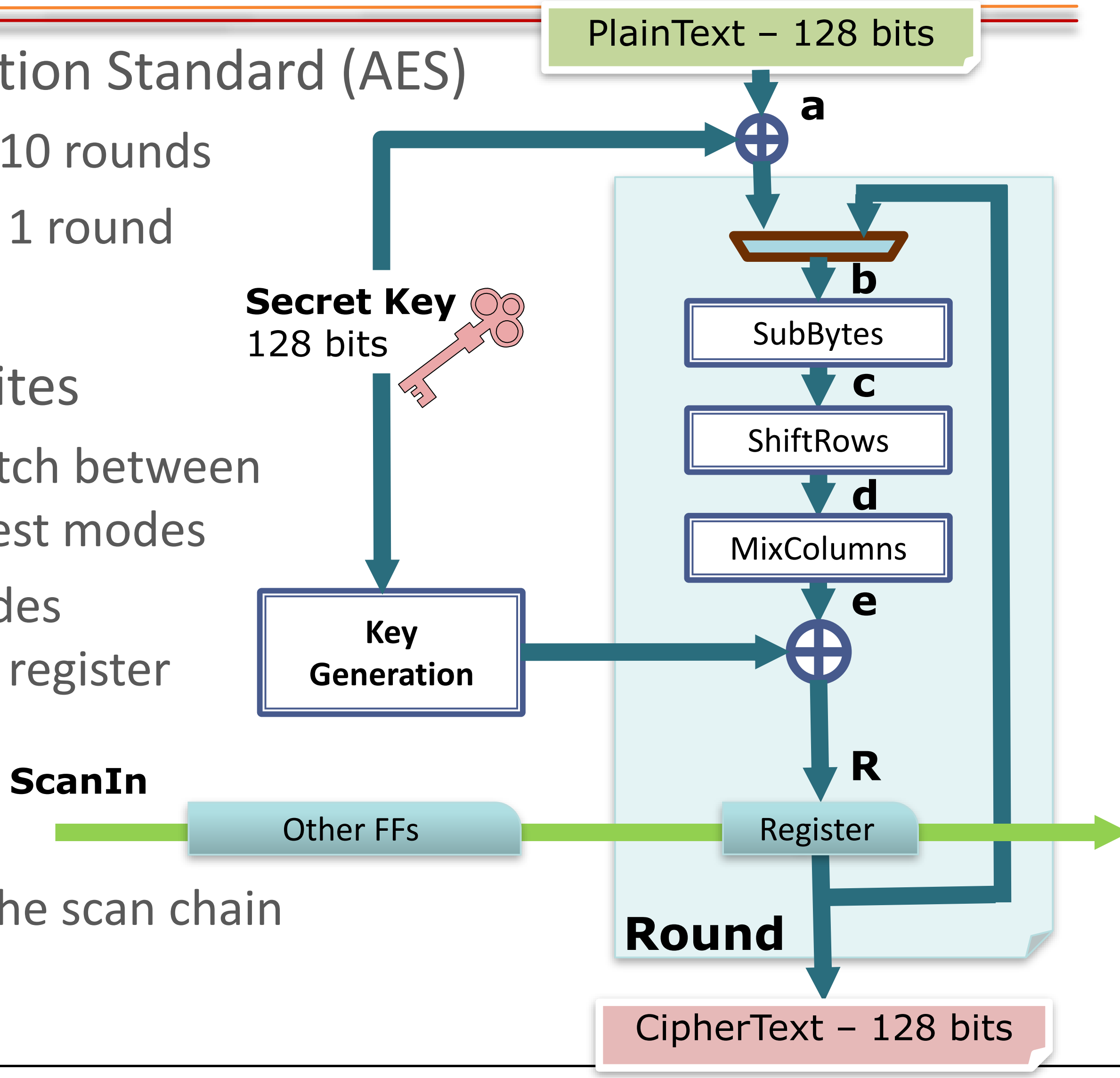
SCAN ATTACK PRINCIPLE

- Goal: Retrieve embedded secret data
- Exploit observability or controllability offered by scan chains
- Principle: switch between functional and scan modes
- Main target: secret key of crypto-processors (example: AES)



SCAN ATTACK ON AES

- Advanced Encryption Standard (AES)
  - Ciphertext after 10 rounds
  - Not secure after 1 round
- Attack pre-requisites
  - Attacker can switch between functional and test modes
  - Scan chain includes FFs of the round register
- Attack principle
  - Observation of the scan chain after 1 round

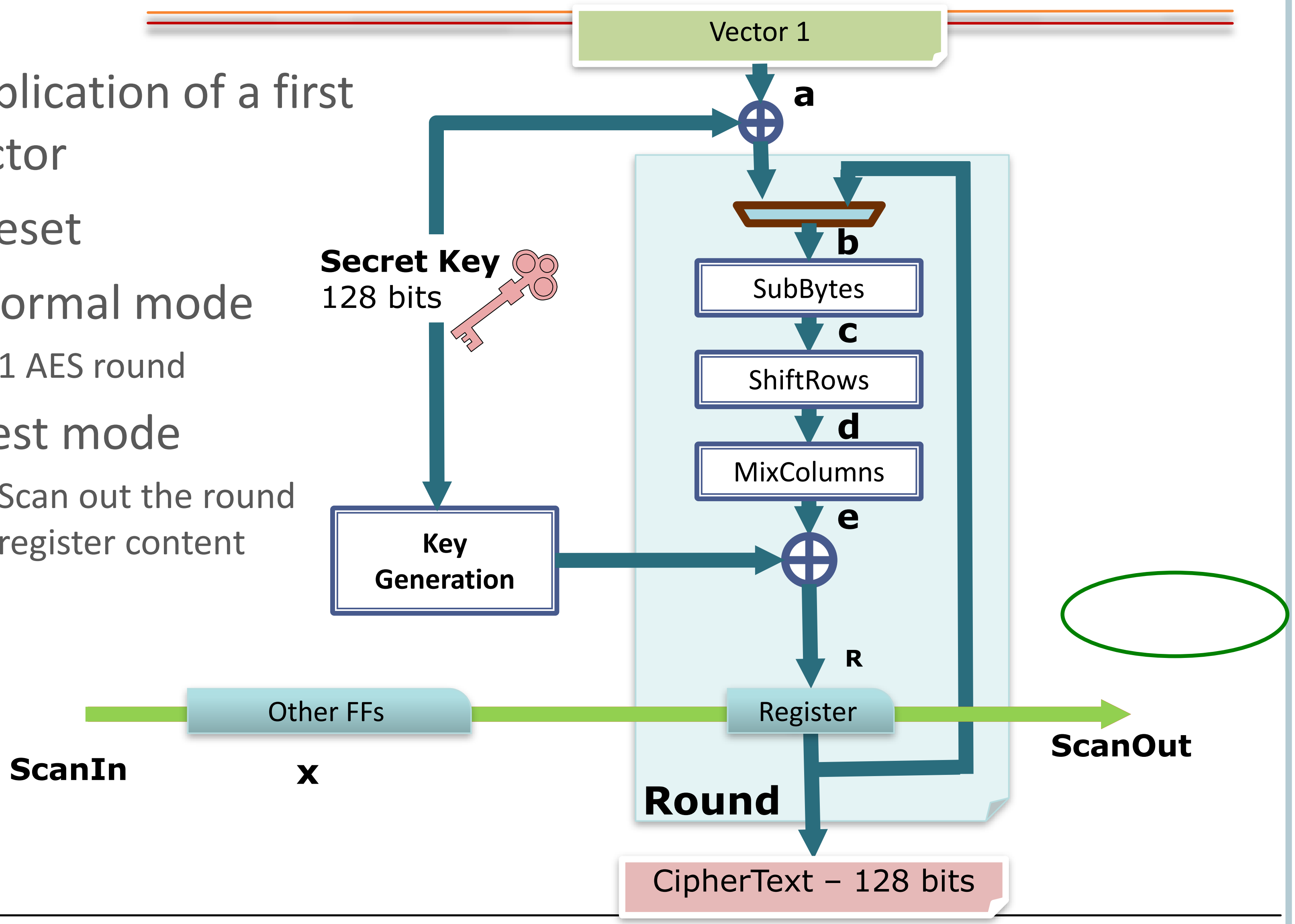


Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD'06

Differential attack

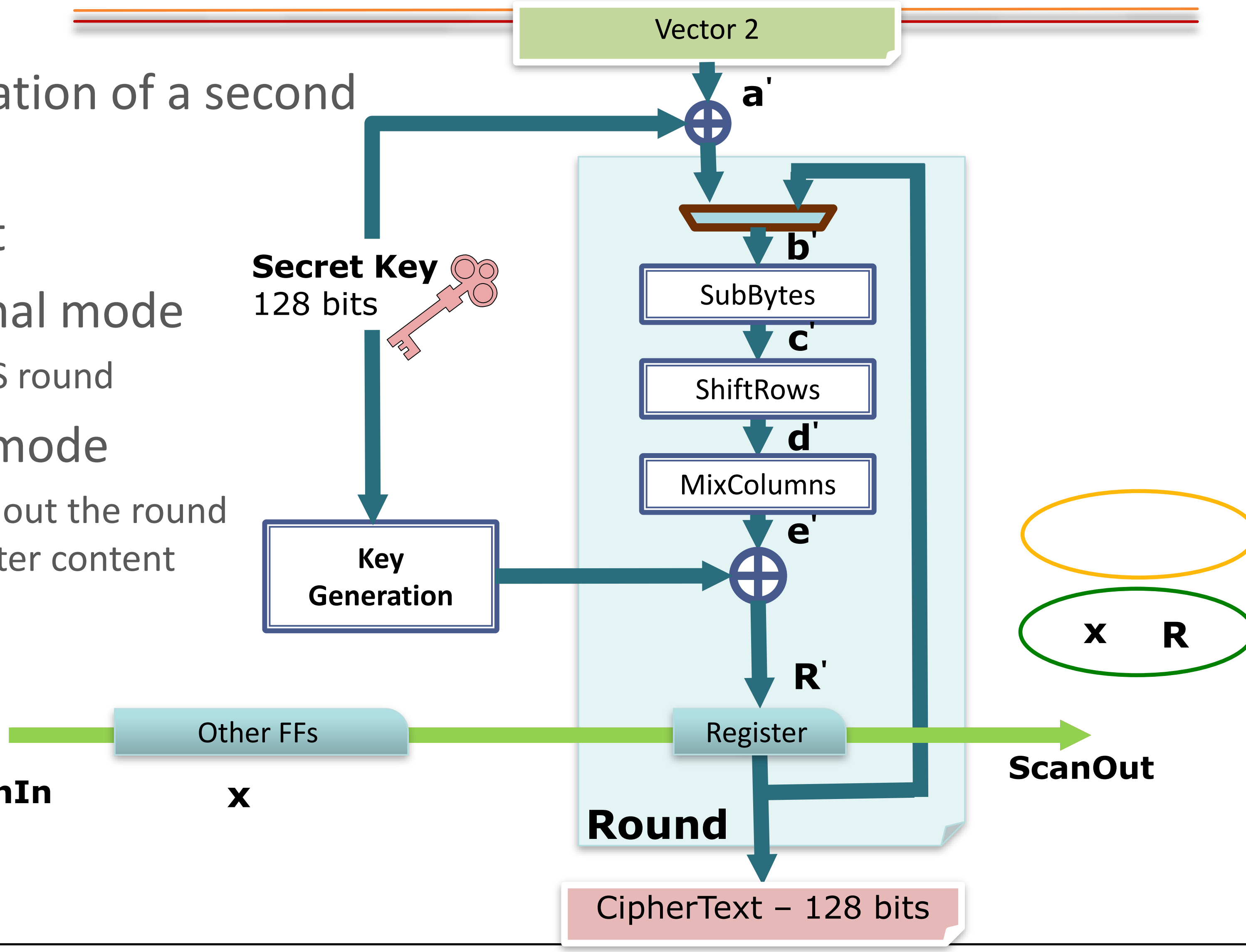
- Application of a first vector

1) Reset
2) Normal mode
   - 1 AES round
3) Test mode
   - Scan out the round register content



DIFFERENTIAL ATTACK

- Application of a second vector
1) Reset
2) Normal mode
   - 1 AES round
3) Test mode
   - Scan out the round register content



Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD'06

DIFFERENTIAL ATTACK

- Hamming distance

  - Attacker applies pairs of input values until hamming distance equal to specific values => key byte revealed

- On average, 32 trials

⇒ 512 trials to retrieve the whole 128-bit key

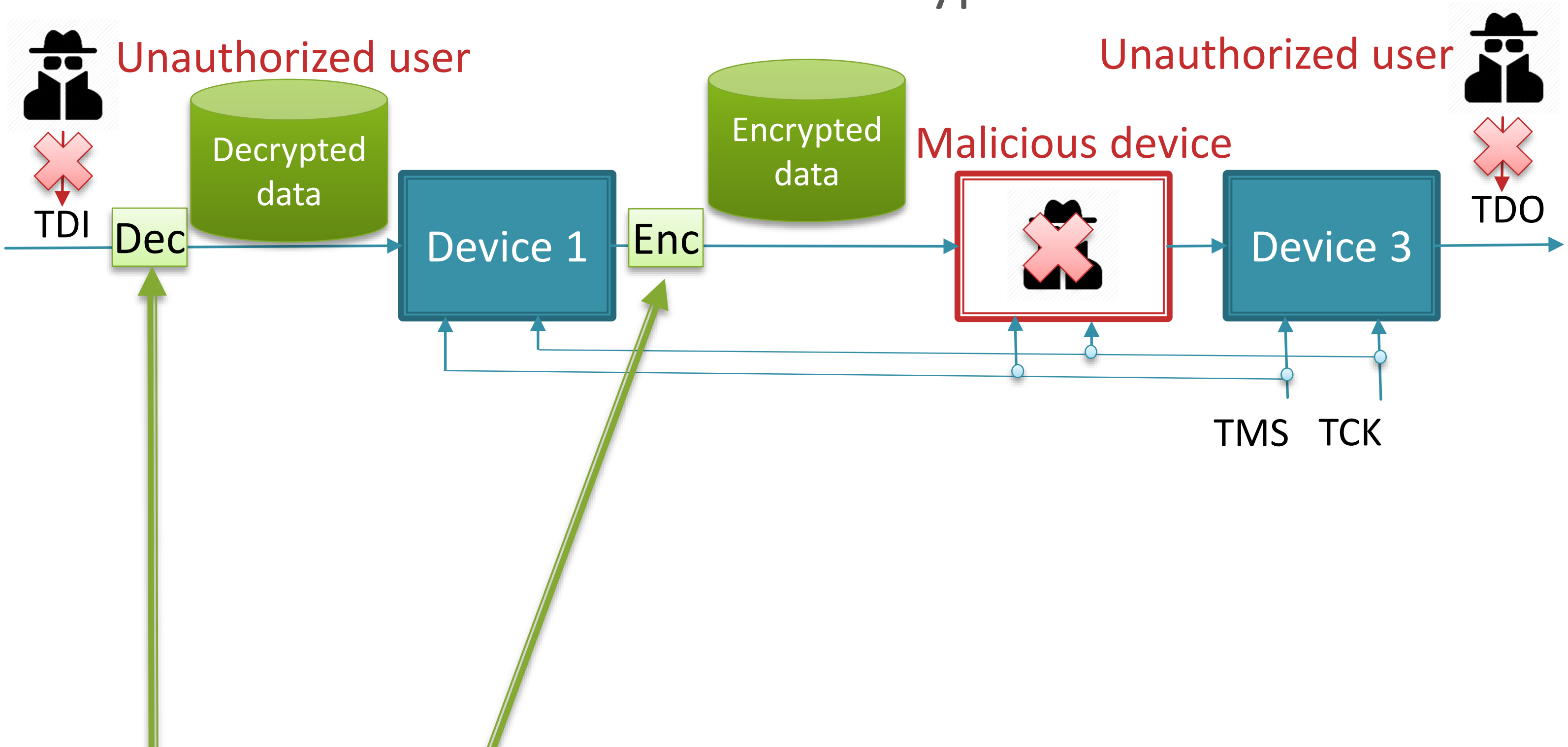
Yang et al., Secure Scan: A Design-for-Test Architecture for Crypto Chips, TCAD'06

SUMMARY

1) Scan attacks
2) **A new countermeasure: Scan chain encryption**
3) Implementation with block cipher
4) Implementation with stream cipher
5) Conclusion

SCAN CHAIN ENCRYPTION

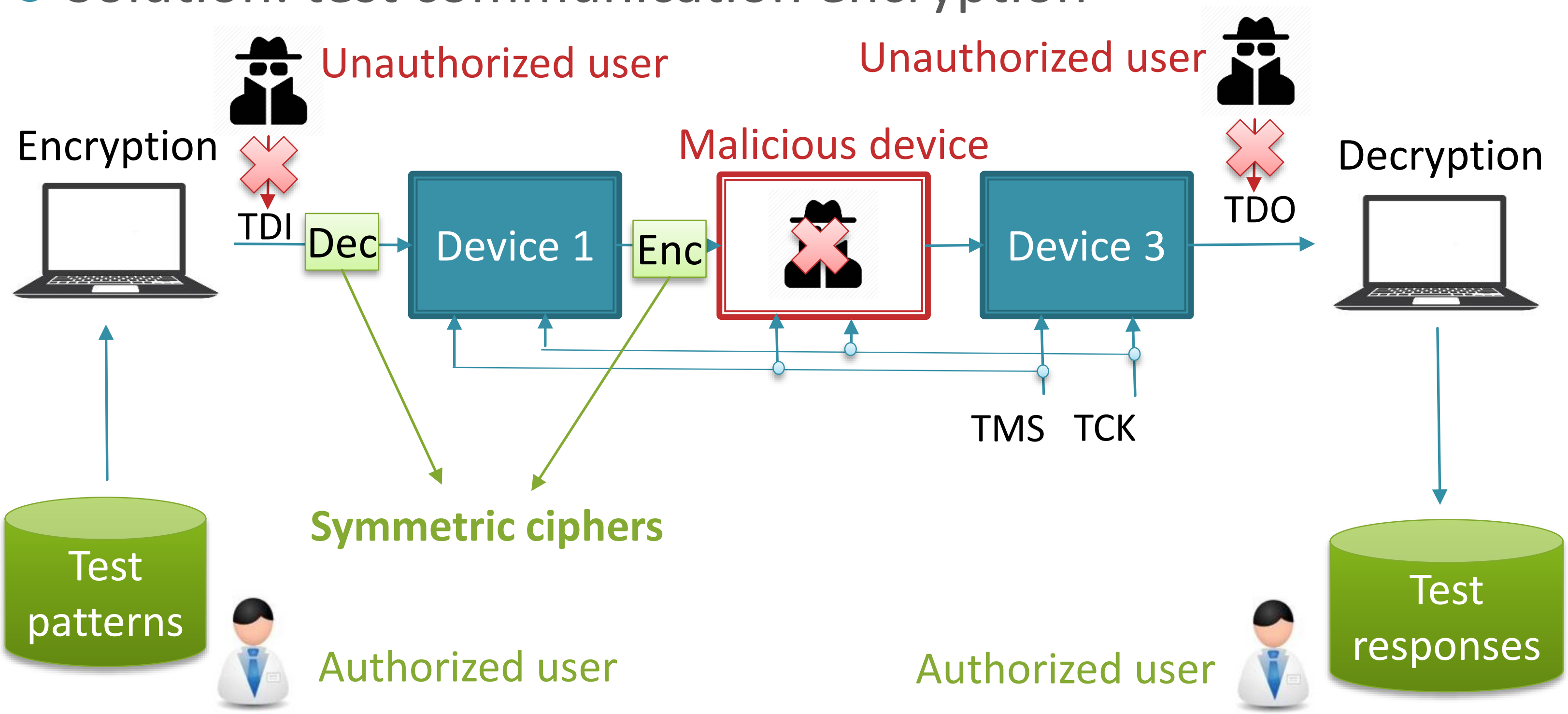
- Solution: test communication encryption



- **Input decryption** prevents sending desired test data
- **Output encryption** prevents reading plain test responses

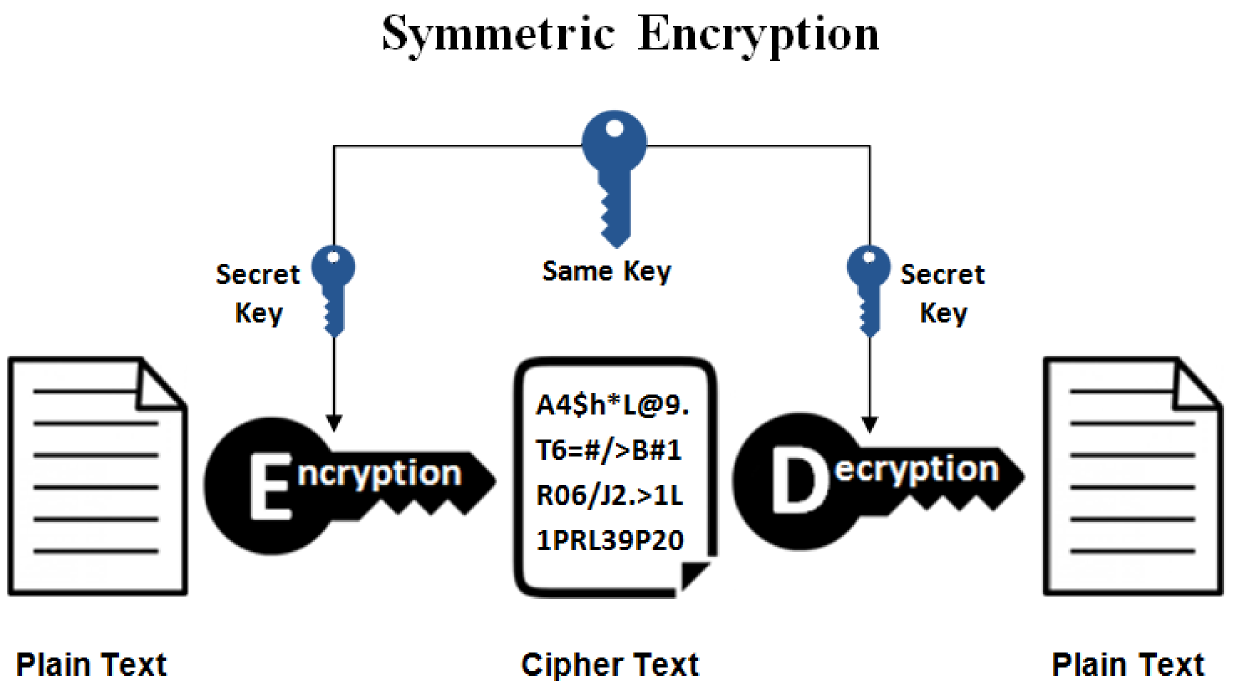
CONTEXT

- Solution: test communication encryption



- Input decryption prevents sending desired test data
- Output encryption prevents reading plain test responses
- Test/debug only possible by authorized user knowing the secret key

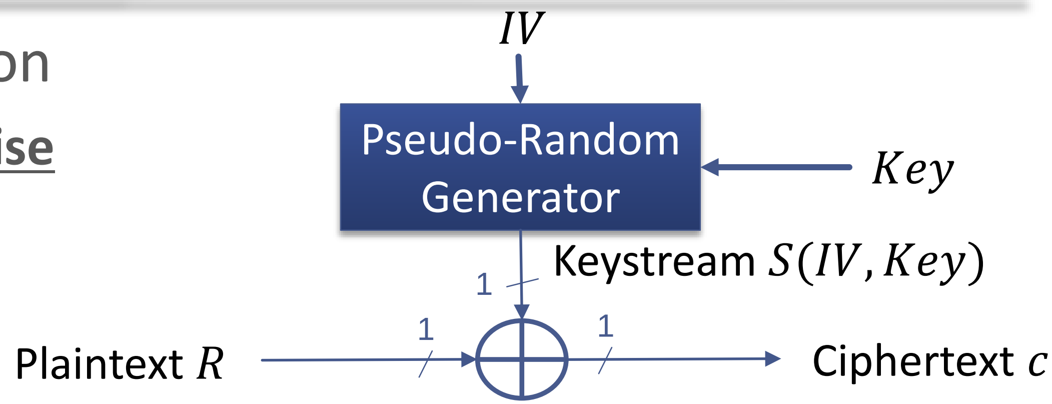
Symmetric cipher



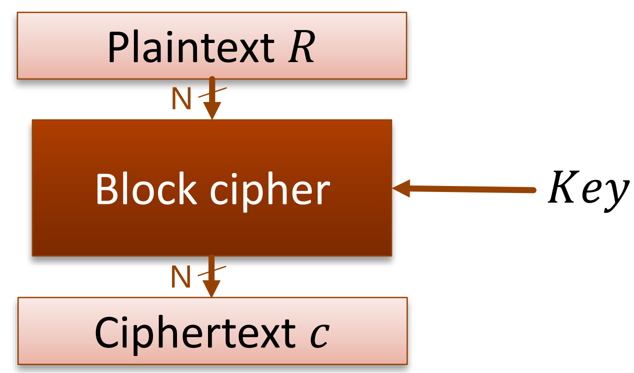
- 2 types of symmetric cipher: stream and block ciphers

Stream cipher / Block cipher

- Stream cipher encryption
  - Keystream XORed **bitwise** with the plaintext



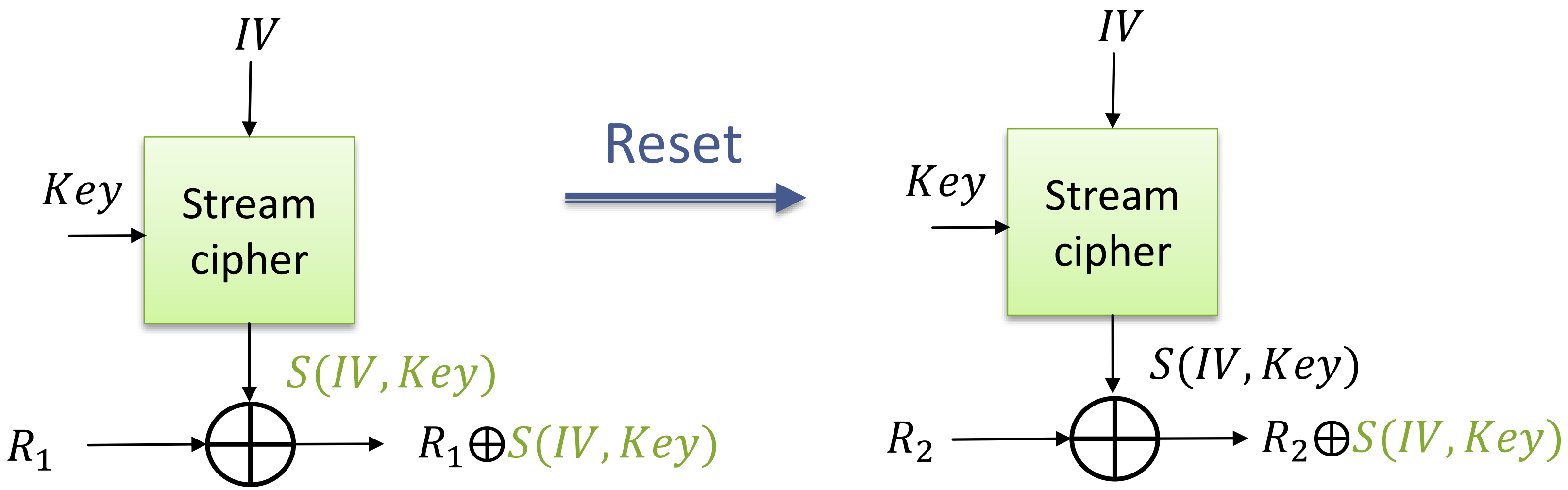
- Block cipher encryption
  - Confusion and diffusion on a **block** of plaintext



- Preference for stream ciphers
  - "Naturally" adapted to serial test communication (JTAG, IEEE 1500, IJTAG)
  - Smaller area footprint compared to block ciphers
  - But ..

Two-times pad: stream cipher requirement

- **Two-times pad**: same key and *IV* re-used => same keystream generated to encrypt different data



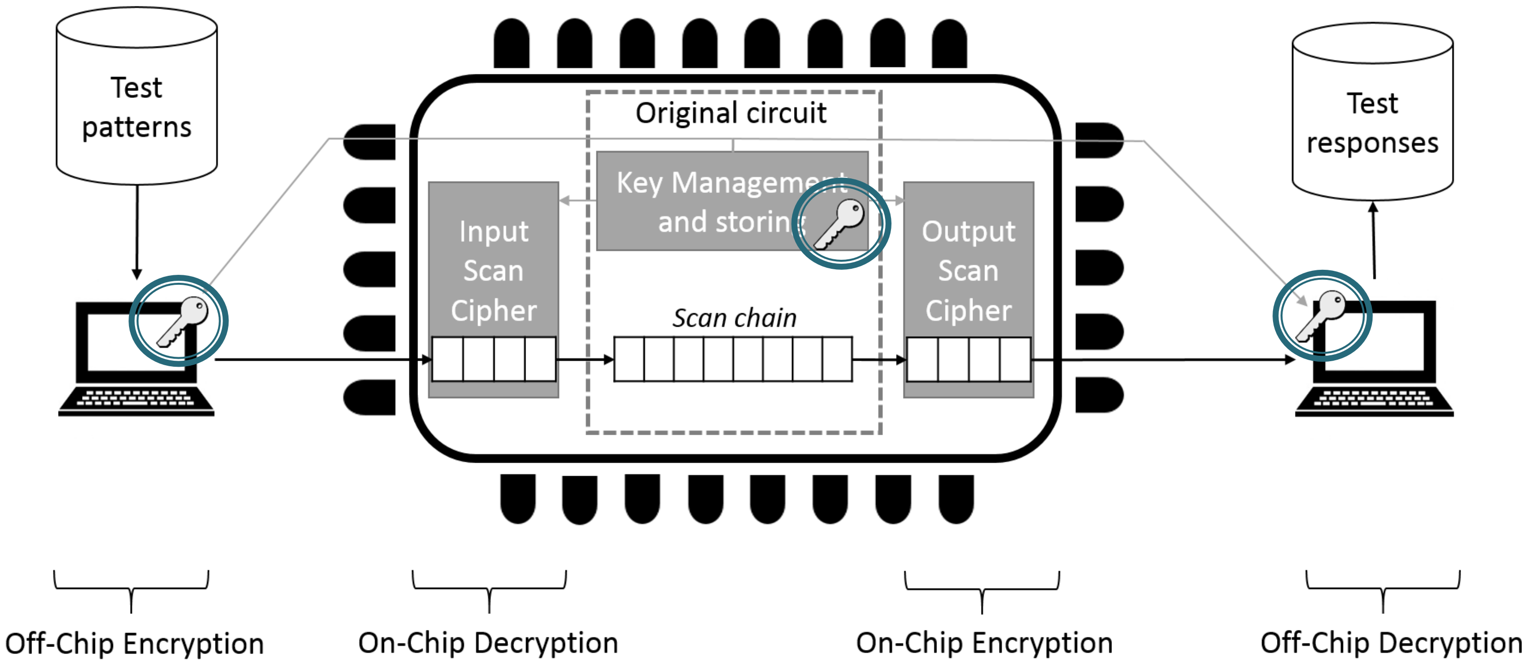
⇒ Possible to carry out attacks if requirement is not fit

$$R1 \oplus \cancel{S(IV, Key)} \oplus R2 \oplus \cancel{S(IV, Key)}$$

⇒ Solution: *IV* generated randomly at each circuit reset

$$R1 \oplus S(IV_1, Key) \oplus R2 \oplus S'(IV_2, Key)$$

BASIC SCHEME



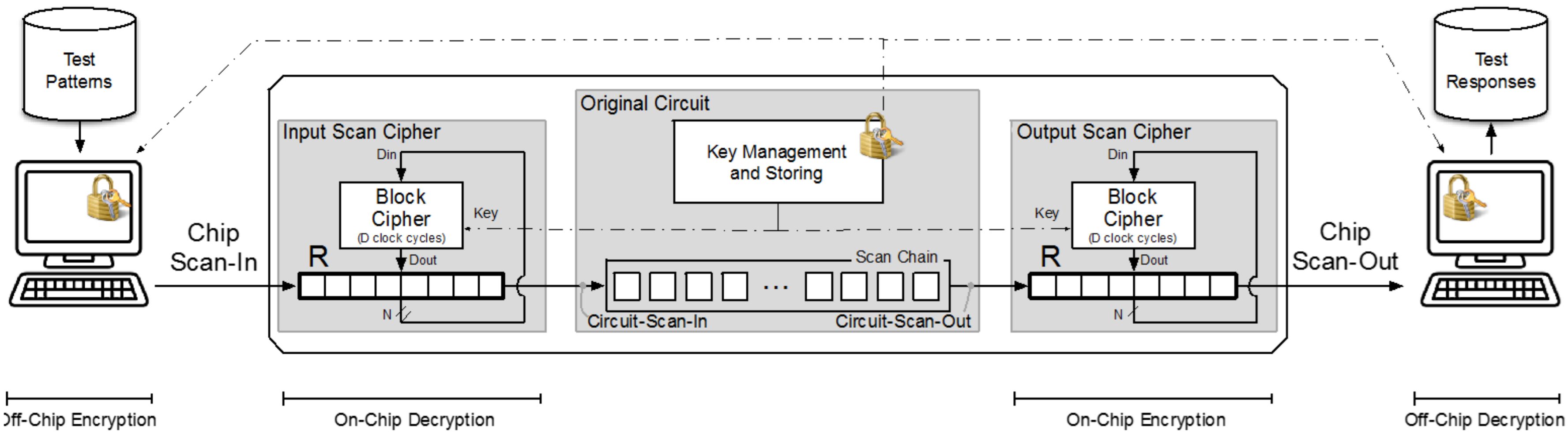
- Assumption: original circuit embedded a crypto-core with its key management and storing
- Scan chain encryption solution shares the key management and storing already implemented

SUMMARY

1) Scan attacks
2) A new countermeasure: Scan chain encryption
3) **Implementation with block cipher**
4) Implementation with stream cipher
5) Conclusion

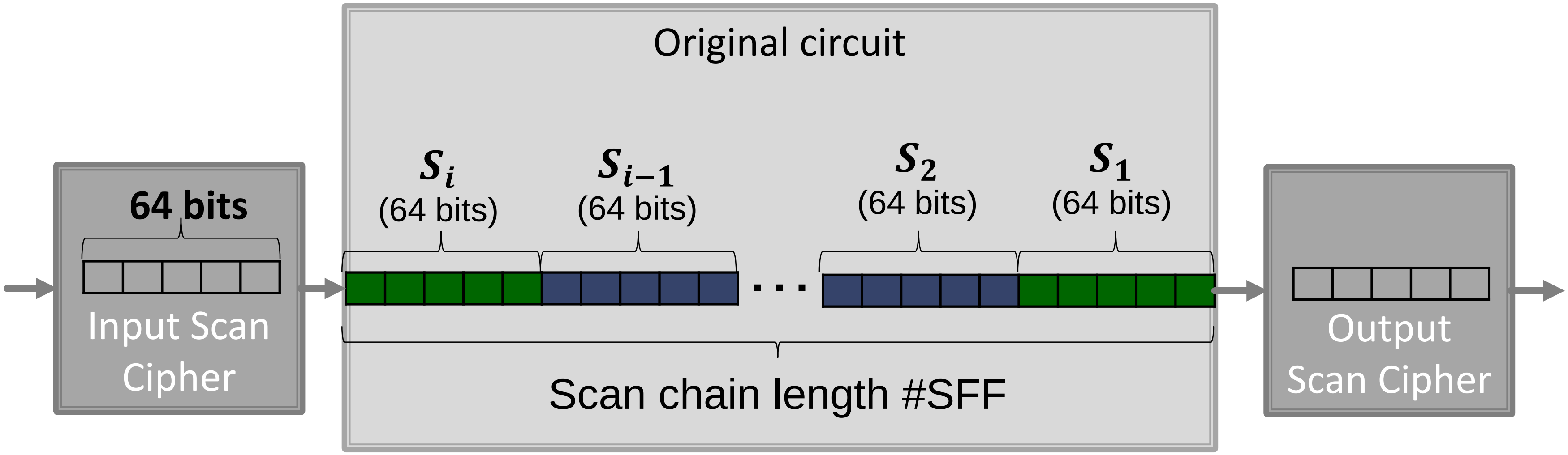
Block Cipher-based Scan Encryption

- Implementation on scan chain with 2 PRESENT block ciphers:
  - Lightweight (1 PRESENT = 2 139 GE)
  - Encryption by 64-bits block size



MODE OF OPERATIONS

- 64 bits encrypted every 32 clock cycles

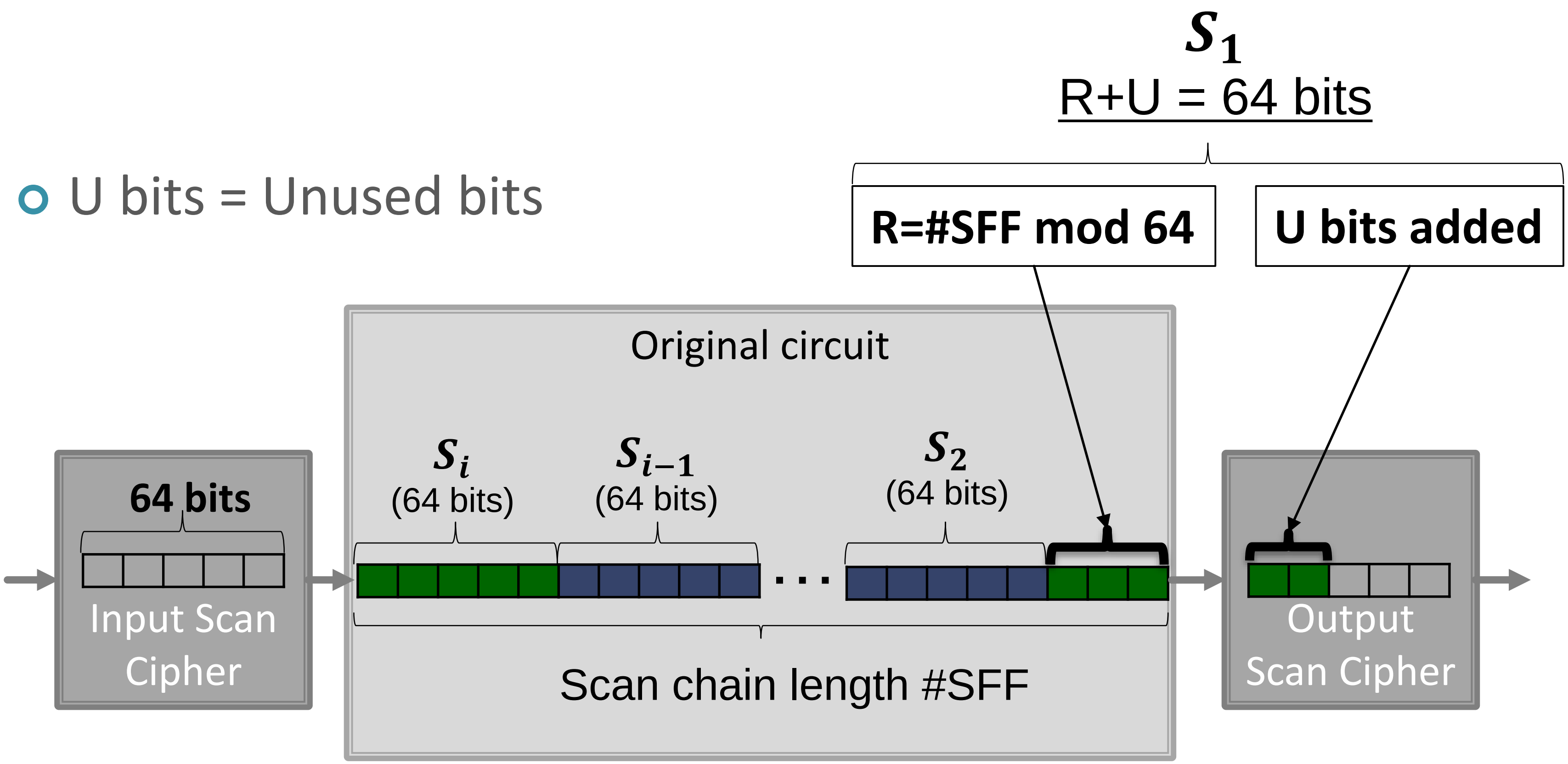


⇒ **#SFF = Px64**

⇒ **No test time overhead on each pattern**

Mode of operations

- U bits = Unused bits



⇒ **#SFF = Px64 + R**

⇒ **Loss of U clock cycles <u>per pattern</u>**

Summary

1) Scan attacks

2) A new countermeasure: Scan chain encryption

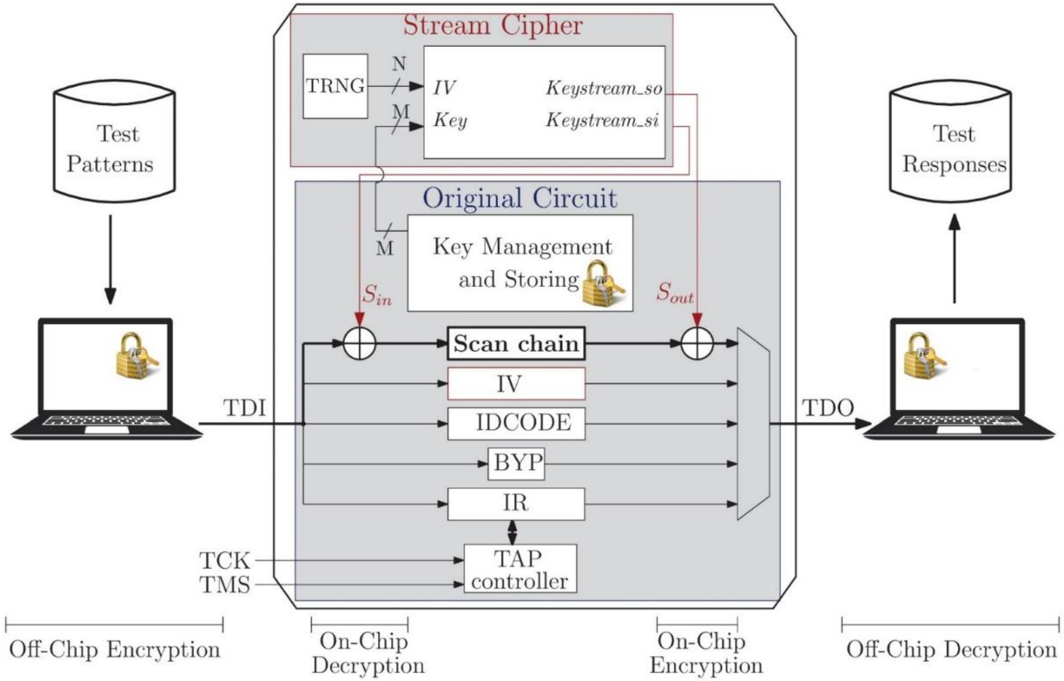
3) Implementation with block cipher

4) **Implementation with stream cipher**

5) Conclusion

Stream cipher-based scan encryption

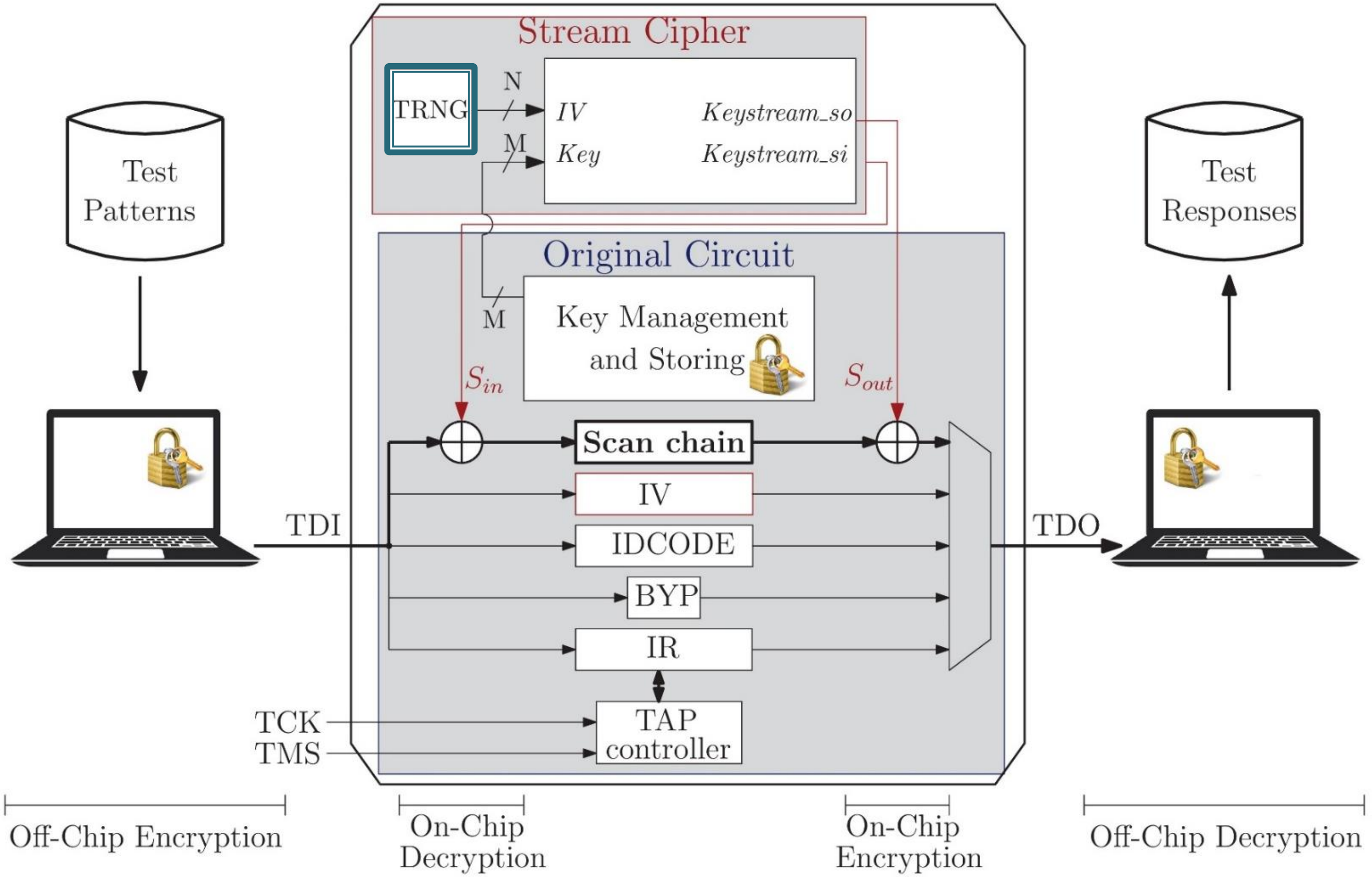
- Implementation on JTAG:
  - 1 TRIVIUM stream cipher (2 016 GE)
  - TRNG to generate random *IV*
  - New instruction *GetIV* with a test data register IV



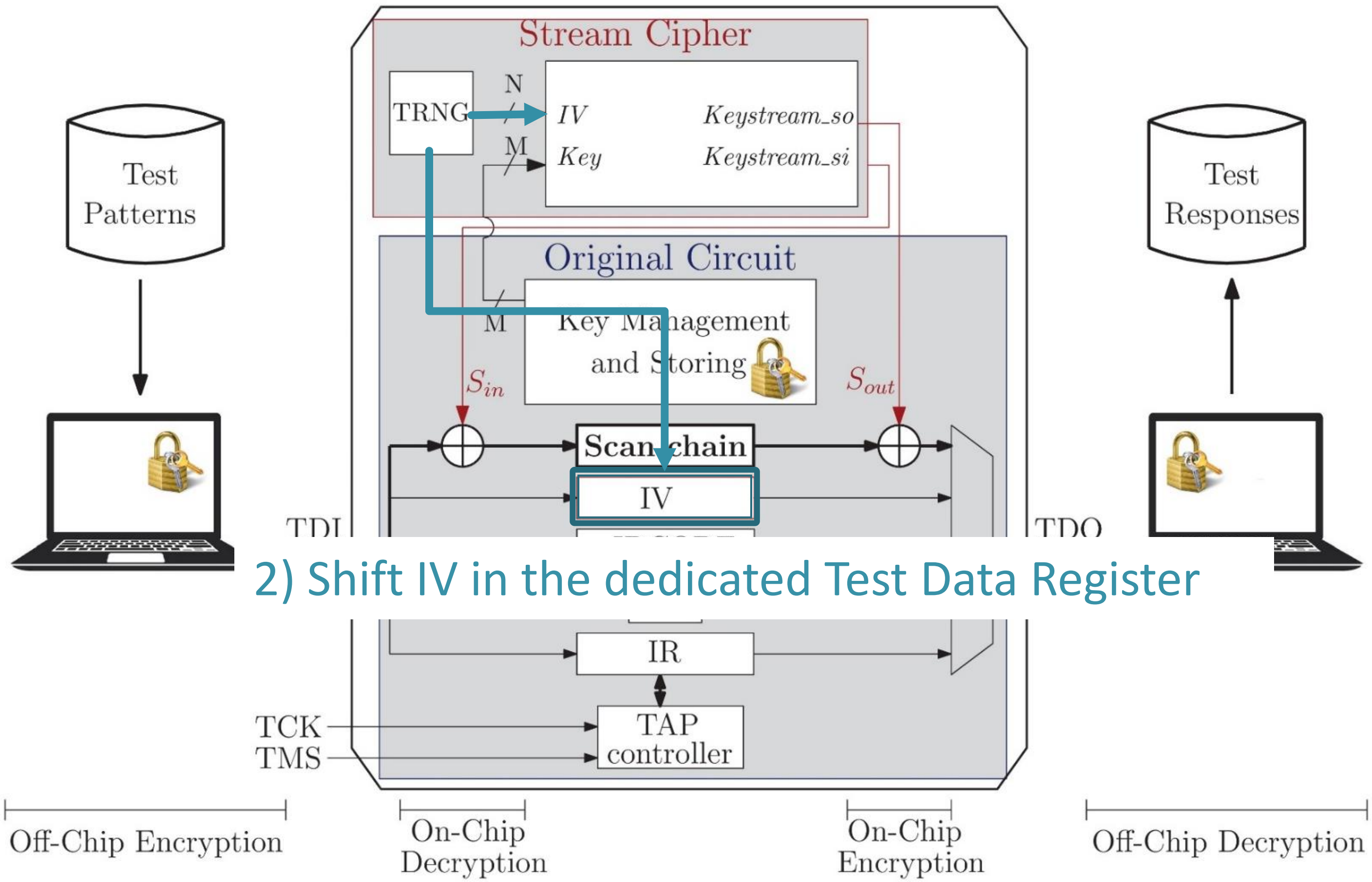
- Mode of operations in 2 phases: initialization and encryption

INITIALIZATION PHASE

1) TRNG initialization: reach sufficient entropy to generate random number



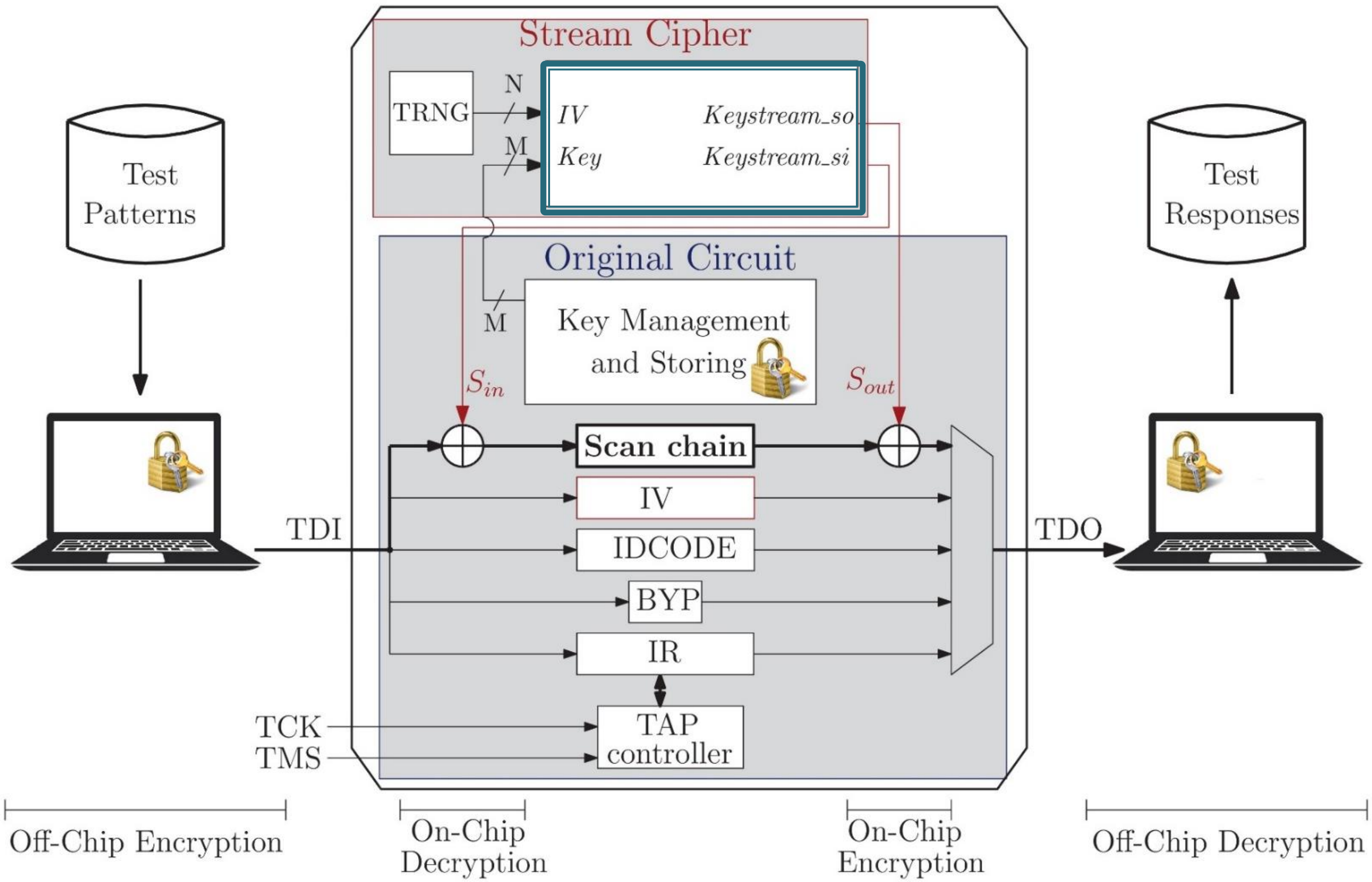
INITIALIZATION PHASE



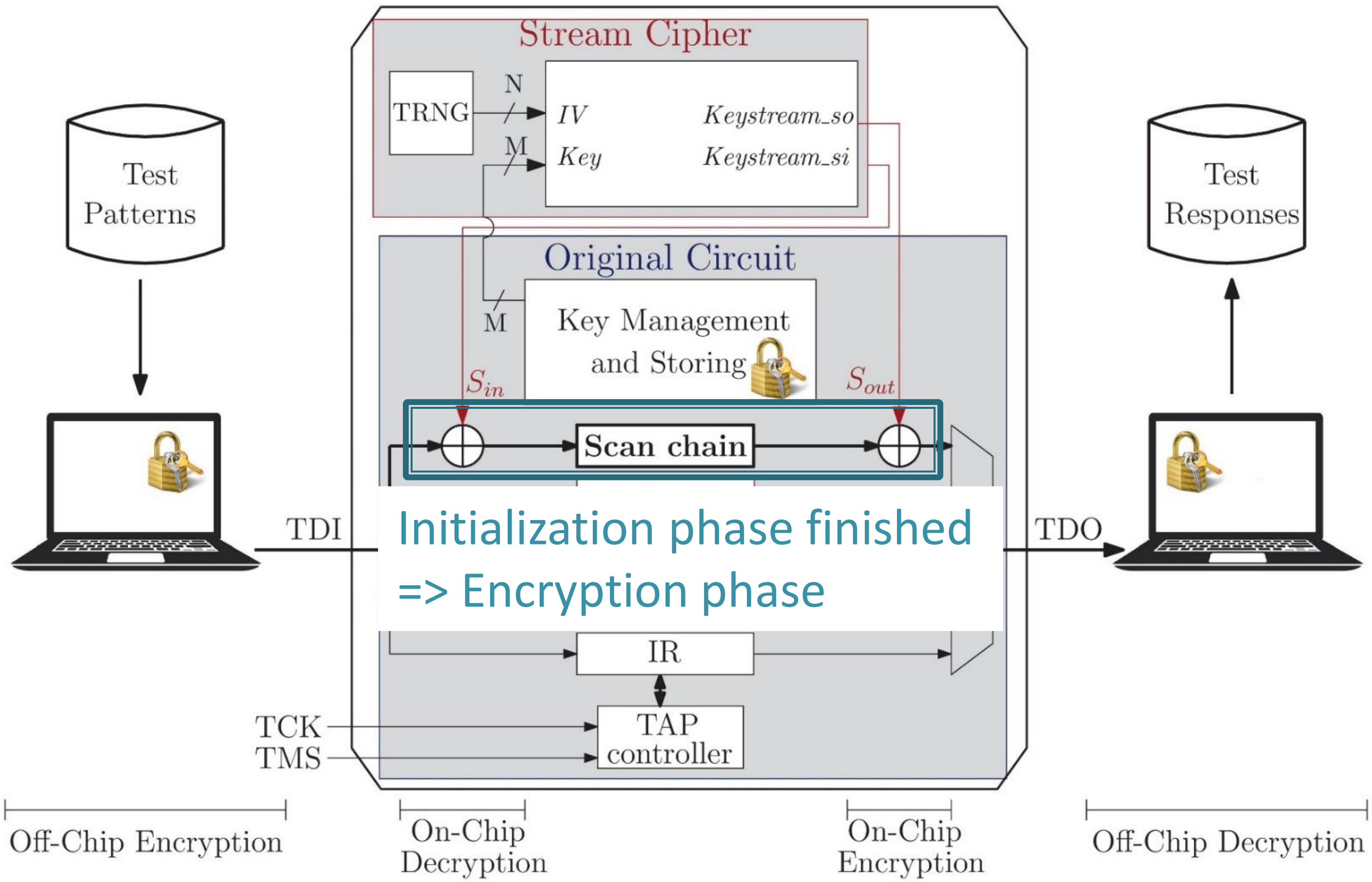
2) Shift IV in the dedicated Test Data Register

INITIALIZATION PHASE

3) Stream cipher setup



INITIALIZATION PHASE

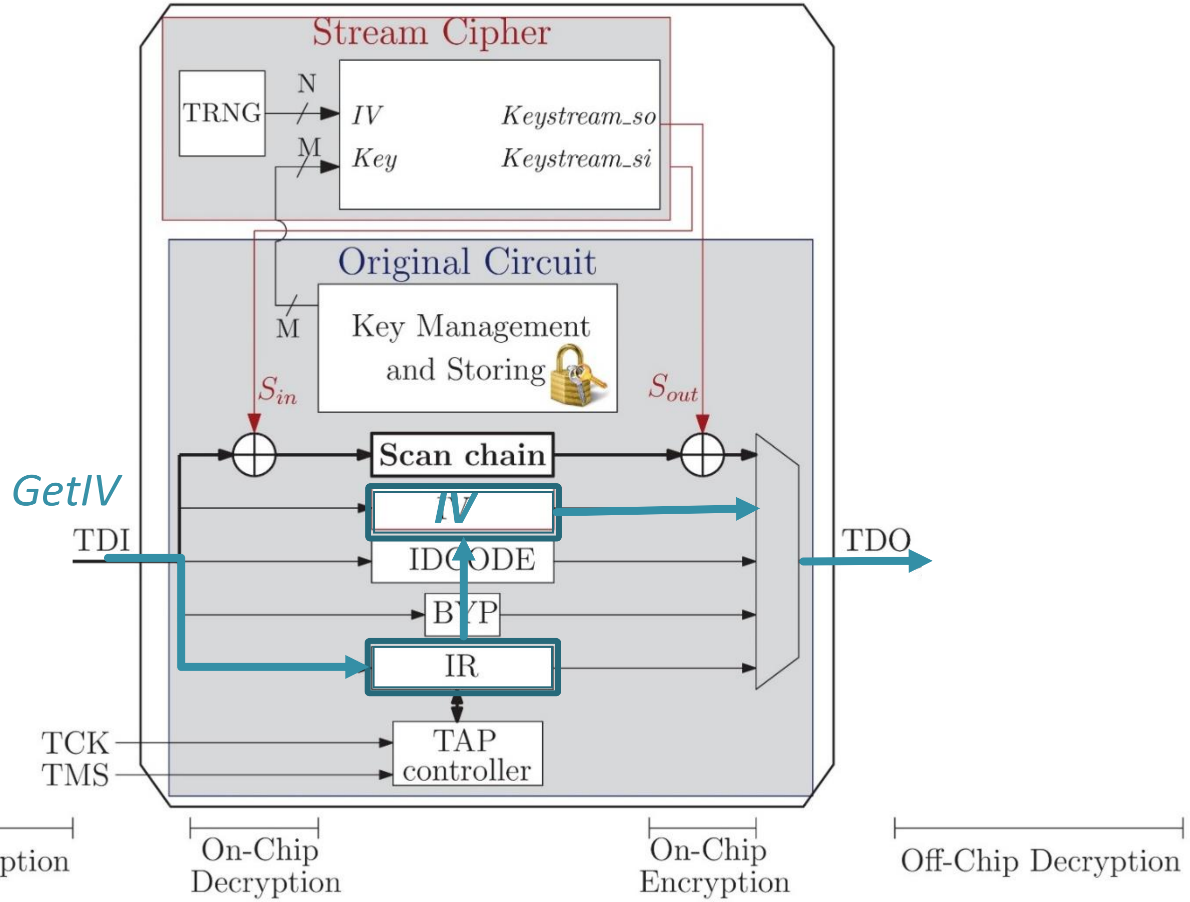


Initialization phase finished
=> Encryption phase

Encryption phase

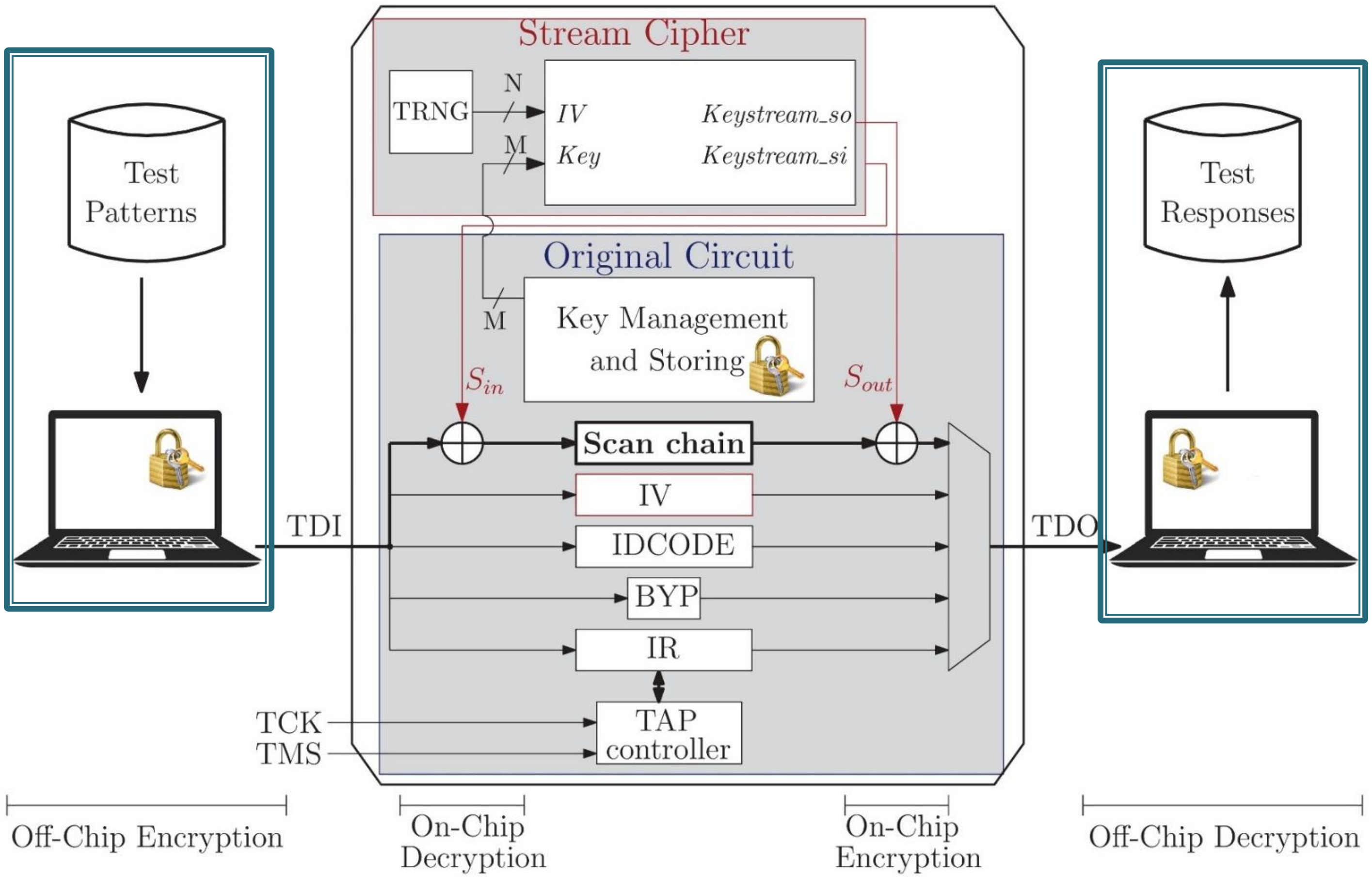
- Send *GETIV* instruction

⇒ Shift the content of the IV register out the circuit



ENCRYPTION PHASE

- User can encrypt and decrypt test data with the **obtained *IV*** and the **shared secret key**



Time for the initialization process

- T_{TRNG_init} to initialize the TRNG
- 80 clock cycles to shift the *IV* in the register
- 1 152 clock cycles for the stream cipher setup

Original circuit	Triple-DES	Pipelined AES-128	Pipelined AES-256	RSA 1024	LEON3
Test time* (clock cycles)	687 101	1 944 877	4 559 845	39 405 239	11 612 051
Test time overhead					
Block-based solution (%)	+0.31	+0.81	+0.006	+0.33	+0.004
Stream-based solution (%)**	+0.18	+0.06	+0.03	+0.003	+0.01

*: Test time considered for a fault coverage of 100%, except for LEON3 where it reaches 70%

**: test time overhead without the initialization of the TRNG

Summary

1) Scan attacks
2) A new countermeasure: Scan chain encryption
3) Implementation with block cipher
4) Implementation with stream cipher
5) **Conclusion**

Comparison between both solutions

	Block cipher-based solution (PRESENT)	Stream cipher-based solution (TRIVIUM)
Security		
- Scan attacks	Protected	Protected (two times pad not possible)
- Malicious core	Protected	Protected
Cost		
- Area	10 658.96 µm²	5 408.52 µm² (+ 31 200 µm² for TRNG)
- Test time	Depends on the scan length (multiple or not of the block size)	Clock cycles required for the initialization phase
Integration		
- Diagnosis & debug	Still possible in-field	
- Key management	Re-use key management already implemented	
- Integration in test daisy-chain	Possible issue with the padding of test data	No issue

Thank You

Acknowledgements

- FUI#20 TEEVA Project

- Partners