UNSW SYDNEY

Hardware Security
Week 3 -
Scan Attacks

26T1
Hammond Pearce

Slide material acknowledgements to
Ramesh Karri, Benjamin Tan,
JV Rajendran, Jason Blocklove
Christian Pilato, Luca Collini

RECAP

1. What is an S-Box?
2. How are permutation tables implemented in hardware?
3. How many rounds does DES have?
4. How many clock cycles do those rounds take with the example hardware?
5. What is the difference between asymmetric key crypto and symmetric key crypto?
6. Is AES asymmetric or symmetric? How about RSA?

RECAP: Cryptographic Hardware

- Custom hardware can accelerate cryptographic operations
- It can also keep secrets safe (e.g. ROM for round keys)
- Example applications
  - Smart/chip credit cards
  - TV/Satellite signal decoders
- Accelerators found in most modern CPUs
  - AES-NI instructions - Intel, AMD
  - ARMv8 instructions
  - Scalar and Vector Cryptographic Instruction Set Extensions for RISC-V

Let’s talk about Hardware Design

When we build an integrated circuit,

how do we know it works?

All integrated circuits need testing

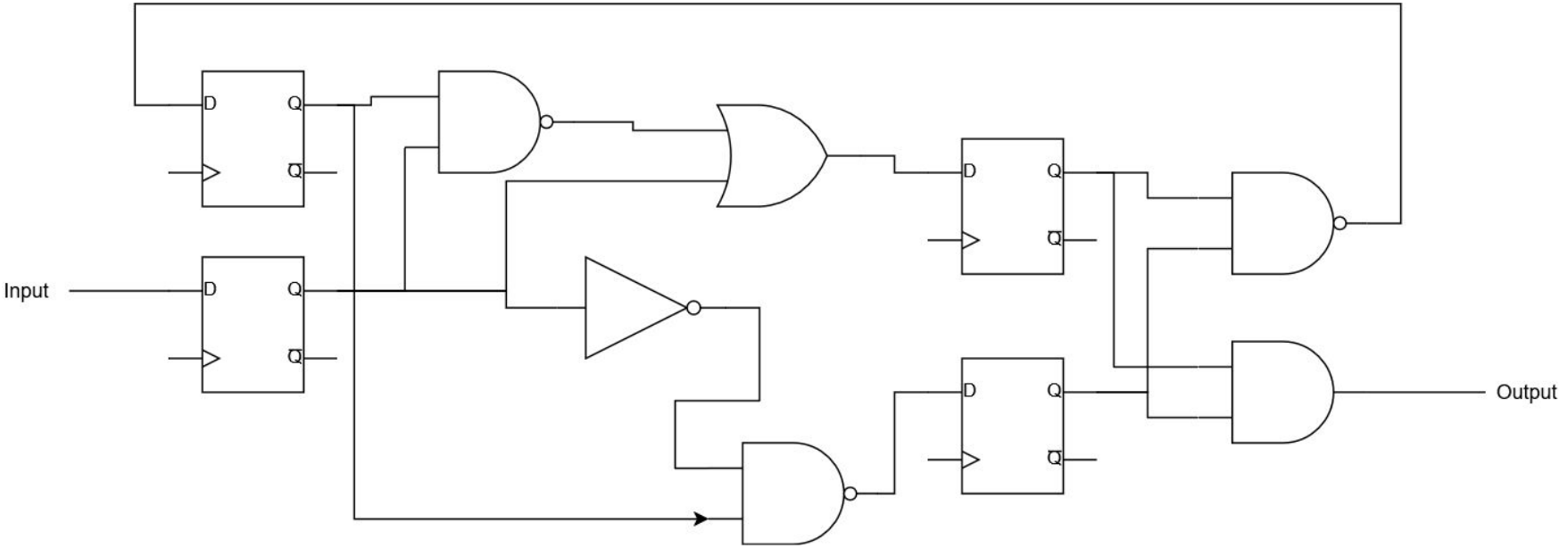
- Need to minimise/eliminate risks of hardware malfunction
  - Test chips before they are used in the field
- Detect faults early in manufacturing
  - Save cost on packaging etc for faulty dies
- Eliminate security risks
  - Faults might allow adversaries to reverse engineer parts of your design

Possible test strategies:

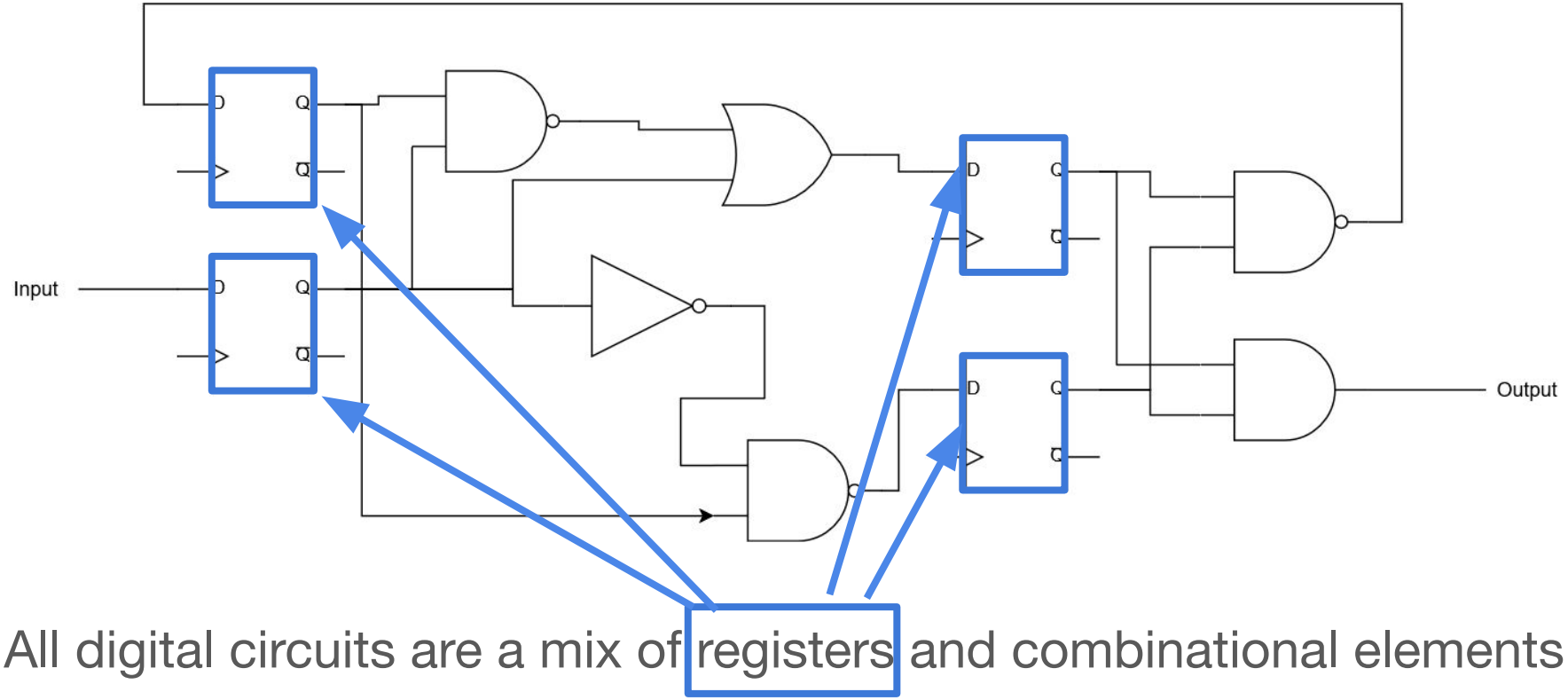
1. Built in self-test?
    a. (i.e. build some test system and controller into your design)
    b. Advantage: self contained
    c. Disadvantage: probably low coverage, high overhead (area), how do you know if the self-test works?
2. Post-silicon functional testing?
    a. Probably low coverage
    b. How to test specific wires/bits?
    c. How to understand a failure?

What might be better strategies?

Let’s look at a hardware circuit

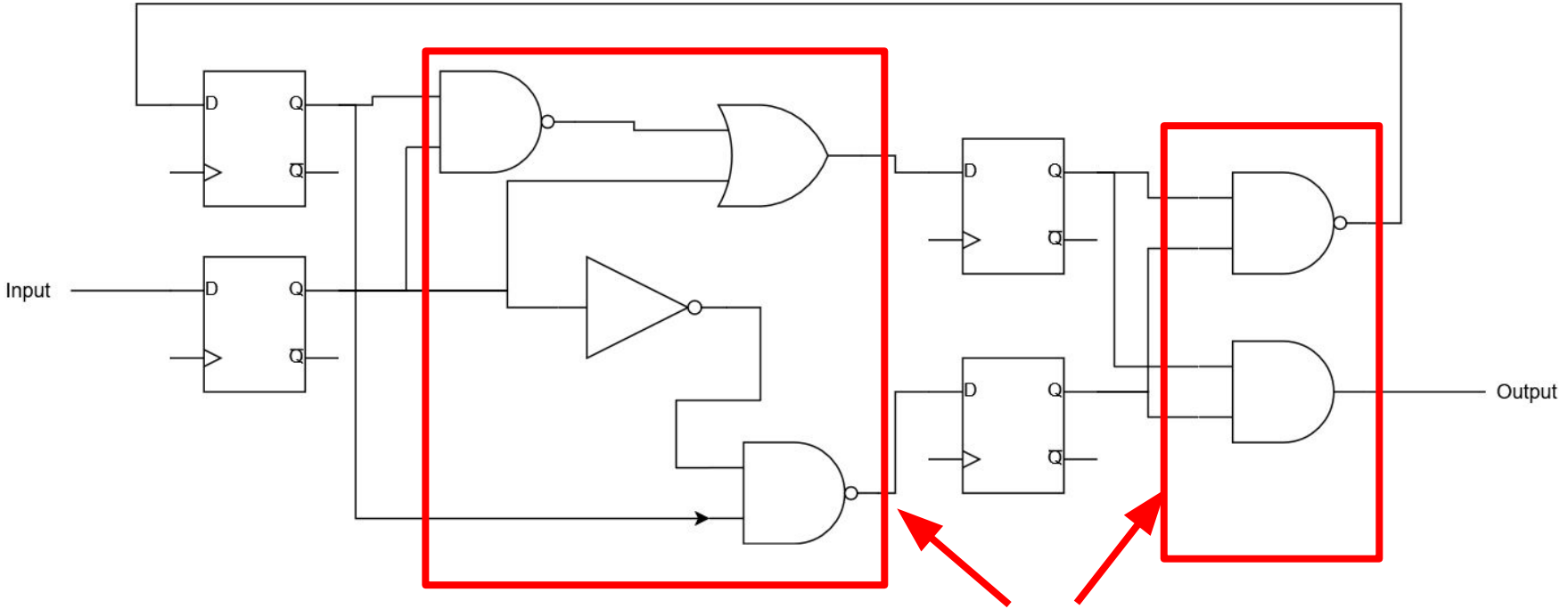


Let’s look at a hardware circuit



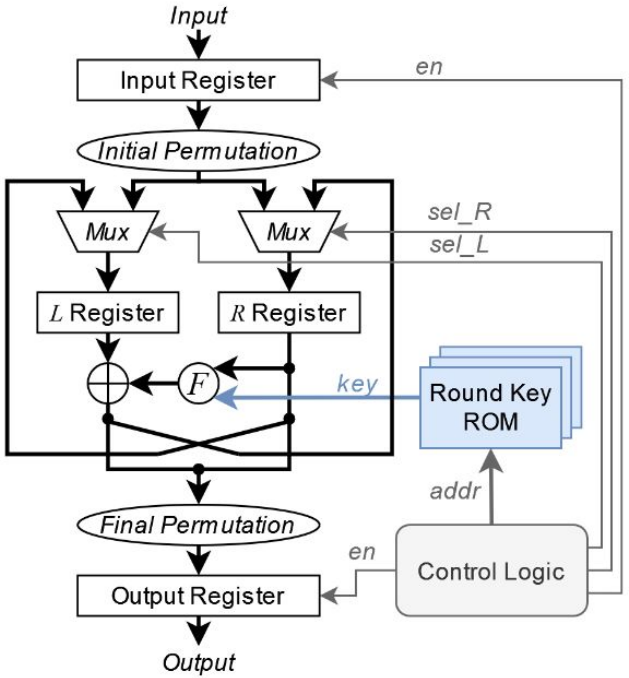
All digital circuits are a mix of registers and combinational elements

Let’s look at a hardware circuit



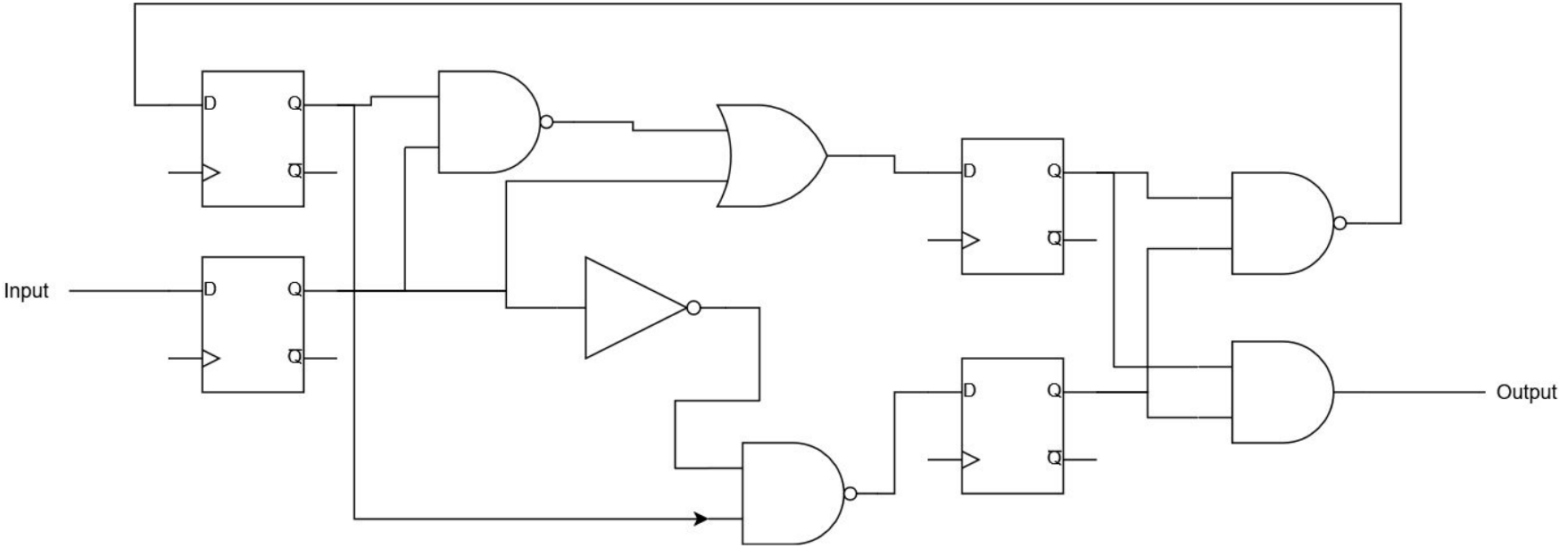
All digital circuits are a mix of registers and combinational elements

Another example hardware circuit



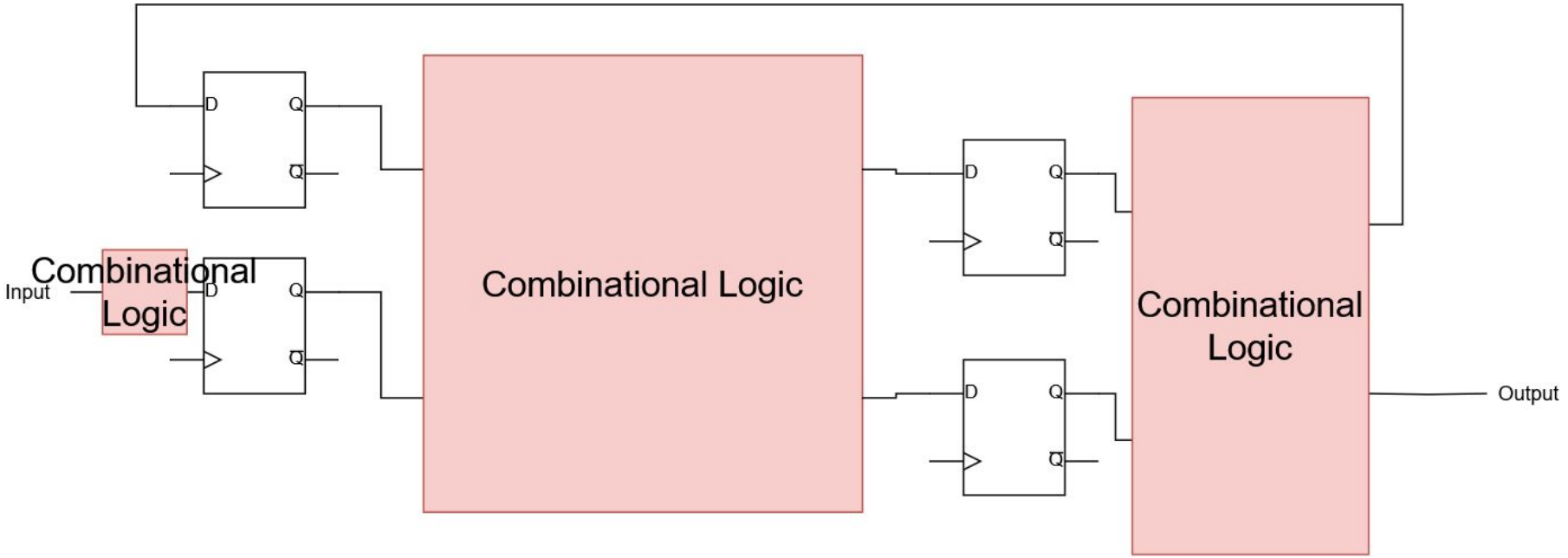
All digital circuits are a mix of registers and combinational elements

Back to first example



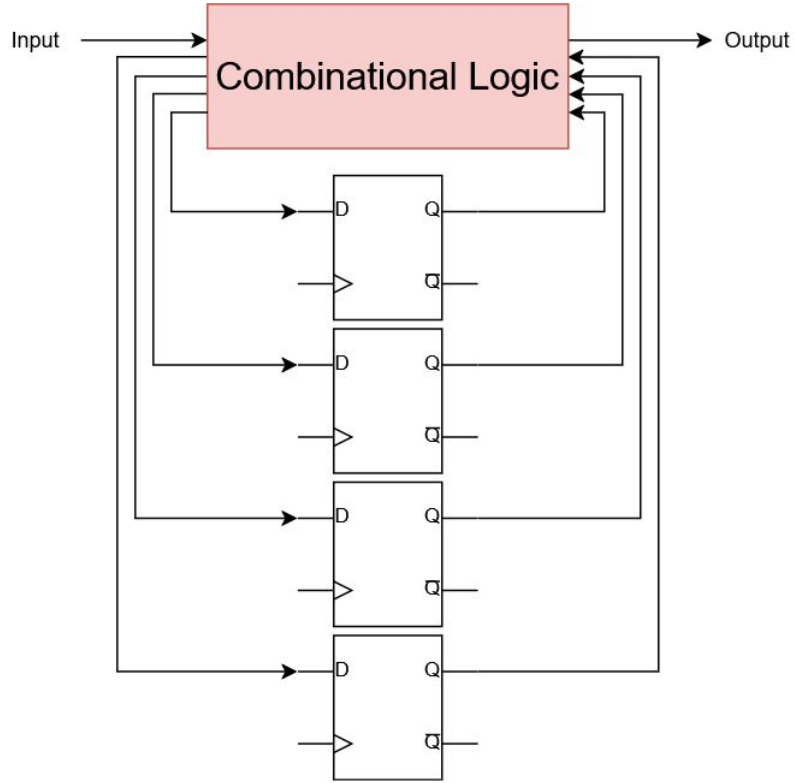
How can we test any random location in this circuit?

Back to first example



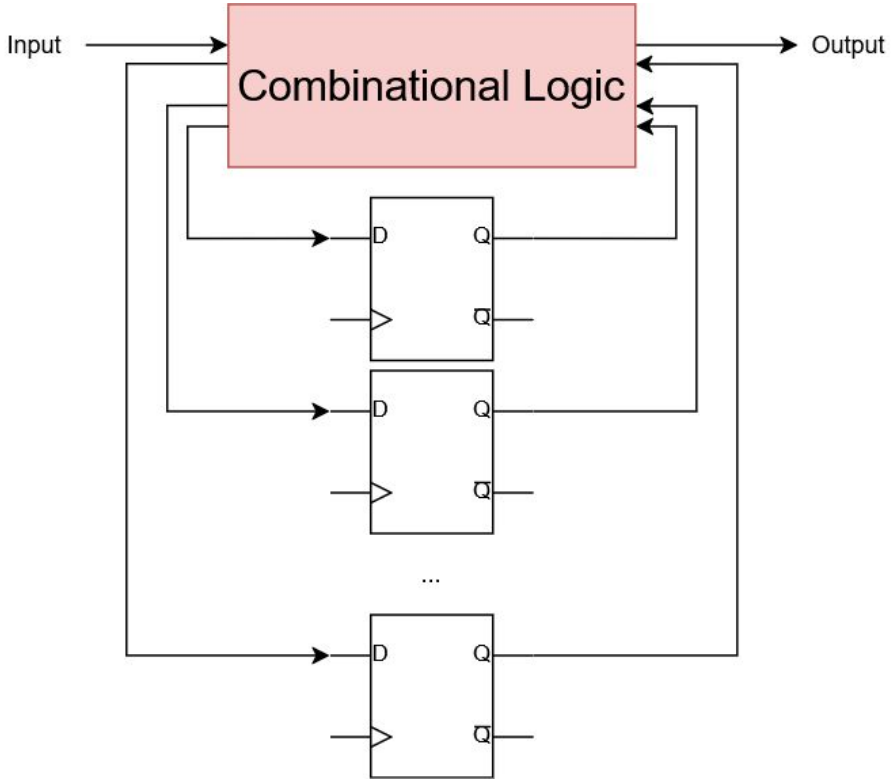
Let's generalize the combinational and register parts

Back to first example



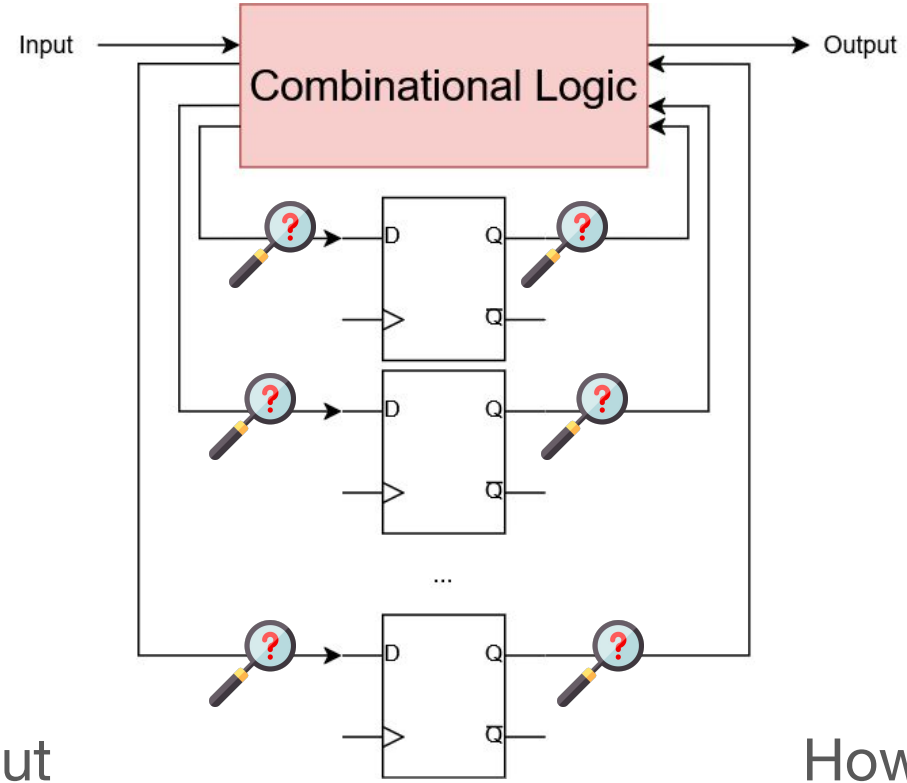
And rearrange:

Back to first example



And generalize:

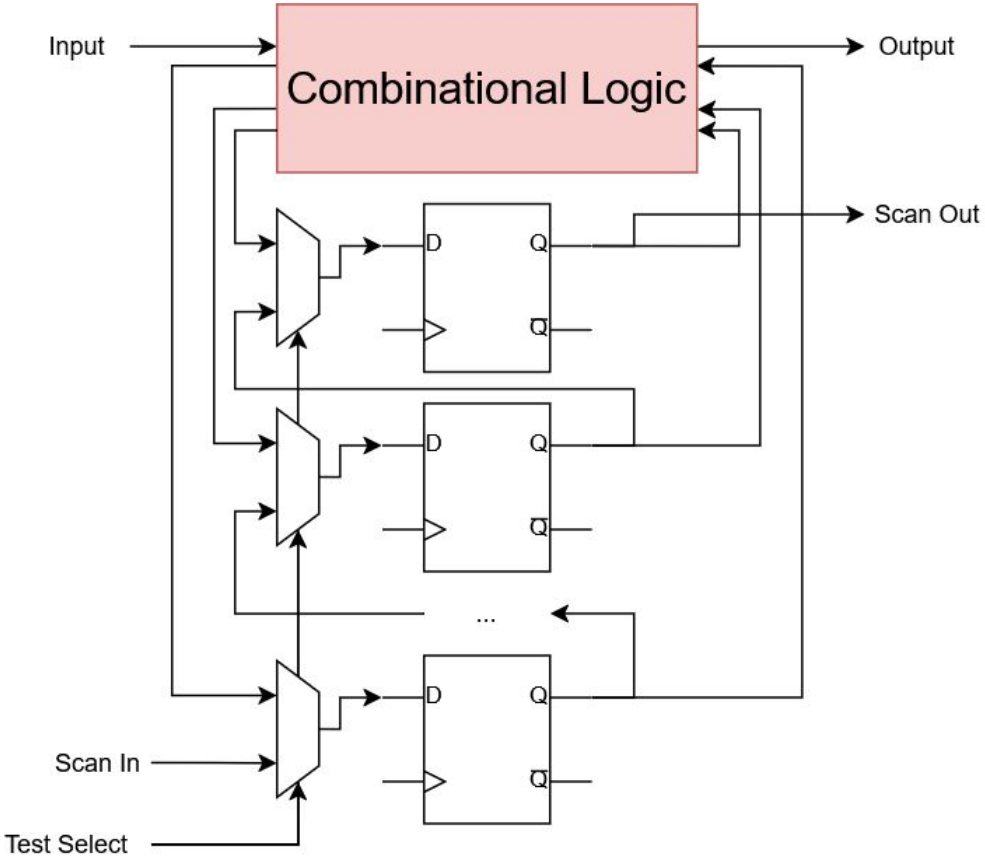
“Design-for-Test” (DFT): Inspecting I/O



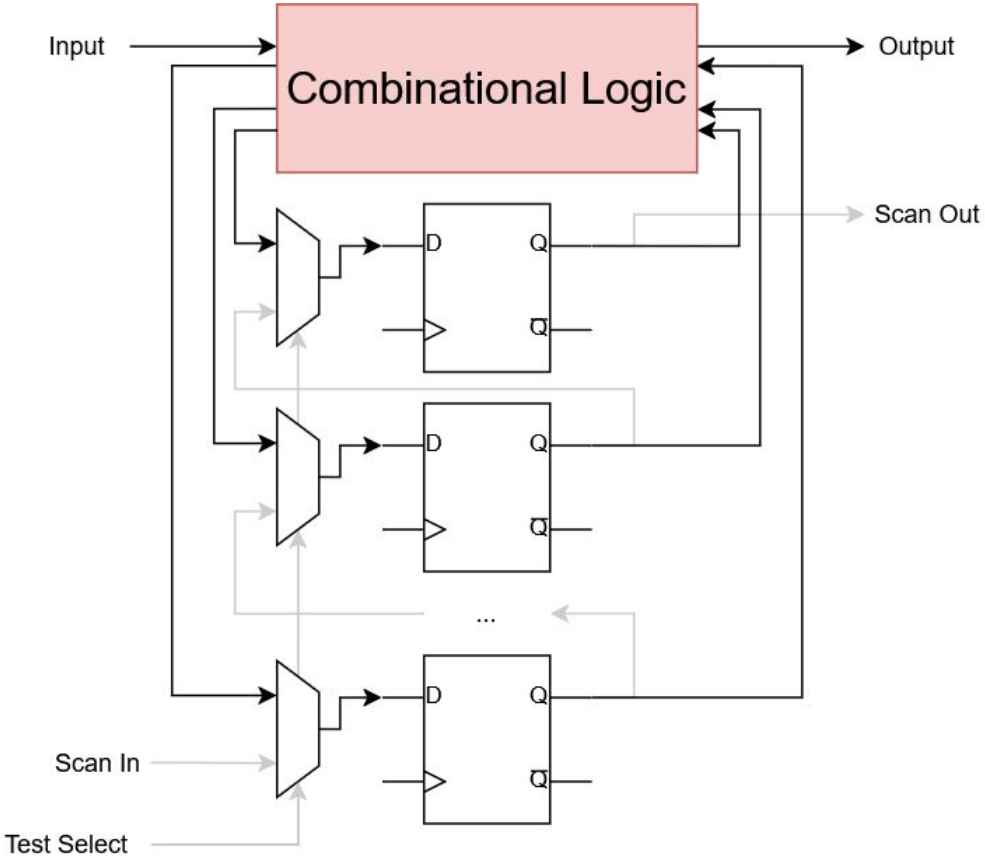
Test Input→Output mappings

How to implement?

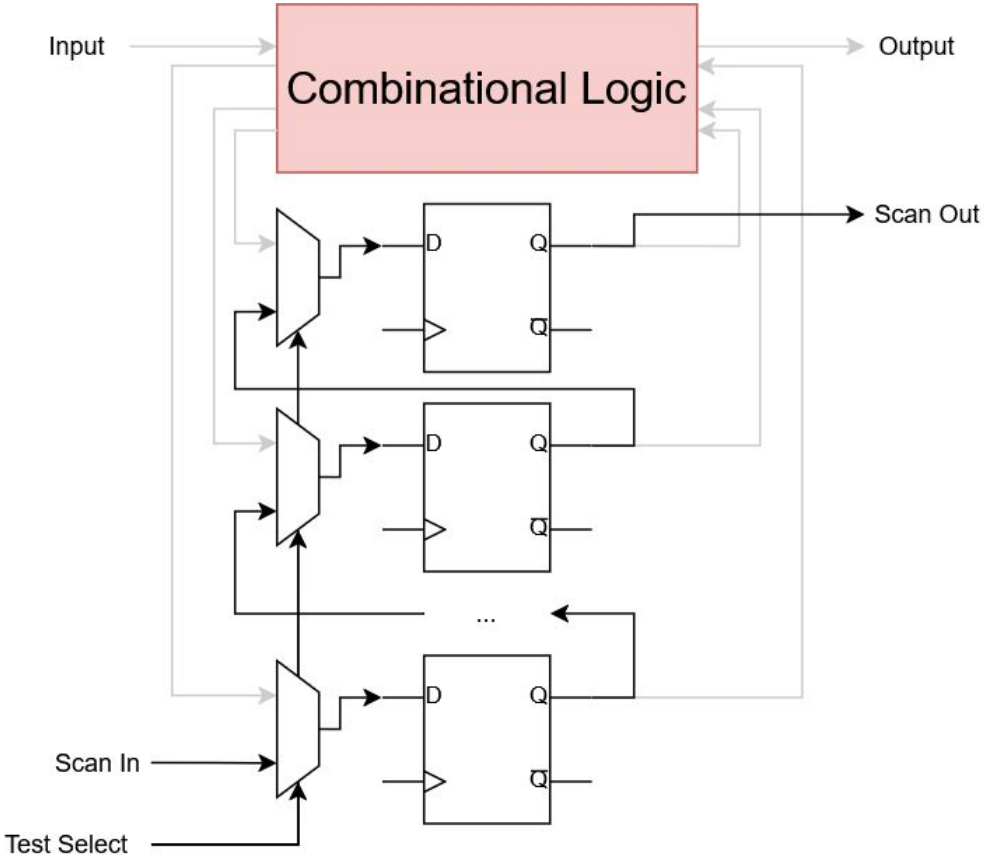
“Design-for-Test” (DFT): the scan chain



“Design-for-Test” (DFT): normal operation



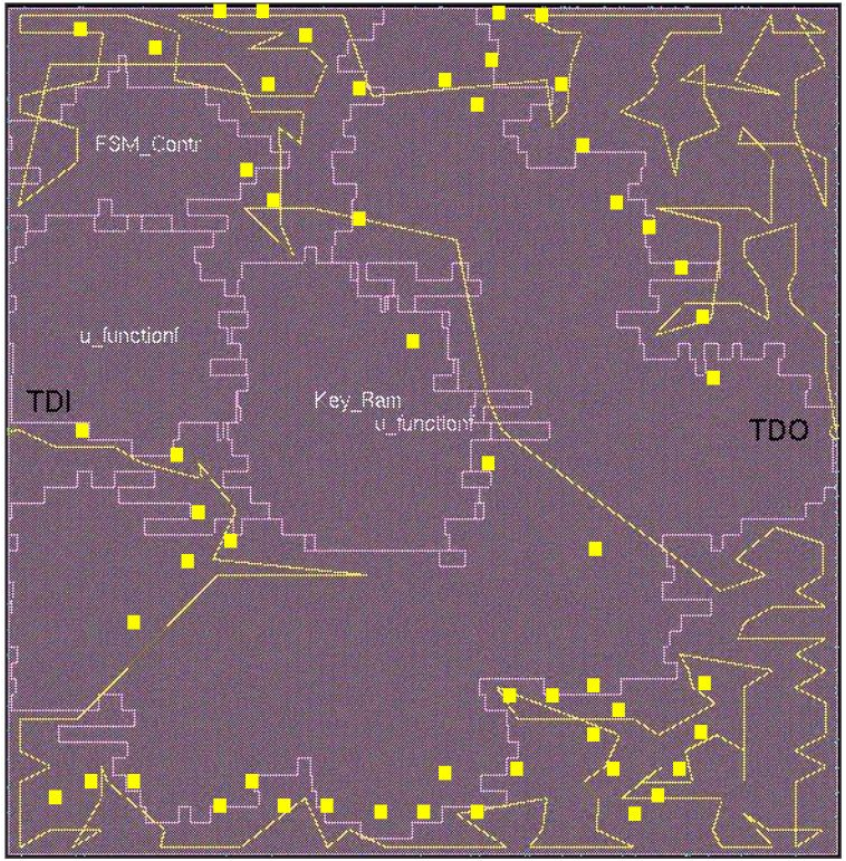
“Design-for-Test” (DFT): scan operation



Scan chains are extremely popular

- 80% of ICs use scan chains for test/debug/validation
- Scan DFT is widely supported
  - Fast Scan/TestKompress: Mentor Graphics
  - DFT compiler/TetraMAX ATPG: Synopsys
- Readback and test infrastructure in FPGAs
  - Load configuration bitstream from external PROM
  - Readout bitstream for debug

Scan chains in a DES chip



?? security?

- We can test any part of our circuit now!
- But… cryptography says:
  - Never permit access to partially encrypted data (intermediate rounds)
  - Never permit access to secret keys (all or part)
  - Detect system failure immediately
  - Detect attempt of unauthorized access immediately

Kerckoff' security principle: An attacker has full knowledge of the algorithm and the design, but not the key…

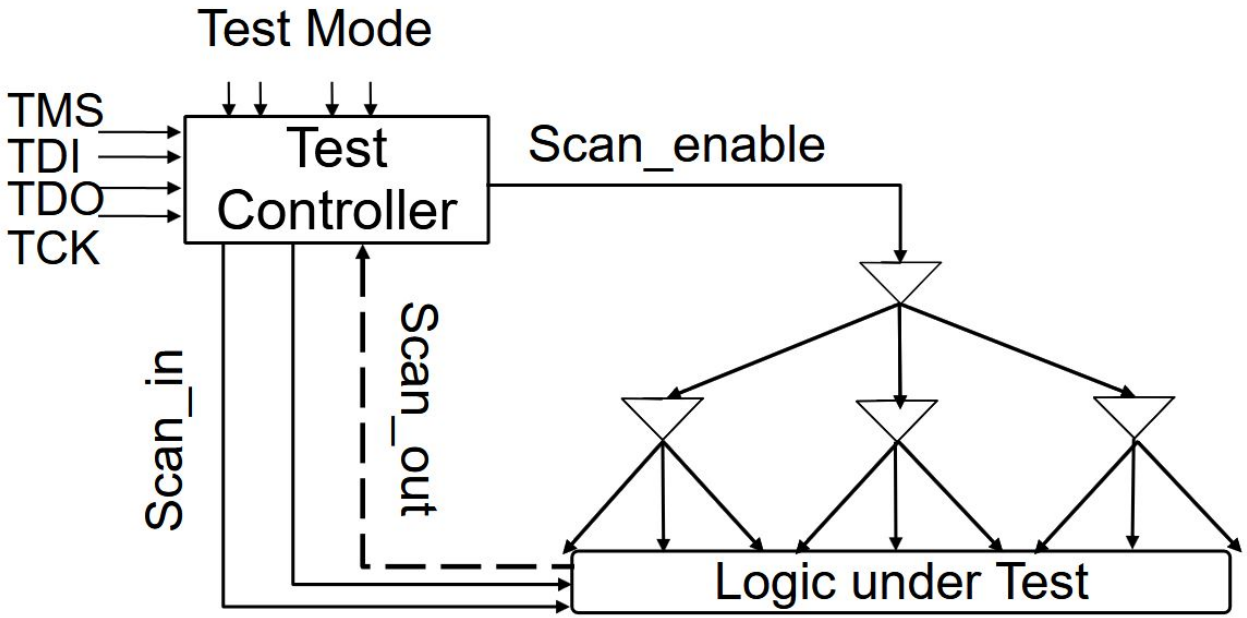
SCAN ATTACK

- Step 1: Access scan chain
  - Approach L: Get lucky (it's already exposed)
  - Approach A: bypass any test authentication
  - Approach B: activate scan chain using a physical attack

- Step 2: Use scan chain to leak secrets
  - Approach A: Observe (normal operations → scan out)
  - Approach B: Control+Observe (scan in → normal → scan out)

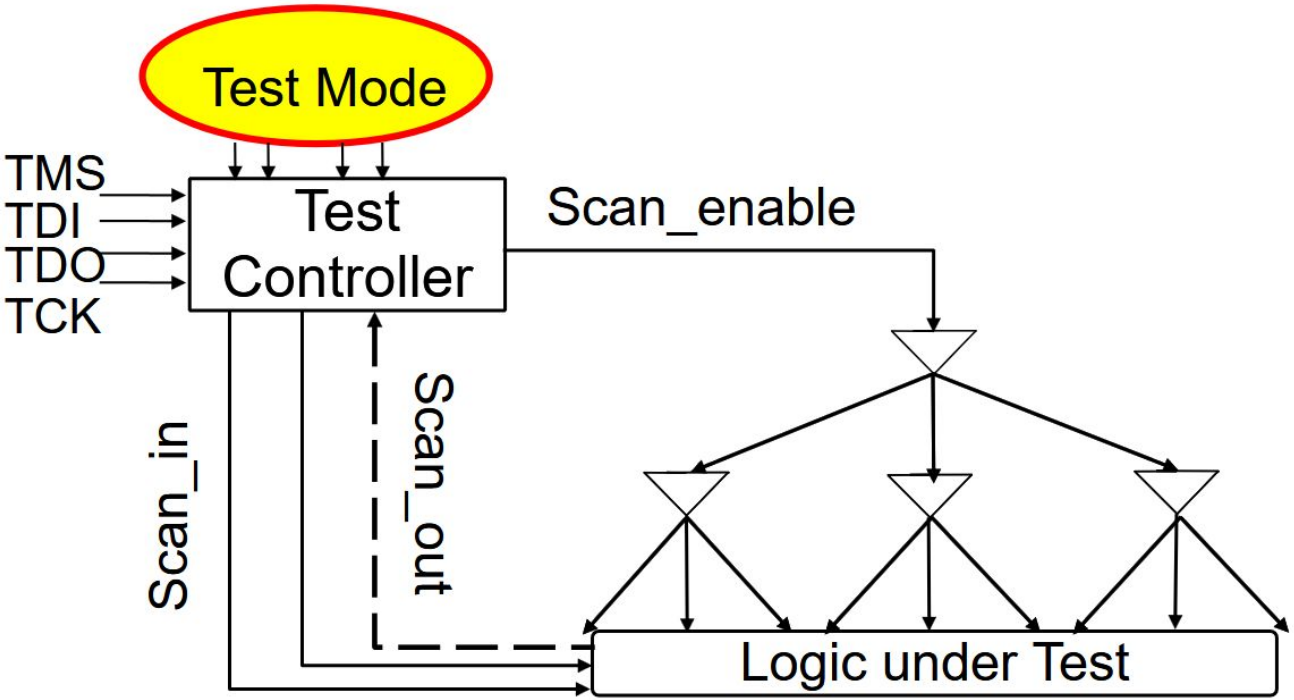
Accessing the scan chain: Get lucky!

No protections!



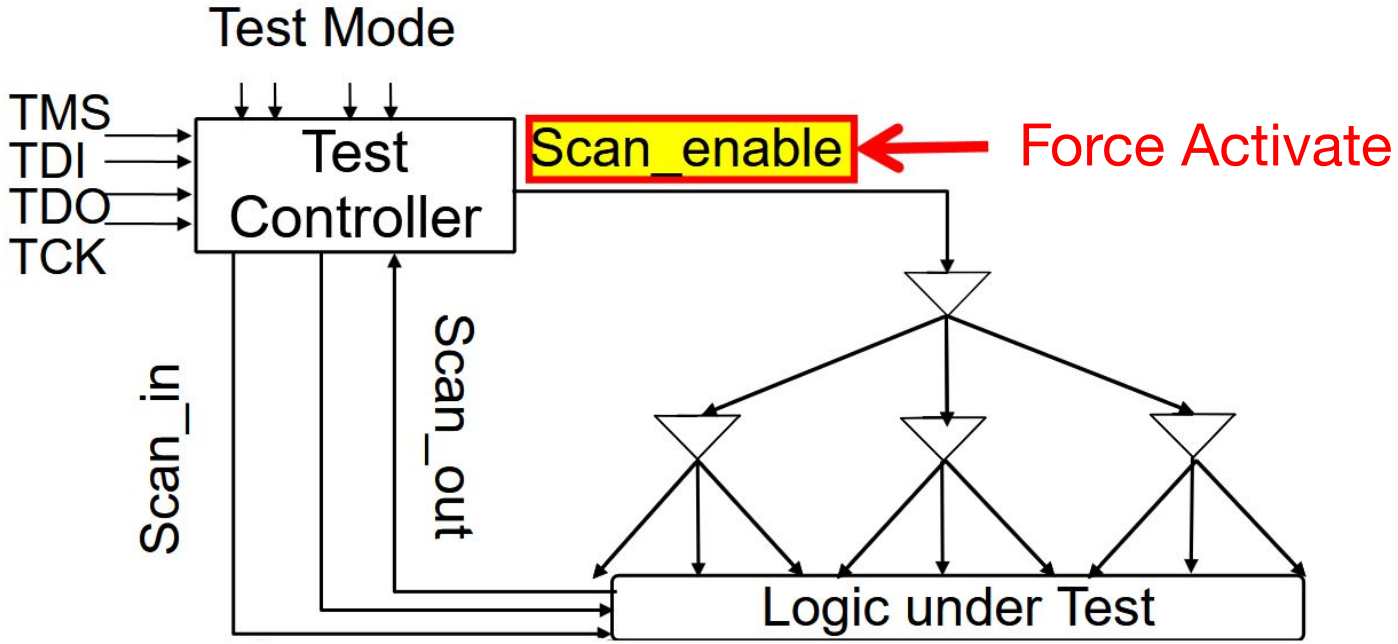
Access the scan chain: Bypass test auth.

Corrupt authentication? Security malfunction? Insider?

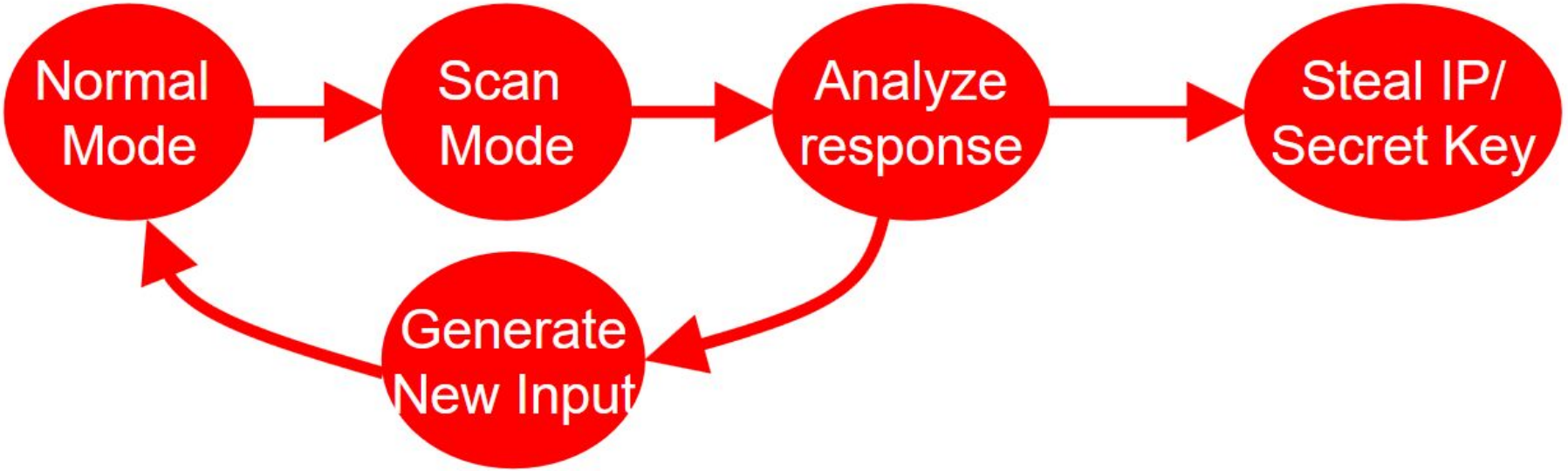


Access the scan chain: Physical Attack

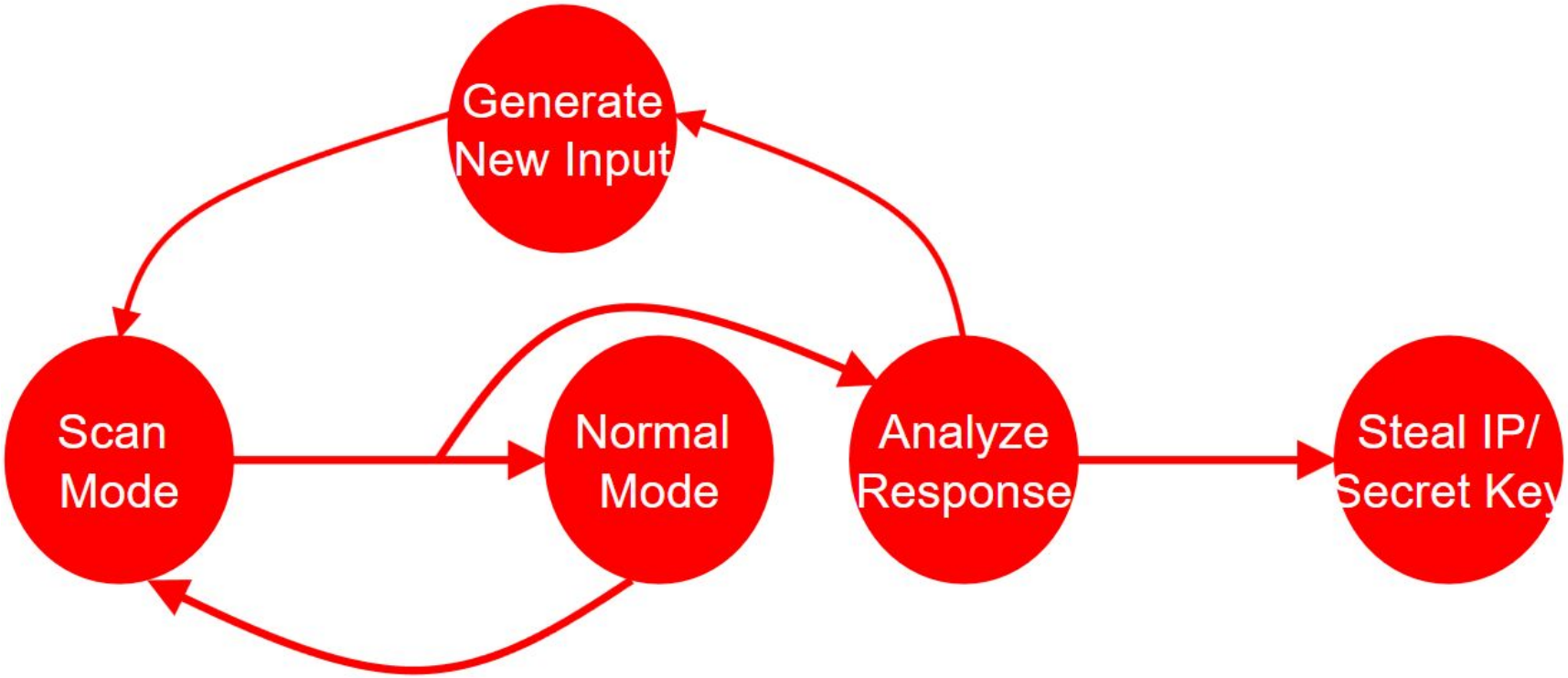
Probing requirement: -Expertise in semi tech, -knowledge of IC
-Expensive equipment, -minor mod. to IC



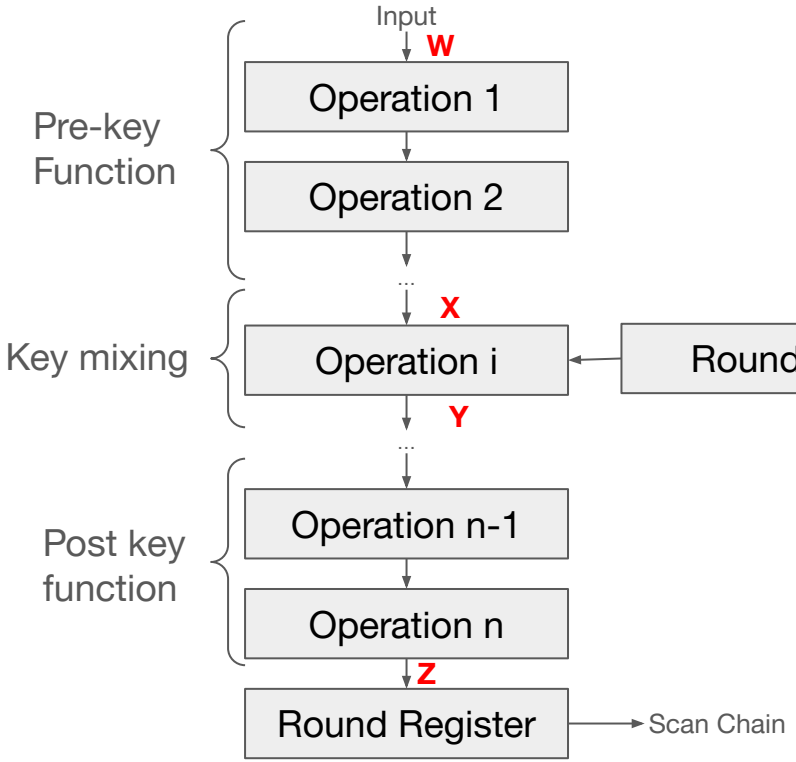
Use scan chain to observe internal state



Or: Scan chain to control + observe state

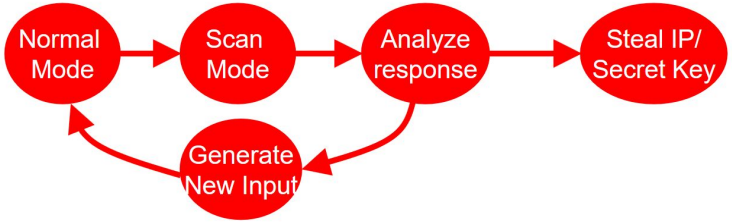


Mounting a scan-based attack



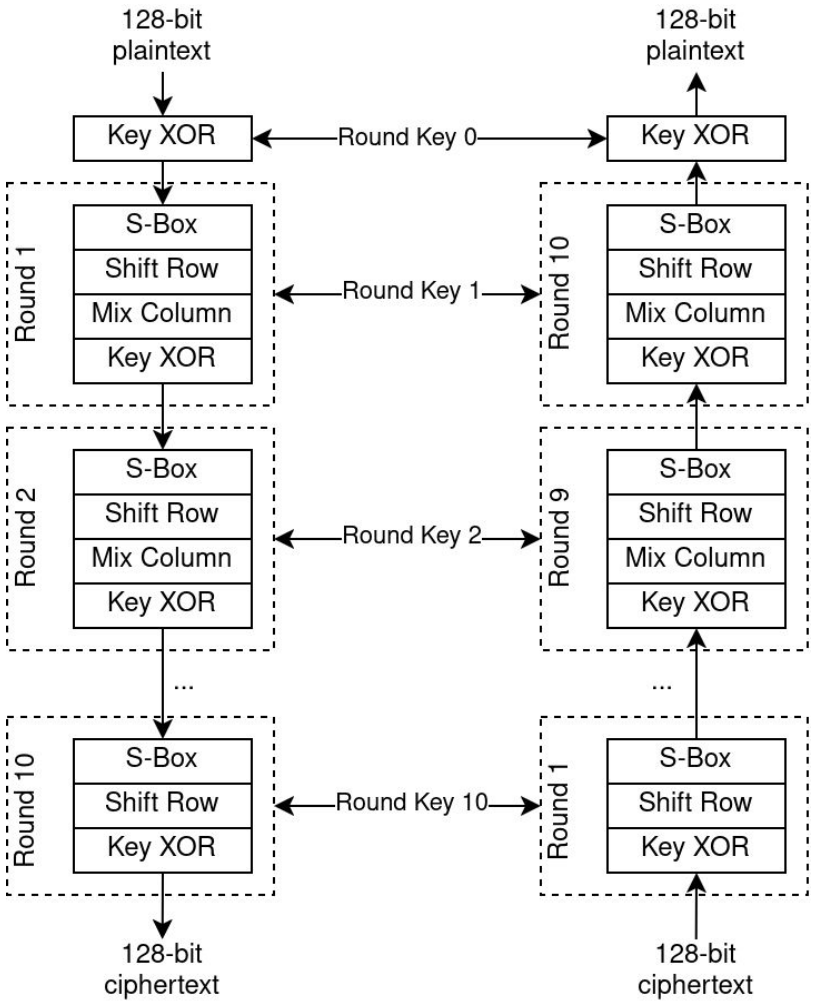
- Round key not in scan chain
- Calculate X from W
- Scan out Z
- Calculate Y from Z

- Solve key XOR

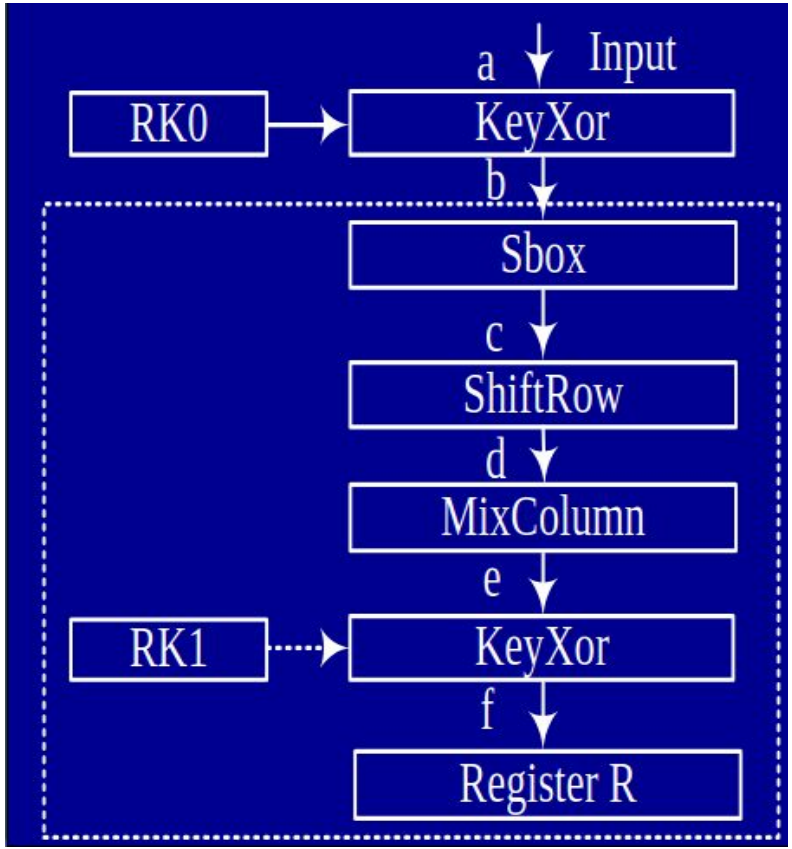


AES Overview

1. Initial round key:
    a. XOR with key
2. 9 rounds:
    a. S-Box (Simple substitution)
    b. Shift Rows (Transposition)
    c. Mix Column (Linear mixing step)
    d. XOR with key
3. Final round
    a. No Mix Column

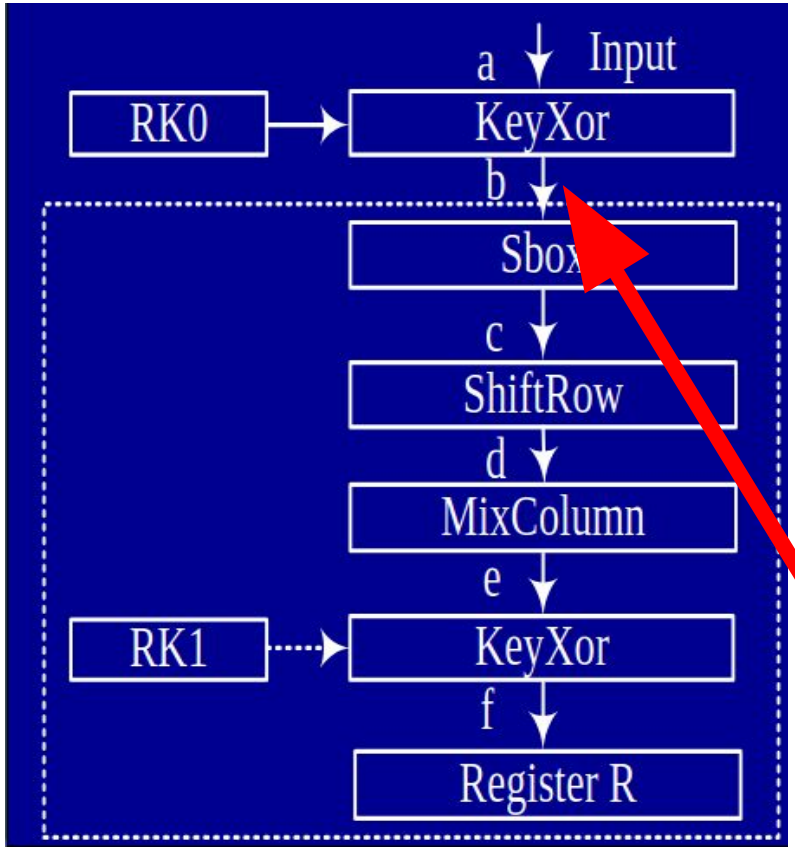


Scan attacks on AES



- Input: 16 bytes
- 128-bit round register R is part of the scan chain
- 128-bit key registers are not in scan chain
- Register R is fed back to B 10 times with RK1 to RK10

Scan attacks on AES



- Input: 16 bytes
- 128-bit round register R is part of the scan chain
- 128-bit key registers are not in scan chain
- Register R is fed back to B 10 times with RK1 to RK10
- Assume register isn't loaded here!!

Scan attack on AES

- Objective: obtain the secret key
- What does the attacker know?
  - Block cipher algorithm
  - Details of the IP core
    - High level timing diagram
    - Number of flip flops in the circuit
    - Possibly even the circuit netlist!

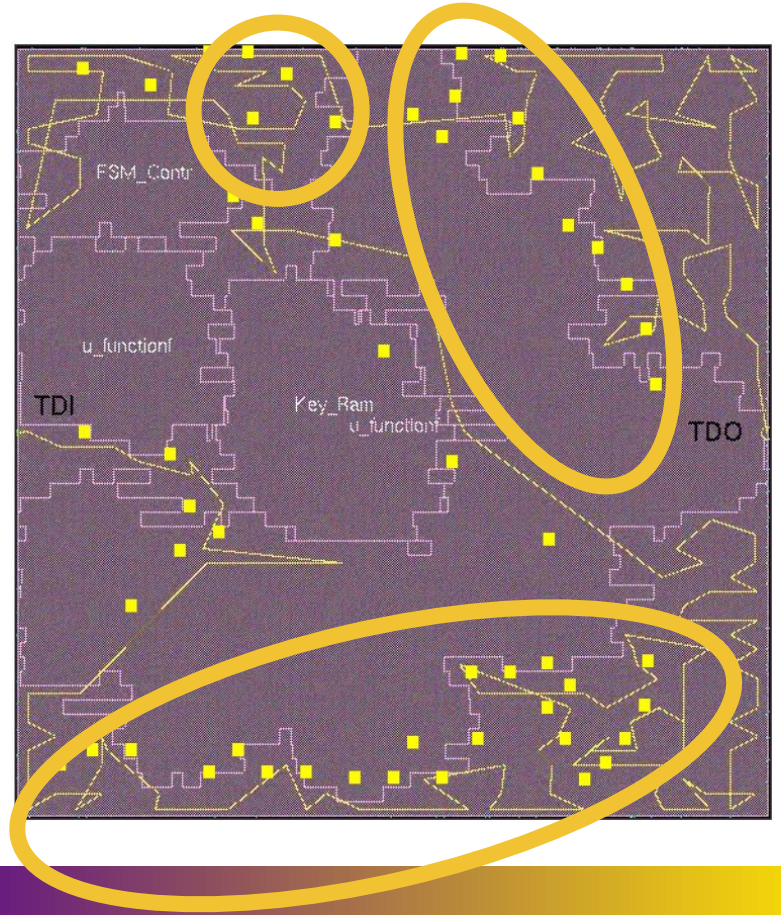
- What does the attacker not know
  - Structure of the scan chain?
  - Secret key itself?

Two-step attack: 1. Identify Scan Chain

Two-step attack: 1. Identify Scan Chain

Determine AES text/ round register R

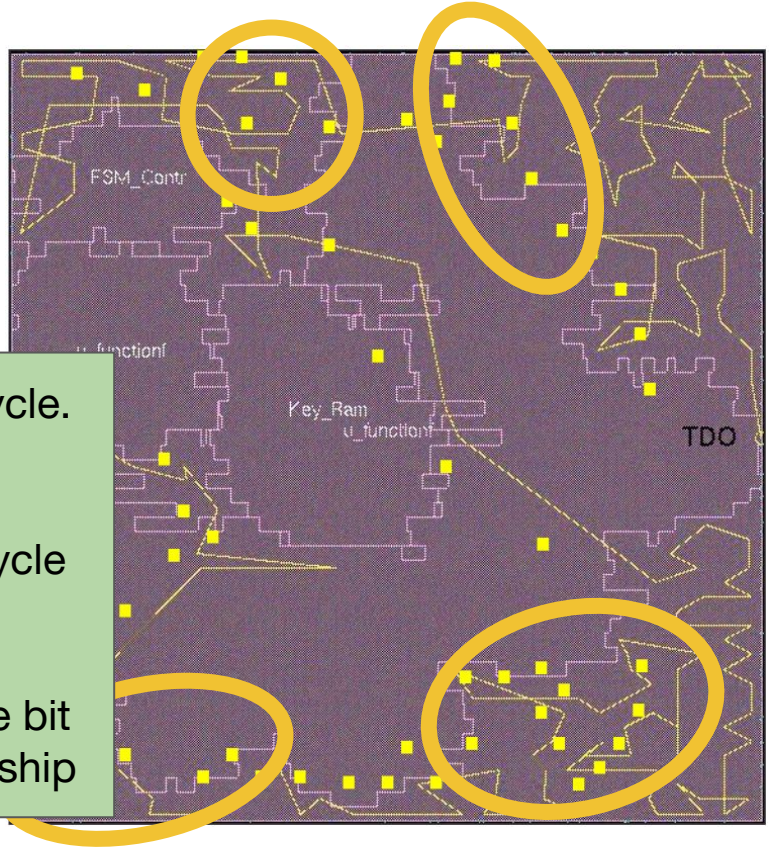
(Mask off bits we don't care about)



Two-step attack: 1. Identify Scan Chain

Determine AES text/ round register R

(Mask off bits we don't care about)



Reset, Apply 0000…00000 and run for one clock cycle.
Scan out the bitstream → 01000…..100001

Reset, Apply 1000…00000 and run for one clock cycle
Scan out the bitstream → 01000…..100101

1 change in bitstream? you have learned about one bit
Many changes? you have learned of a relationship

Determining register bits

Rnd	Plain: 00112233445566778899aabbccddeeff
1	00000000000000000000000000000010c00112233445566778899aabbccddeeff

...

Determining register bits

- Change one bit

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe

Determining register bits

- Change one bit

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe

In-order example, but…
We identified the I/O bit

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeeff
1	00000000000000000000000000000010c00112233445566778899aabbccddeeff
2	073e0f632a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeeff
3	dd435aa222930e50c4b0895c9b7ac50a030e00112233445566778899aabbccddeeff
4	5cf346e64752c7722a67df03b51c081f040e00112233445566778899aabbccddeeff

...

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe
2	aa93e3222a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeefe
3	0f2a33194afbb680a319473b29a6ab64030e00112233445566778899aabbccddeefe
4	64d440b8b2c80b8dfc8f749d9bd47486040e00112233445566778899aabbccddeefe

...

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe
2	aa93e3222a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeefe
3	0f2a33194afbb680a319473b29a6ab64030e00112233445566778899aabbccddeefe
4	64d440b8b2c80b8dfc8f749d9bd47486040e00112233445566778899aabbccddeefe

...

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe
2	aa93e3222a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeefe
3	0f2a33194afbb680a319473b29a6ab64030e00112233445566778899aabbccddeefe
4	64d440b8b2c80b8dfc8f749d9bd47486040e00112233445566778899aabbccddeefe

...

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe
2	aa93e3222a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeefe
3	0f2a33194afbb680a319473b29a6ab64030e00112233445566778899aabbccddeefe
4	64d440b8b2c80b8dfc8f749d9bd47486040e00112233445566778899aabbccddeefe

...

Determining register bits

- Common changes over time?

Rnd	Plain: 00112233445566778899aabbccddeefe
1	00000000000000000000000000000010c00112233445566778899aabbccddeefe
2	aa93e3222a1b9dcd3b3c8f8adabd3a54020e00112233445566778899aabbccddeefe
3	0f2a33194afbb680a319473b29a6ab64030e00112233445566778899aabbccddeefe
4	64d440b8b2c80b8dfc8f749d9bd47486040e00112233445566778899aabbccddeefe

32 bit first-round

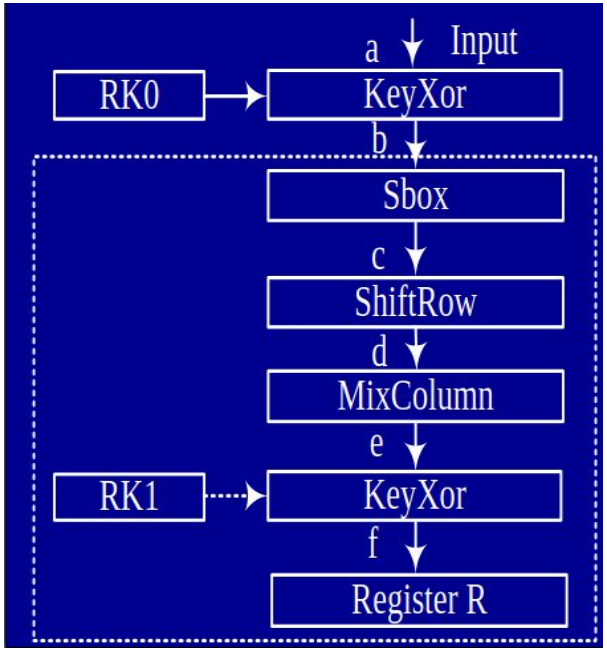
Complete AES text / R

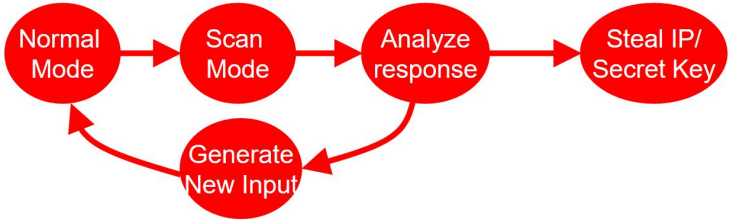
...

control

I/O

Scan attack on AES



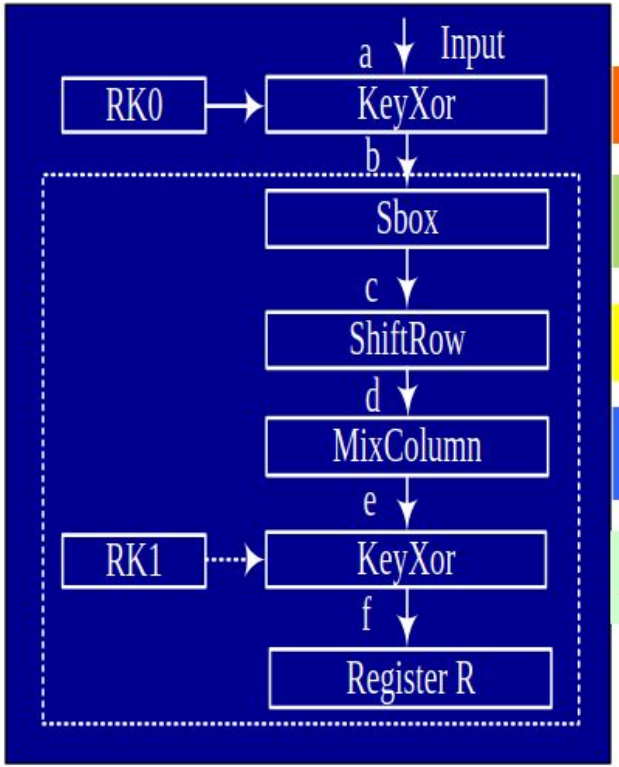


Input: 16 bytes

Scan out: Register R after 5 operations

Goal: Identify RK0

AES: Relationship between input & output



Δ in $a_{11} \Rightarrow \Delta$ in b_{11}

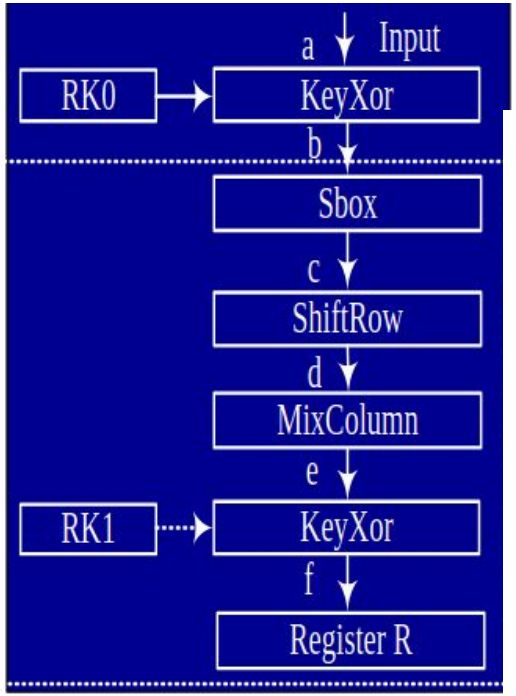
Δ in $b_{11} \Rightarrow \Delta$ in c_{11}

Δ in $c_{11} \Rightarrow \Delta$ in d_{10}

Δ in $d_{10} \Rightarrow \Delta$ in $e_{00}, e_{10}, e_{20}, e_{30}$

Δ in $e_{00}, e_{10}, e_{20}, e_{30} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

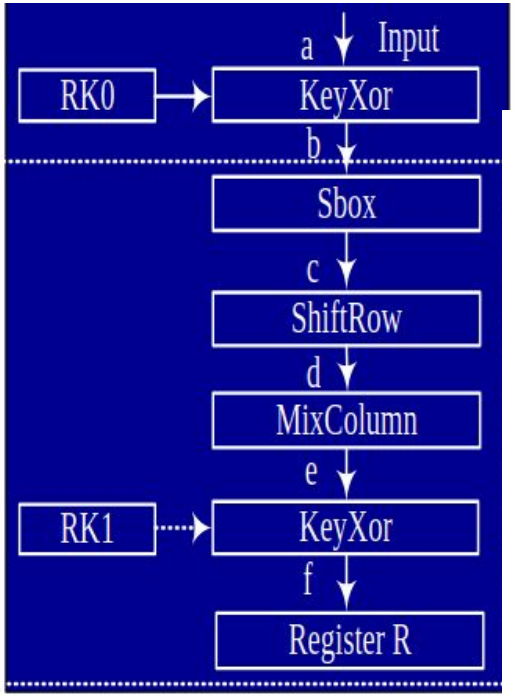
Step 1: Determine scan chain structure



Δ in $a_{11} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

1) Reset chip, provide first plaintext
   a) Run in normal mode for one cycle:
   b) Plaintext will go through RK0 xor and Round 1
2) Switch to test mode, scan out pattern 1
3) Repeat step 1/2 with plaintext different in 1 byte
4) Repeat and collect patterns 3, 4…

Step 1: Determine scan chain structure

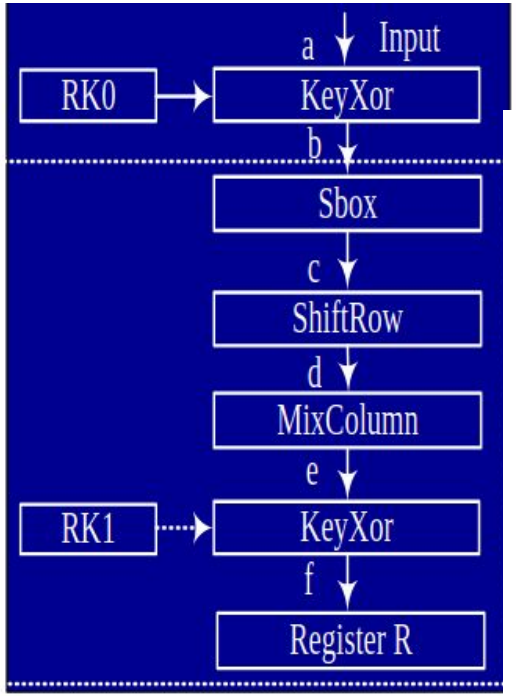


Δ in $a_{11} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

1) Reset chip, provide first plaintext
   a) Run in normal mode for one cycle:
   b) Plaintext will go through RK0 xor and Round 1
2) Switch to test mode, scan out pattern 1
3) Repeat step 1/2 with plaintext different in 1 byte
4) Repeat and collect patterns 3, 4…

32-bits in scanned out bit-stream that correspond to FFs are known - one-to-one mappings not known

Step 1: Determine scan chain structure



Δ in $a_{11} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

1) Reset chip, provide first plaintext
   a) Run in normal mode for one cycle:
   b) Plaintext will go through RK0 xor and Round 1
2) Switch to test mode, scan out pattern 1
3) Repeat step 1/2 with plaintext different in 1 byte
4) Repeat and collect patterns 3, 4…

32-bits in scanned out bit-stream that correspond to FFs are known - one-to-one mappings not known

Summary

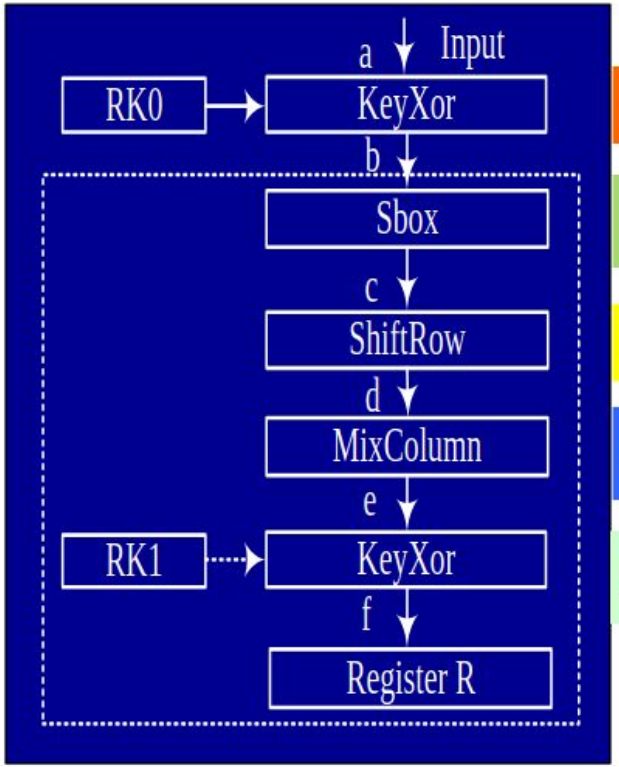
1. Identify I/O register through 1-bit differential loads

2. Identify control registers by running chip, look for consistency

3. Identify AES text / round register by changing 1 bit of input,
   and look for 32 bit change in next cycle

Two-step attack: 2. Extract Round Key

AES: Relationship between input & output



Δ in $a_{11} \Rightarrow \Delta$ in b_{11}

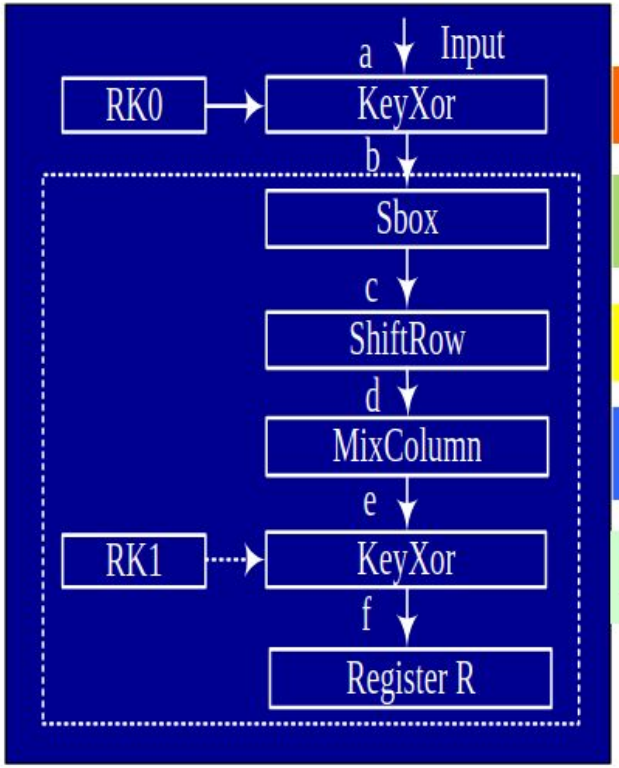
Δ in $b_{11} \Rightarrow \Delta$ in c_{11}

Δ in $c_{11} \Rightarrow \Delta$ in d_{10}

Δ in $d_{10} \Rightarrow \Delta$ in $e_{00}, e_{10}, e_{20}, e_{30}$

Δ in $e_{00}, e_{10}, e_{20}, e_{30} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

AES: Relationship between input & output



Δ in $a_{11} \Rightarrow \Delta$ in b_{11}

Δ in $b_{11} \Rightarrow \Delta$ in c_{11}

Δ in $c_{11} \Rightarrow \Delta$ in d_{10}

Δ in $d_{10} \Rightarrow \Delta$ in $e_{00}, e_{10}, e_{20}, e_{30}$

Δ in $e_{00}, e_{10}, e_{20}, e_{30} \Rightarrow \Delta$ in $f_{00}, f_{10}, f_{20}, f_{30}$

“F” depends on RK0, RK1

Step 2: Recover round key RK0

- “f” depends on RK0 and RK1;

$$f_{i,0}^{1} = e_{i,0}^{1} \oplus RK1_{i,0}, f_{i,0}^{2} = e_{i,0}^{2} \oplus RK1_{i,0}, (0 \le i \le 3) \qquad (6)$$

Step 2: Recover round key RK0

- “f” depends on RK0 and RK1;
- we can rearrange the equations:

$$f^1_{i,0} = e^1_{i,0} \oplus RK1_{i,0}, f^2_{i,0} = e^2_{i,0} \oplus RK1_{i,0}, (0 \le i \le 3) \quad (6)$$

Exclusive or-ing the two terms in (6), we have:

$$\begin{aligned} f^1 \oplus f^2 &= f^1_{i,0} \oplus f^2_{i,0} = (e^1_{i,0} \oplus RK1_{i,0}) \oplus (e^2_{i,0} \oplus RK1_{i,0}) \\ &= (e^1_{i,0} \oplus e^2_{i,0}) \oplus (RK1_{i,0} \oplus RK1_{i,0}) = (e^1_{i,0} \oplus e^2_{i,0}) \quad (7) \\ &= e^1 \oplus e^2 \end{aligned}$$

Step 2: Recover round key RK0

- "f" depends on RK0 and RK1;
- we can rearrange the equations:

$$f_{i,0}^1 = e_{i,0}^1 \oplus RK1_{i,0}, f_{i,0}^2 = e_{i,0}^2 \oplus RK1_{i,0}, (0 \le i \le 3) \qquad (6)$$

Exclusive or-ing the two terms in (6), we have:

$$f^1 \oplus f^2 = f_{i,0}^1 \oplus f_{i,0}^2 = (e_{i,0}^1 \oplus RK1_{i,0}) \oplus (e_{i,0}^2 \oplus RI$$
$$= (e_{i,0}^1 \oplus e_{i,0}^2) \oplus (RK1_{i,0} \oplus RK1_{i,0}) = (e_{i,0}^1 \oplus e_{i,0}^2)$$
$$= e^1 \oplus e^2$$

Observation 1:

Number of "1"s in $f^1 \oplus f^2$

Is the same as in $e^1 \oplus e^2$

Step 2: Recover round key RK0

- "f" depends on RK0 and RK1;
- we can rearrange the equations:

$$f_{i,0}^1 = e_{i,0}^1 \oplus RK1_{i,0}, f_{i,0}^2 = e_{i,0}^2 \oplus RK1_{i,0}, (0 \le i \le 3) \qquad (6)$$

Exclusive or-ing the two terms in (6), we have:

$$f^1 \oplus f^2 = f_{i,0}^1 \oplus f_{i,0}^2 = (e_{i,0}^1 \oplus RK1_{i,0}) \oplus (e_{i,0}^2 \oplus RK1_{i,0})$$
$$= (e_{i,0}^1 \oplus e_{i,0}^2) \oplus (RK1_{i,0} \oplus RK1_{i,0}) = (e_{i,0}^1 \oplus e_{i,0}^2)$$
$$= e^1 \oplus e^2$$

Observation 2:

Number of "1"s in $f^1 \oplus f^2$

Is independent of

RK1

Step 2: Recover round key RK0

- Discovery *(Yang et al., 2005)*:
- When $(a^1_{1,1}, a^2_{1,1})$ are such that:
  - $a^1_{1,1} = 2m, \quad a^2_{1,1} = 2m+1, \quad \text{and } a^1_{1,1} \oplus a^2_{1,1} = 1$
- Then, if:

# of 1s in $f^1_{i0} \oplus f^2_{i0}$	9	12	23	24
(b^1_{11}, b^2_{11}) pairs	226,227	242,243	122,123	130,131

- And:
  - $RK0_{1,1} = a_{1,1} \oplus b_{1,1}$

I.e. Key byte is XOR one of these options

Step 2: Recover round key RK0

- Discovery *(Yang et al., 2005)*:
- When ($a^1_{1,1}$, $a^2_{1,1}$) are such that:
  - $a^1_{1,1} = 2m, \quad a^2_{1,1} = 2m+1, \quad$ and $a^1_{1,1} \oplus a^2_{1,1} = 1$
- Then, if:

# of 1s in $f^1_{i0} \oplus f^2_{i0}$	9	12	23	24
(b^1_{11}, b^2_{11}) pairs	226,227	242,243	122,123	130,131

- And:
  - $RK0_{1,1} = a_{1,1} \oplus b_{1,1}$

"Brute force":
~32 plaintexts to retrieve options for one key byte

Step 2: Recover round key RK0

- “Brute force”:
- ~32 plaintexts to retrieve one key byte

Step 2: Recover round key RK0

1. Apply a value “2t” ($0 \leq t \leq 127$) to a byte of input $a^1_{1,1}$
2. Scan out AES text / round register f^1 after 1 clock tick
3. Apply a value “2t+1” ($0 \leq t \leq 127$) to a byte of input $a^2_{1,1}$
4. Scan out AES text / round register f^2 after 1 clock tick
5. If number of “1s” in $f^1 \oplus f^2$ is 9, 12, 23, or 24, obtain $(b^1_{1,1}, b^2_{1,1})$
   a. Otherwise go back to “1” with different t
6. Determine key byte RK0 as $b^1_{1,1} \oplus a^1_{1,1}$, $b^2_{1,1} \oplus a^1_{1,1}$
   a. This creates two possibilities for the key - we’ll brute force it later
7. If all bytes in RK are “pairwise” determined, stop.
   a. Otherwise, repeat from step 1 for different byte $a^1_{i,j}$

Step 2: Recover round key RK0

- Given all the “pairwise” options - 2 for each byte (i.e. 2^16)...
- Brute force the key!
- 2^16 is a lot smaller than 2^128 !!!

Step 2: Recover round key RK0

- “Brute force”:
- ~32 plaintexts to retrieve one key byte

Read more: B. Yang, K. Wu, R. Karri "Secure scan: a design-for-test architecture for crypto chips." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(10): 2287-2293.

Scan attacks: Summary

- DFT infrastructure provides an avenue for attackers….

Test Engg	Applies test patterns	Analyzes responses (to detect bugs)
Hacker	Applies attack patterns	Analyzes responses (to uncover secret)

- Allow access of internal system state and proprietary info!
- Scan attacks reduce crypto implementations to effectively single-round block ciphers!

Scan attacks on various algorithms:

- Attacks on public key crypto:
  - R. Nara, K. Satoh, et al. (2010). "Scan-Based Side-Channel Attack against RSA Cryptosystems Using Scan Signatures." IEICE- Transactions on Fundamentals of Electronics, Communications and Computer Sciences(12).
  - R. Nara, N. Togawa, et al. (2010). Scan-based attack against elliptic curve cryptosystems. Asia and South Pacific Design Automation Conference.
- Complex SoCs, scan with compaction:
  - C. Liu, and Y. Huang, “Effects of Embedded Decompression and Compaction Architectures on Side-Channel Attack Resistance,” VLSI Test Symposium, 2007.
  - J. Rolt, A Das, G Di Natale, M Flottes, B Rouzeyre, I Verbauwhede, "A New Scan Attack on RSA in Presence of Industrial Countermeasures," 3rd International workshop on Constructive Side-Channel analysis and secure design, 2012
  - G. Darolt, G Di Natale, M Flottes, B Rouzeyre, “Are advanced DFT structures sufficient for preventing scan-attacks?”, IEEE VLSI Test Symposium, 2012.

Break: 5 minutes

Break activity: Talk to your neighbour -

hi
how cool is scan attack

Scan Attack Countermeasures

Countermeasure 1: Unbound scan chains

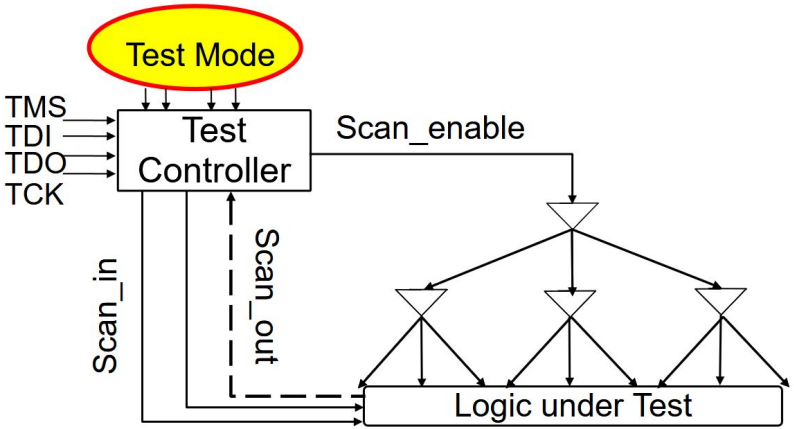
Leave scan chains unbound [Kömm99]!

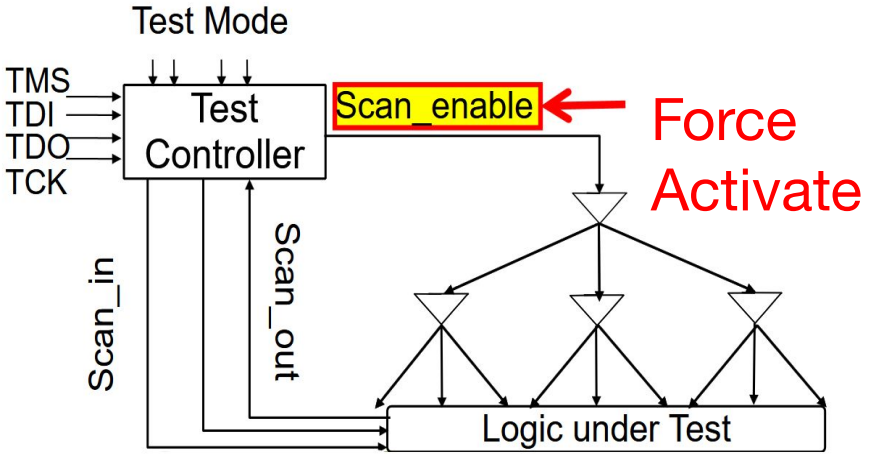
- This does compromise maintenance and in-field debug

O. Kömmerling, M. G. Kuhn, Design Principles for Tamper-Resistant Smartcard Processors, USENIX Workshop on Smartcard Technology, pp.9-20, May, 1999.

Countermeasure 2: Test locks

- Test locking/unlocking
  - Can possibly be brute forced
  - Can possibly be bypassed





Countermeasure 3: “Secure Scan”

- Crypto algorithms are key based
  - Algorithms are public but keys are secret
  - We want to keep the keys secret
  - Can we make the internals less observable/controllable…
    - …without compromising the purpose of DFT?

B. Yang, K. Wu and R. Karri ”Secure scan: a design-for-test architecture for crypto chips,” IEEE/ACM Symposium on Design Automation Conference, 2004
B. Yang, K. Wu and R. Karri ”Secure scan: a design-for-test architecture for crypto chips,” IEEE Transactions on Computer-Aided Design 2006, 25(10): 2287-2293.

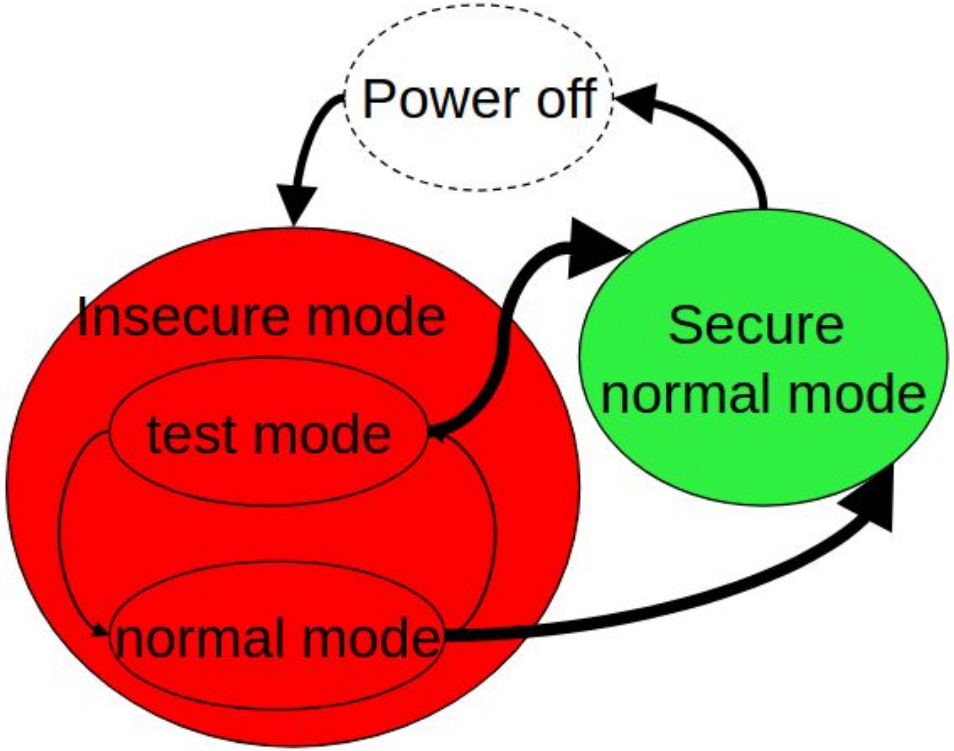
“Secure Scan”

- Information retrieved from scan chains should not be useful for retrieving the secret key

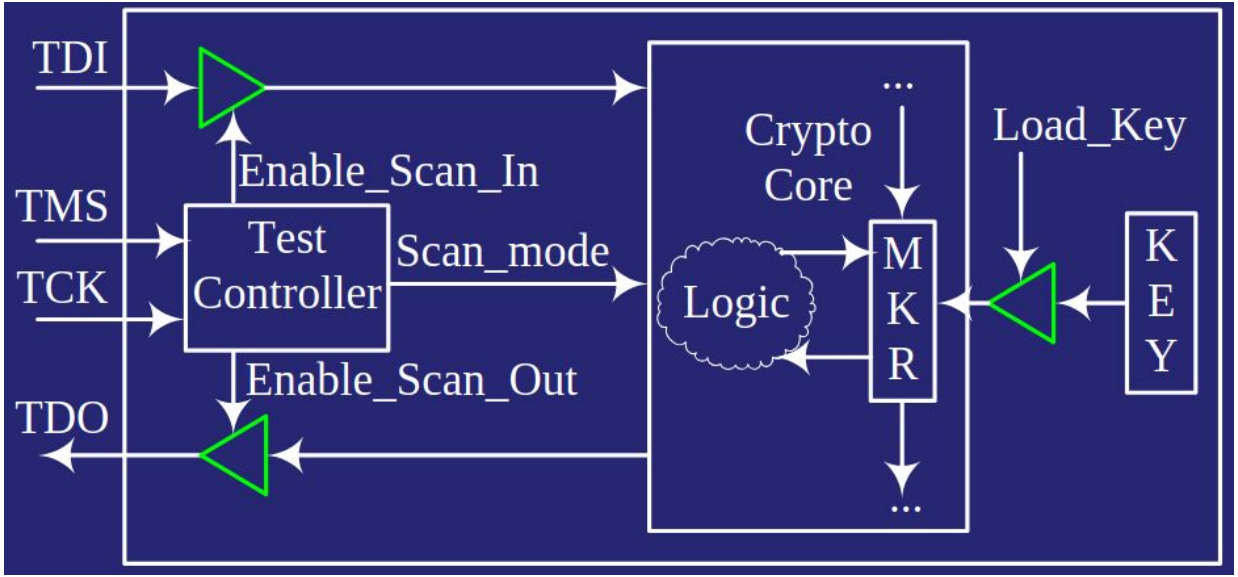
Insight:

- Have two keys
  - Secret key: used in normal mode (as per existing hardware)
  - *Mirror key*: a special key which is just used for testing
    - Stored in mirror key register, MKR
  - When switching from secure/normal mode to test mode:
    - Isolate secret key
    - Load mirror key
    - Enable test/debug

“Secure Scan”: State diagram



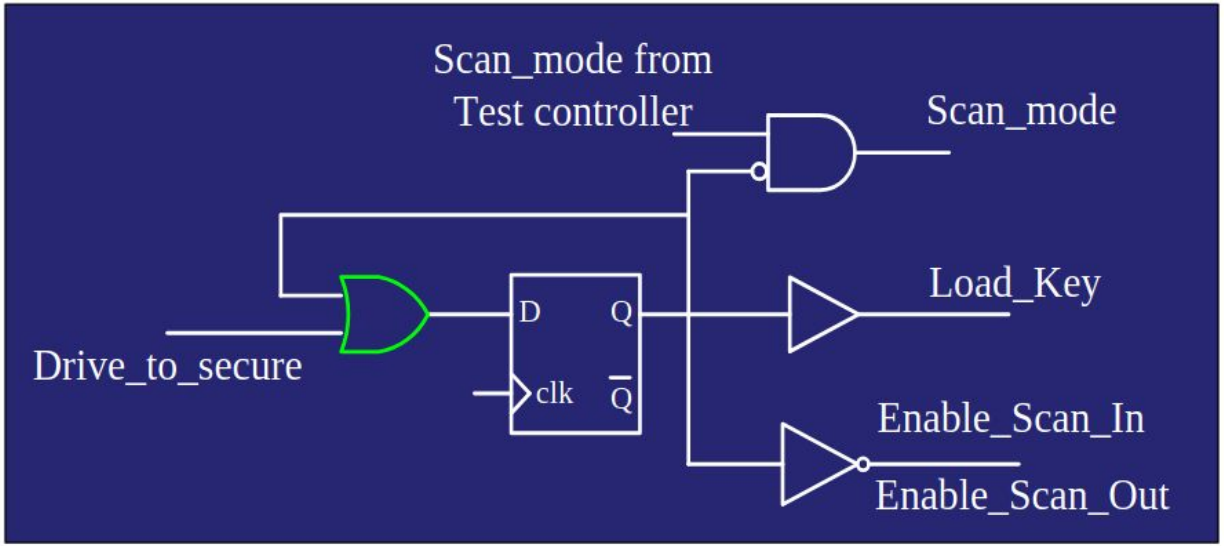
Secure scan architecture



Insecure mode: Enable_Scan_In=1, Enable_Scan_Out=1, Load_Key=0

Secure mode: Enable_Scan_In=0, Enable_Scan_Out=0, Load_Key=1

Secure scan: Test controller

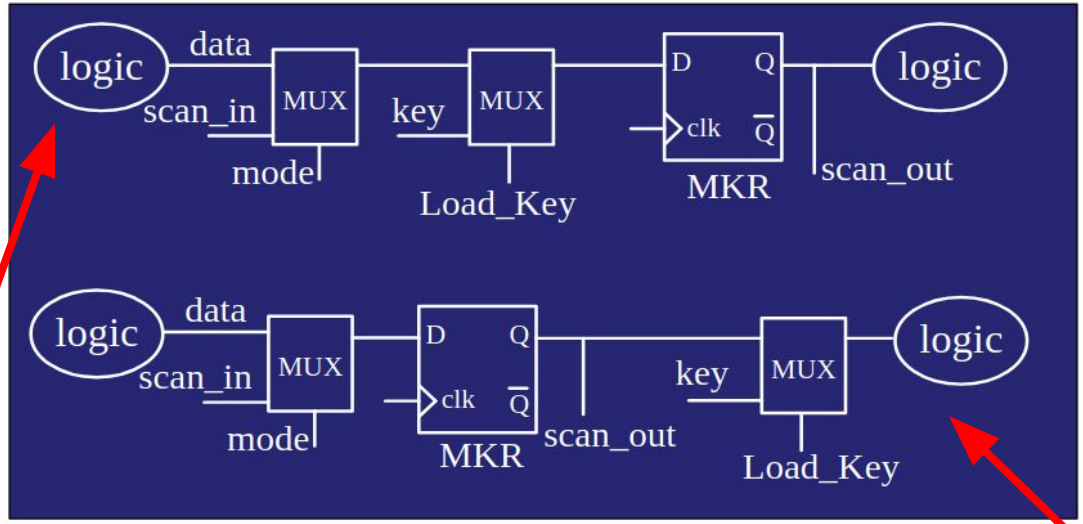


Modify IEEE 1149.1 Test Controller

New instruction: Drive_to_secure

Three new output control signals, dedicated security circuit

Mirror key register (MKR)

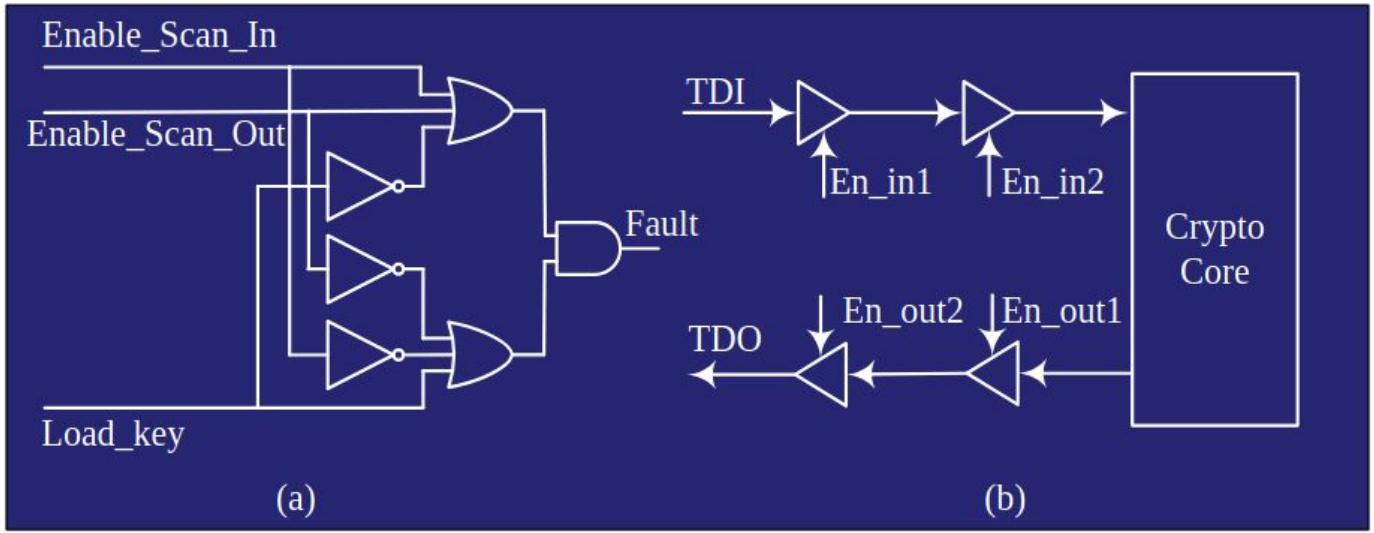


(a) Multiplexer before MKR

(b) Multiplexer after MKR (it does not modify scan cell)

Prevent physical attacks on scan signals

- Three control signals, any fault will compromise security
  - Concurrent faults checker
  - Multi-stage buffer between TDI, TDO, and crypto core



Secure scan: overheads

Architecture	Area (gates)	Area overhead (gates)	Ratio
Iterative (with KS)	31,234	412	1.32%
Iterative (without KS)	30,854	412	1.34%
Pipelined (with KS)	273,187	412	0.15%
Pipelined (without KS)	282,120	4620	1.64%

Secure scan: Perfect solution?

- No

But:

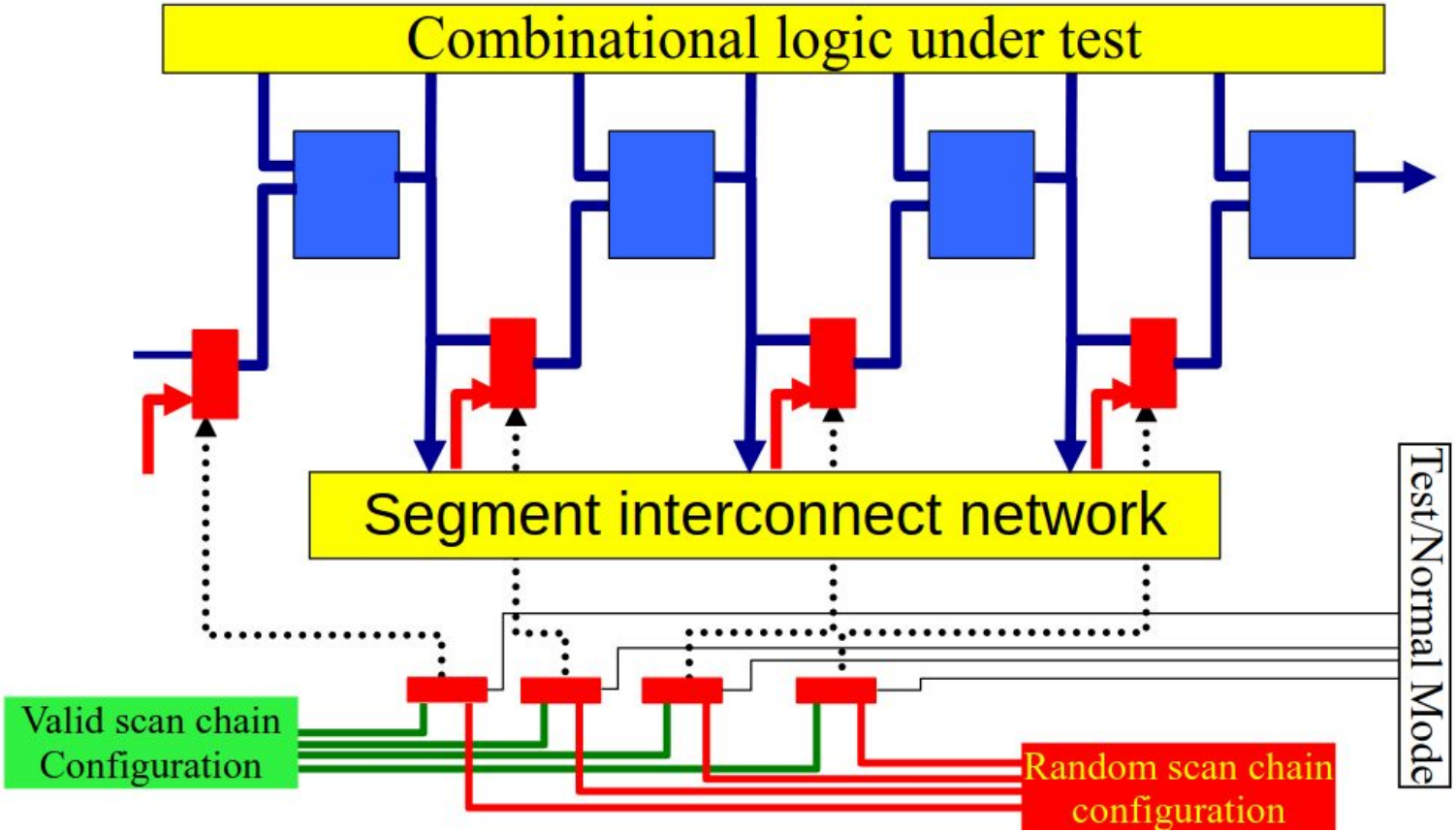
- Time needed to undermine defense: High
- Expertise necessary: High
- Knowledge of the design: Full
- Equipment to attack: advanced

Countermeasure 4: scramble scan chain

- Scan chain requires configuration for use
- Segment ordering is unpredictable and randomly changes
- In test mode, scan chain is configured to correct ordering

D. Hély, F. Bancel, ML Flottes, B. Rouzeyre, M. Renovell and N. Bérard, Scan Design and Secure Chip, IEEE International On-Line Testing Symposium 2004
D. Hély, F Bancel, M. Flottes, B Rouzeyre, IOLTS 04, ETS 2005
D. Hély, F. Bancel, et al.. A secure scan design methodology. DATE, 2006
D. Hély, F. Bancel, et al. "Securing scan control in crypto chips." JETTA 23(5): 457-464, 2007
F. Bancel and D. Hély. Protecting an integrated circuit test mode, US Patent, 2010

Scan chain scrambling hardware



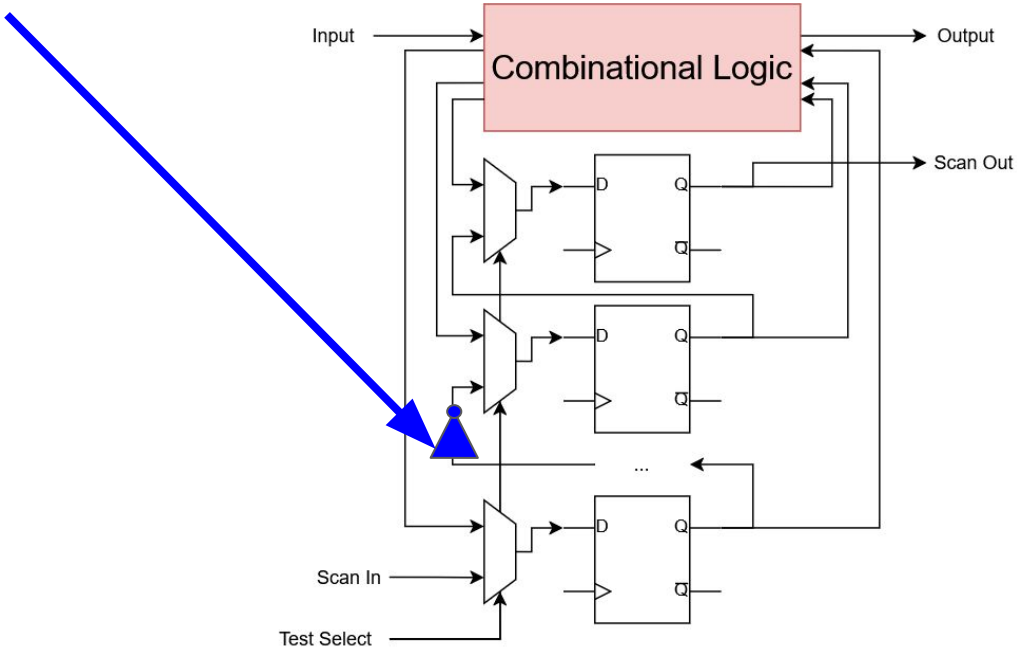
Scan chain scrambling: cost

Design costs: for a 3K flip-flop architecture

- Extra area for FSM modifications
- Extra security involved features
- Reset verification (15% of FFs checked)
- Scan enable integrity checker
- area overhead: 12%

Countermeasure 5: Flipped scan

- Insert inverters on a random subset of scan inputs



G. Sengar, D. Mukhopadhyay and D. R. Chowdhury, “Secured Flipped Scan-Chain Model for Crypto-Architecture," IEEE Transactions on Computer Aided Design, 2007, pp: 200-2084

Flipped Scan

- More difficult to understand randomized scan chain layouts
  - Hides simple relationships
- Very low overhead
- Defense is constant: can eventually be brute forced

Scan DFT: summary

- Scan DFT can be made more secure with modifications
  - To the test controller
  - To the DFT implementation

- Tradeoffs:
  - Security vs power, area overhead, testability

Assessments: Week 3

Lab 1: Scan Attack on AES

- Every student has unique AES core
- Same as examples/spi_aes_scan except:
  - Randomized key
  - Randomized scan chain order
- Each student must perform the AES attack to extract their key!
  - Hand in:
    - Scan chain layout
    - Extracted round key
    - Derived key

Instructions on Course Website!

Lab 1: Due Friday Week 4

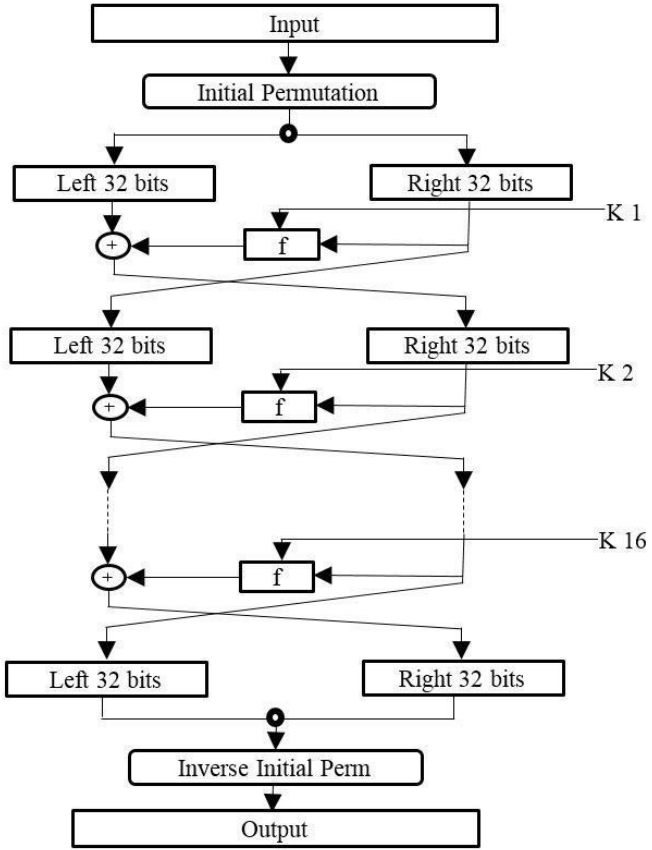
That’s it! Questions?

Appendix: DES Scan Attack

DES Encryption

1. Permute input (IP)
2. Split into L/R
3. New L = old R
4. New R = old L XOR F(old R, RK_n)
5. In 16th round, take pre-out and
   Permute through IP^{-1}

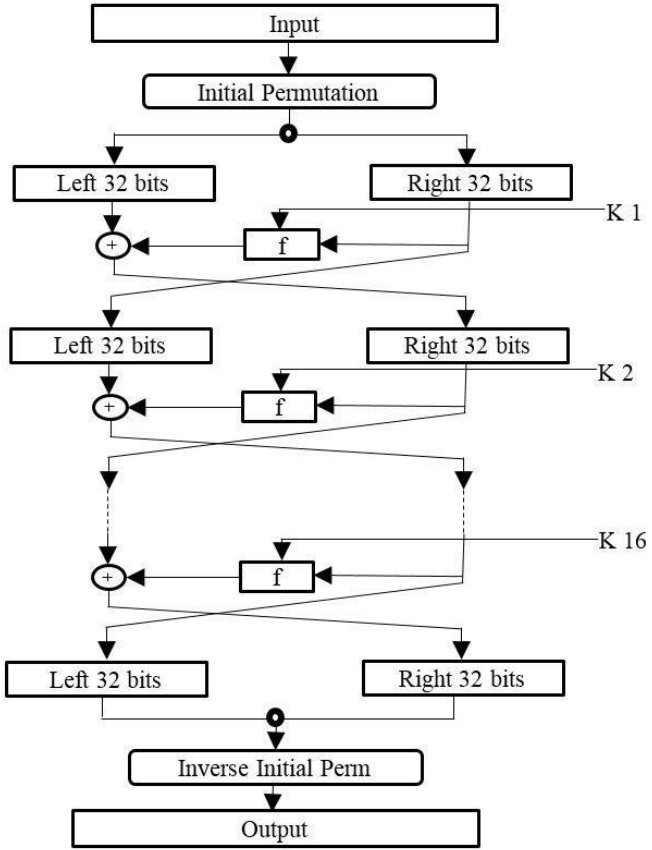
6. Round keys for encryption: K1-K16
7. Round keys for decryption: K16-K1



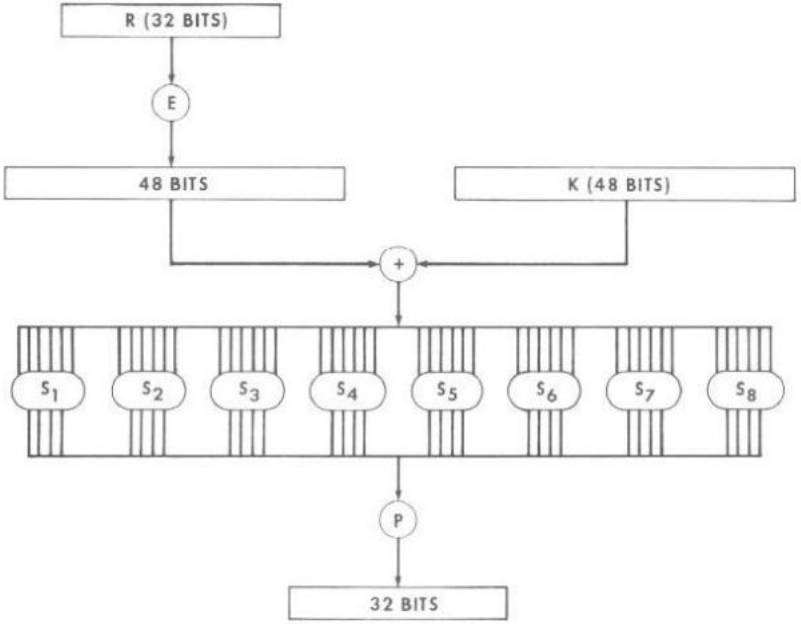
DES Encryption

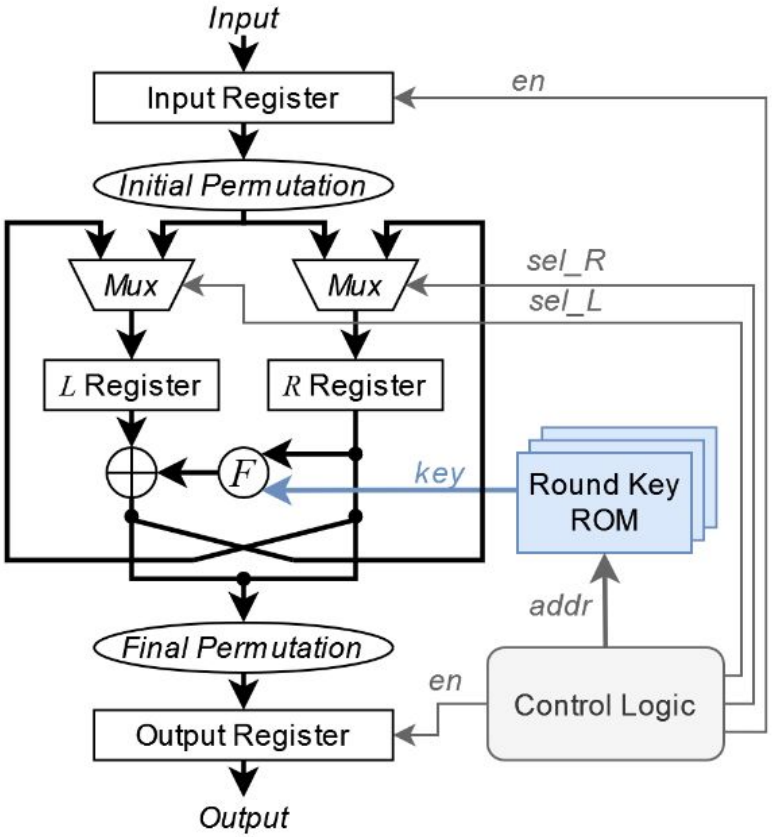
- Decryption = inverse Encryption
- ENCRYPT (plain, key) = cipher
- DECRYPT (cipher, key) = plain

- 64-bit plaintext, 64-bit ciphertext
- 56-bit key
- 16 identical rounds
- 56-bit secret → 16x 48-bit round keys



DES Architecture





Input, L, R, Output Registers

Scan attack on DES

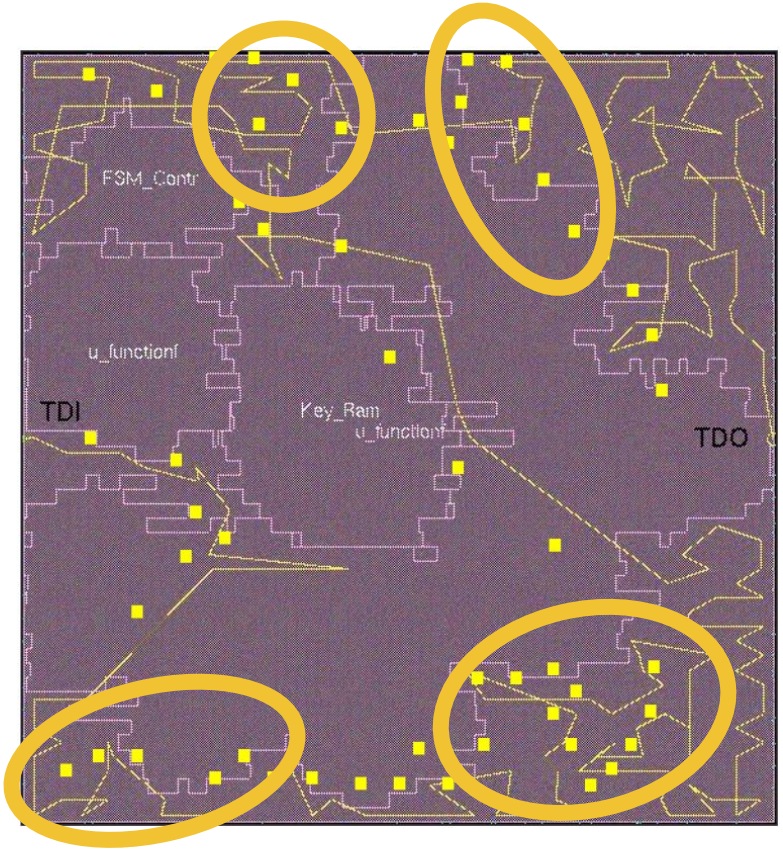
- Objective: obtain the secret key
- What does the attacker know?
  - Block cipher algorithm
  - Details of the IP core
    - High level timing diagram
    - Number of flip flops in the circuit
    - Possibly even the circuit netlist!

- What does the attacker not know
  - Structure of the scan chain?
  - Secret key itself?

Two-step attack: 1. Identify Scan Chain

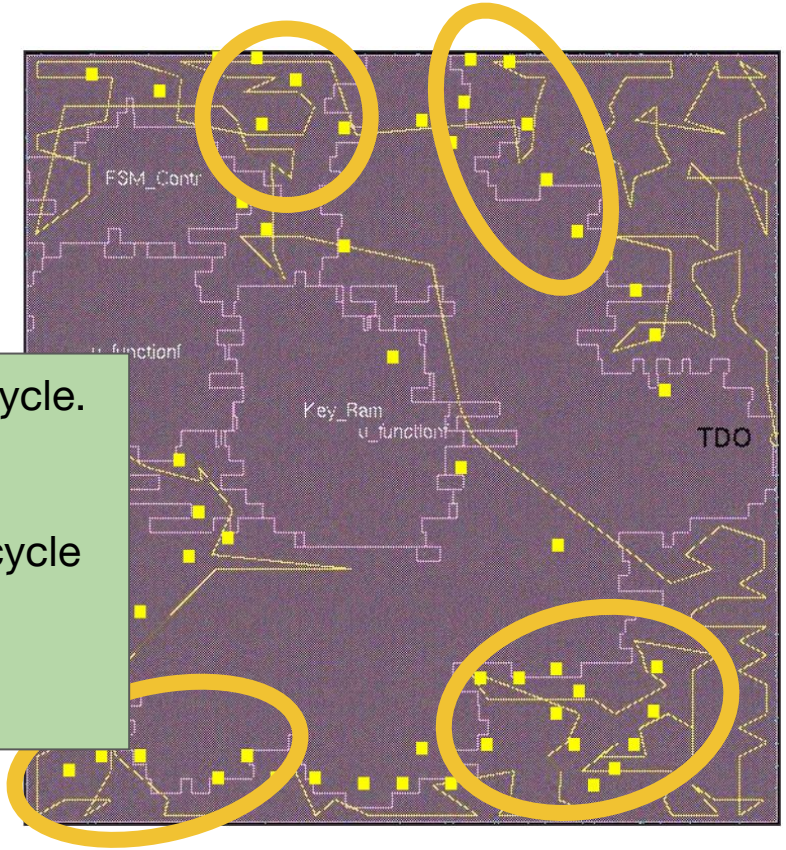
Two-step attack: 1. Identify Scan Chain

Determine Input, L, R, and output registers:



Two-step attack: 1. Identify Scan Chain

Determine Input, L, R, and output registers:



Reset, Apply 0000…00000 and run for one clock cycle.
Scan out the bitstream → 01101…..100101

Reset, Apply 1000…00000 and run for one clock cycle
Scan out the bitstream → 01101…..110101

Change in bitstream: you have located one bit

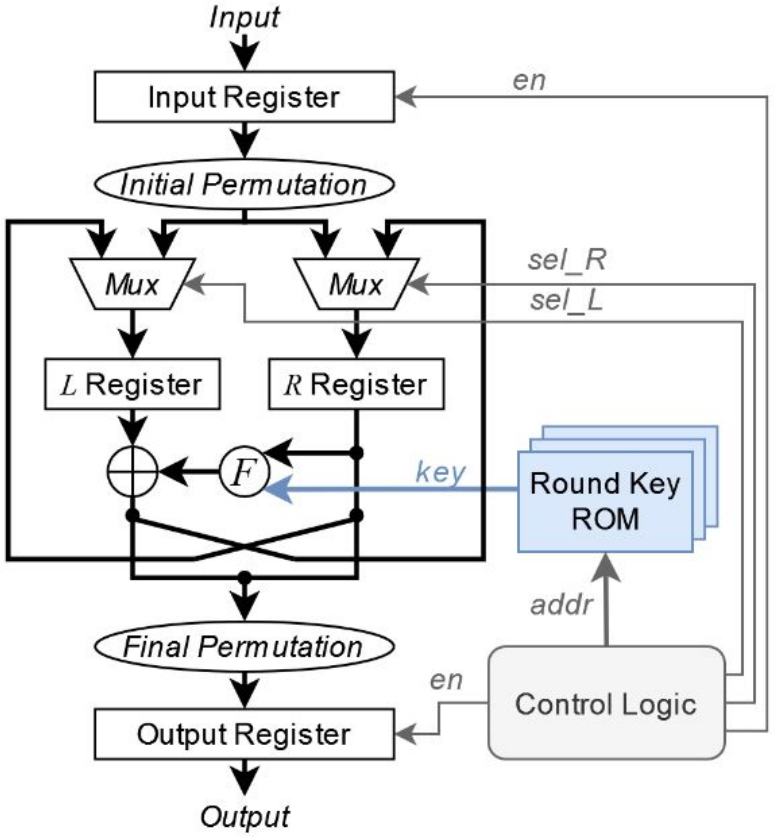
Determining register bits

- Clock cycle 1: Load input register
- Clock cycle 2: Load L/R after IP

Load 0x8000000000000000

Cycle 1: 1 bit of Input Register

Cycle 2: 1 bit of L/R after IP



Determining register bits

- What else is changing?

Rnd	Cipher: 00 00 00 00 00 00 00 00
1	0000000000000000000000000000001f0000000000000000
2	00000000000000000000000000000060000000000000000
3	0000000000000000f6c6bb2400000000610000000000000000
4	00000000000000000949f9327f6c6bb2462000000000000000

Determining register bits

- What else is changing?

Rnd	Cipher: 80 00 00 00 00 00 00 00
1	00000000000000000000000000000001f8000000000000000
2	00000000000000001000000000000060800000000000000
3	00000000000000000f2cffb3001000000618000000000000000
4	00000000000000076dad2f2f2cffb3062800000000000000000

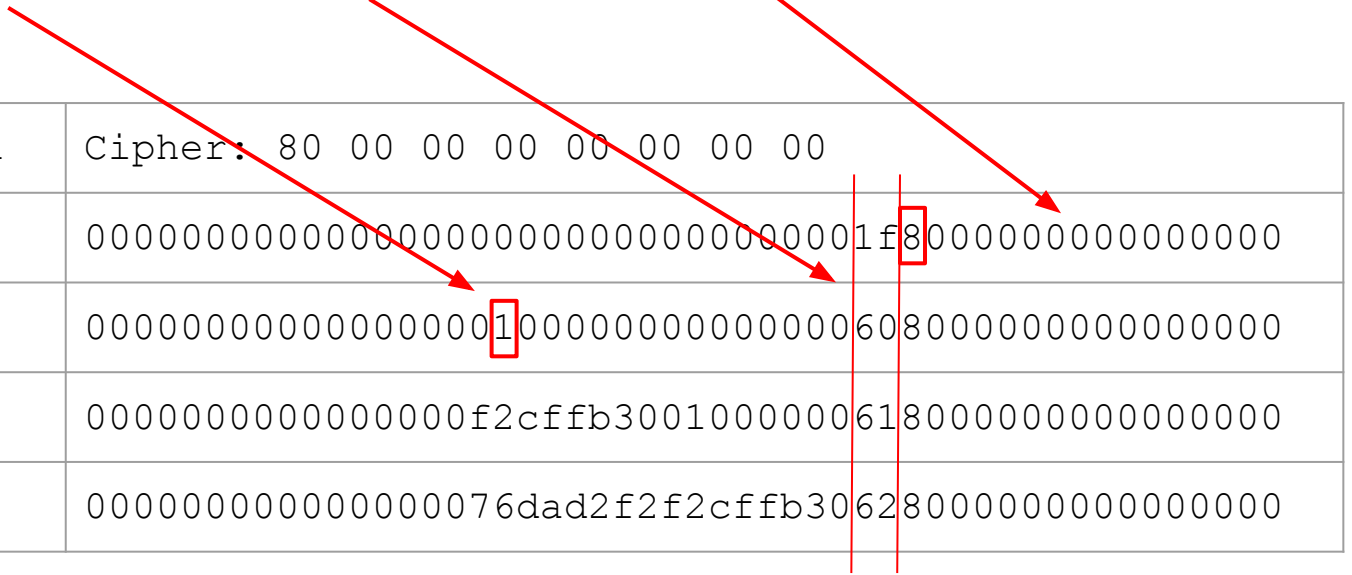
Determining register bits

- What else is changing?

Rnd	Cipher: 80 00 00 00 00 00 00 00
1	00000000000000000000000000000001f8000000000000000
2	00000000000000001000000000000060800000000000000
3	00000000000000f2cffb3001000000618000000000000000
4	00000000000000076dad2f2f2cffb3062800000000000000

Determining register bits

- L/R bit; Control bits, Input bits



Rnd	Cipher: 80 00 00 00 00 00 00 00
1	00000000000000000000000000001f8000000000000000
2	00000000000000001000000000000608000000000000000
3	00000000000000f2cffb3001000000618000000000000000
4	0000000000000076dad2f2f2cffb3062800000000000000

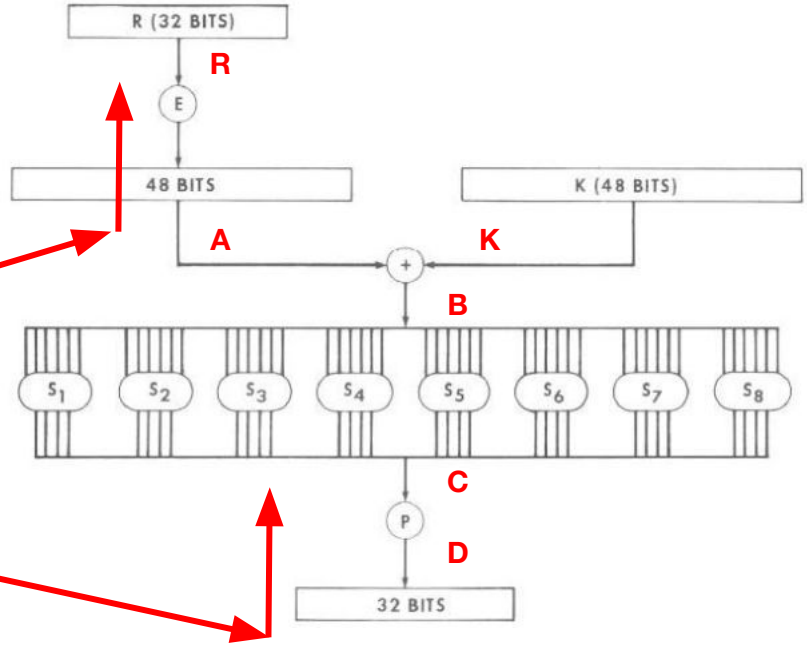
Two-step attack: 2. Extract Round Key

Solving for RK in DES?

- We are interested in K
- B = A xor K

Solving unknowns:

- Expansion is a bijection
  - R→A is easy
- Permutation is a bijection
  - D→C is easy

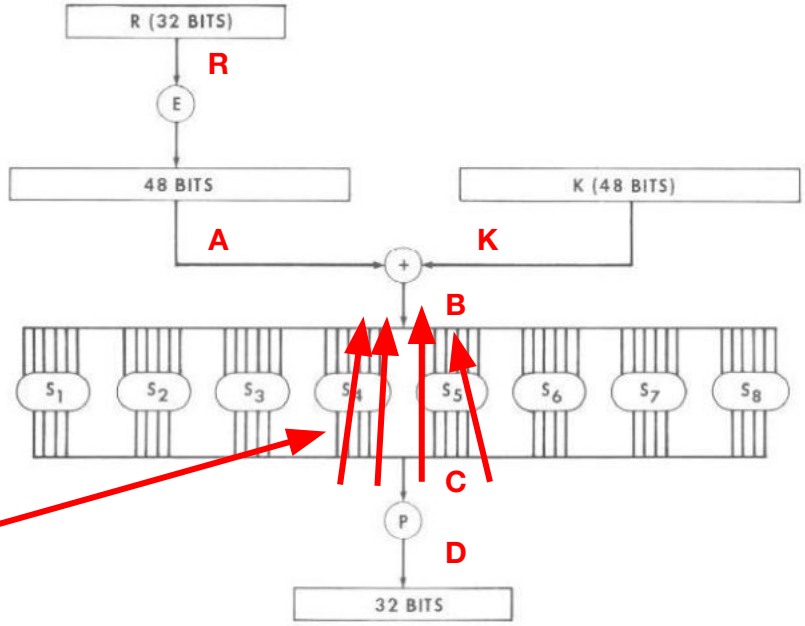


How does this look for DES?

- We are interested in K
- B = A xor K

Solving unknowns:

- Expansion is a bijection
  - R→A is easy
- Permutation is a bijection
  - D→C is easy
- S-box not a bijection
  - C→B is slightly hard!



Undoing the S-boxes

- S-boxes are look up tables where each output value appears 4x
  - E.g.

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- Given output “C”, we have 4 possible inputs for input “B”!

Undoing the S-boxes

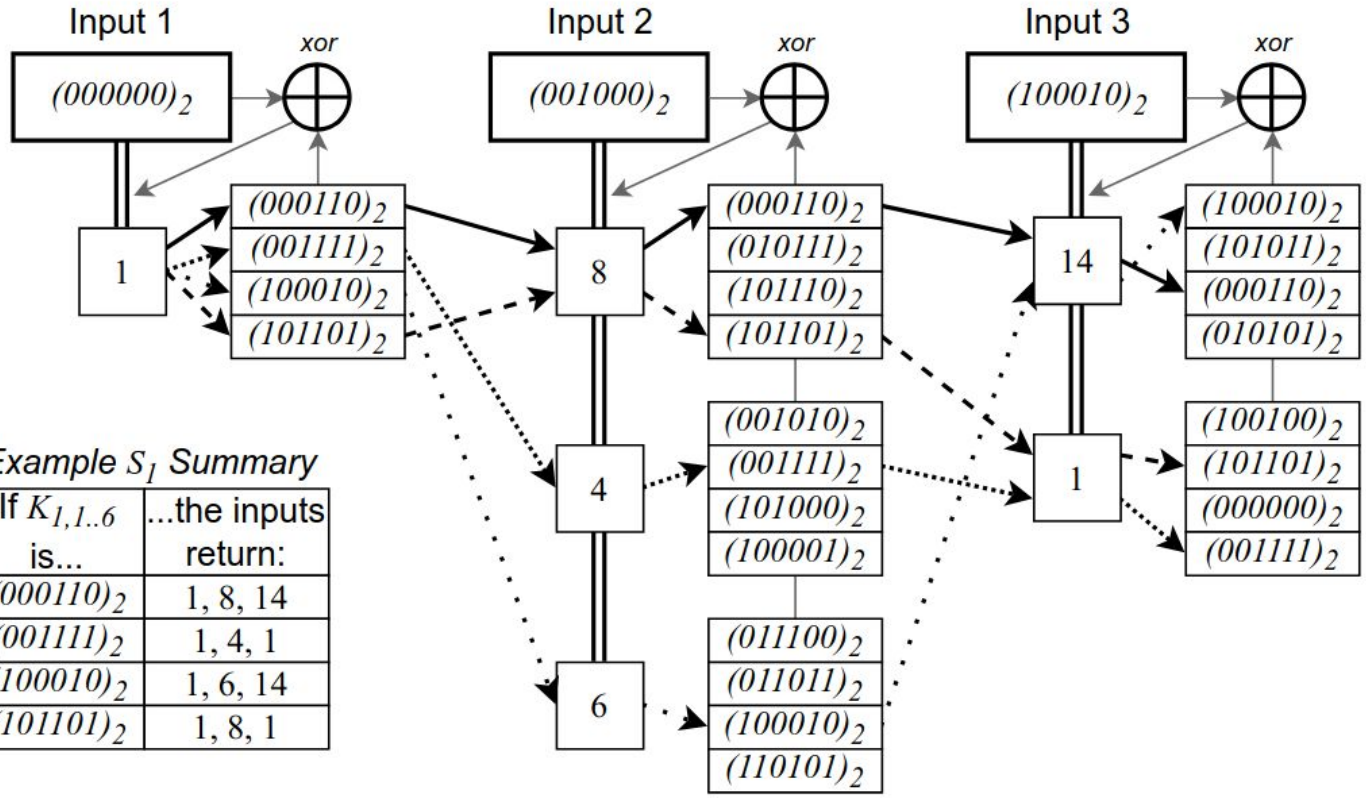
Idea:

- We can control point A
- We can compute point C
- Point K is constant!

Insight:

- We can give multiple different values to point A
- Compute SBox(A xor K) for different values - how many?
  - 3 “well-chosen” inputs will uniquely choose the key

Example for S-box 1



Example S_1 Summary

If $K_{1,1..6}$ is...	...the inputs return:
$(000110)_2$	1, 8, 14
$(001111)_2$	1, 4, 1
$(100010)_2$	1, 6, 14
$(101101)_2$	1, 8, 1

Three well-crafted inputs

$$i^1_{1..64} = (0000000000000000)_{16}$$
$$i^2_{1..64} = (0000AA000000AA00)_{16}$$
$$i^3_{1..64} = (8220000A8002200A)_{16}$$

- These three inputs → three different outputs
  - Each output could be produced by one of four different keys
  - But: Only one key could produce all three of those outputs
- Now you can uniquely identify the round key!

Uncover DES key

- User secret has 56 bits
- By undoing PC2, first shift, and PC1, we get 48 of the 56 bits

Two options:

- Run encryption for more cycles and repeat process to get remaining 8 bits

Or:

- Brute force the remaining 8 bits (only 256 combinations)
- Then we are done - **we got the key!**

How many clock cycles?

- Scan: 200 clock cycles
- Load input: 1 clock cycle? (+ SPI)
- Load L/R: 1 clock cycle
- First round: 1 clock cycle
- All resets: 1 clock cycle

Identifying location of input, L, R:

1 bit:

= Reset + Load input + Scan + Reset + Load input + Load L/R + Scan

= 1 + 1 + 200 + 1 + 1 + 1 + 200 = 405 per bit (excl. SPI)

How many clock cycles?

- Scan: 200 clock cycles
- Load input: 1 clock cycle? (+ SPI)
- Load L/R: 1 clock cycle
- First round: 1 clock cycle
- All resets: 1 clock cycle

Extracting 3 values to determine RK:

= 3* (Reset + Load input + Load L/R + First Round + Scan)

= 3* (1 + 1 + 1 + 1 + 200) = 612 (excluding SPI)

Read more:

- B. Yang, K. Wu, R. Karri, “Scan Chain Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard”, ITC Oct 2004

- H. Pearce, R. Karri, B. Tan, “High-Level Approaches to Hardware Security: A Tutorial”, ACM TECS 2023 https://doi.org/10.1145/3577200