New Scan-Based Attack Using Only the Test Mode and an Input Corruption Countermeasure

Sk Ali, Samah Saeed, Ozgur Sinanoglu, Ramesh Karri

HAL Id: hal-01380298
https://hal.inria.fr/hal-01380298

Submitted on 12 Oct 2016

New Scan-Based Attack
Using Only the Test Mode And
An Input Corruption Countermeasure

Sk Subidh Ali[1], Samah Mohamed Saeed[2], Ozgur Sinanoglu[1], and Ramesh Karri[2]

[1] New York University Abu Dhabi
{sa112,os22}@nyu.edu,
[2] New York University Polytechnic School of Engineering
{sms22,rkarri}@nyu.edu

Abstract. Scan-based design-for-testability, which improves access and thus the test quality, is highly vulnerable to scan attack. While in-field test is enabled through the scan design to provide debug capabilities, an attacker can leverage the test mode to leak the secret key of the chip. The scan attack can be thwarted by a simple defense that resets the data upon a switch from the normal mode to the test mode. We proposed a new class of scan attack in [15] using only the test mode of a chip, circumventing this defense. In this book chapter we extend our earlier work by introducing case studies to explain this new attack in greater detail. Furthermore, we study the effectiveness of existing countermeasures to thwart the attack and propose a new input corruption countermeasure that requires a smaller area overhead compared to the existing countermeasures.

Keywords: AES, Scan Chain, Scan Attack, Scan-based DFT, Testability, Security

1 Introduction

Scan-based design-for-testability (DFT) technique is widely used, which provides internal access to the circuit in order to enhance the testability of manufacturing defects. To provide debug capabilities, in-field test is enabled, by retaining scan capabilities. However, the scan design can be misused by an attacker to leak secret information of a crypto-chip. The intermediate results of a crypto-chip can be shifted out during the test mode through the scan interface. This attack is known as a *scan-based side-channel attack*, and was reported first in [1], which targets Data Encryption Standard (DES) chip. The attacker applies the plaintext in the normal mode after which he/she switches to the test mode to observe the intermediate result of the DES chip. With this information in hand, the attacker can retrieve the secret key by applying differential analysis on the response patterns. Scan-based side-channel attack can also reveal the 128-bit secret key of Advanced Encryption Standard (AES) chip even in the presence of advanced DFT techniques such as partial scan [3], X-masking [4], and X-tolerant architecture [5,6]. Other ciphers such as RSA [7,8], ECC [9,10], and stream ciphers [11], are also vulnerable to scan attack.

The existing scan-based attacks switch the chip between the normal mode and the test mode and assume that the content of the scan cells is retained intact. A mode-reset countermeasure [12] thwarts these attacks by resetting the scan cells upon switching from the normal mode to the test mode.

We proposed a new scan-based attack on AES using only the test mode to circumvent the mode-reset countermeasure [15]. The attack showed how to overcome the challenges introduced by the mode-reset countermeasure and to leak the secret key of a AES crypto core. In this book chapter we explain the attack in greater detail with case studies. We also discuss some of the existing countermeasures and their effectiveness against our test-mode-only attack. We also propose a novel cost-effective input corruption countermeasure, crafted to thwart our test-mode-only scan attack.

The remainder of the paper is organized as follows. Section 2 provides background on AES and DfT. Section 3 provides motivations and challenges of the test-mode-only attack. Section 4 describes the proposed test-mode-only attack. Section 5 gives an overview of the existing countermeasures against scan attacks while Section 6 provides a new countermeasure to thwart the test-mode-only attacks. Finally, we conclude the paper and elaborate on our future work in Section 7.

2 Preliminaries

2.1 AES

AES is a 128-bit symmetric key block cipher available in three different key lengths: 128, 192 and 256 bits. The entire AES algorithm is divided into several identical round operations. The number of rounds in the three different versions of AES are 10 (128-bit key), 12 (192-bit key) and 14 (256-bit key) respectively. Each round comprises of following four basic transformations,

- SubBytes is a non-linear substitution operation.
- ShiftRows is the byte-wise permutation.
- MixColumns is the four-byte mixing operation
- AddRoundKeys is the XORing the state with the round key.

We will refer to these operations as *SB*, *SR*, *MC* and *ARK* respectively. Figure 1 shows the structure of first round of AES which contains an extra key XORing at the beginning.

Each transformation of AES round has specific properties. We use the following properties in our attack.

2.2 Differential Properties of AES

In S-box, for an input X and the input difference α, the output difference β is represented as

$$\beta = SB(X \oplus \alpha) \oplus SB(X) \tag{1}$$

For a given value of (α, β), there could be two, four or no solution of X [13]. In case of two solutions, if one solution is δ, the other solution will always be $\delta \oplus \alpha$. On the other hand for four solutions, the solutions will be δ, $\delta \oplus \alpha$, 0 and α.
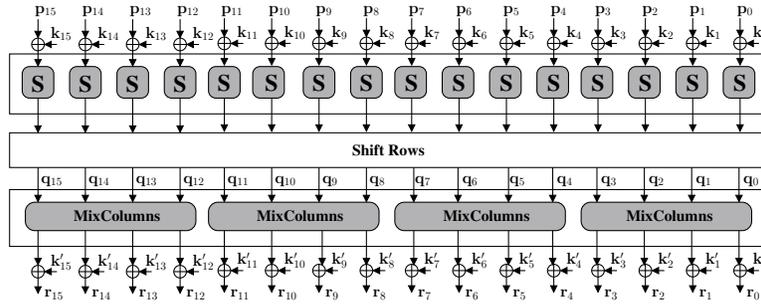
Fig. 1. First round of AES: p_i is the plaintext byte, k_i is the initial key byte, q_i is the *SR* output byte, k'_i is the round key byte, and r_i is the round output byte

Lemma 1. *For a given input X and two one-bit differences e_i and e_j (where e_i and e_j are eight bit strings with a 1 in position i and j respectively where $i \neq j$), the output differences d_i and d_j are generated as*

$$d_i = SB(X) \oplus SB(X \oplus e_i)$$
$$d_j = SB(X \oplus e_j) \oplus SB(X \oplus e_j \oplus e_i)$$

For any value X, d_i can not be the same as d_j.

Proof. We prove this by contradiction. Let as assume that there is a value x of X for which $d_i = d_j$. We also have $y = x \oplus e_j$. Therefore, we have two equations:

$$d_i = SB(x) \oplus SB(x \oplus e_i)$$
$$d_j = SB(y) \oplus SB(y \oplus e_i),$$

where $d_i = d_j$ implies x and y are the two solutions of (1) where $\beta = d_i = d_j$ and $\alpha = e_i$. If x and y are the two solutions of the equation, then either $y = x \oplus e_i$ or one of these two values must be zero and the other equals e_i. If we consider the first case then $y = x \oplus e_i \Rightarrow x \oplus e_j = x \oplus e_i \Rightarrow e_j = e_i$ which contradicts our assumption. Let us consider the second case, $x = 0$ and $y = e_i$ which implies $y = x \oplus e_j = e_i \Rightarrow e_j = e_i$. Again this contradicts our assumption. Therefore, d_i and d_j can not be the same under the given condition.

2.3 DFT Structure

Scan design is a well-known DFT technique that provides direct access to flip-flops. Scan design converts each flip-flop into a fully accessible scan cell that can be controlled and observed easily through shift operations. Due to the limited number of tester channels, especially for integrated circuits with a large number of input/output pins, boundary scan design [14] is used, which associates a boundary scan cell to each primary input/output of the circuit (BSC in Figure 2). Boundary scan design enables a cost-effective few-pin access to the primary inputs/outputs in the test mode. On one

hand, in the normal mode, the input pins directly drive the primary inputs (PIs) of the chip, while the primary outputs (POs) drive the output pins. On the other hand, in the test mode, the boundary scan cells break the connection between the chip pins and the primary input/outputs. The test mode consists of shift and capture operations. During shift operations, the test vectors are serially shifted into the scan chains through the chip input pins, while test responses can be observed through the chip output pins. During the capture operation, the scan cells capture the response of the combinational logic, while the boundary scan cells drive the PIs and capture the POs. As a result, there is no direct path from the chip inputs to the primary inputs in the test mode.
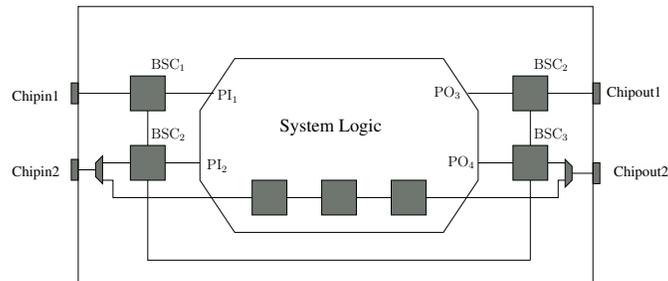

Fig. 2. DFT structure for a design with two scan chains; TDI/TDO pins not shown

3 Motivation and Challenges

In this section we review the existing state-of-the-art scan attack on AES and discuss its limitations. Subsequently, we elaborate on how to overcome these limitations and propose a new scan attack on AES: test-mode-only scan attack.

3.1 State-Of-The-Art Scan Attack

The state-of-the-art scan attack requires switching between the normal and test modes. Figure 3 shows the AES core with a single scan chain which consists of AES round register flip-flops (SC in the figure). In the normal mode, a plaintext is applied to the AES core through the primary input and the results are stored in the round register. The round register content is again applied to the AES in the next round operation. The round operation is highlighted green in Figure 3. After completion of ten rounds the ciphertext output is generated at the primary outputs. In the test mode, the flip-flops of the round register are connected into a scan chain. During the shift cycle a test vector is shifted to the scan chain, while in the capture cycle the response of the AES core is captured and shifted out through the SO line.

The attack works in two steps. In the first step, the chip is run in the normal mode with a desired input plaintext applied from the primary inputs for only one round of

AES. The round output is stored in the round register. Then, in the second step, the chip is switched to the test mode and the stored round output is shifted out through the SO line by applying repeated shift cycles. These two steps are repeated for different plaintexts.

The attack uses one of the basic differential properties of the AES round operation. In AES, if one-bit difference is applied in the least significant bit of any one of the sixteen bytes, the output difference can have 18 possible hamming distances. Out of these hamming distances, only four can be generated by a unique pair of S-box inputs. Therefore, the attacker should follow the above two steps for all possible 2^7 pairs of plaintexts, in order to get any such hamming distance. Once the hamming distance is observed, the corresponding key byte is retrieved by XORing the plaintext byte with the corresponding unique pair of S-box inputs. Therefore, each unique pair will produce two possible key bytes. The same procedure is repeated for all the sixteen key bytes. Finally, by applying $2^7 \cdot 16 = 2^{11}$ plaintext pairs, the search space of the AES key is reduced to 2^{16}. These 2^{16} key hypotheses can be brute-forced by using the plaintext-ciphertext pairs in negligible time.

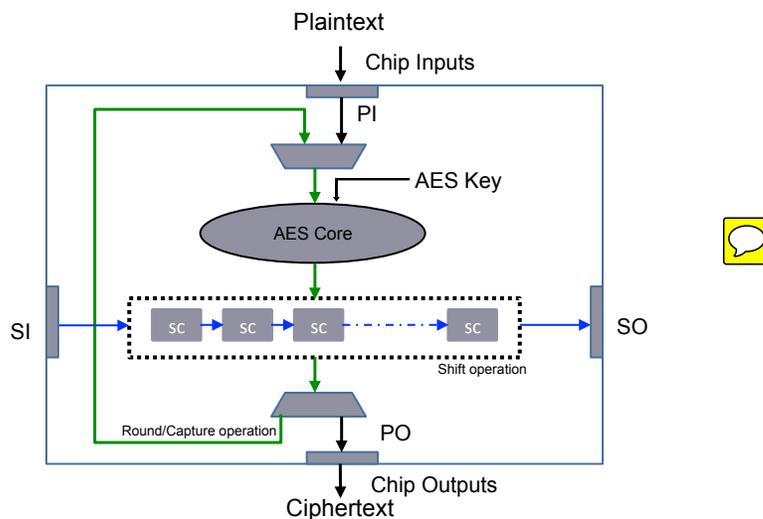**Fig. 3.** AES Core with a single scan chain; SC represents the flip-flop/scan cell of the round register

3.2 Limitations of the Existing Scan Attacks

One fundamental limitation of this kind of attacks is that they require switching from the normal mode to the test mode under the assumption that the data in the scan cells can be preserved intact during the switch. One simple countermeasure is resetting the

device whenever there is a switch from the normal mode to the test mode. In this case, the attacker can perform the first step of the attack and the round output will be stored in the round register. However, whenever the second step of the attack is applied i.e., switching to the test mode, the content of the SCs will reset. Hence, the second step will only produce the zero response of the AES core. Therefore, any scan attack that requires switching between the normal and test modes is bound to fail against this simple countermeasure.

3.3 Overcoming the Limitations of Existing Scan Attacks

In order to overcome the limitations of the above attack, the attacker must come up with an attack that will only use the test mode. As there is no switching required from the normal mode to the test mode, the aforementioned mode-reset countermeasure will be circumvented. The attacker is then constrained to use only the operations highlighted in blue and green in Figure 3.

The major challenge associated with such an attack is to apply the desired input plaintext to AES. As explained in Section 2.3, the primary inputs are not directly accessible in the test mode. The inputs should rather be applied by shifting them in as test vectors through SI line. However, the problem is that the attacker does not know which scan cell drives which input of AES. The physical placement tools determine the mapping between scan cells and the AES inputs.

We need to leverage some of the basic differential properties of AES to (1) find out the exact order of the scan cells which correspond to the AES input register, and (2) launch the attack and leak out the secret key.

4 Test-Mode-Only Attack on AES

In this section we explain our test-mode-only attack on AES [15]. In this attack, the plaintexts are applied in the form of test vectors. Before going into the detail of the attack we first define the attackers ability in the form of the resources and information he/she has. As in the conventional scan attack, we assume that the attacker knows the AES algorithm running inside the security chip and the in-field debug of the chip is enabled. For the sake of simplicity, we assume that there are only 128 scan cells in the design, which correspond to the 128-bit round register.

Let us first consider one example of AES operation. Figure 4 shows the results corresponding to each step of the AES first round, and finally the corresponding round-output difference. P and P_{120} are the two plaintexts (120 is the bit position of 1 from the right), while K_0 and K_1 are the first and the second round keys, respectively. It may be noted that K_0 is the whitening key, which is also the 128-bit AES key. The goal of the attacker is to retrieve K_0. The values in the figure are represented in 32-digit hexadecimal format.

In test-mode-only attack, the major challenge to the attacker is to figure out which scan cell corresponds to which input bit of AES. For P, the test vector is same as the plaintext. However, in order to apply P_{120}, the attacker should precisely know which scan cell corresponds to the 120-th input bit of AES. Therefore, to attack AES using
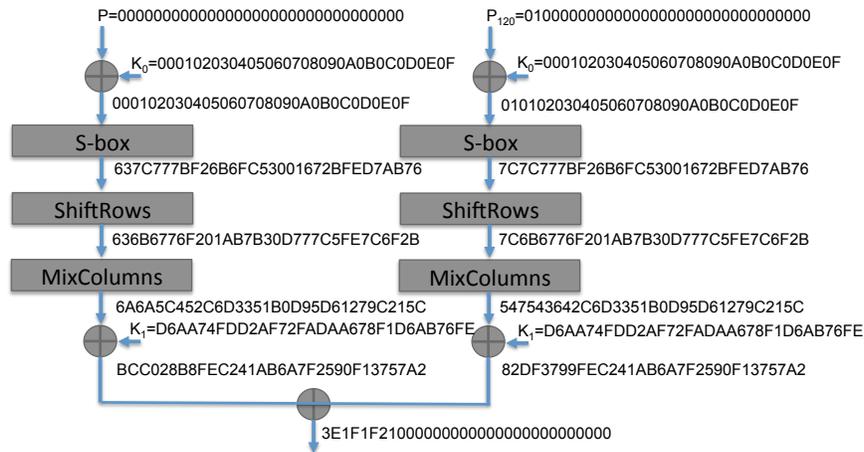
Fig. 4. Example input-outputs of each transformation of first round of AES

only the test mode the attacker has to determine the mapping between the scan cells and the AES inputs. The mapping is determined by the physical placement tool and it is considered as almost random. We determine the mapping in three steps. In the first step, we determine the scan cells corresponding to the AES words. The second step determines the scan cells corresponding to the AES bytes. In the third step, the order of the bytes in a word is determined. Finally, we determine the order of the scan cells in a byte, and identify the corresponding key. Once we have the key bytes, we combine them as per the assumed order, and deduce the possible key hypotheses. In the next subsection, we explain the first step.

4.1 Determining the Mapping Between Scan Cells and AES Input Words

Similar to Figure 4, we use two types of test vectors. One with all zero bits (P) and the other (P_i) with only one bit set to one (i-th bit where $0 \leq i \leq 127$) and the rest of the bits set to zero. At first, the test vector P is shifted in and the response R is captured and shifted out. It may be noted that R is the jumbled up value of the round output $BCC028B8FEC241AB6A7F2590F13757A2$.

Next, in each step a test vector P_i is shifted in and the corresponding response R_i is captured and shifted out. The difference $D_i = R \oplus R_i$ will pinpoint the positions where the bits are flipped. In our example, P_0 and P_{120} differ in the 120-th bit. For this input pair, we apply a one-bit difference to the leftmost input byte. The figure shows that the output difference (0x3e1f1f21) confines within only the leftmost four bytes of the same word. The one-bit input difference affects only one output word. Therefore, two such input differences in the same input word will affect the same output word. We use this property of AES to partition the scan cells into AES input words.

Say, D_i^j is the j-th bit of D_i and $FD_i = \{j \mid D_i^j = 1\}$. So, FD_i is the set of output bit-flip positions corresponding to input bit-flip at i. In 128 iterations, 128 different sets

of output bit-flip positions ($FD = \{FD_i | 0 \le i \le 128\}$) are generated. These 128 sets are combined into four sets by considering the common bit-flip positions in the sets.

Algorithm 1 describes the detailed procedure, where $Enc()$ is the AES round function. The first loop determines the 128 sets $FD_0 \ldots FD_{127}$. In the second loop, in each iteration, a set FD_x is chosen from FD, and compared with the rest of the sets for common output bit-flip positions. If a common bit-flip position is found in any set (FD_y), both sets are combined into one set (FD_x) and the new set (FD_y) is deleted from FD. Therefore, at the end of the iteration, FD_x will have all the output bit-flip positions, which correspond to a particular word. These bit-flip positions, which represent the positions of the scan cells, are saved in W_i. Hence, four iterations of the second loop will partition the 128 scan cells into four words $W_0 \ldots W_3$, which correspond to four different MC operations of AES.

Algorithm 1: Determining the bits in the words

> **Input**: P and P_i where $0 \le i \le 127$
> **Output**: W_0, W_1, W_2, W_3
>
> $R = Enc(P)$
> **for** $i = 0$ *to* 127 **do**
> $R_i = Enc(P_i)$; $D_i = R \oplus R_i$
>   **for** $j = 0$ *to* 127 **do**
>   **if** $D_i^j = 1$ **then**
> $FD_i = FD_i \cup j$
>   **end**
>   **end**
> $FD = FD \cup FD_i$
> **end**
> **for** $i = 0$ *to* 3 **do**
>   Select any element FD_x from FD
> $FD = FD - FD_x$
>   **for** *Each element FD_y in FD* **do**
>   **if** $FD_x \cap FD_y \ne \emptyset$ **then**
> $FD_x = FD_x \cup FD_y$; $FD = FD - FD_y$
>   **end**
>   **end**
> $W_i = FD_x$
> **end**

4.2 Determining the Mapping Between Scan Cells and AES Input Bytes

In order to determine the mapping between scan sells and AES input bytes we use Lemma 1. Similar to previous step, we create four test vectors: P, P_i, P_j and P_{ij}. We already have the response for the base plaintext P. Now we choose any two bit positions, say i and j, where $i \ne j$. We then create P_i and P_j. Say the byte corresponding to bit i is B_l, where $0 \le l \le 15$. We get the response $R_i = Enc(P_i)$ corresponding to P_i. Therefore,

the difference is $D_i = R \oplus R_i$. It may be noted that any non-zero value in D_i is due to the input difference in byte B_l from the plaintext pair (P, P_i). In the rest of the 15 bytes, both plaintexts have the same values.

Next, we create P_j and P_{ij}, where P_{ij} is the same as P_j except in the i-th bit, which is also 1; P_{ij} has only two ones in positions i and j. It may be noted that the j-th bit may be in the same byte as the i-th bit or in a different byte. We get the response $R_j = Enc(P_j)$ and $R_{ij} = Enc(P_{ij})$ corresponding to plaintexts P_j and P_{ij} respectively. Therefore, we have another difference $D_j = R_j \oplus R_{ij}$. Still, this difference is due to the input difference in byte B_l generated from the plaintext pair (P_j, P_{ij}).

Now there are two possibilities: the j-th and the i-th bits are either in different bytes, or in the same byte B_l. Let us consider the first case; the input difference in byte B_l is only due to the flip in the i-th bit which is the same for both plaintext pairs (P, P_i) and (P_j, P_{ij}). Therefore, the differences D_i and D_j corresponding to the two plaintext pairs will be the same. On the other hand if the i-th and j-th bits are both in B_l, then we have the same input difference corresponding to bit i, but in addition to other differences as well. Therefore, as per Lemma 1, this should result in different output differences; the MC operation is linear. Therefore, D_i and D_j will be different in the second case.

For example in Figure 5, the two plaintext pairs (P, P_{124}) and $(P_{120}, P_{124,120})$ have two bit-flips with respect to P; one in the 124-th bit and the other one in the 120-th bit. It may be noted that both bit-flips are in the same byte. Therefore, we have two different input pairs with the same input differences (in bit position 124). Hence, the output difference should always be different. As shown in the figure, the output differences D_0 and D_2 are unequal. If the second bit-flip (120-th) were in a different byte, say in the 100-th bit, the corresponding output differences would be the same. Therefore, by comparing the output differences we can determine whether the two input bit-flips are in the same byte. This way by flipping bits in input pairs we can partition the bits into the AES bytes.

```
P        =00000000000000000000000000000000   D₀=R⊕R₁₂₄=49A9A9E0000000000000000000000000
P₁₂₄     =10000000000000000000000000000000
P₁₂₀     =01000000000000000000000000000000
P₁₂₄,₁₂₀ =11000000000000000000000000000000
                                             0100100110101001101010011110000
                                             0011111000011111000111100100001

                          AES Round          D₁=R⊕R₁₂₀=3E1F1F21000000000000000000000000

                                             D₂=R₁₂₀⊕R₁₂₄,₁₂₀=E7FEFE1900000000000000000000000000

R        =BCC028B8FEC241AB6A7F2590F13757A2
R₁₂₄     =F5698158FEC241AB6A7F2590F13757A2   D₀=R₀⊕R₁₂₄=49A9A9E0000000000000000000000000
R₁₂₀     =82DF3799FEC241AB6A7F2590F13757A2
R₁₂₄,₁₂₀ =6521C980FEC241AB6A7F2590F13757A2
```

Fig. 5. Four inputs and the corresponding one round outputs, and the differences

bits W corresponding to a word, which is generated in the previous step, and partitions the bits into four bytes $B_0 \ldots B_3$. The outer loop selects a bit from W and passes it to the inner loop. The inner loop finds all the bits, which lie in the byte that corresponds to the given bit. The four iterations of the outer loop determine the four bytes $B_0 \ldots B_3$. In order to reduce the number of test vectors, we can use a single pair of test vectors to identify the differences in all the four words in one shot. As the four MC operations of AES in a round are independently calculated, we can apply the above technique to determine the bytes in all the four words simultaneously by flipping bits in four words of the input plaintexts. Therefore, the number of input test vectors will reduce by four.

Algorithm 2: Determining the bits in the bytes

Input: W
Output: B_0, B_1, B_2, B_3

for $l = 0$ *to* 3 **do**
    $B_l = \emptyset; W_t = \emptyset$
    Select i where $i \in W$
    $B_l = B_l \cup i \; ; W = W - i; P_i = P \oplus (\texttt{0x1} \ll i)$
    $R_i = Enc(P_i); R = Enc(P); D_i = R \oplus R_i$
    for *Each* $j \in W$ **do**
        $W = W - j$
        $P_j = P \oplus (\texttt{0x1} \ll j); P_{ij} = P_j \oplus (\texttt{0x1} \ll i)$
        $R_j = Enc(P_j); R_{ij} = Enc(P_{ij}); D_j = R_j \oplus R_{ij}$
        if $D_i \neq D_j$ **then**
            $B_i = B_i \cup j$
        end
        else
            $W_t = W_t \cup j$
        end
    end
    $W = W_t; W_t = \emptyset$
end

Note It may be noted that there could be more than 128 registers in the AES design. However, these extra registers can be identified by considering their effects on the AES round output. If the registers are not part of the round operation (e.g. AES controller registers), they will not affect the round output. If they are part of the round operation, their effect on the output will reveal their position in the round operation. Once identified, the extra registers can be discarded in the attack.

4.3 Determining the Order of Bytes in a Word

In this section, we try to determine the position of each byte in the words. In order to do that, we use the properties of the AES MC operation. Figure 6(a) shows the basic

operation of AES MC in terms of a matrix multiplication operation. As illustrated in Figure 6(b), if there is a non-zero value a in only one of the four input bytes, the output bytes will have values $(a, a, 2a, 3a)$.

(a) AES MC operation

(b) Byte difference distribution in MC

Fig. 6. AES Mixcolumns operation and its differential properties

We analyze the hamming distance to determine the order of the bytes. As per the AES MC operation, two of the four output bytes will always have the same hamming distance (two output bytes with values a).

We apply two plaintexts (P and P') in a way that only one input byte of the MC operation gets a non-zero difference and the rest of the three bytes all get zero differences. Say we have scan cells corresponding to the four bytes B_0, B_1, B_2, B_3 of a word. We need to find the actual order of these bytes corresponding to MC, i.e., to $a_0 \ldots a_3$ in Figure 6(a). If the input difference value a is applied in B_0, the output difference value $2a$ should be loaded in this byte. We need to determine which two bytes will have the same hamming distance. However, for some value of a, the rest of the three bytes (other than the byte B_0) may have the same hamming distance.

We experimentally observed that in 52 out of the 255 different values of a, more than two bytes will have the same hamming distance. We need to try at least 53 different values of a to exactly know the two bytes that correspond to the output differences a and a respectively. Say, B_2 and B_3 are the two bytes. Then we also determine that B_1 corresponds to the output difference $3a$. However, as B_2 and B_3 have the same value, we can't directly identify the order of these two bytes. In order to determine their order, we next apply the input difference to the byte B_1. Then B_0 corresponds to output difference a, and either B_2 corresponds to a and B_3 corresponds to $3a$ or vice versa. In the first case, the exact order of the four bytes is identified to be (B_0, B_2, B_3, B_1), which corresponds to the first of the four cases presented in Figure 6(b). We can have three more possible permutations, which are the circular rotations of this permutation. The procedure is repeated for the rest of the three words. It may be noted that instead of applying difference in one byte at a time, we can apply the byte difference in four words corresponding to four different MC operations, which will help determine the order of the bytes in the four words simultaneously.

So, given the four bytes in a word, we can tell the four possible permutations as per the AES input.

4.4 Determining the Order of Bits in a Byte

At this point, we do not know which scan cell corresponds to which bit of a given byte although we know the order of the bytes. If we apply the existing attack [5] by varying the input pairs in order to get a particular hamming weight, we will uniquely determine the key byte. In our case we don't know the exact value of the input plaintext as we do not know the order of the bits in a byte. Therefore, the key byte which we determined is corresponding to the assumed order of the scan cells in the bytes. Eight scan cells corresponding to a byte can be arranged in $\frac{8!}{4!.4!} = 70$ (consider the average case where half of the bits are one and rest are zero) ways. Therefore, in average each byte can have 70 possible values. If we combine all the sixteen bytes we get $(10)^{16}$ values, which is a large number. Therefore, we must determine the order of the scan cells.

We develop a new technique to determine the key despite the unknown order of the scan cells in the byte. In this scheme, instead of considering the difference in any particular bit, we consider eight different differences, each corresponding to a one-bit difference in one of the eight input bits. As all the bits are considered, the output signature is irrespective of the order of the bits. In order to do that, we choose an input plaintext P with all zero bits. Therefore, the chosen byte will have the value $B = 00000000$. Then we produce eight different plaintexts P_0 to P_7 by varying the bit-flip position 0 to 7.

Let us consider the chosen byte is the first byte. Then the first bytes of the eight plaintexts are $0x01$, $0x02$, $0x04$, $0x08$, $0x10$, $0x20$, $0x40$, and $0x80$. It may be noted that in these eight values, the bit-flip position varies from the right-most bit to the leftmost bit. Each of these eight plaintexts is paired with P. We compute the output difference D_i ($0 \leq i \leq 7$) of each pair and get the hamming distance. From D_i we get the hamming distance of the four bytes. Two of these bytes will have the same hamming distance as shown in Figure 6(b). So, we consider only one of these two.

We maintain a signature table where, for each value of K, we have eight the three-byte hamming distances corresponding to eight input differences. The values are shown in Table 1. The data in the table shows that the last eight columns uniquely determine the row. Therefore, given eight hamming distances corresponding to eight input differences, one can uniquely determine the value of K, which is the key byte itself. It may be noted that we have sixteen tables corresponding to sixteen S-boxes of AES. We need to search the eight hamming distances in all the sixteen tables.

Algorithm 3 shows the detail technique of determining the value of the key byte and the order of the bits in a byte. The attacker get the output differences corresponding to eight pair of plaintexts. She then determines the four output bytes which have got the non-zero output differences. The function $Trim()$ determines four none-zero bytes and reduce them to three after removing the duplicate one. HD will have eight hamming distances correspond to eight input differences. These eight value are searched in the rows of Table 1. When a row matches, the value of K corresponding to that row is the value of the key byte. The position of the bit is determined by the column index of the hamming weight corresponding to the bit. The position of the i-th bit is the index of the column of HD_i in the row K.

Table 1. Signature Table

K	B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7
0	525	224	334	446	334	455	446	455
1	525	233	444	466	367	233	358	233
2	222	224	222	112	556	455	556	334
3	222	233	556	224	222	556	334	345
4	334	255	334	222	112	345	145	255
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
255	255	224	334	266	134	233	244	345

Algorithm 3: Determining the key byte and the order of the bits in a byte

---

Input: P and P_i where $0 \leq i \leq 7$
Output: Key and order of the bits

$HD = \emptyset$ $R = Enc(P)$ **for** $i = 0$ *to* 7 **do**
    $R_i = Enc(P_i)$
    $D_i = R \oplus R_i$
    $HD_i = Trim(D_i)$
    /*Getting three bytes*/ $HD = HD \cup HD_i$
end
for $K = 0$ *to* 255 **do**
    Search HD in row K of Table 1
    if *All elements of HD is found in row K* **then**
        Key=K
        Order of the bit is the column index of the element of HD
    end
end

4.5 Determining the AES Key

In the previous section, we determine the order of the scan cells in a byte as well as the key corresponding to the byte. Now we have to determine the order of bytes in a word (Section 4.3). We have four possible permutations of the four bytes in a word. Therefore, if we know the position of the four words, then there will be $4^4 = 2^8$ possible values of the key. However, we don't know the order of the four words, which can be arranged in $4! = 24$ ways. The total possible hypotheses of the AES key is $24 \cdot 256 = 6144 = 2^{12.58}$. As we know the plaintext and the ciphertext, we can brute-force the key hypotheses in negligible time.

4.6 Attack Complexity Analysis

In the first step of the attack (Section 4.1), we need $128 + 1 = 129$ test vectors to partition scan cells into the AES words. In determining the bytes in a word in the second step (Section 4.2), we need $(1 + 31 \cdot 2) + (1 + 23 \cdot 2) + (1 + 15 * 2) = 141$ test vectors. For getting the order of the bytes in a word in the third step (Section 4.3), we used $52 \cdot 2 = 104$

test vectors. In the fourth step (Section 4.4), where we determine the order of the bits in bytes, we don't need any extra test vectors, because the required test vectors are already available from the first step. Therefore, in total we need $129 + 141 + 104 = 375$ test vectors to determine the secret key.

Regarding the time complexity, it can be observed that most of the execution time is spent on partitioning the scan cells corresponding to bytes and in determining the order of bits in a byte. For the first part, the time complexity is $31 + 15 + 7 = 53$. In the second part the time complexity is $256 \cdot 8 = 2^{11}$. However, the final key hypotheses is around $2^{12.58}$, which need to be brute-forced to get the master key. Therefore, the time complexity of the complete attack is $2^{12.58}$.

5 Existing Countermeasures

So far we have shown that the mode-reset countermeasure, which provides protection against all the existing attacks, fails against the proposed test-mode-only attack. In this section we discuss some of the other existing countermeasures and their level of effectiveness against our proposed test-mode-only attack. Our goal is to devise a countermeasure for our attack.

5.1 Insertion of Inverters in the Scan Path

In this technique the scan chain is divided into multiple subchains in the form of a scan tree [16]. The subchains are placed in a random order so that it will be difficult for the attacker to figure out the exact round output. Some of the scan cell values are flipped by placing inverters selectively on the scan paths (Figure 7). The location of the inverters is known only to the designer or the tester. Therefore, only the designer or the tester can retrieve the exact response of the chip, while the attacker can only observe the modified response of the chip.
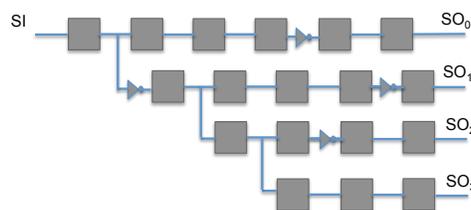

Fig. 7. Insertion of inverters in the scan path

In the differential scan attack, such countermeasures will fail, as the effect of the inverters will be cancelled out in the output difference. Therefore, the proposed test-mode-only attack can easily overcome this countermeasure through the differential analysis it employs. In the same way, our proposed scan attack can thwart the scan scrambling technique proposed in [17].

5.2 Masking

There are two masking countermeasures proposed in [6]. The first method masks the 128-bit round register and then unmasks it for the subsequent round operation. The mask value is generated by using an input and a key of size 128-bit (Figure 8). During the normal operation, the masking remains transparent. In the capture operation during the test mode, the scan chain captures the masked response of the circuit. Only the tester and the designer have the knowledge about the mask value. Therefore, they can unmask the scan output and retrieve the actual response of the circuit. On the other hand, the attacker will be unable to retrieve the actual response, hence failing to apply the differential analysis to reveal the secret key. Thus the attack will fail. The drawback of this countermeasure is not only the area overhead, but also the prolongation of the critical path, and thus the performance degradation.
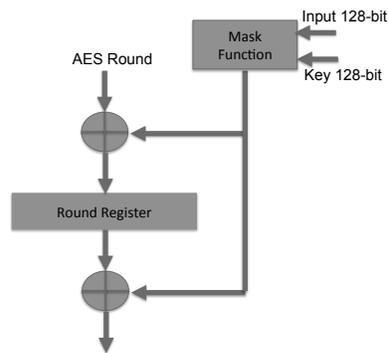

Fig. 8. Round register masking

To overcome the critical path issue, a second masking technique is proposed, which modifies the response compactor output (Figure 9). This technique is based on an en-hanced Linear Feedback Shift Register (eLFSR), which can either operate as a register or an LFSR. In the test mode, the LFSR output is XORed with the response compactor output bit-stream. Therefore, each compacted slice is XORed with a pseudorandom bit. A 128-bit LFSR is used to prevent the attacker from retrieving the initial state of the LFSR. Although the countermeasure has no impact on the critical path delay, the area overhead remains a problem, which is around 5% of the AES chip.

5.3 Noise Injection In the Scan Output

This technique is similar to the previous masking technique. It includes two levels of security: one is the LFSR and the other one is the True Random Number Generator (TRNG) [18]. Figure 10 shows the architecture of the countermeasure. In this case, the masking values are controlled by both the LFSR and the TRNG. Therefore, on average
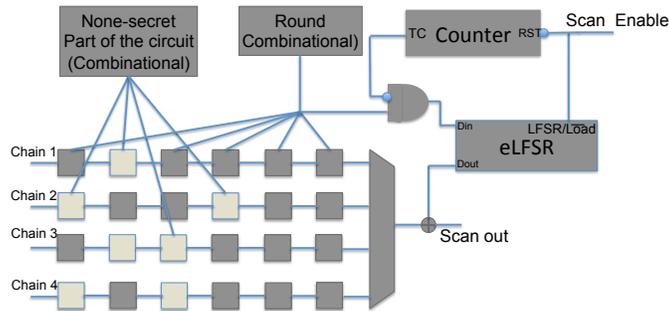
Fig. 9. Masking the compactor output

half of the scan output bits are masked and the rest of the bits remain unchanged. As the TRNG hides some of the LFSR outputs, the designer can choose a smaller LFSR, which will drastically reduce the area overhead of this countermeasure. The success rate of the attack in the presence of this countermeasure depends on how frequently the masking is done. It was shown that the attack will fail, only when the masking is done in each clock cycle.
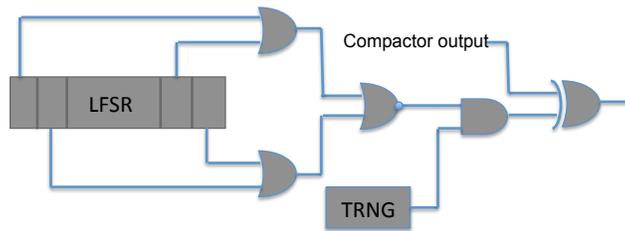

Fig. 10. Randomized noise injection

5.4 On-chip Comparison of Responses

The countermeasure in [19] provides a scan protection scheme using an on-chip comparison of the circuit response and the expected response that is shifted in. The attacker can only observe a one pass-fail bit per test vector, which is insufficient to leak the secret key of the chip. Similar to the traditional test process, the test vector is shifted into the scan architecture during the shift operations. Then, the response is captured during the capture operation. Instead of shifting out the captured response, the expected response is shifted in through another pin that replaces the scan output pin, while shifting in the next test vector by using the scan input pin. This countermeasure performs an on-chip comparison, which produces a pass, only when the whole captured response matches the expected response. The test time and data volume is the same compared to traditional testing.

This countermeasure impacts the diagnosis and debug capabilities of the IC, as the entire information about the IC under test consists of one pass-fail bit per test vector. To support the diagnosis of the IC, an expected faulty response of one fault at a time is shifted into the scan architecture instead of the fault-free response, which is compared with the captured response. However, such procedure is only valid for modeled faults. Furthermore, the diagnostic time becomes unaffordable.

6 Proposed Countermeasure

As we have seen in the previous section, there are various proposed countermeasures to provide a certain level of security against scan-based side-channel attack. However, all these countermeasures have been proposed with the existing scan attack model in mind, thus failing to provide protection against a fundamentally different attack model, such as the proposed test-mode-only attack. In this section, we present a countermeasure judiciously crafted for the test-mode-only attack that relies on the access to the scan input pin, which no other existing scan attack requires.

Our proposed test-mode-only attack is based on launching an individual bit-flip from each scan cell in the round register independently of the remaining scan cells. Disabling this capability will prevent the attacker from mapping each scan cell to its corresponding word and byte of the AES. We propose a defense mechanism, which injects random noise by randomly flipping the bits of the test vectors, while they are shifted through the scan input pin, hampering the attack's ability to classify scan cells into words and bytes. The LFSR randomly selects the locations of the bit-flips of each test vector. The LFSR generates a new bit per clock cycle that is XORed with the shifted test vector bit. In other words, the shifted bit is flipped if the output of the LFSR is one. No countermeasure for the existing scan attacks has considered the alteration of the test vector shifted through the scan input pin, mainly because the previously proposed scan attacks assume that the attacker can apply the plaintext through the primary inputs.

Without the capability of controlling the data delivered into the scan cells, the attacker fails to launch the desired bit-flips. The attacker thus needs to circumvent this defense by deciphering the LFSR sequence, which requires the LFSR structural details. The only way the attacker can identify the LFSR configuration is to apply the same test vector for a sufficient number of times by using the scan flush test capabilities, where the test vector can be shifted in and out without any capture operation. If the LFSR consists of m bits, the attacker has to apply the same test vector for $2^m - 1$ times to figure out the structure of the LFSR that can be chosen to implement a primitive polynomial. For an LFSR of medium size, this can be quite costly for the attacker.
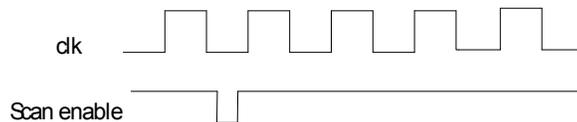

Fig. 11. Timing diagram of the clock signal and the scan enable signal in a dummy capture attack

To provide a higher level of security, we propose the optional integration of a second defense mechanism to disallow a specific number of continuous shift cycles. Another possible smaller LFSR can be used to inject extra noise on the shifted stimulus, once a specific number of consecutive shift cycles with no capture operation in between is detected. The maximum allowed number of continuous shift cycles should be selected carefully to maintain both security and testability of the chip. If the maximum number of continuous shift cycles is too small, the scan flush test will be affected. On the other hand, permitting a large number of continuous shift cycles may enable the identification of the first LFSR structure, and thus, compromise the security of the chip. An n-bit counter can be used to allow up to $2^n - 1$ consecutive shift operations. Every clock pulse during the test mode increments the counter as long as the scan enable signal is high, indicating a shift operation. When the counter saturates, it signals the second LFSR to kick in and add a second level of bit flipping.

To circumvent the second level of defense, an attacker can employ a dummy capture attack as shown in Figure 11; in this scheme, the attacker pulls down the scan enable signal (to indicate capture) in between the active clock edges. The goal of the attacker is to reset the counter and be able to extend the number of the consecutive shift cycles without corrupting the scan chain content with a capture operation. To thwart such attacks, the reset structure of the counter is designed carefully. The counter is reset only with a capture during an active clock. Figure 12 provides the implementation details.
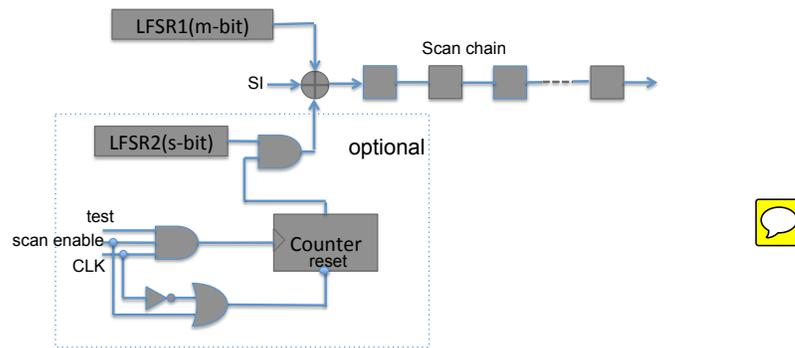

Fig. 12. The proposed countermeasure for test-mode-only attack

The area cost is an m-bit LFSR and one XOR gate for the level-1 defense, and an s-bit LFSR, c-bit counter, 3 AND gates, one OR gate, one inverter, and one XOR gate in addition for the optional level-2 defense.

7 Conclusion and Ongoing Work

With a simple yet effective chip mode-reset countermeasure in place, none of the existing scan attacks can leak the secret information of a security-critical IC. In this work, we bring a new perspective to scan attack research by introducing a new class of scan

attacks, test-mode-only attack, that targets circumventing this commonly used mode-reset countermeasure. The proposed attack misuses the in-field debug capabilities of the IC by leveraging operations, such as shift and capture that the test mode offers. By remaining in the test mode, the proposed attack circumvents the mode-reset countermeasure.

A fundamentally different type of scan attack comes with its own challenges to be tackled. As the boundary scan cells block the access to the primary inputs during the test mode, the proposed attack ends up having to use the scan interface to load the attack patterns, in contrast to all the existing scan attacks that benefit from the available PI access in the normal mode. The prerequisite for applying the proposed test-mode-only attack is therefore to decipher the mapping between the scan cells and the AES inputs, which is dictated by the physical position of the scan cells and the physical placement tools used by the designer.

In this work, we tackle the aforementioned challenges and devise a test-mode-only attack, which we illustrate for AES, while we note that the technique herein can be easily extended for other ciphers as well. The attack analysis shows that only 375 test vectors are sufficient to reveal the 128-bit AES secret key where the 128-bit key is reduced to only 12 bits. Furthermore, we also present a brief overview of existing countermeasures, and devise a cost-effective one that can circumvent the proposed test-mode-only attack. We thus present a two-level defense with an underlying input corruption countermeasure that prevents an attacker from launching the bit-flips required to leak the secret key.

Our future work will focus on contemporary scan architectures where a test stimulus decompressor and a response compactor reside on the scan path. We have already made some progress on attacks using compactor [20] and boundary scan chains [21]. The technique we present in this paper is readily applicable on ICs that enable debug and diagnostics with a bypass of such advanced DFT features. When this bypass capability is not supported, however, the attack should be enhanced by developing techniques to launch bit-flips through the decompressor in the scan cells, and to perform differential analysis on the compacted responses. The ultimate goal of this line of work is to develop techniques that facilitate the design of **testable yet secure ICs**.

References

1. Yang, B., Wu, K., Karri, R.: Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard. In: ITC, IEEE (2004) 339–344
2. Yang, B., Wu, K., Karri, R.: Secure scan: a design-for-test architecture for crypto chips. In Jr., W.H.J., Martin, G., Kahng, A.B., eds.: DAC, ACM (2005) 135–140
3. Kapur, R.: Security vs. test quality: are they mutually exclusive? In: Test Conference, 2004. Proceedings. ITC 2004. International. (Oct.) 1414–
4. DaRolt, J., Natale, G.D., Flottes, M.L., Rouzeyre, B.: Are advanced DfT structures sufficient for preventing scan-attacks? In: VTS, IEEE (2012) 246–251
5. Ege, B., Das, A., Ghosh, S., Verbauwhede, I.: Differential Scan Attack on AES with X-tolerant and X-masked Test Response Compactor. In: DSD, IEEE (2012) 545–552
6. DaRolt, J., Natale, G.D., Flottes, M.L., Rouzeyre, B.: Scan Attacks and Countermeasures in Presence of Scan Response Compactors. In: European Test Symposium, IEEE Computer Society (2011) 19–24

7. Nara, R., Satoh, K., Yanagisawa, M., Ohtsuki, T., Togawa, N.: Scan-based side-channel attack against rsa cryptosystems using scan signatures. IEICE Transactions **93-A**(12) 2481–2489

8. DaRolt, J., Das, A., Natale, G.D., Flottes, M.L., Rouzeyre, B., Verbauwhede, I.: A New Scan Attack on RSA in Presence of Industrial Countermeasures. In Schindler, W., Huss, S.A., eds.: COSADE. Volume 7275 of Lecture Notes in Computer Science., Springer (2012) 89–104

9. Nara, R., Togawa, N., Yanagisawa, M., Ohtsuki, T.: Scan-based attack against elliptic curve cryptosystems. In: Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific. (Jan 2010) 407–412

10. Da Rolt, J., Das, A., Di Natale, G., Flottes, M., Rouzeyre, B., Verbauwhede, I.: A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures. In: Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on. (Oct 2012) 43–48

11. Liu, Y., Wu, K., Karri, R.: Scan-based attacks on linear feedback shift register based stream ciphers. ACM Trans. Des. Autom. Electron. Syst. **16**(2) (April 2011) 20:1–20:15

12. Hely, D., Bancel, F., Flottes, M.L., Rouzeyre, B.: Test Control for Secure Scan Designs. In: Proceedings of the 10th IEEE European Symposium on Test. ETS '05, Washington, DC, USA, IEEE Computer Society (2005) 190–195

13. Nyberg, K.: Generalized feistel networks. In Kim, K., Matsumoto, T., eds.: ASIACRYPT. Volume 1163 of Lecture Notes in Computer Science., Springer (1996) 91–104

14. : IEEE Standard Test Access Port and Boundary-Scan Architecture. IEEE Std 1149.1-2001 (2001) i–200

15. Ali, S.S., Sinanoglu, O., Saeed, S.M., Karri, R.: New scan-based attack using only the test mode. In Margala, M., da Luz Reis, R.A., Orailoglu, A., Carro, L., Silveira, L.M., Ugurdag, H.F., eds.: VLSI-SoC, IEEE (2013) 234–239

16. Sengar, G., Mukhopadhayay, D., Roy Chowdhury, D.: An efficient approach to develop secure scan tree for crypto-hardware. In: Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on. (Dec 2007) 21–26

17. Hély, D., Flottes, M.L., Bancel, F., Rouzeyre, B., Bérard, N., Renovell, M.: Scan Design and Secure Chip. In: IOLTS, IEEE Computer Society (2004) 219–226

18. Das, A., Ege, B., Ghosh, S., Batina, L., Verbauwhede, I.: Security analysis of industrial test compression schemes. IEEE Trans. on CAD of Integrated Circuits and Systems **32**(12) (2013) 1966–1977

19. Da Rolt, J., Di Natale, G., Flottes, M.L., Rouzeyre, B.: On-chip test comparison for protecting confidential data in secure ics. In: Test Symposium (ETS), 2012 17th IEEE European. (May 2012) 1–1

20. Ali, S.S., Sinanoglu, O., Saeed, S.M., Karri, R.: New scan attacks against state-of-the-art countermeasures and dft. In: HOST, IEEE (2014) 142–147

21. Ali, S.S., Sinanoglu, O., Karri, R.: Test-mode-only scan attack using the boundary scan chain. In Natale, G.D., ed.: ETS, IEEE (2014) 1–6