Abstract—Scan-chains are test infrastructures included in a circuit for providing high fault coverage. However, they can be exploited by an attacker as a side-channel in the case of a cryptographic application like AES. Test Compression and thereafter X-tolerance and X-masking over it, which reduce test effort without compromising on testability, can help in counteracting scan-based attacks. This work focuses on the security issues of an AES-circuit containing test compression with X-masking and X-tolerance logic. With experimental results, we show the weakness of such an AES circuit against our modified differential scan-attack. Finally, the paper outlines two suitable countermeasures to prevent such attacks.

I. INTRODUCTION

As the manufacturing costs are economized and complex cryptographic algorithms become a burden to a product in terms of throughput, the tendency to implement cryptographic algorithms in the form of application specific integrated circuits (ASICs) or co-processors is increasing. However, the more functionality one includes in a circuit, the more manufacturing related faults can occur in the final product. Therefore, testing these circuits is of crucial importance for the sake of integrity of the product.

Testing complex circuits in a short time, without compromising test quality, is achieved by test compression, and it is now widely deployed in the semiconductor industry. There are large numbers of unspecified or don’t-care (X) states in the auto-generated test patterns which are filled up randomly with a fixed combination of 0s or 1s for the sake of testing with an automated test equipment (ATE). These unspecified values can be removed leading to a compression of test patterns given as input stimuli to the scan-chain design-for-test infrastructure of a complex circuit. Test Compression not only involves compression of test patterns but also compaction of the output test responses. Therefore, test compression schemes often include a compaction algorithm for either reducing the time or space required for handling test outputs. Moreover, these compaction algorithms often include additional logic for preventing unknown states (X-states) from corrupting test outputs. This is either done with X-tolerant logic, or eliminating the X-states in the output, which is referred to as X-masking.

In test compression schemes, externally observable entities are only the compressed stimuli and compacted responses. A generic representation of such a test compression scheme is presented in Figure 1. In light of this, it is natural to expect these schemes to have some security properties as well. A theoretical security analysis of one of the leading test compression schemes, Embedded Deterministic Test (EDT), is presented in [1]; while the security claims of Tessent TestKompress, which uses EDT for test compression, appeared later in a Mentor Graphics whitepaper [2].

Cryptographic circuits need a special testing strategy owing to the constraints on security. Although scan-chain Design-for-Test (DFT) offers the highest testability, they are prone to scan-based side channel leakages which may help an intruder to perform a non-invasive attack on secure chips to extract secret information such as cryptographic keys from secure hardware implementations. In fact, there are works available in the literature which exploit this property, and they are referred to as scan-based side channel attacks [3], [4], [5]. However, these works do not consider any X-tolerant or X-masking logic which is widely used in industrial test compression schemes, and gravely affects the applicability and running time of these attacks.

In this work, an attack on an AES-128 design with a generic test compression scheme including response compaction in the presence of X-masking and X-tolerant logic is presented. An improved differential scan attack is used with some additional observations for recovering the cryptographic key of the design. Simulation results in terms of success rate of the attack is also presented depending on different distributions of the key-dependent memory elements in the scan-chain structure.

This work is organized as follows: in Section II, we describe the previous work that has been done in the domain of test compression security and scan-based attacks on test
structures with XOR-compaction. In Section III, we present the background of our work, including a discussion of our target cipher AES, the initial scan attack proposal and the recent differential scan-attack on AES. Section IV describes our scan-attack approach, with a discussion on the attack parameters. While Section V proposes our improved scan-based attack on AES implementations containing scan-chain DFT with X-masking, Section VI presents our scan-based attack on AES implementations containing scan-chain DFT with X-tolerant logic. A discussion on the state-of-the-art scan attack countermeasures is presented in Section VII. Finally, we conclude the paper together with some future work ideas in Section VIII.

II. PREVIOUS WORK

The first attempt on analysing the security of test compression circuits is presented by Yang et al. in 2006 [3]. The attack exploits the possibility of scanning out the contents of the round register after execution of one round of encryption or decryption. Later in 2007, Liu and Huang published an analysis [1], which also considers the response compactor of a test compression scheme. In that work, the authors focus on the Embedded Deterministic Test by Mentor Graphics, and evaluate the security of the scheme by identifying the flip-flops (FFs) which can be used for inferring the encryption key. The authors refer to these registers as key registers, and similarly the term key dependent Flip-Flops (KFFs) is used for those registers in the rest of this work. In [1], the authors claim that identification of these KFFs in the scan design is crucial for successful recovery of the encryption key. However, this has been contradicted in later works presented by Da Rolt et al. [4], [5].

In [5], Da Rolt et al. present a scan based attack on an AES design with a scan response compactor, in which the identification of KFFs is not necessary for mounting a successful attack. They show that the attack proposed in [3] is directly applicable to designs which use an XOR tree structure for scan response compaction. They also provide different attack strategies for different distributions of KFFs over the scan chains. However, the scan-attack assumes a simple XOR compactor structure, without considering X-masking or X-tolerant architectures, which are widely used in industrial test compression schemes, and can affect the success of the attack.

Some scan-attack countermeasures have also been proposed in the literature. However, these countermeasures either involve a change in the DFT structure or the design itself; therefore making them less likely choices for an industrial product in an integration environment.

III. BACKGROUND

Block ciphers provide security by transforming the state through a simple collection of operations which is called the round function. The required number of iterations for security is entirely dependent on the algorithm. For instance, Data Encryption Standard (DES) requires 16 iterations whereas the block cipher KATAN requires 254, as KATAN has a much simpler round function than the round function of DES. Generally, the more complicated the round function becomes, the less iterations are needed to achieve cryptographic security. Hence, in the case of AES, the round function is only iterated 10, 12 or 14 times depending on whether the key length is 128, 192 or 256 bits. This is a result of a nicely designed round function with good cryptographic properties. In this work, the attacks are demonstrated on AES as it is the industry standard block cipher and it also enables the reader to compare the work with previous works available in the literature.

A. Advanced Encryption Standard

AES is the industrial standard block cipher which encrypts blocks of 128-bit messages and supports variable key sizes: 128 bits, 192 bits, or 256 bits [6]. The AES round function is constituted of four operations which are applied to the state in the following order: SubBytes, ShiftRows, MixColumns and AddRoundKey. The SubBytes operation is a non-linear S-box which operates on each byte of the state. ShiftRows rotates the rows of the state, as in Figure 2, to improve diffusion when used together with the MixColumns function. The MixColumns function operates on columns of the state which can be visualized as multiplying each column with a Maximum Distance Separable (MDS) matrix of branch number five. Therefore, any byte of the input will affect all four bytes of the output which forms the basis of differential scan-based attacks. For instance, a non-zero byte difference in the first byte, as it is in Figure 3, will transform into a non-zero difference after SubBytes and will not be affected by the ShiftRows operation. However, the MixColumns operation will lead to four non-zero differences on that column. The AddRoundKey operation has no effect on difference propagation as long as the same key is used, since it only XORs the state with the round key generated by the key scheduling algorithm.

Additionally, there is an initial key XOR step before the encryption starts, and this is the operation that we will target
to recover the encryption key of AES. Further details regarding the specification of AES can be found in [6].

B. Scan Testing

Scan design has been generally accepted as the standard method of testing chips due to the high fault coverage and low overhead. Including scan while designing the chip requires one additional pin to the primary I/O to serve as the test control pin (TC). Internally, there is little impact on the design since the standard flip-flops (FFs) are exchanged with scan flip-flops (SFFs) (flip-flops with an input multiplexer) which are then linked to one another creating a scan chain. TC selects between functional and test mode operations. An example of a scan chain is shown in the Figure 4. TC controls each multiplexer, choosing between the normal mode input of the FF or the output of the previous SFF in the chain.

The FF registers make up the I/O to the combinational logic blocks in the chip, so test engineers are able to manipulate the values that are input (controllability) and view the output (observability) of each block. This is performed by multiplexing one primary input pin and one primary output pin as the scan-in (SI) pin and scan-out (SO) pin, respectively.

Using the SI pin while the TC is enabled, a test pattern is scanned into the scan chain as dictated by the system clock. When the entire pattern is scanned in, the TC is disabled, and the chip is run in normal mode for one cycle storing the responses back in the SFFs. TC is again enabled to scan out the response, while at the same time, scanning in a new test pattern to check for new faults previous patterns were not able to detect. Using this method of test, sequential logic essentially becomes combinational logic during test. Creating test patterns that achieve high fault coverage is a much easier task for combinational logic than it is for sequential logic, significantly speeding up the test pattern generation [7].

C. Differential Scan Attack on AES

This attack basically exploits the fact that two particular inputs to the round function of AES can transform into output vectors with a unique Hamming distance in between after one round of encryption. For instance, if two plaintexts with an XOR difference of 0x01 in their least significant byte (LSB), are encrypted using only one round of AES, the Hamming distance between the one round output vectors can only have 18 values. The distribution of these values is given in Figure 5. The y-axis denotes the number of pairs of plaintexts, while the x-axis represents the observed Hamming distances at the output.

The first scan-based attack on AES appears in [8] in which a two-step approach employing chosen plaintexts for attacking the round register of AES is presented. In the first step, the bits in the scan-chain output corresponding to the round register are determined. This proceeds as follows: the chip is run in functional mode for one clock cycle. The result after the first round is stored in the round register. Then the chip is switched to test mode and the contents of the round register are scanned out. This step is repeated for another plaintext input differing in only one byte. Note that a one byte difference in plaintext will transform into a four byte difference due to the structure of the MixColumn operation. In fact, analysing the distribution of the Hamming distances for all 2n pairs generated with byte difference 0x01 in their LSB, one can easily verify that there are four Hamming distance values (9, 12, 23 and 24) which can only be generated by a unique pair of inputs. Therefore, whenever such a Hamming distance is observed between the output vectors, one can XOR the corresponding plaintext byte with the pre-computed values to recover a byte of the encryption key.

As an example, let the attacker be able to encrypt two plaintexts which have a difference of 0x01 in their LSB and observes a Hamming distance of 9 between the one round output vectors. Since the attacker has pre-computed the possible Hamming distances depending on all possible input pairs with 0x01 difference in their LSB, she finds out that the inputs to the S-box should either be 0xE2 or 0xE3. Hence, the only thing that remains is to XOR these values to the corresponding byte of the plaintexts and obtain two possible keys, one of which is definitely the correct key byte. Proceeding in this way, the attacker can reduce the search space for the key from 2^128 to 2^10, and eventually recover the 128-bit encryption key used in that AES implementation in negligible time. Here, the number of encryptions the attacker needs depends on the byte difference used for attacking the system. The more unique values in the Hamming distance distribution, the better the chances are for the attacker to be able to reduce the search space for that key byte.

The simplicity of the attack enables it to be applicable to schemes with a simple compaction scheme such as an
XOR tree compactor. The XOR tree compactor is a simple scheme which XORs the outputs of each scan chain in every clock and therefore saving the space required to keep the test outputs. However, this XOR operation can affect the observed Hamming distances depending on the distribution of KFFs over the scan chain structure. In [5], Da Rolt et al. analyse the effect of an XOR compaction on the basic attack and show that the attack is still applicable to these systems but some additional work is required to successfully recover the key.

In this work, we take this approach to the next level and extend the scan-based differential attack [4], [5] to test compression schemes which use X-masking logic as well as response compactors. For comparability with the previous work, the results are given in next section are for a collection of distinct distributions of KFFs over the scan chain structure of a generic test compression scheme.

IV. OUR APPROACH

For the sake of simplicity and ease of read, the following conventions are used in the rest of the work. Figure 6 shows the distribution of KFFs in the scan chains of a hardware design. As illustrated in the figure, each column of the scan design represents a ‘slice’. The flip-flops denoted by $F_{ij}$ (for example, $F_{11}$, $F_{12}$, $F_{23}$) are ordinary scan flip-flops, where $i$ stands for the scan-chain number and $j$ indicates the position of the respective FF in the scan-chain. Similarly, the flip-flops containing information on the target key byte are denoted by $K_{ij}$. Any slice containing at least one KFF is referred to as an ‘active slice’, while the others are called ‘non-active slices’. For instance in the figure, the slice containing KFFs $K_{42}$ and $K_{82}$ represents an active slice.

The basic approach taken in our modified differential scan attack is presented in Figure 7. This is explained in detail in Sections V and VI.

Through software simulations, we made further observations on the effects of various differences, other than the ones given in previous works [5], [4]. We implemented the differential scan-attack described in [5] on a C-implementation of AES-128, and observed that there are seven more one-byte differences ($0x4A, 0x69, 0x6B, 0x89, 0x92, 0xE7$ and $0xF5$) which result in having four unique Hamming distance values after one round of encryption. We further observed that using XOR difference $0xD1$ results in five unique Hamming differences which prove useful especially if the KFF distribution is unknown. We additionally observed that different XOR differences between inputs can lead to having the same unique Hamming distance values. The unique Hamming distance values corresponding to each XOR difference are given in Table I.

**While attacks on test compression schemes with X-masking are dependent on the distributions of KFFs in the active slices, attacks on test compression schemes with X-tolerant logic are dependent on the distributions of KFFs in the active scan-chains. This effect has not been considered in previous works, but forms the basis of our attack strategy in this paper.**

V. ATTACK ON TEST COMPRESSION SCHEMES IN THE PRESENCE OF X-MASKING

X-masking is the testing technique employed to mask certain scan chains which have a higher probability of occurrence of unknown states (X-states). This is achieved by inserting a mask layer between the test outputs and the test response compactor. The mask layer generally consists of series of 2-input AND gates, where one input comes from the scan chains and the other from the output of a mask decoder fed with a pattern mask, as shown in Figure 8. Though X-masking is primarily intended as a prevention against unknown states (X-states) corrupting the test responses, we observed that they enhance the security features as well, acting as a countermeasure to differential scan-chain attacks.

For instance, if the entire round register is on a scan-chain which is masked, then a scan-attack is not possible. However, even a few round register flip-flops are on the unmasked scan-chains, a differential scan-attack is possible for that particular mask value.
A. Attacking X-Masking

The success of the previously explained attack highly depends on the amount of information one gets about the contents of the KFFs in test outputs. Therefore, although its main purpose is to prevent X-states from corrupting the test output, X-masking also can be beneficial for preventing differential scan attacks to some extent. The rest of this section analyses the impact of X-masking on differential scan attack.

There are two basic approaches to X-masking. One can either discard all information on a particular scan-chain or cleverly discard some particular bits in the scan chain design. These methods will be referred to as static and dynamic masking respectively, throughout the remainder of the work.

1) Attack on Static masking: Static masking can have a significant impact on differential scan attack especially for large designs, in which there are more scan chains than the number of KFFs in the design. For AES, the attack complexity is upper bounded by \(2^{32} \) as there can be at most 32 KFFs for a one-byte input difference. Therefore, recovering the encryption key of any AES design with more than 32 scan chains requires the same amount of effort. The main reason behind this is that, with static masking, an entire scan chain is excluded for the whole testing process. This leads to having a particular scan chain contributing to the test output with probability \( \frac{1}{2} \). Hence, assuming the worst case scenario in which the KFFs are being spread over 32 scan chains, the probability of all these scan chains contributing to the test output is \( \left( \frac{1}{2} \right)^{32} \). However we do not really need each and every KFF to contribute to the test output for AES, since the maximum possible output Hamming distance we can get is 24. The results given in Table III support this point as 1000 different test inputs are observed to be sufficient for attacking a design in which the KFFs are distributed over 16 scan chains.

The attack method that we used is composed of two main parts: data acquisition and analysis. Data acquisition phase of the attack can be summarized as follows:

- Pick a test input at random
- Switch to test mode and fill the scan chain structure with the test data generated by the test input.
- Switch to functional mode and input a random plaintext to AES.
- Let the AES encryption run for only one clock cycle.
- Switch to test mode again, and obtain the test output for that particular input.
- Repeat the above steps for the same test input and with plaintexts which only differ in their LSB (please note that there are 256 such plaintexts)

Although the data acquisition phase is similar to the previously proposed attacks, the analysis phase is slightly different. At the analysis phase, we propose to use only one attack strategy, since knowing the distributions of KFFs beforehand would be an unrealistic assumption.

The analysis phase of the attack can be briefly summarized as follows:

- Pair the test outputs so that their corresponding plaintext inputs have 0xD1 difference on their LSB, and 0x00 difference on all other bytes.
- XOR each pair together to get the observed Hamming distance \( HD_{\text{Obs}} \).
- If \( HD_{\text{Obs}} \) is odd and greater than 9, then assume \( HD_{\text{Real}} = 23 \) and XOR the corresponding precomputed input values with the plaintexts to get possible keys. If \( HD_{\text{Obs}} \) is odd and greater than 5, then assume \( HD_{\text{Real}} \) is either 9 or 23 and XOR the corresponding precomputed input values with the plaintexts to get possible keys. Proceed in a similar fashion if \( HD_{\text{Obs}} \) is even.
- If no key candidate stands out from others, repeat the acquisition and analysis phases for a different test input, which in turn leads to another mask value.

Proceeding this way, the expected number of distinct test inputs required to have a successful attack is \( 2^{\min(\# \text{of scan chains}, 32)} \), where an attack is treated as successful only if the top candidate for the key is the correct one. Table III presents the success rates that we get with different distributions of KFFs which are given in Table II. We have attacked designs with distributions ranging from one KFF on each slice (the distribution is referred to as number 1 in Table II), to five KFFs on one slice and one KFF on the remaining 27 slices (the distribution is referred to as number 12 in the table). To have comparability between the attacks on X-tolerant logic and X-masking, we used the same distributions of KFFs for both attacks.

2) Attack on Dynamic Masking: In dynamic masking, parts of a particular scan chain are discarded by a dynamically generated mask updated at each clock or after some number of clock cycles. As shown in Figure 8, there may be an optional decoder clock to set the mask frequency matching with the test clock or a separate mask clock. For the sake of simplicity, let's say the mask is updated at each clock cycle. In this case, the attack complexity will depend on the number of KFFs in the design since at each clock the probability for a particular KFF to be contributing to the XOR compactor is \( \frac{1}{2} \). Hence, even if there are multiple KFFs on the same scan chain, they have to be unmasked at the appropriate clock cycle which leads
to having an attack complexity independent of the number of scan chains unlike the case in static masking. Therefore, ideally, a successful attack is expected to require \(2^{#\text{ of KFFs}}\) which would be \(2^{32}\) for AES. However, this number may be lower in practice similar to the case in static masking. But so far we could not observe such behaviour with our simulations at the time of writing this work. On the other hand, if the mask is changing only after some clock cycles and if this number is fixed for the design, the attack complexity will definitely be affected in favour of the attacker. In that case, the maximum attack complexity will be the same but the success rate can be better depending on the distribution of KFFs over the scan design.

VI. ATTACK ON TEST COMPRESSION SCHEMES IN THE PRESENCE OF X-TOLERANT LOGIC

In X-tolerant logic, at the input side, there are multiplexers (MUXs) to enable testing of multiple scan-chains using reduced number of scan inputs. At the output side, there is an XOR network to connect multiple scan chains to fewer scan outputs. This way compression is achieved without compromising on testability. The output side XOR compactor network is also known as the unload compressor which provides an X-tolerant and low area overhead design. This helps in diagnosing high volume of scan pattern failures which can be observed on the tester [9].

The details of the X-tolerant unload compressor is provided in [10]. It is based on an algorithm derived from Steiner systems, and provides a good balance between scan compression, X-tolerance and silicon area. To prevent aliasing, which is the cancellation of simultaneous faults on two or more chains, each sub-chain connects to a unique subset of the output. In some cases, the outputs are computed by XORing disjoint subsets of scan outputs (see Figure 9).

Since the X-tolerant XOR compressor has a different structure than an XOR-tree, previously described attack is not directly applicable to this compressor. The Hamming distances vary depending on the structure of the XOR network, therefore having a proprietary X-tolerant logic provides some security through obscurity as long as the structure of the XOR connections is not known. However, this compressor also leaks some information on the Hamming distance (HD) between outputs if it consists of linear operations.

For our scan attack, we used two AES-128 designs (a 16 and a 20 scan-chain design) with three versions of the unload compressor given in [10] with compression ratios of 16:2, 16:4 and 20:2 (where in the notation i:j, i denotes the number of scan chains while j denotes the number of compressor outputs). Our simulations show that the unload compressor outputs are evenly distributed among the output set. Therefore observing \(HD_{\text{Obs}}\) non-zero output differences on the unload compressor output means that the real Hamming distance \(HD_{\text{Real}} \geq HD_{\text{Obs}}\).

Hence, whenever \(HD_{\text{Obs}} \geq 22\) we can make a key guess, as this can be the case only for four particular input pairs (see Figure 5). Therefore we can make key guesses for cases \(HD_{\text{Real}} \in \{22, 23, 24\}\) in the same way explained in Section V-A1.

We have run simulations for different distributions of KFFs over the slices of the design, and observed that the attack is more successful for some specific distributions. Table II presents the distributions that we have chosen to attack. The output side XOR compactor network is also known as the unload compressor which provides an X-tolerant and low area overhead design. This helps in diagnosing high volume of scan pattern failures which can be observed on the tester [9].

The details of the X-tolerant unload compressor is provided in [10]. It is based on an algorithm derived from Steiner systems, and provides a good balance between scan compression, X-tolerance and silicon area. To prevent aliasing, which is the cancellation of simultaneous faults on two or more chains, each sub-chain connects to a unique subset of the output. In some cases, the outputs are computed by XORing disjoint subsets of scan outputs (see Figure 9).

Fig. 9. Example of an X-tolerant unload compressor.

TABLE II
DISTRIBUTIONS USED FOR VERIFYING THE ATTACK ON A GENERIC TEST COMPRESSION SCHEME

<table>
<thead>
<tr>
<th>Slice Dist. #</th>
<th>Active Slices</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>{1, \ldots, 1}</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
<td>{2, 1, \ldots, 1}</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>{3, 1, \ldots, 1}</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>{4, 1, \ldots, 1}</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>{5, 1, \ldots, 1}</td>
</tr>
<tr>
<td>6</td>
<td>27</td>
<td>{6, 1, \ldots, 1}</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
<td>{7, 1, \ldots, 1}</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>{8, 1, \ldots, 1}</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
<td>{9, 1, \ldots, 1}</td>
</tr>
<tr>
<td>10</td>
<td>23</td>
<td>{10, 1, \ldots, 1}</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>{11, 1, \ldots, 1}</td>
</tr>
<tr>
<td>12</td>
<td>21</td>
<td>{12, 1, \ldots, 1}</td>
</tr>
</tbody>
</table>

TABLE III
SUCCESS RATE OF THE ATTACK ON STATIC MASKING FOR THE PREVIOUSLY USED DISTRIBUTIONS.

<table>
<thead>
<tr>
<th>Slice Dist. #</th>
<th>Active Scan Chains</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>99.01%</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>95.05%</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>94.06%</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>88.12%</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>92.08%</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>85.15%</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>72.28%</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>86.14%</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>90.10%</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>85.15%</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td>75.25%</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>80.20%</td>
</tr>
</tbody>
</table>
TABLE IV
SUCCESS RATE OF THE ATTACK ON X-TOLERANT LOGIC FOR DIFFERENT COMPRESSORS

<table>
<thead>
<tr>
<th>Slice Dist. #</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 : 2</td>
</tr>
<tr>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>49.11%</td>
</tr>
<tr>
<td>3</td>
<td>49.36%</td>
</tr>
<tr>
<td>4</td>
<td>49.27%</td>
</tr>
<tr>
<td>5</td>
<td>48.12%</td>
</tr>
<tr>
<td>6</td>
<td>48.12%</td>
</tr>
<tr>
<td>7</td>
<td>27.20%</td>
</tr>
<tr>
<td>8</td>
<td>26.69%</td>
</tr>
<tr>
<td>9</td>
<td>22.33%</td>
</tr>
<tr>
<td>10</td>
<td>16.00%</td>
</tr>
<tr>
<td>11</td>
<td>15.10%</td>
</tr>
<tr>
<td>12</td>
<td>9.80%</td>
</tr>
</tbody>
</table>

designs with one scan output are not presented in this section as the unload compressor in this case is simply an XOR-tree. Here, each scan chain output is XORRed together to produce the scan output, which leads to successfully recovering the key using the attack described in Section III-C, regardless of the number of active slices.

Table IV provides the statistical results to show how successful our attack is. We run the attack 10000 times for each predefined distribution of KFFs to get reliable results and whenever the correct key is included in the guessed key set, which is at most of size 8, we treat the attack as successful. This is due to the fact that there are only four possible pairs of inputs which, after 1 round of AES encryption, transform into output vectors with Hamming distances 22, 23, or 24.

VII. COUNTERMEASURES

Various countermeasures have been proposed in the literature to protect against scan-based side-channel attacks. However, some of them do not provide protection against differential scan-attacks, while others are suitable only when a change to the scan-chain DFT is possible (which may not be the case in some scenarios, for instance in SoC integration). Hence, we propose here some countermeasures which not only protect against the differential scan attack mode, but also only involve change to the external test compression logic (not to the internal scan-chain DFT infrastructure).

A. Previous scan-attack countermeasures

Scan-chain scrambling [11] and insertion of inverters in the scan-tree structure [12] have been proposed as countermeasures against scan-based side-channel attacks. In scan-chain scrambling, an LFSR makes a pseudo-random selection of scan chains to be loaded a time, whereas in the case of inverted scan flip-flops, inverters are placed before some of the scan flip-flops whose locations are kept secret and only known to the designer and the tester. However, since the scrambled scan-chain structure and the locations of the inserted inverters in the scan path are fixed, they do not provide protection against differential scan-based attacks.

B. Proposed countermeasures

1) Authentication before Scan Testing: An authentication mechanism allowing only eligible testers to have an access to the test infrastructure may succeed in protecting against attackers. The entity authentication of the tester to the circuit can be managed through a public-key infrastructure or symmetric-key techniques with on-chip key storage and a corresponding locking/unlocking mechanism. Similar authentication schemes employing a secure version of the IEEE 1500 Test Wrapper haven been proposed in the literature [13], [14]. While a light-weight block-cipher (requiring an on-chip key storage) is used as the security interface for the test wrapper in [13], a Physically Unclonable Function (PUF)-based approach was taken in [14] which avoids key storage and has less implementation overhead.

2) New Noise Injector Countermeasure: We suggest to add random noise at the response compactor outputs for obscuring the test outputs. In this case, only the eligible tester should have the knowledge of the scan chains (with injected noise) which needs to be ignored. Using another test pattern or sustained test vector approach to overcome the loss in observability caused by the noise injection, high testability can be provided without compromising on security. Hence, this can act as a suitable countermeasure against differential scan-attacks. However, with the growing complexity of circuits, the communication of this random mask information to the testers needs to be addressed properly.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we present a novel approach at performing scan-based differential attacks on test compression structures. X-masking and X-tolerant logic present in most industrial DFT compression schemes are investigated for their security features. Different key-dependent flip-flop distributions are considered for a realistic attack scenario. A discussion on existing scan-attack countermeasures with proposals for more effective ones are also presented. This paper demonstrated our scan attack on 128-bit AES, however, the same principle is applicable for higher key lengths.

We plan to extend the scope of this work with more detailed security evaluations of dynamic masking and consider time compaction with Multiple Input Signature Registers (MISR)s in the future. The noise injection countermeasure will also be implemented in hardware to investigate its effect on testing.

ACKNOWLEDGEMENT

This work was supported in part by the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. Amitabh Das received the Erasmus Mundus External Cooperation Window (ECEMW) Lot 15 fellowship for India, till April 2012, when part of the work was performed.
REFERENCES