2014 19th IEEE European Test Symposium (ETS)

Test-Mode-Only Scan Attack Using the Boundary Scan Chain

Sk Subidh Ali, Ozgur Sinanoglu
New York University Abu Dhabi (NYUAD)

Ramesh Karri
Polytechnic Institute of New York University

***Abstract*—Boundary-scan is a very popular technology with wide applications in product life cycle that ranges from product design, prototype debugging, production to field service. However, when it comes to securing a product such as smart card, RFID tag, set-top-box, etc., the technology can be targeted by an attacker to reveal the secret information of the chip. In this paper, for the first time, we will show that the boundary scan chain can be used to bypass the mode-reset countermeasure, which is used to thwart all the scan attacks that rely on switching between the normal mode and the test mode of the chip. We propose two attacks on the AES core. The first attack uses the boundary scan chain to apply input plaintexts to the first round of AES, whereas the second attack targets the final round by applying the inputs through the internal scan chain(s) and the round output is captured in the boundary scan chain. The attacks not only bypass the mode-reset countermeasure but also circumvent the affect of stimulus decompressor (first attack) or the response compactor (second attack). Both attacks retrieve the 128-bit secret key within one minute of execution.**

I. Introduction

Scan is a widely accepted Design for testability (DfT) methodology that provides high fault coverage. It increases the controllability and observability of a sequential circuit by converting flip-flops to fully accessible scan cells, which are connected into scan chain(s). However, during the test of a secure chip, the scan based DfT technique may open a back door to an attacker and may be misused to leak the secret information of the chip. Such vulnerability of scan based test infrastructure was shown for the first time in [1], where the secret key of DES chip was revealed using the scan architecture. It was shown that an attacker, who has access to the test mode of the circuit, can run the cipher for one round in normal mode, and then by switching to the test mode, can retrieve certain information about the round output. Then by applying differential properties of the cipher, the attacker can reveal the secret key.

Subsequently, it was shown that almost all the ciphers such as AES [2], RSA [3], ECC [4] etc., are vulnerable to scan attacks. The security of these ciphers lies in the complete execution of the cipher algorithm. However, if the intermediate execution results are exposed, the security of the cipher is compromised. In the presence of scan architecture, the attacker has the leverage to run the crypto-chip in either mode and efficiently reduce the cipher operation to a single-round, and thus, insecure execution.

There are many countermeasures proposed in the literature, with each having its own strength and weaknesses. These countermeasures can be divided into two categories. One is based on the access control, where the authenticated users have access to the JTAG interface [5], [6]. The others are based on obfuscating the captured intermediate results. In this case the tester, who has the know-how of the internal architecture of the design, can recover the intermediate results to test for possible faults in the chip [7]. There is a third type of countermeasure, which resets all the scan cells whenever there is a switch from normal mode to the test mode [8]. The main advantage of this countermeasure is that it requires almost negligible area overhead. This mode-reset countermeasure was considered to be immune to all scan attacks. However, the recent test-mode-only attack [9], [10] shows that only the test mode is sufficient to successfully recover the secret key of 128-bit AES implementation, obviating the need to switch modes. This attack uses scan chains in test mode to feed the input to the AES core. It first determines the mapping between scan cells and the AES inputs. Then by using the Hamming distance signature table, it recovers the individual key bytes. However, the attack considers basic scan architecture. In the presence of stimulus decompressor and compactor, the attack will not work as is.

In this work we will present a new test-mode-only scan attack using the boundary scan chain architecture [11] of an iterative AES chip, which overcomes the mode-reset countermeasure. In the proposed test-mode-only attack, we target the round counter. In the previous test-mode-only attack [9] it was assumed that the default value of the round counter would choose the first round, which may not always be true. We assume that the flip-flops associated with the round counter are also part of the internal scan chain(s). We identify and control these flip-flops to attack the desired round. We propose two attacks, one in which we use the boundary scan cells to apply the desired input to the first round of AES. The intermediate results are captured and read out through the internal scan chain. The advantage of this attack is that the stimulus decompressor can be bypassed. In the second attack, we target the last round of AES, where the inputs are applied through the internal scan chain(s) and the intermediate results are captured in the boundary scan cells. Both attacks recover the 128-bit AES key within one minute.

978-1-4799-3415-7/14/$31.00 ©2014 IEEE

II. Preliminaries

A. Advanced Encryption Standard

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 *S*B, *S*R, *MC* and *A*RK respectively. Figure 1 shows the structure of the first round of AES which contains an extra key XORing at the beginning. More details about the AES cipher can be found in [12], [13].

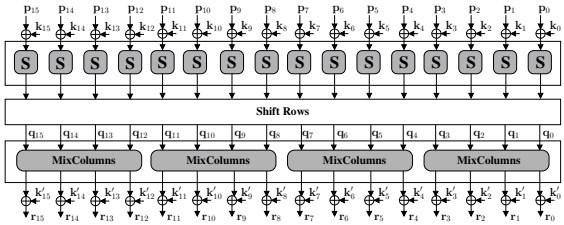

Fig. 1. First round of AES: p_i is the plaintext byte, k_i is the initial key byte, q_i is the SR output byte, k_i is the round key byte, and r_i is the round output byte.

B. Boundary Scan Chain Architecture

IEEE Standard 1149.1 [11] defines the test access port and boundary scan architecture, which is shown in Figure 2. It consists of two major components, TAP (Test Access Port) and the Boundary Scan register. Each primary input and primary output port is supplemented with a special memory element, referred to as boundary scan cell. During the test mode, the test vectors are shifted in the boundary scan cells while in the capture operation, these boundary scan cells drive the primary inputs and sample the output logic that normally drives the primary outputs.

The TAP controller is the control logic that can connect one of the five registers (boundary scan register, internal registers, instruction register, identification register, bypass register) between TDI and TDO. In order to configure the boundary scan architecture, the proper instruction is loaded in the instruction register. Four mandatory instructions (*Extest, Bypass, Sample, Preload*) and six optional instructions (*Intest, Idcode, Usercode, Runbist, Clamp, HighZ*) are specified in the IEEE 1149.1 standard. To test the device, test vectors are shifted in the boundary scan chains as well as in the internal scan chains. The literature shows that the internal scan chain(s) can be used as a backdoor to leak the secret information of the chip. We will show that there are countermeasures in place to thwart basic scan attacks, and that the boundary scan chains can also be used along with the internal scan chain(s) to perform a more lethal form of the attack, circumventing such countermeasures.

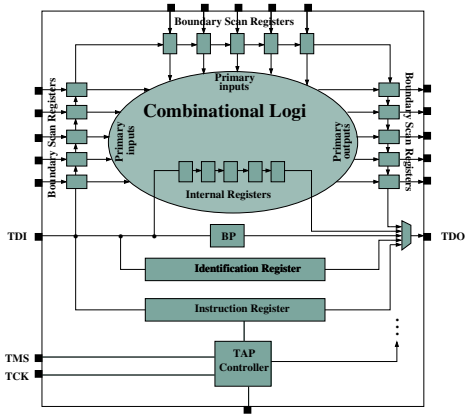

Fig. 2. Boundary scan architecture

C. Mode-reset Countermeasure

The mode-reset countermeasure targets the fundamental weakness of all the existing scan attacks, i.e., that they all require switching from the normal mode to the test mode; all these scan attacks rely on the assumption that the intermediate computation results of the cipher remain intact in the scan chain(s). Mode-reset countermeasure resets all the scan cells on a normal to test mode switch. Hence, the intermediate results, which were captured in the normal mode, can not be retrieved in the test mode.

D. Circumventing Mode-reset Countermeasure

In the presence of the mode-reset countermeasure, the only option that remains available to the attacker is to use only the test mode. Recent work [9] shows that by using only the test mode, one can attack a cipher such as AES. In this attack, the attacker uses only the internal scan chain(s). The use of only the test mode and the internal scan chain(s) may be difficult in the presence of a stimulus decompressor and a response compactor in the scan architecture however. Especially, the stimulus decompressor challenge is unique to the test-mode-only attack [9]; all the other existing scan attacks rely on the normal mode and primary inputs, and therefore bypass the stimulus decompressor naturally. Yet they fail the mode-reset countermeasure. *In this work, we show how the boundary scan architecture can be misused to circumvent the decompressor or compactor, and create a powerful test-mode-only attack.*

E. Challenges For Test-Mode-Only Attack Using Boundary Scan Chains

For the proposed boundary scan attack to be successful, the attacker needs the following two pieces of information. First, the mapping between the primary inputs and the boundary scan

cells is needed, so that the proper plaintexts can be applied to the inputs. Second, the flip-flops associated with the round counter needs to be known, so that the round counter can be controlled to the desired round. These are challenges unique to the proposed boundary scan attack.

F. Assumptions

The following are the assumptions we make to perform the attack.

- The AES core is a 128 bit iterative design and has 128 primary inputs and 128 primary outputs. The round keys are generated prior to the encryption and loaded in memory.
- The attacker has access to the JTAG inputs and can configure it as per his requirements.
- The boundary scan architecture of the device includes *INTEST* instruction.
- The attacker has access to the internal scan chain(s) as well as the boundary scan chain.

III. Controlling the Round Counter Value in Test Mode

We assume that the AES round counter is implemented as down counter. However, the technique we are going to propose will also be applicable to an up counter. The iterative AES operations are shown in Figure 3 (AES with 128-bit key). We can observe that it has ten rounds, among which the first nine rounds are iterative. In each of the first nine rounds, the same round operation is repeated with different round keys. The round outputs are loaded in the 128-bit round register for the next round operation. In the tenth round, the content of the round register is transfered to the primary outputs as a ciphertext.

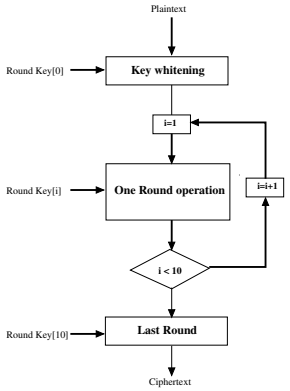

Fig. 3. Iterative AES cipher operations with 128-bit key

For AES with down counter, the initial value is set to ten (1010) to select the first round. Then in each round operation, the counter value is decremented by one. At the end of the final round, when the counter value becomes zero, the round register content is transferred to the primary outputs; in the test mode the boundary scan cells associated with the primary outputs will be loaded with the round register value.

As a first step, we apply all zeros to the internal scan chain(s) and capture the response. The round counter, which is a part of the scan chain(s), will also become zero. This implies that during the capture operation, the content of the round register will be transferred to the primary outputs, thus loading the associated boundary scan cells. In order to distinguish these boundary scan cells from the others, we can apply a test vector with all ones to the boundary scan cells while applying all zeros to the internal scan chain(s). After the capture operation, only the boundary scan cells that are associated with the primary outputs will become zero, signaling the transfer operation.

In the second step, we apply a test vector V_i, with only the i-th bit set to one and all the other bits are zeros, and capture the response. The goal is to identify the four scan cells corresponding to the round counter. If the i-th bit is in the counter, the round register content will not be transferred to the primary outputs. Therefore, by observing the changes in the boundary scan cells associated with the primary outputs (which was observed in the first step), we can determine whether the one-hot test vector has affected the round counter. By applying all possible V_i, we can determine the four scan cells associated with the round counter. However, the order of these scan cells in the round counter remains unknown.

To know the order of the four scan cells, we apply four one-hot values $(0001, 0010, 0100, 1000)$, with each one affecting one of the counter bits. The transfer to the boundary scan cells can take place in either the second, third, fifth or the ninth cycle with one of the four values (1, 2, 4, or 8) in the round counter. To identify exactly when, we perform consecutive capture operations in each of the four cases and observe the boundary scan register to identify the round counter bits; after applying one of the one-hot values in the round counter, we apply c captures and read out the boundary scan register. We set c to 1, 2, .., 9. This way, we know when the transfer to the primary outputs occurs, thus identifying the counter value. Upon repeating this operation for all four one-hot values of the counter, the round counter bits are identified.

IV. Attack 1: Scan Attack Using the Boundary Scan Chain as Input

In this section we propose an attack where the boundary scan chain feeds the first round of AES and the round outputs are retrieved through the internal scan chain(s) as shown in Figure 4. The part of the design marked in red shows the data flow in the attack. To choose the first round, a test vector is loaded in the internal scan chain(s) to load the round counter with the value ten (1010).

The advantage of this attack is that it circumvents the stimulus decompressor, which may possibly prevent the attacker from applying the desired input to the AES.

The following two steps are taken in the proposed attack.

A. Boundary Scan Cells vs AES Input

In this part of the attack, we will determine the mapping between the boundary scan cells and the AES inputs. We set

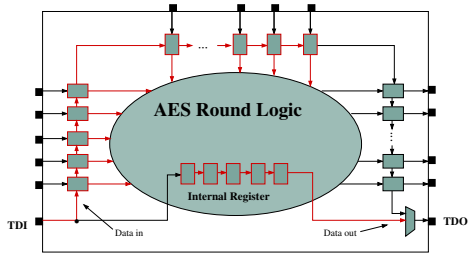

Fig. 4. Data flow: AES input controlled through boundary scan chain and output observed through internal scan chain(s) (shown in red)

the device in test mode to select the boundary scan cells. Proper test vectors are applied in the boundary scan chain while at the same time the internal scan chain(s) is/are loaded with the test vector that sets the round counter to ten. The captured responses are loaded in the internal scan chain(s) (the flip-flops associated with the AES core), and then observed.

In order to determine the mapping between boundary scan cells and the AES inputs, we extract the response of AES to a single input bit-flip. Depending on the design of the chip, there could be many other boundary scan cells in addition to those associated with the AES inputs and outputs. As a first step, we need to determine which boundary scan cells are associated with the AES inputs. We apply a pair of test vectors (V, V_i), where V is the test vector with all zeros in the boundary scan cells, and V_i has a single one-bit in the i-th position, to the boundary scan chain to create a one bit difference in i-th scan cell, and capture the response. If the i-th scan cell corresponds to any one of the AES inputs, we will find a Hamming distance in the captured response pair ranging from 5 to 24. Otherwise, the responses will be identical. It may be noted that the boundary scan cells which are associated with the primary outputs, will have no affect on the captured response. Therefore, we vary i through all the boundary scan cells, and by observing the captured responses, we determine the 128 boundary scan cells associated with the AES inputs.

Once we have this information, we try to determine the mapping between the boundary scan cells and the AES input bytes; i.e., the problem is to classify the 128 boundary scan cells into individual AES input bytes. In this case we consider two test vector pairs (V, V_i) and (V_{ij}, V_j), where V_{ij} is generated by flipping the j-th bit in V_i. Therefore, now we have two different pairs of test vectors with the same input difference. So, if the two bit-flips (i and j) are in the same byte, the output differences should be identical [9]. By fixing the value of i and varying j all through boundary scan cells, we can identify the other seven scan cells corresponding to the same input byte as i. Subsequently, by varying i, we can divide the 128 boundary scan cells into 16 input bytes.

Therefore, at the end of this process, we can apply values to any of the sixteen input bytes of AES using the boundary scan chain. However, the order of the scan cells corresponding to a byte remains unknown. Further, the order of the sixteen bytes is also not known. In the next section we will show how we can apply a Hamming distance analysis technique to determine the secret key without the knowledge of the order of the bits in a byte or the order of the sixteen bytes.

B. Key Recovery

For the key recovery part, we choose the boundary scan cells associated with any particular input byte B. We apply all possible 256 different test vectors in byte B, while values in the other bytes are retained constant. We capture the responses corresponding to all the 256 test vectors. We recall from the basic properties of the AES round operation that given a one-bit input difference, the output difference can have a Hamming distance of 5 to 24, as shown in Figure 5. Among these Hamming distances, only a few (9, 12, 23, and 24) appear exactly once. In other words, they correspond to a unique S-box input; the Hamming distance will occur only for one S-box input pair. We refer these Hamming distances as *unique Hamming distances*. Depending on the bit position in a byte, where the one-bit input difference is applied, there could be two to four possible unique Hamming distances as shown in Table I.

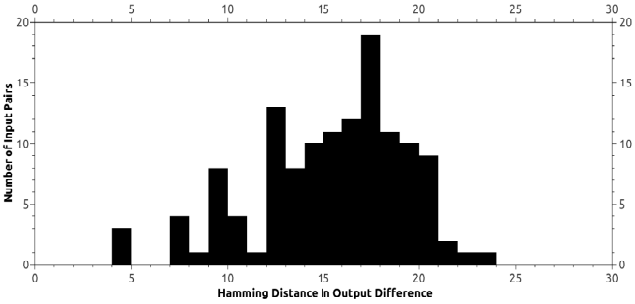

Fig. 5. Distribution of Hamming distances corresponding to one-bit difference in the first bit

Now we consider a bit b of byte B and determine all possible 128 test vector pairs, so that each pair produces a one-bit difference at bit b. We calculate the Hamming distance from the captured response pairs corresponding to the applied test vector pairs. Depending on the position of b in byte B, we may get two to four unique Hamming distances. Using these Hamming distances, we can uniquely identify the position of bit b, except for the positions 4 and 5 (these two positions will produce the same unique Hamming distances 9 and 24 as shown in Table I). For example, if we get four unique Hamming distances of $9, 12, 23$, and 24, we can directly say that the applied bit position is the least significant bit. We vary b across all the bits of B, and determine the bit positions based on the unique Hamming distances. Once we determine the position of the eight bits, we can determine the exact input byte. If we know the bit position, we can also determine the unique inputs. For example, if the bit position is the least significant bit, the unique inputs corresponding to the four unique Hamming distances are either $(226, 242, 122, 130)$ or $(227, 243, 123, 131)$. Each value of b will therefore produce

two key byte values. Intersection of all the eight pairs of key byte values for the eight positions of b, will uniquely determine the key byte.

We can apply the same technique across all the bytes and determine the sixteen key byte values. It may be noted that applying one-bit difference to a round input will affect four bytes at the output, which correspond to a word of AES. So, in the process of retrieving the key byte value, we also determine the four words of AES. The final search space of the key is the permutation of these four words, as well as the permutation of the four bytes in a word, which is $4!^5 = 2^{22.92}$. One can easily brute-force these keys within one minute using a desktop computer with $Intel^{TM}\ Core\ i5$ processor.

TABLE I
HAMMING DISTANCES WITH UNIQUE S-BOX INPUTS CORRESPONDING TO INPUT BIT-DIFFERENCE POSITIONS

7	6	5	4	3	2	1	0
8, 24	11, 24	9, 24	9, 24	11, 23	5, 12, 24	12, 23, 24	9, 12, 23, 24

V. ATTACK 2: SCAN ATTACK USING THE BOUNDARY SCAN CHAIN AS OUTPUT

In this attack we target the last round of AES. Instead of applying test vectors using boundary scan chain, we target the internal scan chain(s). As shown in red in Figure 6, the AES round is getting its inputs through the scan cells associated with the round register, and the output is observed through the boundary scan cells corresponding to the primary outputs. The advantage of this attack is that a response compactor has no affect on the attack, as the intermediate results of the AES operations are retrieved through the boundary scan chain. Without loss of generality we assume that there is only one scan chain, which the attacker can fully control.

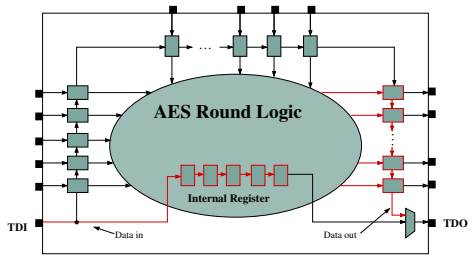

Fig. 6. Data flow: AES input controlled through the internal scan chain and output observed through the boundary scan chain (shown in red)

In this attack, we target the last round of AES. As explained earlier, when the counter value becomes zero, the round register content is transferred to the primary outputs. The major challenge here is to find the mapping between the round register scan cells and the primary outputs.

This attack is performed in three steps.

A. Determining the Scan Cells Associated With the Round Register

Only 128 scan cells in the entire scan chain belong to the round register. In this part of the attack, we are not going to use the TDO port. Therefore, we apply a different technique than what is already described in the previous section. We still apply test vector V_i as explained in the previous step. However, each value of i will try to stimulate one of the scan cells in the scan chain excluding the four scan cells corresponding to the round counter. The round counter should always be at zero, in order to ensure that during the capture operation, the value stored in the round register will be transferred to the primary outputs. This way, if the i-th scan cell belongs to the round register, the effect will be visible at the boundary scan cells associated with the primary outputs. Therefore, among all the test vectors applied in this step, only 128 will distinguish the round register scan cells from the remaining scan cells. The number of test vectors we need to apply is four less than the total number of scan cells in the scan chain.

B. Determining the Mapping Between Boundary Scan Cells and AES Outputs

In this part of the attack we load values in both the round counter and the round register by using the internal scan chain. This way, we can (1) move the AES execution to the last round, and (2) we can apply at least one known plaintext with all zeros in this round. Further, we can also transfer the last round output to the primary outputs and retrieve it through the boundary scan chain. For this, we need to apply a test vector that loads the round counter with the value 1. Therefore, we shift in the desired test vector and perform the capture operation twice, exercising both the last round and the transfer operations back to back.

For the last round, determining the mapping between scan cells vs AES input (which is the same as determining the output) byte is much more easier than the first round [9]. We apply a test vector pair (V, V_i) with one bit difference in the i-th scan cell of the round register, and capture the responses twice. As per the avalanche properties of the AES S-box, flipping one bit at the input should affect half of the output bits. Let us consider the two output differences D_i and D_j corresponding to two test vector pairs (V, V_i) and (V_{ij}, V_j). If they have any common bit-flips, we can conclude that the bits i and j belong to the same byte. Initially, we can apply 128 test vector pairs to create a one-bit difference in all possible 128 round register scan cells. We check for common bit-flips and group bits together in bytes. This way we can classify 128 scan cells into 16 bytes, determining the mapping.

C. Key Recovery in the Last Round

Although we have identified the output bits corresponding to a byte, the order within the byte is still unknown. However, we can apply all zeros or all ones irrespective of the order. We apply two test vectors – V with all zeros, and $\overline{V}$ with all ones – into the round register scan cells. Both of these test vectors should make sure that the round counter is loaded with value

1. We shift in each of these two test vectors, and perform the capture operation twice. We retrieve the responses through the boundary scan cells.

In the last round, we have eight S-box operations, with each receiving a value of 0 or 255, and producing a value of 99 or 22, respectively. These values are being XORed with the round key bytes to generate the ciphertext bytes. Let us denote the first key byte value as K_0. This value is being XORed with 99 and 22, respectively, to produce the corresponding output byte values. Therefore, at the output, we have $C_0 = K_0 \oplus 99$ and $C'_0 = K_0 \oplus 22$. However, we do not know the order of the bits in a byte. Therefore, instead of getting the, exact value, we will get some permutation of the byte, which can be brute-forced. On average, the byte will have 4 zeros and 4 ones; therefore, the expected number of permutations is $\frac{8!}{4!\cdot 4!} = 70$, which is a small number. It is expected that only one out of the 70 permutations will satisfy the above condition. The corresponding permutation will directly determine the key byte value. We apply the same technique across all the bytes and determine all the sixteen key byte values. This way, we can identify the key byte values as well as the mapping between scan cells and the AES outputs. However, the order of the key bytes still remains unknown. The final key search space will be around $16! \approx 2^{44.25}$, which is a large number. By using a desktop machine with $Intel^{TM}\ Core\ i5$ processor, one can brute-force these key hypotheses in around one day.

We can further reduce the search space of the attack by considering the penultimate round of AES, as we know the scan cells corresponding to the round counter. The round counter value corresponding to the penultimate round is 0010. We apply a test vector pair with round counter value set to 0010, and produce one bit difference in the round register. We shift in each of the test vectors and perform three capture operations. The difference in one byte at the input of the penultimate round will spread to four bytes after the final round. These four bytes correspond to a word of AES round. Therefore, by using one pair of test vectors with difference in one byte, we can identify one AES word. We need to apply four such test vector pairs to identify all the four words of the AES round. However, we do not know the order of the four words. We thus have to try all possible permutations of the words. The final search space will reduce to $4!^5 \approx 2^{22.92}$. One can brute-force these key hypotheses within one minute by using the existing desktop machines.

VI. Conclusions

In this paper we show that only the test mode can be used to attack an AES chip, bypassing the mode-reset countermeasure. However, test units such as stimulus decompressor and the response compactor that reside on the scan path may pose a challenge on the success of the scan attack in retrieving the secret key. While the challenges in scan attack associated with a response compactor has been well-understood [7], the interference of the stimulus decompressor in scan attack is unique to test-mode-only attacks. In order to circumvent these units, we propose two attacks in this paper that rely on the use of the boundary scan architecture, and show that the mode-reset countermeasure can be successfully bypassed.

References

[1] B. Yang, K. Wu, and R. Karri, "Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard," in *Proceedings of the International Test Conference, ITC 2004*. IEEE, 2004, pp. 339–344.

[2] B. Yang, K. Wu, and R. Karri, "Secure scan: a design-for-test architecture for crypto chips," in *Proceedings of the 42nd Design Automation Conference, DAC 2005*. ACM, 2005, pp. 135–140.

[3] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, "Scan-Based Side-Channel Attack against RSA Cryptosystems Using Scan Signatures," *IEICE Transactions*, vol. 93-A, no. 12, pp. 2481–2489, 2010.

[4] J. DaRolt, A. Das, G. D. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "A scan-based attack on Elliptic Curve Cryptosystems in presence of industrial Design-for-Testability structures," in *Proceedings of the Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2012*. IEEE Computer Society, 2012, pp. 43–48.

[5] K. Rosenfeld and R. Karri, "Attacks and defenses for jtag," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 36–47, 2010.

[6] R. F. Buskey and B. B. Frosik, in *ICPP Workshops*. IEEE Computer Society, pp. 405–414.

[7] J. DaRolt, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, "Scan Attacks and Countermeasures in Presence of Scan Response Compactors," in *Proceedings of the European Test Symposium, ETS 2011*. IEEE Computer Society, 2011, pp. 19–24.

[8] D. Hely, F. Bancel, M.-L. Flottes, and B. Rouzeyre, "Test Control for Secure Scan Designs," in *Proceedings of the European Test Symposium, ETS 2005*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 190–195.

[9] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "New scan-based attack using only the test mode," in *Proceedings of the VLSI and System-on-Chip, VLSI-SOC 2013*. IEEE Computer Society, 2013, pp. 234–239.

[10] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Scan attack in presence of mode-reset countermeasure," in *Proceedings of the International On-Line Testing Symposium, IOLTS 2013*. IEEE Computer Society, 2013, pp. 230–231.

[11] "IEEE Standard Test Access Port and Boundary-Scan Architecture," *IEEE Std 1149.1-2001*, pp. i–200, 2001.

[12] "Specification for the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[13] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.