Secure Scan: A Design-for-Test Architecture for Crypto Chips

Bo Yang
ECE Department
Polytechnic University
Brooklyn, NY, 11201

Kaijie Wu
ECE Department
University of Illinois, Chicago
Chicago, IL, 60612

Ramesh Karri
ECE Department
Polytechnic University
Brooklyn, NY, 11201

ABSTRACT

Scan-based Design-for-Test (DFT) is a powerful testing scheme, but it can be used to retrieve the secrets stored in a crypto chip thus compromising its security. On one hand, sacrificing security for testability by using traditional scan-based DFT restricts its use in privacy sensitive applications. On the other hand, sacrificing testability for security by abandoning scan-based DFT hurts product quality. The security of a crypto chip comes from the small secret key stored in a few registers and the testability of a crypto chip comes from the data path and control path implementing the crypto algorithm. Based on this key observation, we propose a novel scan DFT architecture called secure scan that maintains the high test quality of traditional scan DFT without compromising the security. We used a hardware implementation of the Advanced Encryption Standard (AES) to show that the traditional Scan DFT scheme can compromise the secret key. We then showed that by using secure scan DFT, neither the secret key nor the testability of the AES implementation is compromised.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing and Fault-Tolerance

General Terms: Algorithms, Design, Security

Keywords

Scan-based DFT, Testability, Security, Crypto Hardware

1. INTRODUCTION

Crypto algorithms are being implemented in hardware to meet high throughput requirements [1]. These crypto chips and associated systems have to be tested at fabrication and in-field. Scan-based DFT is the most popular DFT scheme for integrated circuit testing as it is simple and yields high fault coverage [2]. Furthermore, the scan chains can be connected to external five-pin JTAG interface during chip packaging to provide on-chip debug capability in-field [3]. On-chip debug capability eases the development and maintenance of software running on the chips and is very important for microprocessors. However, once these microprocessors are used in secure applications, the security becomes a very important issue, because information in the flip-flops in a scan chain can be accessed and analyzed. If the flip-flops in a scan chain contain a secret or information related to a secret, the secret can be discovered [4] [5]. Scan chains have been successfully used to retrieve secret Boxkeys in satellite TV receivers [6]. The microcontroller inside the receiver uses the Boxkey to decrypt the signals encrypted by program provider. The Boxkeys of several types of satellite TV receivers can be read out from JTAG interface. With a hacked Boxkey, users can watch TV program packages that they do not subscribe to.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2005, June 13–17, 2005, Anaheim, California, USA.
Copyright 2005 ACM 1-59593-058-2/05/0006…$5.00.

A straightforward way to thwart scan-based attacks is to leave scan chains unbound to prevent further access after validating the function of the chip [7]. However, unbound scan chains will compromise the maintenance and debug capabilities in-field. In [8], flip-flops that contain secret information are included in a special scan chain. This scan chain is only accessible by chip manufacturers but not by customers. In a standalone crypto chip almost all flip flops could contain information related to a secret. This architecture also limits maintenance and debug features to the crypto core in a system. In [9], a scan chain scrambling technique was proposed to provide both security and testability for crypto chips. This technique dynamically reorders the flip-flops in a scan chain to protect the secrets. However, statistical analysis of the information scanned out from chips can still determine the scan chain structure and the secret information. Furthermore, the area overhead incurred by dynamic reordering of scan chains is high. A primary alternative to scan-based DFT is built-in self-test (BIST) [10]. BIST is more secure because it does not require visible scan chains, but BIST incurs more overhead and yields less fault coverage when compared to scan-based DFT.

In this paper, we will show how scan chains can be used to discover a secret key from a hardware implementation of AES. We will then describe the secure scan DFT architecture. The proposed secure scan architecture can provide full testability to crypto circuits without compromising security with very low area overhead.

2. SCAN-BASED ATTACK ON AES

Since being announced as the private-key symmetric block cipher standard by NIST, AES has been widely implemented in hardware [1]. AES encrypts 128-bit data blocks under the control of a 128-bit user key. In both iterative and pipelined AES architectures, the one-round hardware shown in Figure 1 consuming 1 clock cycle is commonly used [1]. In the first clock cycle, the plaintext input is selected and the result after pre-round and round 1 is stored in Register R. In general AES hardware implementations, the temporary result of pre-round is not registered, because it is a simple bit-wise exclusive-or operation. In an iterative AES architecture, the output of Register R is the input to the next round operation and is fed back to the S-boxes through a multiplexer at point b. The AES round operation is repeated 10 times each time with a different

round key RK_i to generate the ciphertext. In a pipelined architecture, the one-round hardware is replicated 10 times. The output of Register R in the current round is applied to the S-box of the next round. The value of Register R in the tenth round is the ciphertext. Round keys are either pre-computed from user key and stored in the on-chip RAM or generated on-the-fly. Compared to the data path, the control logic for AES is very simple.

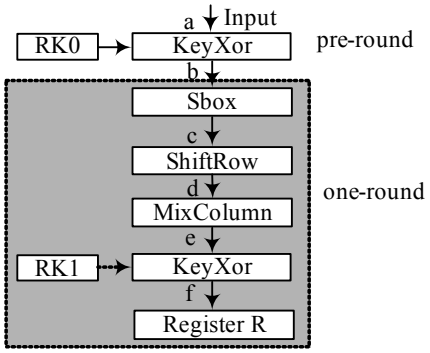

Figure 1: Round operations of AES encryption

Round keys or the user secret key are the targets of an attacker. If the user key or round key is stored in registers and the registers are included in scan chains, then it is easy to scan out the secret of the AES chip, which is similar to the attack on satellite TV receivers. In this section, we will show that even when round key registers (RK0 and RK1) are not included in the scan chain, by accessing the Register R through the scan chain, the secret user key can still be retrieved by analyzing the intermediate results. Since the attacker can access Register R, he can retrieve the intermediate ciphertext of every round. This essentially reduces the analysis of the AES implementation to one round.

One AES round uses Sbox, ShiftRow, MixColumn and KeyXor operations as shown in Figure 1. 128-bit input plaintext is arranged as a 4×4 matrix of bytes with $a_{i,j}(0 \le i,j \le 3)$ representing a byte. KeyXor operation is a bit-wise exclusive-or of 128-bit round key with the data:

$$b_{i,j} = a_{i,j} \oplus RK0_{i,j}(0 \le i,j \le 3) \qquad (1)$$

The non-linear substitution operation S-box substitutes each byte $b_{i,j}$ in the input with byte $c_{i,j}$ according to a one-byte substitution table (Sbox):

$$c_{i,j} = Sbox(b_{i,j}),(0 \le i,j \le 3) \qquad (2)$$

The ShiftRow operation cyclically left shifts the bytes in each row of the matrix. While bytes in the first row are not shifted, bytes in the second, third and fourth rows are cyclically shifted by one, two and three positions to the left, respectively.

$$\begin{aligned}
&(d_{0,0},d_{0,1},d_{0,2},d_{0,3}) = (c_{0,0},c_{0,1},c_{0,2},c_{0,3}) \\
&(d_{1,0},d_{1,1},d_{1,2},d_{1,3}) = (c_{1,1},c_{1,2},c_{1,3},c_{1,0}) \\
&(d_{2,0},d_{2,1},d_{2,2},d_{2,3}) = (c_{2,2},c_{2,3},c_{2,0},c_{2,1}) \\
&(d_{3,0},d_{3,1},d_{3,2},d_{3,3}) = (c_{3,3},c_{3,0},c_{3,1},c_{3,2})
\end{aligned} \qquad (3)$$

MixColumn operation transforms every four bytes in one column by multiplying it with a polynomial $c(x) = '03'x^3 + '01'x^2 + '01'x + '02'$ to generate four bytes in the corresponding output column.

$$(e_{0,j},e_{1,j},e_{2,j},e_{3,j}) = (d_{0,j},d_{1,j},d_{2,j},d_{3,j}) \otimes c(x),(0 \le j \le 3) \qquad (4)$$

Finally, KeyXor introduces the second round key RK1:

$$f_{i,j} = e_{i,j} \oplus RK1_{i,j},(0 \le i,j \le 3) \qquad (5)$$

Round key RK0 (which is the user key) and RK1 are involved before the intermediate result is stored into the register R.

In addition to the knowledge shown in Figure 1, the attacker needs to know the correspondence between the D flip-flops of the round register R and the bits in the scanned out bit-stream. Depending on different applications, a scan chain may contain several hundreds of bits and the sequence of these bits depends on the physical positions of the bits after the netlist of the design has been placed and routed. Such information is not known to an attacker. Hence an attacker should determine the structure of the scan chain and then retrieve the AES round keys.

2.1. Attack Step 1: Determine Scan Chain Structure

The intermediate ciphertext after the pre-round and first round is stored in the Register R. Although R is included in the scan chain, the attacker does not know which bits in the bit stream are from R. By switching the AES circuit between normal mode and test mode, this correspondence can be established.

If we change byte $a_{1,1}$ at point a in Figure 1, byte $b_{1,1}$ will change at point b, byte $c_{1,1}$ will change at point c (output of Sbox), and byte $d_{1,0}$ will change at point d (output of ShiftRow) and bytes $(e_{0,0},e_{1,0},e_{2,0},e_{3,0})$ will change at point e. Finally, the 32 flip flops corresponding to $(f_{0,0},f_{1,0},f_{2,0},f_{3,0})$ will change in Register R. Location of these 32 flop-flips of R in the scanned-out bit stream can be determined as follows:

1. Reset the chip and run it in normal mode for one clock cycle. The plaintext will be processed by pre-round (KeyXor with RK0) and round 1. The intermediate result is stored in Register R. Switch the chip to test mode and scan out the bit stream as pattern 1.
2. Repeat step 1 using a plaintext that is different from the first plaintext in one byte (say $a_{1,1}$). Save the scanned-out bit stream as pattern 2. Repeat this step and collect patterns 3, 4… 256.

The bits in the scanned out stream that have different values in Pattern 1 and 2 corresponds to the flip-flops in register R. By comparing pattern 3 with pattern 1, more such correspondences between bits in scanned out stream and flip flops in register R will be discovered. This process is repeated until all bits that correspond to the 32 flip flops in register R are determined. In the worst case this requires generation of all 256 patterns and performing 255 comparisons. However, due to the avalanche effect of cryptographic algorithms, one-bit change at the input to a round will translate into several bit changes at the output from the round [12], we need to apply very few patterns at $a_{1,1}$ to locate the 32 flip-flops of R in the bit stream. A simulation using the C implementation of AES shows that by applying 15 patterns in the worst case and 6 patterns on average, the 32 flip-flops of R can be located in the output bit stream.

At the end of step 1 an attacker can determine which 32 bits in the scanned-out bit stream correspond to flip-flops $(f_{0,0},f_{1,0},f_{2,0},f_{3,0})$. However, the attacker still does not have a one-to-one correspondence between bits in the bit stream and bits in $(f_{0,0},f_{1,0},f_{2,0},f_{3,0})$. Further he does not know the exact value of $(f_{0,0},f_{1,0},f_{2,0},f_{3,0})$.

2.2. Attack Step 2: Recovering Round Key RK0

In the architecture shown in Figure 1, pre-round and round 1 are performed in one cycle and two round keys, RK0 and RK1, are used. We will apply a chosen plaintext byte at $a_{1,1}$ and observe the corresponding word $(f_{0,0}, f_{1,0}, f_{2,0}, f_{3,0})$ in register R. We will then infer the value at $b_{1,1}$ and determine the round key byte $RK0_{1,1}$ as $b_{1,1} \oplus a_{1,1}$. We will determine all bytes in the round key $RK0$ by repeating this step.

Inferring the value $b_{1,1}$ from $(f_{0,0}, f_{1,0}, f_{2,0}, f_{3,0})$ entails two problems. First, the value at point f is determined by $b_{1,1}$ and $RK1$. Second, the exact value of $(f_{0,0}, f_{1,0}, f_{2,0}, f_{3,0})$ is unknown. As can be seen from Figure 1, the round key RK1 is exclusive or-ed with the output of MixColumn at point e to generate the output word $(f_{0,0}, f_{1,0}, f_{2,0}, f_{3,0})$ at point f. We can cancel the effects of RK1 by applying two selected values $b_{1,1}^1$ and $b_{1,1}^2$ to generate words $e^1 = (e_{0,0}^1, e_{1,0}^1, e_{2,0}^1, e_{3,0}^1)$ and $e^2 = (e_{0,0}^2, e_{1,0}^2, e_{2,0}^2, e_{3,0}^2)$ at point e and words $f^1 = (f_{0,0}^1, f_{1,0}^1, f_{2,0}^1, f_{3,0}^1)$ and $f^2 = (f_{0,0}^2, f_{1,0}^2, f_{2,0}^2, f_{3,0}^2)$ at point f. From equation (5), we have:

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

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

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

Observe that the number of 1s in $f^1 \oplus f^2$ is identical to the number of 1s in $e^1 \oplus e^2$. Furthermore, the number of 1s in $f^1 \oplus f^2$ is independent of the round key RK1 and is only determined by $b_{1,1}^1$ and $b_{1,1}^2$. An attacker can infer $b_{1,1}^1$ and $b_{1,1}^2$ by analyzing the number of 1s in $f^1 \oplus f^2$.

We propose to justify pairs of the form (2m, 2m+1), 0≤m≤127 at $b_{1,1}$ by applying $a_{1,1}^1 = 2t$, and $a_{1,1}^2 = 2t+1$, 0≤t≤127 at $a_{1,1}$. Since 2t and 2t+1 are different only in the least significant bits, $b_{1,1}^1 = 2t \oplus RK0$ and $b_{1,1}^2 = (2t+1) \oplus RK0$ will also be different only in the least significant bit result in the form of (2m, 2m+1) or (2m+1, 2m), 0≤m≤127.

If different pairs $(b_{1,1}^1, b_{1,1}^2)$ can generate the same number of 1s in $f^1 \oplus f^2$, the number of 1s in $f^1 \oplus f^2$ does not uniquely determine an input pair of $(b_{1,1}^1, b_{1,1}^2)$. In AES, when the inputs of the form of (2m, 2m+1) or (2m+1, 2m) are applied at point $b_{1,1}$, the number of 1s in $f^1 \oplus f^2$ is between 7 and 25. More importantly, if the number of 1s in $f^1 \oplus f^2$ is 9, 12, 23 or 24, it uniquely determines the input pair $(b_{1,1}^1, b_{1,1}^2)$ at $b_{1,1}$ as shown in Table 1. For example, if the number of 1s in $f^1 \oplus f^2$ is 9, the input $(b_{1,1}^1, b_{1,1}^2)$ at $b_{1,1}$ is either (226, 227) or (227, 226).

Table 1: The number of 1s in $f^1 \oplus f^2$ and the $(b_{1,1}^1, b_{1,1}^2)$ pairs at $b_{1,1}$ that are uniquely determined by them.

# of 1s	9	12	23	24
Pairs at $b_{1,1}$	226,227	242,243	122,123	130,131

The procedure to determine a byte in the round key $RK0$ and hence the user key is as follows:

1. Apply 2t (0≤t≤127) at $a_{1,1}$ and run AES chip in normal mode for one clock cycle. Switch to test mode and scan out the bit stream pattern and determine f^1.
2. Reset chip. Apply 2k+1 (0≤k≤127) at $a_{1,1}$ and run AES chip in normal mode for one clock cycle. Switch to test mode and scan out the bit stream pattern and determine f^2.
3. If the number of 1s in $f^1 \oplus f^2$ is 9, 12, 23 or 24, determine $(b_{1,1}^1, b_{1,1}^2)$ using Table 1. Otherwise, go to step 1.
4. Determine round key byte $RK0_{1,1}$ as $b_{1,1} \oplus a_{1,1}$.
5. If all bytes in the round key $RK0$ are determined, stop. Otherwise go to step 1 with a different $a_{i,j}$.

Since only 4-out-of-128 pairs of the type discussed above can be used to determine $b_{1,1}$, on average one of those four deterministic pairs can be encountered after applying 32(128/4) such pairs. In the worst case, 124(128-4) pairs have to be tried to discover one of those four deterministic pairs.

The reset operation does not necessarily imply resetting the chip thereby clearing the secret key RAM. If the round keys are stored in ROM, we can physically reset the chip without clearing the round keys. If the round keys are stored in RAM or register and the chip is in system, we do not use physical reset. Rather, we load the same input and run 10 cycles and the chip will be in a fixed state every time. This state can be the initial state.

In attack step 1, 6 plaintexts on average are required to determine the positions of every group of 32 flip-flops in the scanned out bit stream. This translates into 24 plaintexts ((128÷32)×6)) to determine all 128 flip flops in Round Register R. In attack step 2, 32 plaintexts on average are required to retrieve one byte in round key RK0. This translates into 512 plaintexts ((128÷8)×32)) to retrieve RK0. Overall, 544 plaintexts on average are required to discover the user key.

In a complex crypto system, the AES block may be surrounded by other IP blocks and the scan-based attack will be more difficult. However, the increased difficulty is not intractable. If the data scanned out contains secure information, statistical analysis of scanned out bit streams is still powerful enough to retrieve secure information.

3. SECURE SCAN USING MIRROR KEY REGISTERS

3.1. Security vs. Testability

Scan-based attack is powerful because intermediate results can be scanned out and analyzed. Scan-based attack, in fact, reduces the difficulty of analyzing a cryptographic implementation to just a one round encryption and one or two round keys.

In the Federal Information Processing Standards 140-2 [11], the security issues of cryptographic module is defined as "A cryptographic module shall employ physical security mechanisms in order to restrict unauthorized physical access to the contents of the module and to deter unauthorized use or modification of the module (including substitution of the entire module) when installed. All hardware, software, firmware, and data components within the cryptographic boundary shall be protected". On one hand, sacrificing security for testability by continuing the scan-based DFT restricts its use in privacy sensitive applications. On the other hand, sacrificing testability for security by abandoning scan-based DFT hurts product quality. For the hardware implementation of key based crypto algorithms, if the information that can be obtained from scan-based DFT is not relevant to the secret key stored in the device, such scan chains will not compromise the security provided by the

implemented algorithm. The design goal for the secure scan DFT architecture is two folds:

- Crypto chips should be tested and debugged using general scan-based DFT.
- Information obtained from scan chains cannot be used to retrieve the secret key.

A closer look at cryptographic algorithms such as DES, AES, RSA and HMAC shows that they are all key-based. The security of these algorithms does not rely on the secrecy of crypto algorithms. Rather, the security of these algorithms depends on the secret key. We realized that the testability and security of crypto chips are from two different aspects. The data path and control path that implement a crypto algorithm have to be tested for correct function at fabrication and in-field. The security of a crypto chip relies on keeping the secret key inside a crypto chip secret. Based on this key observation, we develop the secure scan DFT architecture that can meet both testability and security requirements.

3.2. Secure Scan Architecture

In the proposed secure scan DFT architecture we define two new modes of operation: insecure and secure mode. When a crypto chip is in the insecure mode, it can be switched between the test mode and the normal mode similar to the general scan-based DFT. However, when a crypto chip is in the secure mode, it can only stay in the normal mode. While a crypto chip can be switched from insecure mode to secure mode at any time, switching back from secure mode to insecure mode is only done through a power off reset.

To support the secure scan DFT architecture, we use mirror key registers (MKR) to isolate the secret key from the data path and control path performing the crypto algorithm. Such mirror key registers work like normal registers during insecure mode and test vectors can be scanned in and the test result can be scanned out. When the circuit is in the secure mode, the mirror key registers load the secret key information and the contents of mirror key registers cannot be scanned out until being reset.

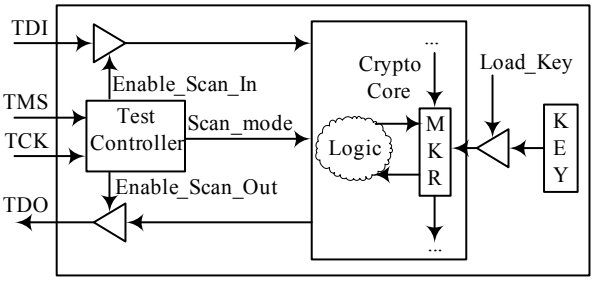

Figure 2: Secure scan architecture with a mirror key register

The secure scan DFT architecture with mirror key registers is shown in Figure 2. A test controller controls the working mode of the crypto chip. In addition to the scan_mode signal used by general scan-based DFT, the test controller generates: Enable_Scan_In and Enable_Scan_Out and Load_Key signals. The crypto core is the circuit implementing the crypto algorithm. In insecure mode, Load_Key signal is disabled to isolate the secret key from the crypto core. The mirror key register works like a normal register. Enable_Scan_In and Enable_Scan_Out signals are active and general scan-based DFT can be performed. The states of all registers in the design can be scanned in and out to verify if the fabricated chip performs as expected. Note that in the model we used for scan-based side channel attack, the key register is not included in the scan chain. On one hand such model provides limited security since we need to perform the two-step attack to recover the key as shown in previous section. On the other hand, it also limits the test capability since only the functionality involving this key is verified. In the secure scan DFT architecture, higher fault coverage can be expected since multiple test vectors can be scanned into the mirror key registers.

Once the function of the crypto chip and upper layer software has been verified, the chip is driven into secure mode for its practical applications. In the secure mode, the secret key is applied to the crypto core by enabling Load_Key signal. At the same time the scan function is disabled by de-activating Enable_Scan_In and Enable_Scan_Out signals. This prevents access to the mirror key register. Scan_mode signal remains inactive during secure mode and hence no shift operation is performed.

Once the chip is in secure mode, it cannot return to insecure mode to perform any test and debug operation. The secret key is loaded to the crypto core only when the chip is in the secure mode. Although the temporary results are stored in other registers, they cannot be scanned out. The only way to re-enter the insecure mode is to reset the chip by power off followed by a power on operation. All registers inside the chip that have temporary results are cleared. Since in crypto chips such as smart cards, a key is usually stored in nonvolatile memory or fabricated as a fixed value, it will not be cleared when the crypto chip is powered off. The operations of crypto chips using the proposed secure scan architecture are summarized in Figure 3.

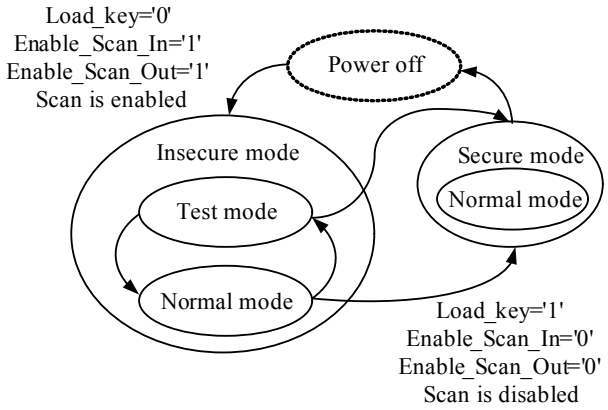

Figure 3: State diagram of secure scan architecture

The design of MKR is shown in Figure 4. A multiplexer controlled by Load_Key can be inserted either at the input to the MKR (Figure 4(a)) or at the output of MKR (Figure 4(b)). In Figure 4 (a), when Load_Key signal is active, the input of the MKR is locked to secret key. Any operation that writes to or scans the MKR is disabled. In Figure 4 (b), the output of the MKR is masked by secret key when Load_Key signal is active. The MKR structure shown in Figure 4 (b) does not modify the scan cell and the insertion of the additional multiplexer does not modify the scan chain. Therefore it is easy to integrate secure scan into current scan DFT design flows.

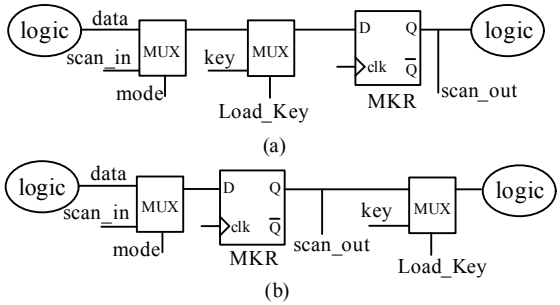

Figure 4: Mirror key register structures. (a) Multiplexer before mirror key register; (b) Multiplexer after mirror key register

The secure scan DFT architecture introduces three new control signals Enable_Scan_In, Enable_Scan_Out and Load_Key. There are two ways to generate these signals. One way is to modify the IEEE 1149.1 Test Controller by adding one more state to generate these signals as shown in Figure 2. A new instruction named Secure_Mode that can be applied through TMS pin is added. The Secure_Mode instruction drives the state machine inside Test Controller into secure mode. Once the Test Controller is in this state, it does not take instructions from TMS any more. The transition from secure mode to insecure mode will not happen until the chip is reset or powered off. In this state, the Enable_Scan_In and Enable_Scan_Out signals are inactive and the Load_Key signal is active. In all other states, they will be set otherwise.

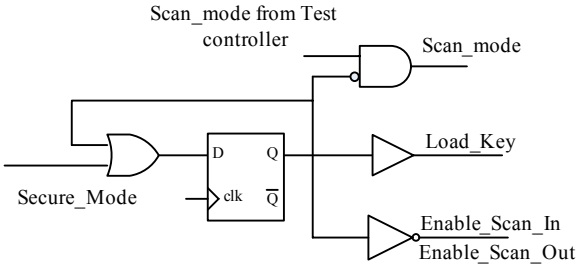

Figure 5: Secure control circuit

The second way is to design a secure control circuit shown in Figure 5. The initial value of the flip flop is zero when the chip is powered up. Once a high pulse Secure_Mode is received through a dedicated pin, the output of the flip flop is 1 and all the signals are driven to the values corresponding to the secure mode. Once the flip-flop is driven to 1, the OR gate before the input will mask any input from Drive_to_secure pin. The chip stays in the secure mode until the chip is reset or powered off.

The security of the proposed secure scan architecture depends on the integrity of the control signals. If the Enable_Scan_In or Enable_Scan_Out is active in the secure mode, the secret key can be scanned out or retrieved by analyzing the temporary results from the scan chains. Any fault on Enable_Scan_In or Enable_Scan_Out signals will compromise the security of the crypto chip. For crypto devices, deliberate faults injected by attackers to break the key are a threat to these critical control signals [12]. Since attackers can only inject deliberate faults randomly, a multiple-stage buffer can be used between TDI, TDO and the crypto core to protect the scan chains. The probability that n-stage buffers are all faulty at the same time and the information related secrete key can be scanned out is much smaller than the probability of a single stage buffer being faulty. For example, an n-stage buffer can be used between TDI, TDO and crypto core as shown in Figure 6.

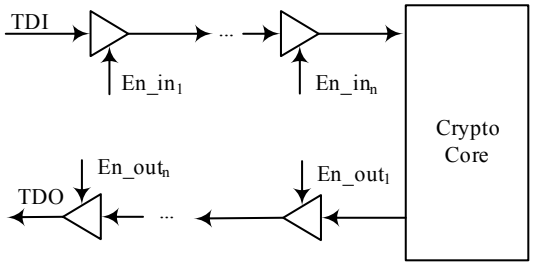

Figure 6: Multi-stage buffer between TDI, TDO and crypto core

4. OVERHEAD ANALYSIS

The secure scan architecture does not degrade the test speed. Since only two-level buffers are inserted in the scan chain, the delay is negligible. The circuits implementing the secret key registers cannot be tested straightforwardly since they are used only in the secure mode. However, by applying a plaintext and observing the corresponding ciphertext can easily reveal if the circuit is correct or not since encrypting a message using different keys will generate different ciphertexts.

We incorporated the secure scan architecture into four implementations of AES namely Iterative with KS (Key Scheduling), Iterative without KS, Pipelined with KS and Pipelined without KS. The AES VHDL models were synthesized using Synopsys Design Compiler. Scan chains were inserted using Synopsys Test Compiler. The additional circuit for the secure scan architecture can be inserted into the netlist generated by Synopsys Test Complier. This modified netlist can be simulated using Modelsim to verify function and can be reloaded into Design Complier to obtain the area overhead. We assume the user key or round keys are stored in nonvolatile memory or fabricated as a fixed value by using ROM. In the architectures with KS algorithm, the round keys are expanded from a 128-bit user key, so only 128-bit MKR is used to separate the user key from the crypto core. For the iterative architecture without key scheduling, round keys are pre-computed and stored in nonvolatile memory or ROM. Only a 128-bit round key is read at each clock. A 128-bit MKR can be used to separate the round keys from crypto cores. However, for the pipelined architecture without key schedule algorithm, all eleven 128-bit round keys are used every clock cycle, each of them uses a 128-bit MKR resulting in a large area overhead compared to the former three architectures.

The area overhead of secure scan architecture using Mirror Key Register for four AES implementations is tabulated in Table 2. The second column shows the area of AES architectures before the secure scan architecture is applied. The third column shows the area overhead incurred by secure scan architecture in terms of gates and percentage. For example, the area of an iterative AES architecture

with key scheduling consumes 31234 gates. The area incurred by secure scan architecture is 412 gates that is 1.32% of the original area.

Table 2 Area overhead of secure scan architecture

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

5. CONCLUSIONS

Secure scan DFT architecture can easily be integrated into the scan-based DFT design flow as the test synthesis. After inserting scan chains during DFT synthesis, the mirror key registers can be specified to the corresponding bit of the secret key. The secure control circuit, and multiplexers between the MKR and the secret key can be inserted. The area overhead of the proposed secure scan architecture is small. The area of secure control circuit is very small. Even if redundant control is used, the area overhead is negligible.

6. REFERENCES

[1] S. Mangard, M. Aigner and S. Dominikus, A Highly Regular and Scalable AES Hardware Architecture, IEEE Transactions on Computer, vol. 52, no.1, pp. 483-491, April 2004.

[2] M.L. Bushnell and V.D. Agrawal, Essentials of Electronic Testing, Kluwer Academic Publishers, 2000.

[3] D, Josephson and S. Poehhnan, Debug methodology for the McKinley processor, International Test Conference, pp.451-460, 2001

[4] B. Yang, K. Wu and R. Karri, Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard, International Test Conference, pp.339-344, 2004

[5] R. Goering, Scan Design Called Portal for Hackers, EE Times, Oct. 2004. http://www.eetimes.com/news/latest/ showArticle.jhtml?articleID=51200146

[6] Maestra Comprehensive Guide to Satellite TV Testing, 2002. http://www.maestra.tv/downloads/Maestra_Guide. pdf

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

[8] R. J. Easter, E. W. Chencinski, E. J. D'Avignon, S. R. Greenspan, W. A. Merz and C. D. Norberg, S/390 Parallel Enterprise Server CMOS Cryptographic Coprocessor, IBM Journal of Research and Development, Vol 43, pp.761-776,1999

[9] D. Hély, F. Bancel, ML Flottes, B. Rouzeyre, M. Renovell and N. Bérard, Scan Design and Secure Chip, IEEE International On-Line Testing Symposium pp.219-226, 2004.

[10] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber and W. Fichtner, A 177Mb/sec VLSI implementation of the international data encryption algorithm, IEEE Journal of Solid-State Circuits, vol. 29, no. 3, pp. 303-307, March, 1994.

[11] National Bureau of Standards, Security Requirements for Cryptographic Modules, Federal Information Processing Standards Publication FIPS PUB 140-2, 2002.

[12] Biham and A. Shamir, Differential Fault Analysis of Secret Key Cryptosystems, CRYPTO, pp. 156-171, 1991.