IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 22, 2022, accepted March 26, 2022, date of publication April 4, 2022, date of current version April 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3164712

A Secure JTAG Wrapper for SoC Testing and Debugging

KUEN-JONG LEE[1], (Fellow, IEEE), ZHENG-YAO LU[2], AND SHIH-CHUN YEH[1]
[1]Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan
[2]Department of Silicon Product Development, MediaTek Inc., Hsinchu 300, Taiwan

Corresponding author: Kuen-Jong Lee (kjlee@mail.ncku.edu.tw)

This work was supported in part by Qualcomm Inc., through the Taiwan University Research Collaboration Project; and in part by the Ministry of Science and Technology of Taiwan.

ABSTRACT IEEE Std. 1149.1, also known as the Joint Test Access Group (JTAG) standard, provides excellent controllability and observability for ICs and hence is widely used in IC testing, debugging, failure analysis, or even online chip control/monitoring. Unfortunately, it has also become a backdoor for attackers to manipulate the ICs or grab confidential information from the ICs. One way to address this problem is to disable JTAG pins after manufacturing testing. However this countermeasure prohibits the in-filed testing and debugging capability. Other countermeasures such as authentication and encryption/decryption methods based on specific static keys have also been proposed. However, these approaches may suffer from side-channel or memory attacks that may figure out the specific keys. This paper presents an authentication-based secure JTAG wrapper with a dynamic feature to defend against the attacks mentioned above. We generate different keys for different test data dynamically. Therefore, only legal test data can be updated to the test data registers (TDRs) through JTAG. Furthermore, the attackers will get fake responses if they shift in illegal test data, which makes it extremely difficult to break our proposed method. We can also employ the physical unclonable function (PUF) to distinguish the legal test data for different chips. Experiments on a RISC-V CPU processor called SCR1 show that our proposed method can have an area overhead of only 0.49%.

INDEX TERMS Hardware security, IEEE test standard security, JTAG security, memory attack, secure JTAG wrapper, physical unclonable function (PUF), in-field testing, in-field debugging.

I. INTRODUCTION

IEEE 1149.1 standard was initially developed for board-level testing. Nowadays, this standard has been adopted in many applications such as post-silicon debugging, chip reconfiguration, verification, power management, and clock control [1]. This standard defines mandatory hardware components, including a test access port (TAP), a test access port controller (TAPC), an instruction register (IR), a bypass register (BR), and a sequence of boundary-scan cells (BSCs), as well as some optional test data registers (TDRs) [2]. Users can access the BSCs and the TDRs through the TAP with excellent observability and controllability.

However this convenient feature may become a backdoor for potential attackers [3], [4]. Many attacks that exploit the controllability and observability of JTAG have been reported. These include breaking the secret key of a cipher circuit [4]–[7], finding the possible backdoor of an FPGA [8], modifying the firmware of a gaming console [9], getting the root privilege of an IoT device [10], etc.

This paper proposes an authentication-based JTAG wrapper with a dynamic feature that can prevent the attacker from arbitrarily accessing the TDRs. The main idea of this method is to authenticate all the test data to be updated to the intellectual properties (IPs) through TDRs. We add a test seed in front of each test data to form legal test data. Only the legal test data can pass the authentication procedure and be updated to the protected IPs. In addition, the attacker will get a fake response if they shift in illegal test data and then try to shift out any data in the IPs. We deploy a golden key generator and a test key generator to the JTAG infrastructure. The golden key generator generates various golden keys for different JTAG instructions based on a physical unclonable function (PUF) or a key stored in memory. A legal test data will make the test key generator generate a correct test key which is the same as the golden key.

The advantages of our proposed method include high security and low area overhead. Traditionally, authentication-based countermeasures authenticate a single user before

The associate editor coordinating the review of this manuscript and approving it for publication was Jonathan Rodriguez.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

using the JTAG infrastructure. Therefore if attackers can pass the authentication by using some brute-force or side-channel attack, they can use the JTAG infrastructure freely afterward. Our proposed method authenticates every test data before it is updated to a TDR. Thus even if attackers find a seed of a specific test data, they still need to break the seeds of other test data. The main components of the proposed test key generator and fake response generator are linear feedback shift registers (LFSRs) which are much smaller than encryption/decryption circuitry. Hence, the area overhead is relatively low.

We organize this paper as follows. Section II describes the threats to JTAG and the countermeasures proposed in literal. Section III presents the proposed secure JTAG wrapper and its components in detail. The test seed generation method is illustrated in Section IV. In Section V, we evaluate the security of our proposed method and analyze the risk of suffering from common attacks. In Section VI, we report experimental results and compare the results with previous works. Finally, Section VII concludes this paper.

II. BACKGROUND AND PRIOR WORKS

A. THREATS TO JTAG

The convenient feature provided by the JTAG may make itself a backdoor for potential attackers. An attacker can achieve malicious purposes by stealing secret information [4]–[7] or modifying the firmware. Below are two examples of JTAG attacks.

In [9], the authors present several attacks to hack some modern gaming consoles. The attacks are a combination of software and hardware attacks. One typical example is to use the JTAG to manipulate the direct memory access (DMA) unit by setting the contents of some target DMA addresses. Then, when the system management controller (SMC) triggers the DMA unit, such as loading the necessary memory address, the attacker can successfully overwrite the memory context and force the kernel to jump to a specific address.

In [10], the authors provide a way to get the root privilege of a popular Wi-Fi router through the on-chip debugging (OCD) mechanism controlled by the JTAG. The OCD mechanism allows one to set watchpoints to pause the device and get the memory content at each memory address. The root privilege can be obtained by modifying the boot arguments in memory to boot the device into the single-user mode. To modify the boot arguments successfully, the attacker has to know the actual memory address of the firmware. Therefore, they utilize the OCD mechanism to set watchpoints at the possible memory addresses to pause the device when it touches the addresses. The actual memory address of the firmware can be figured out by setting enough watchpoints. Then they can modify the booting arguments and continue the process to get the root privilege with the actual address.

The JTAG-based attacks can be implemented successfully if the attacker can directly access the hardware components, including IR, scan chains, and TDRs, through the TAP. The attacker can get the highest privilege as long as they exploit these hardware components.

B. COUNTERMEASURES OF JTAG ATTACKS

Several countermeasures have been proposed to defend against JTAG-based attacks. In this paper, we classify these protection schemes into four categories based on the taxonomy of [4], as shown in Fig. 1 and described next.

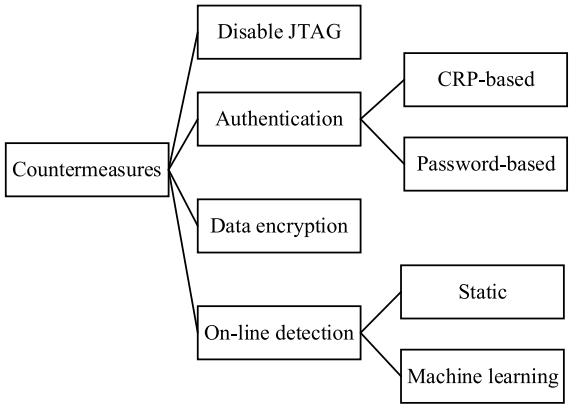

FIGURE 1. Taxonomy of protection schemes [4].

1) DISABLE JTAG

This scheme disconnects the TAP pins after manufacture testing and debugging, which can efficiently prevent attacks through JTAG. However, this disables the in-field testing and debugging functions [11] that are essential for mission-critical applications such as automotive, military, and medical electronics, where high reliability of ICs is demanded [12].

2) AUTHENTICATION-BASED METHOD

The authentication-based scheme can be further divided into password-based [1], [13] and challenge-response-protocol-based (CRP-based) methods [14]–[17]. Their common objective is to verify the legality of users.

The test infrastructure has two states for the password-based methods: locking state and unlocking state. Users cannot access any test data registers (TDR) except the bypass register in the locking state. To unlock the test infrastructure, they must shift a secret password into a dedicated register. An authenticated user can then access all the TDRs in the unlocking state.

Some methods that apply the challenge-response-protocol authentication to the test wrapper have been proposed to enhance security [14]–[17]. The CRP-based methods involve at least two parties: device and user. When a user requires to use a device, the device sends a challenge to the user. In a simpler case, the user just sends the correct response back. In a more complex case, the user cannot generate the correct responses by himself; he would need to obtain the correct responses from a third party, e.g., a secure server.

The CRP-based methods are generally more secure than the password-based methods since the challenge may vary

each time. The attacker needs more effort to implement the brute-force attack on these methods. However the CRP-based methods may require higher area overhead since the authentication protocol is usually based on cryptography.

3) ENCRYPTION-BASED METHOD

When an encryption-based scheme is implemented, the user needs to encrypt the test data with a secret key before shifting the test data via TDI [18]. After the encrypted test data is sent to the device, it will be decrypted with the secret key. Therefore, only the user who knows the secret key in the targeted device can encrypt the test data correctly. Moreover, the test data/response will be encrypted before shifting out via TDO. Therefore, the user cannot get the true test data/response until decrypting it with the correct key. The encryption algorithm can be a stream cipher or a block cipher. The issue with this method is that once the secret keys are figured out, all protections will fail. Also, the area overhead would be high.

4) DETECTION-BASED METHOD

The main idea of authentication-based and encryption-based countermeasures is to prevent an attacker from accessing JTAG infrastructure. However, people can use the JTAG freely once authenticated or have the correct cipher key. Therefore, implementing detectors that monitor the operations of JTAG is an alternative way to avoid potential risks.

Static and machine learning-based detection methods have been proposed [11]. Static detectors rely on some specific rules defined in the IC design stage. A set of JTAG instruction sequences is chosen to represent legitimate operations. If users enter an instruction sequence that does not follow the rules, they will be regarded as attackers. Machine-learning detectors are trained to detect the behaviors of an attacker. In the training phase, the JTAG operations are characterized by a set of features. The features include intra-instruction statics and inter-instruction transition, such as the instruction opcodes and the number of clock cycles within the shift-DR state. Then, models are trained using these features to classify the JTAG operations into normal operations or attacks. This way, models are expected to be classified in real-time when the chips are running, and hence online detection can be achieved.

The detection-based methods highly rely on the collected information of JTAG attacks. A comprehensive analysis of these attacks is needed to ensure the accuracy of the classification. False negatives and false positives are possible without enough collected information for this countermeasure.

III. PROPOSED SECURE JTAG WRAPPER

We propose an authentication-based countermeasure against JTAG attacks. Fig. 2 shows the overview of the proposed secure JTAG wrapper, which comprises three main parts: the original JTAG infrastructure, the secure JTAG architecture, and the golden key generator. The original JTAG infrastructure contains TAP, TAPC, IR, and several TDRs, as shown in the brown boxes. The secure JTAG architecture consists of a test key generator, a key comparator, a gating logic, a fake response generator, a logic controller, and some MUXs, as shown in the blue boxes. The golden key generator is shown in the yellow box. Next, we summarize the basic ideas of our proposed method and describe their details in the following sections.

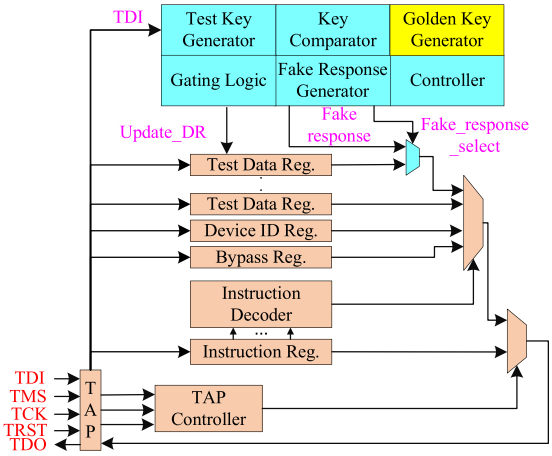

FIGURE 2. Overview of the proposed secure JTAG wrapper.

- The golden key generator generates a unique golden key for each defined JTAG instruction. Our design assumes that different TDRs are accessed by different instructions which the user or company already defines. For example, if two TDRs named IP1 and IP2 are used for debugging, two instructions called DEBUG-IP1 and DEBUG-IP2 may have been defined. The inputs to the golden key generator are a secret key and a JTAG instruction. The secret key can be derived from a master key stored in memory such as flash or ROM memory or from a PUF circuitry. IC designers can choose the source of the key according to security requirements and available resources.
- The test key generator is used to generate a test key for each test data with a seed. The test key is then compared to the golden key in the key comparator. Only when the test key is equal to the golden key can the user exploit the JTAG functions.
- The gating logic will set the Update_DR signal to 1 only when the test key is the same as the golden one. Hence users who do not enter the correct seeds cannot access their target TDR correctly.
- The fake response generator will generate fake responses of the TDRs after a Capture_DR operation has been done. The MUXs are responsible for selecting true or fake responses to shift out, making the attacker confused about the authenticity of responses.
- The test seed and test data are shifted in from the TDI. Hence no additional input pin for keys is needed. Besides, there is no need to add new JTAG instructions. Thus the attackers may not know whether there is

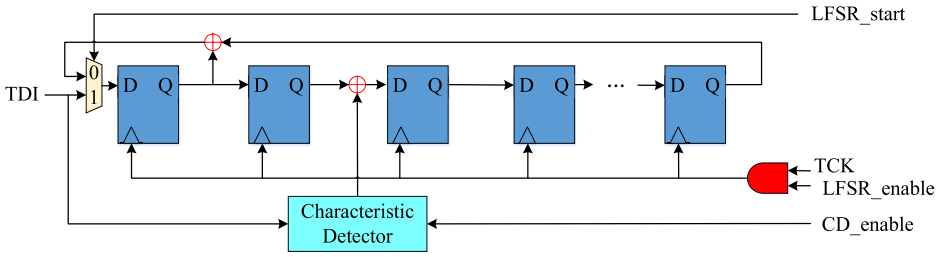

FIGURE 3. Test key generator.

a protection mechanism and cannot determine whether the information obtained is true or fake.

A. GOLDEN KEY GENERATOR

The golden key generator generates different keys for different instructions that access TDRs. The inputs are a secret key and a JTAG instruction. The secret key may be stored in memory or generated by a PUF. The PUF is a function with outputs depending on the fabrication process variation [19]. It generates different values for different ICs under the same input. Many PUF designs have been proposed so far. Our proposed method can adopt any PUF design. It is harder to predict the golden keys if a PUF is adopted. However the area overhead may be larger than using a key stored in memory. Therefore, IC designers may select suitable schemes based on their security requirements.

B. TEST KEY GENERATOR

A test key generator consists of an LFSR called the key LFSR and a characteristic detector, as shown in Fig. 3. The purpose is to generate the test key from the test data and its associated seed. The user must shift the seed derived from a test data into the key LFSR before entering the test data. The contents of the LFSR are changed when the test data is being shifted into the TDR. After the test data is shifted into the TDR, the content of the key LFSR will be the test key. The test key will be compared with the golden key in the key comparator. The detailed procedure for generating the test key is as follows. In the beginning, an instruction is shifted to the instruction register. After that, the LFSR_start and the LFSR_enable signals from the controller are set to 1. Then, the seed is shifted into the key LFSR from the TDI port. After the seed is loaded into the key LFSR, the LFSR_start signal is changed to 0 to start running the LFSR. At the same time, the output of the characteristic detector starts to alternate the contents of the key LFSR as follows.

The purpose of the characteristic detector is to make the key generation algorithm irregular and difficult to predict. Refer to Fig. 3. The inputs to the characteristic detector are TDI and CD_enable, where CD_enable is from the controller. When the CD_enable is set to 1, the characteristic detector will monitor the value of the TDI port. As long as a designer-defined characteristic pattern appears at the TDI port, the characteristic detector will send a trigger signal to change the content of the key LFSR. For example, assume the characteristic pattern is 101. Then whenever 101 appears at the TDI port and the CD_enable is set to 1, the characteristic detector will send 1 to the key LFSR to change its content. Hence, only the test data with the correct seed can generate the correct test key in the test key generator. In this paper, we call the test data with correct seeds *legal test data*. On the contrary, we call the test data with incorrect seeds *illegal test data*.

C. KEY COMPARATOR

As shown in Fig. 4, the key comparator compares a test key and a golden key. It outputs a valid signal if they are identical. The test key generator generates a test key based on the seed and the test data. The comparison result must become available after the test data is shifted in and before the test data is updated. Therefore, we use a D flip-flop driven by the Exit1_DR signal to latch the comparison. The comparison result is then sent to the gating logic and the fake response generator to guide whether the test data can be updated and whether the correct response can be shifted out.

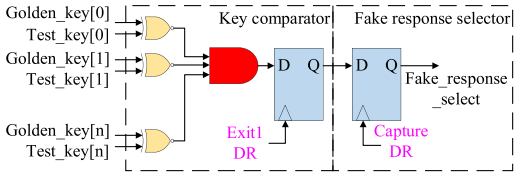

FIGURE 4. Key comparator and fake response selector.

D. GATING LOGIC

A gating logic is in charge of preventing illegal test data from being updated to the TDRs. As shown in Fig. 5, we use AND gates to set the Update_DR signals. The Update_DR signal will be 0 if the comparison_result is 0, in which case the test data cannot be updated. In addition, IC designers can choose which TDRs to protect. The Update_DR signals to unprotected TDRs will not be connected to the AND gates. In this way, some debugging functions not associated with secret information or system manipulation can be opened to normal users. This will reduce the timing overhead when executing some debugging procedures.

E. FAKE RESPONSE GENERATOR

A fake response generator comprises a fake response selector, a shadow LFSR, some XORs, and MUXs, as shown in Fig. 6. Attackers will get fake responses when they shift in illegal test data and capture the response of the TDR afterward.

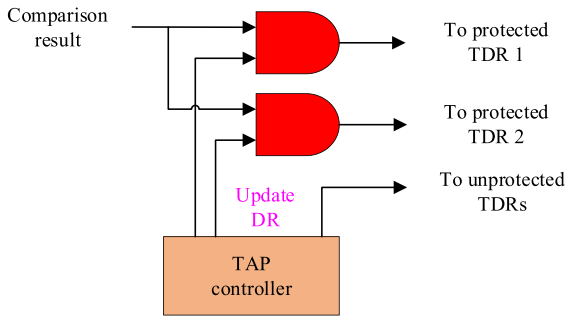

FIGURE 5. Gating logic.

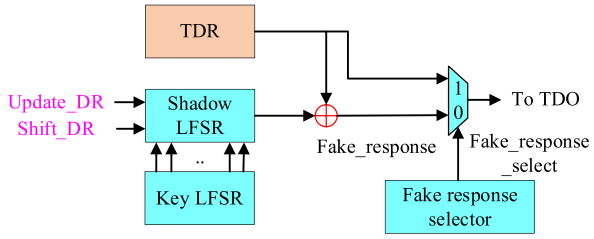

FIGURE 6. Fake response generator.

Furthermore, attackers will get the same data when they shift in the same test data, either legal or illegal. Therefore it is difficult to figure out whether the input data is legal. This also prevents attackers from noticing the existence of the protection circuit and is effective to protect against exhaustive attacks.

The fake response selector selects whether to shift out a true or fake response. Fig. 4 shows the fake response selector. First, it receives the comparison result from the key comparator. Then the comparison result is latched as Fake_response_select using a D flip-flop driven by the Capture_DR signal. Fake_response_select is then sent to the MUXs to determine whether to shift out fake responses. Notice that the initial value of the Fake_response_select is set to 0. The Fake_response_select is set to 1 only when legal test data is entered.

Refer to Fig. 6. The key LFSR updates its value to a shadow LFSR when the Update_DR signal is 1. After that, the shadow LFSR starts running when the TAP controller is in the Shift_DR state. The fake response is thus obtained by XORing the outputs of the protected TDR and itself. Since we do not want the attacker to recognize that the response is fake, the fake response is XORed with the TDR's output. In addition, the fake response will be the same if the same illegal test data is shifted into the protection mechanism because the key LFSR provides the initial value of the shadow LFSR. Note that the characteristic polynomial of the shadow LFSR should be different from the key LFSR to hide the structure of the key LFSR.

IV. SEED GENERATION METHOD

After implementing the proposed protection mechanism inside a chip, the designers need to derive the seed for each test data. This derivation is done by performing a reverse derivation process for LFSRs [20], as described next.

We acquire the golden key of each instruction. It can be done easily since the Boolean function of the golden key generator is predefined during the IC design phase. As long as we know the actual value of the secret key, we can generate all golden keys. In addition, the characteristic polynomial of the key LFSR and the characteristic pattern of the characteristic detector are also predefined. For illustration, assume that the key LFSR has N bits. The trigger signal is connected to the output of the M^{th} D flip-flop of the key LFSR. The length of a TDR is L, and the trigger signal is 1 at the K^{th} cycle when shifting in the test data. Note that the signal may be triggered more than once.

Under the above assumptions, we can utilize an LFSR with a reciprocal characteristic polynomial of the key LFSR to derive the seed. We call this LFSR a reverse LFSR (RLFSR) hereafter. The trigger signal is connected to the output of the $(N\text{-}M\text{-}1)^{st}$ D flip-flop of the RLFSR when the value of (N-M-1) is larger than zero. Otherwise, the trigger signal is connected to the output of the $(N+(N\text{-}M\text{-}1))^{th}$ D flip-flop of the RLFSR. The trigger signal is inverted at the $(L\text{-}Q)^{th}$ cycle. The seeds can be derived by simulating the RLFSR with the initial data equaling the reverse golden key for L cycles. Note that the RLFSR is only for simulation. We do not have to implement it physically inside a chip.

For example, we assume the key LFSR is a 5-bit LFSR, and its characteristic polynomial is x^5+x^2+1. The length of the TDR is 8. The trigger signal is connected to the output of the 3^{rd}D flip-flop, and it will be 1 at the 6^{th}cycle, as shown in Fig. 7. The reverse characteristic polynomial of x^5+x^2+1 is x^5+x^3+1, which is the characteristic polynomial of the RLFSR. The trigger signal is connected to the output of the 1^{st} D flip-flop (5-3-1 = 1), and it will be 1 at the 2^{nd} cycle. Suppose that the golden key is 10100. We can get a reverse golden key 00101. Then we set the initial data of the RLFSR as 00101 and run for eight cycles. Since the trigger signal is 1 at the 2^{nd} cycle, the final value of the RLFSR is 11001, as shown in Fig. 8. We reverse this final value to get the seed 10011. Refer to Fig. 7. To verify the correctness of the seed, we can set the derived seed as the initial data of the key LFSR and run it for eight cycles. With the trigger signal being 1 at the 6^{th} cycle, we will obtain the final value of the key LFSR 10100, which is the same as the golden key we defined.

This seed generation method can obtain the seeds of all test data. Fig. 9 shows the seed generation flow of a chip. First, we set some environment parameters, such as the polynomial of the key LFSR, the golden key's length, and the trigger signal's position. The key LFSR model and the reverse LFSR model can be built with these parameters. Second, we read in the test data file, which records the golden key of each instruction, the length of each TDR, and the needed test data. Third, we read one test data and analyze the trigger cycles. The seed can be generated by simulating the reverse LFSR for L cycles. Fourth, we append the seed in front of the

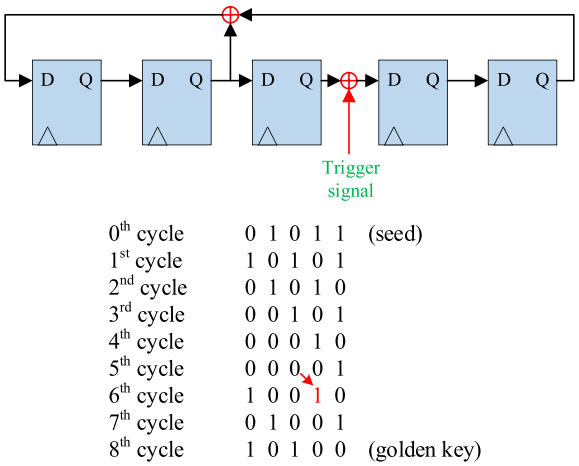

FIGURE 7. A 5-bit key LFSR.

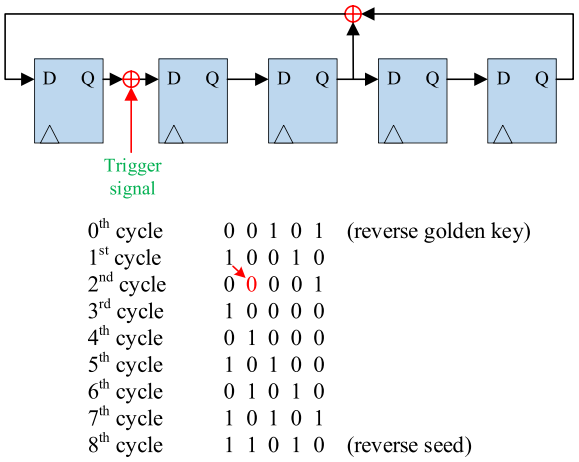

FIGURE 8. The reverse LFSR.

test data to form legal test data and record it. We repeat the above steps until all legal test data belonging to this TDR has been obtained. Then we set the golden key according to the instruction and the length of another TDR and get the corresponding seeds. Finally, the legal test data file will be generated.

V. SECURITY ANALYSIS

In this section, we analyze the security of the proposed method against various typical attacks that aim to break the protection mechanism in different ways. In addition, a comparison with prior works is also given.

A. BRUTE-FORCE ATTACK

The brute-force attack, also known as the exhaustive attack, is based on the trial-and-error method. The attacker finds the correct answer by deleting the wrong candidate repeatedly. It can be applied to the countermeasures mentioned above too. The password-based methods protect the JTAG access by requiring users to shift in a secret password. A static or short password is vulnerable to a brute-force attack [21].

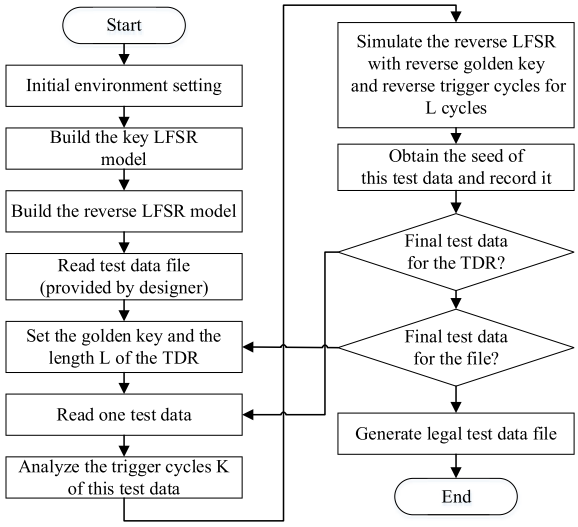

FIGURE 9. Seed generation and legal test data derivation flow.

Our work uses a different seed for different test data to authenticate the user instead of a static password. Moreover, the fake response generator is implemented to obfuscate attackers, making attackers unable to recognize whether the test data is legal. In other words, attackers cannot remove the wrong candidates. Even if attackers could find the correct seed of a specific test data after many tries, they still need much effort to break the seed of another test data. Assume the length of a seed is N, the size of a TDR is L. There are 2^N possible seeds for each test data and 2^L possible test data for the TDR. There are 2^{N+L} possible cases for a TDR. Besides, since the golden key for each instruction is different, the attacker must find a different seed for another instruction.

B. MEMORY ATTACK

The memory attack aims to discover sensitive information in memories. Some methods to retrieve the data have been presented. The "memory cold boot attack" is a famous one [22]–[26]. Taking dynamic random-access memory (DRAM) as an example, the data in DRAM may retain for several seconds after it is powered off, even if it is removed from a motherboard. This characteristic is called memory remanence. Detailed steps to get the data in DRAM based on memory remanence are presented in [25]. With this kind of attack, confidential data such as an encryption key can be retrieved.

We can use a secret key (stored in memory or generated by a PUF) to generate the golden keys. The golden keys are compared with the test keys generated by the seeds and test data using the test key generator to check whether the test data is legal. Even if the memory attack may discover the secret key, it is hard for the attacker to break our protection mechanism due to the following reasons. First, the attacker may not know how to derive the golden keys from the secret key. Second, the attacker cannot derive the seeds shifted into the TDI with test data. Third, the seeds are entered into the chips only when test data is entered; they are never stored

statically in memory. Due to these facts, our proposed method can defend against memory attacks effectively.

C. REVERSE ENGINEERING ATTACK

Research on the reverse engineering attack has been carried out [27]. This type of attack assumes that direct access to a chip is possible, and the connection of wires and the logic gates used in a chip can be fully identified. In other words, it is assumed that the attacker can figure out the detailed logic design information of the protection mechanism.

In this paper, our secure structure uses a secret key stored in memory or generated by a PUF to derive a different seed for each test data. Even if the reverse engineering attack can be completely and accurately carried out and the seed generation algorithm is completely identified (which is very unlikely), the attacker still needs to guess the secret key used in the protection mechanism. Suppose the secret key is a PUF key. In that case, the attacker will need great effort to break another chip by repeatedly guessing the correct values even though a PUF key for a specific chip is obtained.

D. SIDE-CHANNEL ATTACK

Side-channel attacks can be divided into three categories: the active side-channel attack, the passive side-channel attack, and the scan-based side-channel attack [27]. The active side-channel attack uses special equipment to penetrate the hardware circuit to extract or change signals inside the circuit. For example, attackers may utilize electromagnetic fault injection (EMFI) to obtain secret information [28]. This would require some expensive equipment and hence could be pretty costly. The passive side-channel attack uses some devices to collect data from the I/O pins of a chip and then analyze the collected data with various techniques to deduce some secret circuit information. For example, an attacker may use the current or power consumption curves at I/O pins to figure out the secret key of the circuit [27]. This however requires complete control of the chip and full understanding of the detailed operations of the chip, usually at the exact cycle level, and hence may be difficult to implement in practice. The scan-based side-channel attacks utilize the excellent controllability and the observability of the scan design to obtain secret information through scan chains [7].

Our proposed method does not explicitly aim to defend against active and passive side-channel attacks. However, our methods can be easily combined with existing countermeasures for these attacks. For instance, one may use shielding methods such as those used in commercial hardware security modules or incorporate the latch-up technique of [29] in our method to defend against the fault injection attack. One may also use the false glitch cell method of [30] to protect against the power analysis attack. As for the scan-based side-channel attack, if the scan-based testing is controlled by boundary-scan, then our method certainly can defend against the attacks. However if the scan-based testing is totally independent of the boundary scan, then our method cannot protect against the scan-based attack.

TABLE 1. Analysis of possible vulnerability of different countermeasures.

Secure JTAG design	Possible vulnerability
Disable JTAG	Invasive attack
Password-based authentication (Hardcoded)	Brute-force attack, Reverse engineering attack
Password-based authentication (Memory)	Brute-force attack, Memory attack
CRP-based authentication	Memory attack
Encryption	Memory attack
Detection	Unknown attack
Proposed method	Memory or PUF attack, reverse engineering attack, and seed algorithm breaking attack are all successful at the same time

TABLE 2. JTAG instructions for SCR1.

IR code	Instruction	Description
0x01	IDCODE	Read ID code TDR
0x04	BLD_ID	Read BLD_ID TDR
0x09	SCU_ACCESS	Access system control unit TDR
0x10	DTMS	Access debug transport module TDR
0x11	DMI_ACCESS	Access debug module interface TDR
0x1F	BYPASS	Access bypass register

Finally, it is worth pointing out that even if the side-channel attack can figure out the secret keys, the attack still cannot be completed unless both the key reconstruction circuit structure (including the fake generator) and the seed generation procedure are fully explored.

E. COMBINED ATTACKS WITH ELABORATED MODELS

Combined attacks with elaborated models refer to combining different attacks with various resources to break the protection mechanism. Clearly a combined attack method is more powerful than a single model attack because it can attack a circuit through multiple channels. However, attackers need more effort and resources, and hence much more costs, to implement a combined attack.

We have discussed four different types of attacks in the above sub-sections ***A***-***D***. Our proposed method can defend against the brute-force attack, the memory attack, the reverse engineering attack, and the scan-based side-channel attack. Our proposed method is defeated only when all the following conditions are met: 1) the secret key is broken, 2) the architecture of the key generator is cracked, and 3) the seed generation algorithm is figured out. In other words, a combined attack must combine the attack methods that together can meet all the above conditions to break our defense mechanism, which would require extremely high cost in terms of both time and equipment resources. One needs to guess the key exhaustively or use expensive equipment to carry out side-channel attacks

TABLE 3. Operation time overhead of the proposed method with SCR1.

	SCU_ACCESS				DTMS				DMI_ACCESS			
Length of the seeds	8	32	64	128	8	32	64	128	8	32	64	128
Original (clock cycles)	38				86				105			
Proposed method (clock cycles)	46	70	102	166	94	118	150	214	113	137	169	233
Time overhead (clock cycles)	8	32	64	128	8	32	64	128	8	32	64	128

TABLE 4. Area overhead of the proposed method with different size seeds.

Benchmark	SCR1			
Original area (GEs)	44,891			
Length of the seeds (bits)	8	32	64	128
Proposed method area (GEs)	45,111	46,047	46,700	47,834
Area overhead (GEs)	220	1,156	1,809	2,943
Area overhead (%)	+0.49	+2.58	+4.03	+6.56

to figure out the key. To reconstruct the key generator architecture, one also needs to perform the time-consuming and easy-to-fail reverse engineering work. Furthermore, unless the seed generation algorithm code is stolen, there is no way to figure out the detailed and exact procedure of the key generation algorithm.

F. COMPARISON WITH OTHER WORKS

Some countermeasures have been described in Section II. Each countermeasure has its advantages and disadvantages. This section analyzes the pros and cons of these methods and our method.

Disabling the JTAG is a straightforward way to defend against the aforementioned attacks. However, in-field testing and in-field debugging are also disabled. Furthermore, accessing the TAP pins is still possible for the attacker using some invasive attack such as probing [11].

The authentication-based countermeasures tell whether the user is legal or not depending on the passwords or the CRPs [12]–[17]. The password-based methods may suffer from brute-force attacks or some side-channel attacks. For example, suppose a password is compared to the real key bit-by-bit serially. The attacker can use time variance to realize which bit of the password is wrong [31]. They are also vulnerable to memory attacks if the passwords are stored in memory. In addition, if the authentication procedure compares a hardcoded password, it may also be cracked by the reverse engineering attack [21]. As for the CRP-based methods, they also take the risk of a memory attack. The brute-force attack may be ineffective for the CRP-based methods because the challenge may alter each time the user sends an accessing request to the test infrastructure. However, the challenge-response protocol used is usually public to all users. Once the secret information is discovered, the attacker can generate correct responses arbitrarily.

The encryption-based countermeasure decrypts the test data and encrypts the response using a cipher circuit inside a chip [18]. Thus the attacker cannot get the true test response until decrypting with the correct key. However the memory attack is effective on the encryption-based countermeasures. Once the key is leaked, the attacker can encrypt and decrypt the data correctly.

The detection-based countermeasures monitor the behavior of JTAG operations based on the instructions [11]. Its main idea is quite different from the authentication-based and encryption-based countermeasures. The attacks mentioned above usually do not target detection-based countermeasures. However, with the popular detection method based on machine learning, the detector must be trained with an established attack model or predefined rules. An attack that is not considered in the training phase may escape the judging of the detector, which is named the unknown attack in [11].

Our proposed countermeasure authenticates different test data using different keys. Thus, the attacker cannot crack the protection mechanism with a single attack model. The protection method is defeated only when the secret key is broken or stolen, the architecture of the key generator is cracked, and the seed generation algorithm is figured out completely. We summarize the possible vulnerability of the countermeasures in TABLE 1.

VI. EXPERIMENTAL RESULTS

This section presents the experimental results of implementing our proposed method in an open-source processor core called SCR1 provided by Syntacore [32]. The SCR1 processor is equipped with a debugging sub-system based on the RISC-V debug specification. The user can access the IR and TDRs via the JTAG interface. The SCR1 processor provides six JTAG instructions as listed in TABLE 2. Unused IR codes are mapped to the bypass register. Our proposed method protects the TDRs corresponding to SCU_ACCESS, DTMS, and DMI_ACCESS using 8-bit golden keys in the experiment. The operation time overhead and area overhead are described as follows.

A. OPERATION TIME OVERHEAD

For the proposed method, a legal test sequence (data) is comprised of a seed and an original test data. Assume that the length of the golden key is N bits. The operation time overhead for each legal test data is N cycles. TABLE 3 denotes

TABLE 5. Comparison of area overhead with prior countermeasures.

Secure JTAG design		Benchmark	Area overhead (GE)	Technology
Password-based authentication [13]		AES	566	tsmc 0.18 um
CRP-based authentication [17]		NA	46,716	Faraday 130 nm
Encryption [18]		LEON3	3,122	NA
Detection [11]	Random forest	OpenSPARC T2	371,013	0.18 um
	SVM		224,688	
	Anomaly detector		32,445	
The proposed method (8-bit seed)		SCR1	220	tsmc 90 nm

the operation time overhead for the SCU_ACCESS, DTMS, and SMI_ACCESS instruction, with different seed lengths added to the original design. The clock cycles are calculated by performing one instruction. Note that the consumed clock cycles begin from shifting an instruction into the IR to shifting out the response.

We use an 8-bit golden key in our protection scheme, and hence the operation time overhead is eight cycles. We choose an 8-bit golden key because our method can reach a high-security level with a small seed since the correct seed alters with the test data. Furthermore, our protection method can defend against memory attacks, brute-force attacks, and reverse engineering attacks with a small-size golden key.

B. AREA OVERHEAD

TABLE 4 reports the area overhead of our proposed method with the SCR1 processor. The circuit is synthesized using Synopsys Design Compiler with the tsmc 90 nm technology file. The original area of the SCR1 processor is 126,593.81 um^2, which is evaluated as 44,891 gate equivalences (GEs).

When the proposed design with 8-bit to 128-bit seeds is added to the original design, the design area becomes 45,111 to 47,834 GEs. Thus, our proposed method requires 220 to 2,943 extra GEs. The area overhead will increase while a longer seed is adopted. Nevertheless, as mentioned in the previous paragraph, our method can reach a high-security level with a small seed. With the use of 8-bit seeds, the area overhead is only $220/44891 = 0.49\%$. Thus the IC designers may prefer to choose the small-size scheme.

C. COMPARISON WITH PRIOR WORKS

A comparison of our work with prior countermeasures [11], [13], [17], [18] is given in TABLE 5. Although the benchmarks used in these works may be different, most of them focus on an SoC that provides a debug sub-system within a chip. The area overhead data are extracted from these papers, though the environmental parameters of each work may be different. The encryption-based countermeasure [18] uses a lightweight stream cipher, while the CRP-based authentication [17] implements some circuits to achieve the challenge-response protocol. Both countermeasures have an area overhead of thousands of GEs. The work in [11] includes two ML-based detectors and one anomaly detector. As we can see, our work and password-based authentication [13] have the lowest area cost, followed by the encryption-based countermeasure, the static detector, and the CRP-based authentication. The ML-based detectors [11] require the most logic gates and extra memory to store the weights.

VII. CONCLUSION

In this paper, we propose an authentication-based secure JTAG wrapper. We add a lightweight security circuit and do not need to modify the original JTAG infrastructure. Our proposed method allows users to access the TDRs when the legal test data is entered. Furthermore, if attackers shift in illegal test data, they will get fake responses. In addition, we generate a unique, dynamic seed for each test data. Thus we can reach a high-security level with a short golden key due to the dynamic nature. We also develop the seed generation algorithm and the seed generation flow, making the production of seeds automatic.

The secure architecture contains a test key generator, a golden key generator, a key comparator, a gating logic, a fake response generator, and a controller. They require only 220 GEs or 0.49% area overhead for the SCR1 processor. In addition, we append the seed to each test data, so no additional pin is required. Also, there is no need to add any JTAG instruction either. Furthermore, since a fake response generator is adopted, the attacker cannot tell whether he receives real or fake data. Moreover, the proposed method can combine with a PUF key, making it harder for attackers to attack other chips even if they break a chip successfully.

Therefore, our proposed method can effectively defend against the brute-force attack, the memory attack, the reverse engineering attack, and the scan-based side-channel attack with a small area overhead. To sum up, the proposed secure JTAG wrapper is a lightweight and secure countermeasure against JTAG attacks.

REFERENCES

[1] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Amsterdam, The Netherlands: Elsevier, 2006.

[2] *IEEE Standard for Test Access Port and Boundary-Scan Architecture—Redline*, IEEE Standard 1149.1-2013, May 2013, pp. 1–899.

[3] J. D. Rolt, A. Das, G. D. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test versus security: Past and present," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 50–62, Mar. 2014.
[4] E. Valea, M. Da Silva, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "A survey on security threats and countermeasures in IEEE test standards," *IEEE Des. Test*, vol. 36, no. 3, pp. 95–116, Jun. 2019.
[5] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Novel test-mode-only scan attack and countermeasure for compression-based scan architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 808–821, May 2015.
[6] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A scan-based attack based on discriminators for AES cryptosystems," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 92, no. 12, pp. 3229–3237, Dec. 2009.
[7] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2287–2293, Oct. 2006.
[8] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Cryptographic Hardware and Embedded Systems–(CHES)*. Berlin, Germany: Springer, 2012, pp. 23–40.
[9] E. DeBusschere and M. McCambridge, "Modern game console exploitation," Dept. Comput. Sci., Univ. Arizona, Tucson, AZ, USA, Tech. Rep., 2012.
[10] Senrio. (2016). *JTAG Explained (Finally!): Why 'IoT' Software Security Engineers, and Manufacturers Should Care*. [Online]. Available: https://blog.senr.io/blog/jtag-explained
[11] X. Ren, F. P. Torres, R. D. Blanton, and V. G. Tavares, "IC protection against JTAG-based attacks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 1, pp. 149–162, Jan. 2019.
[12] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," in *Proc. 43rd Annu. Conf. Design Autom. (DAC)*, 2006, pp. 7–12.
[13] G. M. Chiu and J. C. M. Li, "A secure test wrapper design against internal and boundary scan attacks for embedded cores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 126–134, Jan. 2012.
[14] C. Clark, "Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust (HOST)*, Anaheim, CA, USA, Jun. 2010, pp. 19–24.
[15] A. Das, Ü. Kocabaş, A. Sadeghi, and I. Verbauwhede, "PUF-based secure test wrapper design for cryptographic SoC testing," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2012, pp. 866–869.
[16] K. Park, S. G. Yoo, T. Kim, and J. Kim, "JTAG security system based on credentials," *J. Electron. Test.*, vol. 26, no. 5, pp. 549–557, Oct. 2010.
[17] A. Das, J. Da Rolt, S. Ghosh, S. Seys, S. Dupuis, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Secure JTAG implementation using schnorr protocol," *J. Electron. Test.*, vol. 29, no. 2, pp. 193–209, Apr. 2013.
[18] E. Valea, M. D. Silva, M.-L. Flottes, G. D. Natale, and B. Rouzeyre, "Encryption-based secure JTAG," in *Proc. IEEE 22nd Int. Symp. Design Diagnostics Electron. Circuits Syst. (DDECS)*, Apr. 2019, pp. 1–6.
[19] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.
[20] K.-J. Lee, C.-A. Liu, and C.-C. Wu, "A dynamic-key based secure scan architecture for manufacturing and in-field IC testing," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 373–385, Jan. 2022.
[21] M. M. Bidmeshki, Y. Zhang, M. Zaman, L. Zhou, and Y. Makris, "Hunting security bugs in SoC designs: Lessons learned," *IEEE Des. Test*, vol. 38, no. 1, pp. 22–29, Feb. 2021.
[22] J. Guan Ooi and K. Horng Kam, "A proof of concept on defending cold boot attack," in *Proc. 1st Asia Symp. Quality Electron. Design*, Jul. 2009, pp. 330–335.
[23] J. Bauer, M. Gruhn, and F. C. Freiling, "Lest we forget: Cold-boot attacks on scrambled DDR3 memory," *Digit. Invest.*, vol. 16, pp. 65–74, Mar. 2016.
[24] S. F. Yitbarek, M. T. Aga, R. Das, and T. Austin, "Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 313–324.
[25] J. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Appelbaum, and E. Felten, "Lest we remember: Cold-boot attacks on encryption keys," *Commun. ACM*, vol. 52, no. 5, pp. 91–98, 2009.
[26] M. Gruhn and T. Müller, "On the practicability of cold boot attacks," in *Proc. Int. Conf. Availability, Rel. Secur.*, Sep. 2013, pp. 390–397.
[27] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer, 2011, ch.7, ch.8, and ch.11.
[28] A. Dehbaoui, J. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of AES," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, Sep. 2012, pp. 7–15.
[29] N. Buard, F. Miller, C. Ruby, and R. Gaillard, "Latchup effect in CMOS IC: A solution for crypto-processors protection against fault injection attacks?" in *Proc. 13th IEEE Int. On-Line Test. Symp. (IOLTS)*, Jul. 2007, pp. 63–70.
[30] A. S. Sichani and W. A. Moreno, "False glitch cells as a countermeasure against power analysis attacks in cryptographic circuits," in *Proc. SoutheastCon*, Apr. 2019, pp. 1–5.
[31] N. Zaidenberg and A. Resh, *Timing and Side-Channel Attacks*. Cham, Switzerland: Springer, 2015.
[32] *Syntacore. SCR1*. Accessed: 2022. [Online]. Available: https://github.com/syntacore/scr1

KUEN-JONG LEE (Fellow, IEEE) received the B.S. degree from the National Taiwan University, Tainan, Taiwan, in 1981, the M.S. degree from The University of Iowa, Iowa, IA, USA, in 1986, and the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA, in 1991. Since 1991, he has been a Faculty Member with the Department of Electrical Engineering, National Cheng Kung University (NCKU), where he is currently a NCKU Distinguished Professor. His current research interests include test compression, silicon debug, fault diagnosis, in-field testing, and hardware security. He has received numerous awards, including the Outstanding Technology Contribution Award of National Applied Research Laboratories, the Best Joint Project Achievement Award of National Science Council, the Macronix Golden Silicon Award, and the Outstanding Technical Achievement Award of IEEE Tainan Section. He also received best paper awards from several conferences/workshops, including the IEEE International Test Conference in Asia, the Workshop on RTL and High-Level Testing (WRTLT), VLSI-CAD Symposium, and the VLSI Test Workshop. He served as the General/Program Chair/Co-Chair for several professional events, including the IEEE Asian Test Symposium, VLSI Design, Automation, Test Symposium, WRTLT, and International Test Conference in Asia. Previously, he also served as the Chair for the Taiwan IC Design Association. He served the Director for the SOC Research Technology Center, NCKU.

ZHENG-YAO LU received the B.S. degree in electrical engineering from the National Sun Yat-sen University, Kaohsiung, Taiwan, in 2018, and the M.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, in 2021. He is currently with MediaTek Inc., Taiwan. His current research interests include hardware security, security of IEEE test standards, and security of testing.

SHIH-CHUN YEH received the B.S. degree in electrical engineering from the National Chung Cheng University, Chiayi, Taiwan, in 2019. He is currently pursuing the M.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan. His current research interests include hardware security, security of IEEE test standards, and security of testing.