Security Against Hardware Trojan Attacks Through a Novel Chaos FSM and Delay Chains Array PUF Based Design Obfuscation Scheme

Mingfu Xue[1,2,3(✉)], Jian Wang[1], Youdong Wang[2], and Aiqun Hu[2]

[1] College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
{mingfu.xue,wangjian}@nuaa.edu.cn
[2] School of Information Science and Engineering,
Southeast University, Nanjing 210096, China
sniperl432@163.com, aqhu@seu.edu.cn
[3] Information Technology Research Base of Civil Aviation Administration
of China, Civil Aviation University of China, Tianjin 300300, China

Abstract. Hardware Trojan has emerged as a major security concern for integrated circuits. This paper presents a novel design obfuscation scheme against hardware Trojan attacks based on chaos finite state machine (FSM) and delay chains array physical unclonable function (PUF). We exploits the pseudo-random characteristics of the *M-sequences* to propose a chaos FSM design method which can generate exponentially many random states and transitions to obfuscate the chip's functional states with low overhead. The chip's functionalities are locked and obfuscated and would not be functional without a unique key that can only be computed by the designer. We also propose a new PUF construction method, named delay chains array PUF (DAPUF), to extract the unique power-up state for each chip which is corresponding to a unique key sequence. We introduce confusions between delay chains to achieve avalanche effects of the PUF outputs. Thus the proposed DAPUF approach can provide large number of PUF instances with high accuracy and reverse-engineering resistant. Through the proposed obfuscation scheme, the designer can control the IC's operation modes (*chaos mode* and *normal mode*) and functionalities, and can also remotely disable the chips when hardware Trojan insertion is revealed. The functional obfuscation prevents the adversary from understanding the real functionalities of the circuit as well as the real rare events in the internal nodes, thus making it difficult for the adversary to insert hard-to-detect Trojans. It also makes the inserted Trojans become invalid since the Trojans are most likely inserted in the *chaos mode* and will be activated only in the *chaos mode*. Both simulation experiments on benchmark circuits and hardware evaluations on FPGA show the security, low overhead and practicality of the proposed method.

Keywords: Hardware Trojan · Design obfuscation · Chaos FSM · Delay chains array PUF

Z. Huang et al. (Eds.): ICCCS 2015, LNCS 9483, pp. 14–24, 2015.
DOI: 10.1007/978-3-319-27051-7_2

1 Introduction

Hardware Trojan attacks have emerged as a major threat for integrated circuits (ICs) [1–3]. A design can be tampered in untrusted design houses or fabrication facilities by inserting hardware Trojans. Hardware Trojans can cause malfunction, undesired functional behaviors, leaking confidential information or other catastrophic consequences in critical systems. Methods against hardware Trojan attacks are badly needed to ensure trust in ICs and system-on-chips (SoCs).

However, hardware Trojans are stealthy by natures which are triggered under rare conditions or specific conditions that can evade post-manufacturing test. Moreover, there are large number of possible Trojan instances the adversary can exploit and many diverse functions of Trojans which makes hardware Trojan detection by logic testing extremely challenging.

Many side-channel signal analysis approaches [4–6] have been proposed to detect hardware Trojans by extracting the parameters of the circuits, e.g. leakage current, transient current, power or delay. However, these methods are very susceptible to process variations and noises. Moreover, the detection sensitivity is greatly reduced for small Trojans in modern ICs with millions of gates. A few regional activation approaches were proposed to magnify Trojan's contributions [7, 8]. However, large numbers of random patterns were applied during detection, while computationally consuming training processes were also used for pattern selection. Besides, these approaches need to alter the original design and add more circuits during design phase which will increase the complexity of the design process and produce a considerable overhead for large designs.

Most of the existing works require golden chips to provide reference signals for hardware Trojan detection. However, obtaining a golden chip is extremely difficult [9]. The golden chips are supposed to be either fabricated by a trusted foundry or verified to be Trojan-free through strict reverse engineering. Both methods are prohibitively expensive. In some scenarios, the golden chips even don't exist, e.g., if the mask is altered at the foundry. Recently, a rare few methods are proposed to detect HT without the golden chips by using self-authentication techniques [9–11]. However, these methods are not without limitations [12]. They always need expensive computations, sophisticated process variation models and a large number of measurements to ensure accuracy for large deigns.

R.S. Chakraborty et al. propose an application of design obfuscation against hardware Trojan attacks [13, 14]. However, the key sequence is the same for all the chips from the same design which makes the approach itself vulnerable to various kinds of attacks, e.g. the fab can use the key to unlock all the chips and then tamper or overbuild the chips arbitrarily. The fab or the user can also release the key sequence of the design in the public domain which makes the chip vulnerable to reverse-engineering attacks. Besides, the procedure of determination of unreachable states is also time consuming and computationally complex.

This paper presents a novel design obfuscation scheme against hardware Trojan attacks based on chaos finite state machine (FSM) and delay chains array physical unclonable function (PUF). The obfuscation scheme is realized by hiding and locking the original FSM using exponentially many chaotic states. We exploits the pseudo-random

characteristics of the *M-sequences* [15] to propose a chaos FSM design method which can generate many random states and transitions with low overhead. The chip is obfuscated and would not be functional without a unique key that can only be computed by the designer. We also propose a new PUF construction method, named delay chains array PUF (DAPUF), to extract the unique signature of each chip. Each chip has a unique key sequence corresponding to the power-up state. We introduce confusions between delay chains to achieve avalanche effects of the PUF outputs. Thus the proposed DAPUF can provide large number of PUF instances with high accuracy and reverse-engineering resistant. Through the proposed scheme, the designer can control the IC's operation modes (*chaos mode* and *normal mode*) and functionalities, and can also remotely disable the chips when hardware Trojan is revealed. The functional obfuscation prevents the adversary from understanding the real functionalities of the circuit as well as the real rare events in the internal nodes, thus making it difficult for the adversary to insert hard-to-detect Trojans. It also makes the inserted Trojans become invalid since the Trojans are most likely inserted in the *chaos mode* which will be activated only in the *chaos mode*. Both simulation experiments on benchmark circuits and hardware evaluations on FPGA platforms show the security, low overhead and practicality of the proposed method.

2 Overall Flow

First, the designer exploits the high level design description to form the FSM. Then, the original FSM is modified based on the nonlinear combination of the *M-sequences* to generate the chaos FSM which adds exponentially many random states and transitions to obfuscate the functional states. Then, the DAPUF module is constructed to generate the unique signature of each IC. After the subsequent design procedures are performed, the foundry will receive necessary information to fabricate the chips. Each chip is locked upon fabrication. In the proposed chaos FSM construction scheme, up to 2^{20} chaotic states are added, so the functional states of the chip are submerged in a large number of chaotic states. When the chip is powered on, it will be a great probability to fall into the chaotic states thus is non-functional, named *chaos mode*.

Then, the foundry applies the set of challenge inputs to the PUF unit on each chip and sends the PUF response to the designer. Each PUF response is corresponding to a unique power-up state. The designer who masters the *M-sequences* and transition table is the only entity who can compute the passkey to make the chip enter into functional states. After applying the passkey, the locked IC will traverse a set of transitions and reach to the functional reset state. Then the IC is unlocked and becoming functional, called *normal mode*.

Figure 1 shows the proposed design obfuscation scheme. In this sample, there are 8 states in the original FSM. 24 chaotic states are generated adding to the original FSM. 4 black hole states are introduced to enable the designer to disable the chips when needed. The DAPUF module generates a unique identifier for each chip corresponding to a unique power-up state. The power-up state is most likely to fall into the chaos FSM making the chip nonfunctional until an IC-unique passkey is applied. The passkey can make the chip traverse through a set of transitions and reach to the functional reset state.

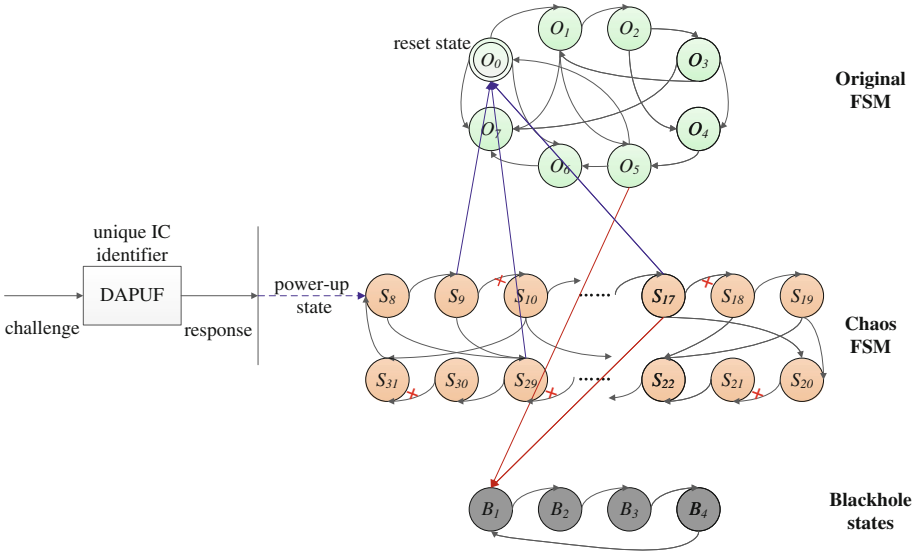

Fig. 1. The chaos FSM and DAPUF based design obfuscation scheme against hardware Trojans

3 Chaos FSM Construction

We define the following symbols for discussion:

$O_0, O_1, O_2 \ldots, O_{n_o-1}$	The original FSM states (O_0 is the reset state)
$S_{n_o}, S_{n_o+1}, \ldots, S_{n_0+2^L-1}$	The added chaotic states
K_{ij}	The required key for traversing from state i to state j
C_{ij}	The connected transition path from state i to state j
D_{ij}	The disconnected path from state i to state j

In this paper, we nonlinearly combine the outputs of the *M-sequences* to generate exponentially many random states and transitions with low overhead. First, we exploit the *M-sequences* to construct a circular state transition graph (STG). A simple illustration is shown in Fig. 2. For simplicity, there are only seven chaotic states and one reset state in this illustration while the original FSM are omitted. The state transition paths are denoted as $C_{12}, C_{23} \ldots, C_{ii+1}$, where i is the i_{th} state of the chaotic states generator. There are n_o states in the original FSM of the design. We assume the order of the chaotic states generator has a large value L to ensure $2^L >> n_o$, thus there are $n_o + 2^L$ states in total. When the chip is powered on, it only has a probability of $\frac{n_o}{n_o+2^L}$ to fall into the functional states. Obviously, this probability is nearly zero.

Second, we perform chaos process of the STG which needs to satisfy the following principles: (1) for any chaotic state S_k, there is at least one transition path C_{ak} to ensure that all the chaotic states are reachable, where a is an arbitrary value; (2) there is a set of transition paths $C_{pi_1}, C_{i_1 i_2}, \ldots, C_{i_m q}$ to ensure that any two chaotic states S_p, S_q can be

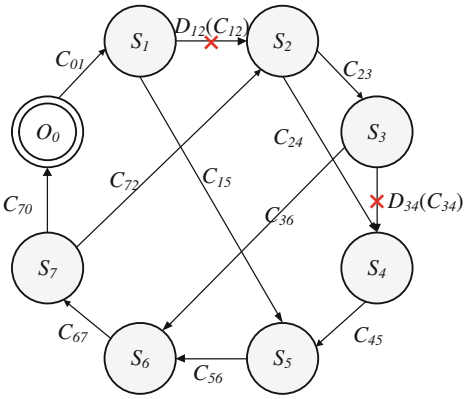

Fig. 2. The state transition graph of the chaos FSM

connected; (3) there is at least one chaotic state can reach the reset state O_0 through one transition C_{iO_0}.

The first step of the chaos process is to add new random transition paths to the existing circular STG. In Fig. 2, that is, adding new C_{ij} where $j \neq i+1$, such as $C_{15}, C_{24}, C_{36}, C_{72}$. The second step is to disconnect several transition paths randomly, that is, replacing C_{ij} with D_{ij}, such as D_{12} and D_{34}. Each time a D_{ij} is added, we should check the connectivity between state i and state j, which means, there is at least one transition path $C_{ii_1}, C_{i_1i_2}, \ldots, C_{i_mj}$ to connect state i and state j. Otherwise, one need to add the corresponding $C_{i_ni_m}$ to ensure the connectivity.

Suppose the power-up state is S_n, there is one transition path $C_{S_ni_1}, C_{i_1i_2}, \ldots, C_{i_mO_0}$ to ensure that the power-up state and the reset state O_0 is connected. We can get $O_0 = F\{F\ldots\{F\{S_n, C_{ni_1}, K_{S_ni_1}\}C_{i_1i_2}, K_{i_1i_2}\}\ldots C_{i_mo_0}, K_{i_mO_0}\}$. To unlock the chip, the transition key $K_{S_ni_1}, K_{i_1i_2}, \ldots, K_{i_mO_0}$ is needed. The designer can increase the number of required transition keys to increase the complexity of the key space. Since there is a huge number of chaotic states, adding one transition path will exponentially increase the key space while the added overhead is negligible.

In the chaos FSM construction, the designer can also create black hole states to disable the chip when needed [16]. Black hole states are the states that can't return to functional states regardless of the input sequences. When the fab provide the power-up state to the designer asking for the unlock passkeys, the designer may deliberately provide a specific input sequence to make the chip enter into black hole states rather than reset state once hardware Trojan insertion is revealed.

4 Delay Chains Array PUF Implementation

The changes of chip's operating conditions, including temperature and voltage, etc., will affect physical characteristics of the chip, thus affecting the PUF. The traditional solution is to add redundancy or use the error correcting codes. Two advantages of the

proposed DAPUF construction method compared to traditional PUF approaches are: (1) exploiting relative value algorithm rather than absolute value to analyze the results, thus can overcome the impact of chips' working conditions; (2) introducing confusions between delay chains which can result in avalanche effects of the PUF output, thus can generate large number of PUF instances with reverse-engineering resistant.

Zou J. et al. use the delay chain to obtain FPGA gate-level timekeeping and the precision of such delay chain can reach to 1 *ns* [17]. In this paper, we propose a PUF implementation approach based on delay chains array, named DAPUF.

First, let's consider the one-dimensional delay chain PUF (DPUF). As shown in Fig. 3, the DPUF is constructed by cascading the buffers and the D flip-flops. We use symbol *start* to represent the high level of the D flip-flop's input signal, while using symbol *end* to represent the high level of the buffer's input signal. The delay of the buffer is denoted as t_B and the delay of the D flip-flop is denoted as t_D, thus the delay difference between the D flip-flop and the buffer is $\Delta t = t_D - t_B$ (note that $t_D > t_B$). At the beginning, both the input of the D flip-flop chain and the input of the buffer chain are at low level. When the first high level signal *start* comes, this high level signal will propagate through the D flip-flop chain. After a certain delay T, the *end* signal will come and propagate along the buffer chain. Because of the delay difference Δt between the D flip-flop and the buffer, when propagating through every stage, the propagation time difference between the *start* signal and the *end* signal will reduce Δt. At a certain time, the *end* signal will catch up with and surpass the *start* signal.

Due to the intrinsic process variations, each chip has different t_B and t_D. Even within one chip, the delays of different D flip-flops (or buffers) also have tiny differences. Thus, we can define t_{Bn} as the inherent delay of the n_{th} buffer while t'_{Bn} is the difference between t_{Bn} and the expected mean value t_B. Similarly, we define t_{Dn} as the inherent delay of the n_{th} D flip-flop while t'_{Dn} is the difference between t_{Dn} and the expected mean value t_D. Denote σ as the random noise. Generally, Δt_n, σ satisfy the normal distribution with a certain expectation, thus we can get $t_{Bn} = t_B + t'_{Bn} + \sigma$ and $t_{Dn} = t_D + t'_{Dn} + \sigma$ under noise condition. Further more, we can get:

$$\sum_{n=1}^{k_B} (t_B + t'_{Bn} + \sigma) + T = \sum_{n=1}^{k_D} (t_D + t'_{Dn} + \sigma) \tag{1}$$

In which, k_B and k_D are the number of propagated buffers and the number of propagated D flip-flops when the *end* signal catches up with the *start* signal. We can obtain the tuple (k_B, k_D) through the time discriminator. This tuple (k_B, k_D) is unique for each chip and physically stochastic.

We define m groups of input signals as $\{X_1, Y_1, T_1\}, \ldots, \{X_m, Y_m, T_m\}$, where $X_1, X_2, \ldots, X_m$ represent the high level input signals of the D flip-flop, $Y_1, Y_2, \ldots, Y_m$ represent the high level input signals of the buffer, and $T_1, T_2, \ldots, T_m$ represent the time difference between input signal X_g and Y_g, $1 \le g \le m$. Let's define:

$$(k_{gB}, k_{gD}) = W\{X_g, Y_g, T_g\} \tag{2}$$

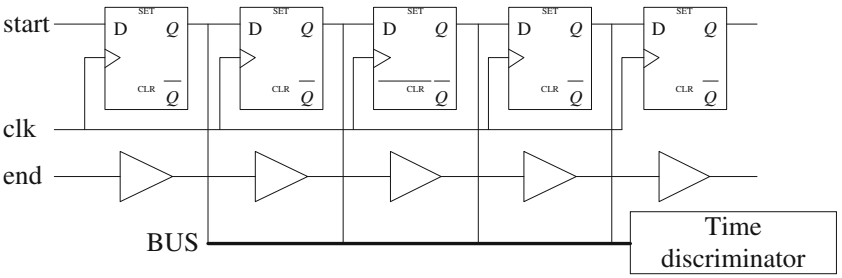

Fig. 3. The structure of the one-dimensional delay chain PUF (DPUF)

where $W\{X_g, Y_g, T_g\}$ represents the delay chain's response to the g_{th} group of input signals. k_{gB} and k_{gD} are the number of propagated buffers and the number of propagated D flip-flops when the signal Y_g catches up with the signal X_g. Thus, we can obtain a group of tuples $\{(k_{1B}, k_{1D}), (k_{2B}, k_{2D}), \ldots, (k_{mB}, k_{mD})\}$ based on the response of the DPUF. Then, we can extract the PUF characteristic vector:

$$\vec{E} = [(k_{2B}, k_{2D}) - (k_{1B}, k_{1D}), (k_{3B}, k_{3D}) - (k_{2B}, k_{2D}), \ldots, (k_{mB}, k_{mD}) - (k_{(m-1)B}, k_{(m-1)D})] \tag{3}$$

In which, $\vec{E}$ is the power-up state of the chip. Generally, one can consider that the working conditions of different regions within one chip are the same. We use relative value for characterization can eliminate the effect of working conditions.

However, there are some shortages if we only use one-dimensional delay chain PUF (DPUF). Since the precision is only 1 *ns*, the physical unique characteristic of each chip is quantified by this precision. Thus, the number of possible PUF instances is reduced due to this limitation. However, less PUF instances may cause the collision problem, which means two chips with different characteristics may have the same DPUF after quantified. Therefore, this paper proposes delay chains array PUF (DAPUF) which can magnify the delay differences thus significantly increase the number of PUF instances.

The block diagram of the proposed DAPUF is shown in Fig. 4. $I_1, I_2, \ldots, I_{2h}$ represent the rising edges of the input signals, in which, h is the number of delay chains. We add confusions between delay chains by the configuration of Con_{ij}. Con_{ij} represents whether the node in row i column j is connected. In other words, it satisfies: $Con_{ij} = \begin{cases} 1\, connected \\ 0\, disconnected \end{cases}$. Obviously, the coming moment of the rising edge in each node $P_{i_0 j_0}$ depends on the values of all $Con_{ij} (i < i_0, j < j_0)$. It's illustrated in a simple example, as shown in Fig. 4, since $Con_{11}, Con_{22}, Con_{31}$ are 1, the time of the rising edge in node P_{23} is affected by I_1, I_2 and I_3. The time discriminator will detect the first (k_B, k_D) rising edges of each delay chain, and extract the corresponding characteristic vector $\vec{E}$. The final DAPUF characteristic vector is $\{\vec{E}_1, \vec{E}_2, \cdots, \vec{E}_h\}$. Obviously, the configurations of the Con_{ij} significantly magnify the delay differences between delay chains and greatly increases the number of PUF instances with avalanche effects.

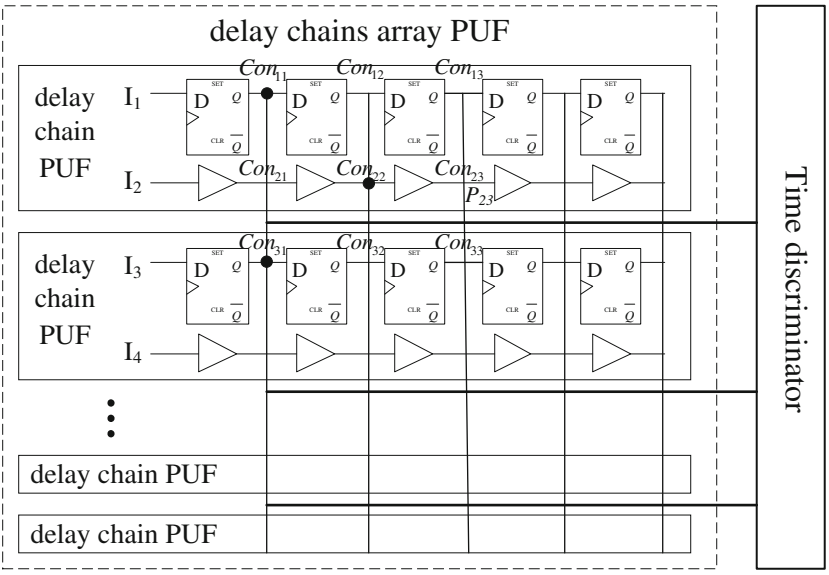

Fig. 4. The structure of the proposed delay chains array PUF (DAPUF)

5 Simulation and Hardware Implementation Evaluation

Simulation experiments are performed on a set of ISCAS89 benchmark circuits while the hardware evaluations are based on the CC1200 wireless transmitter control circuit implemented on Altera EP2C8Q208C8 FPGA platforms.

Figure 5 shows the test waveforms of the one-dimensional DPUF. The rising edges are the *start* signal propagating in the D flip-flop chain and the *end* signal in the buffer chain, respectively. Obviously, at a certain moment, the *end* signal catches up with and surpasses the *start* signal. We can get the tuple (k_B, k_D) of this moment through the time discriminator.

There are m groups of input signals in each delay chain thus there are m groups of (k_B, k_D) and we can calculate $\vec{E}$ of the DPUF. For the delay chains array PUF (DAPUF), there are h delay chains in total and we can obtain $m \cdot h$ groups of (k_B, k_D) through the time discriminator and we can calculate the characteristic vector $\{\vec{E_1}, \vec{E_2}, \cdots, \vec{E_h}\}$. The characteristic vector is corresponding to the unique power-up state of the chip.

Figure 6 shows the test waveforms of the proposed obfuscation scheme. Note that, the state transition waveform is only a simple illustration to show the chaotic states transitions, which contains only seven chaotic states and one reset state. The real chaos FSM is much bigger than this sample thus is not convenient to present in this waveform. As shown in the figure, when the chip is powered on, it falls into a chaotic state

and will traverse in the chaos FSM. The states of the chaos FSM are chaotic and unpredictable. When a wrong passkey is applied, the module starts to perform passkey identification. Since this is a wrong passkey, after the passkey identification, the chip continues to traverse in the chaos FSM until the correct passkey (00010100100100 001100111100000001) is applied. Then the module performs passkey identification again. The chaos FSM will traverse to the reset state "000" through several transition paths and the chip is unlocked. Then, the chip will enter into the *normal mode*.

Table 1 shows the overhead of the proposed scheme on ISCAS89 benchmark circuits as well as the CC1200 wireless transmitter control circuit. The overhead has two main parts, the DAPUF construction and the chaos FSM implementation. The DAPUF module in our experiments consists of 28 delay chains with each delay chain contains 10 buffers and 8 D flip-flops. The results show that this DAPUF module costs 156 logic elements (LE). We use a large scale DAPUF in the experiment to ensure obtaining massive PUF instances and strong avalanche effects. The overhead of the chaos FSM implementation mainly comes from the polynomial coefficients of the *M-sequences*, the added transition paths table, and the storage of the passkeys. In this paper, there are 2^{20} chaotic states and the unlocking process needs 4 passkeys with each key is 128 bit long. Note that, in related works, the PUF weren't included in the evaluations. If we remove the overhead of the DAPUF, the overhead of the scheme is rather small. It is shown that when the circuit's size becomes larger, the overhead becomes much smaller. For modern circuits with millions of gates, the overhead of the scheme is negligible.

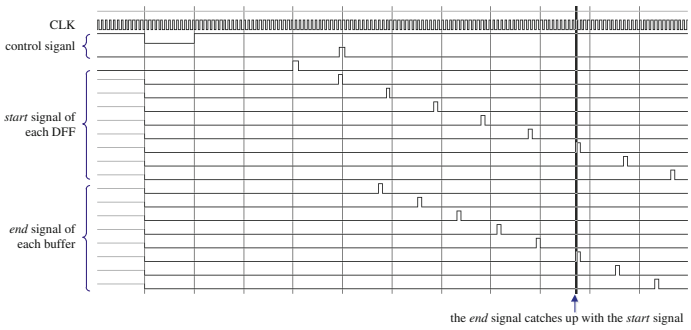

Fig. 5. The test waveform of the DPUF

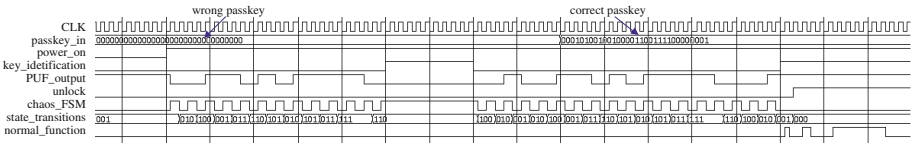

Fig. 6. The test waveform of the proposed obfuscation scheme

Table 1. Overhead of the proposed obfuscation scheme

Circuit	Orig. LE	Overall added LE	Overall overhead (%)	Added LE without PUF	Overhead without PUF (%)
s526	35	232	662.9	76	217.1
s1494	175	248	141.7	92	52.6
s5318	417	226	54.2	70	16.8
s15850	796	256	32.2	100	12.6
s38584	2207	252	11.4	96	4.3
CC1200 cir.	5508	266	4.8	110	2

6 Conclusion

We have developed a novel design obfuscation scheme against hardware Trojan attacks based on chaos FSM and DAPUF. We propose a chaos FSM design method which can generate many random states to obfuscate the original FSM. We also propose a new PUF construction method obtaining large number of PUF instances with avalanche effects. Through the proposed scheme, the designer can control the IC's operation modes and functionalities, and can remotely disable the chips. The obfuscation prevents the adversary from understanding the real function and the real rare events of the circuit, thus making it difficult to insert Trojans. It also makes the inserted Trojans become invalid since they are most likely inserted in the *chaos mode* which will be activated only in the *chaos mode*.

Acknowledgments. This work is supported by Natural Science Foundation of Jiangsu Province, Chinese Postdoctoral Science Foundation, Jiangsu Province Postdoctoral Science Foundation, and Open Project Foundation of Information Technology Research Base of Civil Aviation Administration of China (NO. CAAC-ITRB-201405).

References

1. Bhunia, S., Hsiao, M.S., Banga, M., Narasimhan, S.: Hardware Trojan attacks: threat analysis and countermeasures. Proc. IEEE **102**(8), 1229–1247 (2014)
2. Rostami, M., Koushanfar, F., Karri, R.: A primer on hardware security: models, methods, and metrics. Proc. IEEE **102**(8), 1283–1295 (2014)
3. Bhunia, S., Abramovici, M., Agarwal, D., Bradley, P., Hsiao, M.S., Plusquellic, J., Tehranipoor, M.: Protection against hardware Trojan attacks: towards a comprehensive solution. IEEE Des. Test **30**(3), 6–17 (2013)
4. Agrawal, D., Baktır, S., Karakoyunlu, D., Rohatgi, P., Sunar, B.: Trojan detection using IC fingerprinting. In: Proceedings of IEEE Symposium on Security and Privacy (SP 2007), pp. 296–310. Berkeley, California, 20–23 May 2007
5. Nowroz, A.N., Hu, K., Koushanfar, F., Reda, S.: Novel techniques for high-sensitivity hardware Trojan detection using thermal and power maps. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **33**(12), 1792–1805 (2014)

6. Potkonjak, M., Nahapetian, A., Nelson, M., Massey, T.: Hardware Trojan horse detection using gate-level characterization. In: 46th Design Automation Conference (DAC 2009), pp. 688–693. San Francisco, California, USA, 26–31 July 2009
7. Banga, M., Hsiao, M.S.: A region based approach for the identification of hardware Trojans. In: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008), pp. 40–47. Anaheim, CA, 9–9 June 2008
8. Wei, S., Potkonjak, M.: Scalable hardware Trojan diagnosis. IEEE Trans. Very Large Scale Integr. VLSI Syst. **20**(6), 1049–1057 (2012)
9. Davoodi, A., Li, M., Tehranipoor, M.: A sensor-assisted self-authentication framework for hardware Trojan detection. IEEE Des. Test **30**(5), 74–82 (2013)
10. Wei, S., Potkonjak, M.: Self-consistency and consistency-based detection and diagnosis of malicious circuitry. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(9), 1845–1853 (2014)
11. Xiao, K., Forte, D., Tehranipoor, M.: A novel built-in self-authentication technique to prevent inserting hardware Trojans. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(12), 1778–1791 (2014)
12. Cao, Y., Chang, C.-H., Chen, S.: A cluster-based distributed active current sensing circuit for hardware Trojan detection. IEEE Trans. Inf. Forensics Secur. **9**(12), 2220–2231 (2014)
13. Chakraborty, R.S., Bhunia, S.: Security against hardware Trojan through a novel application of design obfuscation. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 113–116. San Jose, California, 2–5 November 2009
14. Chakraborty, R.S., Bhunia, S.: Security against hardware Trojan attacks using key-based design obfuscation. J. Electron. Test. **27**, 767–785 (2011). Springer
15. Klapper, A., Goresky, M.: 2-Adic shift registers. In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809. Springer, Heidelberg (1994)
16. Alkabani, Y.M., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: Proceedings of USENIX Security Symposium, pp. 291–306. Berkeley, CA, USA (2007)
17. Zou, J., Yu, W., Chen, Q.: Resolution of time-interval measurement based on chain delay difference of FPGA (in Chinese). Opt. Optoelectron. Technol. **12**(5), 43–45 (2014)

http://www.springer.com/978-3-319-27050-0

Cloud Computing and Security
First International Conference, ICCCS 2015, Nanjing, China, August 13-15, 2015. Revised Selected Papers
Huang, Z.; Sun, X.; Luo, J.; Wang, J. (Eds.)
2015, XII, 562 p. 211 illus. in color., Softcover
ISBN: 978-3-319-27050-0