An AES chip with DPA resistance using hardware-based random order execution*

Yu Bo(俞波)†, Li Xiangyu(李翔宇), Chen Cong(陈聪), Sun Yihe(孙义和), Wu Liji(乌力吉), and Zhang Xiangmin(张向民)

Tsinghua National Laboratory for Information Science and Technology, Institute of Microelectronics, Tsinghua University, Beijing, 100084, China

Abstract: This paper presents an AES (advanced encryption standard) chip that combats differential power analysis (DPA) side-channel attack through hardware-based random order execution. Both decryption and encryption procedures of an AES are implemented on the chip. A fine-grained dataflow architecture is proposed, which dynamically exploits intrinsic byte-level independence in the algorithm. A novel circuit called an HMF (Hold-Match-Fetch) unit is proposed for random control, which randomly sets execution orders for concurrent operations. The AES chip was manufactured in SMIC 0.18 μm technology. The average energy for encrypting one group of plain texts (128 bits secrete keys) is 19 nJ. The core area is 0.43 mm^2. A sophisticated experimental setup was built to test the DPA resistance. Measurement-based experimental results show that one byte of a secret key cannot be disclosed from our chip under random mode after 64000 power traces were used in the DPA attack. Compared with the corresponding fixed order execution, the hardware based random order execution is improved by at least 21 times the DPA resistance.

Key words: differential power analysis; advanced encryption standard; dataflow; asynchronous design

DOI: 10.1088/1674-4926/33/6/065009 **EEACC:** 1265Z

1. Introduction

In recent years, security has become a major requirement for an increasing number of embedded systems, such as wireless handsets, smart cards and sensor networks. Side channel attacks (SCAs) have been developed to hack into cryptographic devices through analyzing information leaked from hardware implementations, such as power consumption[1], computing time[2], and electromagnetic radiation. Since the power consumption of embedded devices can be easily measured by existing apparatus, power analysis has become a major SCA method, and threatens to embedded systems, especially portable devices.

Simple power analysis (SPA)[1] guesses the secret keys according to one single power trace of the devices. It is an effective SCA method only when the power traces are dependent on secret keys, so it is not suitable for a symmetric algorithm (e.g. AES) of which power traces are independent of secret keys. Differential power analysis (DPA)[1] deduces the secret keys on the basis of the correlation existing between the power consumption and internal states of the cryptographic device. It measures multiple power traces from encryption or decryption processes, and deduces secret keys by statistic analysis. Owing to its effectiveness, DPA has become a major threat to devices using symmetric encryption algorithms.

Several counter-measures have been developed to resist DPA, including constant power circuits[3–5], masking techniques[6], random execution[7, 8] and decoupling techniques[9]. For counter-measures utilizing constant power circuits, masking and decoupling, extra circuits and operations are needed to make circuits consume constant power, mask intermediate results and implement decoupling capacitors. As a result, a large overhead of extra hardware resources and power consumption are inevitably introduced. Random execution[7, 8, 10], which randomly changes the execution time of each operation, resists DPA through changing the certain relationship between power and internal state at a particular time to a random one. In general, implementing random execution does not need redundant logic or extra mask operations, so random execution is more economic in terms of hardware and power consumption.

Random execution has been employed by programmable processors for DPA resistance. Rivain[7] proposed to randomly insert dummy instructions in software programs to achieve random execution. For such kinds of random execution, the execution order of operations is fixed and the power signature at each possible execution time is only determined by the data. May[8] proposed to utilize a superscalar architecture to achieve random execution through out-of-order execution of parallel instructions. For random executions achieved by out-of-order execution, the power signature is determined not only by the data but also by the execution order. Consequently, out-of-order execution introduces more uncertainty in the power profile, and thus improves the DPA resistance. The DPA resistance of random order execution is determined by the number of parallel operations[11] that can be randomly executed, so fully exploiting intrinsic parallelism in algorithms is critical. However, for cryptographic processors[8] using random order execution, owing to serially composed programs it is difficult to fully exploit intrinsic instruction-level parallelism, even though extremely complicated circuits for checking dependency have been em-

* Project supported by the National Natural Science Foundation of China (No. 61006021) and the Beijing Natural Science Foundation (No. 4112029).

† Corresponding author. Email: yu-b06@mails.tsinghua.edu.cn

Received 5 December 2011, revised manuscript received 11 January 2012

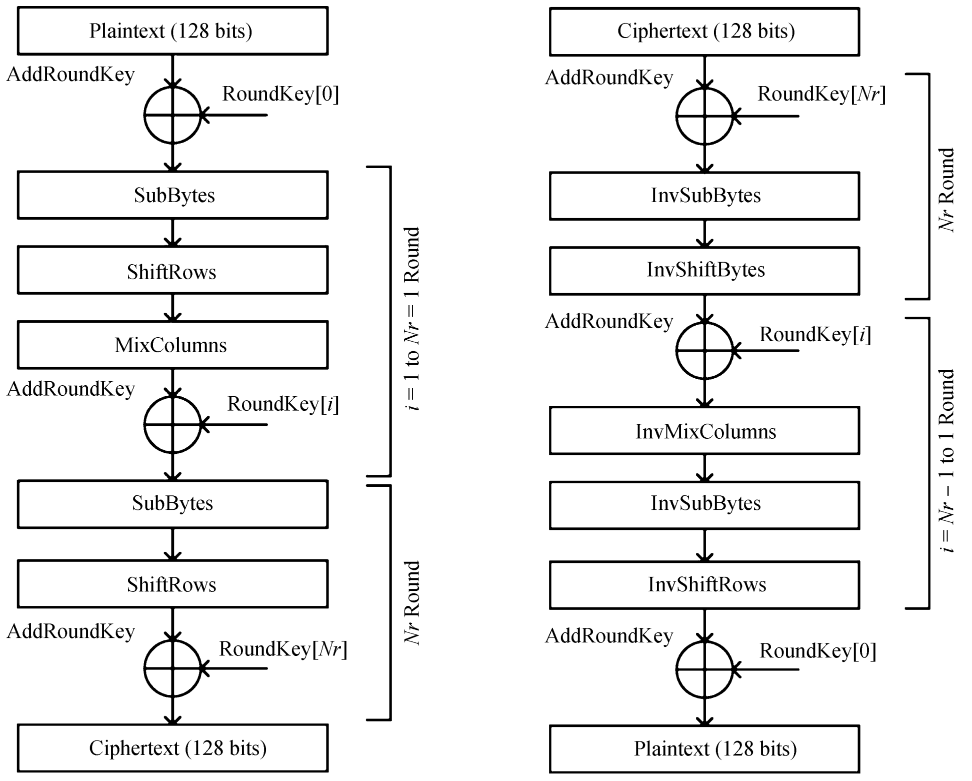


Fig. 1. The (a) encryption and (b) decryption procedure of an AES algorithm, where Nr can be 10, 12 or 14.

ployed.

In this paper, we propose a method for dedicated hardware to utilize random order execution to improve DPA resistance, which can lead to implementations with low hardware cost, low energy consumption and high DPA resistance. An AES that is the most popular symmetric encryption algorithm is taken as an example to illustrate our method. A fine-grained, dataflow-based architecture is proposed, in which operation-level parallelism in the algorithm is fully exploited. A novel circuit structure called a Hold-Match-Fetch (HMF) unit is proposed to randomly dispatch operands to operation units. Asynchronous pipelines working in the data-driven manner are employed in line with the dataflow structure.

The remainder of the paper is organized as follows. Section II briefly introduces an AES algorithm and random order execution. Section III presents the architecture of the chip and the HMF unit. Section IV describes the experimental setup, the method for evaluating DPA resistance and evaluation results. Section V concludes the paper.

2. Preliminary

2.1. AES algorithm

An AES algorithm is a block-based cipher. Data block size is fixed to 128 bits and the key size can be 128 bits, 192 bits and 256 bits. The 128 bits data block, namely the state, is organized as a 4×4 matrix of bytes.

The encryption process of an AES algorithm is shown in Fig. 1(a). After the first AddRoundKey operation, the round transformation including the SubBytes, the ShiftRows, the MixColumns and the AddRoundKey, is applied to the state $Nr-1$ times. Nr is 12, 14 or 16 corresponding to 128 bits, 192 bits and 256 bits keys respectively. The Nr-th round transformation does not need the MixColumns operation. For the round transformation, the SubBytes is a nonlinear substitution that updates each byte of the state using a substitution table, namely an S-box. ShiftRows is a circular shifting operation applied to each byte of rows in the state. The offset of shifting on each sub-state is determined by the row number. The MixColumns operation is an invertible linear transformation in $GF(2^8)$, in which each column of the state is multiplied by a matrix. The AddRoundKey operation combines each byte of the state with the corresponding RoudnKey using bitwise XOR logic. The decryption algorithm is shown in Fig. 1(b). The InvSubBytes, the InvShiftRows, the InvMixColumns and the InvAddRoundKey are inverse operations to corresponding transformations in the encryption process.

The RoundKeys used in encryption and decryption are generated by the key expansion procedure[12]. For the AES algorithm, the initial key consists of Nk words. Nk can be 4, 6, 8. The key expansion algorithm uses initial keys to generate $Nr + 1$ RoundKeys. Each RoundKey is 4×4 bytes.

2.2. Random order execution

The DPA resistance of random order execution is determined by two factors[11], (1) the number of parallel operations that can be randomly executed, and (2) the most chance of an operation to be executed at each possible execution time. Quantitatively discussing the DPA resistance of an out-of-order execution is beyond the scope of this paper. In general, compared with certain order execution, the DPA resistance can increase $O(N^2)$ times if N operations can be uniformly and randomly

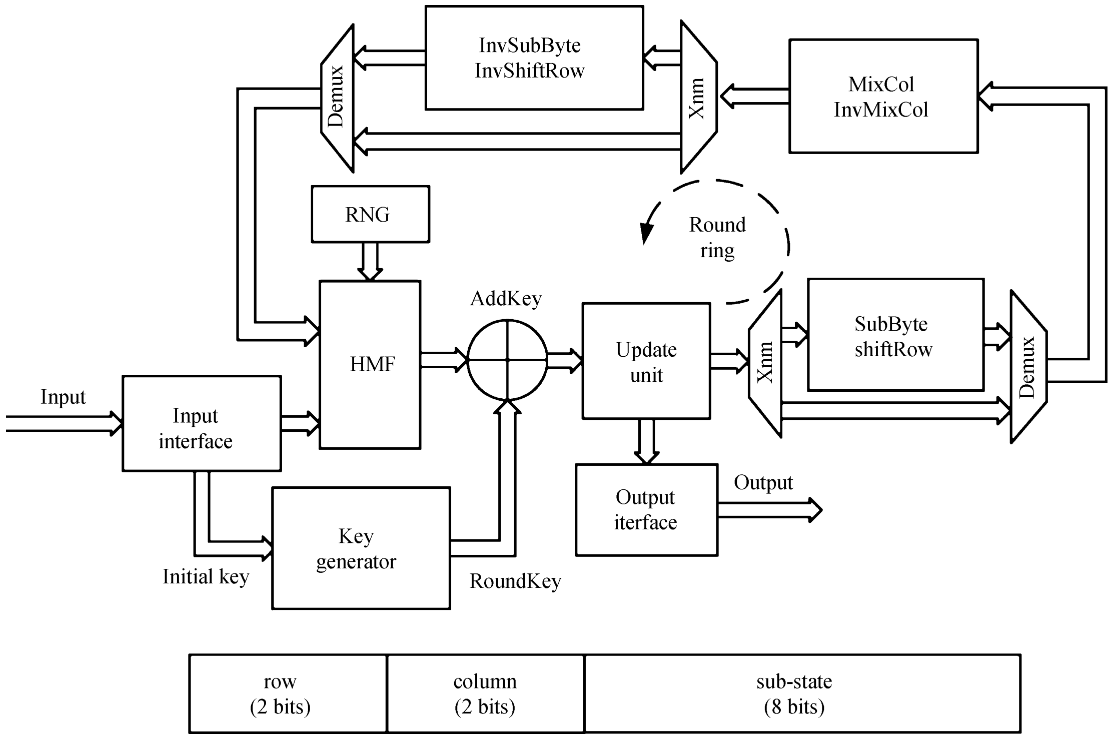


Fig. 2. The hardware architecture (upper) and the token format (lower).

executed[11].

In order to improve the DPA resistance of random execution, the intrinsic data independence and parallelism of an algorithm should be fully exploited, and the number of parallel operations that can be randomly executed should be maximized. In the case of an AES algorithm, AddRoundKey, SubBytes and ShiftRows are defined to perform operations at byte level. 16 bytes of a state can be independently processed by these operations. Although Mixcolumns involves linear multiplications between columns of a state and a constant matrix, it can be decomposed into a set of independent byte-grained multiplications and additions. As a result, the AES algorithm can be performed at byte-level. For each round transformation, 16 byte-grained operations can be executed in any order, and the results will not be affected by random execution. In order to sufficiently utilize the intrinsic data independence in AES and improve DPA resistance, our AES ASIC was designed to perform operations at byte level.

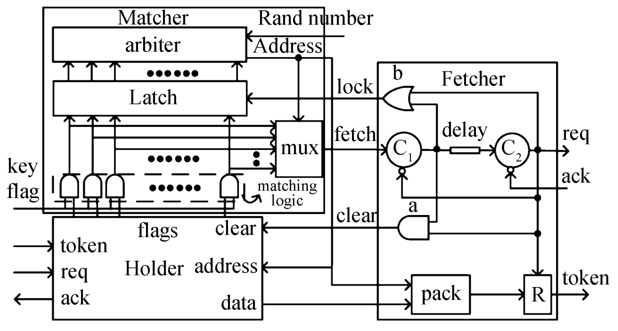
3. Hardware design

3.1. Dataflow-based architecture

Dataflow architectures[13] execute operations according to data-flow graphs (DFG) of algorithms, which consist of a set of nodes and arcs representing operations and data dependency between operations respectively. For dataflow architectures, operations are executed in a data-driven way that operations can be executed only when all the inputs are available. It does not explicitly specify the execution order of operations, in which all the ready operations that satisfy the executing condition can be performed in parallel or in any order. Therefore, dataflow architectures can naturally exploit parallel operations at run time, and are more suitable for random scheduling than control-flow computing.

Inspired by dataflow computing, a dataflow architecture that implements an AES algorithm is proposed as shown in Fig. 2. It consists of input and output interfaces, Round Ring and Key Generator. The input interface sends data and keys to the Round Ring and Key Generator, which implement round transformation and key expansion respectively. The Key Generator computes round keys, which are stored inside the Key Generator and supplied to the Round Ring.

Operations of the round transformations are implemented by corresponding hardware units in the Round Ring. These hardware components are all fine-grained and work in the data-driven way, and are connected by channels (lines with arrows in Fig. 2) according to the data-flow graph of round transformation to form a ring structure. The ring structure performs each loop of round transformation. 'Mux' and 'demux' are switchers that select data paths for encryption and decryption process. The HMF unit implements random order dispatching in the dynamic data-flow way. At the very beginning of the working stage, all the sub-states for first round transformation are available and stored in the HMF unit. It then randomly dispatches these sub-states into the Round Ring. The dispatching order is determined by random numbers generated by an on-chip random number generator (RNG). During the working stage, sub-states are stored back to the HMF unit when one round transformation is accomplished. The HMF unit keeps checking if there are sub-states ready for the next round transformation. The available sub-states for the next round transformation are also randomly dispatched. If the round transformation of a sub-state is accomplished, the 'update unit' outputs the sub-state to the output interface. Otherwise it passes the sub-state to the next unit in the Round Ring. After the encryption or decryp-



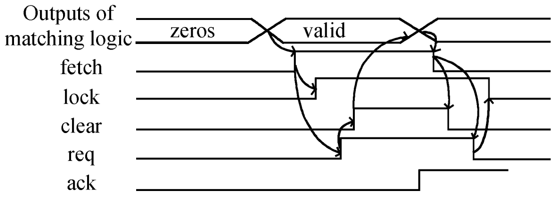


Fig. 3. The structure of (a) the HMF unit and (b) the time sequence diagram of the fetching process.

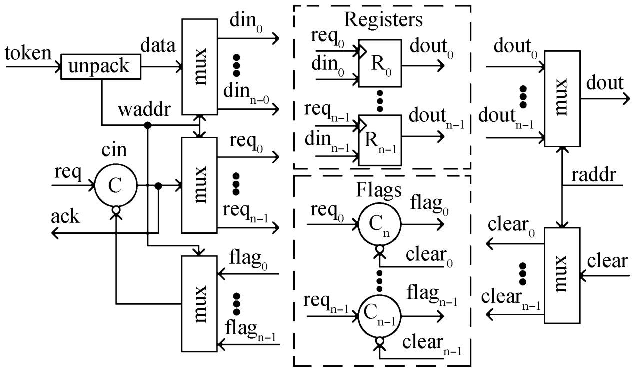


Fig. 4. The structure of the Holder.

tion finishes, all the sub-states are stored in the output interface, which outputs results serially in the fixed order.

Asynchronous circuits were employed to implement the dataflow architecture. It works in a data-driven way[14], which is in line with the behavior of dataflow architecture. Owing to the absence of the clock signal, it is difficult to determine the exact execution time of an operation in an asynchronous design. The uncertainty of execution time makes accurate power analysis of a particular operation become more difficult, and thus a DPA attack needs to make more effort to hack into the chip. In our design, the channels between components are implemented by the four-phase bundle-data handshake protocol, which achieves communication and distributed control. Compared with other commonly used asynchronous circuits, such as quasi delay insensitive (QDI) circuits in dual rail or m-of-n codes[14], asynchronous circuits based on bundle data protocol use conventional data processing elements that can be designed and implemented using traditional EDA tools and design flows for standard cells[15]. In addition, QDI based computing elements in dual rail or m-of-n codes require redundant circuits for coding the completion signal for handshaking, which increase the area and power consumption of the design. For these reasons, circuits based on bundled data handshake protocols were used in our design.

3.2. Token format

Unlike synchronous designs that can utilize an FSM (finite state machine) for global control, our asynchronous dataflow architecture should employ distributed control mechanism due to the lack of a global clock. In the architecture, control information is packed with data to form tokens and is used as tags for each token. These tags are used to facilitate distributed computing and differentiate tokens. For the proposed architecture, position information of each byte in the state is required to perform AddRoundKey, SubBytes, ShiftRows and MixColumns. As a result, position information is packaged with data to form tokens in the architecture. The format of tokens in the architecture is shown in Fig. 2. The position of each sub-state is determined by the 2-bits row field and 2-bits column field that represent the row number and column number of the sub-state. The eight least significant bits in the token are the data field.

3.3. Hold-Match-Fetch unit

The HMF unit is the key element in our design. The structure of the HMF unit is shown in Fig. 3(a), which consists of the Holder, the Matcher and the Fetcher. It implements random order dispatching in the data driven way. Tokens coming from the input interface or Round Ring are stored in the Holder. Available tokens in the Holder can be automatically detected by the Matcher. The Matcher will then randomly select one ready token according to a random number generated by the RNG. According to the address of the selected sub-state, the Fetcher reads the data from the Holder and sends it into the Round Ring for round transformation. Thus, the round transformation is performed on sub-states in an out-of-order way.

The Holder stores tokens for round transformation. Different from general register files, the Holder not only registers tokens but also flags whether tokens are available. There is one storage unit and one flag unit corresponding to each sub-state as shown in Fig. 4. The flag unit is a C-element[14] of which the output is the flag signal. Each flag signal becomes ‘1’ after the data is written into the corresponding register, and the flag signal becomes ‘0’ after the data is read out. The token writing process is controlled by the handshake between ‘cin’ unit and flag units. In a writing process, according to the address extracted from the token, the ‘cin’ unit generates a request signal, req$_i$ $i \in [1, n-1]$, for the flag C-element, C_i. The request signal, req$_i$, is also the latch signal for the register, R_i. After the data is stored into R_i, the output of the flag C-element, flag$_i$, goes high and acknowledges the completion of the writing process to the ‘cin’ unit.

The structure of the Matcher is shown in Fig. 3(a). The ‘matching logic’ performs logic-AND on flag signals from the Holder and key flag that indicates the availability of round keys. The outputs of match logic indicate which pairs of sub-state and key are ready for AddRoundKey. According to the results of match logic and the random number, the arbiter randomly selects one ready sub-state and outputs the address of the selected sub-state to the Holder. According to the address, the sub-state is read from the Holder.

In this paper, we use the R-box[16] to build the arbiter, as

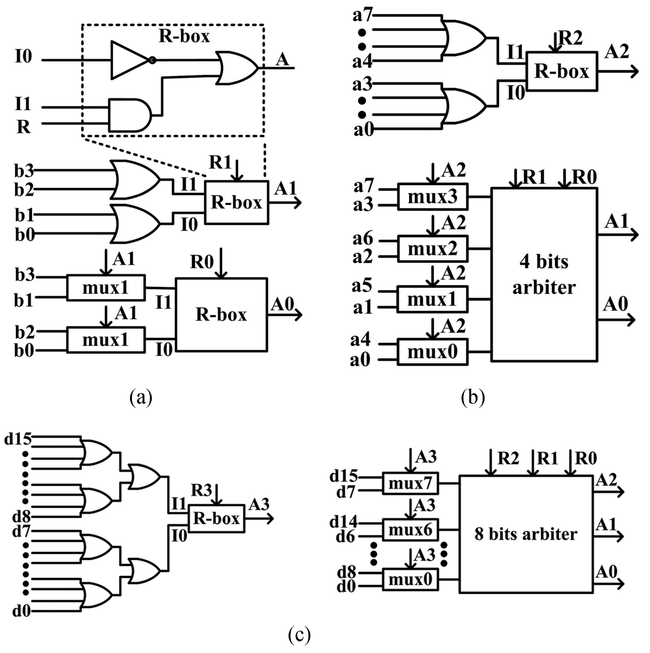


Fig. 5. (a) R-box[16] and 4-bits arbiter, (b) 8-bits arbiter, (c) 16-bits arbiter.

Table 1. Truth table of the R-box.

I0	I1	R	A
0	0	X	1
1	0	X	0
0	1	X	1
1	1	0	0
1	1	1	1

shown in Fig. 5(a). The truth table of the R-box is shown in Table 1. When ‘In0’ and ‘In1’ are both ‘0’, the output of R-box, ‘A’, is ‘1’. If either ‘In0’ or ‘In1’ is ‘1’, ‘A’ is the address of the input with the value of ‘1’. When ‘In0’ and ‘In1’ are both ‘1’, the value of ‘A’ equals the value of ‘R’, which is a random number. In other words, when both inputs are ‘1’, ‘R’ randomly selects the address of one input. Based on the R-box, n-bit arbiters can be built. Fig. 5 show how a 16-bit arbiter is built hierarchically, which is used in our chip. For the 16 bit arbiter, as shown in Fig. 5(c), ‘R0’, ‘R1’, ‘R2’ and ‘R3’ are random signals. The outputs, ‘A0’, ‘A1’, ‘A2’ and ‘A3’, are the address of randomly selected input signals. When all the inputs are ‘0’, the output address is ‘1111’. If more than one input is ‘1’, the output address is determined by the random number. When ‘d15’ is ‘1’ and selected by the arbiter, the output address is ‘1111’ and the fetch signal is high. When all the inputs are ‘zero’, the output address is also ‘1111’, but the fetch signal will not go high, so the two cases with the same output value can be differentiated.

If one or several outputs of match logic become high, the fetch signal goes high and activates the 4-phase bundle data pipeline in the Fetcher unit, as shown in Fig. 3(a). The output of C_1 goes high and sets the lock signal to ‘1’, which locks the latch in the Matcher to make sure the output of the Matcher will not change. After the lock signal becomes one, the output of arbiter will be stable. The data retrieved from the Holder will be stable. The delay unit in the Fetcher is to make sure a stable sub-state can be correctly retrieved from the Holder and stored in the output register. In order to obtain the sub-state correctly, the delay unit should satisfy the following relationship,

$$T_{\text{delay}} + T_{\text{c\_element}} \leqslant T_{\text{or}} + T_{\text{arbiter}} + T_{\text{read}} + T_{\text{setup}}, \quad (1)$$

where T_{delay}, $T_{\text{c\_element}}$ and T_{or} are the propagation delays of the delay unit, C-element and logic-or gate. T_{arbiter} is the delay of the arbiter for generating a stable address signal. T_{read} is the time required to read a sub-state from the Holder. T_{setup} is the setup time of the output register.

The retrieved token is locked in the output register by the output of C_2, which also sets the clear signal to high. The clear signal handshakes with the flag C-elements in the Holder and resets the flag signal to ‘0’. Because the selected flag signal becomes ‘0’, the fetch signal returns to zero. The outputs of C_1 and C_2 return to ‘0’ in sequence due to the handshake protocol. After the lock signal is reset to ‘0’ by C_2, the latch becomes transparent and the arbiter starts to select another available sub-state according to the flag signals. The time sequence diagram of the fetching process is shown in Fig. 3(b).

4. Evaluation and results

4.1. Chip implementation

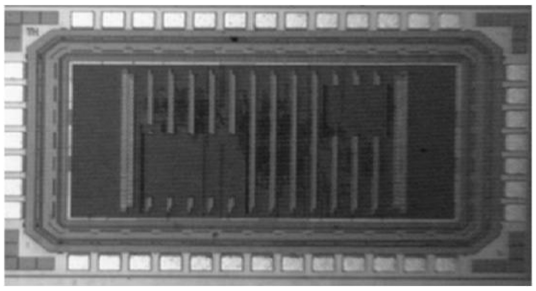
The AES chip was designed using standard library cells and fabricated in SMIC 0.18 μm technology. The photograph of the fabricated chip is shown in Fig. 6(a). Since the standard library does not contain C-elements, we designed the C-element in the full custom way and added it to the standard library. Although the chip is an asynchronous design, synchronous design tools, Synopsys Design Compiler and IC Compiler, were utilized for the logical synthesis and physical synthesis according to the method proposed by us[15].

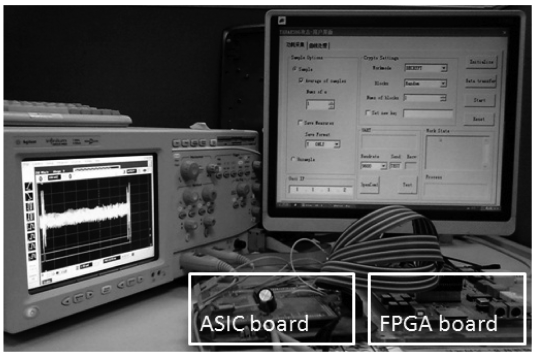
The chip supports standard encryption and decryption defined in AES. In order to evaluate the improvement on DPA resistance by utilizing random order execution, the chip has two working modes, random mode and normal mode. An LFSR (linear feedback shift register) based pseudo-random number generator is employed to support both the mode and generate random number needed by the HMF unit. The length of the LFSR is 64 bits, and the feedback polynomial is $x^{64} + x^{63} + x^{61} + x^{60} + 1$. The initial seed of the LFSR is set through an input interface. In normal mode, the initial seed of the LFSR is set to zero, and the chip executes operations in a fixed order. In random mode, the initial seed is set to a random number. In practical applications, true random number generators should be integrated to guarantee out-of-order execution in the chip.

4.2. DPA resistant evaluation

4.2.1. DPA attack experimental setup

The measurement and DPA attack setup is shown in Fig. 6(b). The FPGA board receives plain texts or cipher texts and secret keys from a PC and sends them to the AES chip. The FPGA also controls the chip to perform encryption or decryption under random or normal mode. A small resistor is placed between the power source and the VDD port of the chip. The


(a)



(b)

Fig. 6. (a) The photograph of the chip. (b) DPA attack experimental setup.

voltage across the resistor is measured by the oscilloscope. The oscilloscope sends recorded traces to a PC for further analysis.

4.2.2. DPA attack method

DPA attacks were carried out on the AES chip under both random mode and normal mode. The target operations of the DPA are the S-boxes of the first round. In our DPA attack for the normal mode (operations are executed in a certain order), the estimation of the power consumption of S-boxes is compared with the measured power consumption of the chip. In line with Ref. [3], we use the Hamming distance of S-boxes to represent the estimation of power consumption. The attack procedure is formulated as follows. Let P_k, K_k and K_k' be the k-th byte of plain texts (inputs of the chip), correct key and guesses of the correct key. Since K_k is 8 bits, there are 256 possibilities for K_k' and 256 possible estimated power signatures. The group of power signatures of various guesses, P_{estimate}, can be formulated as,

$$S_k = \text{SubByte}(\text{AddRoundKey}(P_k, K_k')), \tag{2}$$

$$P_{\text{estimate}} = \text{HamDist}(S_k, S_{k-1}), \quad k \in [1, 16], \tag{3}$$

where S_k is the result of the k-th S-box.

Let P_{measure} be the actual power consumption of the target operation. The correlations between power measurements, P_{measure}, and the 256 variants of P_{estimate} are calculated in the attack. The K_k' with the highest correlation is selected as the correct key. In general, the correlation obtained from a few measurements is inaccurate due to the power noises contributed by other operations. In order to filter the noise, thousands of plain texts should be generated to perform the attack.

Because the output of the SubByte unit can be reset before starting encryption or decryption (S_0 is 0), the Hamming

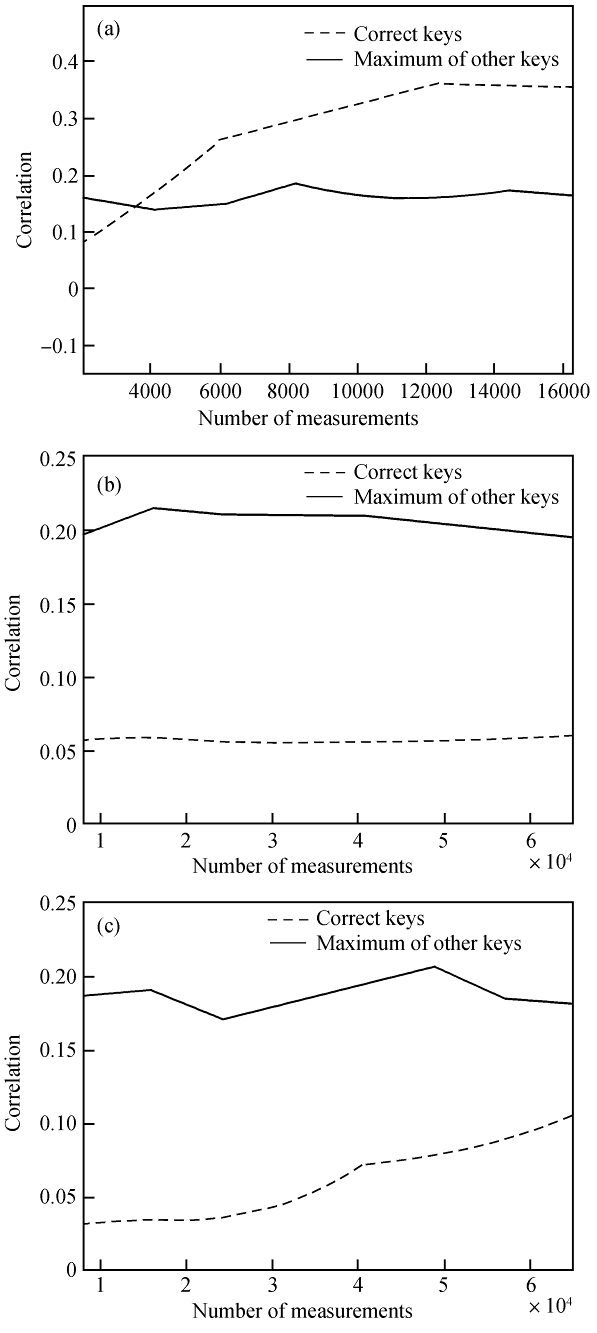


Fig. 7. Attacks on the chip (a) under normal mode and (b) under random mode. (c) Windowing attack under random mode.

distance of the first S-box equals the Hamming weight, which is the result of the first S-box and can be calculated. Consequently, our attack first targets on the first byte of keys. After obtaining the first byte of keys, our attack is performed on the other keys accordingly.

We used the same method to attack the chip under random mode. Due to random order execution, the first SubByte operation may not be performed on the first sub-state. As a result, the Hamming distances of the first SubByte may be incorrectly calculated under random mode. The incorrect Hamming distance makes DPA attacks more difficult. The windowing attack that can reduce attacking efforts[7] was also carried out on the chip under random mode. It integrates the power consumption in the period that covers all possible execution times of the target op-

Table 2. Comparison with previous works.

Reference	Ref. [6]	Ref. [3]	Ref. [9]	This work
Process (μm)	0.18	0.18	0.13	0.18
Function	encryption	encryption	encryption	encryption and decryption
Equivalent gates	2×10^4	5.96×10^5	6.9×10^5	1.5×10^4
Throughput (Mbps)	4	990	1280	20
Throughput/Gates (bps/gate)	200	1661	1855	1333(2000)[2]
Energy (nJ/encryption)	115	129	4.8(15)[1]	18
DPA countermeasure	masking	power constant logic	decoupling	random order execution
Design strategy	standard cell	WDDL logic library, specific routing tool	analogue design	standard cell, asynchronous design

[1]The estimation of energy consumption if the design were fabricated in 0.18 μm technology.
[2]Only consider the gates (2/3 of total number of gates) for encryption.

eration. The integrated results are used to calculate correlation with Hamming distances.

4.2.3. DPA resistant evaluation

In line with Ref. [3], the DPA resistance of the chip is quantified by the number of measurements to disclosure (MTD). In general, compared with wrong key guesses the power estimated by the correct key should have the highest correlation with the measured power signature. However, owing to the power noises contributed by other operations, the highest correlation may not correspond to the correct key when the number of samples (power signatures) is small. In order to reveal the real correlation between estimated power and measured power, the number of samples for DPA attack needs to be large. When the number of samples is large enough, the correlation obtained by the correct key will larger than that obtained by wrong guesses. The MTD is defined as the number of samples when the correlation coefficient of the correct key and the maximum correlation coefficient of other wrong key guesses have a crossover, and the correct key has the largest correlation after the crossover. It means that the keys with the highest correlation can be considered as the correct key when the number of power traces is larger than the MTD.

The MTD curves of the first byte of the secret key under normal mode are shown in Fig. 7(a), which shows that in normal mode the first byte of the key can be disclosed using around 3000 power traces. Figure 7 shows the correlation curves under random mode using non-windowing and windowing attacks respectively. From the figure, we can see that the two curves are closer to each other in windowing attacks, so windowing attacks are more effective than non-windowing attacks. However, although a windowing DPA attack was used, the key cannot be obtained after 64000 power traces, as shown in Fig. 7(c). As a result, the proposed hardware based random order execution improves by least 21 times the DPA resistance when compared with the corresponding fixed order execution.

4.3. Chip performance comparison

Table 2 summarizes the results of the fabricated chip and compares previous work employing other DPA countermeasures. In terms of functionality, our chip implements both standard encryption and decryption while other works only have an encryption function. In order to improve DPA resistance, our chip employs a fine-gained architecture to increase the number of operations for random execution. Owing to the fine-grained structure, our chip has the smallest die area. However, because of serially executing fine-grained operations, the throughput of our design is also compromised. In order to evaluate the tradeoff between throughput and area and make a fair comparison with other designs, we use the metric, Throughput/Gates, for comparison. Since other works only have an encryption function, and the decryption circuits of our design takes up around 1/3 of the area, we use 2/3 total gates for the evaluation. The metric of our design is the highest. It means that our design can achieve the best throughput when using the same amount of hardware resources. This further implies that our DPA counter-measure introduces less hardware overhead than counter-measures used by other designs. Our chip consumes 18 nJ for one encryption, which is higher than the smallest one (around 15 nJ)[9] that uses a decoupling technique. However, our chip implements both encryption and decryption circuits. Decryption circuits also contribute power consumption to the encryption process because they increase the equivalent capacitance. If only considering the encryption circuits, our design may be more energy efficient than that in Ref. [9]. Considering the evaluation results in Table 2, our design suits applications that have astringent area, energy constraints, and high DPA resistance and low throughput requirements.

5. Conclusion

In the paper, a novel AES chip with hardware-based random order execution is presented to enhance DPA resistance. Fine-grained computing is utilized to sufficiently exploit intrinsic operation-level independencies in the algorithm. Dynamic dataflow architecture is proposed to implement random order execution. Owing to the fine-grained structure and data-driven computing, the chip is characterized by small area and low energy consumption. Compared with the fixed order execution, the DPA resistance of our chip is greatly enhanced.

References

[1] Kocher P, Jaffe J, Jun B. Differential power analysis. Advances in Cryptology, Leture Notes in Computer Science, 1999, 1666: 388

[2] Kocher P. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. Advances in Cryptology, Leture Notes in Computer Science, 996, 1109: 104

[3] Hwang D D, Tiri K, Hodjat A, et al. AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis side-channel attacks. IEEE J Solid-State Circuits, 2006, 41(4): 781

[4] Han Jun, Zeng Xiaoyang, Tang Tiangao. VLSI design of anti-attack DES circuits. Chinese Journal of Semiconductors, 2005, 26(8): 1646

[5] Li Xiangyu, Sun Yihe. DPA resistant power-balanced adder for a cryptographic IC. Chinese Journal of Semiconductors, 2005, 26(8): 1629

[6] Pramstaller N, Gurkaynak F K, Haene S N, et al. Towards an AES crypto-chip resistant to differential power analysis. Proceedings of the 30th European Solid-State Circuits Conference, 2004: 307

[7] Rivain M, Prouff E, Doget J. Higher-order masking and shuffling for software implementations of block ciphers. Cryptographic Hardware and Embedded Systems, Leture Notes in Computer Science, 2009, 5747: 171

[8] May D, Muller H L, Smart N P. Non-deterministic processors. Information Security and Privacy, Leture Notes in Computer Science, 2001, 2119: 115

[9] Tokunaga C, Blaauw D. Secure AES engine with a local switched-capacitor current equalizer. IEEE International Solid-State Circuits Conference, 2009: 64

[10] Yu Bo, Li Xiangyu, Zhang Naiwen, et al. A low cost, low power AES ASIC with high DPA resisting ability. IEEE Asian Solid-State Circuits Conference, 2009: 285

[11] Mangard S, Oswald E, Popp T. Power analysis attacks. Springer, 2007

[12] Daemen J, Rijmen V. The design of Rijndael: AES-the advanced encryption standard. Springer, 2002

[13] Veen A H. Dataflow machine architecture. ACM Computing Surveys, 1986, 18(4): 365

[14] Jens S, Steve F. Principles of asynchronous circuit design: a systems perspective. Boston: Kluwer Academic Publishers, 2001

[15] Li Xiangyu, Sun Yihe. Optimize asynchronous design by synchronous synthesis tool (in Chinese). Journal of Computer-Aided Design and Computer Graphics, 2006, 18(8): 1098

[16] May D, Muller H L, Smart N P. Random register renaming to foil DPA. Cryptographic Hardware and Embedded Systems, Leture Notes in Computer Science, 2001, 2162: 28