Helper Data Aware Cloning Method for Physical Unclonable Function

Masaya Yoshikawa and Yusuke Nozaki
Dept. of Information Engineering
Meijo University
Nagoya, Japan
dpa_cpa@yahoo.co.jp

***Abstract*—With the increase in the instances of damage caused by counterfeits of electronic components, a physical unclonable function (PUF) has attracted attention as a technique to prevent counterfeiting. PUF uses arbitrary dispersion as specific identification, which is generated during semiconductor manufacturing. Even if all large-scale integrated (LSI) circuit patterns are copied, PUF can discriminate the LSI circuit from counterfeits. However, the vulnerability of PUF to machine learning attacks has been reported. Although machine learning attacks can predict responses to unknown challenges with high probability, these attacks cannot completely predict responses. For machine learning attacks, the present study proposed a method to clone PUF's responses using error correction techniques, such as Fuzzy Extractor. The present study also demonstrated that the proposed method could clone PUF mathematically. An evaluation experiment revealed that in machine learning attacks, when the number of learning times was 1,024, the ratio of an arbiter PUF with 64 selector steps to predict responses was 95% and that of the proposed method was 100%. Therefore, the proposed method could clone responses completely.**

Keywords—security, physical unclonable function, fuzzy extractor, machine learning attack

I. INTRODUCTION

Recently, with the advancement of reverse engineering technologies, the damage caused by counterfeits of electronic components has become an increasingly serious problem. Counterfeits of electronic components are reported to occupy approximately 5% of the global semiconductor market, and can result in serious financial damage to companies. If counterfeit electronic components are mounted onto electric vehicles or medical appliances, a fatal accident may occur. Therefore, measures against counterfeits must be urgently established. Measures that are necessary to prevent counterfeiting include the management of electronic components by assigning a specific identification to each electronic component and the creation of identification that is difficult to clone.

A physical unclonable function (PUF) [1]–[4] is now attracting attention as a technique to prevent counterfeiting. PUF is a function that returns outputs called responses to inputs called challenges. PUF extracts and uses arbitrary physical dispersion as specific identification, which is generated during semiconductor manufacturing. Since arbitrary physical dispersion during semiconductor manufacturing is difficult to artificially control, PUF is difficult to clone. Therefore, even if large scale integrated (LSI) circuit patterns are dead-copied, PUF can determine whether the LSI circuit is genuine or counterfeit.

By collecting several challenge-response pairs (CRPs) and performing machine learning, PUF can predict responses to unknown challenges [5][6]. Attacks of predicting responses to unknown challenges are called machine learning attacks, and the vulnerability of PUF to machine learning attacks has been reported [5][6]. Although machine learning attacks can predict PUF's responses to unknown challenges with high probability, these attacks cannot predict PUF's responses completely.

For machine learning attacks, the present study proposes a method to clone PUF's responses using error correction techniques, such as Fuzzy Extractor (FE) [7], which is often used for PUF authentication. The present study also demonstrates that the proposed method can clone PUF mathematically.

II. ARBITER PUF

A. Operation Principles of Arbiter PUF

An arbiter PUF [2] extracts the difference in the signal propagation time between two signal paths, the lengths of which were designed to be the same, as specific identification. Since the lengths of two signal paths were designed to be the same, no difference should ideally be observed. However, the signal propagation time slightly differs from each other owing to arbitrary dispersion generated during semiconductor manufacturing. The arbiter PUF extracts and uses this slight difference in the signal propagation time as a specific response. Fig. 1 shows the operation principles of an arbiter PUF with n selector steps.

When two selectors arranged lengthwise are handled as a selector unit, the arbiter PUF consists of n selector units and a delay flip-flop (DFF), as shown in Fig. 1. When challenge Ch [n bit] is input to each selector as a selection signal, two signal paths are determined. When the challenge is 1, the signal paths cross each other. When the challenge is 0, the signal paths go straight. When a pulse signal is input from IN, the input signal is propagated through the two paths determined by challenge

978-1-5386-3684-8/17 $31.00 © 2017 IEEE
DOI 10.1109/SmartCloud.2017.14

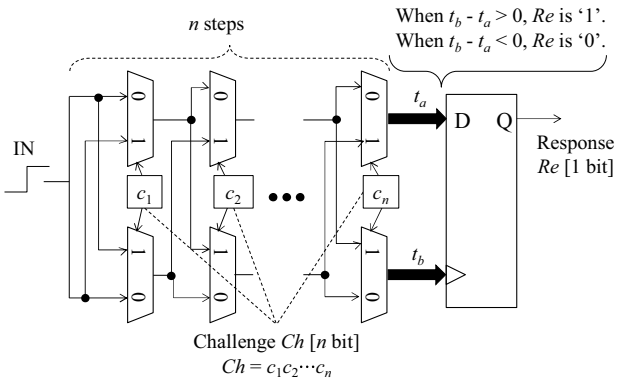

Fig. 1. Operation principles of arbiter PUF

Ch and input to the DFF. At this time, when the signal input to D of the DFF is expressed as t_a and the signal input to CLK is expressed as t_b, response *Re* becomes 1 or 0 when $t_b - t_a > 0$ or $t_b - t_a < 0$, respectively.

B. Modeling of the Arbiter PUF

In modeling of the arbiter PUF, a model for the signal propagation time and a model for the challenge are created. In the present study, the modeling described in [5][6] is used. In the arbiter PUF with n selector steps, the difference between two signal propagation times is expressed as δ_m^0 or δ_m^1 when the challenge c_m of the mth selector is 0 or 1, respectively. At this time, when a model for the signal propagation time is expressed as **u**, the elements of **u** can be expressed using Formula (1). However, $i = 2, \cdots, n$.

$$\begin{aligned} u_1 &= \frac{\delta_1^0 - \delta_1^1}{2} \\ u_i &= \frac{\delta_{i-1}^0 - \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2} \\ u_{n+1} &= \frac{\delta_n^0 + \delta_n^1}{2} \end{aligned} \tag{1}$$

When challenge $\mathbf{C} = c_1, \cdots, c_n$ and a model for the challenge is expressed as **v**, the elements of **v** can be expressed using Formula (2). However, $l = i, \cdots, n$.

$$\begin{aligned} v_i(\mathbf{C}) &= \prod_{l=i}^{n}(1-2c_l) \\ v_{n+1}(\mathbf{C}) &= 1 \end{aligned} \tag{2}$$

When the difference in the signal propagation time between signals that are finally input to the DFF is expressed as Δ, and Formulae (1) and (2) are used, Δ can be expressed as a linear model using Formula (3).

$$\Delta = \mathbf{u}^{\mathrm{T}}\mathbf{v}. \tag{3}$$

When $\Delta > 0$, *Re* becomes 1, and when $\Delta < 0$, *Re* becomes 0. Thus, a model for the arbiter PUF can be created. Therefore, by performing machine learning using several CRPs and creating a model for learning, responses to unknown challenges can be predicted.

III. Fuzzy Extractor

FE [7] is an error correction technique. For data containing certain errors due to environmental variation in biometric authentication, PUF authentication, etc., FE can correct these errors. FE consists of initial generation processing and regeneration processing. In the initial generation processing, helper data are generated. In the regeneration processing, the helper data generated in the initial generation processing are used to correct the responses. Error correction codes used in FE are arbitrarily selected form Reed-Muller codes [8][9], BCH codes [10], etc.

A. Initial Generation Processing

Fig. 2 shows the initial generation processing in FE for PUF. As shown in this figure, random number sequence *Rand* generated using a random number generator is encoded using error correction codes, and code term *C* is obtained. The XOR operation is performed for the obtained code term *C* and PUF's response *Re*, and helper data *He* is obtained.

B. Regeneration Processing

Fig. 3 shows the regeneration processing in FE for PUF. As shown in this figure, the XOR operation is performed for PUF's response *Re** containing errors and helper data *He* generated in the initial generation processing, and code term *C** containing errors is obtained. By performing the correction processing for code term *C** using error correction codes, random number sequence *Rand* is obtained. By encoding the obtained random number sequence *Rand* using error correction codes, code term *C* is obtained. By performing the XOR operation for the obtained code term *C* and helper data *He*, correct response *Re* can be obtained.

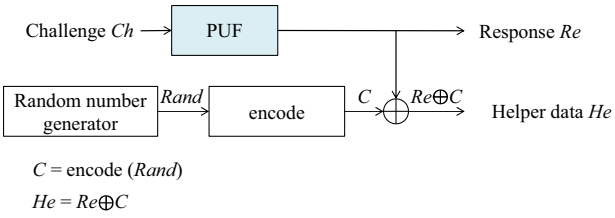

Fig. 2. Initial generation processing

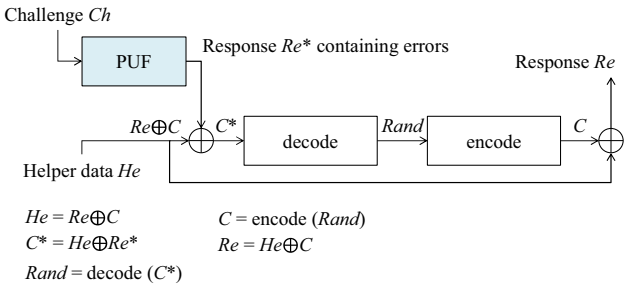

Fig. 3. Regeneration processing

IV. Proposed Method to Clone Responses

In the proposed method, machine learning attacks are performed for PUF. The correction processing is performed for the obtained results using FE. When the prediction of responses by machine learning is incorrect, these responses are corrected. Fig. 4 shows the proposed method. As shown in this figure, the correction processing is performed using FE for response *Re**, which has been predicted using challenge *Ch* of the test data. At this time, data generated in the initial generation processing using response *Re* in test data are used as helper data *He*.

In the proposed method, Reed-Muller (RM) codes are used as error correction codes in FE. Table 1 shows the correction abilities of RM codes used in the proposed method.

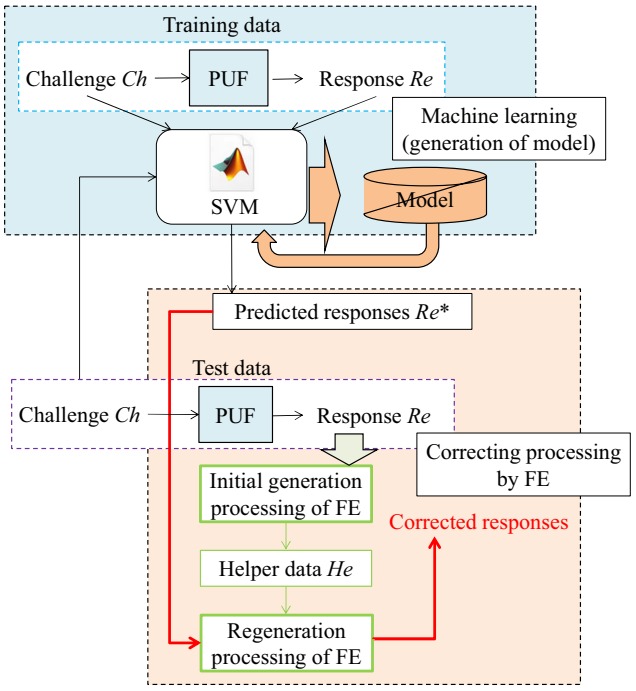

Fig. 4. Proposed method

TABLE I. Correction Abilities of RM Codes

Data size [bit]	Code size [bit]	Correctable error size [bit]
5	16	3
6	32	7
7	64	15
8	128	31

V. Evaluation Experiments

In evaluation experiments, machine learning attacks were first performed for PUF. Subsequently, the proposed method cloned responses.

A. Machine Learning Attacks against PUF

1) Experimental methods: Table 2 shows the experimental environment of machine learning attacks against an arbiter PUF. As shown in this table, a support vector machine (SVM) was used for machine learning, and a library for SVM (LibSVM [11][12]) was used to execute the SVM. Since the kernel function can express the model for the arbiter PUF as a linear model, a linear kernel was used. For the test data, 1,024 CRPs were prepared, and the number of sector steps for the arbiter PUF was 64, 128, or 256. In the experiment, 1,024 sets of test data were handled as one identification, and 100 identifications were generated. A response to each of the 100 identifications was predicted, and the predicted responses were expressed in the form of the prediction ratio.

2) Experimental results: Fig. 5 shows the results of machine learning attacks against the arbiter PUF using SVM. As shown in this figure, above 85% of responses could be predicted at each number of sector steps when the number of learning times was 1,024. Therefore, the model for the arbiter PUF was useful and the arbiter PUF was vulnerable to machine learning attacks. The prediction ratio of responses decreased as the number of selector steps increased. This is probably because the learning model became more complicated as the number of selector steps increased.

TABLE II. Experimental Environment (Machine Learning Attacks)

Machine learning technique	SVM (LibSVM)
Kernel function	Linear kernel
Parameter	Default
Training data for machine learning	CRPs
Number of test data	1,024
Programming language	MATLAB R2013b

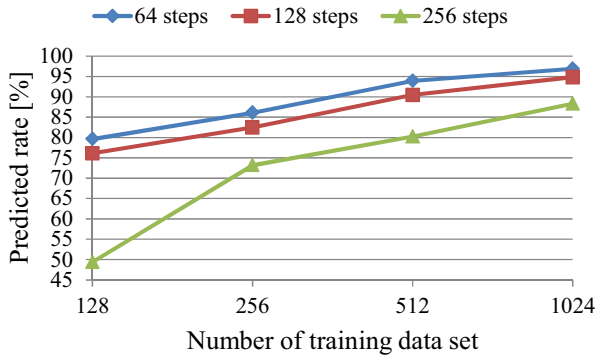

Fig. 5. Experimental results (machine learning attacks)

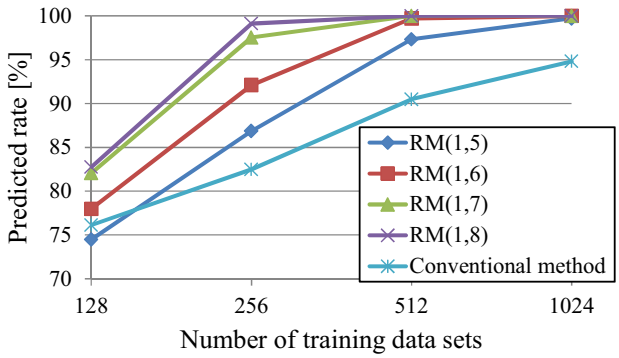

Fig. 7. Experimental results (the number of selector steps: 128)

B. *Evaluation Experiment using the Proposed Method*

1) Experimental methods: In this evaluation experiment, 1,024 sets of test data were handled as one identification, and 100 identifications were generated, as in the case of the experiment on machine learning attacks. Each error in the predicting responses to the 100 identifications by machine learning was corrected using FE.

A one-dimensional RM code, the original data length of which was 5 bits, was expressed as RM (1,5), and errors were corrected using RM (1,5), RM (1,6), RM (1,7), and RM (1,8) for the arbiter PUF with each number of selector steps.

2) Experimental results: Figures 6–8 show the experimental results obtained when the numbers of selector steps were 64, 128, and 256, respectively. As shown in figures 6 and 7, the prediction ratio of responses increased by correcting errors using FE. As shown in Fig. 7, in the case of RM (1,5), the prediction ratio was low when the number of learning times was 128. Also, as shown in Fig.8, when the number of learning times was 256, the prediction ratio was lower than the conventional method. The possible reason for this was that since the number of errors contained in the predicted response value was larger than the allowance, error correction was performed by mistake.

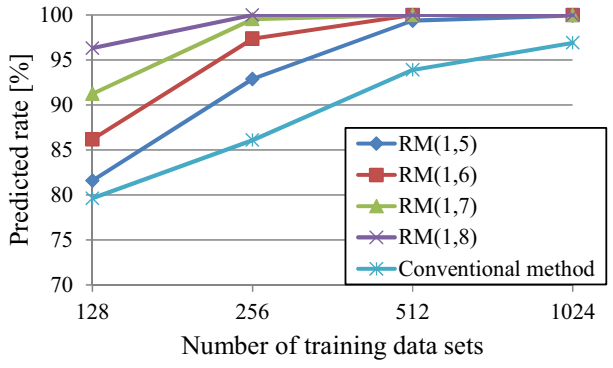

Fig. 6. Experimental results (the number of selector steps: 64)

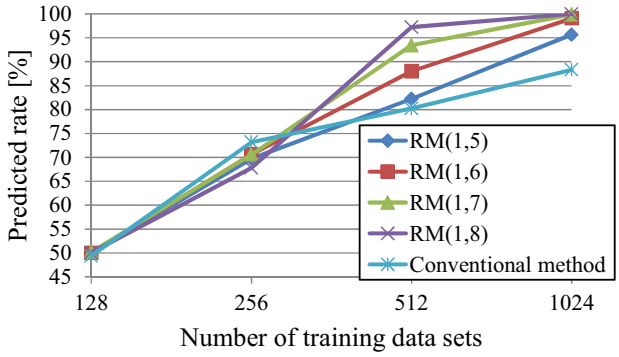

Fig. 8. Experimental results (the number of selector steps: 256)

As shown in figures 6 and 7, when the number of learning times exceeded 512 and a RM code, the correction capability of which is higher than that of RM (1,7), was used, responses could be predicted at 100%. As shown in Fig. 8, when the number of learning times exceeded 1,024 and a RM code, the correction capability of which is higher than that of RM (1,8), was used, responses could be predicted at 100%. Therefore, it was revealed that the proposed method could mathematically clone PUF.

C. *Feasibility Evaluation using FPGAs*

The arbiter PUF (multiplexer (MUX) primitive implementation and lookup table (LUT) implementation) with 64 selector steps was implemented into a field programmable gate array (FPGA) on SASEBO-GII. In these experiments, three SASEBO-GII boards (board A, B, and C) were used. Also, 200 IDs (100 IDs for same challenge and 100 IDs for different challenge) were acquired for 128-bit responses as an ID. Then, same challenge intra-Hamming distance (SC Intra-HD), different challenge intra-HD (DC Intra-HD), and same challenge inter-HD (SC Inter-HD) were used for Steadiness, Diffuseness, and Uniqueness which are typical PUF performance indicators [13].

Fig. 9 shows the SC Intra-HD and DC Intra-HD for each board (A, B, and C). As shown in Fig. 9, the Steadiness and the Diffuseness have almost no changes between several implementations since SC Intra-HD approaches 0 and DC Intra-HD approaches 64 regardless implementation methods. Fig. 10 shows the SC Inter-HD. As shown in Fig. 10, the Uniqueness has almost no changes between several implementations.

VI. Conclusion

For machine learning attacks against PUF, the present study proposed a method for cloning responses using the error correction processing of FE. By performing an evaluation experiment using the proposed method, it was revealed that PUF's responses could be predicted at 100% at any number of selector steps, when the number of learning times exceeded 1,024 and a RM code, the correction capability of which is higher than that of RM (1,8), was used. Therefore, PUF could be mathematically cloned. To prevent PUF from being cloned mathematically, it is important to either make PUF resistant to machine learning attacks to a certain extent or verify the resistance of helper data in FE against machine learning attacks (tamper resistance). In the future, we will develop a PUF that has resistance against machine learning attacks and examine the tamper resistance of helper data.

Acknowledgment

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” Proc. of the 9th ACM Conf. of Computer and Communications Security (CCS’02), pp.148–160, Nov. 2002.

[2] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Debadas, “A Technique to Build a SecretKey in Integrated Circuits for Identification andAuthentication Applications,” Proc. of the IEEE VLSI Circuits Symposium, pp.176–179, June 2004.

[3] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, “FPGA Intrinsic PUFs and Their Use for IP Protection,” Proc. of 9th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007), LNCS 4272, pp.63–80, Speinger-Verlag, Sept. 2007,.

[4] Y. Hori, H. Kang, T. Katashita, and A. Satoh, “Pseudo-LFSR PUF: A Compact, Efficient and Reliable Physical Unclonable Function,” Proc. of IEEE 7th Int. Conf. on Reconfigurable Computing and FPGAs (ReConFig’11), pp.223–228, Nov. 2011.

[5] D. Lim. “Extracting Secret Keys from Integrated Circuits,” M.S. thesis, Dept. Elect. Eng. and Computer Sci., MIT, Cambridge, MA, USA, 2004.

[6] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling attacks on physical unclonable functions,” Proc. of 17th ACM Conf. on Computer and Communications Security (CCS’10), pp.237–249, Oct. 2010.

[7] Y. Dodis, M. Reyzin and A. Smith, “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”, Proc. of EUROCRYPT 2004, LNCS 3027, pp.523–540, Springer-Verlag, May 2004.

[8] D.E. Muller, “Applications of Boolean Algebra to Switching Circuits Design and Error Detection,” IRE Trans. on Electronic Computers, vol.EC-3, issue 3, pp.6–12, Sept. 1954.

[9] I.S. Reed, “A Class of Multiple-Error-Correcting Codes and the Decoding Scheme,” IRE Trans. on Information Theory, vol.IT-4, issue 4, pp.38–49, Sept. 1954.

[10] W. W. Peterson, “Error-correcting codes,” Wiley, N. Y. , 1961.

[11] A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[12] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, “A Practical Guide to Support Vector Classification,” Technical report, Dept. of Computer Science, National Taiwan University, 2003.

[13] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, “Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs,” Proc. of IEEE 6th Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig’10), pp.298–303, Dec. 2010.

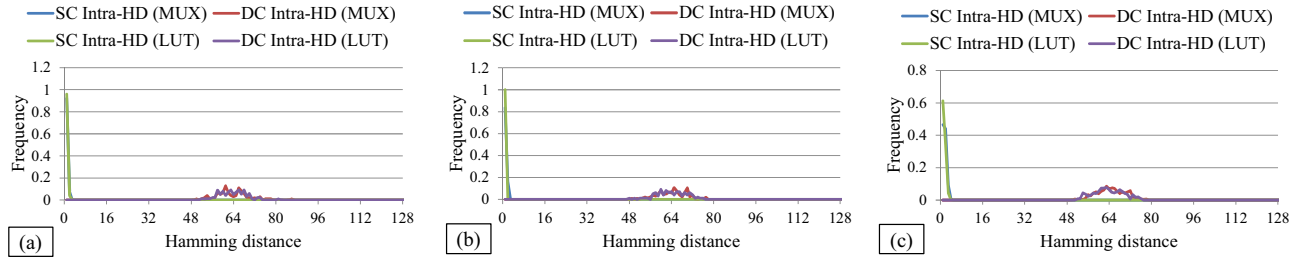

Fig. 9. Distribution of SC Intra-HD and DC Intra-HD with several implementations. (a)Results with board A. (b)Results with board B. (c)Results with board C.

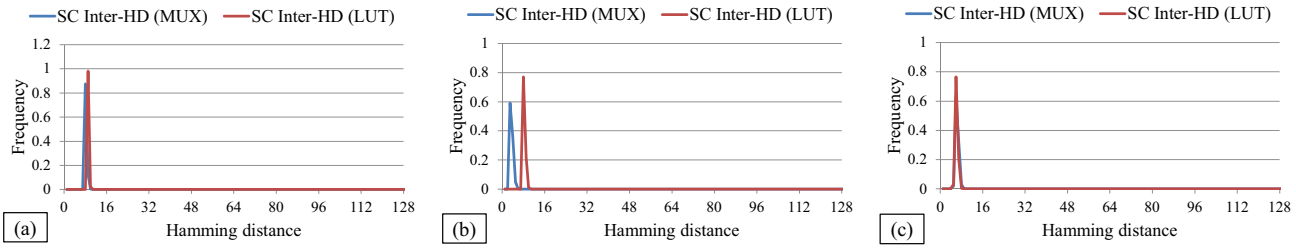

Fig. 10. Distribution of SC Inter-HD with several implementations. (a)Results with board A and B. (b)Results with board B and C. (c)Results with board A and C.