Cryptographic Key Generation from PUF Data Using Efficient Fuzzy Extractors

Hyunho Kang*, Yohei Hori**, Toshihiro Katashita**, Manabu Hagiwara***, and Keiichi Iwamura*

*Dept. of Electrical Engineering, Tokyo University of Science,
6-3-1 Niijuku, Katsushika-ku, Tokyo 125-8585, Japan
{kang, iwamura}@ee.kagu.tus.ac.jp
**Research Institute for Secure Systems (RISEC),
National Institute of Advanced Industrial Science and Technology (AIST)
Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan
{hori, t-katashita}@aist.go.jp
***Dept. of Mathematics and Informatics, Faculty of Science, Chiba University,
1-33 Yayoi-cho, Inage, Chiba 263-0022, Japan
hagiwara@math.s.chiba-u.ac.jp

***Abstract*—Physical unclonable functions (PUFs) and biometrics are inherently noisy. When used in practice as cryptographic key generators, they need to be combined with an extraction technique to derive reliable bit strings (i.e., cryptographic key). An approach based on an error correcting code was proposed by Dodis et al. and is known as a fuzzy extractor. However, this method appears to be difficult for non-specialists to implement. In our recent study, we reported the results of some example implementations using PUF data and presented a detailed implementation diagram. In this paper, we describe a more efficient implementation method by replacing the hash function output with the syndrome from the BCH code. The experimental results show that the Hamming distance between two keys vary according to the key size and information-theoretic security has been achieved.**

***Keywords*—Fuzzy Extractor, Arbiter PUF, Physical Unclonable Functions**

I. Introduction

Physical unclonable functions (PUFs) generate device-unique data streams by using the manufacturing variations of each LSI. If a system requires the best extraction scheme, high security authentication is possible using PUF-based secret key generation. Since PUFs always produce bit errors in their responses, some processing has to be performed to remove the noise for use as a cryptographic key.

In this context, an approach based on an error correcting code was proposed by Dodis et al. [1] and is known as a fuzzy extractor. A fuzzy extractor corrects bit errors in the non-uniform PUF responses and extracts uniform random bits. Experimental studies of fuzzy extractors [2][3][4][5] have received considerable attention since this approach was proposed in 2004.

However, it appears to be a difficult method for non-specialists to implement. In our recent study, we reported on the results of some example implementations using PUF data and presented a detailed implementation diagram [6]. In this paper, we present a more efficient method by replacing the hash function output with the syndrome from the BCH code.

The remainder of this paper is organized as follows. Section II presents the method, detailing our previous work, and proposes a more efficient fuzzy extractor method. The experimental results and conclusions are provided in Sections III and IV, respectively.

II. Proposed Method

A. Summary of our previous work

When implementing a fuzzy extractor scheme, there are two important considerations: information reconciliation and privacy amplification. Information reconciliation guarantees the elimination of noise from the collected noisy data. Privacy amplification guarantees the uniform distribution of the derived key bits. In our recent work, a BCH code and SHA-256 hash function were used to address these two basic requirements.

As shown in Fig. 1, the Generation procedure takes noisy data $\mathbf{w}$ as input and returns a random string $\mathbf{R}$ along with public helper data $\mathbf{P}$. The Reproduction procedure takes noisy data $\mathbf{w}'$ and public helper data $\mathbf{P}$ as input, and outputs $\mathbf{R}$ if $\mathbf{w}$ and $\mathbf{w}'$ are close.

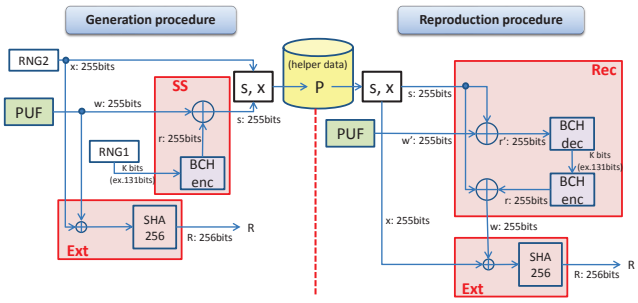

Fig. 1. Implementation diagram for our previous fuzzy extractor based on Dodis et al.'s scheme [6] (N = 255).

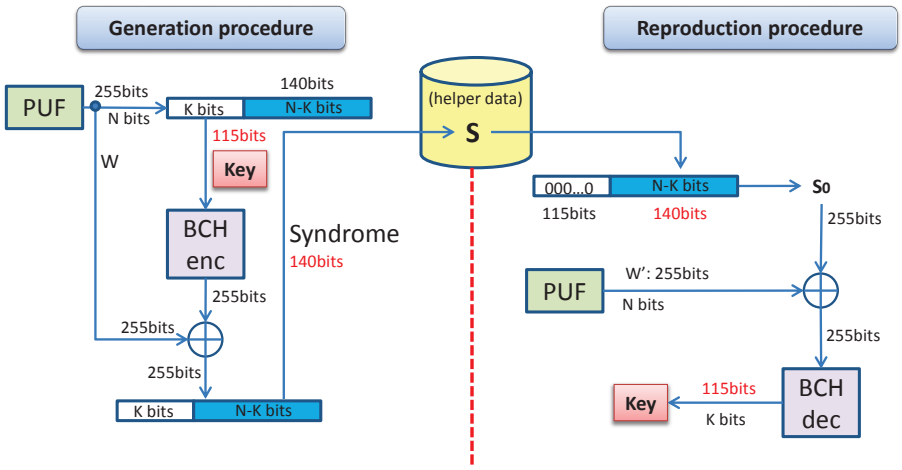

Fig. 2. Implementation diagram for our efficient fuzzy extractor based on the syndrome (N = 255).

B. Proposed syndrome fuzzy extractor

The proposed method is more simply constructed by replacing the hash function output with the syndrome from the BCH code, at the same time enabling an information theoretic security of **K** bits (where K is the key size).

We examine the proposed fuzzy extractor performance by presenting results for all possible combinations of message length K for a BCH code with a fixed codeword length (i.e., 255 and 511) in Section III.

Furthermore, we calculate the entropy of the syndrome data using the formula from [7] to ensure there are uniformly random bits:

$$Entropy = \frac{\mu * (1 - \mu)}{\sigma^2}$$

where μ is mean and σ is standard deviation of the distribution.

Figure 2 shows the implementation diagram for our efficient fuzzy extractor using the BCH code and syndrome concept (N=255). However, the ideas presented in this paper are not limited to BCH codes. In fact, any error correcting code with an efficient syndrome decoding algorithm can be used in this scheme. An approach used for low density parity check (LDPC) codes, for example, was proposed by Hagiwara et al. [8]. In future work, we plan to generalize the extraction scheme (fuzzy extractors) such that other error correcting code methods, including LDPC codes, may be applied.

III. Experimental Results

The field-programmable gate arrays (FPGAs) used in this experiment were two Xilinx's Virtex-5LXs on SASEBO-GII evaluation boards [9].

In this study, we selected challenge response pair data for 100 test iterations using 500 IDs from each Arbiter PUF under test (Fig. 3). We evaluated the performance of the two Arbiter PUFs before evaluating the fuzzy extractor performance. As shown in Figs. 4 and 5, the SC intra-PUF distribution and the DC intra-PUF distribution were computed and plotted to determine how the PUF algorithm separated the two classes. In this experiment, we obtained ideal results with no errors. Therefore, we can conclude that the reliability of each PUF output is high.

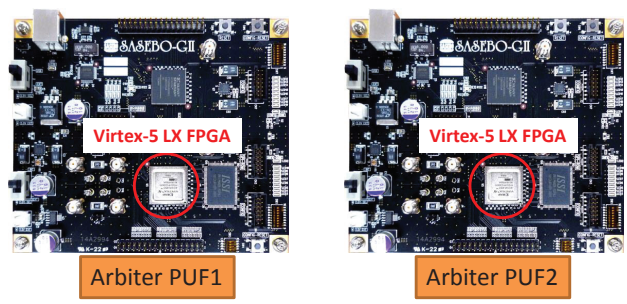

Fig. 3. The two PUFs tested on the SASEBO-GII.

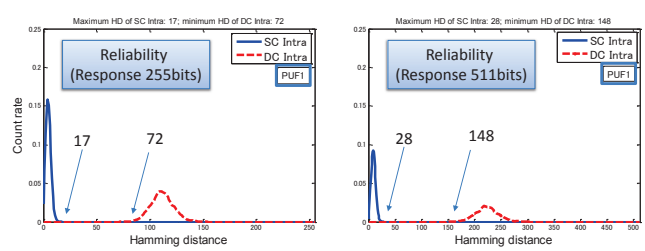

Fig. 4. Reliability (Intra-chip Hamming distance of PUF1).

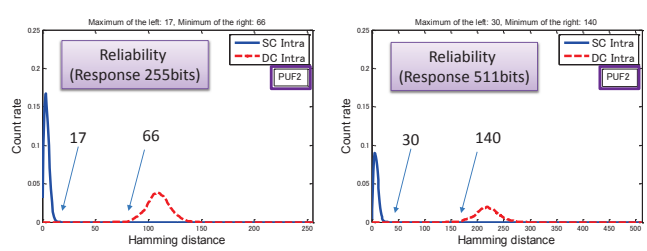

Fig. 5. Reliability (Intra-chip Hamming distance of PUF2).

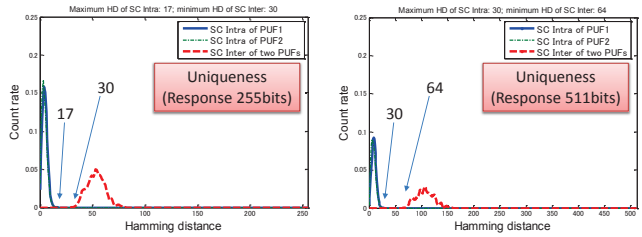

Fig. 6. Uniqueness (Inter-chip Hamming distance of the two test chips).

Figure 6 shows a histogram of the count rate versus the Hamming distance for two PUF outputs combined. The result showed a zero error rate, but not sufficient uniqueness.

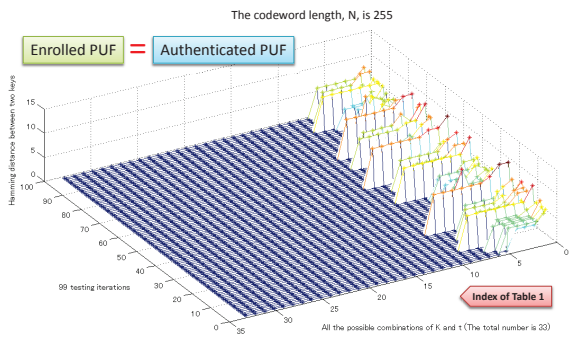

Fig. 7. Hamming distance between two keys extracted from the same PUF (N = 255).

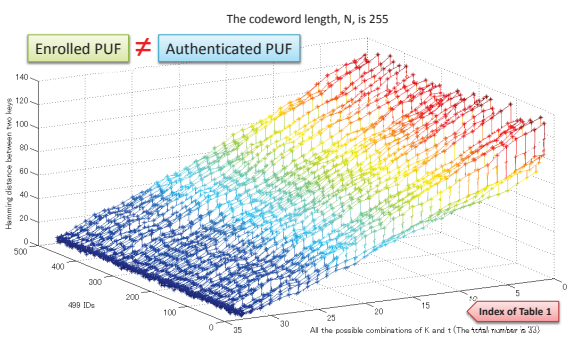

Fig. 8. Hamming distance between two keys extracted from different PUFs (N = 255).

Figures 7 and 9 show the Hamming distance between two extracted keys when the two tested PUFs were the same, demonstrating the dependency of the number of correctable errors, **t**, on the testing index (refer to Table I).

Figures 8 and 10 show the Hamming distance between two extracted keys when two different PUFs were tested. Hence, in the practical implementation of a fuzzy extractor, it is important to consider the size of K (refer to Table II).

Figure 11 shows the Hamming distance distribution of 500 syndromes generated by our fuzzy extractor. As can be seen in this figure, the distribution is perfectly Gaussian with a mean value μ of 0.50011. The standard deviation σ of this distribution is 0.040077. Estimating the entropy of this data gives:

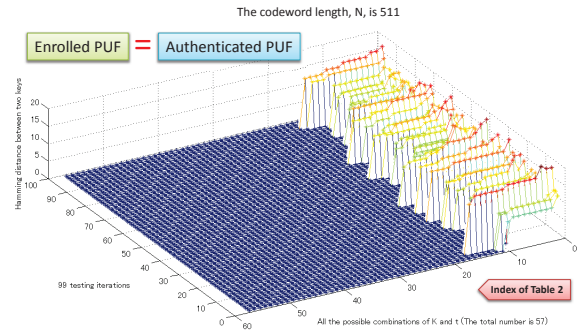

Fig. 9. Hamming distance between two keys extracted from the same PUF (N = 511).

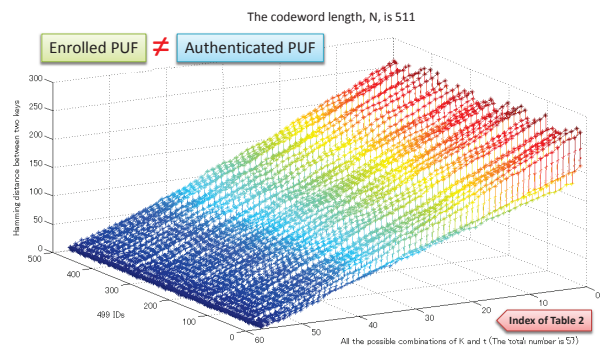

Fig. 10. Hamming distance between two keys extracted from different PUFs (N = 511).

$$Entropy = \frac{0.50011 * (1 - 0.50011)}{0.040077^2} = 155.65\, bits.$$

In this test, the length of the bit sequence extracted from the Arbiter PUF is N (⇒255) and the length of the syndrome produced by the encoder is N-K (⇒255-99). Based on this evaluation, it appears that the output strings of the proposed fuzzy extractor are truly random and contain full entropy, since

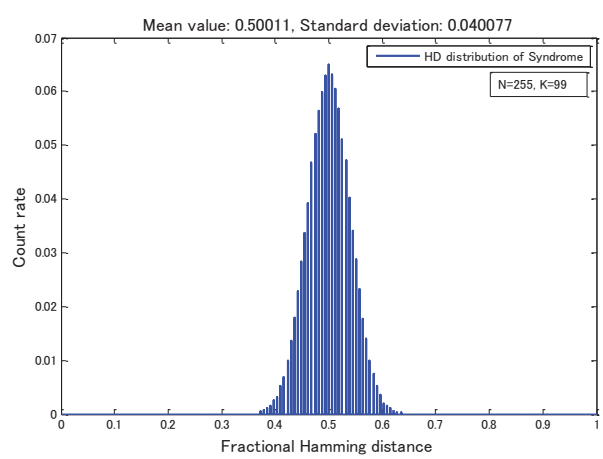

Fig. 11. Hamming distance distribution of 500 generated syndromes (N = 255).

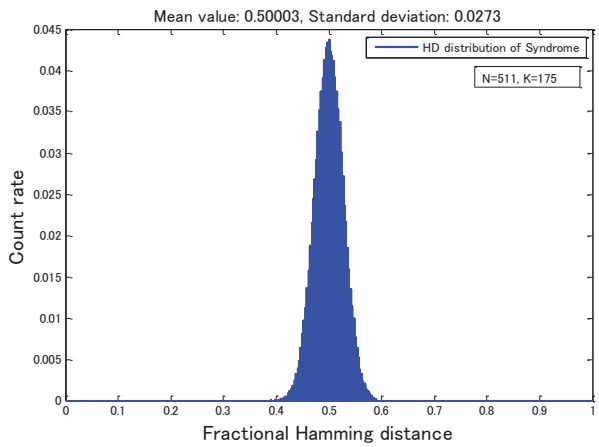

Fig. 12. Hamming distance distribution of 500 generated syndromes (N = 511).

the length of these strings is 156 bits. Then, the probability of guessing the true PUF from the syndrome would be about 2^{-156} (i.e., an information theoretic security of 156 bits).

As can be seen in Fig. 12, the distribution is perfectly Gaussian with a mean value μ of 0.500003. The standard deviation σ of this distribution is 0.0273. Estimating the entropy of this data gives:

$$Entropy = \frac{0.500003 * (1 - 0.500003)}{0.0273^2} = 335.44\, bits.$$

In this test, the length of the bit sequence extracted from the Arbiter PUF is N (⇒511) and the length of the syndrome produced by the encoder is N-K (⇒511-175). Based on this evaluation, it appears that the output strings of the proposed fuzzy extractor are truly random and contain full entropy, as the length of these strings is 336 bits. Then, the probability of guessing the true PUF from the syndrome would be about 2^{-336} (i.e., an information theoretic security of 336 bits).

TABLE I. NUMBER OF CORRECTABLE ERRORS IN THE BCH CODE FOR N = 255

index	N	K	t	index	N	K	t
1	255	247	1	18	255	115	21
2	255	239	2	19	255	107	22
3	255	231	3	20	255	99	23
4	255	223	4	21	255	91	25
5	255	215	5	22	255	87	26
6	255	207	6	23	255	79	27
7	255	199	7	24	255	71	29
8	255	191	8	25	255	63	30
9	255	187	9	26	255	55	31
10	255	179	10	27	255	47	42
11	255	171	11	28	255	45	43
12	255	163	12	29	255	37	45
13	255	155	13	30	255	29	47
14	255	147	14	31	255	21	55
15	255	139	15	32	255	13	59
16	255	131	18	33	255	9	63
17	255	123	19				

IV. CONCLUSION

We showed the results of the proposed fuzzy extractor implementation using Arbiter PUF data and presented a detailed implementation diagram. The proposed method has been very simply constructed by replacing the hash function output with the syndrome from the BCH code, at the same time achieving an information theoretic security of K bits (where K is key size).

TABLE II. NUMBER OF CORRECTABLE ERRORS IN THE BCH CODE FOR N = 511

index	N	K	t	index	N	K	t
1	511	502	1	30	511	241	36
2	511	493	2	31	511	238	37
3	511	484	3	32	511	229	38
4	511	475	4	33	511	220	39
5	511	466	5	34	511	211	41
6	511	457	6	35	511	202	42
7	511	448	7	36	511	193	43
8	511	439	8	37	511	184	45
9	511	430	9	38	511	175	46
10	511	421	10	39	511	166	47
11	511	412	11	40	511	157	51
12	511	403	12	41	511	148	53
13	511	394	13	42	511	139	54
14	511	385	14	43	511	130	55
15	511	376	15	44	511	121	58
16	511	367	17	45	511	112	59
17	511	358	18	46	511	103	61
18	511	349	19	47	511	94	62
19	511	340	20	48	511	85	63
20	511	331	21	49	511	76	85
21	511	322	22	50	511	67	87
22	511	313	23	51	511	58	91
23	511	304	25	52	511	49	93
24	511	295	26	53	511	40	95
25	511	286	27	54	511	31	109
26	511	277	28	55	511	28	111
27	511	268	29	56	511	19	119
28	511	259	30	57	511	10	127
29	511	250	31				

REFERENCES

[1] Y. Dodis, R. Ostrovsky, L. Reyzin and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," (A preliminary version of this paper appeared in Eurocrypt 2004) SIAM J. Comput., 38(1), pp. 97–139, 2008.

[2] P. Bulens, F.-X. Standaert and J.-J. Quisquater, "How to strongly link data and its medium: the paper case," IET Inf. Secur., Vol. 4, Iss. 3, pp. 125–136, 2010.

[3] Ya.N. Imamverdiev and L.V. Sukhostat, "A Method for Cryptographic Key Generation from Fingerprints," Automatic Control and Computer Sciences, Vol. 46, No. 2, pp. 66–75, 2012.

[4] V. van der Leest, E. van der Sluis, G-J. Schrijen, P. Tuyls and H. Handschuh, "Efficient Implementation of True Random Number Generator Based on SRAM PUFs. Cryptography and Security: From Theory to Applications," LNCS6805, Springer-Verlag, pp. 300–318, 2012.

[5] C. Bohm and M. Hofer, Physical Unclonable Functions in Theory and Practice. Springer, 2013.

[6] H. Kang, Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura, "Performance Analysis for PUF Data Using Fuzzy Extractor," Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering, Springer-Verlag, Vol.280, pp.277–284, 2014.

[7] J. Daugman, "The importance of being random: statistical principles of iris recognition," Pattern Recognition, 279–291, 2003.

[8] M. Hagiwara, M.P.C. Fossorier and H. Imai, "Fixed Initialization Decoding of LDPC Codes Over a Binary Symmetric Channel," IEEE Transactions on Information Theory, 58(4), 2321–2329, 2012.

[9] A. Satoh, T. Katashita, H. Sakane, "Secure implementation of cryptographic modules–Development of a standard evaluation environment for side channel attacks–," Synthesiology-English edition, Vol. 3, No. 1, pp. 86–95, 2010.