2015 International Conference on Computational Science and Computational Intelligence

Key Establishment Using Physically Unclonable Functions

SeongHan SHIN and Kazukuni KOBARA
National Institute of Advanced Industrial Science and Technology (AIST)
2-4-7 Aomi, Koto-ku, Tokyo, 135–0064, Japan
Email: seonghan.shin@aist.go.jp

***Abstract*—As an alternative to tamper-resistant modules, PUFs (Physically Unclonable Functions) [1], [6], [7], [8], [9], [10], [11], [14], [15], [18], [25], [26] have received much attention due to their unclonable physical characteristics. In this paper, we propose key establishment (KE1 and KE2) protocols using PUFs which can mitigate side-channel attacks and satisfy two security requirements (i.e., Freshness and Non-Regeneration). Also, we discuss security of the KE1 and KE2 protocols, and compare with the previous PUF-based key exchange protocols [3], [4], [27].**

1. Introduction

In recent years, the needs of cryptographically secure keys have been rapidly increasing for various field applications such as IoT (Internet of Things), M2M (Machine to Machine) communication and CPS (Cyber Physical Systems). Some applications actually require cryptographic keys in order to protect their communication data from eavesdropping, modifications, impersonations, and so on. Ideally, cryptographic keys should be stored in *perfect* tamper-resistant modules, but realizing *perfect* tamper-resistance with low costs is still a challenging open problem. Therefore, better solutions also providing security against side-channel attacks (e.g., differential power analysis [13]) are desirable. As an alternative to tamper-resistant modules, PUFs (Physically Unclonable Functions) [1], [6], [7], [8], [9], [10], [11], [14], [15], [18], [25], [26] have received much attention due to their unclonable physical characteristics (e.g., micro- or nanoscale structural disorder and minuscule manufacturing irregularities).

In this paper, we propose key establishment protocols using PUFs which can mitigate side-channel attacks and satisfy two security requirements (i.e., Freshness and Non-Regeneration). Please, see the following sections for more details.

2. Assumptions and Requirements

In this section, we explain assumptions and security requirements for key establishment protocols using PUFs.

Assumption 1: It is hard for an attacker to let the PUF generate exactly the same response or responses whose Hamming distances are closer than the exhaustively searchable area for the same challenge.

Assumption 2: Modules are equipped with low-level tamper-resistance and an attacker cannot read a memory in the modules directly. But, side-channel information leaks out from an internal value that depends on a part of secrets whose space is exhaustively searchable, and that can be changed with known values by the attacker.

Assumption 1 holds by making length of the response larger (e.g., [11]) and/or combining multiple responses. And Assumption 2 is true at least against differential power analysis [13]. We stress that these assumptions are reasonable because the maximal amount of randomness/entropy in a physical system is polynomially bounded in the size of the system [2], and several attacks on PUFs themselves have been reported recently (e.g., [22], [23]).

Requirement 1 (Freshness): New and independent common keys should be shared between two modules after successful key establishment.

Requirement 2 (Non-Regeneration): It should be hard for an attacker to regenerate the same secrets in the modules where a certain internal value (which is possibly used for side-channel attacks) depends on a part of the regenerated secrets whose space is exhaustively searchable, and that can be changed with known values by the attacker.

Freshness is required to refresh session keys (possibly to be revealed by side-channel attacks), and Non-Regeneration is needed to make side-channel attacks (especially, differential power analysis) much harder.

3. Bad Examples

Before showing our proposals, we give some bad examples so as to get useful lessons in regard to Freshness and Non-Regeneration.

3.1. First Example (Pre-Shared Key in Memory)

In this example, two modules share the same secret beforehand, and then generate fresh session keys depending on exchanged nonce values and the pre-shared secret.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.136

This allows to regenerate the same pre-shared key in the module where an internal value depends on a part of the pre-shared secret, and the internal value is changed with the nonce values. Hence, an attacker can collect a lot of side-channel information on the internal value and the (guessed) pre-shared secret, and then can apply powerful side-channel attacks. The same method holds even if a private (i.e., decryption/signature) key for a public-key cryptosystem is used in order to establish fresh session keys.

3.2. Second Example (Key Update)

In this example, two modules update the key stored in the modules after successful key establishment.

Though this loads different secrets in the memory every time after successful key establishment, an attacker can halt it so that the same secret would be used repeatedly.

3.3. Third Example (Typical Use of PUF for Key Generation [12], [24])

This example proceeds as follows:

1) To let a module hold a cryptographic key, Center sets a helper data $D = R \cdot \mathbf{H}$ to the module where $\mathbf{H}$ is an $n \times (n-k)$ parity check matrix of the underlying error correction code (or a CRC (Cyclic Redundancy Check) generator if the underlying error correction code is cyclic), R is a binary vector of n coordinates (which is a response of the PUF to a challenge C), and '$\cdot$' is an inner product.
2) In order to restore the key, the module accepts the same challenge C and then obtains its response R' from its PUF. Since R' is usually a little bit different from R, it applies a syndrome decoding to $R' \cdot \mathbf{H} + D$ and obtains $E = R' + R$ where '$+$' is an exclusive-or operation. Finally, it calculates a keying material KM with $KM = R' + E = R$.
3) The keying material KM is then used to generate a session key after hashing it or applying a key derivation function [5] to it.

The above satisfies neither Freshness nor Non-Regeneration because the purpose of this example [12], [24] is to restore the same key every time in the memory using PUF. The restored key may be used to establish fresh session keys between modules, but it does not satisfy Non-Regeneration.

3.4. Fourth Example (Naive Key Establishment Using PUF)

This example, which establishes fresh session keys using PUF, proceeds as follows:

1) In order to let Module1 and Module2 establish the same (but changing every time) session key, Center sets a helper data $D_1 = R_2 + N$ to Module1 and $D_2 = R_1 + N$ to Module2, respectively, where R_1 and R_2 are responses of the PUFs in the modules to the respective challenges C_1 and C_2, and N is a random code word of the underlying error correction code (which is a binary vector of the same length as R_1 and R_2).
2) To establish a fresh session key with Module2, Module1 accepts the challenge C_1, obtains its response R'_1 from its PUF, and then sends the syndrome of R'_1 (i.e., $S'_1 = R'_1 \cdot \mathbf{H}$), to Module2. In the same way, Module2 accepts the challenge C_2, obtains its response R'_2 from its PUF, and then sends $S'_2 = R'_2 \cdot \mathbf{H}$ to Module1.
3) Module1 applies the syndrome decoding algorithm to $S'_2 + D_1 \cdot \mathbf{H}$ and then obtains $E_2 = R_2 + R'_2$. It calculates a keying material KM with $KM = D_1 + E_2 + R'_1$, which is equivalent to $N + R'_2 + R'_1$. In the same way, Module2 applies the syndrome decoding algorithm to $S'_1 + D_2 \cdot \mathbf{H}$ and then obtains $E_1 = R_1 + R'_1$. It calculates a keying material KM with $KM = D_2 + E_1 + R'_2$, which is equivalent to $N + R'_2 + R'_1$.
4) The keying material KM is then used to generate a session key after hashing it or applying a key derivation function [5] to it.

Even though this example enables both modules to establish the fresh session key every time (for the same pair of challenges C_1 and C_2), it does not satisfy Non-Regeneration since an attacker can regenerate previously-established session keys in the module. See the next subsection.

3.5. Key Regeneration Attack on Fourth Example

A key regeneration attack on the fourth example is as follows:

1) An attacker eavesdrops communications between Module1 and Module2, and then records a pair of exchanged S'_1 and S'_2.
2) The attacker intrudes in the middle of Module1 and Module2, intercepts new S''_1 and S''_2, and then sends $\widetilde{S''_1} = S'_1 + S''_2 + S'_2$ to Module2 instead of S''_1.
3) After receiving $\widetilde{S''_1}$, Module2 applies the syndrome decoding algorithm to $\widetilde{S''_1} + D_2 \cdot \mathbf{H}$ by following the procedure. Since $\widetilde{S''_1} + D_2 \cdot \mathbf{H} = S'_1 + S''_2 + S'_2 + D_2 \cdot \mathbf{H}$, it obtains $E_1 = R'_1 + R''_2 + R'_2 + R_1$ after the error correction. It calculates a keying material KM with $KM = D_2 + E_1 + R''_2$, which is the same as the previous keying material $KM = N + R'_2 + R'_1$.

4. Our Proposals

In this section, we propose key establishment (KE1 and KE2) protocols using PUFs which satisfy two security requirements (i.e., Freshness and Non-Regeneration). The main idea of the KE1 and KE2 protocols is to make an attacker's modifications on communication data not affect

calculating a keying material in order to prevent the key regeneration attack in Section 3.5.

Let Hash be a cryptographic hash function (e.g., SHA-2/3 [16], [17]) and let KDF be a secure key derivation function (e.g., [5]).

4.1. KE1 (Asymmetric Type)

The KE1 protocol proceeds as follows:

1) In order to let Module1 and Module2 establish the same fresh session key securely, Center sets a helper data $D_1 = R_1$ to Module2 where R_1 is a response of the PUF in the Module1 to a challenge C_1.
2) To establish a fresh session key with Module2, Module1 accepts the challenge C_1, obtains its response R'_1 from its PUF, and then sends the syndrome (e.g., [28]) of R'_1 (i.e., $S'_1 = R'_1 \cdot \mathbf{H}$) to Module2. Also, Module1 calculates a keying material KM with $KM = \mathsf{Hash}(R'_1)$.
3) After receiving S'_1, Module2 applies the syndrome decoding algorithm to $S'_1 + D_1 \cdot \mathbf{H}$ and then obtains $E_1 = R_1 + R'_1$. It calculates a keying material KM with $KM = \mathsf{Hash}(D_1 + E_1)$.
4) Finally, both modules generate a session key $SK = \mathsf{KDF}(\text{Module1}, \text{Module2}, S'_1, KM)$.

4.2. KE2 (Symmetric Type)

The KE2 protocol proceeds as follows:

1) In order to let Module1 and Module2 establish the same fresh session key securely, Center sets a helper data $D_2 = R_2 + N_2$ and N_1 to Module1, and $D_1 = R_1 + N_1$ and N_2 to Module2, respectively. Here, R_1 and R_2 are responses of the PUFs in the modules to the respective challenges C_1 and C_2, and N_1 and N_2 are random code words of the underlying error correction code (which are binary vectors of the same length as R_1 and R_2).
2) To establish a fresh session key with Module2, Module1 accepts the challenge C_1, obtains its response R'_1 from its PUF, and then sends the syndrome of R'_1 (i.e., $S'_1 = (N_1 + R'_1) \cdot \mathbf{H}$), to Module2. In the same way, Module2 accepts the challenge C_2, obtains its response R'_2 from its PUF, and then sends $S'_2 = (N_2 + R'_2) \cdot \mathbf{H}$ to Module1.
3) Module1 applies the syndrome decoding algorithm to $S'_2 + D_2 \cdot \mathbf{H}$ and then obtains $E_2 = R'_2 + R_2$. It calculates a keying material KM with $KM = \mathsf{Hash}(KM_1, KM_2)$ where $KM_1 = R'_1 + N_1$ and $KM_2 = D_2 + E_2 = R'_2 + N_2$. In the same way, Module2 applies the syndrome decoding algorithm to $S'_1 + D_1 \cdot \mathbf{H}$ and then obtains $E_1 = R'_1 + R_1$. It calculates a keying material KM with $KM = \mathsf{Hash}(KM_1, KM_2)$ where $KM_1 = D_1 + E_1 = R'_1 + N_1$ and $KM_2 = R'_2 + N_2$.
4) Finally, both modules generate a session key $SK = \mathsf{KDF}(\text{Module1}, \text{Module2}, S'_1, S'_2, KM)$.

5. Discussions

5.1. Security

In this subsection, we show that the KE1 and KE2 protocols provide two security requirements (i.e., Freshness and Non-Regeneration).

Theorem 5.1. The KE1 and KE2 protocols satisfy Freshness under Assumption 1.

Proof. It is obvious from the descriptions of Section 4.1 and 4.2. □

Theorem 5.2. Under Assumption 1 and 2, the KE1 and KE2 protocols satisfy Non-Regeneration if Hash is a cryptographic hash function.

Proof. Breaking Non-Regeneration of Module1 in the KE1 protocol and Module1 and Module2 in the KE2 protocol is as hard as finding a collision of the hash function Hash. According to Assumption 1 and 2, it is hard for an attacker to let the PUF generate exactly the same response. This guarantees that the inputs (R'_1 and/or R'_2) to the hash function Hash are distinct. If the attacker could regenerate the same KM, it would mean that a collision was found for the hash function Hash since the outputs of Hash are the same while the inputs are distinct. □

5.2. Comparison

Here, we compare the KE1 and KE2 protocols (Section 4) with the previous PUF-based key exchange protocols [3], [4], [27].

In [27], Tuyls and Skoric proposed a PUF-based session key exchange protocol. Later, Busch et al., [4] showed an impersonation attack on Tuyls and Skoric's protocol [27] when an attacker has access to the PUF for a short period of time, and then proposed PUF-based authentication and key establishment protocols using Bloom filters and hash trees. In [21], Rührmair et al., also showed a key regeneration attack on Tuyls and Skoric's protocol [27] under the provision that an attacker gains physical access to the PUF twice. At CRYPTO 2011, Brzuska et al., [3] proposed PUF-based protocols for Oblivious Transfer (OT), Bit Commitment (BC) and Key Exchange (KE). In [19], Rührmair and Dijk gave a quadratic attack on Brzuska et al.,'s OT and BC protocols [3] if optical PUFs or electrical PUFs with challenge length of 64 bits are used. Also, Brzuska et al.,'s KE protocol [3] turned out to be insecure in the PUF re-use model, and in the combined PUF re-use and bad PUF model [20].

The KE1 and KE2 protocols of Section 4 are quite different from [3], [4], [27] in that:

- Freshness and Non-Regeneration of the KE1 and KE2 protocols are guaranteed under Assumption 1 and 2.
- The KE1 and KE2 protocols allow modules to establish a fresh session key *without* using a set (or

an exponential number) of PUF's challenge-response pairs as in [3], [4], [27].
- The helper data D_1 and/or D_2 do not leave Module1 and Module2 all the time.
- The helper data in the KE2 protocol are composed of PUF responses and random code words of the underlying error correction code.

6. Conclusions

In this paper, we have proposed key establishment (KE1 and KE2) protocols using PUFs which can mitigate side-channel attacks and satisfy two security requirements (i.e., Freshness and Non-Regeneration) under Assumption 1 and 2. Also, we have discussed security of the KE1 and KE2 protocols, and compared with the previous PUF-based key exchange protocols [3], [4], [27].

Future works include formal security proofs for the KE1 and KE2 protocols, and their implementations along with experimental evaluations in terms of side-channel attacks.

Acknowledgment

We appreciate the anonymous reviewers' constructive comments on this paper. This research was partly supported by JST, Infrastructure Development for Promoting International S&T Cooperation.

References

[1] Y. Alkabani, F. Koushanfar, N. Kiyavash and M. Potkonjak, "Trusted Integrated Circuits: A Nondestructive Hidden Characteristics Extraction Approach," In *Proc. of Information Hiding*, LNCS 5284, pp. 102-117, Springer, 2008.

[2] J. D. Bekenstein, "How does the Entropy/Information Bound work?," *Foundations of Physics*, Vol. 35, No. 11, pp. 1805-1823, Springerm, 2005.

[3] C. Brzuska, M. Fischlin, H. Schröder and S. Katzenbeisser, "Physical Unclonable Functions in the Universal Composition Framework," In *Proc. of CRYPTO 2011*, LNCS 6841, pp. 51-70, Springer, 2011.

[4] H. Busch, S. Katzenbeisser and P. Baecher, "PUF-Based Authentication Protocols − Revisited," In *Proc. of WISA 2009*, LNCS 5932, pp. 296-308, Springer, 2009.

[5] L. Chen, "Recommendation for Key Derivation Using Pseudorandom Functions (Revised)," NIST Special Publication 800-108, 2009. Available at http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf.

[6] B. Gassend, D. E. Clarke, M. v. Dijk and S. Devadas, "Silicon Physical Random Functions," In *Proc. of ACM Conference on Computer and Communications Security*, pp. 148-160, 2002.

[7] B. Gassend, D. E. Clarke, M. v. Dijk and S. Devadas, "Controlled Physical Random Functions," In *Proc. of 18th Annual Computer Security Applications Conference*, pp. 149-160, IEEE, 2002.

[8] B. Gassend, M. v. Dijk, D. Clarke, E. Torlak, S. Devadas and P. Tuyls, "Controlled Physical Random Functions and Applications," *ACM Transactions on Information and System Security (TISSEC)*, Vol. 10, No. 4, pp. 1-22, 2008.

[9] J. Guajardo, S. S. Kumar, G. J. Schrijen and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protection," In *Proc. of CHES 2007*, LNCS 4727, pp. 63-80, Springer, 2007.

[10] B. Gassend, D. Lim, D. Clarke, M. v. Dijk and S. Devadas, "Identification and Authentication of Integrated Circuits," *Concurrency and Computation: Practice & Experience*, Vol. 16, No. 11, pp. 1077-1098, 2004.

[11] Y. Hori, H. Kang, T. Katashita, A. Satoh, S. Kawamura and K. Kobara, "Evaluation of Physical Unclonable Functions for 28-nm Process Field-Programmable Gate Arrays," *Journal of Information Processing*, Vol. 22, No. 2, pp. 344-356, 2014.

[12] H. Kang, Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura, "Cryptographic Key Generation from PUF Data Using Efficient Fuzzy Extractors," In *Proc. of The 16th International Conference on Advanced Communications Technology (ICACT 2014)*, pp. 23-26, IEEE, 2014.

[13] P. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," In *Proc. of CRYPTO'99*, LNCS 1666, pp. 388-397, Springer, 1999.

[14] S. Katzenbeisser, Ü. Koçabas, V. v. d. Leest, A.-R. Sadeghi, G. J. Schrijen and C. Wachsmann, "Recyclable PUFs: Logically Reconfigurable PUFs," *Journal of Cryptographic Engineering*, Vol. 1, No. 3, pp. 177-186, 2011.

[15] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. v. Dijk and S. Devadas, "A Tchnique to Build a Secret Key in Integrated Circuits with Identification and Authentication Applications," In *Proc. of 2004 Symposium on VLSI Circuits*, pp. 176-179, IEEE, 2004.

[16] NIST FIPS PUB 180-4, "Secure Hash Standard (SHS)," March 2012. Available at http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

[17] NIST FIPS PUB 202 (Draft), "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," May 2014. Available at http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf.

[18] R. Pappu, B. Recht, J. Taylor and N. Gershenfeld, "Physical One-Way Functions," *Science*, Vol. 297, pp. 2026-2030, 2002.

[19] U. Rührmair and M. v. Dijk, "Practical Security Analysis of PUF-based Two-Player Protocols," In *Proc. of CHES 2012*, LNCS 7428, pp. 251-267, Springer, 2012.

[20] U. Rührmair and M. v. Dijk, "PUFs in Security Protocols: Attack Models and Security Evaluations," In *Proc. of 2013 IEEE Symposium on Security and Privacy*, pp. 286-300, IEEE, 2013.

[21] U. Rührmair, C. Jaeger and M. Algasinger, "An Attack on PUF-based Session Key Exchange and a Hardware-based Countermeasure: Erasable PUFs," In *Proc. of Financial Cryptography 2011*, 2011.

[22] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas and J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," In *Proc. of ACM Conference on Computer and Communications Security*, pp. 237-249, 2010.

[23] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson and S. Devadas, "PUF Modeling Attacks on Simulated and Silicon Data," *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 11, pp. 1876-1891, 2013.

[24] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," In *Proc. of The 44th Design Automation Conference*, pp. 9-14, IEEE, 2007.

[25] A.-R. Sadeghi and D. Naccache, *Towards Hardware-Intrinsic Security*, Springer, 2010.

[26] D. Suzuki and K. Shimizu, "The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes," In *Proc. of CHES 2010*, LNCS 6225, pp. 366-382, Springer, 2010.

[27] P. Tuyls and B. Skoric, "Strong Authentication with Physical Unclonable Functions," In *Proc. of Security, Privacy and Trust in Modern Data Management*, 2007.

[28] M.-D. Yu and S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Function," *IEEE Design & Test of Computers*, Vol. 27, pp. 48-64, IEEE, 2010.