J Inf Process Syst, Vol.20, No.1, pp.116~130, February 2024
https://doi.org/10.3745/JIPS.01.0099

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

SoC Virtual Platform with Secure Key Generation Module for Embedded Secure Devices

Seung-Ho Lim[1,*], Hyeok-Jin Lim[2], and Seong-Cheon Park[2]

Abstract

In the Internet-of-Things (IoT) or blockchain-based network systems, secure keys may be stored in individual devices; thus, individual devices should protect data by performing secure operations on the data transmitted and received over networks. Typically, secure functions, such as a physical unclonable function (PUF) and fully homomorphic encryption (FHE), are useful for generating safe keys and distributing data in a network. However, to provide these functions in embedded devices for IoT or blockchain systems, proper inspection is required for designing and implementing embedded system-on-chip (SoC) modules through overhead and performance analysis. In this paper, a virtual platform (SoC VP) was developed that includes a secure key generation module with a PUF and FHE. The SoC VP platform was implemented using SystemC, which enables the execution and verification of various aspects of the secure key generation module at the electronic system level and analyzes the system-level execution time, memory footprint, and performance, such as randomness and uniqueness. We experimentally verified the secure key generation module, and estimated the execution of the PUF key and FHE encryption based on the unit time of each module.

Keywords

FHE, PUF, Secure Key Generation, SoC Virtual Platform

1. Introduction

The authentication and authority for data exchange have become important when transmitting data in networks that general users use in daily life, such as Internet-of-Things (IoT) devices or blockchain networks [1-3]. In the field of digital asset protection, such as non-fungible tokens (NFTs), as digital asset transactions linked to the real world become more active, it is important to ensure the privacy and safe use of sensitive information. As with the secure key of a general secure network system, a critical issue in generating a secure key for digital asset protection is the randomness and uniqueness of the secure key [4-6]. To guarantee this randomness and uniqueness, various research and development studies on secure key generation methods have been conducted by applying encryption and decryption with various features, such as a physical unclonable function (PUF) [7-9], fully homomorphic encryption (FHE) [10,11], and error correcting codes (ECCs), on various levels of systems ranging from networks to dedicated devices [12-14].

Manuscript received April 19, 2023; first revision September 22, 2023; second revision November 30, 2023; accepted November 30, 2023.

* **Corresponding Author:** Seung-Ho Lim (slim@hufs.ac.kr)
[1] Division of Computer Engineering, Hankuk University of Foreign Studies, Yongin, Korea (slim@hufs.ac.kr)
[2] Sudogwon Research Center, ETRI, Seongnam, Korea (hyeokjin.lim@etri.re.kr, scpark@etri.re.kr)

In IoT- or blockchain-based network systems, secure keys are often stored on individual devices. Individual devices should protect data by performing secure key generation operations on the data transmitted and received over networks. In other words, the device should have a secure key generation module. Hence, the development and verification of a secure key generation module at the chip level within individual embedded devices has become important to ensure timely and proper secure key generation. Specifically, in NFT systems, along with virtual digital assets, it is necessary to develop a system-on-chip (SoC) that provides physical duplicability and guarantees uniqueness for real-world assets using appropriate methods and performance guarantees. However, it is difficult to verify the security and performance of the system by applying various secure key generation modules to hardware-level SoCs and chips in a short period.

In this paper, a SoC virtual platform (SoC VP), which includes a secure key generation module, was designed and implemented. The SoC VP flexibly applies various techniques of secure key generation and analyzes performance, such as randomness and uniqueness, as well as system-level performance, such as execution time and memory footprint. Our SoC virtual platform was implemented using SystemC [15], which enables the execution and verification of various aspects of the secure key generation module at the electronic system level (ESL) [16]. Specifically, the SoC virtual platform is configured based on a RISC-V-embedded processor [17,18] and main memory, has a secure key generation module as a controller, and is connected to the processor through a system bus. The secure key generation module consists of internal modules, such as PUF, FHE, number theoretic transform (NTT), Bose–Chaudhuri–Hocquenghem (BCH), pseudo-random number generator (PRNG), and secure hash algorithms (SHA) to generate and encrypt/decrypt secure keys. The internal modules are connected through SystemC-based interfaces and channels so that system-level modeling is performed through the interfaces between the modules. This secure key generation module-based SoC virtual platform can be used as a preceding study for SoC platforms that provide secure keys in embedded devices.

2. Background and Related Work

This section describes related work on security modules and algorithms related to PUF and FHE and the background on the SoC VP with the secure key generation module for embedded devices.

2.1 PUF

Recently, in IoT or blockchain devices that perform network transmission of secure information such as personal or financial information, PUF has been widely selected as a generation method for unique identifiers in devices because it is regarded as a fingerprint for the device and the unique key cannot be exposed to the outside [19-21]. In principle, PUF is a technology for generating secure keys that cannot be physically duplicated owing to differences in the microstructures of semiconductor devices, even if they are produced in the same manufacturing process. Various studies and developments have been conducted to create PUFs using semiconductor devices as a medium, of which SRAM is a representative device [19]. If the best extraction method is required for an embedded device, highly secure authentication can be achieved when a PUF is used. Because there is a probability that errors will occur whenever data are read from the original medium source, the PUF may consist of enrollment to generate a secure key based on the original source and reproduction of the generated key. In [20], the authors developed an encryption key generation method using fingerprints, and the authors [21] proposed a fuzzy extractor for a PUF with a hashing function. Research on embedded devices or systems to which PUF is applied is as follows. Kang et al. [22] used a PUF IC for RFID tags for frequency identification, and Ismari and Plusquellic [23] developed PUF IP using resistance variances. Akhundov et al. [24] used an SRAM PUF for public-key-based authentication for IoT systems and devices, and in [25], they developed an authentication mechanism for embedded systems using PUF modeling. In [26], the authors reviewed the industrial concerns regarding PUF operation.

There are several issues associated with the implementation of a PUF on a virtual platform. While the original PUF obtains unique random values from real devices, we implemented PRNG to generate PUF at the enrollment stage. In addition, an ECC such as BCH is required to model error generation during the reading of PUF values and their corrections. We implemented these modules on a virtual platform for PUF to generate a secure unique key.

2.2 Homomorphic Encryption and NTT

Important personal data, such as medical or financial information, should not only be transmitted as encrypted from the device to the network, but should also be processed in an encrypted form that is not decrypted to ensure data security. FHE can process data without decryption; therefore, it is useful as a secure means of treating personal privacy data [27-29]. Homomorphic encryption enables personal information protection because there is no decryption of personal information in IoT or blockchain networks. However, homomorphic encryption requires high computational complexity, which limits its use in embedded devices.

Homomorphic encryption of data, such as images or videos is time consuming; therefore, it is not executable in embedded devices. However, encryption can be simple for text data such as financial information or text messages. Additionally, many homomorphic encryption operations are based on polynomial arithmetic methods. If we use fast polynomial arithmetic operations, such as polynomial multiplication using the NTT [30], homomorphic encryption can be realized in embedded devices. Therefore, we implemented NTT- and inverse NTT (INTT)-based homomorphic encryption functions in the secure key generation module of the virtual platform.

2.3 SoC Virtual Platform and Module

The secure key generation module is designed for embedded devices for secure networks that require personal information protection. The module is composed of PUF and homomorphic encryption functions, which have recently attracted attention for generating secure keys and secure management of encrypted data. In summary, the SoC VP presented in this study is based on an embedded processor, and the secure key generation module is connected to the embedded processor as a controller through a system bus. The secure key generation module consists of a PUF part for generating a secure key that guarantees randomness and uniqueness and an FHE part that provides homomorphic encryption for the secure key and data. We analyzed the performance and resource usage for secure operations of secure key generation and encryption schemes on the implemented virtual platform.

3. SoC Virtual Platform for Secure Key Generation

3.1 Overall Architecture

The SoC VP was implemented using SystemC, which is capable of ESL modeling of embedded devices. SystemC enables interoperation between modules through channel connections, thread operations, and event processing. We designed and implemented a secure key generation module that enables secure operation in embedded devices through SystemC-based modeling. For fast prototyping of the secure key generation module of the SoC VP, we used the recently released RISC-V virtual platform as an open-source [31] that has already been used as a reference virtual platform prototype for other verifications [32,33]. Fig. 1 shows the overall architecture of a SoC VP. The SoC VP is based on an RISC-V processor. The major modules in the overall structural diagram include the RISC-V processor core, Main Memory, Flash Memory Controller, Interrupt Controller, and DMA Controller. Each module was connected to the RISC-V processor core through the SystemC TLM (transaction-level modeling) 2.0 bus, and the processor core served as the master of each module.

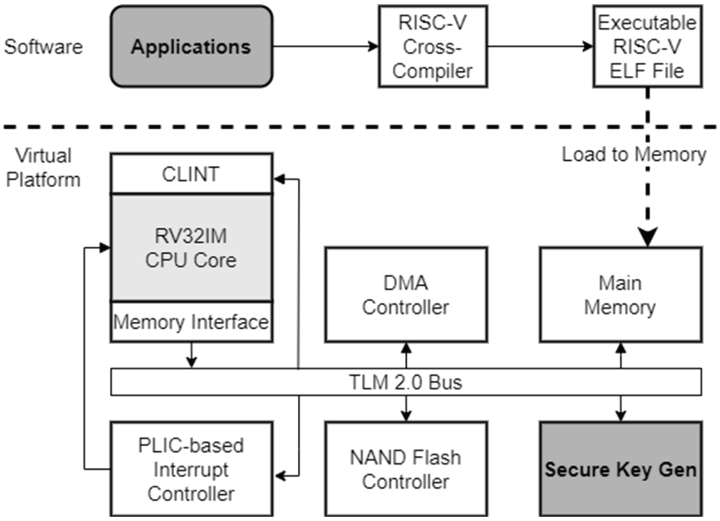
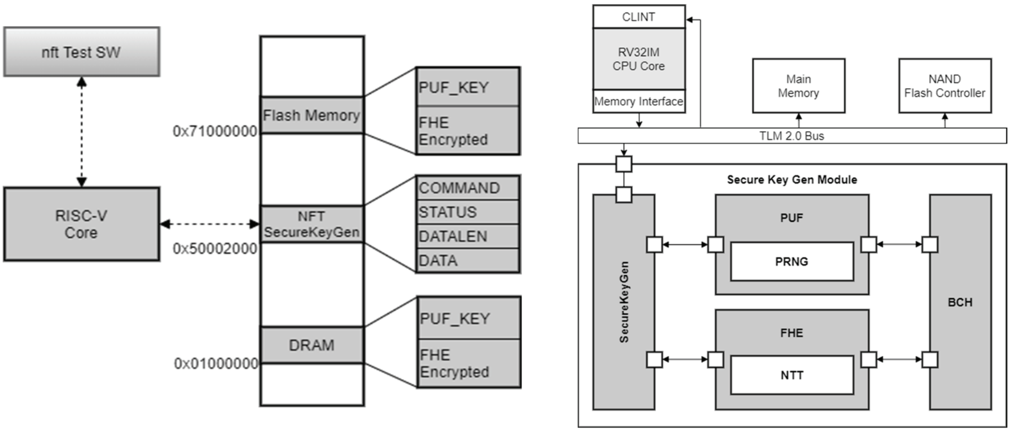


Fig. 1. Overall architecture of SoC virtual platform.

The additional module designed and implemented in this study is the secure key generation module, or SecureKeyGen module. The SecureKeyGen module is also connected to the system bus, similar to the other modules, and the processor core controls the module by accessing the memory-mapped register allocated to the SecureKeyGen module through the TLM 2.0 bus.

The modules used for secure key and homomorphic encryption operations in the virtual platform are the processor, main memory, flash memory, and SecureKeyGen modules. The application software for verifying the SoC VP can be executed as an RISC-V-based user-level program created through RISC-V cross-compilation. The application software performs two tasks: generating a secure PUF key and homomorphic encrypted data. To achieve this, the application software transmits specific commands to the virtual platform, waits until the task for the command is completed on the virtual platform, and receives a completion signal via an interrupt signal. When an interrupt is received, the status register of the virtual platform is read to determine whether the instructions have been completed.

In general, an embedded processor accesses the peripheral modules through the registers defined in the memory map. For instance, the memory map addresses of the SecureKeyGen, main memory, and flash memory modules, which are the main modules used in our virtual platform, are shown on the left side of Fig. 2. Application software running on the virtual platform can access each module by directly accessing each address area allocated to the memory map. The memory address sizes and the main contents of the three modules are as follows: the start address and size of the flash memory were 0 × 71000000 and 4 kB, and the start and end addresses of the DRAM were 0 × 01000000 and 0 × 02000000, respectively. The start and end addresses of SecureKeyGen are 0 × 50002000 and 0 × 50004000, respectively, and SecureKeyGen has the following four registers within the address map—COMMAND (register for user command), STATUS (status register of SecureKeyGen module), DATALEN (register that records the size of data transmitted from the application), and DATA (register for exchanging data with the application).



(a) (b)

Fig. 2. (a) Memory map for SecureKeyGen, main memory, and flash memory modules, which are the main modules used in the virtual platform and (b) structure of the secure key generation module.

3.2 Secure Key Generation Module

The detailed structure of SecureKeyGen module is shown on the right of Fig. 2. This module is comprises SecureKeyGen, PUF, PRNG, FHE, NTT, and BCH modules. The SecureKeyGen module is connected to the TLM system bus in the form of a socket and operates as an initiator-target type connection with the RISC-V core. The internal modules are connected to each other by a SystemC in-out channel, and each module is synchronized by event delivery, which is a SystemC function.

The SecureKeyGen module internally contains SecureKeyGen, PUF, FHE, and BCH modules as submodules. The SecureKeyGen module in SecureKeyGen has a port directly connected to TLM 2.0 system bus plays a role in linking the RISC-V processor core and main memory through this port, and it is directly connected to the PUF and FHE modules through channels. The internal SecureKeyGen module has two SystemC threads: SKGThreadForPUF() and SKGThreadForFHE(). The first thread processes events delivered from the PUF module and the second thread processes events delivered from the FHE module.

The PUF module generates a secure PUF key by receiving commands from the SecureKeyGen module. We modeled and implemented a fuzzy extractor-based PUF enrollment and reproduction method [8,19-21]. For data generation, random number generation was applied based on a pseudo-random number corresponding to the seed through the PRNG module. The PRNG module generates a pseudo-random number by receiving the seed value from the PUF, in which the Mersenne twister random number generation [34] algorithm was applied as the random number generation algorithm.

The FHE module receives commands from SecureKeyGen module and performs homomorphic encryption. The NTT operation was applied to polynomial multiplication for homomorphic encryption. The FHE module performs data encryption through BCH encoding using NTT. The NTT module inside the FHE module performs NTT and convolution operations on the input data, and the results are sent back to the FHE module, where the NTT(conv)NTT and INTT operations are performed. The BCH module was implemented by referring to a binary BCH open-source based on the BCH error correction code [35]. Currently supported BCH code lengths are as follows; (n, k, t) = (255,63,30), (127,64,10), (255,115,21), (511,112,59), and so on, where n is 2^m–1, that is, size of the multiplicative group of GF(2^m), k is (n – deg(g(x))) which means dimension of the code (number of information bits/codeword), and t is error correcting capability (maximum number of errors the code corrects).

3.3 PUF Enrollment and Reproduction

The enrollment and reproduction sequences of the application software for verifying the secure PUF key using the virtual platform are shown in Fig. 3. The operational process is as follows: when the application software transmits the PUF_KEYGEN command to the virtual platform, the processor of the

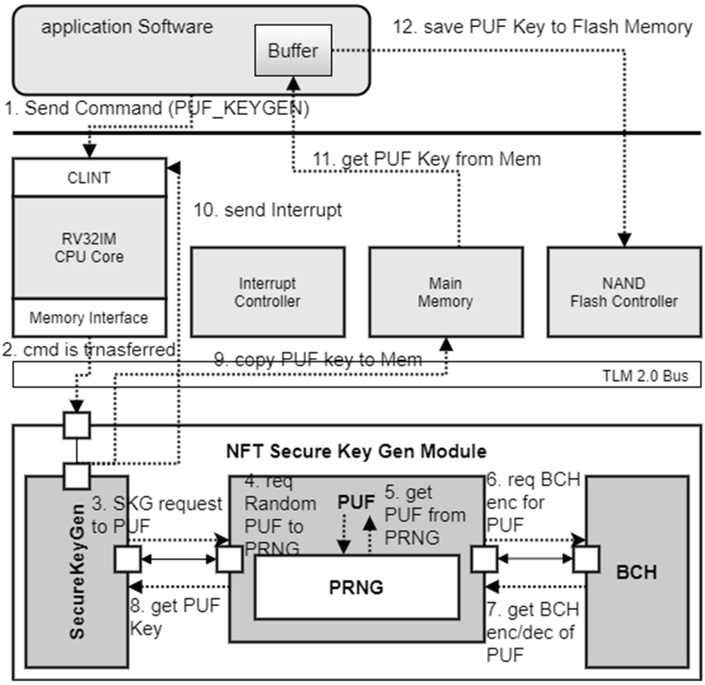


Fig. 3. Enrollment and reproduction sequence of the application software with the SecureKeyGen module on the virtual platform.

virtual platform transmits the command to the SecureKeyGen module. The module inside SecureKeyGen sends a PUF creation request with an identification seed to the PUF module and notifies the user of the event. The PUF module generates the PUF key. It requires the PRNG module to generate a PUF value based on a pseudo-random number. The PRNG module returns seed-based PUF values.

In the enrollment phase, the PUF module requests BCH encoding from the BCH module and returns the encoded data. These are the PUF enrollment data, called helper data, and are stored in flash memory for future PUF reproduction. In the reproduction step, the PUF module generates a PUF key by performing BCH decoding using the helper and PRNG PUF data. After the reproduction step, a secure PUF key is generated. The SecureKeyGen module copies the generated PUF secure key to the main memory. The SecureKeyGen module generates an interrupt and informs the application software of PUF secure key generation.

3.4 Homomorphic Encryption with NTT

The homomorphic encryption sequence using the FHE and NTT modules is shown in Fig. 4. The operation of the application software through the virtual platform is described as follows: first, the application software copies the data for homomorphic encryption from the internal buffer to the main

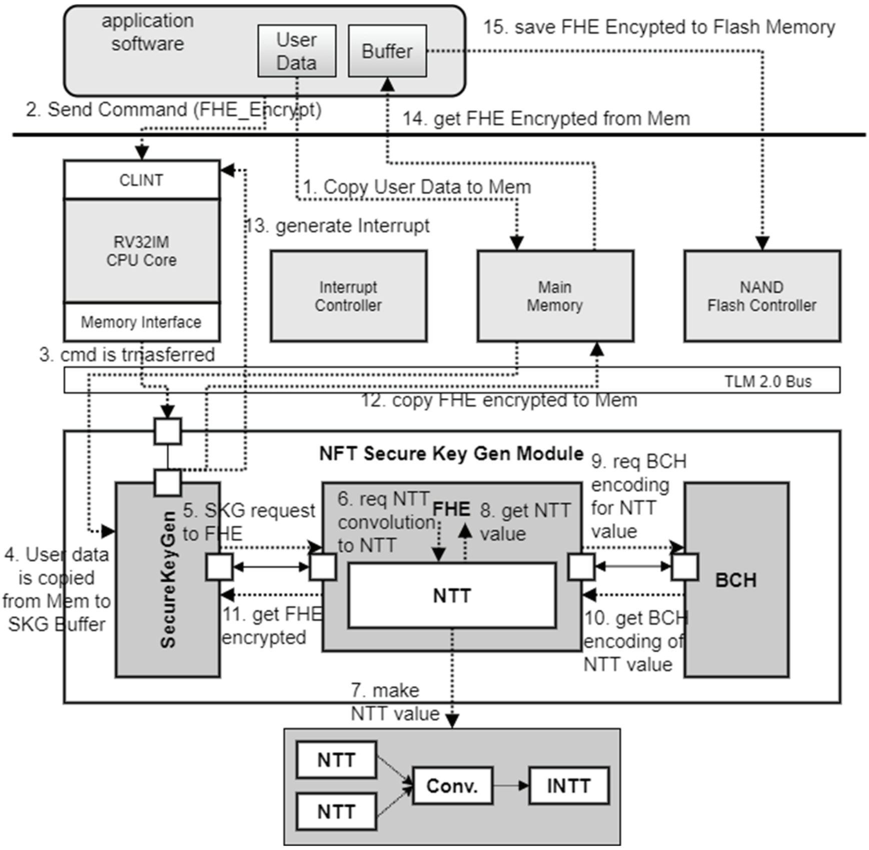


Fig. 4. Sequence of homomorphic encryption of data with the SecureKeyGen module on the virtual platform.

memory and then sends the FHE_Encrypt command to the virtual platform. The virtual platform processor sends the corresponding command to the SecureKeyGen module. The SecureKeyGen module copies the data in the main memory to the internal buffer of the module, sends a request for homomorphic encryption to the FHE module, and notifies the event. The FHE module requests the NTT module to perform homomorphic encryption of the received data, which is a polynomial multiplication using NTT consisting of NTT(conv)NTT → INTT operations. The NTT module performs the corresponding convolution and returns the results to the FHE module.

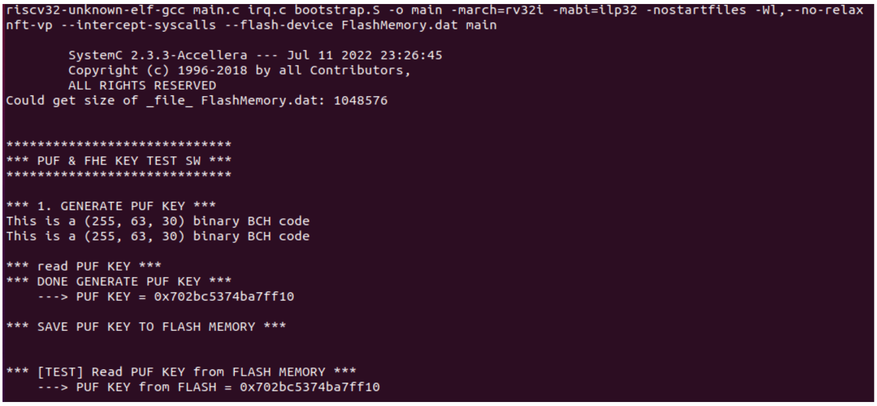
To evaluate and verify the double-encryption security, we added the process of calculating the BCH encoding for the encrypted data. As shown in Fig. 4, the FHE module sends the encrypted data to the BCH module to request BCH encoding, and the BCH module performs the BCH encoding and returns it to the FHE module. The FHE module returns the BCH encryption data to the SecureKeyGen module, which copies the data to the main memory. The SecureKeyGen module then generates an interrupt to notify the application software of encryption completion. The application software copies the encrypted data from the main memory to its own buffer and writes them to the flash memory.

4. Experiments and Verification

4.1 Applications and Verification

Based on the RISC-V virtual platform implemented in SystemC, we added secure key generation modules such as PUF and FHE. The RISC-V virtual platform consists of a virtual platform that configures the hardware system and an application software component that can be executed on the virtual platform. Therefore, it is possible to verify and conduct experiments using a specific application software on the virtual platform.

First, we performed functional verification using simple application software for the implemented SecureKeyGen module. The verification software performs the following operations for the PUF and FHE: for PUF, it transmits the PUF generation command through the memory-mapped register of the SecureKeyGen and waits until it receives an interrupt signal, indicating that the PUF key has been generated from the SecureKeyGen module. Then, it reads the generated PUF key from the corresponding memory area and saves it in flash memory. The results of the functional verification of PUF key generation and retrieval are shown in Fig. 5. As the figure shows, a 64-bit PUF key is typically generated for a PUF set with BCH encoding (255, 63, 30). As the PUF key is a unique value generated using device-specific information, its uniqueness and randomness must be guaranteed. The random number may not be unique depending on the seed, because the PUF implemented in this study was generated with a PRNG number generated using a random seed. To determine the uniqueness of PUF keys, we generated 50,000 keys and evaluated their uniqueness. Fig. 6 presents the results of the uniqueness experiment with 50,000 PUF keys. As shown in Fig. 6, the uniqueness maintains a value higher than 94% up to 50,000. However, as the number of PUF keys increases, the uniqueness gradually decreases. This is considered a limitation of the PRNG generator and the seed number integer value generation. It is expected that uniqueness can be further improved if the randomness is further increased by shuffling and shifting using an additional seed with a PRNG operation.



```
riscv32-unknown-elf-gcc main.c irq.c bootstrap.S -o main -march=rv32i -mabi=ilp32 -nostartfiles -Wl,--no-relax
nft-vp --intercept-syscalls --flash-device FlashMemory.dat main

        SystemC 2.3.3-Accellera --- Jul 11 2022 23:26:45
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
Could get size of _file_ FlashMemory.dat: 1048576


*****************************
*** PUF & FHE KEY TEST SW ***
*****************************

*** 1. GENERATE PUF KEY ***
This is a (255, 63, 30) binary BCH code
This is a (255, 63, 30) binary BCH code

*** read PUF KEY ***
*** DONE GENERATE PUF KEY ***
    ---> PUF KEY = 0x702bc5374ba7ff10

*** SAVE PUF KEY TO FLASH MEMORY ***


*** [TEST] Read PUF KEY from FLASH MEMORY ***
    ---> PUF KEY from FLASH = 0x702bc5374ba7ff10
```

Fig. 5. Results of functional verification for PUF key generation and retrieval.

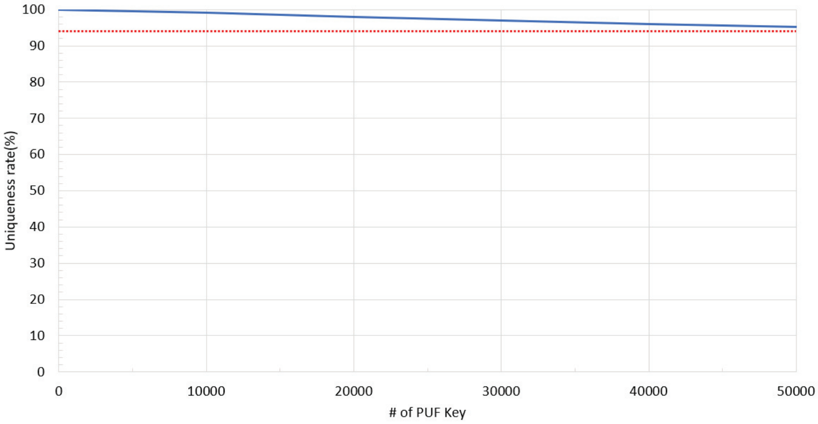
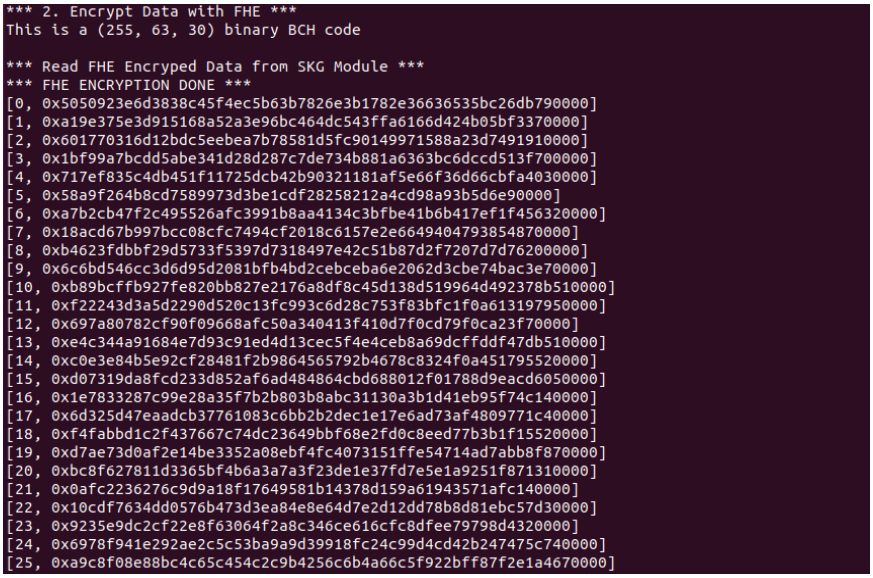


Fig. 6. Results of randomness and uniqueness for PUF key generation and retrieval.



```
*** 2. Encrypt Data with FHE ***
This is a (255, 63, 30) binary BCH code

*** Read FHE Encryped Data from SKG Module ***
*** FHE ENCRYPTION DONE ***
[0, 0x5050923e6d3838c45f4ec5b63b7826e3b1782e36636535bc26db790000]
[1, 0xa19e375e3d915168a52a3e96bc464dc543ffa6166d424b05bf3370000]
[2, 0x601770316d12bdc5eebea7b78581d5fc90149971588a23d7491910000]
[3, 0x1bf99a7bcdd5abe341d28d287c7de734b881a6363bc6dccd513f700000]
[4, 0x717ef835c4db451f11725dcb42b90321181af5e66f36d66cbfa4030000]
[5, 0x58a9f264b8cd7589973d3be1cdf28258212a4cd98a93b5d6e90000]
[6, 0xa7b2cb47f2c495526afc3991b8aa4134c3bfbe41b6b417ef1f456320000]
[7, 0x18acd67b997bcc08cfc7494cf2018c6157e2e66494047938548700000]
[8, 0xb4623fdbbf29d5733f5397d7318497e42c51b87d2f7207d7d76200000]
[9, 0x6c6bd546cc3d6d95d2081bfb4bd2cebceba6e2062d3cbe74bac3e70000]
[10, 0xb89bcffb927fe820bb827e2176a8df8c45d138d519964d492378b510000]
[11, 0xf22243d3a5d2290d520c13fc993c6d28c753f83bfc1f0a613197950000]
[12, 0x697a80782cf90f09668afc50a340413f410d7f0cd79f0ca23f70000]
[13, 0xe4c344a91684e7d93c91ed4d13cec5f4e4ceb8a69dcffddf47db510000]
[14, 0xc0e3e84b5e92cf28481f2b9864565792b4678c8324f0a451795520000]
[15, 0xd07319da8fcd233d852af6ad484864cbd688012f01788d9eacd6050000]
[16, 0x1e7833287c99e28a35f7b2b803b8abc31130a3b1d41eb95f74c140000]
[17, 0x6d325d47eaadcb37761083c6bb2b2dec1e17e6ad73af4809771c40000]
[18, 0xf4fabbd1c2f437667c74dc23649bbf68e2fd0c8eed77b3b1f15520000]
[19, 0xd7ae73d0af2e14be3352a08ebf4fc4073151ffe54714ad7abb8f870000]
[20, 0xbc8f627811d3365bf4b6a3a7a3f23de1e37fd7e5e1a9251f871310000]
[21, 0x0afc2236276c9d9a18f17649581b14378d159a61943571afc140000]
[22, 0x10cdf7634dd0576b473d3ea84e8e64d7e2d12dd78b8d81ebc57d30000]
[23, 0x9235e9dc2cf22e8f63064f2a8c346ce616cfc8dfee79798d4320000]
[24, 0x6978f941e292ae2c5c53ba9a9d39918fc24c99d4cd42b247475c740000]
[25, 0xa9c8f08e88bc4c65c454c2c9b4256c6b4a66c5f922bff87f2e1a4670000]
```

Fig. 7. Results of functional verification for FHE encryption and retrieval of the FHE data.

Subsequently, FHE functional verification experiments were performed. To perform FHE encryption, the application software first copies the text data to the memory area and sends the FHE encryption command to the SecureKeyGen module. It waits until it receives an interrupt signal, indicating that the FHE encryption has been completed by the SecureKeyGen module. If the interrupt arrives, it reads the encrypted FHE data from the corresponding memory area and saves them to flash memory. The results of the functional verification of the FHE encryption and the retrieval of the FHE data are shown in Fig. 7. As shown in Fig. 7, the FHE-encrypted data are created for the corresponding text data in code format. It was decrypted in addition to its original text format.

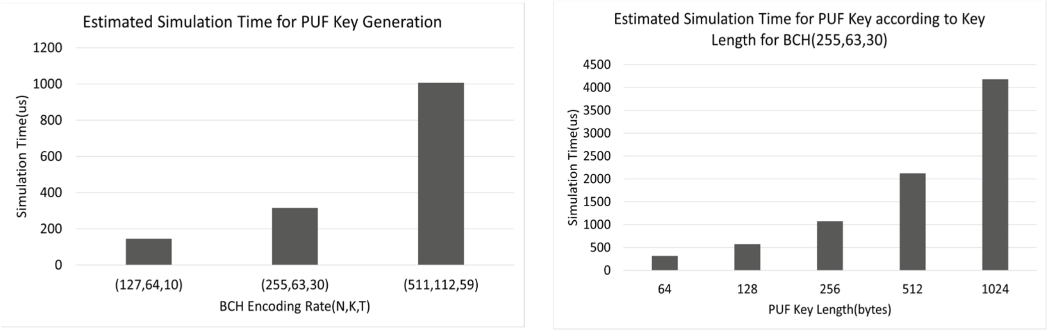
4.2 Experiments on PUF Key Generation

We conducted experiments to estimate the key generation times of the PUF module in the RISC-V virtual platform. The experiments executed the PUF key generation command in the software of the RISC-V virtual platform, received the result, and stored it in flash memory. The estimated simulation time was measured during this sequence. For proper estimation, time events provided by SystemC primitive were appropriately applied to the major modules and intermodule interfaces according to the features of the modules. The basic configuration values of the processor, memory, bus, and secure key generation modules for the time events are listed in Table 1. Based on the unit time for each module configured in the table, a PUF key generation simulation was performed, and the execution times were measured.

We measured and estimated how the PUF key generation time changed as the BCH encoding levels changed. The experimental results of the estimated simulation time for the PUF key generation according to the BCH encoding configuration level are shown in Fig. 8(a). In this study, we configured three BCH levels: (127,64,10), (255,63,30), and (511,112,59). BCH encoding is important in PUF key generation, because the BCH encoding level is a crucial configuration parameter that reflects the randomness and error robustness of the PUF key. As shown in the results, the execution time increased as the BCH encoding scheme became more complex. For the PUF key, BCH encoding provides the following characteristics: the longer the BCH codeword length, the higher the randomness, the higher the error correctability, and the higher the probability of guaranteeing uniqueness. However, the higher the complexity of the BCH encoding scheme and the longer the codeword and PUF key lengths, the longer the execution time related to the PUF key. In addition, we experimented with generating a PUF key while changing the PUF key length for a specific BCH encoding. Fig. 8(b) shows the experimental results of the estimated simulation time for the PUF key generation at BCH (255,63,30) as the key length varies from 64 to 1024. By estimating the execution time for the three BCH encoding schemes, we can analyze the correlation between the PUV key length and the PUF operation time and provide insight into determining the appropriate PUF key length and BCH encoding scheme for designing a custom embedded device.

Table 1. Basic configuration values of processor, memory, bus, and secure key generation modules for the time events

Configuration parameter	Unit time (ns)
CPU cycle time	10
CPU memory access	40
Bus data rate per byte	10
SKG execution cycle	20



(a) (b)

Fig. 8. (a) Experimental results of estimated simulation time for PUF key generation according to the BCH encoding configurations (127,64,10), (255,63,30), and (511,112,59), and (b) estimated simulation time for PUF key length for BCH (255,63,30) encoding.

In summary, the experiment and verification process confirmed that the PUF and FHE modules operated properly for the PUF generation and FHE encryption functions. The approximate execution times for the PUF and FHE modules were estimated based on the configuration unit time of each module on the virtual platform. Verification of the experiments was performed using the developed micro-level software. It is suitable for functional inspection and operation verification of operations such as internal PUF enrollment and regeneration, FHE NTT operations, and BCH modules. However, greater rationality of the methodology may be required in terms of system-level verification or performance evaluation.

4.3 Discussion

We presented experimental results on the uniqueness and execution time of FHE and PUF on the RISC-V virtual platform, as well as their functional verifications. As inferred from the experimental results, it can be regarded as an acceptable level of performance verification when an embedded device operates in a small-scale network environment. For verification at a more sophisticated system level, it is necessary to perform encryption data transmission verification through FHE transmission and the reception of PUF data in a network system. For a more sophisticated analysis of the virtual platform simulation of the execution time, the execution of multiple instances and performance analysis are required. Additionally, there may be a lack of scalability validation in large-scale networks. If in a large-scale network, to be usable for modules on many devices, the probability of uniqueness should be higher, and a guaranteed execution time should be provided. The generation of random seeds is important for increasing the uniqueness. In our virtual platform, it is necessary to upgrade the seed generation algorithm in the PRNG, and it may also be necessary to increase the key size to enhance randomness. Enhancing the PRNG algorithm or increasing the key size also increases the complexity and execution time of the module, which also affects the scalability. We confirmed that the execution time gradually increased with key length in the previous experiment. To guarantee scalability of key generation time, it is necessary to increase parallelism through the parallel operation of modules and the pipelined data flow of the data path between modules.

5. Conclusion

As the data used in an IoT system or blockchain contain valuable personal information, personal device authentication and data encryption are important for protecting personal information. A PUF is useful for generating keys that guarantee device uniqueness, and homomorphic encryption is effective for safe data distribution in networks. However, to include security functions, such as PUF and FHE, in embedded devices for IoT or blockchain systems, proper inspection is required to design and implement embedded SoC modules through overhead and performance analysis.

In this paper, an embedded process-based virtual platform was designed and implemented for security key generation and homomorphic encryption of data. Our SoC VP, which simulates the RISC-V embedded processor, was implemented using SystemC, which can run and verify the secure key generation module at the ESL and analyze the system-level execution time, memory footprint, and performance, such as randomness and uniqueness. We experimentally verified the secure key generation module, and estimated the execution of the PUF key and FHE encryption based on the unit time of each module. With the SoC virtual platform developed in this study, we further conducted research that utilizes the PUF and FHE for various functions required for data security and operations in IoT or blockchain systems. In future work, we will further secure methodological relationality by performing functional verification and performance evaluation of FHE-encrypted transmission and reception of PUF data at a large-scale networked system level.

Acknowledgement

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government (No. 22ZT1100, ICT convergence technology support and development based on local industry in the metropolitan area). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1F1A1048026). This work was supported by Hankuk University of Foreign Studies Research Fund.

References

[1] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, “A PUF-based secure communication protocol for IoT,” *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 3, article no. 67, 2017. https://doi.org/10.1145/3005715

[2] Y. Zhang, B. Li, Y. Wang, J. Wu, and P. Yuan, “A blockchain-based user remote autentication scheme in IoT systems using physical unclonable functions,” in *Proceedings of 2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, Nanjing, China, 2020, pp. 1100-1105. https://doi.org/10.1109/ICSIP49896.2020.9339402

[3] Z. Li, Y. Chu, X. Liu, Y. Zhang, J. Feng, and X. Xiang, “Physical unclonable function based identity management for IoT with blockchain,” *Procedia Computer Science*, vol. 198, pp. 454-459, 2022. https://doi.org/10.1016/j.procs.2021.12.269

[4] D. Kim, U. Jo, Y. Kim, Y. E. Eko, and H. Kim, “Design and implementation of a blockchain based interworking of oneM2M and LWM2M IoT systems,” *Journal of Information Processing Systems*, vol. 19, no. 1, pp. 89-97, 2023. https://doi.org/10.3745/JIPS.01.0093

[5] Z. Siddiqui, J. Gao, and M. K. Khan, “An improved lightweight PUF–PKI digital certificate authentication scheme for the Internet of Things,” *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19744-19756, 2022. https://doi.org/10.1109/JIOT.2022.3168726

[6] H. H. Kim and J. Yoo, “Analysis of security vulnerabilities for IoT devices,” *Journal of Information Processing Systems*, vol. 18, no. 4, pp. 489-499, 2022. https://doi.org/10.3745/JIPS.03.0178

[7] C. Bohm and M. Hofer, *Physical Unclonable Functions in Theory and Practice*. New York, NY: Springer, 2013. https://doi.org/10.1007/978-1-4614-5040-5

[8] R. Maes, “PUF-based entity identification and authentication,” in *Physically Unclonable Functions*. Heidelberg, Germany: Springer, 2013, pp. 117-141. https://doi.org/10.1007/978-3-642-41395-7_5

[9] A. Al-Meer and S. Al-Kuwari, “Physical unclonable functions (PUF) for IoT devices,” *ACM Computing Surveys*, vol. 55, no. 14s, article no. 314, 2023. https://doi.org/10.1145/3591464

[10] M. Marcantoni, B. Jayawardhana, M. P. Chaher, and K. Bunte, “Secure formation control via edge computing enabled by fully homomorphic encryption and mixed uniform-logarithmic quantization,” *IEEE Control Systems Letters*, vol. 7, pp. 395-400, 2022. https://doi.org/10.1109/LCSYS.2022.3188944

[11] G. Xu, J. Zhang, and L. Wang, “An edge computing data privacy-preserving scheme based on blockchain and homomorphic encryption,” in *Proceedings of 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)*, Huaihua City, China, 2022, pp. 156-159. https://doi.org/10.1109/ICBCTIS55569.2022.00044

[12] V. Pal, B. S. Acharya, S. Shrivastav, S. Saha, A. Joglekar, and B. Amrutur, “PUF based secure framework for hardware and software security of drones,” in *Proceedings of 2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Kolkata, India, 2020, pp. 1-6. https://doi.org/10.1109/AsianHOST51057.2020.9358264

[13] J. Choi, B. Ahn, S. Pedavalli, S. Ahmad, A. Villasenor, and T. Kim, “Secure firmware update and device authentication for smart inverters using blockchain and phyiscally uncloable function (PUF)-embedded security module,” in *Proceedings of 2021 6th IEEE Workshop on the Electronic Grid (eGRID)*, New Orleans, LA, USA, 2021, pp. 1-4. https://doi.org/10.1109/eGRID52793.2021.9662155

[14] G. Vaidya, T. V. Prabhakar, and L. Manjunath, “GPIO PUF for IoT devices,” in *Proceedings of 2020 IEEE Global Communications Conference*, Taipei, Taiwan, 2020, pp. 1-6. https://doi.org/10.1109/GLOBECOM42002.2020.9322590

[15] Wikipedia, “SystemC,” 2024 [Online]. Available: https://en.wikipedia.org/wiki/SystemC.

[16] B. Bailey, G. Martin, and A. Piziali, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. San Francisco, CA: Morgan Kaufmann Publishers, 2007.

[17] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, “The RISC-V instruction set manual (Volume I: Base user-level ISA),” Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA, *Technical Report No. UCB/EECS-2016-118*, 2017. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.pdf

[18] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovic, “The RISC-V instruction set manual (Volume II: Privileged architecture),” Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA, *Technical Report No. UCB/EECS-2016-116*, 2017. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-161.pdf

[19] 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,” in *Cryptography and Security: From Theory to Applications*. Heidelberg, Germany: Springer, 2012, pp. 300-318. https://doi.org/10.1007/978-3-642-28368-0_20

[20] Y. N. Imamverdiev and L. V. Sukhostat, “A method for cryptographic key generation from fingerprints,” *Automatic Control and Computer Sciences*, vol. 46, pp. 66-75, 2012. https://doi.org/10.3103/S0146411612020022

[21] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura, “Cryptographie key generation from PUF data using efficient fuzzy extractors,” in *Proceedings of the 16th International Conference on Advanced Communication Technology*, Pyeongchang, South Korea, 2014, pp. 23-26. https://doi.org/10.1109/ICACT.2014.6778915

[22] H. Kang, Y. Hori, and A. Satoh, “Performance evaluation of the first commercial PUF-embedded RFID,” in *Proceedings of the 1st IEEE Global Conference on Consumer Electronics*, Tokyo, Japan, 2012, pp. 5-8. https://doi.org/10.1109/GCCE.2012.6379926

[23] D. Ismari and J. Plusquellic, “IP-level implementation of a resistance-based physical unclonable function,” in *Proceedings of 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Arlington, VA, USA, 2014, pp. 64-69. https://doi.org/10.1109/HST.2014.6855570

[24] H. Akhundov, E. Van der Sluis, S. Hamdioui, and M. Taouil, “Public-key based authentication architecture for IoT devices using PUF,” 2020 [Online]. Available: https://arxiv.org/abs/2002.01277.

[25] M. S. E. Quadir and J. A. Chandy, “Embedded systems authentication and encryption using strong PUF modeling,” in *Proceedings of 2020 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, 2020, pp. 1-6. https://doi.org/10.1109/ICCE46568.2020.9043104

[26] A. A. Pour, V. Beroulle, B. Cambou, J. L. Danger, G. Di Natale, D. Hely, S. Guilley, and N. Karimi, “PUF enrollment and life cycle management: solutions and perspectives for the test community,” in *Proceedings of 2020 IEEE European Test Symposium (ETS)*, Tallinn, Estonia, 2020, pp. 1-10. https://doi.org/10.1109/ETS48528.2020.9131578

[27] C. Gentry, “A fully homomorphic encryption scheme,” Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2009.

[28] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, Bethesda, MD, USA, 2009, pp. 169-178. https://doi.org/10.1145/1536414.1536440

[29] F. Armknecht, C. Boyd, C. Carr, K. Gjosteen, A. Jaschke, C. A. Reuter, and M. Strand, “A guide to fully homomorphic encryption,” 2015 [Online]. Available: https://eprint.iacr.org/2015/1192.

[30] O. Ozerk, C. Elgezen, A. C. Mert, E. Ozturk, and E. Savas, “Efficient number theoretic transform implementation on GPU for homomorphic encryption,” *The Journal of Supercomputing*, vol. 78, pp. 2840-2872, 2022. https://doi.org/10.1007/s11227-021-03980-5

[31] V. Herdt, D. Grosse, H. M. Le, and R. Drechsler, “Extensible and configurable RISC-V based virtual prototype,” in *Proceedings of 2018 Forum on Specification & Design Languages (FDL)*, Garching, Germany, 2018, pp. 5-16. https://doi.org/10.1109/FDL.2018.8524047

[32] S. H. Lim, W. W. Suh, J. Y. Kim, and S. Y. Cho, “RISC-V virtual platform-based convolutional neural network accelerator implemented in SystemC,” *Electronics*, vol. 10, no. 13, article no. 1514, 2014. https://doi.org/10.3390/electronics10131514

[33] S. H. Lim, S. H. Kang, B. H. Ko, J. Roh, C. Lim, and S. Y. Cho, “An integrated analysis framework of convolutional neural network for embedded edge devices,” *Electronics*, vol. 11, no. 7, article no. 1041, 2022. https://doi.org/10.3390/electronics11071041

[34] M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30, 1998. https://doi.org/10.1145/272991.272995

[35] R. Morelos-Zaragoza, “Encoder/decoder for binary BCH codes in C,” 1994 [Online]. Available: https://www.eccpage.com/bch3.c.

Seung-Ho Lim https://orcid.org/0000-0003-3096-0785

He received B.S., M.S., and Ph.D. degrees in the Division of Electrical Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2001, 2003, and 2008, respectively. He worked in the memory division of Samsung Electronics Co. Ltd. from 2008 to 2010, where he engaged in developing a high-performance solid state disk for server storage systems. Since 2010, he has been working as a professor at the Division of Computer Engineering at Hankuk University of Foreign Studies. His research interests include deep learning for embedded systems, interconnect network, distributed system, flash memory and security system design.

Hyeok-Jin Lim https://orcid.org/0009-0009-8269-4416

He received B.S., M.S., and Ph.D. degrees in the Department of Information and Communication Engineering, Sejong University in 2012, 2014, and 2018, respectively. He worked in the Samsung Engineering Mega Solution (SEMES) Co. Ltd. from 2019 to 2021. Since 2021, he has been working as a senior at the Electronics and Telecommunications Research Institute. His research interests include SoC design for low-cost system and AI semiconductor system.

Seong-Cheon Park https://orcid.org/0000-0002-8716-6366

He received B.S. degrees from the Division of Electronics and Information Engineering, Seoul National University of Science and Technology, Seoul, Korea, in 2002. The M.S. degree from the Division of Electronics and Computer Sciences, Korea University, Seoul, Korea, in 2008. He completed a doctoral course in the field of embedded security SoC at the Graduate School of Electronic Engineering, Hanyang University, Seoul, Korea, in 2012. Currently he joined at Electronics and Telecommunications Research Institute, where he has been working in the area of hardware security circuit design. His research interests include hardware security technology for smart ICT devices and ICs.