PAPER

Bitstream Protection in Dynamic Partial Reconfiguration Systems Using Authenticated Encryption

Yohei HORI[†a)], ***Member***, **Toshihiro KATASHITA**[†], ***Nonmember***, **Hirofumi SAKANE**[†], **Kenji TODA**[†], ***and*** **Akashi SATOH**[††], ***Members***

SUMMARY Protecting the confidentiality and integrity of a configuration bitstream is essential for the dynamic partial reconfiguration (DPR) of field-programmable gate arrays (FPGAs). This is because erroneous or falsified bitstreams can cause fatal damage to FPGAs. In this paper, we present a high-speed and area-efficient bitstream protection scheme for DPR systems using the Advanced Encryption Standard with Galois/Counter Mode (AES-GCM), which is an authenticated encryption algorithm. Unlike many previous studies, our bitstream protection scheme also provides a mechanism for error recovery and tamper resistance against configuration block deletion, insertion, and disorder. The implementation and evaluation results show that our DPR scheme achieves a higher performance, in terms of speed and area, than previous methods.

key words: *dynamic partial reconfiguration (DPR), field-programmable gate array (FPGA), Advanced Encryption Standard (AES), Galois/Counter Mode (GCM), authenticated encryption*

1. Introduction

Dynamic partial reconfiguration (DPR) is of great use in field-programmable gate arrays (FPGAs) for realizing compact, cost-efficient, and multifunctional dedicated hardware systems. Under DPR, one portion of a circuit is dynamically replaced with another while the rest of the circuit remains fully operational. Thus, by using DPR, the functionality of the system is dynamically altered by replacing hardware modules according to, e.g., user-requested content, performance requirements, and environmental changes. Many applications of DPR have been reported, including content distribution security [1], image processing [2], fault-tolerant systems [3], and software-defined radio [4].

In a DPR system, hardware configuration data (i.e., *bitstream*) can be downloaded from the Internet. Hence, a DPR system is under constant threat of attack from bitstream piracy (i.e., illegal cloning), tampering, reverse engineering, and Trojan insertion. As such, protecting the confidentiality and integrity of bitstreams is a major concern.

Based on the above considerations, we have developed a DPR system where bitstreams are protected using the Advanced Encryption Standard (AES) [5] in Galois/Counter Mode (GCM) [6], [7]. AES-GCM is one of the latest authenticated encryption (AE) algorithms, which guarantee both the confidentiality and the integrity of a message, and could therefore be an effective tool for DPR systems. Indeed, data encryption and authentication can be achieved with two separate algorithms, but if their area and speed performance are not balanced, the overall performance will be determined by the worst-performing algorithm. Therefore, we expect AE to enable a more area-efficient, high-speed DPR implementation. Because other AE algorithms cannot be parallelized or pipelined and are thus not necessarily suitable for hardware implementation [8], the use of AES-GCM is currently the best solution for protecting bitstreams.

Thus far, several systems without a recovery mechanism have been reported [9]–[11]. We developed a bitstream protection scheme for DPR systems using AES-GCM in [12], and then, we added an error recovery mechanism and investigated the optimal internal memory size in [13]. This advanced DPR system provided bitstream confidentiality, bitstream integrity, and error recovery mechanisms. This paper reports a revised and extended version of our previous studies, in which we have added a novel tamper-resistance mechanism against bitstream block (BSB) deletion, insertion, and disorder. We call such tampering a *BSB disorder attack*. BSB disorder attacks have previously been mentioned in [13], but they were not implemented or evaluated.

Following our work in 2008, several related studies have been reported, but most seem to underestimate the importance of dividing a bitstream into BSBs and the danger of BSB attacks. Some studies use only one message authentication code (MAC) to check the integrity of an entire bitstream; however, secure DPR cannot be achieved in this manner because the decrypted bitstream cannot be used for configuration before its integrity is verified (*cf.* Sect. 5.1). The decrypted bitstream must be stored in the internal buffer until its integrity has been confirmed, and therefore, a bitstream must be divided into small BSBs. In our previous implementation, although the integrity of each BSB was successfully guaranteed, BSB disorder attacks could not be prevented. For example, our previous DPR system could not detect an attack in which two BSBs were swapped unless the BSBs were themselves altered.

To the best of our knowledge, this study represents the first DPR system to incorporate all four mechanisms of bitstream encryption, bitstream verification, error recovery, and tamper-resistance against BSB disorder attacks. This paper describes the architecture, memory configuration, implementation results, and performance evaluation of

Manuscript received June 12, 2013.

[†]The authors are with the Research Institute for Secure Systems (RISEC), National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba-shi, 305–8568 Japan.

[††]The author is with The University of Electro-Communications, Chofu-shi, 182–8585 Japan.

a) E-mail: hori.y@aist.go.jp

DOI: 10.1587/transinf.E96.D.2333

an AES-GCM-based DPR system featuring an error recovery mechanism. The system is implemented in an off-the-shelf Virtex-5 board, allowing us to successfully demonstrate the bitstream encryption, verification, and error recovery features. The remainder of this paper is organized as follows. Section 2 discusses some previous studies on DPR security. Then, Sect. 3 explains the partial reconfiguration process in a Xilinx FPGA. Section 4 presents a brief overview of cryptographic algorithms related to our implementation. Section 5 describes the architecture of our DPR system and explains the functions implemented within. Section 6 presents the implementation and evaluation results of our DPR system. Finally, Sect. 7 summarizes this paper and outlines some ideas for future work.

2. Related Work

Recent commercial FPGAs can be configured from encrypted bitstreams through built-in decryptors. For example, the Xilinx Virtex-II and Virtex-II Pro support the Triple Data Encryption Standard (Triple-DES) [14] with a 56-bit key, whereas Virtex-4 through Virtex-7 support AES with a 256-bit key. The key is stored in the dedicated volatile memory inside the FPGA. Therefore, the storage must always be supplied with power through an external battery. Unfortunately, the functionality of configuration through encrypted bitstreams is not available when using DPR, and if the device is configured using the built-in bitstream decryptor, the DPR function is disabled [15]. In DPR systems, therefore, partial bitstreams must be decrypted by utilizing user logic.

Bossuet et al. proposed a secure configuration method for DPR systems [9]. Their approach allows the use of arbitrary cryptographic algorithms, because the bitstream decryptor itself is implemented as a reconfigurable module. However, although their method uses bitstream encryption, it does not consider the authenticity of the bitstreams.

Zeineddini and Gaj developed a DPR system with two separate algorithms for bitstream protection [10]; they implemented AES for bitstream encryption and SHA-1 for authentication. AES and SHA-1 were implemented as C programs and run on two embedded microprocessors: PowerPC and MicroBlaze. The total processing time required for the authentication, decryption, and configuration of a 14-KB bitstream was approximately 400 ms on PowerPC and 2.3 s on MicroBlaze. Such performance levels, however, would be insufficient for practical DPR systems.

Parelkar used AE to protect FPGA bitstreams [11] using various AE algorithms such as Offset CodeBook (OCB) [16], Counter with CBC-MAC (CCM) [17], and EAX [18]. To compare the performance of the AE method with separate encryption and authentication methods, SHA-1 and SHA-512 were also implemented using AES-CBC (Cipher Block Chaining).

Following our previous studies [12], [13], several secure update methods were proposed [19]–[21], but these were unable to protect the integrity of the bitstream.

Drimer and Kuhn proposed a secure remote update protocol that performed decryption and authentication [22], but this required a non-volatile external memory and protected interfaces, i.e., the non-volatile memory and FPGA must be contained in a tamper-proof package.

Devic et al. used three pairs of secret keys for the secure remote update of the FPGA [23] because the TAGs (i.e., the version of the circuit) before and after update must be securely transferred between the device and the system designer. However, the management of three secret keys is more complicated than that of one key. In our DPR system using AES-GCM, the security tag is not secret and need not be encrypted; hence, one pair of secret keys is sufficient for secure DPR.

Kepa et al. developed SeReCon [24], which aimed to ensure the integrity of a DPR system based on public-key cryptography including a trusted authority. In contrast to the method presented in this paper, SeReCon performs public-key cryptography and other controls using a MicroBlaze processor, and it does not focus on the throughput or area performance of the security module.

Here, we should emphasize that none of the above studies discuss the importance of bitstream splitting and buffering or BSB disorder attacks.

3. Partial Reconfiguration

This section briefly describes some features of the partial reconfiguration of the Xilinx FPGA employed in the proposed system. In 2010, partial reconfiguration was officially supported in Xilinx Integrated Software Environment (ISE) version 12. In the older version, only Early Access Partial Reconfiguration (EAPR) patches were opened to limited users. For more detailed information on Xilinx partial reconfiguration, see [25], [26].

3.1 Partial Reconfiguration Overview

In this study, we develop a DPR system using the latest DPR design flow. The latest flow supports Virtex-5, Virtex-6, and all 7-series devices. In Xilinx FPGAs, the module to be dynamically replaced is called the *Reconfigurable Module (RM)*, and the area where the RM is placed is called the *Reconfigurable Partition (RP)*. RM can be an arbitrarily sized rectangle. Figure 1 shows an example of the partially reconfigurable design.

The smallest unit of a bitstream that can be accessed

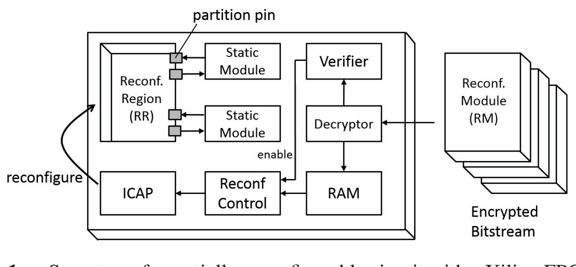


Fig. 1 Structure of a partially reconfigurable circuit with a Xilinx FPGA.

Table 1 The port descriptions of ICAP [27].

Port	Direction	Width	Function
O	Output	32	Configuration data output bus
Busy	Output	32	Busy/Ready output
I	Input	32	Configuration data input bus
WRITE	Input	1	Active Low Write Input
CE	Input	1	Active Low Clock Enable Input
CLK	Input	1	Clock Input

is called a *frame*. In Virtex-5 devices, a frame is a 1,312-bit configuration of information corresponding to the height of 20 configurable logic blocks. An RM bitstream is a collection of frames. Each device family has different frame structures, but this paper does not focus on other devices.

3.2 Partition Pins

All signals between an RM and a fixed module must pass through *partition pins* to lock the wiring. The partition pin is automatically inserted at the RM boundary during the first mapping process of the DPR design, and it can then be freely moved to any place on the RM boundary by the designer. Partition pins correspond to the *bus macro* in the old EAPR design flow, although the bus macro must be explicitly described in the source code and manually placed inside the RM.

3.3 Internal Configuration Access Port

Virtex-II as well as newer Virtex series and all 7-series devices support *self DPR* through the *Internal Configuration Access Port (ICAP)*. The I/O port descriptions are provided in Table 1, which is referred from [27]. ICAP works in the same manner as the SelectMAP configuration interface. As the user logic can access the configuration memory through ICAP, partial reconfiguration of the FPGA can be controlled by internal circuits. In Virtex-5 devices, the bus width of ICAP is 32 bits and the actual data width can be either 8, 16, or 32 bits. The configuration data are not necessarily continuously sent to port I. By controlling the write enable signal (WRITE), the configuration of RM can be suspended and resumed after some other tasks, *e.g.* integrity check of the bitstream to be loaded.

4. Cryptographic Algorithms

4.1 Advanced Encryption Standards

AES is a symmetric key block cipher algorithm standardized by the U.S. National Institute of Standards and Technology (NIST) [5]. Whereas the previous DES [28] had a Feistel network architecture, AES employs a substitution–permutation network (SPN) architecture. Under AES, the block length is 128 bits, and the key length is selected to be 128, 196, or 256 bits.

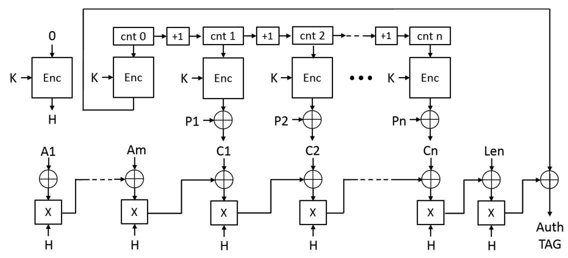


Fig. 2 Example of operation of Galois/Counter Mode (GCM).

4.2 Galois/Counter Operation Mode

GCM [6] is one of the latest operation modes to be standardized by NIST [7]. Figure 2 shows an example of the GCM operation mode.

To generate a MAC, also known as a *security tag*, GCM uses universal hashing based on product-sum operations in the finite Galois field (GF) $GF(2^w)$. This enables faster and more compact hardware implementation compared to integer computation. In GCM, the encryption and decryption is based on the CTR mode of operation [29], which can be highly parallelized and pipelined. Therefore, GCM is suitable for hardware implementation, and it affords a wide variety of performance advantages such as compactness and high speed [30], [31]. Other AE algorithms are not necessarily suitable for hardware implementation as they are impossible to parallelize or pipeline [8].

AES-GCM is basically a GCM application that uses AES as the encryption core. Because AES is also based on the product-sum operation in $GF(2^w)$, compact or high-speed hardware implementations are possible. Therefore, the use of AES-GCM can meet various performance requirements and is the best solution for protecting FPGA bitstreams in DPR systems.

The security tag is calculated using the *GHASH* function defined below, where A denotes additional authentication data; C, the ciphertext; and H, the hash subkey.

$$X_i = \begin{cases} 0 & i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & i = 1, \dots, m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{128-v})) \cdot H & i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & i = m+1, \dots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* \| 0^{128-u})) \cdot H & i = m+n \\ (X_{m+n} \oplus (\mathrm{len}(A) \| \mathrm{len}(C))) \cdot H & \\ & i = m+n+1 \end{cases} \tag{1}$$

The final value X_{m+n+1} becomes the security tag. In the *GHASH* function, the 128×128-bit multiplication over GF is achieved using a 128×16-bit GF multiplier eight times to save hardware resources. Figure 3 shows the GF multiplier implemented in the AES-GCM module. The partial products of the 128×16-bit multiplier are summed in the 128-bit register Z, and the calculation of Z finishes in 8 clock cycles.

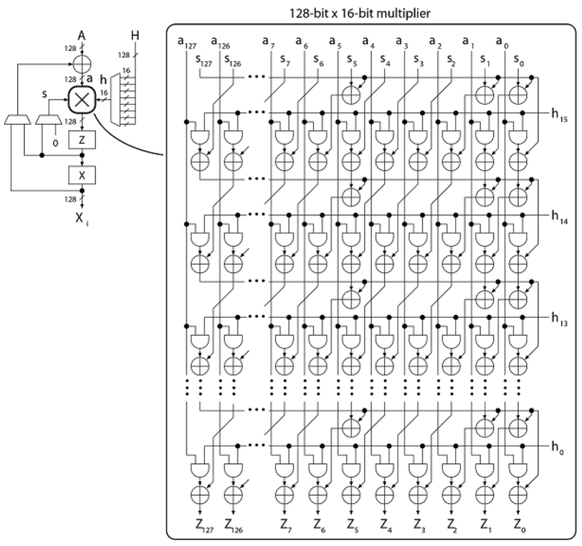


Fig. 3 Architecture of Galois Field multiplier.

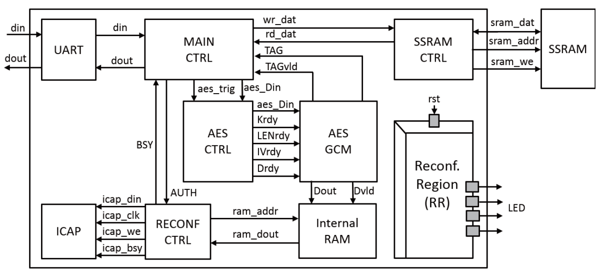
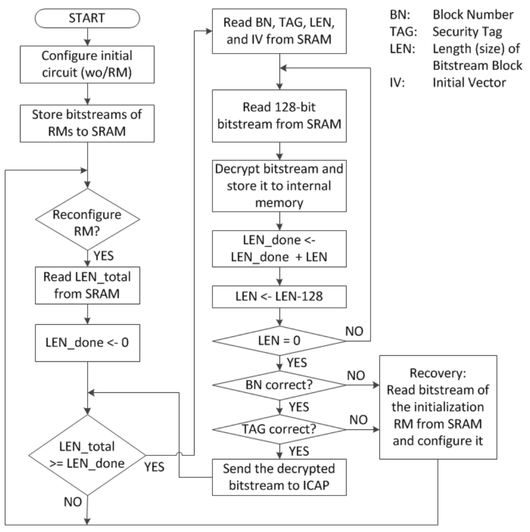


Fig. 4 Overview of the PR-AES-GCM system.



START
Configure initial circuit (wo/RM)
Store bitstreams of RMs to SRAM
Reconfigure RM?
Read LEN_total from SRAM
LEN_done <- 0
LEN_total >= LEN_done
Read BN, TAG, LEN, and IV from SRAM
Read 128-bit bitstream from SRAM
Decrypt bitstream and store it to internal memory
LEN_done <- LEN_done + LEN
LEN <- LEN-128
LEN = 0
BN correct?
TAG correct?
Send the decrypted bitstream to ICAP
Recovery: Read bitstream of the initialization RM from SRAM and configure it
BN: Block Number
TAG: Security Tag
LEN: Length (size) of Bitstream Block
IV: Initial Vector

Fig. 5 Flowchart of the PR-AES-GCM system.

4.3 Secure Hash Algorithm

SHA is a cryptographic hash function that generates a message of a particular length. It is widely used to guarantee a message's authenticity. Currently, five types of SHA are defined (SHA-1, SHA-224, SHA-256, SHA-386, and SHA-512), with the suffix denoting the length of the output message digest (the output length of SHA-1 is 160 bits). The latter four algorithms are collectively referred to as SHA-2. Because SHA-1 reportedly suffers from a security vulnerability [32], SHA-2 should be used for message authentication.

5. DPR System with AES-GCM

This section describes the architecture of our DPR systems. We first explain the AES-GCM-based DPR system (hereafter referred to as *PR-AES-GCM*). For comparison, we then describe a DPR system with AES-CBC and SHA-256 (hereafter referred to as *PR-AES-SHA*).

5.1 Configuration Flow Overview

The block diagram and the flowchart of our proposed PR-AES-GCM system is shown in Figs. 4 and 5, respectively. The PR-AES-GCM is supposed to be included in a user system such as consumer electronics, and the function of the system can be partly changed by downloading bitstreams of various RMs. To protect the confidentiality and integrity of the bitstream, it is encrypted and transferred with its security tag. Since the PR-AES-GCM is a prototype here, encrypted bitstreams of RMs are transferred from the host computer via RS232 and are stored in the external 36x256K-bit SSRAM. The decryption key is also sent from a host computer and set to an internal register. In a practical system, transferring a key via a network is insecure, and therefore, the key should be safely stored or generated in the FPGA. For this purpose, one of the solutions is to use a physical unclonable function (PUF) [33] for generating a key [34]–[36]. However, the main purpose of this study is to clarify the speed and area performance of the AES-GCM-based DPR system, and all of the target systems for performance comparison (*cf.* Sect. 6.2) omit the key storing and generation modules. Therefore, the key storing and generation mothods are beyond the scope of this paper but remain as future work.

The configuration of the RM starts when a configuration command is sent from the host computer. The downloaded bitstreams are decrypted by the AES-GCM module, and their authenticity is verified simultaneously. As the plain bitstreams must not leak out to the device, the decrypted bitstreams must be stored in the internal memory (Block RAM). In the system, the memory size is set to 128x2^k bits, and it is at most 128x8192 (1 Mb) due to device resource limitations. As the size of the internal memory is relatively small, large bitstreams are split into several blocks, and decryption and verification is performed for each bitstream block. As explained in Sect. 3.3, a bitstream need not be continuously transferred to ICAP, and thus, such bitstream splitting is possible. To distinguish the divided bitstream block from the AES 128-bit data block, we define the former as *Bitstream Block* (*BSB*). By appending a security tag to each BSB, decryption and verification of each BSB can be independently performed. The size of the BSB is multiple of the AES block (128 bits), and consequently is multiple of the width of ICAP data bus (32 bits). Therefore, the decrypted BSB can be simply transferred to ICAP without considering a fractional ($<$ 32 bits) bitstream. The decrypted BSBs compose a bitstream that is completely equivalent to the original one as long as they are concatenated in the correct order.

It should be mentioned that AES-GCM requires initial processing such as key scheduling and initial vector (IV) setup for each BSB. Therefore, the computation effort for the same bitstream increases with the number of BSBs. The smaller the internal memory, the more compact the system will be; however, the computation effort will increase. Conversely, if the memory size is large, the computation effort will decrease, although the system will require more hardware resources. Furthermore, as additional data such as a security tag, IV, and data length are attached to each BSB, the size of the downloaded data increases with the number of BSBs.

5.2 Countermeasure against BSB Removal and Insertion

The consideration is that simply dividing a bitstream into several BSBs will be vulnerable to the removal, insertion, and swapping of BSBs. We refer to such attacks as *BSB disorder attacks*. Note that AES-GCM can detect whether a BSB has been tampered with, but it does not check the number or order of successive BSBs. For example, even if one of the successive BSBs is removed, AES-GCM cannot detect its disappearance, and therefore, the system would be incompletely configured. In addition, if a malicious BSB with a correct security tag is inserted into the series of BSBs, AES-GCM will recognize the inserted BSB as being legitimate, and therefore, the malicious BSB will be configured in the device, causing system malfunction, data leakage, and so on. Therefore, DPR systems require some protection scheme to prevent BSB disorder attacks.

We implemented a scheme to protect the system from BSB disorder attacks. We included a sequential block number (BN) in the BSB and used it as a part of the IV; for example, the least 32 bits of the IV are the BN. Appending a sequential number to packet data is a general technique and used in, for example, the IP authentication header [37] of IPsec [38], and it is quite effective for securing a BSB-based DPR system. The BN denotes the position of the BSB in the bitstream, and it need not start with 0. The BN of the first BSB is used as the initial BN and stored in the internal register or memory. The stored BN is incremented and used as the IV every time a BSB is loaded. If the loaded BSB has a different BN from the stored value, the configuration is immediately terminated and the recovery process is started. Here, it should be noted that the BN is not secret information, and therefore, only an integrity check is required.

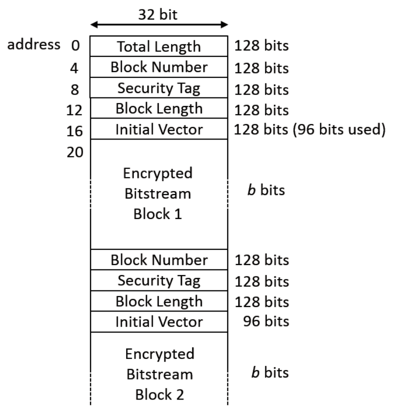


Fig. 6 General structure of bitstreams stored in SSRAM.

5.3 Data Structure

To decrypt an RM bitstream with AES-GCM, information about the security tag, data length, and IV need to be appended to the head of the bitstream. Large bitstreams are divided into several BSBs, and each BSB contains such header information. In addition, the first BSB contains information about the total bitstream length. Figure 6 shows the structure of the downloaded bitstream together with the header information, which is loaded from SSRAM and set to the registers in the AES-GCM module when the RM configuration begins.

5.4 Bitstream Decryption and Verification

In the AES-GCM module, the major component (S-box) is implemented using a composite field. The initial setup of AES-GCM requires 63 cycles, and the first BSB requires an 19 additional cycles for setting up the total length of the entire bitstream. A 128-bit data block is decrypted in 13 clock cycles, including the SSRAM access time, and the decrypted data are stored in the internal memory. The last block of the BSB requires 10 clock cycles in addition to the usual 13 for calculating the security tag. The security tag is calculated using the *GHASH* function defined below, where A is the additional authentication data; C, the ciphertext; and H, the hash subkey.

An example timing chart of the AES-GCM module including the initial setup is shown in Fig. 7, and the functions of the signals are described in Table 2. Suppose that the size of the entire bitstream is S bits and that it is split into n BSBs. Let the size of the kth BSB be b_k bits, and $b_1, b_2, \ldots, b_{n-1}$ be BSBs of the same size b. Then, the entire size S is expressed as follows:

$$S = \sum_{k=1}^{n} b_k = \sum_{k=1}^{n-1} b + b_n = (n-1) \cdot b + b_n. \tag{2}$$

As Fig. 7 illustrates, the number of clock cycles T_{aes} required for the decryption and verification of the entire bitstream is

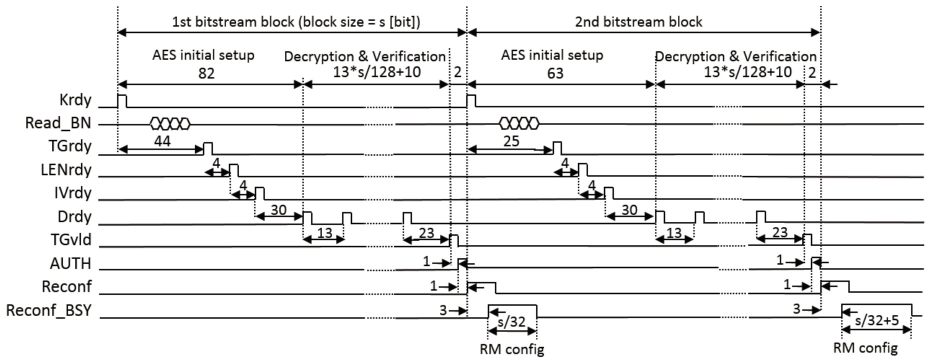


Fig. 7 Timing chart of decryption, verification, and reconfiguration in PR-AES-GCM.

Table 2 Descriptions of the signals in the PR-AES-GCM system.

Signal	Description
Krdy	Asserted when the key of AES-GCM is ready.
BN	Block Number. Loaded from SSRAM and set to an internal register.
TGrdy	Asserted when the expected security tag is loaded from SSRAM and set to an internal register.
LENrdy	Asserted when the length of BSB is loaded from SSRAM and set to an internal register.
IVrdy	Asserted when the initial vector is loaded from SSRAM and set to an internal register.
Drdy	Asserted when the AES block data are loaded from SSRAM. AES-GCM runs immediately after this signal is asserted.
TGvld	Asserted when the security tag of the BSB is calculated by AES-GCM.
AUTH	Asserted when the calculated security tag of the BSB is identical to the expected one.
Reconf	Asserted to start RM configuration.
Reconf_BSY	Asserted while the RM configuration is performed.

$$T_{aes} = 19 + (n-1)\cdot\left(63 + 13\cdot\frac{b}{128} + 10 + 2\right) + \left(63 + 13\cdot\frac{b_n}{128} + 10 + 2\right) \tag{3}$$

$$= 19 + \frac{13\,(n-1)\,b + 13\,b_n}{128} + 74\,n$$
$$= \frac{13\,S}{128} + 74\,n + 19 \qquad (\because S = (n-1)\,b + b_n)\,. \tag{4}$$

As the above equation indicates, the computation effort for AES-GCM increases with the number of BSBs n.

5.5 RM Configuration

Unlike other DPR systems, our system does not use an embedded processor to control the partial reconfiguration. The input data and control signals from the ICAP are directly connected to and controlled by the user logic. Thus, our system does not suffer from the delay of processor buses. In the system, the width of the ICAP data port is set to 32 bits. In Virtex-5, the maximum frequency of the ICAP is limited to 100 MHz; therefore, the ideal throughput of the reconfiguration process is 3,200 Mbps.

Figure 7 also shows the timing of the configuration of the RM bitstream. When the size of the BSB is b bits, the configuration of the BSB finishes in $b/32$ cycles. The last BSB requires 5 additional cycles to flush the buffer in the device. Therefore, the required number of computation cycles for the RM configuration T_{reconf} is

$$T_{reconf} = (n-1)\cdot\frac{b}{32} + \left(\frac{b_n}{32} + 5\right) = \frac{S}{32} + 5$$
$$(\because S = (n-1)\,b + b_n). \tag{5}$$

5.6 Error Recovery

In the system, the first several bytes of the SSRAM are reserved for the *initialization RM*, which is used for recovering the system from DPR errors. The use of the initialization RM enables the system to return to the start-up state without rebooting the entire system. Thus, processes executed in other modules can retain their data even when DPR errors occur. The bitstream of the initialization RM is encrypted and processed in the same manner as that of other RMs. If the bitstream size is S bits, the computation time for decryption, verification, and configuration is derived from Eqs. (4) and (5).

When bitstream verification fails with AES-GCM, the current process is abandoned and configuration of the initialization RM is started. Note that the unauthorized BSB is still in the internal memory and it will be overwritten by the initial RM. Therefore, the unauthorized bitstream will be safely erased and will not be configured in the system. If the verification of the initialization RM fails due to, for example, bitstream tampering or memory bus damage, the system discontinues the configuration process and prompts the user to reboot the system.

5.7 DPR System with AES-CBC and SHA-256

To compare the performance of the AE method and the separate encryption/authentication method, we also implement PR-AES-SHA, which utilizes AES-CBC and SHA-256 for bitstream encryption and authentication. Instead of taking results from previous studies, we develop and evaluate our own AES-CBC and SHA-256 modules. This is because it is quite difficult to fairly compare the performance of our system with the previous studies owing to the differences in the target device, implemented functions, and so on. Many previous studies of AES and SHA use older FPGAs; our system uses Virtex-5 for implementation. Several AES studies use Virtex-5, but the implemented functions are not identical; some implement an Electric Cipher Book (ECB) mode [29], some do not include key scheduling, and some do not include a decryption core. Furthermore, many previous studies focus on the performance only of the core of AES or SHA, whereas our system includes DPR, SSRAM, and UART controllers and so on. Considering these issues, we develop AES-CBC and SHA-256 and implement them in the same environment as PR-AES-GCM; this should be the best way to enable a fair comparison.

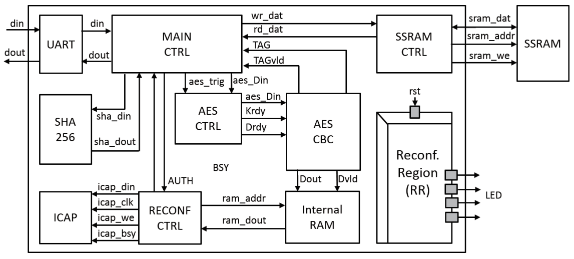


Fig. 8 Overview of the PR-AES-SHA system.

Figure 8 shows a block diagram of the DPR system with AES-CBC and SHA-256. SHA-256 is selected because the old SHA-1 has been reported to suffer from a security vulnerability [32]. Although CBC is not sufficiently secure for practical use, it is selected because it has the simplest mode of operation.

Similarly to AES-GCM, the S-box of the AES is implemented as a table using Block RAM. When the system is booted up, the AES-CBC module requires 18 clock cycles for the setup process. In AES-CBC, a 128-bit block is decrypted in 11 clock cycles.

Figure 9 shows the hash calculation timing chart and Table 3 provides descriptions of the signals in the AES-CBC and SHA-256 modules. As the data bus of the SHA-256 module has a 32-bit width, it requires 16 clock cycles to load the input data and 8 cycles to read the output. A 512-bit message block is calculated in 49 cycles.

It requires more cycles to process the message using SHA-256 than using AES; therefore, the overall throughput of the bitstream processing is restricted by SHA-256. Although the SHA-256 algorithm is relatively simple, it cannot be pipelined or parallelized, and therefore, it is difficult to improve the performance of the SHA-256 module.

Reconfiguration of the RM starts after the bitstream authenticity has been inspected by the SHA-256 module. The RM reconfiguration performance in the AES-CBC + SHA-

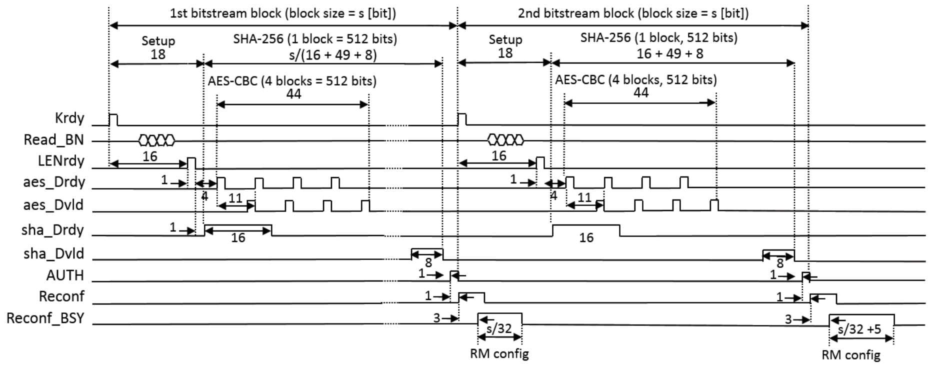

1st bitstream block (block size = s [bit])
2nd bitstream block (block size = s [bit])
Setup 18
SHA-256 (1 block = 512 bits) s/(16 + 49 + 8)
Setup 18
SHA-256 (1 block, 512 bits) 16 + 49 + 8
AES-CBC (4 blocks = 512 bits) 44
AES-CBC (4 blocks, 512 bits) 44
Krdy
Read_BN
LENrdy
aes_Drdy
aes_Dvld
sha_Drdy
sha_Dvld
AUTH
Reconf
Reconf_BSY
16
1
4
11
16
8
1
1
3
s/32
RM config
s/32 +5
RM config

Fig. 9 Timing chart of the PR-AES-SHA.

Table 3 Descriptions of the signals in the PR-AES-SHA system.

Signal	Description
Krdy	Asserted when the AES-CBC key is ready.
BN	Block Number. Loaded from SSRAM and set to an internal register.
LENrdy	Asserted when the length of BSB is loaded from SSRAM and set to an internal register.
aes_Drdy	Asserted to start AES-CBC after a data block is loaded from SSRAM.
aes_Dvld	Asserted when AES decryption of a block finishes.
sha_Drdy	Asserted to start SHA-256 after a data block is loaded from SSRAM.
sha_Dvld	Asserted when calculation of SHA-256 finishes.
AUTH	Asserted when the calculated hash tag of the BSB is identical to the expected one.
Reconf	Asserted to start RM configuration.
Reconf_BSY	Asserted while the RM configuration is performed.

Table 4 Hardware utilization of static module of PR-AES-GCM on Virtex-5 (XC5VLX50T).

Module	Register	(%)	LUT	(%)	Slice	(%)
Overall	3,169	11.0%	3,938	13.7%	1,538	21.4%
AES-GCM	1,454	5.0%	2,187	7.6%	969	13.5%
MAIN_CTRL	1,013	3.5%	922	3.2%	419	5.8%
AES_CTRL	302	1.0%	263	0.91%	105	1.5%
SSRAM_CTRL	96	0.33%	66	0.23%	27	0.38%
RECONF_CTRL	81	0.28%	124	0.43%	40	0.56%
RAM_CTRL	140	0.49%	142	0.49%	38	0.53%
UART	36	0.13%	41	0.14%	17	0.27%

Table 5 Hardware utilization of static module of PR-AES-SHA on Virtex-5 (XC5VLX50T).

Module	Register	(%)	LUT	(%)	Slice	(%)
Overall	2,500	8.7%	3,865	13.4%	1,938	26.9%
AES-CBC	664	2.3%	953	3.3%	537	7.5%
SHA256	499	1.7%	1016	3.5%	620	8.6%
MAIN_CTRL	463	1.7%	1,133	4.1%	713	5.2%
AES_CTRL	149	0.52%	151	0.52%	46	0.64%
SSRAM_CTRL	103	0.4%	218	0.8%	132	1.0%
RECONF_CTRL	81	0.28%	124	0.43%	40	0.56%
RAM_CTRL	139	0.48%	141	0.49%	36	0.50%
UART	36	0.13%	41	0.14%	17	0.27%

256 system is the same as that in the AES-GCM system.

6. Implementation

This section describes the implementation results of the PR-AES-GCM and PR-AES-SHA systems. Both systems are targeted at Virtex-5 (XC5VLX50T-FFG1136) on an ML505 board, and it was verified that DPR could be successfully implemented on PR-AES-GCM and PR-AES-SHA. The systems are designed using Xilinx ISE 14.2i and PlanAhead 14.2.

To test whether all mechanisms of bitstream encryption, verification, error recovery, and BSB attack prevention work properly, we implemented two simple function blocks, a 28-bit up-counter and a 28-bit down-counter, as RMs. The most significant 4 bits of the counter were outputted from the RM and connected to LEDs on the board.

6.1 Hardware Resource Utilization

Table 4 shows the hardware utilization of PR-AES-GCM implemented on a Virtex-5. The “Overall” column shows the total amount of hardware resources used by all modules except RM. Table 4 also shows the hardware utilization of each module as a standalone implementation.

The hardware architecture of Virtex-5 is vastly different from that of earlier devices such as Virtex-II Pro and Virtex-4. Each slice in Virtex-5 contains four 6-input LUTs, whereas that of earlier devices contains two 4-input LUTs. Thus, the number of slices is smaller in the Virtex-5 implementation.

6.2 Performance Evaluation

The clock frequency of PR-AES-GCM and PR-AES-SHA is 100 MHz. To enable a comparison with [10], the computation time required to configure a 14,112-byte (112,896-bit) RM is described in Table 6. Decryption, verification, and configuration with PR-AES-GCM can be implemented in a pipeline.

In PowerPC and MicroBlaze systems, authentication, decryption, and reconfiguration are performed sequentially, and therefore, the overall processing time is simply the sum of the processing times of each step. Table 6 also shows the throughput of other AE algorithms as reported in [11].

6.3 Analysis of Results

The implementation results and performance evaluation described in the previous sections indicate that all functions such as bitstream decryption, verification, configuration, and error recovery work properly. Thus, the above-described system is the first operational DPR system featuring both bitstream protection and error recovery mechanisms.

As shown in Table 6, PR-AES-GCM achieved the highest overall throughput of over 900 Mbps with a slice utilization of only around 1/3. Note that PR-AES-GCM includes error recovery logic, an SSRAM controller, etc. Additionally, the AES-GCM module achieved a throughput of around 913 Mbps, which is faster than those of AES-SHA and other AE methods such as OCB, CCM, and EAX. Interestingly, the number of registers and LUTs used in AES-GCM is larger than those in AES-CBC and SHA256; the number of slices used in AES-GCM is smaller than that in AES-CBC and SHA256. This shows that the registers and LUTs in AES-GCM are easy to pack into a single slice, reducing the overall usage of slices. This is one of the advantages of using AE where encryption and authentication are processed simultaneously.

Furthermore, PowerPC and MicroBlaze DPR systems

Table 6 Comparison of performances of different PR systems with bitstream protection schemes (14,112-byte RM).

System	Device	Slice	Verification	Decryption	Configuration	Overall	Ratio
PR-AES-GCM	XC5VLX50T	1,538*	119.110 μs		35.3 μs	123.72 μs	1
			947.8 Mbps		3195 Mbps	913 Mbps	
PR-AES-SHA	XC5VLX50T	1,938*	160.79 μs	97.14μs	35.3 μs	196.27 μs	1.59
			701 Mbps	1164 Mbps	3200 Mbps	575 Mbps	
PowerPC [10]	XC2VP30	1,334**	139 ms	208 ms	56 ms	403 ms	3257
			812 kbps	543 kbps	2016 kbps	280 kbps	
MicroBlaze [10]	XC2VP30	1,706**	776 ms	1472 ms	32 ms	2280 ms	18429
			145 kbps	77 kbps	3528 kbps	50 kbps	
AES-OCB [11]	XC4VLX60	2,964	601 Mbps		-	-	
AES-CCM [11]	XC4VLX60	2,799	255 Mbps		-	-	
AES-EAX [11]	XC4VLX60	2,993	287 Mbps		-	-	

* Slice utilization of Virtex-5 is shown.
** Includes only reconfiguration controllers.

require an overall computation time of between several hundred milliseconds and several seconds, which is unacceptable for practical DPR systems. Therefore, authentication, decryption, and reconfiguration should be processed using dedicated hardware in order to realize practical DPR systems. Compared to software AE systems, our approach attained extremely high performance, where PR-AES-GCM achieved 3,257 times higher throughput than the PowerPC system and 18,429 times higher throughput than the MicroBlaze system.

7. Conclusions

We developed a DPR system with AES-GCM that protects both the confidentiality and authenticity of FPGA bitstreams. AES-GCM achieved a throughput of around 950 Mbps with reasonable resource utilization, and the entire system achieved a throughput of around 913 Mbps, which is sufficient for practical DPR use.

We also implemented AES-CBC and SHA-256 to compare the performance and resource utilization. The AES-CBC + SHA-256 system was slightly more area efficient than AES-GCM, but it is not suitable for high-speed implementation. Although AES can achieve a wide range of performance levels, from compact to high speed, SHA is a straightforward algorithm that cannot be effectively parallelized or pipelined. Therefore, the performance of the AES/SHA-based system will always be restricted by the SHA module.

The AES-GCM-based system achieved higher throughput than other operation modes such as OCB, CCM, and EAX. The results show that AES-GCM is currently one of the most promising approaches for protecting the confidentiality and authenticity of a bitstream in DPR systems.

The future work of this study includes to implement a secure key generation and exchange scheme using, for example, a PUF. Another direction of this study would be to implement countermeasures against various attacks such as side-channel attacks [39], [40] and fault attacks [41], [42].

Acknowledgements

The authors are grateful to Xilinx Inc., who supported our study on DPR, as well as Tokyo Electron Device Co., Ltd., who supported us in developing the secure DPR system.

References

[1] Y. Hori, H. Yokoyama, H. Sakane, and K. Toda, "A secure content delivery system based on a partially reconfigurable FPGA," IEICE Trans. Inf. & Syst., vol.E91-D, no.5, pp.1398–1407, May 2008.

[2] C. Claus, J. Zeppenfeld, F. Muller, and W. Stechele, "Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system," DATE'07, pp.498–503, 2007.

[3] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, "Dynamic fault tolerance in FPGAs via partial reconfiguration," FCCM 2000, pp.165–174, 2000.

[4] J.P. Delahaye, G. Gogniat, C. Roland, and P. Bomel, "Software radio and dynamic reconfiguration on a DSP/FPGA platform," J. Frequenz, vol.58, no.5-6, pp.152–159, 2004.

[5] U.S. Department of Commerce/National Institute of Standards and Technology, "Announcing the advanced encryption standard (AES)," FIPS PUB 197, Nov. 2001.

[6] D.A. McGrew and J. Viega, "The Galois/counter mode of operation (GCM)," May 2005. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html

[7] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, SP 800-38D ed. National Institute of Standards and Technology, Nov. 2007.

[8] D.A. McGrew and J. Viega, "The security and performance of the Galois/counter mode (GCM) of operation," INDOCRYPT 2004, pp.343–355, 2004.

[9] L. Bossuet and G. Gogniat, "Dynamically configurable security for SRAM FPGA bitstreams," Int. J. Embedded Systems, vol.2, no.1/2, pp.73–85, 2006.

[10] A.S. Zeineddini and K. Gaj, "Secure partial reconfiguration of FPGAs," ICFPT'05, pp.155–162, 2005.

[11] M.M. Parelkar, Authenticated encryption in hardware, Master's thesis, George Mason University, 2005.

[12] Y. Hori, A. Satoh, H. Sakane, and K. Toda, "Bitstream encryption and authentication with aes-gcm in dynamically reconfigurable systems," FPL 2008, pp.23–28, 2008.

[13] Y. Hori, A. Satoh, H. Sakane, and K. Toda, "Bitstream encryption and authentication using AES-GCM in dynamically reconfigurable systems," IWSEC 2008, pp.261–278, 2008.

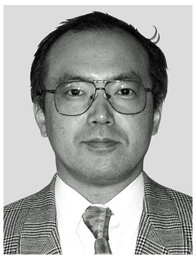
[14] National Institute of Standards and Technology, "Recommendation for the triple data encryption algorithm (TDEA) block cipher," May

2004.
[15] Xilinx, Inc., Virtex-5 FPGA Configuration User Guide (UG191, v3.11), 2012.
[16] P. Rogaway, M. Bellare, and B. John, "OCB: A block-cipher mode of operation for efficient authenticated encryption," ACM Trans. Information and System Security, vol.6, no.3, pp.365–403, Aug. 2003.
[17] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," RFC3610, Sept. 2003.
[18] M. Bellare, P. Rogaway, and D. Wagner, "A conventional authenticated-encryption mode," http://www-08.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf, 2003.
[19] A. Errandani, A. Doumar, and E. Chatêlet, "Secure configuration schemes for fpga-based systems with simple key management," SPL VIII, 2012.
[20] W. Adi and K. Benkrid, "Ultimate design security in self-reconfiguring non-volatile environments," AHS 2010, pp.230–234, 2010.
[21] S. Gören, O. Ozkurt, A. Yildiz, and H.F. Ugurdag, "FPGA bitstream protection with PUFs, obfuscation, and multi-boot," ReCoSoc 2011, 2011.
[22] S. Drimer and M.G. Kuhn, "A protocol for secure remote updates of FPGA configurations," ARC 2009, pp.50–61, 2009.
[23] F. Devic, L. Torres, and B. Badrignans, "Secure protocol implementation for remote bitstream update preventing replay attacks on FPGA," FPL 2010, pp.179–182, 2010.
[24] K. Kępa, F. Morgan, K. Kościuszkiewicz, and T. Surmacz, "SeReCon: a ecure reconfiguration controller for self-reconfigurable systems," Int. J. Critical Computer-Based Systems, vol.1, no.1/2/3, pp.86–103, 2010.
[25] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs," FPL'06, pp.12–17, 2006.
[26] Xilinx, Inc., Partial Reconfiguration User Guide UG702 v14.2, 2012.
[27] Xilinx, Inc., Virtex-5 Libraries Guide for HDL Designs (UG621, v11.3), 2009.
[28] U.S. Department of Commerce/National Institute of Standards and Technology, "Data encryption standard (DES)," FIPS PUB 46-3, 1999.
[29] M. Dworkin, Recommendation for Block Cipher Modes of Operation, SP 800-38A ed. National Institute of Standards and Technology, Dec. 2001.
[30] A. Satoh, "High-speed parallel hardware architecture for Galois counter mode," ISCAS'07, pp.1863–1866, 2007.
[31] A. Satoh, T. Sugawara, and T. Aoki, "High-speed pipelined hardware architecture for Galois counter mode," ISC'07, pp.118–129, 2007.
[32] W. Xiaoyun, Y. Yiqun, and Y. Hongbo, "Finding collisions in the full SHA-1," CRYPTO 2005, pp.17–36, 2005.
[33] S.R. Pappu, Physical One-Way Functions, Ph.D. thesis, MIT, 2001.
[34] J. Guajardo, S.S. Kumar, G.J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," CHES'07, pp.63–80, 2007.
[35] J. Guajardo, S.S. Kumar, G.J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," FPL'07, pp.189–195, 2007.
[36] K. Kepa, F. Morgan, and K. Kosciuszkiewicz, "IP protection in partially reconfigurable FPGAs," FPL 2009, pp.403–409, 2009.
[37] S. Kent, "IP authentication header," RFC 4302, 2005.
[38] S. Kent and S. Karen, "Security architecture for IP," RFC 4301, 2005.
[39] P.C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," CRYPTO'96, pp.104–113, 1996.
[40] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO'99, pp.388–397, 1999.
[41] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," CRYPTO'97, pp.513–25, 1997.
[42] S.P. Skorobogatov and R.J. Anderson, "Optical fault induction attacks," CHES'02, pp.2–12, 2002.

Yohei Hori received his BE, ME, and PhD degrees from the University of Tsukuba, Ibaraki, Japan, in 1999, 2001, and 2004, respectively. After receiving his PhD, he spent five years as a postdoctoral researcher at the National Institute of Advanced Industrial Science and Technology (AIST). He moved to Chuo University as a research scientist in 2008, before returning to AIST in 2010 as a researcher. His current research interests include partially reconfigurable systems, side-channel analysis, fault analysis, and physically unclonable functions. He is a member of IEICE, IEEE, IEEE-CS, and IPSJ.

Toshihiro Katashita completed the doctoral program of the Graduate School of Systems and Information Engineering, University of Tsukuba, in 2006, whereupon he joined AIST as a fixed-term researcher. In 2008, he became a tenure-track researcher at AIST. He is involved in research projects on high-performance computation circuit design and hardware security. He is engaged in the development of cryptographic hardware and software as well as side-channel attack experiments.

Hirofumi Sakane received his BE degree from Yamaguchi University in 1990 and ME from the University of Electro-Communications (UEC) in 1992. He subsequently joined Electrotechnical Laboratory and initially studied parallel computer architecture. He joined the doctoral program of the Graduate School of Information Systems of UEC in 1998, and he received his PhD in 2001. He has been working as a senior researcher at AIST since 2001. His current research interests include developing test methods and metrics for the side-channel security of cryptographic modules. In connection with his research, he was also engaged in the standardization of security requirements for cryptographic modules from 2008 through 2012 at NIST.



Kenji Toda received an MS degree from Keio University, Japan, in 1982. He is the leader of the Real-Time Embedded System Semi-Group at AIST. His research interests include real-time computing, embedded systems, and network applications. He is a member of IPSJ and IEICE.

Akashi Satoh received BS and MS degrees in Electrical Engineering from Waseda University, Tokyo, Japan, in 1987 and 1989, respectively. In 1989, he joined IBM Research, Tokyo Research Laboratory, and was involved in the research and development of digital and analog VLSI circuits. He received a PhD in Electrical Engineering from Waseda University in 1999. In 2007, he joined the National Institute for Advanced Industrial Science and Technology (AIST) and managed the SASEBO Project. Since 2013, he has been with the University of Electro-Communications where he is currently a Professor of the Graduate School of Informatics and Engineering. His current research interests include algorithms and architectures for data security and high-performance VLSI implementations.