We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 4

A Double Cipher Scheme for Applications in Ad Hoc Networks and its VLSI Implementations

Masa-aki Fukase

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/56145

1. Introduction

The ubiquitous environment was described as a vision for 21st century computing [1]. In the last ten years, the ubiquitous network has become one of the remarkable trends of information and communications technology. One of the most important characteristics of the ubiquitous network is open access anytime anywhere. This corresponds to the mobility and diversity of power conscious PC processors, mobile processors, cryptography processors, RFID tags, and so forth. In view of the desire for better cost performance, simplicity, functionality, usability, and so on, the ad hoc network is an emerging technology for next generation ubiquitous computing [2]. However, these specific features involve fundamental issues as follows.

i. Currently, the promotive force of diversity is not the conventional wired network based on large servers, but convenient wireless LANs and handheld small devices, like the PDA (personal digital assistant), mobile phone and so forth. Although the diversity of various platforms is inevitable, it also causes notorious security issues such as insecurity, security threats or illegal attacks, such as tapping, intrusion and pretension. Nowadays, WEP (Wired Equivalent Privacy) is not so effective for wireless LANs. Worldwide diversity vs. the security threat are the two faces characteristic of the ubiquitous network [3]. Safety of the ubiquitous network has two aspects. One is the front-end security of the ubiquitous device and the other is the protection of the multimedia data itself, stored in the ubiquitous device. Neither approach always promises complete safety, but they complement one another. A cutting-edge technique for front-end security is the TPM (trusted platform module), commonly known as the security chip. Since the TPM implements RSA (Rivest-Shamir-Adelman), it works for short, password-size text data, and its major role is

INTECH
open science | open minds

© 2013 Fukase; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

implicitly digital signing. In view of the running time, the encryption of long-length multimedia data, such as an image, is definitely outside the TPM. The other approach, back-end security, protects huge amounts of data because multimedia information, crucial for the interaction between ubiquitous devices and human beings, uses massive amounts of data. Back-end security is usually covered by common key schemes. Common key module-embedded processors are built-in cryptography processors for IC cards and portable electronic devices.

ii. On the other hand, considering that cipher algorithms are open to third parties in the evaluation of cipher strength, hardware specifications are more important than cipher strength in developing an ad hoc network infrastructure. A fundamental issue in maintaining the mobility of the ubiquitous environment is how to achieve power-saving mobility. The power consuming factors of ubiquitous networks are large server systems controlled by network providers and small ubiquitous platform systems, handheld devices and so forth. These small systems consist of processors, memory and displays. The power dissipation of memory strongly depends on that of a memory cell and the memory space required for embedded software. LCDs (liquid crystal displays) and processors consume similar power in running mobile devices. While the LCD is turned on only when it displays some information, processors are always in the standby state to receive calls. Thus, power-saving restrictions for mobile devices are inevitably imposed on the processors.

iii. Another issue of ubiquitous computing is its strong dependency on embedded software. This has a crucial effect on the total design of ubiquitous devices. Performable features of ubiquitous devices in processing multimedia data have mainly relied on embedded software. For example, if a new protocol appears, embedded software requires users to download an update package. Despite embedding, so far, it has rapidly increased software size due to the RTOS (real time OS), firmware, application software and so on. This results in more software costs and wider memory space. Since cutting-edge ubiquitous devices need not only sophisticated and complicated processing, but also power conscious high-speed operation, the embedded software approaches taken so far will not always continue to play key roles in ubiquitous computing.

In order to solve the fundamental issues described above, power conscious management of ubiquitous networks and cryptographic protection of massive data spreading over ubiquitous networks are required. It is really the practice of cryptography network security technologies to show optimum design for the trade-off to achieve specific features of the ubiquitous network. The double cipher scheme presented in this paper combines two cipher algorithms [4]. One is random number addressing cryptography (RAC), closely related to the internal behaviour of the processors. RAC is a transposition cipher devised from the direct connection of a built-in random number generator (RNG), a register file and a data cache. The register file plays the role of a streaming buffer. A random store, based on the direct connection, scrambles

http://dx.doi.org/10.5772/56145

or transposes a series of multimedia data at random without any special encryption operation. The other algorithm of the double cipher is a data sealing algorithm. This is implemented during the data transfer from the register file to the data cache, by using another built-in RNG. This complements the RAC's shortcoming and enhances the security of the data information as a whole.

Since the double cipher scheme uses built-in RNGs at the micro-operation level, it is more effective than normal usage based on the processing of random number operands at the instruction level. In addition, the double cipher scheme requires no additional chip area or power dissipation. A linear feedback shift register (LFSR) is used as the RNG to achieve a longer cycle with negligible additional area. Thus, the double cipher scheme is a microarchitecture-based, software-transparent hardware cipher that offers security for the whole data with negligible hardware cost and moderate performance overhead. This is very suited to very large scale integration (VLSI) implementation. The VLSI implementation of the double cipher follows the multicore structure for bi-directional communication and multiple pipelines for multimedia processing and cipher streaming [5]. The cipher streaming is executed by the SIMD (Single Instruction stream Multiple Data stream) mode cipher and decipher codes. They do not attach operands, as is described above, but repeat instances to transfer byte-structured data from a register file to a data cache.

In this paper, we describe the double cipher scheme, hardware algorithm, architectural organization, structural aspect, internal behaviour and VLSI implementation of the double cipher in a sophisticated ubiquitous processor named HCgorilla by using a 0.18-μm standard cell CMOS chip. We evaluate the prospective specifications of the HCgorilla chip with respect to the hardware resource or cost, power dissipation, throughput and cipher strength. Potential aspects compared with the usual security techniques, cipher techniques and cryptography module-embedded processors are also described. HCgorilla is a power conscious hardware approach that provides multimedia data with practical security over a ubiquitous network.

2. Preliminaries

The problem statement and the course of the research of this study are described in more detail in this section. In addition, the area of application is also explained.

2.1. Trends and issues of ubiquitous networks

Faced with the progressive ubiquitous environment, we have experienced the alternative requirements, diversity or security [1]. Figure 1 illustrates that the diversity of the various devices has caused illegal attacks, intrusions, pretensions and so forth. When diversity is from the small mobile phone and the PDA to large traditional servers, in normal circumstances ubiquitous networks are functional and useful, but they are hard to control in abnormal circumstances. Since diversity brings about open access to ubiquitous networks anytime anywhere, this really threatens user security.



Figure 1. Diversity vs. security threat in a ubiquitous network

In order to achieve secure ubiquitous networks, both machines and data must be protected from abnormal phenomena. Table 1 surveys the current status of security techniques related to the ubiquitous network. Since tremendous network issues need complicated algorithms to detect and recognize individual phenomena, in the main, software techniques have been used. However, they are inflexible to individual demands, and are not always sufficient from the practical viewpoint. The hardware implementation of an IDS (intrusion detection system) and an IPS (intrusion prevention system) is another result we exploited independently [6]. As is clear from Table 1, cryptography is used for the protection of ubiquitous platforms. The cryptography adopted for the front-end security of an individual machine is public key cryptography to protect short, password-size text data. On the other hand, common key cryptography is used for the protection of data. Comparing the numerical values in Table 1 is irrelevant because they strongly depend on process technologies.

Ubiquitous device	Technique		Target	Speed		Secureness
	HW/SW	Cryptographic means		Running time	Transfer rate	
HCgorilla	Hardware	Common key	Full text	Short	160-320 Mbps	Practical
Security chip		Public key	Password	Out of account for multimedia data		Large
Secure coprocessor						
Cryptographic core						
Elliptic curve processor						
Cryptography processor		Public key, common key	Password, biometrics	Short	2 Mbps [7]	Practical
Network processor	Software	IDS, IPS	Sampling	Large		Medium
Server						

Table 1. HCgorilla vs. regular security techniques

http://dx.doi.org/10.5772/56145

Table 2 shows various aspects of ubiquitous media. They are classified into discrete and streaming media. Both types are expressed by byte structure. Discrete media is still useful in the ubiquitous environment. Interactive games use many algorithmic processes for discrete data. Streaming media is more important because most ubiquitous applications use streaming media. This is further divided into two types in view of its complexity. Text data is one type of streaming data, because it is useful as refrain information in the event of a disaster. Considering endless data is hard for mobile devices; the target of this work is discrete media and stream data. Yet, they need a sophisticated and complicated process. Since streaming media is massive, it is reasonable to protect the security by a common key, which is preferable to protect large quantities of byte-structured ubiquitous information.

		Discrete media	Streaming media	
			Stream data	Data stream
Definition		Individual data	A sequence of similar elements	A sequence of data, which may be different from each other
Characteristic		Discrete	Stream of continuous media	
Size or quantity		Short	Long	Endless
Complexity		Low	Medium	High
Examples		Games, intelligent processes	Text, audio, video	Seismography, tsunami, traffic
Basic structure		Byte string		
Buffer storage		Respectable reregister file		
Data handling	Media	Algorithmic process	SIMD mode applications like signal processing, graphic rendering, data compression, etc.	
	Security	Public key	Common key cryptography	

Table 2. Ubiquitous media

Although the performable features of various ubiquitous devices illustrated in Figure 1 have mainly relied on embedded software, such an approach inevitably exhausts much hardware resources and results in the deterioration of speed and power, which are worsening alongside the rapid increase in ubiquitous technologies in recent years. The majority of these technologies are resource constrained in terms of chip area and battery energy available. It is very difficult for the regular techniques of the massive quantity of multimedia information to satisfy the overall demands. Thus, a drastic improvement is required for the embedded system to achieve really promising ubiquitous devices. In this respect, a practical solution will be a security aware high-performance sophisticated single VLSI chip processor [8].

2.2. Challenge and goal

The practices of an ad hoc environment require resource-constrained security. To achieve mobility, various processor chips embedded in ubiquitous platforms are designed so that the occupied area and energy budget are as small as possible [9–11]. On the other hand, the temporal

formation of wireless and mobile ad hoc networks does not have the benefit of a permanent network infrastructure but relies on the connections themselves. Thus, a practical solution to achieve ad hoc security is single-chip VLSI processor built-in hardware cryptography. However, to the best of our knowledge, safety aware chips to protect multimedia data over ad hoc networks have never appeared. Thus, it is a challenging task to actualize not only the processing, but also the protection of multimedia data by unifying the role of PC processors, mobile processors, Java CPUs, cryptography processors and so on into a ubiquitous processor [5].

The goal of our study, described in this article, is the development of hardware cryptography, named the double cipher, to protect multimedia data over ad hoc networks and the implementation of the double cipher into a VLSI processor named HCgorilla. The hardware algorithm of the double cipher is based on the analysis of the internal behaviour of processors. The microarchitecture-level analysis is advantageous in achieving power-conscious multimedia data protection with high performance. Since power consumption and throughput are the basic metrics of processor specifications, careful attention is paid to them at each design step from the topmost architecture level to the transistor level. Actually, in recent years, the VLSI trend has exploited power conscious high performance not higher speed. Parallelism is really the global standard approach to the development of contemporary VLSI processors.

2.3. Application area

Figure 2 illustrates an application scenario of mobile phones which embed HCgorilla. Here, a standard image is multimedia data sent from the sender's mobile phone to the receiver's. Since the sender and receiver embed the same cipher chip, the entire encryption of the standard image is completely decrypted by the receiver. HCgorilla is able to carry out simultaneous processing of the encryption and decryption, taking into account bi-directional communication over networks. The common key is delivered ad hoc without relying on the network provider. Of course, public key infrastructure can be available for common key delivery. An electronic signature is also useful to certificate the message by using a security chip.

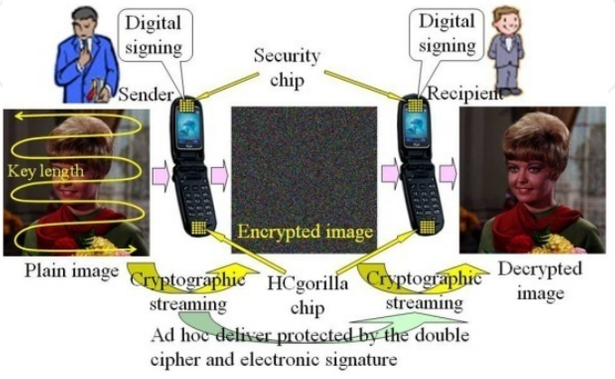


Figure 2. Application scenario

http://dx.doi.org/10.5772/56145

The double cipher is applicable to any multimedia data because

i. the double cipher falls into the category of block cipher as described in Chapter 3,

ii. multimedia data (image, audio, text) is byte structured as shown in Table 2.

Image data is expressed by PPM (portable PixMap), JPEG (joint photographic expert group), BMP (bit MaP) and so on. Figure 3 exemplifies the PPM image file of a standard image. This image has 256 lines and each line consists of 256 pixels. A PPM file consists of a header and pixel data. The header contains the PPM format ID, number of pixels in width and height and graduation. The pixel data contains all the graduation elements that are the 1-byte quantization of R-, G-, or B-elements. The 1-byte R-, G-, or B-graduation elements are the target of the double cipher process.

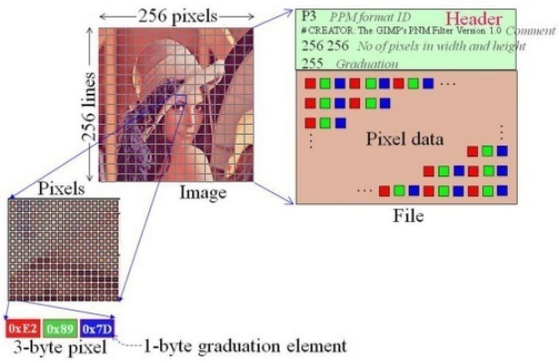


Figure 3. PPM image file

The double cipher can be applicable to cryptographic streaming. Here, the stream is a sequence of pixels, and the cryptographic streaming is the continuous encryption or decryption of a whole image and a moving picture. In view of the video display, the flame rate is 30 flames/sec. So, the resolution of the PPM format requires a bandwidth of

$$256 \times 256 \times 3 \times 30 \text{bytes} / \text{sec} \approx 6 \text{Mbytes} / \text{sec} \approx 50\ Mbps \quad (1)$$

On the other hand, the resolution of a QVGA (quarter video graphics array) format (320×240 pixels/flame) requires a bandwidth of

$$0.23 \times 30 \text{bytes} / \text{sec} \approx 55\ Mbps \quad (2)$$

Figure 4 shows the scanning modes of the image data. The continuous scan follows the exact sequence of the data format. The discontinuous scan accesses the data format in a predetermined order of discrete addresses. The mixed scan mode is also shown.

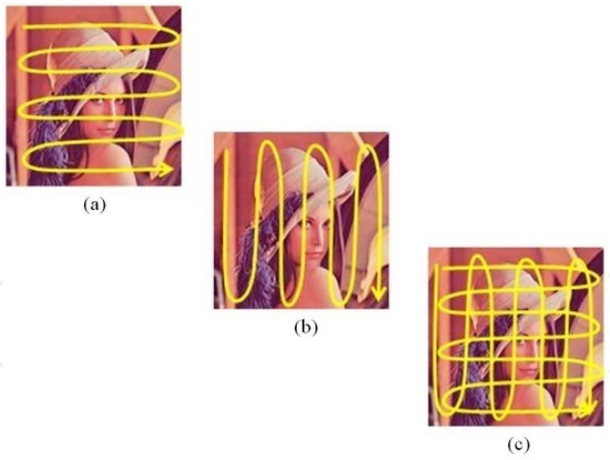


Figure 4. Scanning modes of an image (a) Continuous scan (b) Discontinuous scan (c) Mixed scan

Figure 5 illustrates the structure and encryption of audio data. This is also formed in bytes. In the case of WAV (waveform audio file) format, it consists of a header and waveform data derived by sampling and quantizing the analogue data. The quantization derives the byte form of a sampling bit at each sampling point.

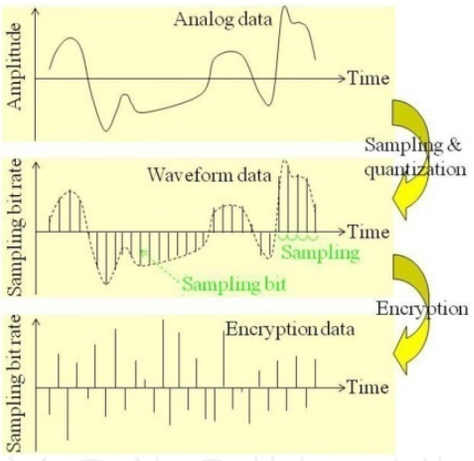


Figure 5. Audio data

3. Double cipher

The cipher strength does not merely depend on the algorithm of encryption itself, taking into account a round robin attack. It does not seek how to encrypt, but searches a key used in the encryption. Since the key is produced by an RNG, it is the essence of cipher strength. For example, the Vernam cipher lacks sealing ability, that is, the information of a plaintext is leaked by simply observing the ciphertext on the communication channel. Yet, it is assured that the Vernam cipher is ideally strong due to the use of a full length random number string.

http://dx.doi.org/10.5772/56145

In general, the cipher strength can be improved by increasing not only the key length but also all kinds of bit operations, which can be seen in the improvement from DES (data encryption standard) to AES (advanced encryption standard) [12]. This is based on the general rule of deciphering a secret key cryptography that seeks an unknown key or password, assuming that the plaintext, ciphertext and encryption algorithms are open [13]. According to this, the author proposes the double cipher scheme with two RNGs [4]. Consequently, this increases the key length. In practice, the double cipher approach promises strong cipher strength by providing double kinds of operations. Another advantage of this approach is power consciousness with negligible hardware cost and high throughput due to the microarchitecture-level hardware mechanism.

3.1. Proposed scheme

The hardware algorithm of the double cipher is proposed based on the analysis of the internal behaviour of the processors. The additional chip area and power dissipation required for this algorithm are negligibly small. The first scheme, RAC, is a transposition cipher devised from the direct connection of a built-in RNG (LFSR), register file (buffer of external data) and a data cache. A random store based on the direct connection scrambles or transposes a series of multimedia data at random without any special encryption operation. The second scheme is a data sealing algorithm implemented during the data transfer from the register file to the data cache. This complements the RAC's shortcoming and enhances the security of data information as a whole.

Figure 6 shows the basic algorithm of the double cipher in more detail. Here, $d_1d_2d_3d_4d_5$ exemplifies a plaintext block; d_i (i is an integer) is a 1-byte character, graduation element, and quantization of a sampling bit when the plaintext is formed into text style, image, and audio, respectively; *30241* is the corresponding key or the output of the first RNG, LFSR1; $h(d_2)h(d_5)h(d_3)h(d_1)h(d_4)$ is a ciphertext that is the result of the double encryption. In the execution of the RAC, the plaintext and LFSR1's output are synchronized according to their sequence. For example, the first data "d_1" and the first random number "*3*" are synchronized. During the storage in the third location of the data cache, a hidable function h works for the plaintext block. The sequence of a random addressing store like this results in the formation of a cipher in the data cache.

Double cipher encryption proceeds according to the following micro-operations.

i. Make the LFSR1 output integer specify a register file address.

ii. Synchronize a data cache address with the current clock count.

iii. Transfer the specified register file's content to the synchronized data cache address. During the transfer, a hidable function works for the plaintext block by using LFSR2 output.

The sequence of a random addressing store like this results in the formation of a cipher in the data cache. Double cipher decryption similarly proceeds. These micro-operations are practiced by a simple wired logic, which is effective in maintaining usability, speed and power consciousness.

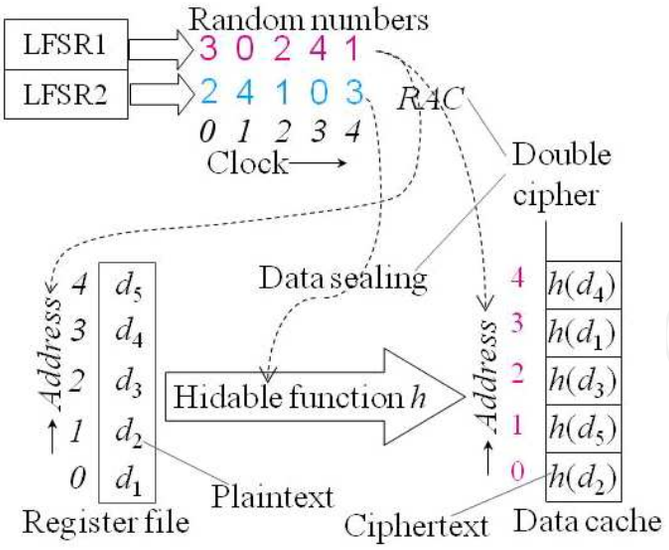


Figure 6. Double cipher

Since multimedia data is much longer than the plaintext shown in Figure 6, the adoption of block ciphers is inevitable in order to satisfy the demand for data quantity and performance. Figure 7 illustrates the relation among plaintext, blocks, dominant stages of the cipher pipeline (shortened to pipes hereafter) and the double cipher process. The relation between the cipher pipe and core is clear from Figure 8. The reason that both encryption and also decryption are shown in Figure 7 is that a single ubiquitous processor should cover bi-directional communication over networks as described in Section 2.3. A practical buffer for the external plaintext or ciphertext data is a register file whose space and speed are limited. So, the external data is divided into blocks and stored in a register file. The transfer of the block to the register file is assumed to be in DMA (direct memory access) mode, though it is not our concern in this study.

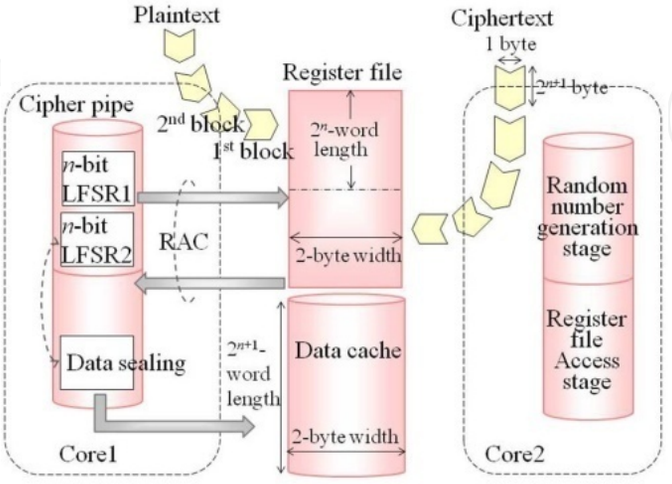


Figure 7. Double cipher mechanism within a single ubiquitous processor

http://dx.doi.org/10.5772/56145

Regarding the quantitative aspect of the double cipher process, the double cipher scheme regulates the block length to be the same as a buffer size. On the other hand, the block width is usually fixed to a byte because ubiquitous media takes the form of a byte-structured stream. The extensibility of the block structure is useful in high-speed processing. Another effect of the extension of the block length is to make the key length long, and thus the strength of the double cipher is expected to be strong in practice. The relation between the block, *n*-bit LFSR, register file and data cache is proved as follows.

$$\text{No.of blocks} = \text{plain or ciphertextsize} / \text{block size} \tag{3}$$

$$\text{Block size} = \text{logical space size} \tag{4}$$

$$\text{Block's word length} = \text{logical space length} \tag{5}$$

$$2^n = \text{register file's logical space size} = \text{data cache's logical space size} \tag{6}$$

The block transfer is subject to transposition and substitution ciphers. The interaction between the block, register file, data cache and LFSR is as follows. Core1 carries out RAC by making the LFSR1 output specify a register file address, synchronizing a data cache address with the current clock count, and transposing the specified register file's content to the synchronized data cache address. Then, LFSR2 makes the hidable function *h* on the data lines work for the substitution of the transferred data. The resultant content, stored in the data cache, is the encryption of the register file's content. The sequence of random addressing storage like this results in the formation of a cipher in the data cache. Double cipher decryption similarly proceeds within Core2.

3.2. VLSI implementation

According to the micro-operations shown in Figure 7, the architecture of the double cipher scheme-implemented VLSI processor is designed as shown in Figure 8. It is a ubiquitous processor named HCgorilla. The reason it is called a ubiquitous processor is due to its specific features for ubiquitous computing being power consciousness to achieve mobility, cost performance, simplicity, functionality, usability to actualize diversity and secureness to protect spreading platforms. HCgorilla is one of the most promising solutions for ubiquitous computing.

The correspondence of the double core, plaintext/ciphertext, register file and data cache is obvious in Figures 7 and 8. The double core contributes to cover both bi-directional communication and the recent trend of parallelism. Media pipes with sophisticated structures are newly added in Figure 8. This aims to cover media processing, indispensable for ubiquitous computing. Thus, HCgorilla unifies basic aspects of PC processors, mobile processors, media processors, cryptography processors and so forth and follows multicore and multiple pipelines.

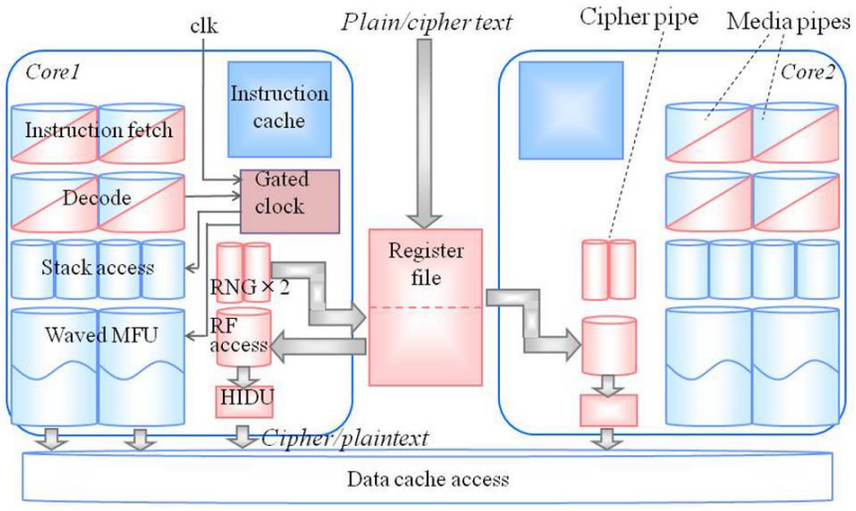


Figure 8. Architecture of HCgorilla

Each aspect specific to ubiquitous computing is achieved as follows. To begin with, secureness is achieved by the cipher pipe. The cipher pipe undertakes double encryption during the transfer from the register file to the data cache. While an LFSR controls the transposition cipher, RAC, another LFSR controls a substitution cipher or data sealing implemented by the hidable unit, HIDU (HIdable Data Unit). The double cipher executes the SIMD mode cipher and decipher codes. They do not attach operands, but repeat instances to transfer byte-structured data from a register file to a data cache. *rsw* encrypts the content stored in one half of the register file and *rlw* decrypts the content of the other half of the register file. These codes occupy the cipher pipe as long as the corresponding data stream continues. Thus, the SIMD mode sequence forms double cipher streaming.

The second aspect, power conscious resource-constrained implementation, is achieved by the design steps in the following.

i. Architecture level parallelism: HCgorilla exploits parallelism not higher speed in order to achieve power consciousness. Parallelism at the architecture level takes a multicore and multiple pipeline structure. Each core is composed of Java-compatible media pipes and cipher pipes. In addition, the register file and data cache are shared by the double core. Following the HW/SW co-design approach, two symmetric cores run multiple threads in parallel.

ii. Circuit module level: LFSR is used as the RNG built in the cipher pipe. LFSR falling into the category of M-sequence requires minimal additional chip area and power dissipation. A tiny n-bit LFSR produces the huge 2n-length random numbers. 1K-, 1M-, 1G-byte length texts require only 10-, 20-, and 30-bit LFSRs respectively.

http://dx.doi.org/10.5772/56145

iii. Instruction level parallelism (ILP): A wave-pipelined MFU (MultiFunctional Unit) is built in the execution stage of the media pipe to achieve effective ILP. This is the combination of wave-pipelining and multifunctionalization of the arithmetic logic functions for media processing. Since the latency of the waved MFU is constant, independent of the arithmetic logic operations, the media instructions are free from scheduling [14]. The wave-pipelining is effective also in achieving power conscious high speed.

iv. Microarchitecture level: Gated clocking is applied as described below.

Another aspect of HCgorilla, specific for ubiquitous computing, is its functionality and usability. Usability is an indispensable aspect of a multimedia mobile embedded system. Platform neutrality especially is very promising in providing multimedia entertainment such as music and games, the GPS (Global Positioning System) and so forth [15, 16]. In order to fulfil this feature, sophisticated language processing is required. In this respect, Java is expected to be useful. Thus, the media pipe shown in Figure 8 is a sort of an interpreter-type Java CPU [17]. The instruction set of HCgorilla is composed of 58 Java-compatible instructions together with two SIMD mode cipher instructions.

In Figure 8, the "Gated clock" block is a circuit module to control gated clocking [18]. Gated clocking is a cell-based approach for power saving at the microarchitecture level. It stops the clocking of such circuit blocks with low activity that waste switching power. Since leakage power is not a critical factor in the case of the 0.18-μm CMOS standard cell process used in this study, the gated clock is very effective for power saving. HCgorilla controls the clocking of the stack access and execution stages where switching probability is higher. In addition, the media pipe introduces scan logic for DFT (design for testability). This makes the pipeline register a shift register by serially connecting the FFs in order to read, write, and retrieve the pipeline stage status. The retrieval is useful for detecting and solving design errors. The scan logic is useful for the verification of the media pipe with a sophisticated structure. On the other hand, the scan logic is not applied to the cipher pipe because the cipher pipe is simpler and easier to verify. In addition, the scan logic and hardware cryptography are inconsistent with each other because the scan logic is apt to induce side-channel attack [19]. While traditional cryptographic protocols assume that only I/O signals are available to an attacker, every cryptographic circuit leaks information through other physical channels. An attack that takes advantage of these physical channels is called a side-channel attack. Side-channel attacks exploit easily accessible information, such as power consumption, running time, I/O behaviour under malfunctions and electromagnetic emissions.

Although Java is preferable as described above, in view of language processing, Java language systems are actually more complicated than regular language systems. The platform neutrality of Java applications is due to an intermediate form or class file produced by using Java compilers. This is really convenient for the JVM (Java virtual machine), but secondary for a processor itself. Since ubiquitous clients use small scale systems, the pre-processing of

complicated class files should be covered by large servers with Java applications. Even Java bytecodes have a problem, that is, the running of Java bytecodes is time-consuming due to the interpreting process. Although JVM or JIT (just-in-time compilation) built-in runtime systems are common for mobile devices, like mobile phones, they need more ROM (read only memory) space. This degrades the usability, cost and performance features of small ubiquitous devices.

In order to solve the issues described above, we have so far developed the software support system for the HCgorilla chips shown in Figure 9 [20]. The system is composed of a Java interface and parallelizing compilers. For example, the software support may run on proxy servers. Web delay, installing the software support on web servers, is one of the anticipative drawbacks of this approach. Obviously, it will take some time to transfer the executable code over the Internet. However, the transfer of class files to commercial processors also takes some time. In addition, the transfer time is not so important for the evaluation of web delays [21]. The main factor in web delays is the response time of the web servers. Another concern with this approach is maintaining security during the transfer. However, transferring the executable codes over the Internet does not generate a trust problem, because Java basically seeks the global standard of the Internet.

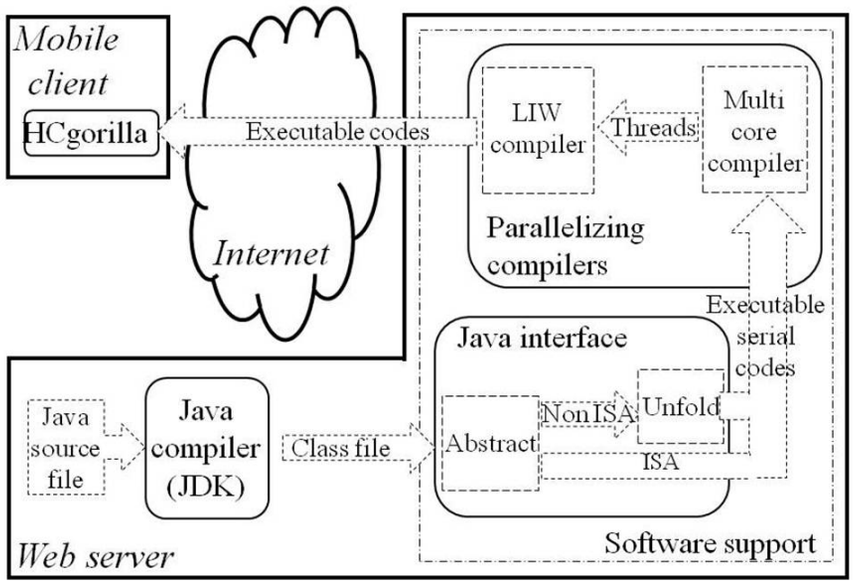


Figure 9. HCgorilla, web server, software support, and parallelizing compiler

HCgorilla, shown in Figure 8, is implemented in a 0.18-μm CMOS standard cell chip. The design environment is summarized in Table 3. Figure 10 shows the chip structure, die photo and floor planning of HCgorilla.7. This corresponds to Figure 8.

http://dx.doi.org/10.5772/56145

Software	
OS	Red Hat Linux 4/CentOS 5.4
Synthesis tool	Synopsys - Design Compiler D-2010.03
Simulation tool	Synopsys - VCS version Y-2006.06-SP1
Physical Implementation tool	Synopsys - IC Compiler C-2009.06
Verification tool	Mentor - Calibre v2010.02_13.12
Equivalent verification tool	Synopsys – Formality B-2008.09-SP5
Static Timing analysis tool	Synopsys – Primetime pts,vA-2007.12-SP3
Language	
Synthesis	VHDL
Simulation	Verilog-HDL
Technology	
ROHM 0.18-µm CMOS Kyoto univ. Standard Cell Library	

Table 3. Design environment of HCgorilla

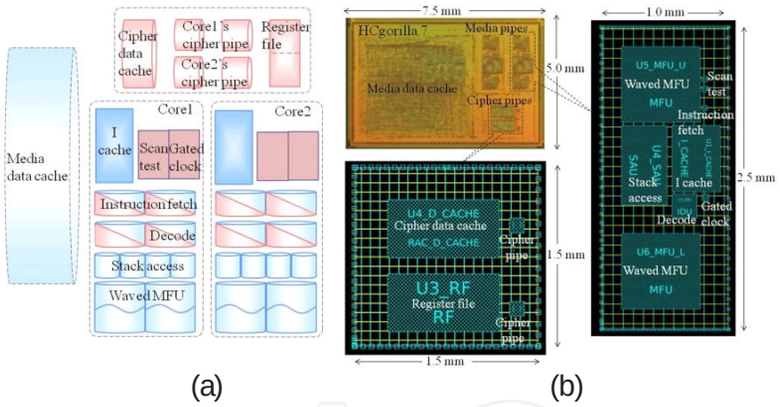


Figure 10. HCgorilla.7 (a) Structure (b) Die photo and floor planning

4. Evaluation and discussion

The overall evaluation of the HCgorilla chip ranging from the hardware cost, power dissipation, and throughput to cipher strength are described. Except for the hardware cost, quantitative measurement of the real chip and actual processor is difficult at this point. However, the simulation-based evaluation using the powerful CAD tools shown in Table 3 is reasonable enough. We prepared a DUV (design under verification) simulator and a test program run on the HCgorilla chip. Employing netlist, extracted from the chip layout, and analysing the algorithmic complexity are partly introduced. Table 4 summarizes the basic evaluation of

HCgorilla.7 compared with a previous derivative that we developed. These employ the same 0.18-μm CMOS standard cell technology. The overall aspects, chip parameters, and hardware specifications are also shown in this table.

			HCgorilla.6	HCgorilla.7
Design Rule			ROHM 0.18-μm CMOS	
Wiring			1 poly Si, 5 metal layers	
Area	Chip		2.5×5 mm	5.0 mm×7.5 mm
	Core			4.28 mm×6.94 mm
Assembly	Pad	Signal	158	
		VDD/VSS	32	
	Package		PGA257	
Power supply			1.8 V (I/O 3.3 V)	
Power consumption			275 mW	274 mW
Instruction cache			16 bits×64 words×2	
Data cache			16 bits×128 words×2	
Stack memory			16 bits×16 words×8	
Register file			16 bits×128 words	
RNG			6 bits×2	
No. of cores			2	
ILP degree			4	
Clock frequency			200 MHz	
Throughput	Media pipe		0.17 GIPS	
	Cipher pipe		0.1-0.2 GOPS	
Transfer rate			160-320 Mbps	

Table 4. Prospective specifications and potential aspects of HCgorilla chips

4.1. Hardware cost and power consumption

The hardware resource or cost is measured by the area occupied on the real chip, shown in Figure 10 (b). Figure 11 (a) shows the sharing of the occupied area. The portions denoted by "Stack access" and "Waved MFU" show the sum of four media pipes. The portion denoted by "Cipher pipes" shows the sum of four RNGs, register file and two HIDUs. The portion of the "D cache" is the sum of the media data cache and the cipher data cache. The media pipe, cipher pipe, and data cache employ 24,367, 270, and 20,625 cells, respectively. HCgorilla.7 and HCgorilla.6 have almost the same architecture. Yet, their chip areas are different. This is due to whether or not floor planning is undertaken. Since floor planning takes more area, it

http://dx.doi.org/10.5772/56145

contradicts low resource implementation. In addition, a clear layout often withstands side-channel attacks, tampering and so forth. Nevertheless, floor planning is indispensable for local and global clock separation, effective gated clocking and so on. Figure 11 (b) demonstrates the distribution of power dissipation derived from static evaluation, which summarizes the mean value of every cell. It does not take into account the switching condition. Figure 11 (c) shows the register file length dependency of power dissipation. The register file length swings backwards and forwards from HCgorilla.7's register file length, 128 words.

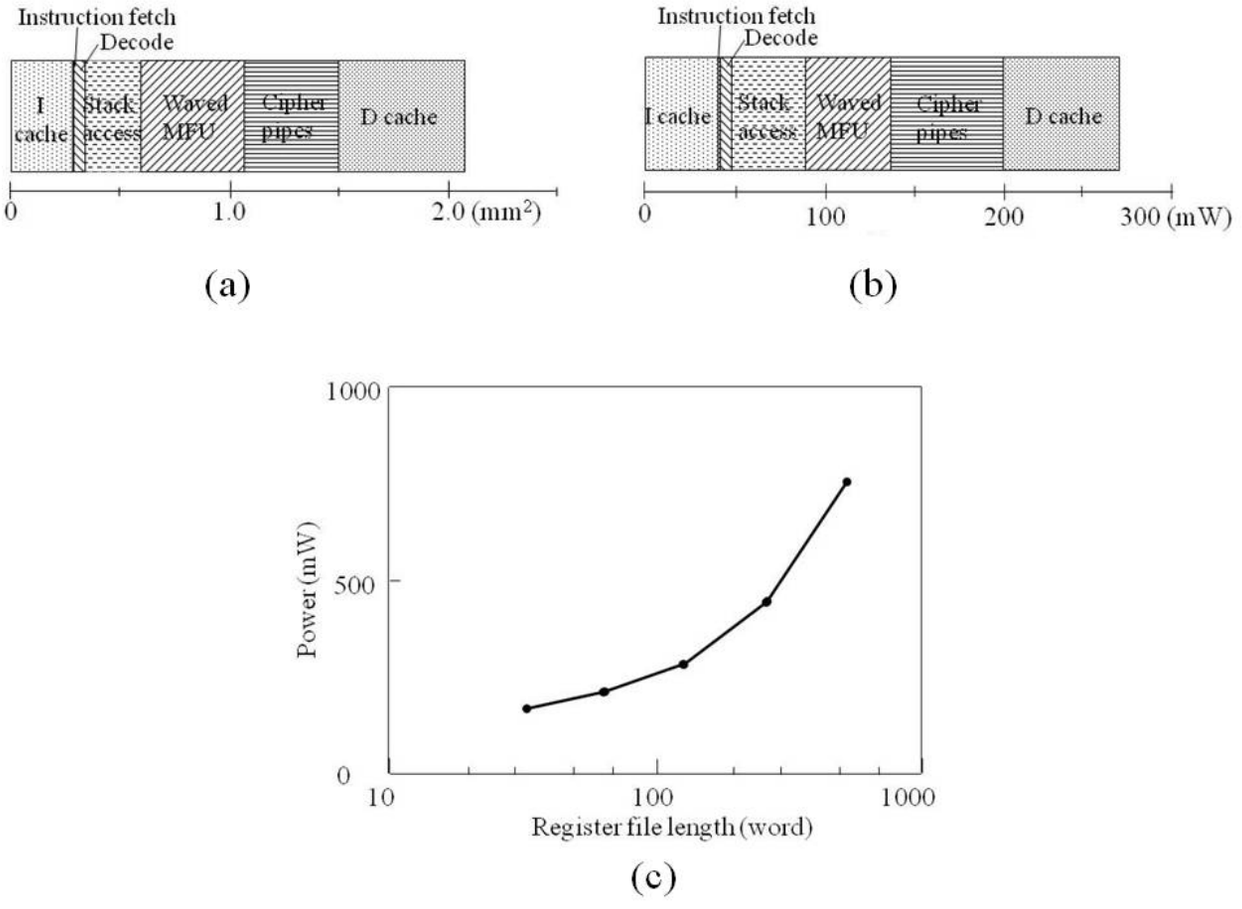


Figure 11. Overall evaluation of HCgorilla.7 (a) Occupied area (b) Power distribution (c) Power dissipation vs. register file length

4.2. Throughput

The cipher pipe's throughput, that is the mean value of the number of double cipher operations per unit of time, is derived from

$$\text{Throughput}\,[\text{OPS}] = \frac{\text{no.of double cipher operations}}{\text{running time}\,[\text{sec}]} \tag{7}$$

Since the running in Equation (7) is the repetition of the block transfer, rewriting the register file and double cipher operation, the running time is derived from

$$\text{Running time} = m(t_1 + t_2) + t_3 \tag{8}$$

Here, m is the number of blocks. The block width is usually fixed to a byte because ubiquitous media, like pixels, takes the form of a byte-structured stream. As a consequence, the register file width is evaluated by bytes. On the other hand, the register file length is measured expediently by word as shown in Figure 7. t_1 is block access time or the latency taken to transfer a block to the register file. t_2 is the time of an SIMD mode cipher operation. t_3 is the latency taken to transfer a block from the data cache. t_2 is evaluated by the DUV simulator that simulates a test program run on the HCgorilla chip. As for t_1 and t_3, let the memory access speed of mobile phones be 208 to 532 Mbytes/s and the mean value be adopted. Although such a method based on Equation (8) compromises the analysis, simulation and measurement, it is reliable considering that the cipher streaming of media data is undertaken regularly.

The cipher pipe's transfer rate, that is the mean value of the amount of transferred data per unit of time, is given by

$$\text{Transfer rate [bps]} = \frac{\text{full text size [b]}}{\text{transfer time [sec]}} \tag{9}$$

Identifying the transfer time in the denominator of Equation (9) with the running time in Equation (8), the following relation is derived.

$$\text{Transfer rate[Mbps]} = \text{throughput[GOPS]} \times \text{register file width[b]} \times 10^3 \tag{10}$$

Figure 12 shows the register file length dependency of HCgorilla.7's throughput in running a test program as shown in Figure 13. The register file length swings similarly to Figure 11 (c). The test program is composed of the double cipher and media processing. The plaintext used in the double cipher processing is 240×320-pixel QVGA format data. Then, the time of the SIMD mode cipher operation, t_2, is derived and the throughput in GOPS is derived from Equations (8) and (7). Similarly, the throughput in GIPS is derived from

$$\text{Throughput [IPS]} = \frac{\text{no.of instructions}}{\text{running time [sec]}} \tag{11}$$

The running time of the media processing is also derived by using the DUV simulator that simulates the test program. The media pipe's throughput is almost constant in Figure 12. This is because the clock speed is kept constant in varying the length of the register file.

In order to justify the instruction scheduling the free media pipe, the media processing of the test program is coded in three ways, that is, routines *A*, *B*, and *C*. These are distinguished in that the variable *k* and loop count are integers or floating point numbers. Routine *A* uses only

http://dx.doi.org/10.5772/56145

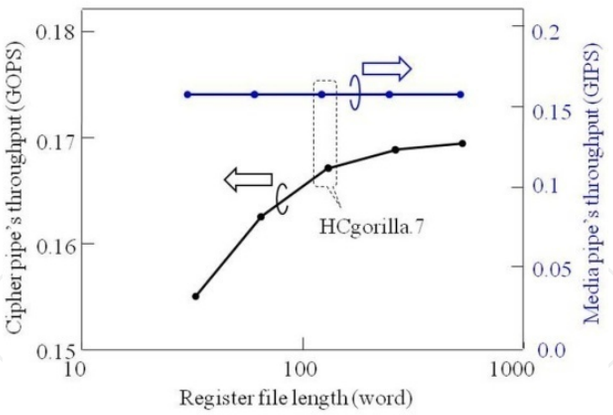


Figure 12. Throughput vs. register file length

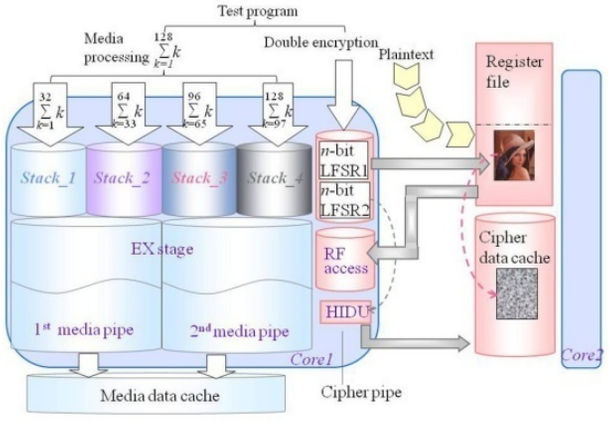


Figure 13. A test program and the internal behaviour of HCgorilla

integers and Routine *B* floating point numbers. Routine *C* uses both integer and floating point numbers. The hardware parallelism is utilized by dividing the summation into four threads and assigning them into four stacks in order to make full use of the two waved MFUs. The simulation result shows that the media pipe's throughput differs little between routines *A*, *B*, and *C*.

4.3. Cipher strength

Figure 14 shows how to measure the double cipher strength by experimenting with a rough-and-ready guess or round robin attack in a ubiquitous environment, where HCgorilla built-in platforms are used. The cipher strength is the degree of endurance against attack by a malicious third party. The attack is the third party's irregular action to decipher, break, or crack the cipher. This is clearly distinguished from decryption, that is, the right recipient's regular process of recovering the plaintext by using the given key.

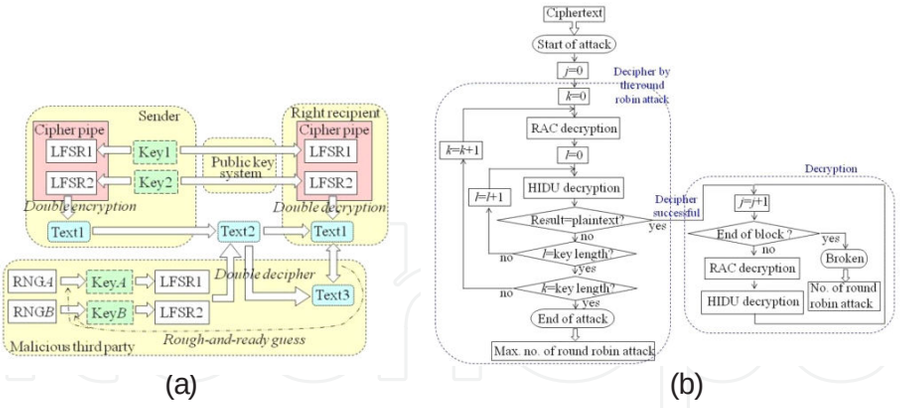


Figure 14. Measurement of double cipher strength (a) Round robin attack (b) Measurement flow

According to a normal scenario, the rules applied in deciphering the secret key cryptography in Figure 14 are as follows.

i. A plaintext, a ciphertext and the cipher algorithm are open to third parties.

ii. The key or the initial value of the RNG used in encryption is secret from third parties, though it is open to the right recipient.

A true key is sought out in deciphering. Sometimes it is called a password. The reason a plaintext and a ciphertext are open is because they are numerous and so, in turn, their quantity is beyond protection. In addition, it is reasonable that the cipher algorithm, or its specification, is open because its value is in its usability in the communication stages. This demands the spread of the algorithm in certain communities.

LFSR1 and LFSR2 in Figure 14 are RNGs for the double cipher built in the cipher pipes of a sender and a right recipient. They are also used by third parties according to rule (a). Key1 and Key2 are the initial values of LFSR1 and LFSR2 issued by the sender. Further encryption of these secret keys that are the target of attack is conventionally applied to maintain their confidentiality. For example, WEP cipher keys are encrypted by the RC4 cipher. In Figure 14, a public key system is available to exchange the key between a sender platform and a right recipient platform.

Text1 is a plaintext/ciphertext and Text2 is a ciphertext/plaintext derived by applying Key1 and Key2 to Text1. Key*A* and Key*B* are the guesses for Key1 and Key2 and are the initial values of the third party's LFSR1 and LFSR2. RNG*A* and RNG*B*, which are completely independent of LFSR1 and LFSR2, are used for rough-and-ready guesses or random guesses by the third party. Text3 is the guess of Text1 by the third party. When Text3 disagrees with Text1, one of RNG*A* and RNG*B* is forced to proceed to the next stage. If the disagreement continues by the end of the cycle of random number generation, the comparison of Text3 and Text1 is repeated by using the other RNG. Thus, the round robin attack against the double cipher undergoes nested loops.

From the discussion described above, the cipher strength is given by

http://dx.doi.org/10.5772/56145

$$\begin{aligned}\text{Cipher strength} &= \text{time for the round robin attack} = \\ &= \text{no.of round robin attacks} \times \text{clock cycle time} \le 2^{\text{LFSR1 size+LFSR2 size}} \times \text{clock cycle time}\end{aligned} \tag{12}$$

Since

$$\text{LFSR1 or LFSR2 size} \ge \log_2\left\{\frac{\text{register file length}}{2}\right\} \tag{13}$$

holds from Figure 13, the register file length is a critical factor of cipher strength. However, enlarging memory size surely causes an increase in power dissipation, the deterioration of clock speed, throughput and so forth. Thus, the demand of cipher strength is inevitably limited.

Figure 14 (b) shows the flow of measuring the number of round robin attacks in Equation (12). A result is achieved for every round robin attack. *j* is the number of blocks. The reason the measurement steps are distinguished in the cases of j=0 and j>1 is because the same random number sequence is issued for all the blocks from Figure 13. *k* is the number of RAC trial attacks. *l* is the number of HIDU trial attacks. Counting *k* and *l* through the experiment, the double cipher strength is derived from the number of nested loops or the time needed to decipher. This evaluates the degree of endurance or the strength. Actually, the cryptographic strength is the number of attack trials multiplied by the time for decryption. Each of the nested loops guesses a key at random, decrypts the ciphertext by using the key and judges if the decipher is successful.

Figure 15 shows the cipher strength achieved by practicing the method shown in Figure 14 and by using the 240×320-pixel QVGA format data, which is the same test data as is used by the test program shown in Figure 13. Note that the test data does not affect the cipher strength from Figure 13 and Figure 14 (b). It depends entirely on the block size or the half size of the register file because the blocks, after the success of the first attack, are simply decrypted by the known key. The abscissa is notched by the full length of the register file and the half size indicates a logical space. Although, from Table 3, HCgorilla.7's register file length is 128 words,

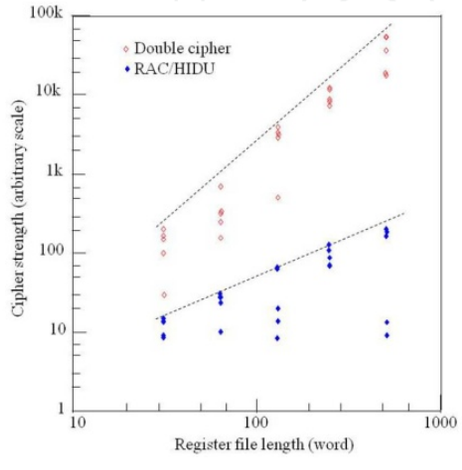


Figure 15. Cipher strength vs. register file length

to understand the dependency of the cipher strength, the register file length is varied from 32 to 512 words. Correspondingly, the size of LFSR is varied from 5 to 9 bits. The measurements are undertaken five times for each length. The dotted lines show the upper limit or the maximum number of the round robin attacks in the right hand side of Equation (11). The maximum strength of the double cipher is proportional to $2^{\text{LFSR1 size+LFSR2 size}}$ from Figure 14 (b). Similarly, the single cipher reaches $2^{\text{LFSR1 size}}$.

4.4. Discussion

Overall, the discussion is based on the evaluation described above. In view of the cipher strength, Figure 15 indicates that a longer buffer size is desirable. The double cipher increases the cipher strength as the key length or the cycle of random numbers expands. Although the hardware implementation of longer cycle random number generation is very easy, it surely involves a power consuming increase in the size of the stream buffer or register file. Considering that cipher algorithms are open to third parties in the evaluation of cipher strength, hardware specifications are more important than cipher strength in developing HCgorilla. While the power dissipation rapidly increases from the 128-word length in Figure 11 (c), the cipher pipe's throughput almost saturates at the 128-word length in Figure 12. The 128-word length is the optimum buffer size because (*i*) the power dissipation of mobile processors is usually less than 1 watt, and (*ii*) the cipher pipe's transfer rate shown in Table 4 is comparable to that of an ATM.

In view of cipher streaming, 0.1 GOPS is allowable for video format, because the running time used for cipher streaming occupies a very small portion of the video processing time. Actually, 1-Mbyte of text forms 4.3 flames of QVGA format. It takes 143 msec in video processing. The running time of cipher streaming is only 3.6% of 143-msec video processing. On the other hand, a 1-minute video takes 2.2-sec running time for cipher streaming, because, as shown in Equation (2), the text size is 414 Mbytes from the bandwidth. In the case of PPM format, 1-Mbyte of text forms 5 flames. This takes 167 msec in video processing, that is, only 3%. Then, as shown in Equation (1), a 1-minute video's text size is 360 Mbytes from the bandwidth. In this case, cipher streaming takes only 1.9 sec.

However, HCgorilla's throughput in GOPS is not always reasonable in view of CPU performance. The throughput of commercial mobile processors is more than 10 GOPS, though this has the benefit of the cutting-edge technologies of process, clock and hardware parallelism. In order to further enhance HCgorilla's GOPS value, which directly affects the increase in Mbps value, the running time should be decreased from Equation (7). This is possible with respect to the following strategies.

i. Reduce t_1 and t_3 by using a memory buffer with faster access speed.

ii. Reduce the summation of block access and transfer times, $\sum(t_1+t_3)$, by increasing the register file size. Expanding the register file length leads to an increase in cipher strength. However, it needs to take into account the trade-off between throughput and power dissipation. Judging from Equation (7), increasing the register file size

http://dx.doi.org/10.5772/56145

causes more power dissipation. In fact, from Figure 11 (c), the power dissipation rapidly increases from the 128-word length.

iii. Reduce t_2 by the increase in speed of the cipher pipe's clock. Increasing the number of pipeline stages is also useful for this aim.

Table 5 summarizes various aspects of the double cipher vs. usual common key ciphers. The double cipher, especially RAC, has the following characteristic aspects.

i. RAC simplifies the processing of multimedia data, because RAC directly handles a byte string whose structure is the same as that of the multimedia data as shown in Table 2. The effect of simplicity ranges over every aspect.

ii. RAC allows expandable block length. Different from usual common key ciphers, RAC does not fix the block length, but regulates it to be the same as the buffer size. Although the register file's logical length is 64 words at this point, we are planning to increase it.

iii. RAC handles wider blocks. The byte string is wider than the bit string used by other usual ciphers. At this point, the width is 2 bytes and is 16 times wider.

iv. RAC encrypts a plaintext block without any arithmetic logic operation. The cipher mechanism due to the random transfer between a register file and a data cache is quite different from other ciphers.

		Block		Cipher means		Through-put	Running time	Cipher strength	Resource
		String unit	Length	Key	Transformation				
Double cipher	RAC	Byte	As long as a buffer (register file) length		Needless	High	Medium	Practically strong	Small
	Data sealing	Bit			Bitwise XOR				
Vernam		Bit	Full length				Short	Strong	Large
Stream	LFSR		A few bits or a character				Medi-um	Medi-um	Small
	A5								
	AES-CTR		128 bits	1-2 times length of AES key	Bitwise XOR, scramble, shift, etc.		Long	Strong	Large
Block	AES								
	DES		64 bits						

Table 5. Double cipher vs. regular common key cryptography

These aspects allow double cipher higher throughput, shorter running time and practical strength. The throughput is given by the product of block data size and clock frequency, assuming the processing of one block per clock. Thus, higher clock frequency together with expandable data size provides higher throughput. Since the block data size is the product of expandable block length and width, it is allowed to increase with ease. The expandable block length allows the double cipher to have practical cipher strength. The extension of the block

length surely makes the key length long. Thus, the double cipher strength is expected to be strong in practice, because the cipher strength is closely related to the key length. The evaluation of running time is based on computational complexity and the dominant factor is the total number of iterative loops. AES has nesting loops for arithmetic, logic and functional operations. The first loop is for matrix operation and the second for rounds. However, RAC is released from such complexity.

5. Conclusion

The author has proposed a cipher scheme useful in practice for ad hoc networks with temporarily sufficient strength. The proposal is based on two cipher schemes. The first scheme is based on RAC and the second uses a data sealing algorithm. This double cipher scheme can be implemented in a security aware, power conscious and high-performance single VLSI chip processor by using built-in RNGs. Streaming the buffer size is determined from the trade-off between cipher strength, power dissipation and throughput. In practice, this is important because hardware specifications are more important than cipher strength in VLSI implementation.

HCgorilla is a sophisticated ubiquitous processor implementing the double cipher scheme. The VLSI implementation of HCgorilla is undertaken by using a 0.18-μm standard cell CMOS chip. The hardware cost, power dissipation, throughput and cipher strength of the latest HCgorilla chip are evaluated from real chip, logic synthesis and simulation by using CAD tools. Examining algorithmic complexity is partly used. The evaluation shows that HCgorilla is a power conscious, high-performance hardware approach that treats multimedia data with practical security over a ubiquitous network.

The future work of this research is the implementation of the double cipher into HCgorilla's media pipe. Although the cipher pipe and the media pipe are explicitly distinguished from each other in this study, the mixing of instruction scheduling free media processing with cipher processing at the microarchitecture level will further contribute power conscious security in ad hoc networks. Since such improvement applies the scan logic to encrypted data flow, an additional problem is raised, that is, whether the scan logic is able to avoid attacks by third parties. Apart from the disclosure of cipher algorithms, a design tolerant to side-channel attack and resistant to tamper is inevitable for VLSI implementation.

Author details

Masa-aki Fukase

Address all correspondence to: slfuka@eit.hirosaki-u.ac.jp

Graduate School of Science and Technology, Hirosaki University, Hirosaki, Japan

References

[1] Saha, D, & Mukherjee, A. Pervasive Computing: A Paradigm for the 21st Century. Computer Magazine (2003). , 36(3), 25-31.

[2] Wu, J, & Stojmenovic, I. Ad Hoc Networks. Computer Magazine (2004)., 37(2), 29–31.

[3] Satyanarayanan, M. Privacy: The Achilles Heel of Pervasive Computing? IEEE Pervasive Computing (2003). , 2(1), 2-3.

[4] Fukase, M, Uchiumi, H, Ishihara, T, Osumi, Y, & Sato, T. Cipher and Media Possibility of a Ubiquitous Processor: proceedings of International Symposium on Communications and Information Technologies, ISCIT (2009). September 2009, Incheon, Korea., 2009, 343-347.

[5] Fukase, M, & Sato, T. Double Cipher Implementation in a Ubiquitous Processor Chip. American Journal of Computer Architecture (2012). , 1(1), 6-11.

[6] Sato, T, Imaruoka, S, & Fukase, M. Hardware-Based IPS for Embedded Systems: proceedings of the 13th World Multi-Conference on Systemics, Cybernetics and Informatics, WMSCI 2009, IIII July (2009). Orlando, Florida., 74-79.

[7] Chikazawa, T, & Matui, M. Globalization of Japanese Cryptographic Technology. Journal of Digital Practice (2011). , 2(4), 267-273.

[8] Jerraya, A, Tenhunen, H, & Wolf, W. Multiprocessor Systems-on-Chips. Computer Magazine (2005). , 38(7), 36-40.

[9] Oppliger, R. Security and Privacy in an Online World. Computer Magazine (2011). , 44(9), 21-22.

[10] Stavrou, A, Voas, J, Karygiannis, T, & Quirolgico, S. Building Security into Off-the-shelf Smartphones. Computer Magazine (2012). , 45(2), 82-84.

[11] Burns, F, Bystrov, A, Koelmans, A, & Yakovlev, A. Security Evaluation of Balanced 1-of-n Circuits. IEEE Transactions on VLSI Systems (2011). , 19(11), 2135-2139.

[12] Wang, M-Y, Su, C-P, Horng, C-L, Wu, C, Huang, W, & Single- And, C. -T. Multi-core Configurable AES Architectures for Flexible Security. IEEE Transactions on VLSI Systems (2010). , 18(4), 541-552.

[13] Matsui, M. Survey of the Research and Development of MISTY Cryptography. Journal of Digital Practice (2011). , 2(4), 282-289.

[14] Fukase, M, & Sato, T. A Ubiquitous Processor Built-in a Waved Multifunctional Unit. ECTI-CIT Transactions (2010). , 4(1), 1-7.

[15] Lawton, G. Moving Java into Mobile Phones. Computer Magazine (2002). , 35(6), 17-20.

[16] Kochnev, D. S, & Terekhov, A. A. Surviving Java for Mobiles. IEEE Pervasive Computing (2003). , 2(2), 90-95.

[17] Chen, K-Y, Chang, J. M, & Hou, T-W. Multithreading in Java: Performance and Scalability on Multicore Systems. IEEE Transactions on Computers (2011). , 60(11), 1521-1534.

[18] Lee, Y, Jeong, D-K, & Kim, T. Comprehensive Analysis and Control of Design Parameters for Power Gated Circuits. IEEE Transactions on VLSI Systems (2011). , 19(3), 494-498.

[19] Alioto, M, Poli, M, & Rocchi, S. A General Power Model of Differential Power Analysis Attacks to Static Logic Circuits. IEEE Transactions on VLSI Systems (2010). , 18(5), 711-724.

[20] Fukase, M. A Ubiquitous Processor Embedded With Progressive Cipher Pipelines. International Journal of Multimedia Technology (2013)., 3(1), 31-37.

[21] Zari, M, Saiedian, H, & Naeem, M. Understanding and Reducing Web Delays. Computer Magazine (2001). , 34(12), 30-37.

IntechOpen