Performance Evaluation of an Emerging Stream Cipher Engine

Masa-aki Fukase* and Tomoaki Sato†
Hirosaki University, Hirosaki 036-8561 Aomori Japan
*E-mail: slfuka@eit.hirosaki-u.ac.jp Tel: +81-172-39-3630
†E-mail: tsato@cc.hirosaki-u.ac.jp Tel: +81-172-39-3723

***Abstract*— Considering the overall characteristics of ubiquitous environment, a practical solution of ubiquitous security is not to achieve ideally strong strength but to achieve temporary strength without relying on permanent network infrastructure. According to this concept, the authors have exploited a multimedia stream cipher engine. This is a safety aware, high-performed single chip processor. In order to keep security, usability, speed, and power consciousness, the stream cipher engine takes a compact multicore architecture. Each core implements a double cipher scheme that covers RAC (random addressing cryptography) and data sealing. The double cipher is microarchitecture-based, software-transparent hardware cryptography that offers the protection of the whole data with negligible hardware cost and moderate performance overhead. The double cipher increases cipher strength as the expansion of key length or the cycle of random numbers. Although the hardware implementation of longer cycle random number generation is very easy, it surely involves the power consuming increase of the size of a stream buffer or register file. Based on the previous implementation of the stream cipher engine chip by using 0.18-μm standard cell CMOS technology, this paper explores the tradeoff between cipher strength and buffer size. From the logic synthesis by using Synopsys Design Compiler, power dissipation, clock speed, running time, and throughput are studied. The cipher streaming buffer size dependency of these speculative factors achieves the guideline of optimum buffer size in view of cipher strength, power dissipation and throughput for ubiquitous computing.**

I. INTRODUCTION

One of crucial issues of ever growing ubiquitous network is worldwide diversity vs. security threat. This is the two-faced characteristic of ubiquitous network [1]. Another issue is massive quantity of multimedia information coming and going over ubiquitous network. Massive data is crucial for the interaction between ubiquitous devices and human being. However, it is very difficult to handle. Regular techniques used in ubiquitous devices lack rapidity, because embedded software is ever growing in spite of embedding.

In order to satisfy overall demands for not only security but also usability, speed, and power consciousness in ubiquitous environment, the authors' main concern has been not to explore an extremely strong cipher scheme, but to achieve practically enough strength without relying on permanent network infrastructure. According to this concept, the authors have exploited a sophisticated single VLSI chip processor for ubiquitous environment [2].

A cipher scheme developed for ubiquitous security is double cipher that covers RAC (random addressing cryptography) and data sealing [3]. The double cipher is microarchitecture-based, software-transparent hardware cryptography that offers the temporary protection of the whole data with negligible hardware cost and moderate performance overhead. Although the double cipher is not so strong in general, it increases cipher strength as the expansion of key length. The hardware implementation of longer cycle random number generation is very easy, but it involves the increase of stream buffer or register file size. Since enlarging memory size surely causes the increase of power dissipation and the deterioration of clock speed, the demand of cipher strength is inevitably limited [4].

Double cipher has been implemented in a stream cipher engine by using 0.18-μm standard cell CMOS technology [3, 5, 6]. The stream cipher engine has high potential for a safety aware, high-performed single chip ubiquitous processor. Based on the chip implementation of the stream cipher engine, this paper explores streaming buffer size dependency of specificative factors more in detail from the synthesis by using Synopsys Design Compiler. Those factors are cipher strength, power dissipation, clock speed, running time, and throughput. The tradeoff between cipher strength and power dissipation is made clear and the guideline of optimum buffer size is achieved in view of safety, power consumption, and performance for ubiquitous computing.

II. BASIC ORGANIZATION OF THE STREAM CIPHER ENGINE

Table 1 summarizes the progress of the stream cipher engine chips so far developed by using 0.18-μm CMOS standard cell process technologies. Design environments are basically Synopsys and Cadence tools. In view of architectural characteristics, the 1st version fixes the role of double core [5]. The one core is distinguished for encryption and the other core is dedicated for decryption. On the other hand, the 2nd version [6] and the 3rd version [3] do not fix the role of each core. In order to achieve the guideline of optimum buffer length in view of safety and performance for ubiquitous computing, the 4th version varies the length of

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

register file and data cache of the 3rd version stream cipher engine chip. The 4th version has the same microarchitecture as the 3rd version.

TABLE I
SPECIFICATIONS OF THE STREAM CIPHER ENGINE CHIPS

Development version			1st	2nd		3rd	4th
Architec-ture	Cipher		RAC			Double cipher	
	Register file		2byte × 64word			2byte × 512word	Parameterized
	Data cache		2byte × 64word			2byte × 512word	Parameterized
	Paralle-lism	Core	2				
		ILP/core	2			1	
Chip	Process		HITACHI 0.18μm CMOS			ROHM 0.18μm CMOS	
	Area	Die	2.8-mm square			2.5mm × 5.0mm	
		Core	1.38-mm square			1.62mm × 4.28mm	
	Voltage		1.8V (I/O 3.3V)				
	Power		73mW	137mW		536mW	
	Clock		400MHz	666MHz		200MHz	
	Throughput					0.1GOPS	
	Current status			Synthesis		Implemented	Synthesis

Fig. 1 shows the basic organization of the 3rd and 4th version of the stream cipher engine chip. This is composed of two symmetric cores in order to cover bidirectional communication and to achieve power conscious high-speed performance. Each core is a 5-stage pipeline composed of IF (Instruction Fetch), D (Decode), RNG (Random Number Generator)/RFA (Register File Access), DCA (Data Cache Access), and WB (Write Back). The core executes SIMD (Single Instruction Stream Multiple Data Stream) mode double cipher coeds. These are a random store code *rsw* and a random load code *rlw*. The execution of these codes is due to the direct connection of the output of a built-in RNG to the access line of data cache. The direct connection makes the transfer of data from register file to data cache at random, which results in transposition, permutation, or position rearrangement cipher.

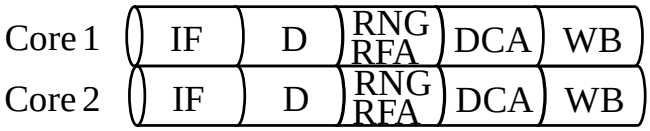

Fig. 1 Basic organization of the stream cipher engine chip.

Due to restricted hardware quantity, the built-in RNG is composed of LFSR (Linear Feedback Shift Register). LFSR falling into the category of M-sequence requires trivial additional chip area and power dissipation. A tiny *n*-bit LFSR produces the huge 2^n–length random numbers. A 1K-, 1M-, 1G-byte length texts require only 10-, 20-, and 30-bit LFSR respectively.

The one of cipher codes, *rsw* encrypts the content stored in the one half of the register file. The other code, *rlw* decrypts the content of the other half of the register file. Register file plays the role of streaming buffer. It buffers a block of external data that is plaintext or ciphertext. Register file and data cache are logically divided into two. Each of the double cipher coeds undergoes the two random processes of transposition and substitution. Double cipher encryption proceeds according to following microoperations.

(*i*) Make RNG's 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.

The resultant content stored in the data cache is the encryption of the register file's content. The sequence of random addressing store like this results in the formation of a cipher in data cache. Such a microarchitecture-based, software-transparent mechanism offers the protection of the whole data with negligible hardware cost and moderate performance overhead. Double cipher decryption similarly proceeds.

Fig. 2 illustrates the data transfer mechanism observed in the stream cipher engine. Especially, this focuses on the interaction between block, register file, data cache, and LFSR. Although regular block ciphers like AES (Advanced Encryption Standard) and DES (Data Encryption Standard) divide plain and cipher texts into blocks, the double cipher does not always divide plain/ciphertext, because it can treat a full text if an ideal buffer is available to immediately store a full text. However, practical buffer built in the stream cipher engine is a register file whose space and speed are limited. So, plain/ciphertext is practically divided into blocks and stored in register file.

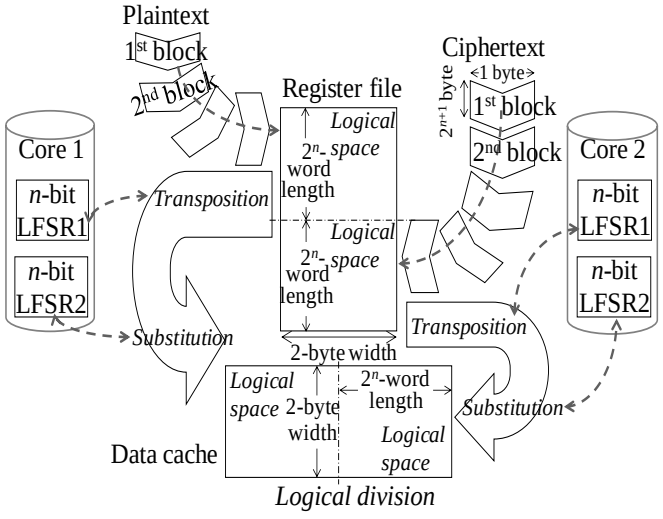

Fig. 2 Data transfer mechanism.

Block transfer between register file and data cache follows multiple data stream. Also, it undergoes transposition and substitution ciphers. The length and width of block and register file are different in general, because block size that is the product of word length and width in byte is flexible depending on communication and cipher systems. On the other hand, register file size is fixed. Practically, register file stores one block. Register file and data cache have the same size. Since they are composed of 2-byte words, one pixel occupies 1.5 words. Thus, the relation between the block, *n*-bit LFSR, register file, and data cache is approved as follows.

Block size = logical space size (1)

Block's word length = logical space length (2)

2^n = register file's logical space size
= data cache's logical space size. (3)

III. SPECIFICATIVE FACTORS VS. CIPHER STREAMING BUFFER SIZE

Streaming buffer size is crucial for various performance factors. Thus, their evaluation targets to register file size and data cache size. Especially, further expanding the register file and data cache is explored in order to achieve practical cipher strength.

A. *Streaming Data*

The streaming data used in this study is a standard image shown in Fig. 3. Since 1 color is expressed by 1 byte and 3 colors are used for 1 pixel, a 0.5-kbyte logical space of the 3rd version stream cipher engine buffers $2^9/3$ pixels. This is less than 1 line of the standard image. On the other hand, the whole of the standard image needs 0.2 Mbytes to buffer.

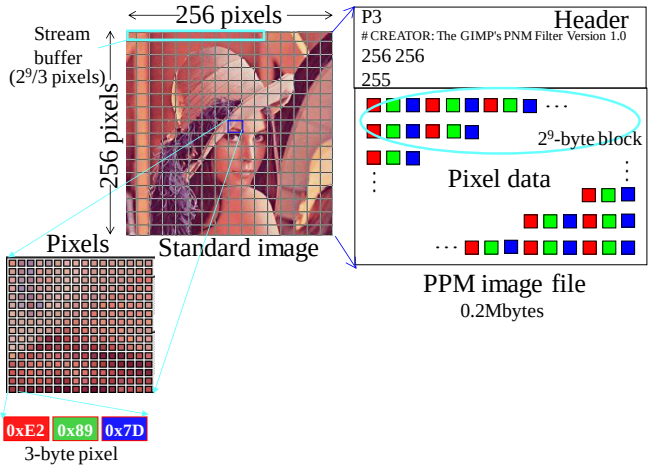

Fig. 3 Data format.

In view of video display, the flame rate is 30 flame/s. Then, the resolution of PPM (Portable Pix Map) format (256×256pixel/flame) requires the bandwidth of

0.2×30 Mbyte/s = 50 Mbps. (4)

On the other hand, the resolution of QVGA (Quarter Video Graphics Array) format (320×240pixel/flame) requires the bandwidth of

0.23×8×30 = 55 Mbps. (5)

B. *Cipher Strength*

Fig. 4 shows the round robin attack of the stream cipher engine. *j* is the number of block. *k* is the number of trial attacks. The round robin attack does not search operation but does key. It repeats trial attack that produces a key at random, deciphers, breaks, or cracks ciphertext by using the key. The strength or the degree of enduringness is reasonably measured by the time needed in the attack and the discovery of a true key.

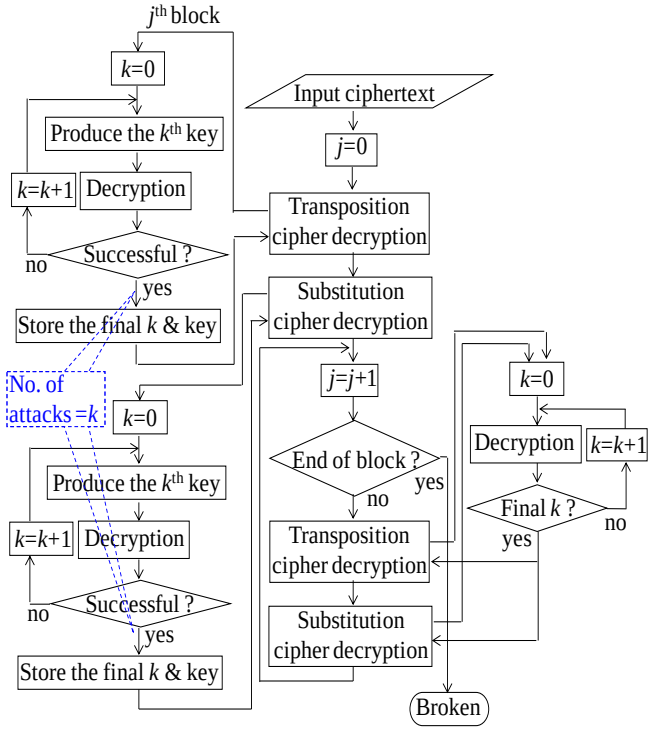

Fig. 4 Cipher strength measurement.

Thus, the cryptographic strength is the product of the time for decryption and the number of attack trials. The number of trials is measured in case of round robin attack. The maximum number is the number of random numbers or LFSR output size given by $2^{\text{key length}}$. Then, the strength of the double cipher for a round robin attack is given as follows.

Cipher strength
= max. time of the round robin attack
= no. of the round robin attack
× time for decryption
= $2^n \times 2^{\text{buffer width}}$ × time for decryption. (6)

Although the length and width are both crucial for cipher strength from (6), the width is usually fixed to byte width because ubiquitous media takes the form of byte structured stream. Thus, the buffer length is a critical factor. Fig. 5 shows register file length dependency of the cipher strength of the 4th version stream cipher engine.

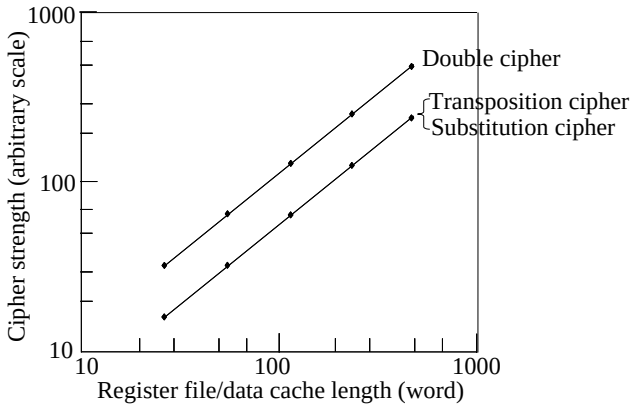

Fig. 5 Cipher strength vs. register file length.

C. Power Dissipation

Fig. 6 shows power dissipation vs. register file length of the 4th version stream cipher engine chip. Power consumption is a rough approximation or the maximum power derived from the summation of mean value of every gate. It does not take into account of switching condition. Thus, occupied area is also shown in Fig. 6.

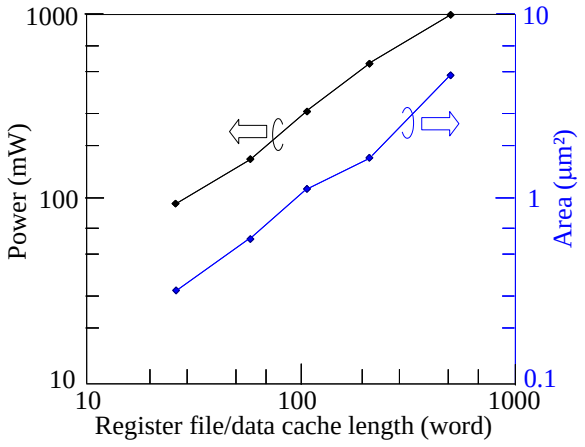

Fig. 6 Power vs. register file length.

D. Clock Speed

Fig. 7 shows clock speed vs. register file length of the 4th version stream cipher engine chip. Clock frequency is derived from the timing analysis of the netlist. The maximum critical path is an address line between RNG and register file. Although clock speed decreases as the increase of buffer size, this does not deteriorate running time and throughput as shown in Fig. 9.

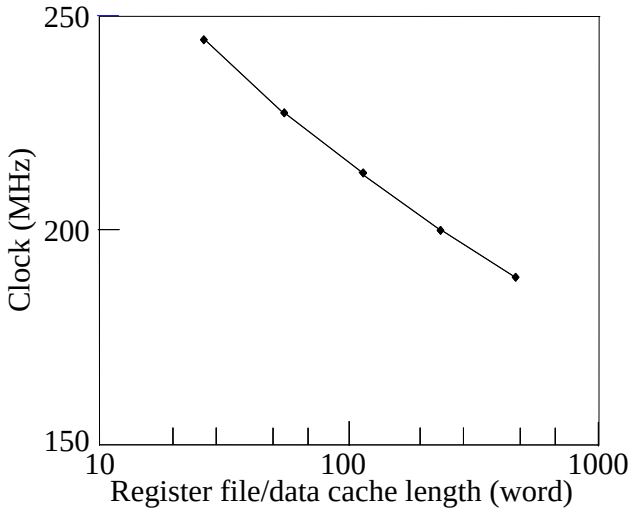

Fig. 7 Clock speed vs. register file length.

E. Running Time and Throughput

During the cipher streaming, three factors take place as shown in Fig. 8. These are block access or storing a block into register file, SIMD mode double cipher process, and drawing the block from data cache to external. Thus, the running time is given as follows.

Running time = $m(t_1+t_2)+t_3$ (7)

m = no. of blocks
= full text's size/block size (8)

t_1 = block access time
= the latency taken to transfer a block to the register file
= block size/access speed (9)

t_2 = time of a SIMD mode cipher operation
= {instruction pipeline degree
+ (block's word length - 1)}
× clock cycle time (10)

t_3 = latency taken to transfer a block from data cache
= t_1. (11)

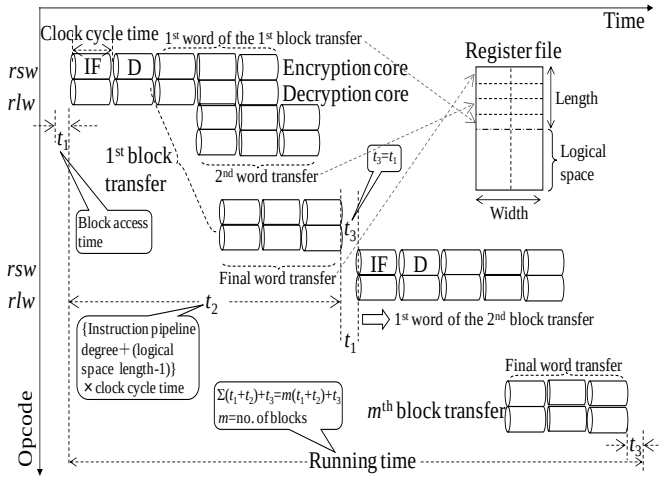

Fig. 8 Derivation of running time and throughput.

Since the double cipher process is SIMD mode, it does not fetch a cipher code per clock, but automatically repeats the process. By using the running time, throughput is given as follows in OPS (Operations Per Second).

$$\text{Throughput} = \text{no. of word transfers} / \text{running time}. \quad (12)$$

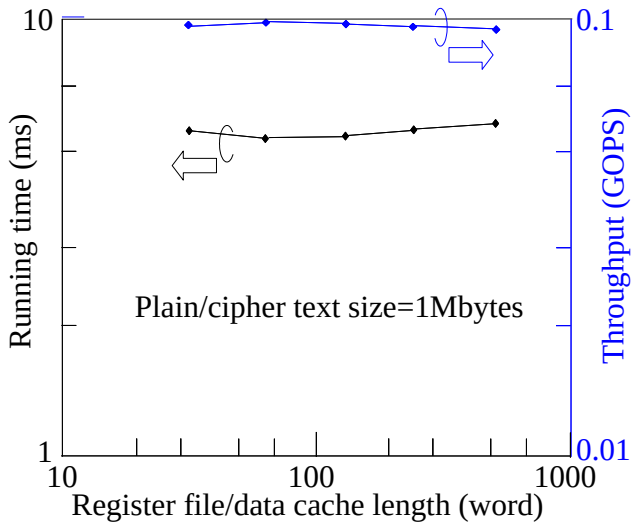

Fig. 9 Running time, throughput vs. register file length.

Fig. 9 shows the register file length dependency of running time and throughput. The value of t_1 and t_3 is approved from the mean value of the memory access speed of cellar phones. Let us assume it varies between 208 and 532 Mbytes/s. Although m is integer, the right hand side of (8) is a real number in general. However, this discrepancy is negligible in case of m>1000, which sufficiently holds for Fig. 9. The number of word transfers shown in the right hand side of (12) is text size because block width is 1 byte as shown in Fig. 2.

As is clear from Fig. 9, running time is almost independent on register file length. Similarly, the register file dependency of throughput is weak. This is due to two reasons. The one is the decrease of m, the number of register file accesses as the increase of register file length. The other is, as shown in Fig. 7, the deterioration of clock speed as the expansion of register file. Another explanation is that the register file size is included in the denominator and numerator of (7)-(12).

F. Discussion

In view of cipher strength, longer buffer size is desirable from Fig. 5. However, this consumes large power. Since the power dissipation of mobile processors is usually less than 1 watt, the register file size should be at most 512 words from Fig. 6. Register file length should be limited in view of power dissipation. This validates the 512-word register file and data cache of the 3rd version stream cipher engine.

512-word register file achieves sufficient throughput for video. The 0.1-GOPS (Giga OPS) cipher streaming shown in Fig. 9 is allowable for video format, because running time used for cipher streaming occupies very small portion of video processing time. Actually, 1-Mbyte text forms 4.3 flames of QVGA format. It takes 143 ms in video processing. Only 3.6 % of 143-ms video processing is taken for cipher streaming. On the other hand, 1-minute video takes 2.2-sec running time for cipher streaming, because text size is 414 Mbytes from the bandwidth shown in (5). In case of PPM format, 1-Mbyte text forms 5 flames. This takes 167 ms in video processing that is only 3 %. Then, 1-minute video's text size is 360 Mbytes from the bandwidth shown in (4). In this case, cipher streaming takes only 1.9 sec.

IV. Conclusion

In order to achieve safety and performance for ubiquitous computing, this paper has explored the stream cipher engine's tradeoff between cipher strength and buffer size. From the logic synthesis by using Synopsys Design Compiler, power dissipation, clock speed, running time, and throughput have been studied. From the streaming buffer length dependency of specificative factors, especially in view of cipher strength, power dissipation and throughput, the guideline of optimum buffer length is at most 512 words.

The next steps of this study are as follows.

(*i*) Power and timing evaluation more in detail by using place and route tools after netlists and delay information.

(*ii*) Experimental measurement.

(*iii*) Design improvement by introducing clock multiplication.

Acknowledgment

This work is supported in part by Grant-in-Aid for Scientific Research (C) (19500036) from Ministry of Education, Culture, Sports, Science and Technology, Japan.

REFERENCES

[1] M. Satyanarayanan, "Privacy: The Achilles Heel of Pervasive Computing?" IEEE pervasive, Vol. 2, No. 1, pp. 2-3, Jan.-Mar. 2003.

[2] A. Jerraya, H. Tenhunen, and W. Wolf, "Multiprocessor Systems-on-Chips," Computer Magazine, Vol. 38, No. 7, 2005.

[3] M. Fukase, K. Noda, and T. Sato, "Emerging Hardware Cryptography and VLSI Implementation," Proc. of ISPACS 2008, pp. 445-448, Feb. 2009.

[4] M. Fukase, Y. Ohsumi, and T. Sato, "Exploring the Optimum Buffer Size of an Emerging Stream Cipher Engine," Proc. of ECTI-CON 2009, pp. 607-610, May 2009.

[5] M. Fukase, H. Takeda, R. Tenma, K. Noda, Y. Sato, R. Sato, and T. Sato, "Development of a Multimedia Stream Cipher Engine," Proc. of ISPACS 2006, pp. 562-565, Dec. 2006.

[6] M. Fukase and T. Sato, "A Stream Cipher Engine for Ad-hoc Security," Proc. of CIS'2007, pp. 902-906, Dec. 2007.