COMPARISON OF THE HARDWARE ARCHITECTURES AND FPGA IMPLEMENTATIONS OF STREAM CIPHERS †

M. D. Galanis[1], P. Kitsos[2], G. Kostopoulos[3], N. Sklavos[4], O. Koufopavlou[5], and C.E. Goutis[6]

VLSI Design Laboratory, Electrical and Computer Engineering Department,
University of Patras, Patras, Greece
{mgalanis[1], pkitsos[2], gkostop[3], nsklavos[4], odysseas[5], goutis[6]}@ee.upatras.gr

ABSTRACT

In this paper, the hardware implementations of five representative stream ciphers are compared in terms of performance and consumed area. The ciphers used for the comparison are the A5/1, W7, E0, RC4 and Helix. The first three ones have been used for the security part of well-known standards. The Helix cipher is a recently introduced fast, word oriented, stream cipher. W7 algorithm has been proposed as a more trustworthy solution, due to the security problems that occurred concerning A5/1 strength. The designs were coded using VHDL language. For the hardware implementation of the designs, an FPGA device was used. The implementation results illustrate the hardware performance of each cipher in terms of throughput-to-area ratio. This ratio equals to: 5.88 for the A5/1, 1.26 for W7, 0.21 for the E0, 2.45 for the Helix and 0.86 for the RC4.

1. INTRODUCTION

Secret key cryptographic systems can be categorized into either block or stream ciphers. Block ciphers are memoryless algorithms that permute *N*-bit blocks of plaintext data under the influence of the secret key and generate *N*-bit blocks of encrypted data. Stream ciphers contain internal states and typically operate serially by generating a stream of pseudo-random key bits, the keystream (stream ciphers are also called *keystream generators*). The keystream is then bitwise XORed with the data to encrypt/decrypt. Stream ciphers do not suffer from the error propagation, as in the block ones, because each bit is independently encrypted/decrypted from any other. They are generally much faster than block ciphers and they have greater software efficiency. Due to these features stream ciphers have been the choice for several communication standards, like IEEE 802.11b [1] and Bluetooth [2].

There are numerous stream cipher algorithms. Five of them have been chosen, implemented and compared in this paper. A5/1, E0 and RC4 are well-known stream ciphers since they have been specified in popular standards and protocols - the A5/1 in GSM [3], the E0 in Bluetooth [2], and the RC4 in IEEE 802.11b [1] and in other ones (e.g in Oracle secure SQL). Helix [4] is a word-oriented stream cipher, which also provides Message Authentication Code (MAC) function. Its functions are easily implemented and it is faster (in software implementations) than the best Advanced Encryption Standard (AES) implementation [4]. The W7 algorithm is a synchronous stream-cipher optimized for efficient hardware implementation at very high data rates [5]. W7 has been proposed in order to replace A5/1 in GSM security scheme, due to the security problems of the A5/1 [6].

The rest of the paper is organized as follows. Section 2 describes the A5/1 cipher, while section 3 the W7. Sections 4 and 5 describe the E0 and the Helix ciphers, respectively. RC4 cipher is presented in section 6. The hardware designs of the ciphers and the implementation results are presented and analyzed in section 7. Finally, section 8 draws the conclusions for the stream cipher comparison.

2. A5/1 CIPHER

A5/1 is a stream cipher used for encrypting over the air transmissions in the GSM standard [3]. A GSM conversation is transmitted as a sequence of 228-bit frames (114 bits in each direction) every 4.6 millisecond. Each frame is XORed with a 228-bit sequence produced by the A5/1 keystream generator. The initial state of this generator depends on a 64-bit secret key, K_c, which is fixed during the conversation, and on a 22-bit public frame number, F_n.

The A5/1 cipher is composed of three LFSRs, R_1, R_2, and R_3 of lengths 19, 22, and 23 bits, respectively. Each LFSR is shifted, using clock cycles that are determined by a majority function. The majority function uses three bits C_1, C_2, and C_3. Among the bits C_1, C_2, C_3 if two or more of them are 0 then the majority $m = 0$. Similarly if two or more of them are 1 then the majority $m = 1$. If $C_k = m$ then R_k is shifted, where k=1,2,3. The feedback polynomials for R_1, R_2, R_3 are: $x^{19} + x^5 + x^2 + x + 1$, $x^{22} + x + 1$ and $x^{23} + x^{15} + x^2 + x + 1$, respectively. At each cycle, after the initialization phase, the last bits of each LFSR are XORed to produce one output bit. This process is summarized in Figure 1.

The process of generating the keystream bits from the key K_c and the frame number F_n is performed in four steps. In *step 1*, all the LFSRs are initialized to zero. Then the bits of K_c, starting from the least significant bit, are shifted into the three LFSRs in parallel, ignoring the majority function. During each cycle the bit from K_c is fed in and XORed with bit 0 of each LFSR.

† This work was partially funded by the Alexander S. Onassis Public Benefit Foundation.

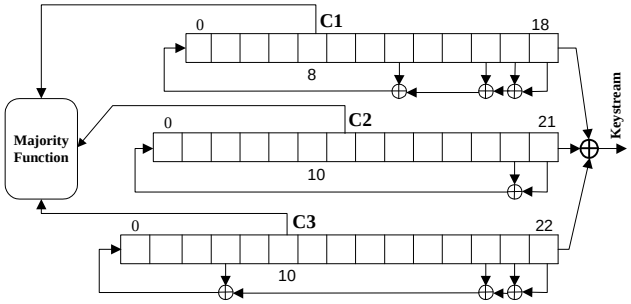

Figure 1: A5/1 stream cipher proposed architecture

In *step 2*, the 22 bits of F_n are fed in using the same process as in step 1. In *step 3*, 100 additional cycles are performed using the majority function, but without any output. Finally in *step 4*, another 228 cycles are performed to get the 228 pseudo-random keystream bits.

3. W7 CIPHER

The W7 algorithm is a symmetric key stream-cipher that supports key lengths of 128-bit. W7 cipher contains eight similar models, C1, C2,…,C8. Each model consists of three LFSR's and one majority function.

W7 architecture consists of a control and a function unit. The function unit is responsible for the keystream generation. This unit contains eight similar cells. The proposed architecture for the hardware implementation of one cell is presented in Figure 2. Each cell has two inputs and one output. The one input is the key and it's the same for all the cells. The other input consists of control signals. Finally the output is one bit long. The outputs of each cell complete the keystream byte.

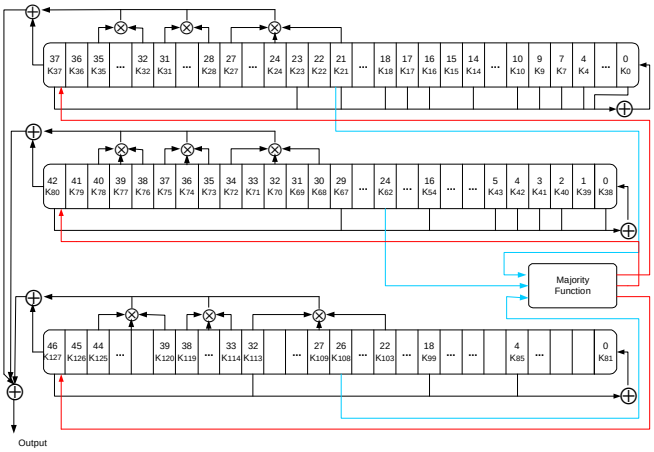

Figure 2: Proposed architecture for W7 cell two

Each cell consists of three LFSRs 38-, 43- and 47-bit long and a majority function. Their initial state which is the same for all is the symmetric encryption key. The 128-bit of the key map to the LFSRs initial state as:

LFSRa (38-bit): $LFSR_0 = K_0, LFSR_1 = K_1, \ldots, LFSR_{36} = K_{36}, LFSR_{37} = K_{37}$

LFSRb (43-bit): $LFSR_0 = K_{38}, LFSR_1 = K_{39}, \ldots, LFSR_{41} = K_{79}, LFSR_{42} = K_{80}$

LFSRc (47-bit): $LFSR_0 = K_{81}, LFSR_1 = K_{82}, \ldots, LFSR_{45} = K_{126}, LFSR_{46} = K_{127}$

The three LFSRs together determine when each shift register is clocked. One bit in each register is designated as the clock tap for that register as it's shown in Fig. 2. At each clock cycle the majority value for these taps determines which LFSRs advance. Only the LFSRs whose clock taps agree with the majority advance. The output bit arises after a non-linear function in the register which is a combination of several bits in the LFSR as it's shown in figure. The non-linear function is a combination of logical-AND functions. The actual keystream output is taken as the exclusive-OR of the three LFSRs. The keystream byte is the aggregation of each cell output.

4. E0 CIPHER

The encryption of packet payloads in Bluetooth is performed by the E0 stream cipher [2], which consists of three components, as illustrated in Figure 3. The first one is the *payload key generator* component, which performs the initialization (payload key generation). The second one, the *keystream generato*r, generates the keystream bits, and uses for this purpose four LFSRs, whose output is the input of a 16-state finite-state machine (called the *summation combiner*). The state machine output is the keystream sequence or the randomized initial start value during the initialization phase. The lengths L_i of the four LFSRs are 25, 31, 33, 39 and their feedback polynomials $f_i(x)$ are: $x^{25} + x^{20} + x^{12} + x^8 + 1$, $x^{31} + x^{24} + x^{16} + x^{12} + 1$, $x^{33} + x^{28} + x^{24} + x^4 + 1$, $x^{39} + x^{36} + x^{28} + x^4 + 1$, respectively, with i=1, 2, 3, 4.

For the LSFRs initialization, the keystream generator needs to be loaded with an initial value for the four LFSRs (in total 128 bits) and with 4 bits that specify the values of registers in the summation combiner. The 132-bit initial value is derived from four inputs by using the keystream generator itself. The input parameters are the encryption key K_c, a 128-bit random number, a 48-bit Bluetooth address, and the 26 master clock bits. Within the payload key generator, the K_c is modified into another key denoted K_c', by using a polynomial modulo operation described in [2]. The maximum effective size of this key is factory preset and may be set to any multiple of eight between one and sixteen (8-128 bits).

When the encryption key has been created, all the bits are shifted into the LFSRs, starting with the least significant bit. Then, 200 stream cipher bits are created by operating the generator. Of these bits, the last 128 ones are fed back into the keystream generator as an initial value of the four LFSRs. The values of the state machine are preserved. From this point on (after 239 cipher bits), when clocked the generator produces the encryption (decryption) sequence

which is bitwise XORed to the transmitted (received) payload data, in the *third component* of the cipher.

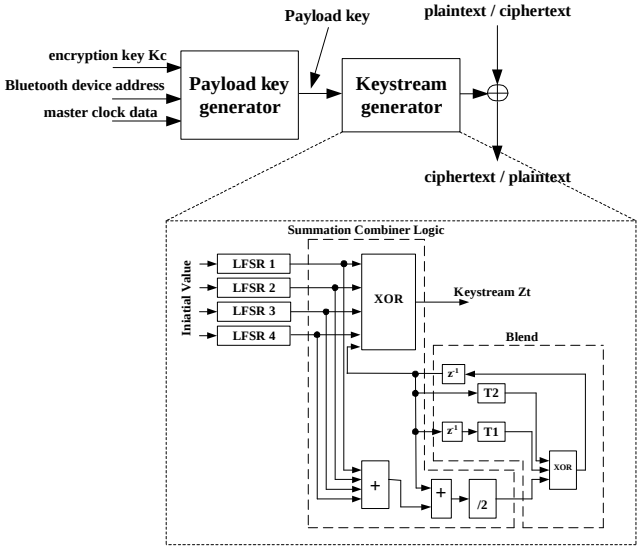

Figure 3: The E0 stream cipher

5. HELIX CIPHER

Helix is a combined stream cipher and MAC function that directly provides the authenticated encryption functionality [4]. By incorporating the plaintext into the stream cipher state, Helix can provide the authentication functionality without extra costs.

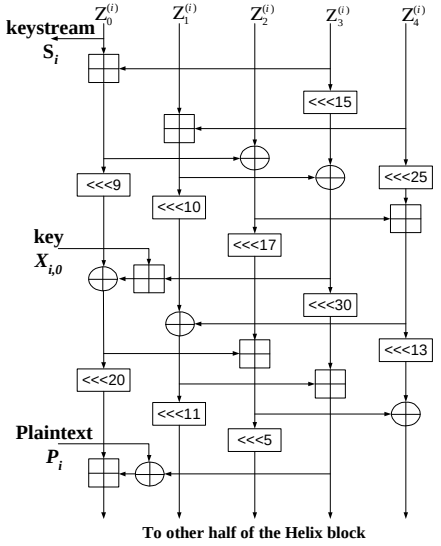

Figure 4: The half of one block of the Helix cipher

Helix uses a 256-bit key and a 128-bit parameter (called *nonce*). The key is secret, and the nonce is typically public knowledge. All operations in Helix are on 32-bit words. These operations are addition modulo 2^{32} (denoted $\boxplus$), XOR (denoted $\oplus$), and left rotation by fixed numbers of bits (denoted <<<). The design philosophy of Helix can be summarized as "many simple rounds". Helix has a state that consists of 5 words (Z_0 to Z_4) of 32 bits each. A single round of Helix consists of adding (or XORing) one state word into the next, and rotating the first word. Multiple rounds are applied in a cyclical pattern to the state. The horizontal lines of the rounds wind themselves in helical fashion through the five state words. Twenty rounds make up one block. Helix actually uses two interleaved helices; a single block contains two full turns of each of the helices. The critical path through the block function consists of six modulo 2^{32} additions and five XORs. In Figure 4, the half of the block of the Helix cipher is illustrated. The other half of the block is the same as the part shown in Figure 4.

6. RC4 CIPHER

RC4 is a variable key-size stream cipher developed by Ron Rivest for RSA Data Security, Inc. The RC4 stream cipher has two phases, the *key set-up* and the *keystream generation*. Both phases must be performed for every new key. During an *n*-bit key set-up the encryption key is used to generate an encrypting variable using two arrays - the state and the key array - and *n*-number of mixing operations [7].

RC4 works in Output Feedback (OFB) mode [7] of operation. In RC4 there are two 256-byte arrays, the S-box and the K-box. The S-box is linearly filled, such as S_0=0, S_1=1, S_2=2, ..., S_{255}=255. The K-box is composed of the key, repeated as many times in order to fill the array. RC4 uses two counters, *i* and *j*, which are initialized to zero. In the key set-up phase the S-box is being modified according to the following pseudo-code:

for i = 0 to 255
j = (j + S_i + K_i) mod256
swap S_i and S_j

The keystream generation phase is described by the following pseudo code:

i = (i + 1) mod256
j = (j + S_i) mod256
swap S_i and S_j
t = (S_i + S_j) mod256
K = S_t

7. COMPARISON OF IMPLEMENTATION RESULTS

The hardware implementations of A5/1, W7, Helix and E0 stream ciphers are quite straightforward, since their block diagrams are actually their hardware architectures. For more details about E0 implementation the reader is referred to [8]. For the RC4 cipher an efficient implementation, which is parameterized in order to support variable key lengths, and proposed in [9], has been adopted. The key length can be 8 up to 128-bit, opposed to the previous designs [10], [11] that support only fixed key lengths.

The results of performance (in terms of throughput) and of consumed area (in terms of CLB slices), for the implemented stream ciphers, are presented in Table 1. All the designs were synthesized in a Xilinx Virtex-II™

2V250FG256 FPGA [12], so as to have a common hardware device for the comparison. The selected FPGA has 18K bit selectRAM^{TM} blocks. Each block is synchronous and it can be easily configured in 256-byte RAM blocks. The proposed RC4 implementation utilizes a $3 \cdot 256$-byte RAM block.

Table 1: Cipher performance and area comparison

Cipher	Area (slices)	Frequency (MHz)	Throughput (Mbps)	Throughput / Area
A5/1	32	188.3	188.3	5.88
W7	608	96.0	768.0	1.26
E0	895	189.0	189.0	0.21
Helix	418	32.0	1024.0	2.45
RC4	140	60.8	120.8	0.86

As illustrated in Table 1, the Helix stream cipher achieves the largest throughput. Also, it has the second best throughput-to-area ratio. This ratio is a measure of the hardware performance of the ciphers. The A5/1 cipher has the best throughput-to-area ratio. This is an expected outcome since A5/1 has a rather simple architecture. The results for the throughput-to-area ratio for all ciphers are graphically shown in Figure 5. The E0 cipher has the smallest ratio, while the A5/1 has the largest ratio. So, among the ciphers used in well-known standards, the A5/1 achieves the best hardware performance. The throughput of W7 implementation is much better compared with the one that the A5/1 implementation achieves, but this comes with an area cost.

The latencies for the key set-up phases (initialization phases) of the presented stream ciphers are: 12.6 μs, 1.27 μs, 0.99 μs, 0.25 μs for the RC4, E0, A5/1 and Helix, respectively. So, RC4 has the largest start-up time and Helix the smallest one. The respective clock cycles for the key set-up phases are: 768, 240, 188 and 8 cycles for RC4, E0, A5/1 and Helix, respectively.

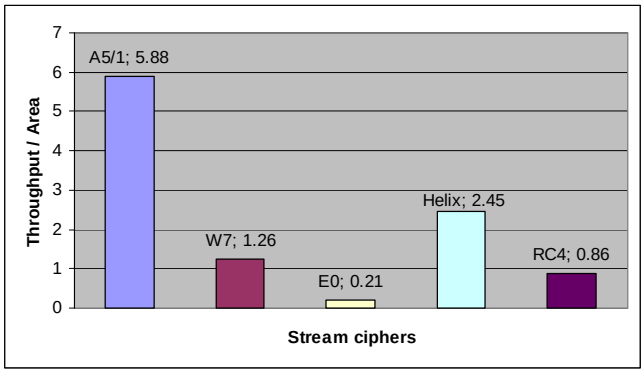

Figure 5: Throughput to area ratio results

To our knowledge there are no published hardware implementations results for the Helix, A5/1 and W7 ciphers, which can be compared with our respective implementations. The implementation results for RC4 are comparable with the ones in [11]. Our implementation is faster and consumes less area, since in [11] the area was 255 CLB slices and the throughput was 17,76 Mbps. So, our design outperforms their RC4 implementation.

8. CONCLUSIONS

In this paper, five representative stream ciphers are implemented in hardware and compared in terms of performance and consumed FPGA area. These ciphers were coded in VHDL language and synthesized in an FPGA device. The largest throughput-to-area ratio has been achieved by the A5/1 cipher and is equal to 5.88 Mbps/slice. The Helix cipher achieves the largest throughput (1024 Mbps). The throughput of the hardware implementation of the Helix cipher proves that this cipher is indeed fast, as it was shown in the comparison of its software implementation with other ciphers [4]. Also the Helix has the smallest key set-up time (0.25 μs). The implementation of the W7 cipher is much faster than the one of the A5/1. Finally, the adopted RC4 architecture outperforms previous published designs both in terms of performance and of consumed area.

9. REFERENCES

[1] "Overview of IEEE 802.11b Security", *Intel Technology Journal Q2*, 2000.

[2] "Specification of the Bluetooth system", vol. 1.1, Feb. 2001. http://www.bluetooth.com/dev/specifications.asp .

[3] "*Recommendation GSM 02.09*", European Telecommunications Standards Institute (ETSI), Security aspects.

[4] N. Ferguson, "Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive", in *Fast Software Encryption* (FSE), Feb. 24-26, 2003.

[5] S. Thomas, D. Anthony, T. Berson and G. Gong, "*The W7 Stream Cipher Algorithm*", Internet Draft, April 2002.

[6] P. Ekdahl and T. Johansson, "*Another attack on A5/1*", IEEE Transactions on Information Theory, vol. 49, no. 1, pp. 284 – 289, Jan. 2003.

[7] "*Recommendation for Block Cipher Modes of Operation. Methods and Techniques*". National Institute of Standards and Technology (NIST), Technology Administration, U.S. Department of Commerce.

[8] P. Kitsos, N. Sklavos, K. Papadomanolakis, and O. Koufopavlou, "Hardware Implementation of Bluetooth Security", in *IEEE Pervasive Computing*, vol. 2, no.1, Jan.-March 2003.

[9] P. Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware Implementation of the RC4 Stream Cipher", proceedings of 46th IEEE Midwest Symposium on Circuits & Systems (MWSCAS), December 27-30, Cairo, Egypt, 2003.

[10] Alan J. Hu, "A CPLD-Based RC4 Cracking System", in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, May 1999.

[11] Panu Hamalainen et al., "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals", in *European Signal Processing Conference* (EUSIPCO), Tampere, Finland, pp. 2289-2292, September 5-8, 2000.

[12] *Xilinx Inc*., San Jose, California, Virtex-II 2.5V FPGAs, www.xilinx.com.