ASIAN JOURNAL
ISSN: 0976-3376

Available Online at http://www.journalajst.com

ASIAN JOURNAL OF SCIENCE AND TECHNOLOGY

Asian Journal of Science and Technology
Vol. 4, Issue 07, pp.023-026, July, 2013

RESEARCH ARTICLE

HARDWARE IMPLEMENTATION OF DRAGON – A FAST WORD BASED STREAM CIPHER FOR WIRELESS SENSOR NETWORKS

Kayalvizhi, R., Vaidehi. V., Charan Kumar, U., Venkat Dubash, L. and Dinesh Chara, R. J.

Anna University, MIT campus

ARTICLE INFO

Article History:
Received 16th April, 2013
Received in revised form
28th May, 2013
Accepted 03rd June, 2013
Published online 19th July, 2013

Key words:
FPGA/Xilinx,
Dragon
Low complexity and
High speed.

ABSTRACT

This paper presents the hardware implementation of Dragon – a fast word based stream cipher, for use in wireless sensor networks. Dragon is a highly efficient word based stream cipher which takes a 128 or 256 bit key and 128 or 256 bit Initialization vector as input and produces a 64-bit key stream for each round of operation. Dragon is resistant to various forms of attacks. This paper introduces an implementation of Dragon in FPGA/Xilinx, whose memory occupancy is only 2KB which is less than the available memory of MICA2 sensor board, which is 4KB, giving compatibility for use in sensor networks due to its low complexity and high speed. The time taken for encryption is found to be 320ns due to the parallelism introduced in the hardware model and giving a data rate of 150.92MByte per second which is also compatible with 4G networks.

INTRODUCTION

The Dragon is a fast word based stream cipher which produces key stream in 64-bit words. Traditionally stream ciphers generate the key stream bit by bit. But Dragon produces the key stream in 64-bit words which allows it to combine the advantages of both block and stream ciphers. Also stream ciphers are generally based on a linear feedback shift register (LFSR) and produce statistically predictable outputs. Also bit based LFSRs are very slow. So Dragon uses a Non-linear feedback shift register (NLFSR) in its internal state which introduces non-linearity in the internal state, in addition to a non-linear filter function. The general structure of Dragon uses two blocks-a large internal state and an update function. The large internal state in Dragon makes the expected period length to be very large 2^{512}. There is also a 64-bit counter with expected period 2^{64}, which is incremented once for each iteration. So the final expected period comes to about 2^{576}. For a single value of Key and IV, 2^{64} separate key streams can be generated, which is a very small fraction of the expected period, thus preventing key stream collisions.

The existing algorithms for incorporating security in wireless sensor networks are MAC, TESLA, KAZUMI and ECC. These algorithms use key exchange methods like Diffie-Helman which are very processor intensive. All these add confusion, diffusion and non-linearity making the memory utilization as well as the power requirements very high. The proposed hardware model introduces high complexity just by using non-linearity and diffusion. The memory occupied and the data rate achieved have been calculated which outperform those of existing algorithms. Concerning with security of the algorithm, this model is found resistant to flooding attacks and gossiping attacks. Since the key and feedback are taken separately from the output, Dragon prevents the possibility of birthday paradox. The Dragon uses the same structural block containing the internal state and F-function for both key initialization and key generation, with only changes in the taps taken from the internal state and the feedback taken from the output of the F-function changing for each block. This feature of Dragon can be used to make the hardware implementation much simpler. The F-function too can be implemented as a parallel structure which improves the speed of processing.

The mathematical functions used are only XOR and 2^{32} modulo addition, which makes the complexity much simpler. Thus the hardware implementation of Dragon seems to work much faster than software implementations. Also the memory of requirements of Dragon are also very less with only two S-boxes which need to be remembered. The G and H functions can be implemented as look-up tables. So the total memory requirements of Dragon are found to be less than 4KB making it suitable for use in constrained environments like sensor networks. Also Dragon is efficient for usage in environments where frequent re-keying is necessary. Power requirements are also considerably low since the algorithm uses only non-linearity and diffusion. So this implementation of Dragon proves to be efficient for use in sensor network environment. Section 2 elaborates on the specifications of Dragon. Section 3 introduces the hardware architecture and design details. Section 4 describes the design summary and compares the performance of software and hardware implementations.

Corresponding author: R Kayalvizhi
Anna University, MIT campus

Specification of Dragon

Dragon consists of two functional blocks--- the Key initialization and Key generation block. The Dragon takes a 128 or 256 bit Key and 128 or 256 bit IV as input [1].

Key Initialization

The key initialization block receives the Key and IV and generates the 1024-bit internal state by concatenation of the Key and IV. The internal state is divided into eight words of 128 bits or four words of 256 bits in the internal state. Considering the 128-bit case, which has been implemented in this paper, the 1024 bit internal state has been divided into eight 128-bit words labeled W0, W1……W7. The inputs to the F-function a, b, c, d are derived from the taps (0, 4, 6, 7) taken from the internal state. The inputs e and f are derived from the counter M. The counter M is a 64-bit incremental counter which is initialized to a constant value at the start of the Key setup process. This block makes extensive use of the F-function. The outputs of the F-function are a', b', c', d', e', f' all 32-bit words. The outputs a', b', c', d' are given as feedback to the internal state. The counter is updated with the values of e' and f'. The process of the Key initialization process is shown below.

Input = (K, IV) (128-bit)
1. $W_d0 \| ... W_d7$ = k ‖ K'⊕ IV' ‖ IV ‖ K⊕ IV' ‖ K' ‖ K ⊕ IV ‖ IV' ‖ K'⊕ IV [128 bit]
// where K' and IV' signify swapping of key K and IV respectively)
2. M = 0x0000447261676F6E0
Key Initialization
Perform steps 3-6 Sixteen times
3. a ‖ b‖ c‖ d = (W_d0 ⊕ W_d6 ⊕ W_d7)
4. e ‖ f = M
5. (a′, b′, c′, d′, e′, f′) = F (a, b, c, d, e, f)
6. W_d0 = (a′‖ b′‖ c′‖ d′) ⊕ W_d4
7. for (W_d i = W_d i−1; i< = 7; i++) (state shifting by one)
8. M = e′‖ f′
Output = {W_d0 ‖...‖ W_d7}

Figure 1. Key initialization

Here x' refers to the swapping of the lower and the upper values of x. x- refers to the complement of x. The final values of the internal state and the final counter value are given as input to the key generation block.

The F-function

The filter function in Dragon consists of three stages. The Pre-mixing, S-box and the Post-mixing stages. The inputs to the F-function are six 32-bit words a, b, c, d, e, f which are subjected to diffusion through the G and H functions which are two non-linear 32×32 mappings. The F-function functions as both the update function and the non-linear function. The feedback factor given back to the internal state is taken from the outputs of the F-function.

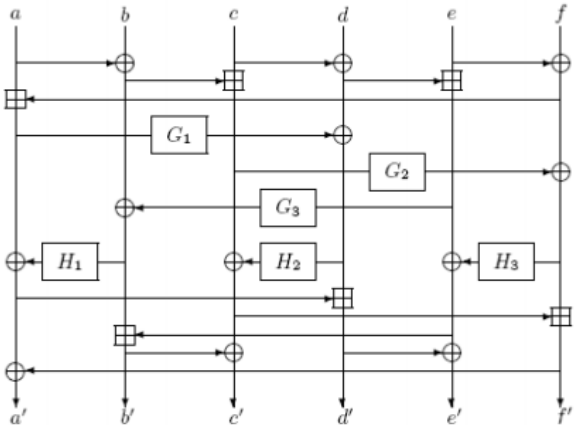

Figure 2. F function

The G and H functions are constructed from two 8×32-bit s-boxes, S1 and S2 to form virtual 32×32 s-boxes. The G function contains three S1s and one S2, while the H functions have three S2s and one S1. The 32-bit input is broken into four bytes. Each byte is passed through an 8 × 32 s-box and the four 32-bit outputs combined using binary addition. G and H functions are defined as

$$G_1(x) = S_1(x_0) \oplus S_1(x_1) \oplus S_1(x_2) \oplus S_2(x_3)$$

$$G_2(x) = S_1(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_1(x_3)$$

$$G_3(x) = S_1(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_1(x_3)$$

$$H_1(x) = S_2(x_0) \oplus S_2(x_1) \oplus S_2(x_2) \oplus S_1(x_3)$$

$$H_2(x) = S_2(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_2(x_3)$$

$$H_3(x) = S_2(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_2(x_3)$$

Figure 3. G and H functions

Key stream generation

The final value of the internal state and the counter M, from the key initialization process is given as input to this block. The counter which is a 64-bit value adds to the complexity by incrementing once for each iteration.

Input = [B0 ‖ ... ‖ B31, M]
1. (M1 ‖ M2) = M, C1‖C2 =C
2. a = B0, b = B9, c = B16, d = B19, e = B30⊕M1, f = B31⊕M2
3. (a′, b′, c′, d′, e′, f′) = F (a, b, c, d, e, f)
4. B0 = b', B1 = c'
5. Bi = B_{i-2}, 2≤ i ≤ 31
6. M = M + 1
7. k = a' ‖e'
Output = {k, B0 ‖ ... ‖ B31, M}

Figure 4. Key generation

The internal state is divided into 32×32-bit words denoted by B0, B1,……B31. The words B0, B9, B16, B19, B30and

B31which form a full positive difference set are taken from the internal block to form the inputs a, b, c, d, e, f to the F-function. The feedback value is taken from outputs b' and c', while the values e' and f' are taken as the 64-bit Key value. After the feedback has been given, the whole internal state is shifted left word by word. A 64-bit key stream is produced for each iteration of this block.

Architecture of the Dragon functional blocks

The proposed architecture of Dragon consists of the following two blocks for Key initialization and Key generation. Both blocks perform the same functions with difference in the taps for the input and the feedback from the output.

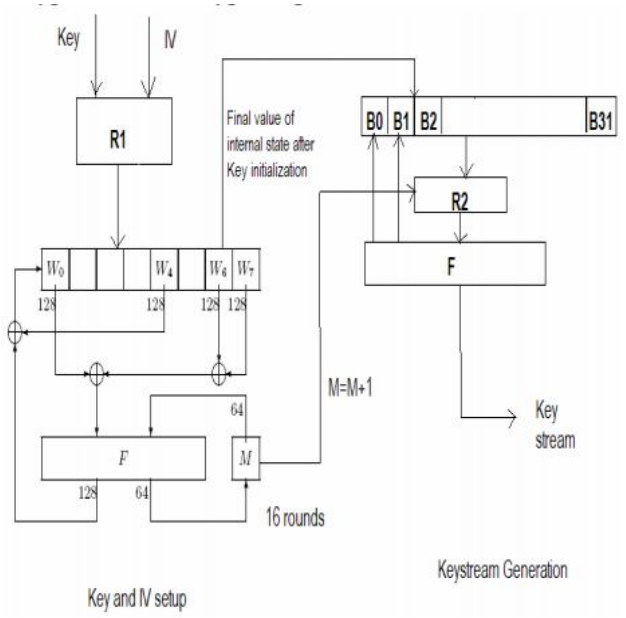

Figure 5. Hardware architecture

The S-box has been implemented as look up table to reduce the propagation delay during each memory read. There is a 1024-bit register provided to operate on the input and output data, to introduce parallelism by working on each word simultaneously. The Implementation of the three stages inside the F-function have been implemented in such a way that the output of all three stages – Pre mixing, S-box and Post mixing are produced in a single clock cycle, thus introducing a parallelism in each stage of the encryption and decryption process. Initially the key and IV are given as input to the Key initialization block. The steps 1 and 2 mentioned in the figure 1 are performed inside the R1 block. The steps 3-8 are performed 16 times during the key initialization process. Then the final values of the key initialization block are transferred to the key generation block as input and for each iteration of this block a 64-bit key stream is generated. An arbitrary 48-bit plain text is then encrypted by performing XOR with the key stream and then transmitted. At the receiver the cipher text is decrypted by performing XOR with the key stream produced at the receiver by running Dragon.

Design summary of Dragon in Xilinx

The Dragon stream cipher has been implemented in Xilinx/FPGA using verilog code. As shown in figure 6 the time taken for key setup, Encryption and Decryption has been found to be 340ns, 320ns and 320 ns from the timing diagram. There are two counters used. The first counter on decrementing to zero completes the key setup. The second counter which starts at that instant decrements to zero, generating the 64-bit key stream for each count, which is shown by the variable key gen [63:0] in Figure 6. By giving an arbitrary 48-byte input data, the number of clock cycles taken for are found to be 32 for encryption and decryption. Using the frequency of operation of the device which is 100Mhz, the throughput has been calculated by dividing the frequency of operation of the device by the number of clock cycles per byte. The throughput for this implementation in FPGA device

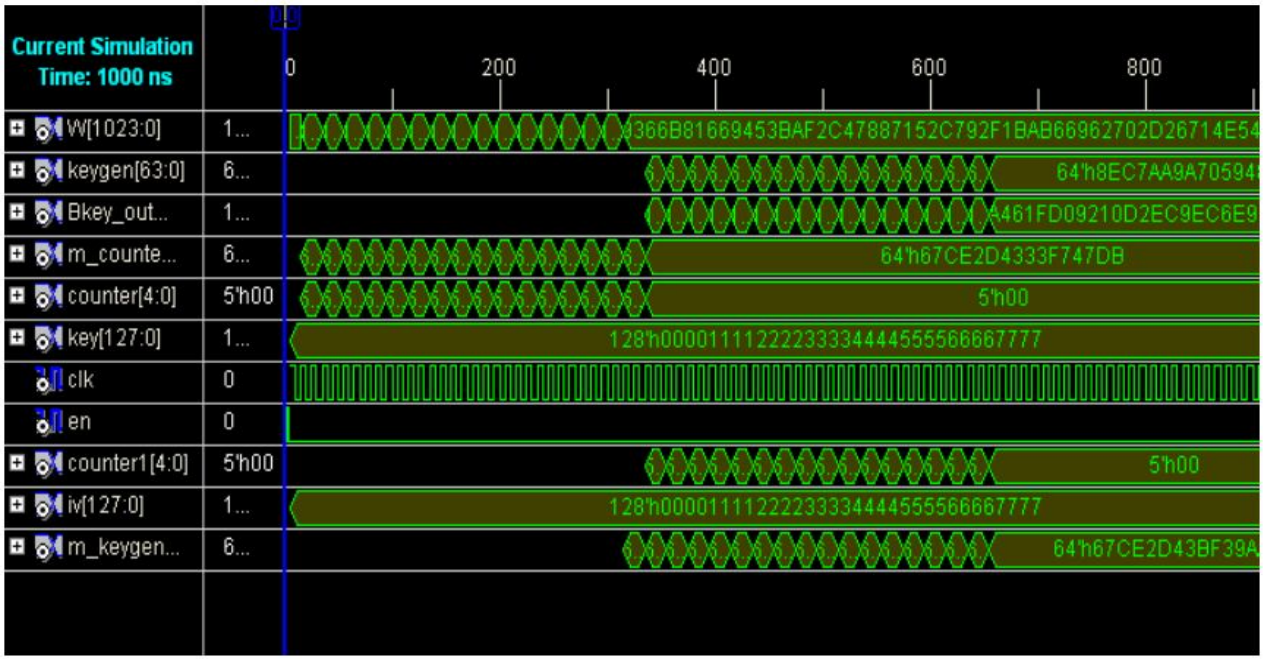

Figure 6. Simulation of Key generation in Xilinx

3s1000epq208-5 is found to be 150.92 Mbytes per second. This is compared with the software implementation using a 8-Mhz microcontroller in Table 1 and Table2.

Table 1. Comparison of performance in terms of number of clock cycles

Process	8-Mhz microcontroller (software) in CPU cycles	Xilinx/FPGA in clock cycle
Key setup	2136	34
Encryption	24227	32
Decryption	24222	32

Table 2. comparison of performance in terms of throughput

Process	8-Mhz microcontroller	Xilinx/FPGA
Encryption	42.267KB/sec	150.92MB/sec
Decryption	42.276KB/sec	150.92MB/sec

The simulation of key generation obtained in 32 clock cycles is shown in Figure 6.

Conclusion

Thus the Dragon stream cipher has been implemented in hardware and the performance analysis has been done. Based on the memory occupancy which is less than 4KB, the speed and the board utilization, it is inferred that the hardware model is suitable for use in wireless sensor networks. The device summary is shown below.

Device utilization summary:

Number of slices	17663 out of 22358	79%
Number of flip flops	7040 out of 9312	75%
Number of LUTs	32565 out of 40710	80%
Number used as Shift registers	128	
Number of IOs	2434	
Number of bonded IOBs	2433 out of 4590	53%
IOB Flip Flops	2048	
Number of GCLKs	6 out of 24	25%

REFERENCES

K. Chen, M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. Lee and S. Moon. “Dragon: A fast word based stream cipher”. ECRYPT Stream Cipher Project Report 2005/006.

Mark Luk, Ghita Mezzour, Adrian Perrig, Virgil Gligor,” Mini Sec: A Secure Sensor Network Communication Architecture” IPSN'07, Cambridge, Massachusetts, USA. April 2007.

Shuyun Lim, Chuan Chin Pu, Hyo Taek Lim, Hoonjae Lee, "Dragon-MAC: Securing Wireless Sensor Networks with Authenticated Encryption," JWIS2007 Proceeding, pp.253-264, Waseda Univ., Tokyo, Japan. August 2007.

Matt Henricksen “Tiny Dragon - an Encryption Algorithm for Wireless Sensor Networks” The 10th IEEE International Conference on High Performance Computing and Communications. 2007.

Laurent Eschenauer and Virgil D. Gligor (2002): “A Key-Management Scheme for Distributed Sensor Networks,” Conference on Computer and Communications Security. Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, 2002.

H. Chan, A. Perrig, D. Song, “Random Key Pre distribution Schemes for Sensor Networks”, IEEE Synposium on Security and Privacy, 2003

****