Implementation of Lightweight Crypto Processor Using Logistic Map for Wireless Sensor Network

Monjul Saikia and Md. Anwar Hussain

Abstract Use of a suitable cryptographic algorithm for wireless sensor networks is important due to the limitations of energy, computation capability and storage resources of the sensor nodes. Among two basic types of cryptographic techniques, namely asymmetric and symmetric key cryptography, symmetric cryptography technique is considered to be efficient over other in terms of computation cost. In symmetric key encryption, the secret key is known prior to encryption and decryption process. Therefore, in a wireless sensor network where keys are pre-distributed with Key Pre-distribution Schemes (KPS), can be used by both sides, i.e. sender and receiver nodes. In this paper, we will discuss symmetric key encryption techniques that can be efficiently used in wireless sensor network and explain the design possibilities and computation cost of using symmetric key cryptographic method. We have discussed the design of a crypto processor using a well-known chaotic map called Logistic map. Also, we have performed some experiments on the crypto processor and the results were stated.

Keywords Symmetric key encryption · Wireless sensor nodes · Encryption standards · Block cipher · Stream cipher · Security · Key pre-distribution scheme

M. Saikia (✉)
Department of CSE, North Eastern Regional Institute of Science and Technology, Nirjuli, Arunachal Pradesh, India
e-mail: msk@nerist.ac.in
URL: https://www.nerist.ac.in

M. A. Hussain
Department of ECE, North Eastern Regional Institute of Science and Technology, Nirjuli, Arunachal Pradesh, India
e-mail: ah@nerist.ac.in

R. Bera et al. (eds.), *Advances in Communication, Devices and Networking*, Lecture Notes in Electrical Engineering 537,
https://doi.org/10.1007/978-981-13-3450-4_55

Fig. 1 Symmetric key encryption setting

1 Introduction

Because of the limitations of node function, it is preferable to use symmetric key encryption technology [1, 2] over the use of public key encryption technology [3]. A series of security mechanisms for wireless sensor network is studied under security protocol IEEE802.15.4, based on Advanced Encryption Standard (AES) algorithm [1]. The Security Protocols for Sensor Network (SPINS) is established based on the symmetric key system, and a more practical security scheme for sensor networks in the security system is put forward. Simply symmetric key ciphers are used to communicate certain secret data among two parties over communication channel. The both parties need to agree on a same key called symmetric key (say K_a and K_b; $K_a = K_b$) and uses an encryption algorithm E at the sender side and similarly a decryption algorithm D at destination end. The process can be shown diagrammatically as in shown Fig. 1.

In general, the communication channel is assumed to public and therefore, there is always chance of eavesdropper who tries to sneak into the communication channel being his objective to understand the messages. Eavesdropper knows the algorithm E and D, but not K_a or K_b at the same time, which made it difficult for him to decrypt the actual message being sent. The symmetric key encryption is categorized into two types block ciphers and stream ciphers. In block ciphers, a block of data is encrypted at a time whereas in stream cipher process 1 bit at a time. In wireless sensor network application point of view stream cipher is considered to be more useful where a sensor node detects an event in real time and the same needs to be forwarded to Base station. Essentially, block cipher will be mostly applicable when there is a need of deal with tons of data. Due to advancement in processing power block cipher is used widely, where n-bit block of plaintext is processed by the encryption algorithm to produce a ciphertext block and same n-bit block is decrypted by the decryption algorithm to get back the plaintext. Block cipher may be of two types, namely Transposition cipher and Substitution cipher. The transposition cipher is rearrangement of the bits using a transposition function, whereas substitution cipher substitute number of bits with different set of bits.

Although cryptography is said to be a method for secure communication, still there is a possibility of attack. Typically, the objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext. There may be cryptanalytic attacks where it tries to gather knowledge of the algorithm from sample plaintext and ciphertext and attempts to find out the key used. Another type of attack may be Brute-force attack where it tries every possible key to obtain plaintext. Therefore, it is important to estimate the amount of effort or time required to cryptanalysis successfully before using it in a specific application. As we have stated earlier that wireless sensor nodes are less capable of complex computation and therefore light version of cryptographic method is to be adopted. Stream cipher takes one byte of plaintext as input to the encryption algorithm and produces a ciphertext. It uses pseudorandom number at each step of encryption. The algorithm is simple but yet robust against attack. In rest of the paper, we discuss about stream cipher in details and its applicability option in wireless sensor network environment.

2 Stream Cipher

Typical stream cipher encrypts plaintext one byte at a time; although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. In stream cipher, a key is input to a pseudorandom bit generator [4] that produces a stream of 8-bit numbers that are apparently random. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bit-wise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is $11001100 \oplus 01101100 = 10100000$ (ciphertext bitstream). Decryption requires the use of the same key bitstream sequence $10100000 \oplus 01101100 = 11001100$ (plaintext bitstream).

The stream cipher is similar to the one-time pad but the difference is that a one-time pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream [5]. Figure 2 is a representative diagram of stream cipher structure.

The basic idea is to generate a key stream $z = z_1 z_2 \ldots$ and use these bitstream to encrypt a plaintext string $x = x_1 x_2 \ldots$ according to the rule in Eq. 1.

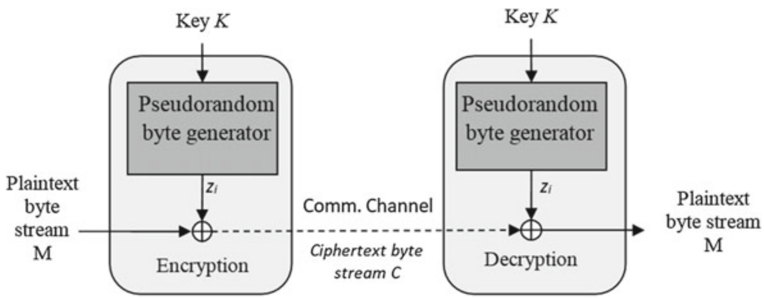


Fig. 2 Stream cipher setup

$$y = y_1 y_2 \ldots = e(z_1)(x_1)e(z_2)(x_2)\ldots \quad (1)$$

Definition 1 A synchronous stream cipher is a tuple(P, C, K, L, E, D) together with a function g, such that the following conditions are satisfied.

- P is a finite set of possible plaintext.
- C is finite set of possible ciphertext.
- K, the keyspace, is a finite set of possible keys
- L is a finite set called keystream alphabet
- g is the keystream generator. g takes a key form set K as input and generates a infinite string $z_1 z_2 \ldots$ called the keystream, where $z_i \in L$ for all $i \geq 1$.
- For each $z \in L$, there is an encryption rule $e_z \in E$ and a corresponding decryption rule $d_z \in D$. $e_z : P \to C$ and $d_z : C \to P$ are functions such that $d_z(e_z(x)) = x$ for every plaintext element $x \in P$.

As the pseudorandom byte generator is the key function in the stream cipher, we will discuss the possibility of design of random number generator which is applicable in wireless sensor network in the next section.

3 Principles of Random Number Generations

Random bitstream generation is an important cryptographic function. These random bits streams are used in both key generation and encryption process. While in generation and assignment of keys to sensor nodes, we used a specific key pre-distribution scheme [6, 7] to preload the keys to a sensor node from a key pool of keys generated using random number generator. In case of encryption we can design a pseudorandom bitstream generator that produce bitstream which is used for encryption in latter phase.

In essence, there are two fundamentally different strategies for generating random bits or random numbers. One strategy, which until recently dominated in cryptographic applications, computes bits deterministically using an algorithm. This class of random bit generators is known as pseudorandom number generators (PRNGs) or deterministic random bit generators (DRBGs). The other strategy is to produce bits nondeterministically using some physical source that produces some sort of random output. This latter class of random bit generators is known as true random number generators (TRNGs) or nondeterministic random bit generators (NRBGs).

Definition 2 Let k, l be positive integers such that $l \geq k + 1$. A (k, l)-bit generator is a function $f : \mathbb{Z}_2^k \to \mathbb{Z}_2^l$ that can be computed in polynomial time (as a function of k). The input $s_0 \in \mathbb{Z}_2^k$ is called the *seed*, and the output $f(s_0) \in \mathbb{Z}_2^l$ is called generated bitstream. It will always be required that l is a polynomial function of k.

The function f is deterministic; therefore the bitstream $f(s)$ is dependent only on the seed.

3.1 Logistic Map as Sequence Generator

The logistic map is a type of recurrence relation which is a polynomial mapping of degree 2 and chaotic behaviour can arise from very simple nonlinear dynamical Eq. 1 [8]. In the year 1976 biologist Robert May [9] discussed the logistic equation as Simple mathematical models with very complicated dynamics.

Mathematically, the logistic map is written as

$$x_{n+1} = r x_n (1 - x_n) \tag{2}$$

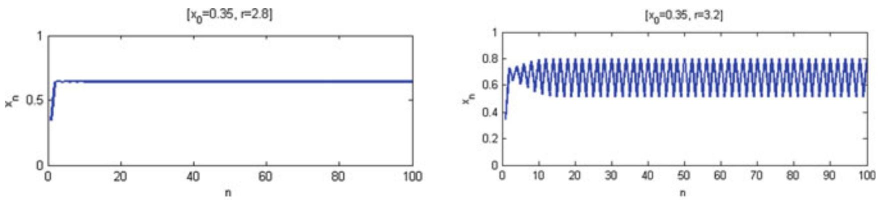
where x_n is a number in the between $[0, 1]$ and the parameter r are those in the interval $[0, 4]$. Algorithm for Logistic Map sequence generator is given in Algorithm 1.

Algorithm 1 LogisticMap()

```
x_0 ← seed                     ▷ 0 < seed < 1
r ← val                        ▷ 0 < val ≤ 4
for i = 0 → ∞ do
    x_{n+1} ← r x_n (1 − x_n)
end for
```

3.2 Chaotic Behaviour Analysis of the Logistic Map

Chaotic behaviour does not exist for all values of r. It can be seen from the following experiments by setting different values of r. At first we set values of $r = 2.8$ and we consider initial seed as $x_0 = 0.35$. It is seen that it doesn't give any randomness after some iteration Fig. 3a. Increasing the value of r to $r = 3.0$ and setting same initial seed as $x_0 = 0.35$ just oscillates the sequence between two values as shown in Fig. 3b.



(a) $x_0 = 0.35$ and $r = 2.8$ (b) $x_0 = 0.35$ and $r = 3.2$

Fig. 3 Chaotic behaviour analysis: plot of first 100 points of the orbit

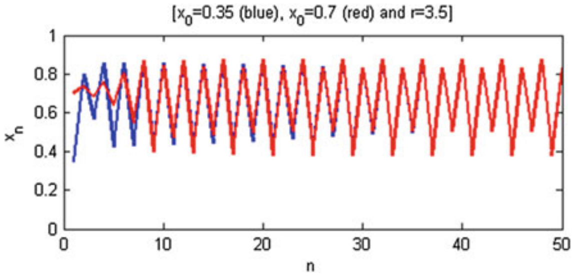


Fig. 4 Plot of first 50 points of the orbit [$x_0 = 0.35$ (blue), $x_0 = 0.70$ (red) and $r = 3.5$]

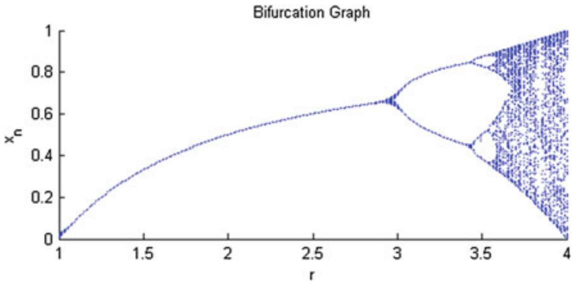


Fig. 5 The bifurcation graph for logistic map

In the next experiment we take two cases, $case1$ with the initial value of $x_0 = 0.35$ and $case2$ with initial value $x_0 = 0.7$ and set the value of $r = 3.5$. It is seen from Fig. 4 that both converge rapidly to a stable period of orbit 4.

It can also be shown that convergence occurs for any initial condition in the interval (0, 1) while we consider value of $r = 3.5$.

The general behaviour of the logistic map depends critically on value of parameter r, as we have already seen in the previous examples. If we produce graphic that captures the change in behaviour as a function of r and $r \in [0, 4]$, we get a graph called *bifurcation graph* as shown in Fig. 5.

From various experiments, it is seen that when parameter r in the range of $3.5699 < r \leq 4$, the numbers generated in successive iterations of the mapping become chaotic in nature.

Therefore we consider two nearly identical $x_0 = 0.35$ and $x_0 = 0.3501$ as initial seed conditions with the parameter $r = 4$. This shows the chaotic behaviour, which is often thought of as 'sensitive dependence on initial conditions'. In this scenario, even though the orbits are nearly identical at the start, after 100 points or so, there's no way to detect, either statistically or by looking at Fig. 6, any such correlation between the two orbits.

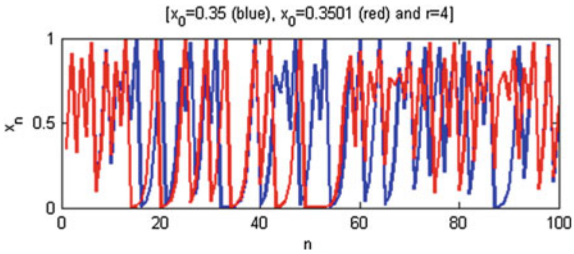


Fig. 6 Plot of two nearly identical seed [$x_0 = 0.35$ and $x_0 = 0.3501$

4 Hardware Implementation of Logistic Sequence Generator

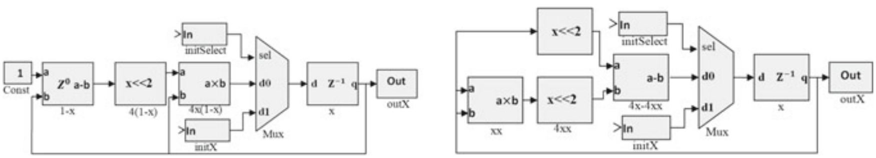
As random number generator is the basics for stream cipher, an efficient processor that can compute the bitstream is often desirable. Considering the limitations of a sensor node in wireless sensor network, design of lightweight hardware architecture [10, 11] for generation of random sequence is an important issue. The function $f(x)$ for Logistic map can be implemented in two type of architecture by taking the basic equation in the form of the following equations.

$$x_{n+1} = 4x_n(1 - x_n) \tag{3}$$

$$x_{n+1} = 4x_n - 4x_n x_n \tag{4}$$

Design makes simple while we set $r = 4$. With minimal requirement of multiplier, adder, shift registers and subtractor blocks a Logistic Map sequence generator module can be designed as shown in Fig. 7a, b using the said equations. The initial seed is feedback by module Z^{-1} to the module Z^0, where it is subtracted from constant 1. Then, multiplication with $r = 4$ is done by 2-*bit* shift register, the resultant value then passed to multiplier to get $4x(1 - x)$. Then, MUX selector selects this output to the output register, which is the generated Logistic sequence.

In the second design two 2-bit shift registers are used. The first shift register produces $4x$ by taking value of x from the feedback and second shift register finds $4xx$ from the output xx produced by the multiplier.



(a) Design with equation (3) (b) Design with equation (4)

Fig. 7 Simulink block design for logistic sequence generator

The requirement of hardware for designing a Logistic bitstream depends of number of precession considered.

5 Model for Stream Cipher Using Logistic Sequence Generator

The basic model for stream cipher in wireless sensor network is depicted in Fig. 8. The key pre-distribution scheme discussed in [7], assigns a set of keys $K_i = \{k_1, k_2, \ldots\}$ to the sensor nodes i, which were generated using a Logistic sequence generator. Sensor nodes having a shared key may undergo message exchange using that key. The key is passed to Logistic sequence generator, which produce random byte stream then and performs the encryption over plaintext. Similarly, the receiver sensor node uses the same key for its key chain to decrypt the message. If it requires further forwarding of the message, then the same strategy is being used, which is shown in Fig. 9. Figure 9 shows process of message transmission in case of multiple hop

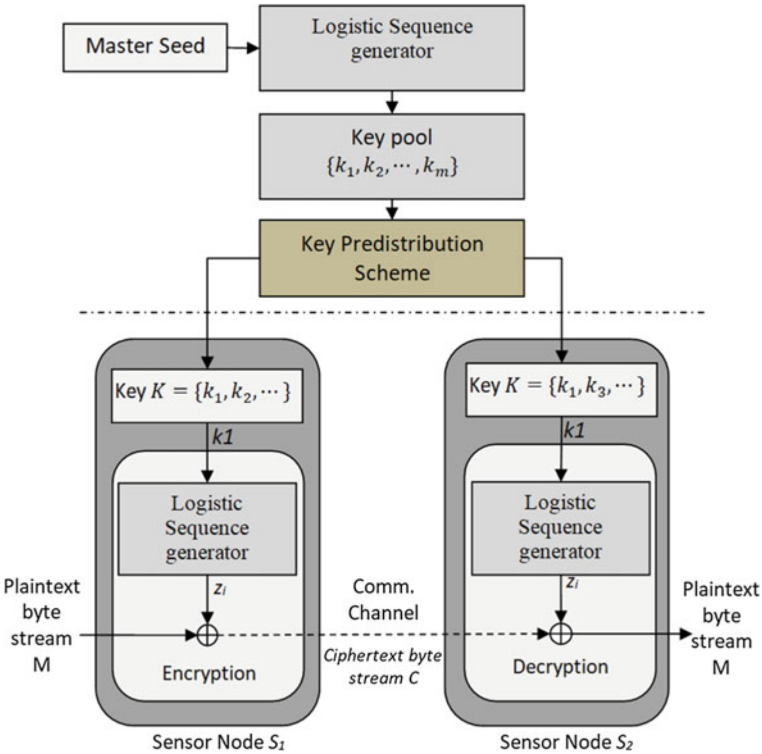


Fig. 8 Stream cipher setup for WSN

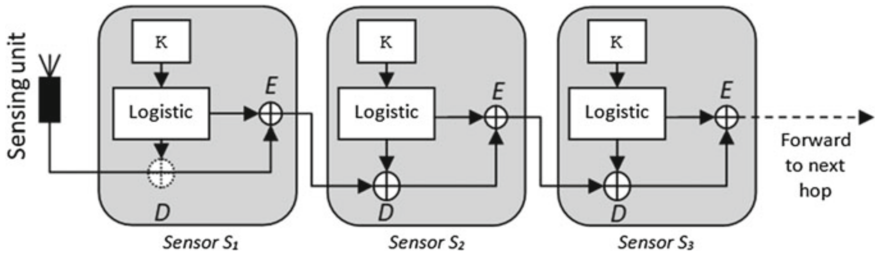


Fig. 9 Secure message forwarding in WSN multi-hop path

paths. Sensor S_1 encrypts the sensing data using shared key of sensor S_2, sensor S_2 then decrypts the data and encrypts the same with the shared key of sensor S_3 and so on.

6 Experimental Results

For our experiments, we take simple 8-*bit* logistic sequence generator. We take eight registers to store the 8 successive sequence produced by the generator. 8-*bit* precision is the initial seed to the logistic generator. The generator produces 8-*bit* sequences which are stored in 8 registers successively. Bitstreams are fetched from each significant bit of the registers. Figure 10a shows an example of the method used, where 16 numbers of bitstream outputs were shown by taking $x_0 = [1011010]$.

Encryption process is simply the XOR with the sequence of plaintext bitstream, which is as shown clearly in Fig. 10b. For the experiment, we take 8-bit stream

(a)

		$x_0 = [01011\ 010]$ precision digits							
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
bit stream (bs)	bs_1	0	0	1	1	0	0	1	1
	bs_2	1	0	0	1	1	1	1	0
	bs_3	1	0	0	0	1	1	1	0
	bs_4	1	1	0	0	1	0	1	0
	bs_5	0	0	1	0	0	0	0	0
	bs_6	0	0	1	1	0	1	1	1
	bs_7	0	1	1	0	1	1	1	0
	bs_8	1	0	0	1	1	0	0	1
		$x_8 = [10000101]$ precision digits							
		x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}
bit stream (bs)	bs_9	1	1	1	1	1	0	1	1
	bs_{10}	1	1	0	0	1	1	0	1
	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.
	bs_{16}	0	0	1	0	0	1	0	0
	.	.	.	.	.	.	.	.	.

(b)

	pt_i	bs_i	ct_i	ciphertext
plaintext 'CRYPTOGRAPHY'	01000011	01110001	00110010	2
	01010010	00010010	01000000	@
	01001001	10001110	11000111	Ç
	01010000	11000101	10010101	
	01010100	01110011	00100111	'
	01001111	01100110	00101001	)
	01000111	11110110	10110001	±
	01010010	10000101	11010111	×
	01000001	11100000	10100001	¡
	01010000	11100000	10110000	°
	01001000	10011101	11010101	Õ
	01011001	10110010	11101011	ë

Fig. 10 **a** A bitstream generation process, **b** encryption of plaintext '*CRYPTOGRAPHY*' using 8-bit sequence generator

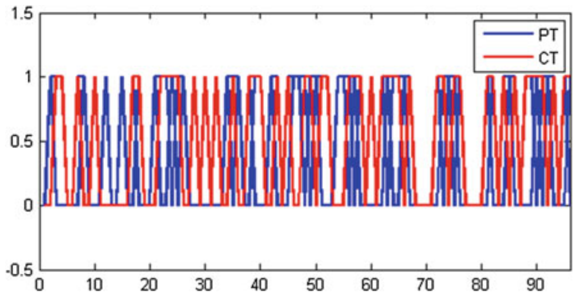


Fig. 11 Overlapping plot of plaintext and cipher text bitstream

generator and plaintext '*CRYPTOGRAPHY*' is converted to ASCII character, then converted to binary 8-bit and perform *bit-wise XOR* operation to get ciphertext.

Correlation between the plaintext and ciphertext is found to be 0.0161. A overlapping plot with plaintext (Blue line) and ciphertext (Red line) is shown in Fig. 11. Similarly, decryption can be done in the similar manner by the receiver sensor.

7 Conclusion

In this paper, we have discussed requirement of symmetric cipher as a security tool for wireless sensor network. We have discussed importance of lightweight crypto processor for a sensor node. Symmetric cipher is a simple but yet effective technique of encryption. Implementation of symmetric cipher requires random bitstream generator, which is discussed here. Hardware implementation of Logistic Map based sequence generator is discussed with its minimum requirement of logic units in two possible ways. Wireless sensor network uses key pre-distribution scheme prior to deployment of sensor node into target field, the same logistic sequence generator can be used to generate a key pool, from where keys are preloaded to the sensor nodes. In case of multi-hop path, the same crypto processor can be used for encryption and decryption processes, until data reached at its final destination.

References

1. Delfs H, Knebl H (2007) Symmetric-key encryption. Introduction to cryptography. Springer, Berlin, pp 11–31
2. Agrawal M, Mishra P (2012) A comparative survey on symmetric key encryption techniques. Int J Comput Sci Eng 4(5):877
3. Wander, AS et al (2005) Energy analysis of public-key cryptography for wireless sensor networks. In: Third IEEE international conference on pervasive computing and communications, 2005. PerCom 2005. IEEE
4. Zulfikar Z (2014) FPGA implementations of uniform random number based on residue method. Trans J Rekayasa Elektrika 11(1)

5. Lehmer DH (1954) Random number generation on the BRL high speed computing machines, by M. L. Juncosa. Math Rev 15:559
6. Eschenauer L, Gligor VD (2002) A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM conference on computer and communications security. ACM
7. Boeing G (2016) Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. Systems 4(4):37. https://doi.org/10.3390/systems4040037
8. May RM (1976) Simple mathematical models with very complicated dynamics. Nature 261(5560):459467. https://doi.org/10.1038/261459a0
9. Dabal P, Pelka R (2011) A chaos-based pseudo-random bit generator implemented in FPGA device. In: 2011 IEEE 14th international symposium on design and diagnostics of electronic circuits & systems (DDECS). IEEE
10. Blum L, Blum M, Shub M (1986) A simple unpredictable pseudo-random number generator. SIAM J Comput, no. 2
11. Pathan A-SK, Lee H-W, Hong CS (2006) Security in wireless sensor networks: issues and challenges. In: The 8th international conference on advanced communication technology, 2006. ICACT 2006, vol 2. IEEE
12. Zulfikar Z (2012) Novel area optimization in FPGA implementation using efficient VHDL code. Trans J Rekayasa Elektrika 10(2) (2012)
13. Walters JP et al (2007) Wireless sensor network security: a survey. Secur Distrib Grid Mob Pervasive Comput 1:367
14. Hoang T (2012) An efficient FPGA implementation of the advanced encryption standard algorithm. In: 2012 IEEE RIVF international conference on computing and communication technologies, research, innovation, and vision for the future (RIVF). IEEE
15. Intel Corp (2012) Intel digital random number generator (DRNG) software implementation guide, 7 Aug 2012
16. Ma D, Tsudik G (2010) Security and privacy in emerging wireless networks. IEEE Wirel Commun
17. Park S, Miller K (1988) Random number generators: good ones are hard to find. Commun ACM
18. Chan H, Perrig A, Song D (2003) Random Key predistribution schemes for sensor networks. In: Proceedings of the 2003 IEEE symposium on security and privacy. IEEE
19. May RM (1976) Simple mathematical models with very complicated dynamics. Nature 261:459
20. Tanaka H, Sato S, Nakajima K (2000) Integrated circuits of map chaos generators. Analog Integr Circuits Signal Process 25(3):329–335