Design and implementation of a new lightweight chaos-based cryptosystem to secure IoT communications.

Roll: 1807021 Date: 07/05/2023

Recap: Previously I read and grasped the contents till the "Introduction" section of this paper. This week I have read the entire paper and got a basic idea about the core contents of this paper. Summary is as follow:

My work: First comes "Background" section of the paper:

- Chaotic Systems: Choice of such systems are based on some key features:
  1. Deterministic: Deterministic and Non-probabilistic equations.
  2. Aperiodic: Irregular during temporal evolution. Doesn't converge to any steady state.
  3. Sensitivity to initial conditions: Diverge exponentially.
  4. Nonlinearity: Defined by non-linear systems of equations.
  5. Unpredictability: Extremely difficult to predict the evolution.

- Types of chaotic systems:
  1. Continuous: Defined by first order differential equations as follow:

$$\frac{dx(t)}{dt} = F(x(t), t)$$

t = Time
x(t) = State vector of dimensions >=3
F = Nonlinear dynamics of the system

Example: Lorenz, Rossler, Chen and Lu.

  2. Discrete: Generated by a temporal discontinuous progression characterized by a nonlinear recurrence equation. They are driven by an equation written in the form:

$$x(n+1) = G(x(n), n)$$

n = Index of the iteration
x(n) = State vector
G = Nonlinear dynamics of the system

Example: Henon Map, Logistic Map, Tent Map and Chirikov Map.

Now we shall see "Proposed Cyptosystem" section of the paper:

Overview:

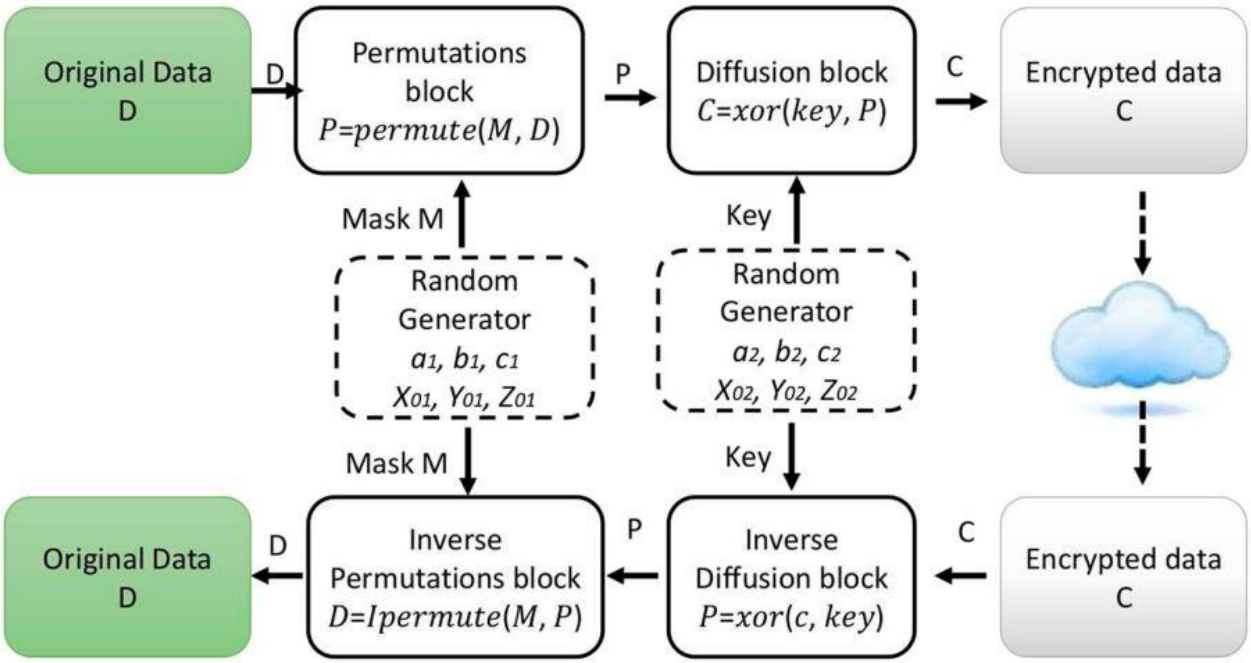

Fig: Overview of the proposed cryptosystem

This proposed cryptosystem is based on three essential blocks:

1. *Random generator block* to generate a key with high statistical properties.
2. *Chaotic permutations block* to permute data with variable permutations.
3. *Diffusion block* to "XOR" the output of the previous step with a key to produce ciphered data.

Random generator: For the random generator, "Lorenz system" is adopted to generate random keys due to its better statistical performances and low computational resource requirements. It's a deterministic chaotic system described by:

$$\frac{d\,x(t)}{dt} = a(y - x)$$

$$\frac{d\,y(t)}{dt} = cx - y - xz$$

$$\frac{d\,z(t)}{dt} = xy - bz$$

Here a, b and c are the system parameters and x_0, y_0 and z_0 are the initial conditions. To solve the above equations, numerical methods can be used such as Runge-Kutta method.

In the paper, the simulation of the above equations are performed using R-K method in MATLAB tool. This generator is used in the encryption and decryption processes.

Encryption Process

Permutation Block: Here a new pseudo-random permutation of data bits using the permute function is proposed as below:

P = permute(M, D)

Where,

D = data, M= Permutation mask generated by a chaotic generator, P = permuted data

Here 32 bits of M and 32 bits of D are considered. The algorithm is described as below:

Algorithm 1 Pseudo-code of the proposed permute function

Input: Data D (n bits), Mask M (n bits)
Output: Permuted data P (n bits)
Initialization: i=1, j=n

```
for each bit k of M do
  if M_k = 0 then
    P_k = D_i
    i = i+1
  else
    P_k = D_j
    j = j-1
  end if
end for
```

Diffusion Block: In this step, the output of the permutation block is mixed with a key generated by the chaotic generator using the XOR operation to spread redundancy of the permuted data over the cipher:

C = xor(key, P)

Decryption Process: Since it's a symmetric encryption algorithm, the decryption phase performs same process of encryption in reverse order. Therefore, the decryption step begins with the inverse diffusion and then the inverse permutation.

Inverse Diffusion: This process requires the generation of the same key used in encryption. Using the chaotic generator, the random key is generated and used in the inverse diffusion. The permuted data is retrieved using the "XOR" function:

P = xor(C, key)

Inverse Permutation: This block takes the permuted data P and the mask M to produce the data D. The function *Ipermute* takes n(32) bits of M and n bits of P to produce n bits of D as follows:

D = Ipermute(M, P)

The process of above function is depicted in pseudocode as below:

Algorithm 2 Pseudocode of the proposed inverse permute function

Input: Permuted data P (n bits), Mask M (n bits)
Output: Data D (n bits)
Initialization: i=1, j=n
  for each bit k of M **do**
    if M_k = 0 **then**
      $D_i = P_k$
      i = i+1
    else
      $D_j = P_k$
      j = j-1
  end if
end for

Performance Evolution: In this segment of the paper, the authors showed results of numerous tests analysis. These are: Memory usage, Cryptosystem speed, Energy Consumption, Security analysis, NIST statistical test, DIEHARD statistical test, Key size analysis, Key sensitivity analysis, Differential analysis, Entropy analysis, Statistical histograms, and Comparison with the closest related works. For most of the analysis, authors used the results which were found when they implemented a system of IoT devices and embedded the cryptosystem in this use case.

Conclusion: Lastly, the authors concluded the paper by summarizing their work mentioned in this paper and deciding based on results from various analysis and tests that this cryptosystem is very promising to provide a robust tool against many attacks. They also mentioned that in future they intend to improve the proposed cryptosystem by proposing a key sharing mechanism.

Doubts:

1. How initial conditions are shared to generated the same key in decryption phase?
2. How permutation mask is shared to generate data in decryption phase?