# **Anubis algorithm**

This document purpose is to describe the Anubis algorithm in general terms.

The Anubis algorithm is a cryptographic algorithm, its purpose is to encrypt a given “word”, the **plain text**, to another “word”, **cipher text**, using a specific **key** in order to transmit it safely to the receiver, which decrypt the word in his side, using the same specific key (with small changes in the algorithm itself). The algorithm flow can be seen in figure 1:

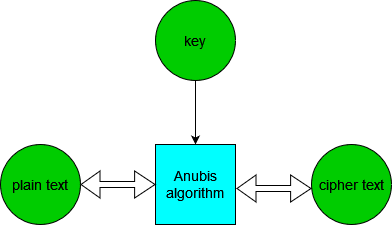


Figure 1: Anubis cryptographic flow chart

The algorithm itself is involutional, meaning : excluding some small algorithm changes (which round key goes to which round and additional theta in the key selection function), this algorithm is bidirectional, we use the same algorithm to do encryption and decryption.

**mathematical background:**

The Anubis algorithm refer the plain text and cipher text as matrices. each element in the matrix is belong to Galois field 2^8. the immediate implication is the mathematic operation are different from the regular ones:

1. adding will be represented by bitwise xor
2. multiplication will be represented by polynoms multiplication with primitive polynom: x^8 + x^4 + x^3 + x^2 + 1

the size of the matrices in the algorithm are 4x4 for the plain text and Nx4 () for the key (to keep it simple, we decided to go with 4x4 key in our project). the cipher size is the same as the plain text size, 4x4.

there for, we will treat our inputs and output (plain text, key and cipher), which represent as 128 bits vector in verilog, as 4x4 matrices, when the MSB byte is the bottom right parameter in the matrix, and the LSB is the upper left parameter in the matrix.

**algorithm flow:**

the Anubis algorithm build from 6 basic functions:

1. gamma - the sbox of the algorithm
2. tau - matrix transpose
3. theta - diffusion layer
4. pi - permutation
5. omega - key extraction
6. sigma - bitwise xor (7. round constant - using for sigma in key evolution module)

with these 6 functions we build the **round** and **key schedule** of the Anubis. the key schedule itself is build form 2 functions: key evolution and key selection.

The algorithm schema can be seen in figure 2:

Figure 2: algorithm’s schema

In figure 2, each cube represents a module in the algorithm, and the order of the modules are from the top to the bottom, where the top module output is the input of the next one underneath it.

the algorithm is built from 12 rounds. each round gets a unique key produced by key schedule module. The key schedule build from 2 modules: **key evolution** and **key selection**.

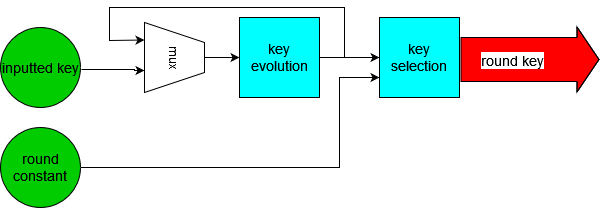
The flow of the key schedule and round modules can be seen in figures 3 and 4:

Figure 3: key schedule module flow

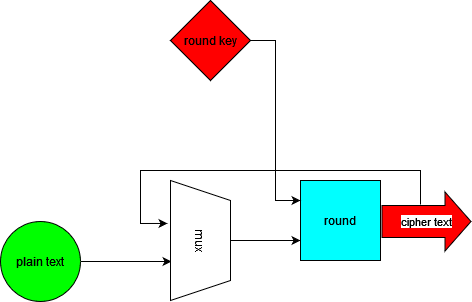


Figure 4: round module flow

In figure 3, we see that the key schedule module is activating the key evolution on the inputted key, and on the results, it’s activated the key selection function. The output of that is the **round key.** In addition, the results of the key evolution function is used as the input key for the next round.There is one exception to this: in the first round, the key schedule module activates only the key selection function on the inputted key.

in figure 4, we see the round module flow. It gets the round key created by the key schedule, and the plain text as inputs. the output in every round will be used as the input of the next round (just like the key evolution function).

At the beginning of every round, the key schedule create a round key, which later on is sent to the round module as an input. And so, it’s happened in every round.

At the end of all the rounds, we will get the **cipher text**.

**The basic functions**[[1]](#footnote-1).**:**

**Gamma:**

Gamma is the sbox in the algorithm. It replaces the input value, byte by byte, with an LUT defined in the official document of the algorithm1.

**Tau :**

Tau function is a transpose function. It takes a matrix and transpose it (matrix lines become the rows)[[2]](#footnote-2) .

**Theta:**

This is the diffusion layer. In short, this function, multiply a given matrix in with a based on MDS code generator matrix:

So, for a given 4x4 matrix, the multiplication is:

**Pi:**

Pi is the permutation function. It replaces the positions of the matrix elements in the matrix.

In general, in every column it “pushes” the column down in column number minus 1 steps. For example:

**Sigma:**

The sigma function do bitwise xor between two matrices. Sigma is used in both key schedule and round modules, but in key schedule we do xor with round constant, and in round module we do xor with the round key

**Omega**:

Omega is the key extraction function. We said earlier the key is Nx4 () matrix, and our plain text and cipher are 4x4 matrices. And since we sigma with the round key in the round module, we need it to be the same size as the plain text. That is the exact purpose of Omega function.

The way to do this is to multiple the key matrix with an 4xN () matrix.

The matrix the algorithm use is:

And so, the matrix multiplication is:

Note: the exponent in the matrix is exponent over the Galois field.

**Round constants:**

Round constants are constants that had been defined in the official document1 .

1. All the definitions of the functions and algorithm were taken from the official document of the algorithm, can be found in this link: <https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions/anubis.zip> [↑](#footnote-ref-1)
2. Matrix transpose from Wikipedia: <https://en.wikipedia.org/wiki/Transpose> [↑](#footnote-ref-2)