# HENKOS Stream Cipher

Marius Oliver Gheorghita
331 Cotofenii Fata 207013 Dolj, Romania
redwire05@yahoo.com

## Abstract

The purpose of this paper is to revise the HENKOS Stream Cipher paper present in the archive at 2004/080 and to provide a clear description of the proposed HENKOS cryptographic algorithm.

## Algorithm description

The proposed algorithm is a synchronous stream cipher, more precisely a binary additive stream cipher because it using the XOR function to encrypt the plaintext. This variant of the HENKOS stream cipher uses a named “master key (MK)” of 256 numbers (8-bit numbers) as a secret key and a named “data key (DK)” of 256 bytes (correspondent of the initialization vector - IV).

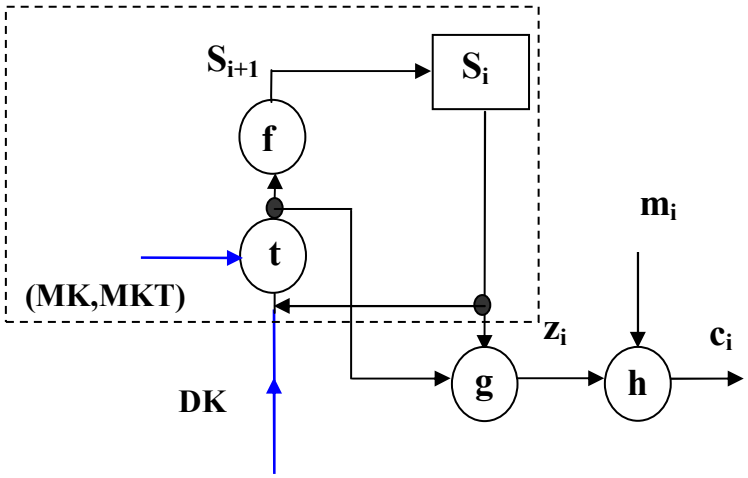


Figure 1. Model of the HENKOS stream cipher

$m_i$ – stream of plaintext
$c_i$ – stream of ciphertext
$z_i$ – keystream
t=SW function;
f=AD function;
$S_i$ – internal state i; $S_0$ – initial state;
g=XOR; h= XOR;

*Master Key Transformation*

In this part of the algorithm, from the master key (MK) is obtained the "master key transformed" (MKT). This key will be the pair of the master key in the mixing bytes process from data key initialization cycle, where the "switch" function described below will use pairs of (MK, MKT). The numbers from this key are obtained as partial sums of the numbers from MK and then taking a "mirror image" on these sums. (e.g. a mirror image of the number 255 is 552)

The master key transformed (MKT) is

$MKT_i$ =mirror $[(\sum_{j=0}^{i} MK(j)\%256)]\%256$, i = 0..255;

*Data Key initialization*

In this part of the algorithm, transform the data key (DK) in order to obtain a proper initialization before it can be used to generate the keystream. It is done using two major functions: one is the "switch" function (SW), which will mix the bytes of the data key as follows:

- the byte *j* is switched with byte *k* in the data key, where *j* is the number from the MK in the *i* position and *k* is the number from the MKT in the *i* position.

$DK_j = DK_j$ XOR $DK_k$ ;$DK_k = DK_j$ XOR $DK_k$ ;$DK_j = DK_j$ XOR $DK_k$ ;
where $j = MK_i$ and $k=MKT_i$;
i = 0..255;

The next function is an additive function AD that will replace the byte from each position with the sum between it and the byte from the right, except the last byte who is added with first byte.

$DK_i = DK_i + DK_{i+1}$ modulo 256 i = 0..254;
$DK_{255} = DK_{255} + DK_0$; $DK_0$=initial value, not calculated in this cycle.

After these two transformations, obtain an intermediate data key; to initialize the data key properly, these cycles will be repeated 64 times, without producing any output (in the figure the initialization key is the dashed-line box). After the last cycle a DK is released.

*Keystream generation*

To obtain the keystream, $z_i = g(S_{i+1}, t(S_i))$, where $S_i$ is the output from the last cycle of key initialization.

For generation of a keystream with predefined length, function g must be applied as long as necessary, using the functions described above in the data key initialization cycle.

*Encryption/decryption*
The encryption/decryption between the plaintext/ciphertext is done using XOR:
$c_i=h\ (\ m_i,\ z_i)\ ;m_i=h\ (\ c_i,\ z_i)\ ;c_i$ = ciphertext,$m_i$ = plaintext,$z_i$ = keystream;

**Security analysis**
*Time/Memory/Data tradeoff attacks*
This kind of attack has two phases: During pre-computation phase the attacker exploits the structure of the stream cipher and summarizes his findings in large tables. During the attack phase, the attacker uses these tables and the observed data to determine the secret key or the internal state of the stream cipher.
The size of the tables in the pre-computation stage, the required keystream, and the computational effort required to recover the secret key determine the feasibility of this attack. A simple way to provide security against this attack in stream ciphers is to increase the search space. In HENKOS stream cipher the size of the internal state and the secret key space is 2048 bits.

*Chosen-DK Attack*
A necessary condition for defeating differential cryptanalysis or statistical chosen-DK attacks is that the initial states for any two chosen DK's are statistically unrelated.
However, in a chosen-DK attack it is possible to reinitialize the cipher with the same master key but with a data key that differs in only one position from the previous data key. As long as data keys differ through the last bit from one byte the keystreams are not correlated, otherwise they are correlated.
In order to avoid completely these correlated keystreams, possible variants are: it can be made variable the number of the cycles in the data key initialization and/or in the SW function used in the keystream generation, the corresponding pair of the MK values can be dependant of the DK.

*Related Master Key Attack*
Related key attack is attempted to find two different master keys that will produce the same keystream. The cipher isn't vulnerable to this kind of attack, it was verified correlation between keystream produces from related master keys that differ through one bit one of another,

under a fixed data key. The test was made on 2048 keys, starting from initial key and changed every bit at the time in order to observe how that modifies the output keystream.
The test verifies also the correlation between all the key pairs derived from the tested master key and shows that all keystreams are not correlated. It can be assumed that for keys that differ through more bits, the possibility to appear correlation between produced keystreams under the same data key become negligible.

*Statistical tests analysis*
A keystream generator that exhibits basic statistical biases or detectable characteristics is weak. I have extensively tested output from HENKOS using the following statistical test package and have detected no statistical weaknesses.
The test was done for all of the 2048 keystreams tested for the related-key attack test; it was used the NIST Statistical Test Suite and every keystream was analyzed as 100 sequences of 1000000 bit each (totally the quantity of keystream involved in that test was over 190Gb).
The parameters of the battery were: block frequency length=128; non-overlapping template block length=9; overlapping template block length=9; universal block length=7; universal initialization steps=1280; approximate entropy block length=10; serial block length=16; linear complexity substring length=500;
Number of keystreams who not pass the tests was under 5%.
The included statistical tests: frequency, block frequency, cumulative sums, runs, long runs, *Marsaglia's* rank, spectral (based on the Discrete Fourier Transform), non-overlapping template matchings, overlapping template matchings, *Maurer's* universal statistical, approximate entropy, random excursions and random excursion-variant, Lempel-Ziv complexity, linear complexity, and serial.

**Summary**
This paper is intended to give a clear description regarding the design and preliminary security analysis of the proposed HENKOS algorithm in order to receive scrutiny from the cryptologic community.