HENKOS Cryptanalysis –Related keys attack

Marius Oliver Gheorghita
redwire05@yahoo.com

Abstract: This paper describes a series of vulnerabilities found on HENKOS algorithm (http://eprint.iacr.org/080) having a description below, regarding to the related key attacks, mounting this type of attack for a particular relation between keys and showing that is a practical attack, having results in real time.
Keywords: HENKOS, stream cipher, rekeying, related key attack.

Overview of HENKOS

This cryptosystem analysed in this paper is a symmetric stream cipher encryption system using two keys: a master key (MK) and a data key (DK); the master key is a secret unique key and the data key is a self generated key for each session; initially the sender and the receiver share the two keys on a trusted way.
In each session the generator uses the master key and the last generated key for encryption to produce a new data key and the key-stream which is XOR-ed with the stream of plaintext.
This cryptosystem uses a binary additive stream cipher and two types of keys:
- a short-term key named data key (DK) with a fixed length of 1024 bytes that is input in the keystream generator.

This key can be generated with a PRNG (not necessarily a cryptographic secure PRNG) or can be an ordinary file, if is not available any PRNG. DK is used in the first communication session.
- a long-term key named master key (MK) with a fixed length, which contains 1024 numbers, used to mix the data key and the internal state of the keystream generator. This key must be generated with a true RNG (hardware) and shared between the parties involved in transmission only once using a secure channel.

The algorithm can be divided in four parts: master key generation, data key generation, keystream generation, and encryption/decryption.

Master key generation

In this section of the algorithm we transform the master key (MK) in the "transformed master key" (MKT) in two steps. We used two functions: the first one is an additive function SUM and the second function INV produce a sort of symmetrical figures of number transformation.
$MKS_i = SUM(MK_i)$ $i = 0,1023$

Finally the transformed master key (MKT) is MKT_i = INV(SUM (MK_i)), i = 0,1023
The transformation has two targets:

- Not to use the original MK key directly in the process.
- To create confusion and diffusion for master key.

Data key generation
In this section of the algorithm we transform the data key (DK) to obtain the real key (K) for encryption using two important functions: the first one is the essential function in this algorithm 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 corespondent number from master key (MK) in the *i* position and *k* is the corespondent number from the transformed master key (MKT) in the *i* position.

(SW): DKj ↔ DKk where j = MKi and k=MKTi for i = 0,1023
The next function is an additive function AD which will replace the value from each position with the sum between two near bytes.
(AD): DKi = DKi + DKi+1 modulo 256 i = 0,1023
These functions create a totally changed image of the data key.
After these two transformations we obtain DK1; these cycles will be repeated 64 times: DK → DK1 → DK2 → …. → DK63 → DK64

Keystream generation
To obtain the keystream bytes (Ki) of real key (K), the last operation is the next one: Ki = (DK64i + DK64i+1) □ DK64i i = 0,1023; DK641024 = DK640

Encryption / decryption
The encryption function described bellow defines the process:
ci = XOR(Ki, pi) ci = ciphertext, pi = plain text, Ki = keystream

Rekeying / Related Key Attacks
It is very common for stream ciphers to be used repeatedly with the same secret key, loaded in combination with some varying non-secret initialisation vector. There is therefore good reason to consider the effect of this rekeying — which in effect amounts to a related key attack.
The simplest way to do this is to keep the secret key and to change the IV and if HENKOS is rekeyed in this way, then the system can become extremely weak, as we now explain.
The attack proceeds in two phases, which we will first outline and then describe in slightly more detail:

- in first phase is establishing which key is keep fixed and which is being changed under the chosen rule.
- in the second phase are produced the corresponding keystreams from the derived keys and is analyzed the correlation grade.

For the first phase are analyzed two cases: first case is when master key corresponding to the secret key is keep fixed and from data key are generated related keys, this is the common reusing of secret key and rekeying of the algorithm.
The chosen relation is the bit negation, a related key is produced from the original key with the bit from position i changed; i=1,n, where n is the length in bits of the key. This relation was chosen to observe the effect of a minimum changing in the input how is propagated in the output keystream.
The second case is the inverse one in which the data key corresponding to the IV is keep fixed and the related keys are produced from the master key based on the same relation.

Application of this attack with a chosen relation (bit negation)
The data to initiate the attack are:
Case 1:

- the corresponding secret key is considered master key mk;
- the base key for the derivation is data key dk;
- a number of successive keys are obtained based on the above relation (bit negation) from the base key;
- the algorithm is initiated with these vectors;
- the corresponding keystream sequences are available to the cryptanalyst.

Case 2:

- the corresponding IV is considered data key dk;
- the base key for the derivation is master key mk;
- a number of successive keys are obtained based on the above relation (bit negation) from the base key;
- the algorithm is initiated with these vectors;
- the corresponding keystream sequences are available to the cryptanalyst.

Input: It is been chosen a base key well defined K=(k1,…,kn) with ki
values {0,1}
Output: Weak points of the algorithm (through input key) regarding the resistance to correlation.

Step 1: It has been constructed n sets of perturbed key starting from key K:
For i=1 to n do K(i) =(S1i xor k1, …., Sni xor kn)
Sij={1 if j!=i, and 0 if i=j}
For i,j =1,….,n. so that key i is obtained from base key through bit negation on position i.

Step 2: It construct n+1 output samples (for analysis) starting from base key, perturbed keys, and a plaintext M (in this case the plaintext
is considered 0). Note this samples with C(i), i=1,…,n+1.

Step 3: It construct correlation matrix (n+1)x(n+1) CM in which the correlation values are: cv(ij)= correlation(sample i, sample j) calculated as summing the values assigned as +1 for a 1 bit and -1 for a 0 bit , divided to n for the sample(i)_xor_sample(j).
This correlation matrix is a symmetric one, having 1 on the main diagonal.

Step 4: Is counting semnificative values above the main diagonal, a semnificative value means that is in a correlation interval greater than 50% (also is verifying the strict avalanche effect, a single bit changed in input must produce a change in output greater than 50%). This number R of semnificative values is the number of rejections for the correlation test.

Step 5: If this number R counted as number of the meaning values is > 0, means that elements (i,j) from the samples, with n>=i>j>=1, are semnificative values. These elements are weak points for the algorithm, showing vulnerability for this type of cryptanalysis attack.

Results and interpretation

After the analysis of both cases, in the second case in which the related keys are obtained from the master key, there are no meaning values for a correlation grade greater than 50%. In the first case when the related keys are obtained from the data key, a common case of algorithm reinitialization, are found a number of approximate 20% from the total number of analyzed keystreams with a correlation grade greater than 50%.

Summary

If a general-purpose cipher has an n-bit key, it is expected that there should be no attack faster than n-bit exhaustive search. Has been demonstrated a related key attack against HENKOS which requires only a small number of related keys, and has very low complexity; these related keys could be available in practical use if the system is rekeyed in a certain common way, or in any other context in which a related key attack could become practical.

Bibliography

[1] Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption April 19, 2004—Version 0.15 (beta)

[2] S. Babbage, “Cryptanalysis of the LILI-128 stream cipher” in Proceedings of the second NESSIE workshop, 2001

[3] M. Blunden and A. Escott, “Related key attacks on reduced round KASUMI.” in Proceedings of Fast Software Encryption – FSE’01 (M. Matsui, ed.), Lecture Notes in Computer Science, pp. 277–285, Springer-Verlag, 2001

[4] J.Kelsey, B.Schneier, D.Wagner, C. Hall,”Cryptanalytic Attacks on Pseudorandom Number Generators” Fast Software Encryption, Fifth International Workshop Proceedings (March 1998), Springer-Verlag, 1998

[5] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography CRC Press, 1996. chapter 5, www.cacr.math.uwaterloo.ca/hac

[6] B. Schneier Applied Cryptography, Second Edition, J. Wiley & Sons Inc. 1996

[7] E.Biham, “New types of cryptanalytic attacks using related keys” in Proceedings of Eurocrypt ’93 (T. Helleseth ed.), no 765 in Lecture Notes in Computer Science pp. 398-409, Springer-Verlag, 1993

[8] R. A. Rueppel, Analysis and Design of stream ciphers. Springer-Verlag, 1986