Cryptanalysis of Hermes8F

Steve Babbage[1], Carlos Cid[2], Norbert Pramstaller[3] and Håvard Raddum[4]

[1] Vodafone Group R&D,
Newbury, United Kingdom
steve.babbage@vodafone.com
[2] Information Security Group,
Royal Holloway, University of London
Egham, United Kingdom
carlos.cid@rhul.ac.uk
[3] IAIK, Graz University of Technology
Graz, Austria
norbert.pramstaller@iaik.tugraz.at
[4] Dept. of Informatics, The University of Bergen,
Bergen, Norway
haavardr@ii.uib.no

Abstract Hermes8 [3,4] is one of the stream ciphers submitted to eSTREAM, the ECRYPT Stream Cipher Project [2]. In this paper we present an attack requiring very few known keystream bytes that recovers the cipher secret key in less than a second on a normal PC.
Keywords: Hermes8, Stream Cipher, Cryptanalysis.

1 Introduction

Hermes8 is one of the 34 stream ciphers submitted to eSTREAM, the ECRYPT Stream Cipher Project [2]. Two versions of the cipher have been defined. Originally, the cipher Hermes8 [3] was submitted. Although no weaknesses of Hermes8 were found during the first phase of evaluation, the cipher seemed to present a particularly poor performance in both software and hardware. As a result, a slightly modified version of the cipher, named Hermes8F [4], was submitted for consideration during the second phase of eSTREAM. In this paper, we focus on this second, updated version of the cipher.

2 Description of Hermes8F

According to [4], Hermes8F is a stream cipher based on the Substitution-Permutation network principle. Hermes8F is defined for two different key lengths: Hermes8F-80 uses 80-bit keys, while Hermes8F-128 uses 128-bit keys. The cipher uses two byte-oriented registers: a 17-byte state register and a 10-byte key register (16 bytes for Hermes8F-128). Additionally there is a single byte register Accu, which seems to have the use of a memory register. The diffusion is provided by moving pointers through both registers, while non-linearity is provided by the AES S-Box S [1].

The main operation of the cipher consists essentially of the following steps:

1. XOR the value stored at Accu with a byte from the state register and a byte from the key register;

2. Use the previous result as input for the AES S-Box;

3. Replace the state register value used in step 1. by the output of the S-Box;

4. Store the output of the S-Box also in Accu;

5. Increment both the state and key register pointers (denoted by $p1$ and $p2$, respectively).

The steps above are performed at each clocking. A round of the cipher consists of 17 clockings. At every 7 clockings, two bytes of the key register are updated. The updating function is also based on the AES S-Box. In the cipher's initialization, the encryption key is loaded into the key register, and the IV is loaded into the state register. The register Accu starts with the zero byte 0x00 as content. The initialization process consists of 5 rounds (i.e. 85 clockings), and so all the state registers are updated five times before the cipher enters the normal mode of operation. The first bytes of the keystream are produced after two further rounds (34 clockings). The output consists of 8 bytes from the state register, taken from alternating positions of the register. Further bytes of the output are produced at every two rounds. More details of the algorithm can be found in [4].

2.1 Alternative Description of Hermes8F

We note that it follows from the description above that during the cipher operation, the contents of the registers Accu and state$[p1-1]$ are always the same[1]. Thus a more natural description of Hermes8F is given in Figure 1. It consists of the state register R, which is represented as a feedback shift register of length 17, defined as

$$s_i^t = \text{state}[p1+i] \quad , \quad 0 \le i \le 16,$$

where state$[p1]$ is the byte addressed by pointer $p1$ at time t. This FSR is updated according to the following relations:

$$\begin{aligned} s_i^{t+1} &= s_{i+1}^t \quad , \quad 0 \le i \le 15, \\ s_{16}^{t+1} &= S(s_0^t \oplus s_{16}^t \oplus k^t), \end{aligned}$$

where the byte k^t is the output of the key register K at time t (that is, $k[p2]$), and S represents the AES S-Box.

[1] except on the very first clocking of the initialization, where Accu starts with 0x00, and state$[p1-1]$ may have another value, depending on the IV.

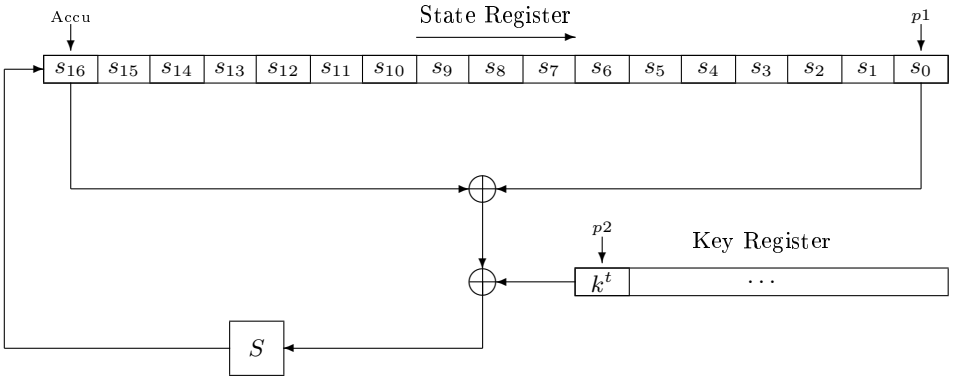

Figure1. Hermes8F as a feedback shift register.

In our attack, we need to consider the reverse cipher (clocking the generator backwards, and so generating the keystream blocks in reverse order). The relation of the feedback register of the reverse cipher is given by

$$\begin{aligned} s_0^t &= S^{-1}(s_{16}^{t+1}) \oplus s_{16}^t \oplus k^t \\ &= S^{-1}(s_{16}^{t+1}) \oplus s_{15}^{t+1} \oplus k^t. \end{aligned}$$

The inverse cipher is depicted in Figure 2.

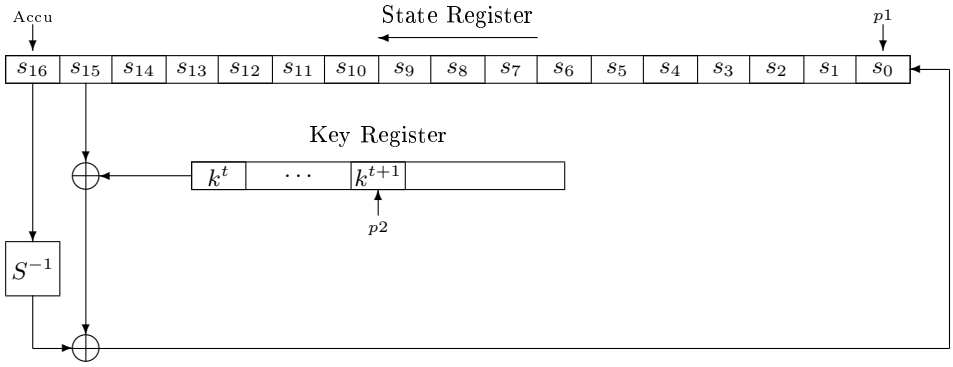

Figure2. The inverse of Hermes8F.

3 Description of the Attack

The attack we describe exploits two features of Hermes8F:

1. In contrast to the forward cipher, the reverse cipher has slow diffusion. (In the forward cipher, the new byte s_{16} contributes to the feedback in the very next clock. But in the reverse cipher, the new byte s_0 has no influence on the feedback until it has shifted all the way along to the s_{15} position.)

2. The IV does not affect the key register.

Let us consider the keystream produced by Hermes8F under a secret key and a random IV, and let B_j be the j^{th} set of 8 bytes output by the cipher. Thus, if we define $T = 34 \cdot j + 85$, we have

$$B_j = [s_0^T, s_2^T, s_4^T, s_6^T, s_8^T, s_{10}^T, s_{12}^T, s_{14}^T].$$

Consider the first two sets of B_1 and B_2, for which T is equal to $7 \times 17 = 119$ and $9 \times 17 = 153$ respectively. Suppose that in addition to the last two bytes of B_2 (that is, s_{12}^{153} and s_{14}^{153}), we also know the values of s_{13}^{153}, k^{150} and k^{149}. Then we have

$$S^{-1}(s_{14}^{153}) \oplus s_{13}^{153} \oplus k^{150} = S^{-1}(s_{16}^{151}) \oplus s_{15}^{151} \oplus k^{150} = s_0^{150}.$$

Likewise, we have that

$$S^{-1}(s_{13}^{153}) \oplus s_{12}^{153} \oplus k^{149} = S^{-1}(s_{16}^{150}) \oplus s_{15}^{150} \oplus k^{149} = s_0^{149}.$$

Now, assuming that we also know k^{133}, we have

$$S^{-1}(s_0^{150}) \oplus s_0^{149} \oplus k^{133} = S^{-1}(s_{16}^{134}) \oplus s_{15}^{134} \oplus k^{133} = s_0^{133} = s_{14}^{119}.$$

We note however that s_{14}^{119} is the last byte of B_1.

Thus consider an attack where we guess on the values of k^{133}, k^{149} and k^{150} and verify against the known byte s_{14}^{119}. The equation we have is

$$S^{-1}(S^{-1}(s_{14}^{153}) \oplus s_{13}^{153} \oplus k^{150}) \oplus S^{-1}(s_{13}^{153}) \oplus s_{12}^{153} \oplus k^{149} \oplus k^{133} = s_{14}^{119} \qquad (1)$$

where the key bytes and s_{13}^{153} are unknown. By setting $c_1 = S^{-1}(s_{14}^{153}) \oplus k^{150}$ and $c_2 = s_{12}^{153} \oplus s_{14}^{119} \oplus k^{149} \oplus k^{133}$ the equation can be more simply written as

$$S^{-1}(s_{13}^{153} \oplus c_1) \oplus S^{-1}(s_{13}^{153}) = c_2.$$

That is, a particular guess of the three key bytes is possible if and only if an input difference of c_1 to S^{-1} can lead to an output difference of c_2. We know that S^{-1} is affine equivalent to the inverse mapping in $GF(2^8)$, and thus it is rather close to being APN. This means that just under one half of all (c_1, c_2)-values are possible, or equivalently that one half of the guesses of the three key bytes remain as possible after checking them against (1).

Note that since c_2 depends on the sum $k^{149} \oplus k^{133}$ we can never learn the individual values of k^{149} and k^{133} this way, only the sum of them. Hence we are not guessing on 3-byte values but only on 2-byte values, and the complexity of guessing once is 2^{16} and not 2^{24}. By repeating the guessing for several IVs we can remove all wrong guesses, and find two bytes of information - the values of k^{150} and $k^{149} \oplus k^{133}$.

The process above can be repeated using the output bytes s_{12}^{153} and s_{10}^{153} to obtain k^{148} and $k^{147} \oplus k^{131}$, and so on, until we have 14 (or 30 in the case of Hermes8F-128) bytes of information about the key register at times $121 \leq t \leq 150$. It is then not

too hard to find the content of the key register at a specific time t, and we can run the key register back to obtain the original encryption key.

The attack requires no more than 16 bytes of output under a few (about 16) distinct IVs. In general, the complexity of the attack is of the order of $7 \times 16 \times 2^{16} < 2^{23}$ very simple operations for Hermes8F-80 (and $15 \times 16 \times 2^{16} < 2^{24}$ for Hermes8F-128). The attack (for Hermes8F-80) has been implemented on a normal workstation, and succeeds in recovering the key in less than a second.

4 Conclusion

We showed in this paper how to mount an attack to recover the secret key of Hermes8F-80 with complexity of around the order of 2^{23} operations, requiring a very small number of known keystream bytes.

We note that the attack described above only works against the latest version of the cipher, which in our opinion is the one of most interest. However, our impression is that the attack can probably be modified to apply against the original version Hermes8. Certainly the features that we have exploited - the simpler representation of the generator as a shift register, slow diffusion of the reverse clocking cipher, and the fact that the key register is not IV-dependent - apply also to Hermes8.

Acknowledgments

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

1. J. Daemen and V. Rijmen. *The Design of Rijndael.* Springer–Verlag, 2002.
2. eSTREAM, the ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/`.
3. U. Kaiser. Hermes8 : A Low-Complexity Low-Power Stream Cipher. Cryptology ePrint Archive, Report 2006/019. `http://eprint.iacr.org/2006/019.pdf`.
4. U. Kaiser. Hermes8F : A Low-Complexity Low-Power Stream Cipher. eSTREAM, the ECRYPT Stream Cipher Project, Second Phase Ciphers. `http://www.ecrypt.eu.org/stream/buildsite/p2ciphers/hermes8/hermes8f_p2.pdf`.