A Chosen-key Distinguishing Attack on Phelix

Yaser Esmaeili Salehani* and Hadi Ahmadi**

* Zaeim Electronic Industries Co., Tehran, Iran.
** School of Electronic Engineering, Sharif University of Technology, Tehran, Iran.
yesmaeili@zaeim.com, hadimc@mehr.sharif.edu

Abstract. Phelix is an asynchronous word-oriented stream cipher submitted to the ECRYPT stream cipher project, with a variable-length key between 128 and 256 bits and is claimed to have 128-bit security. The algorithm is similar to the earlier stream cipher Helix which is showed to be vulnerable. In this paper, we introduce a chosen-key differential distinguishing attack on the stream cipher Phelix with no large complexity. The results show that this stream cipher is vulnerable if the attacker has the ability to re-encrypt the unknown message with a defined modification of the unknown secret key.

1 Introduction

In a synchronous stream cipher, a stream of pseudo-random digits is generated independently of the plaintext and ciphertext messages, and then combined with the plaintext (to encrypt) or the ciphertext (to decrypt). Another approach uses several of the previous ciphertext digits to compute the key stream. Such schemes are known as self-synchronizing (or asynchronous) stream ciphers stream ciphers. In other words, asynchronous stream ciphers are a group of stream ciphers in which the running key is obtained from both the secret key and the plain text.

Phelix is a high-speed asynchronous stream cipher with built-in message authentication code (MAC) functionality, submitted in 2004 to the eSTREAM contest by Whiting *et al* [5]. It uses a 256-bit key and a 128-bit nonce, and is claimed to have 128 bit security. An unusual feature of Phelix is that it uses the plaintext to update the state, which in turn affects the key-stream. This allows Phelix to perform authentication as well as encryption.

Phelix is a slightly modified form of an earlier cipher, Helix, published in 2003 by Ferguson *et al* [1]. In fact, the block functions of the two ciphers are identical, except for the key stream output function. The enlarged internal state of Phelix, along with moving the key stream output point to force larger plaintext diffusion, would seem to increase the security margin significantly. These changes were made largely in response to [2].

In 2004, Muller published two attacks on Helix [2]. The first has a complexity of 2^{88} and requires 2^{12} adaptive chosen-plaintext words, but requires nonces to be reused. Paul and Preneel later showed that the number of adaptive chosen-plaintext words of Muller's attack can be reduced by a factor of 3 in the worst case (a factor of 46.5 in

the best case) using their optimal algorithms to solve differential equations of addition [3]. In a later development, Paul and Preneel showed that the above attack can also be implemented with chosen plaintexts (CP) rather than adaptive chosen plaintexts (ACP) with data complexity $2^{35.64}$ CP's [4]. Muller's second attack on Helix is a distinguishing attack that requires 2^{114} words of chosen plaintext [2].

In this paper, we analyze the chosen plaintext distinguishing attack scenario to find whether the stream cipher Phelix is vulnerable against this type of attack. Next, we change the scenario to a chosen key attack to analyze the strength of Phelix. In section 2 a short description of Phelix is given. Afterwards, we apply a chosen plaintext distinguishing attack scenario on Phelix in section 3. Next, we introduce our chosen key distinguisher in section 4. Section 5 includes the results along with concluding remarks.

2 A Short Description of Phelix

Phelix is a combined stream cipher and MAC function, and directly provides authenticated encryption functionality. By incorporating the plaintext into the stream cipher state, Phelix can provide authentication functionality without extra cost. It produces a ciphertext message and a tag that provides authentication.

Phelix uses a variable-length key (**U**) up to 256 bits and a 128-bit nonce (**N**). The key is secret, and the nonce is typically public knowledge. Phelix is optimized for 32-bit platforms; all operations are on 32-bit words. The only operations used are addition modulo 2^{32} (ADD), exclusive or (XOR), and rotation by fixed numbers of bits (ROTATE).

Three types of basic operations on words are used in the round function 'H' of Phelix:

- bitwise addition represented as $\oplus$
- addition modulo 2^{32} represented as $\boxplus$
- cyclic shifts represented as <<<

H relies on two sequential applications of a single "phelix" function, which constitutes half of H round function. This "phelix" function is represented as H^* shown in Fig. 1 and the relation (1).

$$(O_0, O_1, O_2, O_3, O_4) = H^*(I_0, I_1, I_2, I_3, I_4, A, B) \quad (1)$$

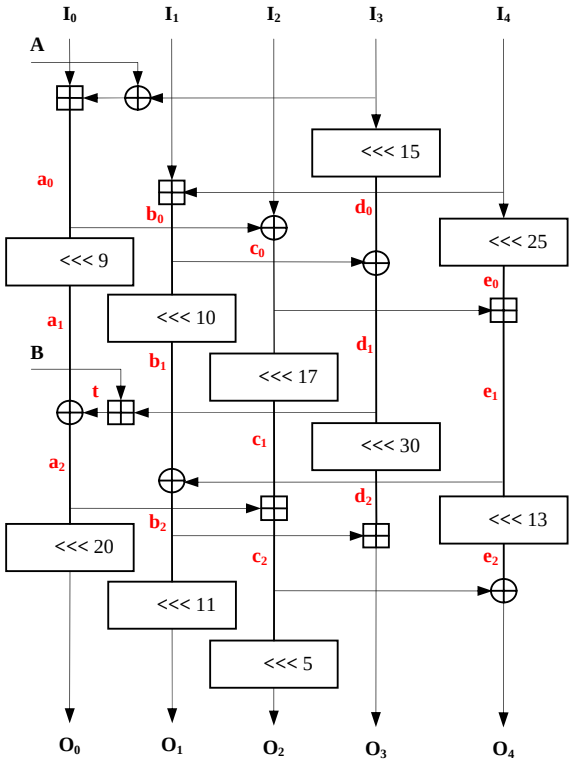

Fig. 1. The half-round "Phelix" function H^* (phelix)

H^* uses two auxiliary inputs (A, B). In the first half of the round function, (A, B) = (0, $X_{i,0}$) and in the second half, (A, B) = (P_i, $X_{i,1}$). Thus, the block function can be described by the following relations:

$$\begin{aligned}(Y_0^{(i)},\ldots,Y_4^{(i)}) &= H^*(Z_0^{(i)},\ldots,Z_4^{(i)},0,X_{i,0})\\ (Z_0^{(i+1)},\ldots,Z_4^{(i+1)}) &= H^*(Y_0^{(i)},\ldots,Y_4^{(i)},P_i,X_{i,1})\end{aligned}, \qquad (2)$$

where $(Y_0^{(i)},\ldots,Y_4^{(i)})$ is the internal state in the middle of the computation. To protect the cipher against related-key attacks, a first step is applied that computes a working key **K** from the actual secret key **U**. Independently of its length l(**U**), **K** is always 256 bits long and is used in all subsequent operations instead of **U**. The derivation of **K** is based on 8 rounds of a Feistel network. The result is also represented as 8 words: K_0, …, K_7. The output key stream is obtained as follows:

$$S_i = Y_4^{(i)} \boxplus Z_4^{(i-4)} \qquad (3)$$

The nonce **N** is used to obtain different key stream sequences with the same secret key. **N** is always 128 bits long and is represented as 4 words: N_0, …, N_3. An expansion phase turns it into a 256 bits value by creating 4 additional words N_4, …, N_7 defined as $N_{k+4} = (k \bmod 4) - N_k$, $k = 0,1,2,3$. During the *i*-th round of encryption, the round key words $X_{i,0}$ and $X_{i,1}$ are computed as:

$$\begin{aligned} X_{i,0} &= K_{i \bmod 8} \\ X_{i,1} &= K_{(i+4) \bmod 8} + K_{i \bmod 8} + \hat{X}_i + i + 8 \ , \\ , \hat{X}_i &= \begin{cases} \lfloor (i+8)/2^{31} \rfloor, & if\ i \bmod 4 = 3 \\ 4.l(U), & if\ i \bmod 4 = 1 \\ 0, & \text{Otherwise.} \end{cases} \end{aligned} \qquad (4)$$

where + and – mean addition and subtraction modulo 2^{32} respectively. Note that it is straightforward to reconstruct the secret key from these values for consecutive rounds when the nonce is known.

2.1 Key Mixing

The key mixing process converts a variable-length input key **U** to a fixed-length working key **K**. First, Phelix block function is used to create a round function *R* that maps 128 bits to 128 bits, as follows:

```
Function R(w0,w1,w2,w3)
Begin
      Local Variable w4 = l(U) + 64;
      (w0,w1,w2,w3,w4) = H(w0,w1,w2,w3,w4,0,0);
      (w0,w1,w2,w3,w4) = H(w0,w1,w2,w3,w4,0,0);
      Return (w0,w1,w2,w3);
End.
```

The input key **U** is first extended with 32-*l*(**U**) zero bytes. The 32 key bytes are converted to 8 words K_{32}, …, K_{39}. Further key words are defined by the following relation and the words K_0, …, K_7 form the working key of the cipher.

$$(K_{4i},\ldots, K_{4i+3}) = R(K_{4i+4},\ldots, K_{4i+7}) \oplus (K_{4i+8},\ldots, K_{4i+11}), \quad i = 7,\ldots, 0 \qquad (5)$$

3 A Chosen Plaintext Distinguishing Attack Scenario

In this scenario, the attacker uses a cryptosystem to encrypt pairs of different messages with a unique secret key and wants to know if the cryptosystem is Phelix or not. Suppose that the unknown cryptosystem is Phelix, so when he requests the system to encrypt a pair of messages $P = (P_1, \ldots, P_{n-1}, P_n)$ and $P' = (P_1, \ldots, P'_{n-1}, P_n)$, the key streams are $S = (S_1, \ldots, S_{n-1}, S_n)$ and $S = (S_1, \ldots, S_{n-1}, S'_n)$ respectively, according to the description of Phelix stream cipher. Consequently the ciphertexts are

$C = (C_1, \ldots, C_{n-1}, C_n)$ and $C = (C_1, \ldots, C_{n-1}, C'_n)$. Since P_i and C_i are both 32-bit words, if a known difference between P'_{n-1} and P_{n-1} leads to a definite deference between C'_n and C_n with a probability of more than 2^{-32}, we can imply that this scenario is able to distinguish Phelix from an unknown random cryptosystem. In order to find if such a high probability differential path from plaintext to ciphertext exists, we first define two notations $\delta_{i_1,i_2,\ldots,i_n}$ and $\Delta_{i_1,i_2,\ldots,i_n}$ which indicate a 32-bit known word with 1s in positions $i_1,i_2,\ldots,i_n$ and 0s in other positions, respectively used for XOR and modular differentials. It is obvious that the two notations are the same and such a distinct definition is just used for the simplicity of describing the process.

A general process of finding a differential path from plaintext to ciphertext is proposed as follows. We first set a modular difference between P'_{n-1} and P_{n-1}, ex. Δ_{31}. So the XOR difference is also δ_{31} with the probability of 1. Passing through the encryption process we change the modular differential to XOR differential, whenever we face into XOR or ROTATE operations, and vice versa, whenever we face into a modular ADD operation (the corresponding differential is indicated in **bold** script in Table 1). Note that this transformation holds with a definite probability. For example if we have the XOR differential $\delta_{1,4,31}$ the corresponding modular differential is $\Delta_{1,4,31}$ with the probability of 2^{-2}, due to the numbers of 1s in the differential (except the position 31) which may cause carries in modular differential with the probability of ½. Table 1 shows such a differential path when $P'_{n-1} \oplus P_{n-1} = \delta_{31}$.

Table 1. A chosen plaintext differential path from P_{n-1} to $Y_4^{(n)}$

a) Round #1: $(Z_0^{(n)},\ldots,Z_4^{(n)}) = H^*(Y_0^{(n-1)},\ldots,Y_4^{(n-1)},P^*_{n-1}, X_{n-1,1})$

Pair	(XOR diff., Pr)	(Modular diff., Pr)
(a'_0, a_0)	$(\delta_{31}, 1)$	**$(\Delta_{31}, 1)$**
(a'_1, a_1)	**$(\delta_8, 1)$**	$(\Delta_8, 2^{-1})$
(b'_0, b_0)	$(0, 1)$	**$(0, 1)$**
(b'_1, b_1)	**$(0, 1)$**	$(0, 1)$
(c'_0, c_0)	**$(\delta_{31}, 1)$**	$(\Delta_{31}, 1)$
(c'_1, c_1)	**$(\delta_{16}, 1)$**	$(\Delta_{16}, 2^{-1})$
(d'_0, d_0)	**$(0, 1)$**	$(0, 1)$
(d'_1, d_1)	**$(0, 1)$**	$(0, 1)$
(e'_0, e_0)	**$(0, 1)$**	$(0, 1)$
(e'_1, e_1)	$(\delta_{31}, 1)$	**$(\Delta_{31}, 1)$**
(t', t)	$(0, 1)$	**$(0, 1)$**
(a'_2, a_2)	**$(\delta_8, 1)$**	$(\Delta_8, 2^{-1})$
(z'_0, z_0)	**$(\delta_{28}, 1)$**	$(\Delta_{28}, 2^{-1})$
(b'_2, b_2)	**$(\delta_{31}, 1)$**	$(\Delta_{31}, 1)$
(z'_1, z_1)	**$(\delta_{10}, 1)$**	$(\Delta_{10}, 2^{-1})$
(c'_2, c_2)	$(\delta_{8,16}, 2^{-4})$	**$(\Delta_{8,16}, 2^{-2})$**
(z'_2, z_2)	**$(\delta_{13,21}, 2^{-4})$**	$(\Delta_{13,21}, 2^{-6})$
(d'_2, d_2)	**$(0, 1)$**	$(0, 1)$

(z'_3, z_3)	$(\delta_{31}, 1)$	$\mathbf{(\Delta_{31}, 1)}$
(e'_2, e_2)	$\mathbf{(\delta_{12}, 1)}$	$(\Delta_{12}, 2^{-1})$
(z'_4, z_4)	$\mathbf{(\delta_{8,12,16}, 2^{-4})}$	$(\Delta_{8,12,16}, 2^{-7})$

b) Round #2: $(Y_0^{(n)},\ldots,Y_4^{(n)}) = H^*(Z_0^{(n)},\ldots,Z_4^{(n)}, 0, X_{n,0})$

	(XOR diff., Pr)	**(Modular diff., Pr)**
(a'_0, a_0)	$(\delta_{28,31}, 2^{-2})$	$\mathbf{(\Delta_{28,31}, 2^{-1})}$
(a'_1, a_1)	$\mathbf{(\delta_{5,8}, 2^{-2})}$	$(\Delta_{5,8}, 2^{-4})$
(b'_0, b_0)	$(\delta_{8,10,12,16}, 2^{-12})$	$\mathbf{(\Delta_{8,10,12,16}, 2^{-8})}$
(b'_1, b_1)	$\mathbf{(\delta_{18,20,22,26}, 2^{-12})}$	$(\Delta_{18,20,22,26}, 2^{-16})$
(c'_0, c_0)	$\mathbf{(\delta_{13,21,28,31}, 2^{-6})}$	$(\Delta_{13,21,28,31}, 2^{-9})$
(c'_1, c_1)	$\mathbf{(\delta_{6,13,16,30}, 2^{-6})}$	$(\Delta_{6,13,16,30}, 2^{-10})$
(d'_0, d_0)	$\mathbf{(\delta_{14}, 1)}$	$(\Delta_{14}, 2^{-1})$
(d'_1, d_1)	$\mathbf{(\delta_{8,10,12,14,16}, 2^{-12})}$	$(\Delta_{8,10,12,14,16}, 2^{-17})$
(e'_0, e_0)	$\mathbf{(\delta_{1,5,9}, 2^{-4})}$	$(\Delta_{1,5,9}, 2^{-7})$
(e'_1, e_1)	$(\delta_{1,5,9,13,21,28,31}, 2^{-22})$	$\mathbf{(\Delta_{1,5,9,13,21,28,31}, 2^{-16})}$
(t', t)	$(\delta_{8,10,12,14,16}, 2^{-22})$	$\mathbf{(\Delta_{8,10,12,14,16}, 2^{-17})}$
(a'_2, a_2)	$\mathbf{(\delta_{5,10,12,14,16}, 2^{-24})}$	$(\Delta_{5,10,12,14,16}, 2^{-29})$
(z'_0, z_0)	$\mathbf{(\delta_{25,30,0,2,4}, 2^{-24})}$	$(\Delta_{25,30,0,2,4}, 2^{-29})$
(b'_2, b_2)	$\mathbf{(\delta_{1,5,9,13,18,20,21,22,26,28,31}, 2^{-34})}$	$(\Delta_{1,5,9,13,18,20,21,22,26,28,31}, 2^{-44})$
(y'_1, y_1)	$\mathbf{(\delta_{0,1,5,7,10,12,16,20,24,29,31}, 2^{-34})}$	$(\Delta_{0,1,5,7,10,12,16,20,24,29,31}, 2^{-44})$
(c'_2, c_2)	$(\delta_{5,6,10,12,13,14,17,30}, 2^{-47})$	$\mathbf{(\Delta_{5,6,10,12,13,14,17,30}, 2^{-39})}$
(y'_2, y_2)	$\mathbf{(\delta_{3,10,11,15,17,18,19,22}, 2^{-47})}$	$(\Delta_{3,10,11,15,17,18,19,22}, 2^{-54})$
(d'_2, d_2)	$\mathbf{(\delta_{6,8,10,12,14}, 2^{-12})}$	$(\Delta_{6,8,10,12,14}, 2^{-17})$
(y'_3, y_3)	$(\delta_{1,5,6,8,9,10,12,13,14,18,20,21,22,26,28,31}, 2^{-76})$	$\mathbf{(\Delta_{1,5,6,8,9,10,12,13,14,18,20,21,22,26,28,31}, 2^{-61})}$
(e'_2, e_2)	$\mathbf{(\delta_{2,9,12,14,18,22,26}, 2^{-22})}$	$(\Delta_{2,9,12,14,18,22,26}, 2^{-29})$
(y'_4, y_4)	$\mathbf{(\delta_{2,5,6,9,10,13,17,18,22,26,30}, 2^{-69})}$	$(\Delta_{2,5,6,9,10,13,17,18,22,26,30}, 2^{-80})$

Table 1 indicates that there is an XOR/modular differential path from P_{n-1} to $y_4^{(n)}$ which occurs with the probability of 2^{-80} (considering output modular differential). The relation (3) implies that the modular differential of S_n, in this scenario, is equal to the modular differential of $y_4^{(n)}$ and is 2^{-80} which is greater than 2^{-32} and could not be a distinguisher. It is considerable that we can not state that having the plaintext (P_{n-1}) differential of Δ_{31} leads to key word (S_n) differential of $\Delta_{2,5,6,9,10,13,17,18,22,26,30}$ with the probability of 2^{-80}, because this probability is provided that the differential path from plaintext to the key stream is limited by Table 1. This general process of finding an XOR/modular differential path shows that Phelix stream cipher is well designed against this type of attack.

4 The Chosen Key Distinguishing Attack

In this section, we show that applying two chosen key to encrypt same plain texts could lead to a successful distinguisher. To describe the attack more precisely, it is necessary to consider the chosen key distinguishing attack scenario as follows:

a. The attacker requests encryption of a random (or even unknown) message P = (P_1, …, P_n) under the pair of (key, nonce) = (**K**, **N**). The resulting cipher text is C = (C_1, …, C_n).
b. She requests re-encryption of the same plaintext under the pair of (key, nonce) = (**K**$'$, **N**). This assumption yields the cipher text is C$'$=(C'_1, …, C'_n).
c. If she perceives $Y'^{(n)}_4$ - $Y^{(n)}_4$ = Δ' with a probability of less than 2^{-32}, so the cryptosystem is distinguished to be the desired algorithm.

Applying the above-mentioned scenario on Phelix restricts us to use n < 4, due to the relation (4). Suppose that the attacker chooses the two secret keys **K** and **K'** so that the XOR and modular differences between $X'_{n,0}$ and $X_{n,0}$ (n = 1, 2, 3) are δ_{31} and Δ_{31}. It is very simple to find that how such a key differential can affect the running key stream by tracing the differential path as shown in Table 2.

Table 2. A chosen key differential path which can lead to a distinguisher

a) Round #1: $(Z_0^{(n)},...,Z_4^{(n)}) = H^*(Y_0^{(n-1)},...,Y_4^{(n-1)},P_{n-1}, X_{n-1,1})$

Pair	(XOR diff., Pr)	(Modular diff., Pr)
(a'_0, a_0)	(0, 1)	(0, 1)
⋮	⋮	⋮
(z'_4, z_4)	(0,1)	(0,1)

b) Round #2: $(Y_0^{(n)},\ldots,Y_4^{(n)}) = H^*(Z_0^{(n)},\ldots,Z_4^{(n)},0, X^*_{n,0})$

	(XOR diff., Pr)	(Modular diff., Pr)
(a'_0, a_0)	(0,1)	**(0,1)**
(a'_1, a_1)	**(0,1)**	(0,1)
(b'_0, b_0)	(0,1)	**(0,1)**
(b'_1, b_1)	**(0,1)**	(0,1)
(c'_0, c_0)	**(0,1)**	(0,1)
(c'_1, c_1)	**(0,1)**	(0,1)
(d'_0, d_0)	**(0,1)**	(0,1)
(d'_1, d_1)	**(0,1)**	(0,1)
(e'_0, e_0)	**(0,1)**	(0,1)
(e'_1, e_1)	(0,1)	**(0,1)**
(t', t)	(δ_{31}, 1)	**(Δ_{31}, 1)**
(a'_2, a_2)	**(δ_{31}, 1)**	(Δ_{31}, 1)
(z'_0, z_0)	**(δ_{19}, 1)**	(Δ_{19}, 2^{-1})
(b'_2, b_2)	**(0,1)**	(0,1)

(y'_1, y_1)	**(0,1)**	(0,1)
(c'_2, c_2)	$(\delta_{31}, 1)$	$\mathbf{(\Delta_{31}, 1)}$
(y'_2, y_2)	$\mathbf{(\delta_4, 1)}$	$(\Delta_4, 2^{-1})$
(d'_2, d_2)	**(0,1)**	(0,1)
(y'_3, y_3)	(0,1)	**(0,1)**
(e'_2, e_2)	**(0,1)**	(0,1)
(y'_4, y_4)	$\mathbf{(\delta_{31}, 1)}$	$(\Delta_{31}, 1)$

Thus we have found a differential path from the mixed-key to the ciphertext with the probability of 1. Considering section 2.1 and the relation (5), the mixed-key words K_0 to K_7 are calculated from the secret key words K_{32} to K_{39} by repeatedly using the mixing function 'R', we call this transformation function *MIX(.)*. It is obvious that we can also find the secret key words from the mixed-key words by another inverse function, so called *RMIX(.)*. Therefore, by calculating $(K_0, \ldots, K_7) = MIX(K_{32}, \ldots, K_{39})$, adding the n-th resulted word (n = 1, 2, 3) by Δ_{31} and, obtaining $(K'_{32}, K'_{33}, \ldots, K'_{39}) = RMIX(K_0, \ldots, K'_n, \ldots, K_7)$ we can generate a dual secret key which can lead to distinguishing the cryptosystem. In other words, suppose that the attacker does not know the secret key and the plaintext, but she has the ability to force the cryptosystem to encrypt every plaintext twice, once with the secret key and once with a new dual key obtained from the above-mentioned process. In this situation, the attacker can distinguish the cryptosystem if Phelix, when the ciphertext differential is Δ_{31}, without any large complexity.

5 Conclusions

In this paper we have introduced a general method of finding XOR/modular differential paths to find a chosen plaintext distinguisher of Phelix, but this stream cipher seems to be tolerant against such a distinguisher. So, we introduced a chosen key differential distinguishing attack which can distinguish the cryptosystem with no complexity if the attacker has the ability to re-encrypt the previous message with a typical modification of the old key. An open problem is whether there is a systematic method based on some criteria to find the best differential path or paths in these scenarios, so that we can modify our chosen plaintext distinguishing attack.

6 References

1. Ferguson N., Whiting D., Schneier B., Kelsey J., Lucks S., and Kohno T., "*Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive*", FSE 2003, pp. 330–346.
2. Muller F., "*Differential Attacks against the Helix Stream Cipher*", FSE 2004, pp. 94–108.
3. Paul S., and Preneel B., "*Solving Systems of Differential Equations of Addition*", ACISP 2005, pp. 75-88.

4. Paul S., and Preneel B., "*Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries*", Indocrypt 2005, pp. 90-103.
5. Whiting D., Schneier B., Lucks S., and Muller F., "*Phelix: Fast Encryption and Authentication in a Single Cryptographic Primitive*", eSTREAM, ECRYPT Stream Cipher Project Report 2005/027, see: http://www.schneier.com/paper-phelix.html.