ElsieFour (LC4)

Author: Anna Lena Rotthaler

Sep 29, 2017

Contents

1 Introduction

ElsieFour (LC4) is a low-tech cipher that can be computed by hand. It is intended for encrypted communication between humans and therefore, it encrypts and decrypts plaintexts and ciphertexts consisting of the English letters plus a few other characters.
LC4 is an amalgam of ideas from the classic RC4 stream cipher, the historical Playfair cipher, and the notion of plaintext-dependent keystreams. It is based on a state that is continually updated as encryption progresses. The state is a permutation of the integers 0 to 35 in a 6x6 matrix.
In encryption a plaintext character becomes a ciphertext character and vice versa in decryption. Plaintext and ciphertext work on the same alphabet.
This document describes the encryption and decryption process. Most parts are taken from [1], where you can find further information and examples.

2 Alphabet

LC4 encrypts and decrypts plaintexts and ciphertexts consisting of the following 36 characters (case insensitive):
_ 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o p q r s t u v w x y z
Internally, the encryption and decryption algorithms treat the characters # to z as integers 0 to 35, respectively. The character # is used, rather than the digit 0, to avoid confusion with the uppercase letter O. The character _ is used, rather than the digit 1, to avoid confusion with the uppercase letter I and the lowercase letter l. The # and _ characters can be used as separators in the plaintexts.

3 Nonce

LC4 uses a sequence of six or more characters as a nonce that should be unique for different plaintexts. It can be chosen at random from the alphabet. In the encryption process the nonce is placed at the beginning of the plaintext P (see 7 for further information and example). The unencrypted nonce is sent to the recipient, but the encrypted characters of the nonce are not sent with the ciphertext.

4 Signature

LC4 provides authenticated encryption. Therefore, a signature is chosen that should consist of ten or more characters from the alphabet that uniquely identifies the sender. The signature is part of the plaintext P (see 7 for further information and example).

5 Key

The LC4 key is a permutation of the integers 0 to 35 chosen randomly from the set of all such permutations. The size of the key space is 36!, equivalent to a 138-bit key. The key can be represented as a string such as:
_ie497g5oyhwqnazb3xsltcvfprd82u#6jmk
The key is required to be a random permutation. If the key is based on a keyword, such as goldameir..., key recovery attacks can be sped up considerably by guessing dictionary words for the initial portion of the key.

6 State

The LC4 state consists of a 6x6 matrix S containing a permutation of the integers 0 to 35, as well as two indexes i and j. The initial state is created from the key. These indexes constitute a "marker" that designates a certain matrix element S[i][j] (the first index is the row, the second index is the column). The state initialization algorithm is as follows. The "/" operation is truncating integer division. The "mod 6" operation returns a remainder in the range 0 to 5. The initialization algorithm does the same as entering the key line-by-line into the 6x6 state matrix.

Input: key array K
Output: state matrix S; indexes i and j
Algorithm:
    For k = 0 to 35:
        S[k/6][k mod 6] = K[k]
    i = 0
    j = 0

Example
Key: hxo_tp#a3jdnmq2glf75kw469eyrzbvci8us

State: $S = \begin{pmatrix} h & x & o & \_ & t & p \\ \# & a & 3 & j & d & n \\ m & q & 2 & g & l & f \\ 7 & 5 & k & w & 4 & 6 \\ 9 & e & y & r & z & b \\ v & c & i & 8 & u & s \end{pmatrix}$

The state can also be written in one line with spaces between the rows to save space:
hxo_tp #a3jdn mq2glf 75kw46 9eyrzb vci8us

7 Plaintext P

The plaintext P consists of the concatenation of the nonce, the plaintext message and the signature.
P = nonce||message||signature

Example
Nonce: 9kqhuo
Plaintext message: its_my_fathers_son_but_not_my_brother
Signature: #its_me
Plaintext P: 9kqhuoits_my_fathers_son_but_not_my_brother#its_me

8 Encryption

Choose a plaintext message, a random nonce, a signature and a key. The initial state is derived from the key (see 6). Build the concatenation of nonce, plaintext message and signature (see 7).
P = nonce||message||signature
The plaintext characters are treated as integers (see 2).
Perform the following encryption algorithm and discard the ciphertext of the nonce characters. Send the unencrypted nonce followed by the remaining ciphertext to the recipient.

Input: state matrix S; indexes i, j; the sequence of plaintext characters P
Output: the sequence of ciphertext characters C
Algorithm:
For each character in P:
r = row of S in which P appears ($0 \leq r \leq 5$)
c = column of S in which P appears ($0 \leq c \leq 5$)
x = (r + (S[i][j]/6)) mod 6
y = (c + (S[i][j] mod 6)) mod 6
C = S[x][y]
Right-rotate row r of S
Down-rotate column y of S
i = (i + (c/6)) mod 6
j = (j + (c mod 6)) mod 6

Computing LC4 by hand, the encryption algorithm can be simplified by using a method explained in chapter 6 of [1].

Example
Plaintext message: its_my_fathers_son_but_not_my_brother
Key: hxo_tp#a3jdnmq2glf75kw469eyrzbvci8us
Nonce: 9kqhuo
Signature: #its_me
Plaintext P: 9kqhuoits_my_fathers_son_but_not_my_brother#its_me
Initial state S: hxo_tp #a3jdn mq2glf 75kw46 9eyrzb vci8us
Ciphertext of the nonce: p69af9
Remaining ciphertext: sdzyj54mwaibwzr9gd_79ogy789357fqv5ks_o2pxyqs
Send to the recipient: 9kqhuosdzyj54mwaibwzr9gd_79ogy789357fqv5ks_o2pxyqs

9 Decryption

You received an unencrypted nonce and a sequence of ciphertext characters C.
C = nonce||Shortened(E(P)) with Shortened(E(P)) = E(P) without E(nonce)
You know the key you need to derive the initial state.
First encrypt the nonce using the LC4 encryption algorithm. Then decrypt the received ciphertext using the following decryption algorithm and the state after encrypting the nonce. You get the concatenation of the plaintext and the signature. Verify the decrypted signature.

Input: state matrix S; indexes i, j; the sequence of ciphertext characters C
Output: the sequence of plaintext characters P
Algorithm:
For each ciphertext character C:
x = row of S in which C appears ($0 \leq x \leq 5$)
y = column of S in which C appears ($0 \leq y \leq 5$)
r = (x - (S[i][j]/6)) mod 6
c = (y - (S[i][j] mod 6)) mod 6
P = S[r][c]
Right-rotate row r of S
Down-rotate column y of S
i = (i + (C/6)) mod 6
j = (j + (C mod 6)) mod 6

Example
Received message: 9kqhuosdzyj54mwaibwzr9gd_79ogy789357fqv5ks_o2pxyqs
Unencrypted nonce: 9kqhuo
Remaining ciphertext: sdzyj54mwaibwzr9gd_79ogy789357fqv5ks_o2pxyqs
Key: hxo_tp#a3jdnmq2glf75kw469eyrzbvci8us
Plaintext characters: its_my_fathers_son_but_not_my_brother#its_me
Plaintext message: its_my_fathers_son_but_not_my_brother
Signature: #its_me

10 About LC4

LC4 is an amalgam of ideas from the classic RC4 stream cipher, the historical Playfair cipher, and the notion of plaintext-dependent keystreams.
Like RC4, LC4 is based on a state that is continually updated as encryption progresses. The state is a permutation of integers: 0 to 35 for LC4 and 0 to 255 for RC4.
The state of LC4 is a 6x6 matrix, similar to the matrix Playfair uses.
LC4 can be viewed as having a plaintext-dependent keystream, because the state initially derived from the key is plaintext-dependent during encryption and therefore, authenticated encryption is possible.

References

[1] Kaminsky, A. (2017). ElsieFour: A Low-Tech Authenticated Encryption Algorithm For Human-to-Human Communication
Available at: `https://eprint.iacr.org/2017/339.pdf`