Secure and Dependable Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO21-320203
Date: 2019-04-04
Due: 2019-04-25

SADS 2019 Problem Sheet #4

Problem 4.1: *substitution-permutation network* (7+1+1+1 = 10 points)

We define a substitution-permutation network implementing an 8-bit block cipher with keys of a length of 32 bits. We call this cipher *sads crypt*, or short *scrypt*.

The substitution step uses 4-bit S-boxes applied to the lower and upper 4 bits of an 8-bit word. The substitution $S : \{0,1\}^4 \to \{0,1\}^4$ is given by $x \mapsto ((x+1) \cdot 7) \bmod (17-1)$. This is a bijection of $\{0,1\}^4$, where 4-bit chunks are seen as natural numbers via their binary encoding.

The permutation step uses an 8-bit P-box $P : \{0,1\}^8 \to \{0,1\}^8$, which does a cyclic 2-bit left-shift of its argument.

The substitution-permutation network uses the following rounds:

- Round 0: Key step with the first (most significant) 8 bits of the key.
- Round 1: Substitution step followed by a permutation step followed by a key step with the next 8 bits of the key.
- Round 2: Substitution step followed by a permutation step followed by a key step with the next 8 bits of the key.
- Round 3: Substitution step followed by a key step with the last (least significant) 8 bits of the key.

a) Write a file `scrypt.c` implementing the public interface defined by `scrypt.h`. We provide you with unit tests so that you can check your implementation. Consider implementing the S-boxes and P-boxes as internal helper functions.

b) Encrypt the cleartext “secret” (0x736563726574) in electronic codebook mode with the key 0x98267351.

c) Encrypt the cleartext “hacker” (0x66b6bbe90e21) in cipher block chaining mode with the key 0x98267351 and the initialization vector 0x42.

d) Decrypt the ciphertext 0xc65e05946b86eb2e33f58fdaff0f42, which has been produced using cipher block chaining mode with the key 0x98267351 and the initialization vector 0x42.

Below is the `scrypt.h` header file defining the public interface. To answer the questions b)-d), you may want to implement a small main program that allows you to play with your *scrypt* implementation.

```c
/*
 * scrypt/src/scrypt.h --
 */

#ifndef _SCRYPT_H
#define _SCRYPT_H

#include <stdint.h>

/**
 * \brief Encrypt an 8-bit cleartext using a 32-bit key.
 *
 * \param m 8-bit cleartext.
 * \param k 32-bit key.
 * \result 8-bit ciphertext.
 */

uint8_t
sc_enc8(uint8_t m, uint32_t k);

/**
 * \brief Decrypt an 8-bit ciphertext using a 32-bit key.
 *
 * \param m 8-bit ciphertext.
 * \param k 32-bit key.
 * \result 8-bit cleartext.
 */

uint8_t
sc_dec8(uint8_t c, uint32_t k);

/**
 * \brief Encrypt a variable-length cleartext using a 32-bit key in ECB mode.
 *
 * \param m cleartext.
 * \param c ciphertext.
 * \param len length of the cleartext and ciphertext buffers.
 * \param k 32-bit key.
 */

void
sc_enc_ecb(unsigned char *m, unsigned char *c, size_t len, uint32_t k);

/**
 * \brief Decrypt variable-length ciphertext using a 32-bit key in ECB mode.
 *
 * \param c ciphertext.
 * \param m cleartext.
 * \param len length of the ciphertext and cleartext buffers.
 * \param k 32-bit key.
 */

void
sc_dec_ecb(unsigned char *c, unsigned char *m, size_t len, uint32_t k);

/**
 * \brief Encrypt a variable-length cleartext using a 32-bit key in CBC mode.
 *
 * \param m cleartext.
 * \param c ciphertext.
 * \param len length of the cleartext and ciphertext buffers.
 * \param k 32-bit key.
 * \param iv 8-bit initialization vector.
```

```
 */

void
sc_enc_cbc(unsigned char *m, unsigned char *c, size_t len,
           uint32_t k, uint8_t iv);

/**
 * \brief Decrypt variable-length ciphertext using a 32-bit key in CBC mode.
 *
 * \param m ciphertext.
 * \param m cleartext.
 * \param len length of the ciphertext.
 * \param k 32-bit key.
 * \param iv 8-bit initialization vector.
 */

void
sc_dec_cbc(unsigned char *c, unsigned char *m, size_t len,
           uint32_t k, uint8_t iv);

#endif
```