Cryptography, exercise sheet 5 for 05 Oct 2021

We will work through these exercises on wonder.me; you can find the URL for the room in the Canvas announcement and in Zulip chat under the topic ”Wonder session” in the “general” stream. These are exercises to challenge your understanding of the lectures you have watched already, in particular lectures I – VII about symmetric-key cryptography.
These are not for homework.

You can call one of us over by choosing “invite to circle”. Please note, though, that if we are in another circle busy talking we will not come right away and invitations expire quickly. So pick one of us who is in the TA corner, thus not occupied.

1. The least significant bits of x and y in LeftTEA start as b_0 and b_{32} respectively. Compute the values of the LSBs after round 1, round 2, round 3, ..., round 32 in terms of the bits of the inputs, the key parts, and the constant.
   Hint: This function repeats, so you do not need to write out 32 steps. You should see some pattern after 6 steps.

2. TEA4 has only 4 rounds. Consider inputs (x, y) and $(x+2^{31}, y)$ and trace their difference through all 4 rounds.
   Hint: The slides already show what you should see, but calculate it yourself.

3. This exercise uses the example version of the Wegman-Carter message authentication code with $p = 1000003$.

   To authenticate the i-th ciphertext c_i the sender expresses c_i in base 10^6 as $c_i = c_{i,0} + c_{i,1}10^6 + c_{i,2}10^{12} + \cdots + c_{i,k}10^{6k}$ and computes the authenticator as

   $$t_i = (c_{i,0}r^{k+1} + c_{i,1}r^k + c_{i,2}r^{k-1} + \cdots + c_{i,k}r \bmod p) + s_i \bmod 1000000.$$

   For simplicity we will do $i = 1$ and omit the extra indices. Compute the authenticator for $c = 45435654243597928347592843 7, r = 483754, s = 342534$.

4. Majordomo is a program that manages Internet mailing lists. If you send a message to `majordomo@foodplus.com` saying `subscribe recipes`, Majordomo will add you to the `recipes` mailing list, and you will receive several interesting recipes by e-mail every day.

   It is easy to forge mail. You can subscribe a victim, let’s say `God@heaven.af.mil`, to the `recipes` mailing list, and thousands more mailing lists, by sending fake subscription requests to Majordomo. `God@heaven.af.mil` will then be flooded with mail.

   Majordomo 1.94, released in October 1996, attempts to protect subscribers as follows. After it receives your subscription request, it sends you a confirmation number. To complete your subscription, you must send a second request containing the confirmation number.

   Majordomo 1.94 generates confirmation numbers as follows. There is a function h that changes strings to numbers. The `recipes` mailing list has a secret string k. The confirmation number for an address a is $h(ka)$. For example, if the secret string is `ossifrage`, and the address is `God@heaven.af.mil`, the confirmation number is $h($`ossifrageGod@heaven.af.mil`$)$.

   The function h produces a 32-bit result. Each letter is naturally represented in a computer as a byte, i.e., an integer in $[0, 255]$. The string is read from left to right. In the following “rotate left 4 bits” turns $(b_{31}, b_{30}, \ldots, b_1, b_0)$ into $(b_{27}, b_{26}, \ldots, b_1, b_0, b_{31}, b_{30}, b_{29}, b_{28})$.

The function h is computed as follows. Start with 0. Add the first byte of the string. Rotate left 4 bits. Add the next byte of the string. Rotate left 4 bits. Continue adding and rotating until the end of the string.

Explain how to subscribe `God@heaven.af.mil` to the `recipes` mailing list despite this protection, and explain what Majordomo 1.94 should have done.

5. In 2016 a bug was found in Signal for Android which meant that in some cases the MAC was over a shorter part of the message, allowing an attacker to append data to a message. More specifically, this bug applied to attachments and came from an error in the code taking a 64-bit value for a 32-bit one. The part that makes this relevant for 2MMC10 is that the implementation used AES in CBC mode. Please read `https://pwnaccelerator.github.io/2016/signal-part2.html`.