Cryptography, exercise sheet 7 for 17 Oct 2023

1. Show that ElGamal encryption is re-randomizable, i.e., show that (r, c) and $(rg^{k'}, mh_A^{k'})$ decrypt to the same message for any k'. (We have covered this in class last Thu).

2. Show that ElGamal encryption is homomorphic, i.e., find some way to compbine ciphertexts (r_1, c_1) encrypting m_1 and (r_2, c_2) encrypting m_2 (both encrypted to public key h_A) so that the resulting ciphertext is an encryption of m_1m_2.
Note: This only involves the public values, no decryption.

3. Alice and Bob use ElGamal encryption. Eve learns that Bob's random-number generator is broken (details below) and she learns the decryption m_1 of (r_1, c_1).
   (a) Assume that Bob uses the same nonce k for all encryptions. Show how Eve can decrypt (r_2, c_2).
   (b) Assume that Bob increments his k for each encryption, i.e., that $k_{i+1} = k_1 + i$. Show how Eve can decrypt (r_i, c_i).

4. 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 $c = 454356542435979283475928437, r = 483754, s = 342534.$

5. The proper definition of Wegman–Carter MAC puts

$$t_i = \left(\sum_{j=1}^{k} c_{i,j}r^{k+1-j} \bmod p\right) + s_i \bmod 2^n$$

   for c_i a ciphertext of kn bits and $p > 2^n$ a prime.

   Show that it is important that the powers of r start at r^1 rather than at r^0, i.e., show how an outside attacker who does not have access to r or any of the s_i but sees some (c_i, t_i, i) can compute some valid (c', t', i) on a new ciphertext $c' \neq c_i$ if instead the definition is

$$t' = \left(\sum_{j=1}^{k} c_{j,}r^{k-j} \bmod p\right) + s_i \bmod 2^n.$$

6. 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.

7. Show how to retrieve the message m in RSA-OAEP from $M = (s, t)$. (See RSA I for the definition of RSA-OAEP.) This is just considering the encoding and decoding of the message and skips the RSA part. The functions G and H are cryptographic hash functions, so you cannot invert them.

8. To do after Thursday's lecture: 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.