Cryptography, homework sheet 5

Due for 2MMC10: 10 October 2019, 10:45

and for Mastermath: 14 November 2019, 10:45 by email to `crypto.course@tue.nl`

Team up in groups of two or three to hand in your homework. We do not have capacity to correct all homeworks individually. Do not email Tanja your homework or put homework in mailboxes.

You may use computer algebra systems such as mathematica, gp, or sage or program in C, Java, or Python. Please submit your code if any) as part of your homework. If you do, make sure that your programs compile and run correctly; my students will not debug your programs. The program should also be humanly readable.

1. Use the schoolbook version of Pollard's rho method to attack the discrete logarithm problem given by $g = 3, h = 245$ in $\mathbb{F}_{1013}^*$, i.e. find an integer $0 < a < 1012$ such that $h = g^a$, using the t_i and r_i (the twice as fast walk) as defined in class (aand repeated here). Let $t_0 = g, a_0 = 1$, and $b_0 = 0$ and define

$$t_{i+1} = \begin{cases} t_i \cdot g \\ t_i \cdot h \\ t_i^2 \end{cases}, a_{i+1} = \begin{cases} a_i + 1 \\ a_i \\ 2a_i \end{cases}, b_{i+1} = \begin{cases} b_i \\ b_i + 1 \\ 2b_i \end{cases} \text{ for } t_i \equiv \begin{cases} 0 \bmod 3 \\ 1 \bmod 3 \\ 2 \bmod 3 \end{cases},$$

   where one takes t_i as an integer. The twice as fast walk has $r_i = t_{2i}$.

   The walk could start at any $t_0 = g^{a_0} h^{b_0}$ for known a_0 and b_0 – but then the homework would be harder to correct.

2. 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, computed as follows. Start with 0. Add the first byte of the string, starting from the left end 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.

3. Here is a toy version of a Wegman-Carter message authentication code with which A and B can authenticate t messages: Fix $p = 1000003$. A and B randomly generate $r \in \mathbb{F}_p^*$ and randomly pick integers $s_1, s_2, \ldots, s_t \in \{0, 1, 2, \ldots 999999\}$. These values are the shared secrets; r is the overall secret and the s_i are per message secrets.

   To authenticate the i-th message m_i the sender expresses m_i in base 10^6 as $m_i = m_{i,0} + m_{i,1}10^6 + m_{i,2}10^{12} + ... + m_{i,n}10^{6n}$ and computes the authenticator as

   $$\text{auth}(m_i) = (m_{i,0}r + m_{i,1}r^2 + m_{i,2}r^3 + ... + m_{i,n}r^{n+1} \bmod p) + s_i \bmod 1000000.$$

   For simplicity we will do $i = 1$ and omit the extra indices. Compute the authenticator for $m = 454356542435979283475928437, r = 483754, s = 342534$.

4. The above (and what was shown in the lecture) are correct examples of Wegman–Carter MACs. A more general set up specifies a prime p and 2^k a power of 2 less than p. (Computers prefer binary representation over decimal). Messages have nk bits. A and B agree on $r \in \mathbb{F}_p^*$ and $s_1, s_2, \ldots, s_t < 2^k$. To authenticate message $m_i = (m_{i1}, m_{i2}, \ldots, m_{in})$, with $0 \leq m_{ij} < 2^k$, the sender computes $\text{auth}(m_i) = (\sum_{j=1}^{n} m_{ij}r^j \bmod p) + s_i \bmod 2^k$ and sends $(m_i, \text{auth}(m_i), i)$.

   Show that it is important that the powers of r starts 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 $(m_i, \text{auth}'(m_i), i)$ can compute some valid $(m, \text{auth}'(m), i)$ on a new message $m \neq m_i$ for $\text{auth}'(m) = (\sum_{j=0}^{n-1} m_{j+1}r^j \bmod p) + s_i \bmod 2^k$.