Cryptography, homework sheet 5

Due for 2MMC10: 08 October 2020, 10:45
and for Mastermath: 12 November 2020, 10:45

Please submit your homework by email to `crypto.course@tue.nl` and put your team mates in cc to that email. Also mention their names and student numbers in the *body of the email* (yes, neatly typed so that we can copy it easily).

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.
Make sure to justify your answers. You do not need to document intermediate results of exponentiations or modular inversion. However, you should include all intermediate results of the algorithms that constitute a step.
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; the TAs will not debug your programs. The program should also be humanly readable. Do not include the code in the pdf of your homework submission but submit it as a separate file.

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 Floyd's cylce finding method. The step function is as defined in class (and repeated here).

   Define the update function as

$$x \leftarrow \begin{cases} x \cdot g \\ x \cdot h, \\ x^2 \end{cases} \quad b \leftarrow \begin{cases} b+1 \\ b, \\ 2b \end{cases} \quad c \leftarrow \begin{cases} c \\ c+1, \\ 2c \end{cases} \quad \text{for } x \equiv \begin{cases} 0 \bmod 3 \\ 1 \bmod 3. \\ 2 \bmod 3 \end{cases}$$

   Start the fast walk at $x_f = g, b_f = 1$, and $c_f = 0$ and the slow walk at $x_s = g, b_s = 1$, and $c_s = 0$. The fast walk does two steps in one iteration while the slow walk does one. You may compare the results only after an iteration is complete.

2. Use factor base $\mathcal{F} = \{2, 3, 5, 7, 11, 13\}$ to solve the DLP $h = 281$, $g = 2$, in $\mathbb{F}_{1019}^*$. I.e. pick random powers of $g = 2$, check whether they factor into products of powers of 2,3,5,7,11, and 13; if so, add a relation to a matrix. The columns of the matrix correspond to the discrete logs of 2,3, 5,7,11, and 13. Once you have 6 rows try to solve the matrix; note that these computations take place modulo the group order 1018. It might be that some of the rows are linearly dependent, in that case you need to generate another relation. Once you have all discrete logs of the primes in the factor base, check whether h is smooth and if not find a hg^i (for some i) which is smooth. You only need to document the successful choices of i or submit a working program that has comments.

   Here are two examples. Let $a_j = \log_g j$. $2^{291} \equiv 52 \bmod 1019$; over the integers $52 = 2^2 \cdot 13$, so we incluclude the relation $291 \equiv 2a_2 + a_{13} \bmod 1018$. Note that you can run into difficulties inverting modulo 1018 since it is not prime. E.g. $2^{658} \equiv 729 \bmod 1019$; over the integers $729 = 3^6$, so we incluclude the relation $658 \equiv 6a_3 \bmod 1018$ but 6 is not invertible modulo 1018 and we can only determine $a_3 \equiv 449 \bmod 509$ and need to test whether $a_3 = 449$ or $a_3 = 449 + 509$. Here $2^{449} \equiv 1016 \bmod 1019$ and $2^{449+509} \equiv 3 \bmod 1019$, thus $a_3 = 958$. [Now you only need 5 more.]

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 $\text{auth}'(m) = (\sum_{j=0}^{n-1} m_{j+1}r^j \bmod p) + s_i \bmod 2^k$.