Security

Assignment 12, Friday, December 15, 2017

Deadline: Monday, January 8, 09:00 sharp!

Goals: After completing these exercises successfully you should be able to

- be able to perform necessary computations of a Diffie–Hellman key exchange;
- recognise the main weakness of the Diffie–Hellman key exchange;
- be able to perform necessary computations for ElGamal encryption/decryption;
- understand the problem with reusing primes.

Marks: You can score a total of 100 points.

1. **(32 points)** The Diffie–Hellman (DH) key exchange is used to agree on a secret key between Alice and Bob. The prime $p = 1021$ determines the group $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$ in which all operations are performed (i.e. all computations are performed modulo 1021).
   The following messages are exchanged:

   $$\begin{array}{lll} 1. & A \longrightarrow B & : \quad p = 1021, g = 10, g^a = 93 \\ 2. & B \longrightarrow A & : \quad g^b = 491 \end{array}$$

   (a) Given Alice's secret $a = 317$, compute the shared secret key. Show how you came to the solution.

   (b) Since the modulus is very small, one can compute the secret values. Derive Bob's secret from the exchanged messages. Feel free to use a calculator[1] or you can write a small progam. In any case, explain your steps.

   (c) Check that Bob has the same (shared) key as Alice using the private key from (b) (by doing the DH-computation for Bob's side).

   (d) We describe a modified communication when there is a middle-man. Assume that message 1. $A \to B$ is as showed above, but Eve captures the message and picks two random values: $r_A = 37, r_B = 404$. She uses these random values for the communication with Alice and Bob, respectively.

      i. Show the *four messages*: $A \to E(B), E(A) \to B, B \to E(A), E(B) \to A$. Use the protocol notation as used earlier in this course.

      ii. Compute the *established keys* K_{AE}, K_{BE} between Alice and Eve, and between Eve and Bob, respectively.

2. **(48 points)** Consider the ElGamal public-key encryption system. For $p = 31$, $G = \mathbb{Z}_p^*$ is a multiplicative cyclic group with generator $g = 3$. Suppose that the secret number in the system is $a = 17$. You will encrypt messages and decrypt ciphertexts in this group. Describe your computations.

   (a) Determine the corresponding value $A = g^a \in G$.

   (b) We are going to encrypt the message "remember" (in ECB mode) using ElGamal. To map letters to integers we use the mapping $a \mapsto 1$, $b \mapsto 2$,$\ldots$,$z \mapsto 26$. For the following steps, fill in each row in the table below, and explain the required computations:

      i. For each integer block, calculate a separate ephemeral public key A^r using the following values for r: 3, 6, 9, 12, 15, 18, 21 and 24.

[1] e.g. `https://www.wolframalpha.com`

ii. For each integer block, calculate the first component $c_1 = R = g^r$ of the ciphertext using that same sequence for r.

iii. Finally, for each integer block, calculate the second component $c_2 = m \cdot A^r$ of the ciphertext.

(c) Let's now decrypt the ciphertext; complete the table below

i. For each integer block calculate the inverse of the ephemeral public key $(A^r)^{-1} = {c_1}^{-a}$. (*Note*: ${c_1}^{-a}$ can be calculated as ${c_1}^{p-1-a}$, using Euler's Theorem and the fact that $\varphi(p) = p - 1$).

ii. For each integer block, use the inverse $(A^r)^{-1}$ to cancel out A^r in c_2 and thus retrieve $m = c_2 \cdot A^{-r}$.

	r	**e**	**m**	**e**	**m**	**b**	**e**	**r**
Encryption								
Mapping	18	5	.	.	.	.	.	.
r	3	6	9	12	15	18	21	24
A^r	.	.	.	.	.	.	.	.
$c_1 = g^r$	.	.	.	.	.	.	.	.
$c_2 = m \cdot A^r$	.	.	.	.	.	.	.	.
Decryption of ciphertext (c_1, c_2)								
$(A^r)^{-1} = {c_1}^{-a}$	.	.	.	.	.	.	.	.
$m = c_2 \cdot A^{-r}$	.	.	.	.	.	.	.	.

3. **(20 points)** It is generally hard to factor integers; this is why RSA is secure. It is however easy to find *common factors* of two integers. This was used in independent research by two groups in 2012 to factor various SSL, SSH and GnuPG keys [2][3], and was used as a starting point to find various private keys of Taiwanese citizen smartcards[4].

   For this exercise, it is useful to be able to compute the GCD of two large values in some automated way. You can do this using your favorite programming language[5], but we have also set up an online page to make this more convenient, at the following URL. Note that many other online GCD calculators do not support integers of this size.

   - `http://www.sos.cs.ru.nl/applications/courses/security2017/gcd/`

   Use the above ideas to attempt to factor the RSA moduli (i.e. find the p and q such that $p \cdot q = N$) listed below. Note that some of the moduli N may not share factors with any of the others, so not all of them can be factored using this method.

   - 595581987651106688365284842778515858399666547859870373300567
   - 732521324063413291774595255009269986704084399047286433357607
   - 697998237255232517803133139640937207091669333334886072165381
   - 665759389457622825753076124570026166878147870317677657070179
   - 176294427788887166758409622538881387638478405478915857712513
   - 592339248856319601455928821705423109007342115448431777433343

[2] `https://factorable.net/`
[3] `http://eprint.iacr.org/2012/064`
[4] `http://cr.yp.to/papers.html#smartfacts`
[5] See for example the `gcd` function in Python's `fractions` module