Cryptography, exercise sheet 7 for 19 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.

You can use Sage or other computer-algebra systems for the computations but do not use it to solve the exercises by brute force; use the algorithms you learned.

This sheet might be too much for 90 min; you will encounter the last exercise again when you check out old exams, this is from the first exam 2016.

1. You learn that I sent ciphertext
   $c = 1468256278693980617525887783092320419596710415981586 22$ to a user with RSA public key $(e, n) = (3, 52977421076224667516131861674699561783556524625163 5147)$ and that this was the result of a form which sends a stereotyped message `myfavoritenumberis____` in base 36, where the empty spaces indicate 6 unknown characters. Use LLL to recover those 6 characters.
   Note that you are not guaranteed to succeed with the first output of LLL. Also note that you can (and should) check your solution.

2. Show that ElGamal encryption is re-randomizable, i.e., show that (r, C) and $(rg^{k'}, Ch_A^{k'})$ decrypt to the same message for any k'.

3. Show that ElGamal encryption is homomorphic.

4. 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).

5. $13 \in \mathbb{F}_{1321}^*$ generates a group of order $1320 = 2^3 \cdot 3 \cdot 5 \cdot 11$. Solve the discrete logarithm problem $g = 13, h = 320$ by using the Pohlig-Hellman attack, i.e. find an integer $0 < a < 1320$ such that $h = g^a$ by computing first a modulo $2, 4, 8, 3, 5$, and 11 and then computing a using the Chinese Remainder Theorem.

6. 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 include 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 include 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.]

7. Show that verification succeeds for honestly generated signatures in DSA.

8. This exercise is about Diffie-Hellman (DH) key exchange in finite fields. As we have seen in class, index calculus attacks on the DLP in $\mathbb{F}_p^*$ are faster than Pollard's rho attack, so implementations use smaller subgroups or limit the exponent.

   DSA typically specifies three parameters (p, ℓ, g): p is the modulus, meaning that the group $\mathbb{F}_p^*$ is used, and ℓ is the order of the subgroup generated by g. These groups can also be used in DH applications. The implementer is expected to use g as the generator and to choose secret keys smaller than ℓ.

   (a) Alice sends Bob a request to use her preferred parameter set $(234917, 281, 19452)$. However, Bob's library expects only two arguments and reads Alice's parameters as $p = 234917$ and $g = 281$. Bob uses a secret $b < 400$ and sends his DH share $h_b = g^b = 92646$. Compute Bob's b without resorting to a brute force attack.
   Verify your solution.
   Hint: You might find the factorization of $p - 1 = 2^2 \cdot 11 \cdot 19 \cdot 281$ useful. Note that this g has order $117458 = 2 \cdot 11 \cdot 19 \cdot 281$.
   Hint 2: You know everything to run this attack.

   (b) Alice sends Bob a request to use her preferred parameter set $(234977, 1049, 202367)$. However, Bob's library expects only two arguments and reads Alice's parameters as $p = 234977$ and $g = 1049$. Bob uses a secret $b < 400$ and sends his DH share $h_b = g^b = 7409$. Compute Bob's b without resorting to a brute force attack.
   Verify your solution.
   Hint: You might find the factorization of $p - 1 = 2^5 \cdot 7 \cdot 1049$ useful. Note that this g has order $117488 = 2^4 \cdot 7 \cdot 1049$.

   (c) Eve knows that Charlie's server uses group 23 from RFC 5114, i.e., a 2048-bit prime p to be used with a subgroup of prime order ℓ, where ℓ has 224-bits. The factorization of $(p-1)/\ell = 2 \cdot 3^2 \cdot 5 \cdot 43 \cdot 73 \cdot 157 \cdot 387493 \cdot 605921 \cdot 742327609 \cdot 5213881177 \cdot 112486462861 \cdot 3528910760717 \cdot C489$, where $C489$ is the product of 3 larger primes. Charlie's server uses static DH, that means the same value of c for all connections. Eve can easily see this by Charlie offering the same $h_c = g^c$ for all connections. This means, that Eve can send Charlie input values $h = g^e$ and Charlie will reply with an AES encryption of `ACKNOWLEDGE` under key $\text{hash}(h^c)$.
   Furthermore, Eve knows what software Charlie's server uses and she knows that it does not verify the order of the input values it receives from users.
   Describe an attack with which Eve can compute Charlie's secret c in time less than 2^{64}. State the number of queries, i.e. the number of values h_i that Eve sends, and

describe how she should choose these values h_i? How much computation does Eve need to do? Note that Charlie's c has 224 bits.