Problem sheet 11 for Course 02231, 2025

These practice problems have the purpose of helping you understand the material better and learning the skills that are necessary to analyze cryptographic constructions, and sometimes to prepare you for the next class. All answers should be supported by a written justification. To gauge whether a justification is sufficient, ask yourself if your peers would be convinced by it without additional explanations.

Exercise 1. (An elliptic curve group)

Let p be a prime and $a, b \in \mathbb{Z}_p$. Define the set

$$G = \{(x, y) | x, y \in \mathbb{Z}_p, y^2 = x^3 + a \cdot x + b\} \cup \{O\}.$$

O is sometimes called "the point at infinity". We define the group law as follows. For all $P, Q \in G$, $P = (x_1, y_1)$, $Q = (x_2, y_2)$,

- $O + P = P + O = P$.
- $-P = (x, -y)$
- $P - P = O$
- If $Q \neq -P$, set

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2 + a}{2y_1} & \text{if } x_1 = x_2 \text{ and } y_1 \neq 0. \end{cases}$$

  and define $R = P + Q$, $R = (x_3, y_3)$ with $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = (x_1 - x_3)\lambda - y_1$.

Note: Here all computations are mod p, so $\frac{s}{t} = s \cdot t^{-1}$ where t^{-1} is the inverse of t mod p.

1. For $p = 11$, $a = 2$, $b = 7$, compute $P = (6, 2) + (7, 1)$, $Q = -(7, 1)$ and $R = P + Q$.
2. (Bonus exercise) Show that the defined group addition is commutative, i.e. show that for all $P, Q \in G$ it holds that $P + Q = Q + P$.

Exercise 2. (X3DH)

Consider the X3DH protocol as introduced in the lecture, without the optional 4th Diffie-Hellman exchange which uses $k_4 = DH(EK_A, EK_B)$.

1. For each of the three intermediate keys k_i, $i = 1, 2, 3$, describe what security property is lost if key k_i is omitted from the final key derivation. Example for $i = 1$: in what way is the protocol insecure if we compute the final key as $k = H(k_2, k_3)$?

Exercise 3. (**Schnorr signatures**)

In the lecture we learned about RSA-FDH signatures. There are also signature schemes which rely on the DLOG problem, one of which is called Schnorr's signature scheme.
As before, let p be a prime and $g \in \mathbb{Z}_p$ of order q. Furthermore, let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a collision-resistant hash function. The Schnorr signature scheme consists of the following 3 algorithms:

$Keygen$

1. Sample $s \leftarrow \mathbb{Z}_q$ uniformly at random.
2. Compute $S = g^s \bmod p$.
3. Output $(pk, sk) = (S, s)$.

$Sign$ on input $m \in \{0,1\}^*$ and sk:

- Sample $a \leftarrow \mathbb{Z}_q$ uniformly at random.
- Compute $A = g^a \bmod p$.
- Compute $e = H(A, m)$ and $z = a + e \cdot s \bmod q$.
- Output $\sigma = (A, z)$.

$Verify$ on input $pk = S$, $m \in \{0,1\}^*$ and $\sigma = (A, z)$:

- Compute $e = H(A, m)$.
- Output 1 if $A \cdot S^e = g^z \bmod p$, otherwise output 0.

1. Verify that the Schnorr signature scheme is correct. For that, imagine that pk, sk is generated correctly and σ is a correctly generated signature on m. Then $Verify(pk, m, \sigma) = 1$ should always hold.

2. Assume that there exists an algorithm $\mathcal{A}$ which, on input $g, A \in \mathbb{Z}_p$ outputs $a \in \mathbb{Z}_q$ such that $g^a = A \bmod p$. How can you use $\mathcal{A}$ on a Schnorr public key pk to generate signatures on arbitrary messages that are valid for pk?

Exercise 4. (**DDH and key reuse**)

For both the El Gamal encryption scheme and for X3DH, it is important that Diffie-Hellman remains secure even if one of the private exponents has been used before. For $g \in \mathbb{Z}_p$ of order q, consider therefore the following variant of the DDH problem:
DDH': *Sample* $a, b, c, d, e \in \mathbb{Z}_q$ *uniformly random and independent. The attacker must distinguish* $(g, g^a, g^b, g^c, g^d, g^e)$ *from* $(g, g^a, g^b, g^c, g^{ab}, g^{ac})$.
We can write this as two libraries, similar to $L_{ddh-real}$ and $L_{ddh-ideal}$ from the lecture, as follows:

$L_{ddhp-ideal}$**:**

- Initially, sample $a, b, c, d, e \in \mathbb{Z}_q$ uniformly random and independent.
- The function $GetInstance()$ outputs $(g, g^a, g^b, g^c, g^d, g^e)$.

$L_{ddhp-real}$**:**

- Initially, sample $a, b, c \in \mathbb{Z}_q$ uniformly random and independent.
- The function $GetInstance()$ outputs $(g, g^a, g^b, g^c, g^{ab}, g^{ac})$.

In this exercise, you will show that any attack on DDH' can be translated into an attack on DDH.

1. Let $\mathcal{A}$ be an algorithm that distinguishes $L_{ddhp-real}$ from $L_{ddhp-ideal}$. What in the behavior of both libraries is different, so that $\mathcal{A}$ can distinguish it?

2. We want to construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ as a sub-routine, so that $\mathcal{B}$ solves DDH. The algorithm $\mathcal{B}$ would interact either with $L_{ddh-real}$ or $L_{ddh-ideal}$ and has to tell which one it is. It therefore locally simulates a library (based on the output of the library it talks to) and lets $\mathcal{A}$ talk to the simulated library. Can you describe this process? In your simulation, you should simulate $L_{ddhp-real}$ when $\mathcal{B}$ talks to $L_{ddh-real}$ and simulate $L_{ddhp-ideal}$ when you talk to $L_{ddh-ideal}$.

3. Based on your simulation, define how $\mathcal{B}$ uses the output of $\mathcal{A}$ to tell the two libraries apart. How successful will $\mathcal{B}$ be, if $\mathcal{A}$ is correct α percent of the time?

Exercise 5. (**The Pedersen Commitment**)

Commitments are an advanced cryptographic primitive. They allow a sender to "commit" to a message m towards the receiver by sending a value c. Having only c (i.e. before m is "opened" to the receiver), the receiver cannot say what message m is contained inside c. At the same time, once c is sent to the receiver then the sender cannot change his mind and open c to another message m' anymore towards the sender. More formally, a commitment scheme consists of two algorithms:

Commit A Com algorithm which, on input m outputs values c, d.

Open An $Open$ algorithm which, on input m, c, d outputs a bit.

It is required that the commitment scheme is binding and hiding:

Binding It should be computationally difficult for a sender to generate values m, m', d, d', c such that $Open(m, c, d) = Open(m', c, d') = 1$ while $m \neq m'$. Note that sender has a free choice of all these values, as long as both messages m, m' are different but open the same commitment c which the sender can also choose.

Hiding Given m_0, m_1 by an adversary, this adversary should not be able to decide if it is a commitment to m_0 or m_1 for an honestly generated commitment c (similar to the IND-CPA property for encryption schemes, where the adversary can pick two "potential" messages but cannot say which one is ultimately encrypted in the ciphertext).

Towards constructing a commitment scheme, let us, as before, assume that p is a prime and $g \in \mathbb{Z}_p^*$ is of large prime order $q|p-1$. We assume that p, q, g are public knowledge for everyone. A first attempt for a commitment scheme is the following:

Commit On input $m \in \mathbb{Z}_q$, output $c = g^m \bmod p$ and $d = \perp$.

Open On input $m \in \mathbb{Z}_q, c \in \mathbb{Z}_p^*$ output 1 if $c = g^m \bmod p$ and 0 otherwise.

Show that this construction is insecure because it is not hiding!

A version of this, which bears resemblance to the previous exercises, is actually secure! It is called the *Pedersen Commitment* and it works as follows, assuming an additional $h \in \langle g \rangle$ (i.e. $h = g^a$ for some value a). h is also of prime order q and also publicly known (and fixed for sender and receiver):

Commit On input $m \in \mathbb{Z}_q$, sample a random $r \in \mathbb{Z}_q$ and output $c = g^m h^r \bmod p$ and $d = r$. The receiver will obtain c while the sender keeps d to itself.

Open On input $m \in \mathbb{Z}_q, c \in \mathbb{Z}_p^*, d \in \mathbb{Z}_q$ output 1 if $c = g^m h^r \bmod p$, otherwise output 0.

1. Assume that neither sender nor receiver know the discrete logarithm of h to the base g modulo p. Then the aforementioned commitment scheme is binding. To prove this, assume for contradiction that there exists a sender algorithm that can, on input p, q, g, h, generate values m, m', c, r, r' such that $g^m h^r \bmod p = g^{m'} h^{r'} \bmod p$ where $m \neq m'$. Then show that you can use this sender algorithm to compute the discrete logarithm of h to base g modulo p!

2. Show that the commitment scheme is also hiding! You can show that an honestly generated commitment $c = g^m h^r \bmod p$ could have been generated by any other message m' using a different randomness r'.