Solutions for week 4, Cryptography Course - TDA 352/DIT 250

In this weekly exercise sheet: you will define and apply the Diffie-Hellman key exchange protocol, RSA, and the ElGamal encryption scheme.
Completing the ex. sheet: you will be able to use and describe the most widely known public key exchange protocols.

Easy

---

1. - **Unconditional security**. The cryptosystem cannot be broken even by an Adversary with unlimited computational power.
   - **Computational security**. The best known attack to the cryptosystem is infeasible in practice.
   - **Provable security**. An Attack that breaks the cryptosystem, could also be used to solve some well-known difficult problem (thus to falsify some hardness assumption).

2. The RSA encryption scheme is defined by the following algorithms:

   - Gen(λ): Let λ be a security parameter. Let p, q be two λ-bit prime numbers.
     Compute $N = pq$ and $\varphi(N) = (p-1)(q-1)$. Choose a public exponent e such that $\gcd(e, \varphi(N)) = 1$ and compute a private exponent d such that $d \equiv e^{-1} \pmod{\varphi(N)}$.
     The public key is (N, e) and the secret key is (N, d).
   - Enc$((N, e), m)$: To encrypt a message m, compute the ciphertext $c \equiv m^e \pmod{N}$.
   - Dec$((N, d), c)$: To decrypt a ciphertext c, compute $m \equiv c^d \pmod{N}$.

3. The ElGamal encryption scheme is defined by the following algorithms:

   - Gen(λ): Generate a description of a cyclic group $G = \langle g \rangle$ of order q which is a λ-bit long integer. Choose a random value $x \in \{1, \cdots, q-1\}$ and compute $h = g^x$ in G.
     The public key is $\text{pk} = (G, g, q, h)$ and the secret key is $sk = (x)$.
   - Enc(pk, m): To encrypt a message m, choose a random $r \in \{1, \cdots, q-1\}$, then compute $c_1 = g^r$, and $c_2 = m \cdot h^r$ in G. The ciphertext is $c = (c_1, c_2)$.
   - Dec(sk, c): To decrypt a ciphertext c, compute $k = c_1^x$. The message is $m = c_2 k^{-1}$.

4. The IND-CPA security game is defined as follows:

   (a) The attacker $\mathcal{A}$ challenges the challenger $\mathcal{C}$.
   (b) $\mathcal{C}$ generates the public and secret key pair (pk, sk) and returns to $\mathcal{A}$ the public key pk.
   (c) $\mathcal{A}$ can now perform polynomially-many encryptions of messages of his choice.
   (d) $\mathcal{A}$ chooses two messages m_0, m_1 with same length and sends them to $\mathcal{C}$.
   (e) $\mathcal{C}$ generates a random bit $b \xleftarrow{R} \{0, 1\}$ and computes $c = \text{Enc}(pk, m_b)$, which is then sent to $\mathcal{A}$.
   (f) $\mathcal{A}$ uses a PPT algorithm.
   (g) $\mathcal{A}$ outputs $b' \in \{0, 1\}$; a guess for b.

   The adversary $\mathcal{A}$ wins the game if $b' = b$.

5. The IND-CCA security game is defined as follows:

   (a) The attacker $\mathcal{A}$ challenges the challenger $\mathcal{C}$.
   (b) $\mathcal{C}$ generates the public and secret key pair (pk, sk) and returns to $\mathcal{A}$ the public key pk.
   (c) $\mathcal{A}$ can now perform polynomially-many encryptions of messages of his choice.
   (d) $\mathcal{A}$ asks for the decryption of q ciphertexts c_i. $\mathcal{C}$ decrypts them and sends the messages m_i to $\mathcal{A}$. This is the **query phase 1**.
   (e) $\mathcal{A}$ chooses two messages m_0, m_1 with same length and sends them to $\mathcal{C}$.

(f) $\mathcal{C}$ generates a random bit $b \xleftarrow{R} \{0,1\}$ and computes $c = \text{Enc}(pk, m_b)$, which is then sent to $\mathcal{A}$.

(g) $\mathcal{A}$ asks for the decryption of q ciphertexts c_i with $c_i \neq c$. $\mathcal{C}$ decrypts them and sends the messages m_i to $\mathcal{A}$. This is the **query phase 2**.

(h) $\mathcal{A}$ uses a PPT algorithm.

(i) $\mathcal{A}$ outputs $b' \in \{0,1\}$; a guess for b.

The adversary $\mathcal{A}$ wins the game if $b' = b$.

Medium

6. We consider an ElGamal encryption scheme with a group generator g. Alice has a private key x and a public key $X = g^x$. Let $c = (c_1, c_2)$ be the given ciphertext that the Adversary wants to decrypt. We know that the corresponding plaintext is $m = c_2 \cdot c_1^{-x}$; of course the Adversary cannot compute this, since he does not know the private key x, but may request the decryption of a modified ciphertext and tries to learn the original message. To see one possibility, we just multiply the equation above by 2 to get

$$2m = 2c_2 \cdot c_1^{-x}$$

and we see that a suitable choice is $c' = (c_1, 2c_2)$. If we get the plaintext m' back, we know that $m' = 2m$, i.e. $m = m'/2$.

It is possible to adapt the previous attack to the case of RSA. Let N denote the RSA modulus, $e, d \in \mathbb{Z}_N$ be the encryption and the decryption exponents respectively. Let $c = m^e$ be the given ciphertext that the Adversary wants to decrypt. In order to learn m, the adversary queries for the decryption of the ciphertext $c' = k^e c$ for any $k \in \mathbb{Z}_N^*$ (recall that e is public). Indeed let m' be the decryption of c', then $m' = (c')^d = (k^e c)^d = k^{de} c^d = km$, and thus the adversary can compute $m = m' k^{-1}$.

7. The exercise asks to find **a** solution for $313 \cdot x + 276 \cdot y = 1$.
The standard way of resolving this exercise is to compute the extended Euclidean algorithm (preferably using the table method) which returns a solution x, y **and** if 313 and 276 are coprime.
In this exercise however, we will show you a different (but equivalent) way to achieve the same solution.

First, we consider the expression we want to compute in modular arithmetic:

$$313 \cdot x + 276 \cdot y = 1 \iff \begin{cases} 313 \cdot x = 1 \pmod{276} \\ 276 \cdot y = 1 \pmod{313} \end{cases}$$

and from the Chinese Remainder Theorem, we can consider

$$313 \cdot x = 1 \pmod{276} \iff \begin{cases} 313 \cdot x = 1 \pmod{4} \\ 313 \cdot x = 1 \pmod{3} \\ 313 \cdot x = 1 \pmod{23} \end{cases}$$

and it is

$$\begin{cases} 313 \cdot x = x = 1 \pmod{4} \\ 313 \cdot x = x = 1 \pmod{3} \\ 313 \cdot x = 14 \cdot x = 1 \pmod{23} \end{cases}$$

Since $\gcd(1,4) = 1$, $\gcd(1,3) = 1$ and $\gcd(14,23) = 1$, we know that there exists a solution x.[1].

Now, $14 \cdot x = (2 \cdot 7)x = 1 \pmod{23}$, and it can be easily seen that $2^{-1} = 12 \pmod{23}$ as $2 \cdot 12 = 24 = 1 \pmod{23}$.
For $7^{-1} \pmod{23}$, we observe that $7 \cdot 3 = 21 = (-2) \pmod{23}$ and since $2^{-1} = 12 \pmod{23}$, we have $(-2)^{-1} = -12 \pmod{23}$.

$$\begin{cases} 2^{-1} = 12 \pmod{23} \\ (-2)^{-1} = -12 \pmod{23} \end{cases} \longleftrightarrow \begin{cases} 7 \cdot 3 = 21 = (-2) \pmod{23} \\ (-2)^{-1} = -12 \pmod{23} \end{cases} \longleftrightarrow$$

[1] And this implies that $\gcd(313, 276) = 1$.

$$\longleftrightarrow 7 \cdot 3 \cdot (-12) = 21 \cdot (-12) = (-2)(-12) = (-2)(-2)^{-1} \pmod{23} = 1 \pmod{23}$$

and so $7^{-1} = 3 \cdot (-12) = 10 \pmod{23}$.

We obtain

$313 \cdot x = 14 \cdot x = 1 \pmod{23} \iff x = (14)^{-1} \pmod{23} = (2)^{-1}(7)^{-1} \pmod{23} = 12 \cdot 10 \pmod{23} = 5 \pmod{23}$

.

We have

$$\begin{cases} x = 1 \pmod{4} \\ x = 1 \pmod{3} \\ x = 5 \pmod{23} \end{cases}$$

from which, since $4 \cdot (1) + 3(-1) = 1$ (Bezout's identity), we have

$$\begin{cases} x = 1 \pmod{4} \\ x = 1 \pmod{3} \\ x = 5 \pmod{23} \end{cases} \iff \begin{cases} x = (1) \cdot 4(1) + (1) \cdot 3(-1) = 1 \pmod{12} \\ x = 5 \pmod{23} \end{cases}$$

and since $1 = 12 \cdot (2) + 23 \cdot (-1)$, we have

$$\begin{cases} x = (1) \cdot 4(1) + (1) \cdot 3(-1) = 1 \pmod{12} \\ x = 5 \pmod{23} \end{cases} \iff x = 1 \cdot (23) \cdot (-1) + 5 \cdot (12)(2) = 97 \pmod{276}$$

and so we get that $x = 97 \pmod{276}$.

If we consider the starting equation

$$313 \cdot x + 276 \cdot y = 1 \overset{x=97}{\Longrightarrow} 313 \cdot 97 + 276 \cdot y = 1 \Rightarrow 276 \cdot y = 1 - 313 \cdot 97 = 1 - 30361$$

and we can compute

$$y = \frac{1 - 30361}{276} = \frac{-30360}{276} = -110$$

and so we get a solution $x = 97$ and $y = -110$.

Remark: the values that you find are exactly the same that you compute with the extended Euclidean algorithm.

8. Let $p = 13, q = 17$ be two primes and $N = p \cdot q = 221$ be the RSA modulus. Consider as a public exponent, $e = 11$.
We have that $\phi(N) = (13 - 1)(17 - 1) = 192$. Notice that $192 = 2^6 \cdot 3$.

   - To compute the private exponent d, we have to solve (using the CRT)

$$d = e^{-1} \pmod{192} = \begin{cases} e^{-1} \pmod{2^6} \\ e^{-1} \pmod{3} \end{cases} = \begin{cases} 11^{-1} \pmod{64} \\ 11^{-1} \pmod{3} \end{cases}$$

   Now, $11 = (-1) \pmod{3}$ and so $d = 11^{-1} = (-1) \pmod{3}$.

   For the case $d = 11^{-1} \pmod{64}$, we can only use the extended Euclidean algorithm[2] and obtain $d = 35 \pmod{64}$.

   From the fact that $3 \cdot (-21) + 64 \cdot (1) = 1$, we have

$$\begin{cases} d = 35 \pmod{64} \\ d = 2 \pmod{3} \end{cases} \Longrightarrow d = 2 \cdot (64 \cdot 1) + 35 \cdot (3 \cdot (-12)) = -2077 = 35 \pmod{\phi(N)}$$

   and so $d = 35$.

[2] Just for your information, the Hensel's lemma, that describes an algorithm which although harder to compute, can be useful and retrieves the same result.

- To encrypt the message $m = 2$, we have to compute $2^e = 2^{11}$. Since $2^{10} = 1024$, we have

$$2^e = 2^{11} = 2^{10} \cdot 2 = 1024 \cdot 2 = 140 \cdot 2 \pmod N = 59 \pmod N$$

- To decrypt the message $c = 126$, we have to compute $126^d = 2^{35}$. Since $126 = 2 \cdot 3^2 \cdot 7$, we can consider

$$126^d = (2 \cdot 3^2 \cdot 7)^d = 2^d \cdot 3^{2 \cdot d} \cdot 7^d \pmod N$$

Now, $d = 35 = 32+2+1$, we can iteratively square the results and compute the different elements:

$$\begin{cases} 2^1 = 2 \pmod N \\ 2^2 = 4 \pmod N \\ 2^{10} = 140 \pmod N \\ 2^{20} = 140^2 = 152 \pmod N \\ 2^{32} = 2^{20}2^{10}2^2 = 152 \cdot 140 \cdot 4 = 35 \pmod N \\ 2^{35} = 2^{32}2^2 2^1 = 35 \cdot 4 \cdot 2 = 59 \pmod N \end{cases} \qquad \begin{cases} 3^1 = 3 \pmod N \\ 3^2 = 9 \pmod N \\ 3^4 = 81 \pmod N \\ 3^8 = 152 \pmod N \\ 3^{16} = 120 \pmod N \\ 3^{32} = 35 \pmod N \\ 3^{35} = 35 \cdot 9 \cdot 3 = 61 \pmod N \\ 3^{70} = 185 \pmod N \end{cases}$$

$$\begin{cases} 7^1 = 7 \pmod N \\ 7^2 = 49 \pmod N \\ 7^4 = 191 \pmod N \\ 7^8 = 16 \pmod N \\ 7^{16} = 35 \pmod N \\ 7^{32} = 120 \pmod N \\ 7^{35} = 120 \cdot 49 \cdot 7 = 54 \pmod N \end{cases}$$

and obtain

$$126^{35} = 2^{35}3^{70}7^{35} = 59 \cdot 185 \cdot 54 \pmod N = 3 = m$$

and so the decryption is $m = 3$.

9. Given that textbook-RSA is a deterministic encryption scheme, an adversary knowing the public key can pre-compute the ciphertexts c_0, c_1 for the messages m_0 and m_1 respectively.
At this point, the adversary sends m_0, m_1 to the challenger and gets c. The adversary then only has to check if c is equal to c_0 or c_1 to obtain b.

10. The DH key exchange protocol is defined by the following protocol between two parties A and B:

- A generates a description of a cyclic group $G = \langle g \rangle$ of order q and B accepts the public parameters.
- A chooses at random a value $a \in \{1, \cdots, q-1\}$ and computes $A = g^a$ in G and sends it to B.
- B chooses at random a value $b \in \{1, \cdots, q-1\}$ and computes $B = g^b$ in G and sends it to A.
- A computes the common shared secret key as $B^a = \text{sk} = g^{ab}$.
- B computes the common shared secret key as $A^b = \text{sk} = g^{ba}$.

The Man-in-the-Middle attack is an attack where the adversary can see and modify the messages in a conversation between two parties.
In this attack, the adversary Eve has *the power* to steel the identity of Bob, for example, and convince Alice that *"she is Bob"*. In other words, Eve can *impersonate* Bob.
In a key-exchange protocol, a MitM attacker has full control of the communication channel.

Consider $G = \mathbb{Z}_p$ with $p = 11$ and a generator $g = 2$.
Bob sends you $B = g^b = 8$.

- To simulate the DH protocol, we have to randomly choose $a \in \{1, \cdots, 10\}$.
Carlo rolls a D10., say $a = 5$, then Alice communicates $A = 2^5 = 32 = 10 \pmod{11}$.
The common secret will be $B^5 = 8^5 = 2^{15} = 2^5 = 10 \pmod{11}$.

- The MitM attacker is quite powerful, and there are some heuristic techniques to defend against such an attacker.

Hard

11. (a) We have that
$$c_2^q = (m \cdot (g^x)^y)^q = m^q \cdot (g^q)^{xy} = m^q$$
since $g^q = 1$ (g is a generator of a group of order q).
Note: The adversary cannot, of course, compute m from m^q; this is exactly the discrete log problem.

    (b) To clarify the context of the problem: If the adversary has computed m_1^q and m_2^q as in (a) and found them equal, can she conclude that $m_1 = m_2$? As we shall see, this is not mathematically certain, but the exceptional cases are rare. That is, the answer to this question is positive with *high probability*. First we compute the inverse of m_2, and then we consider the product
$$(m_1 m_2^{-1})^q = 1,$$
i.e., $(m_1 m_2^{-1})$ has order q or 1 (no other possibilities, since q is prime). m_1 and m_2 are p-bit integers, so the probability that $m_1 m_2^{-1}$ will generate a group of order q is extremely small; the adversary can safely assume that $m_1 m_2^{-1} = 1$, i.e., $m_1 = m_2$.

12. (a) Victor checks that $R \cdot S = X$ (since $R \cdot S = g^{r+(x-r)} = g^x = X$), and that $R = g^z$ (if $b = 0$) or $S = g^z$ (if $b = 1$).

    (b) If the false Peggy guesses that she will get $b = 0$ in message 2, she chooses r at random, and sends $R = g^r$, $S = R^{-1}X$. Her values will then pass Victor's check. If she guesses that $b = 1$, she just exchanges R with S. In both cases, $z = r$.

    (c) Repeat the protocol t times and accept only if the check succeeds each time. Then a false Peggy has probability $(\frac{1}{2})^t = 2^{-t}$ to be accepted.

Think

13. The main reason is the security of the *addition discrete logarithm*.
It is possible to change the RSA encryption scheme and use just additions. If we follow the security proof, we have that this RSA scheme will base its security on the hardness of finding x in this problem
$$a \cdot x = b \pmod{N}$$
with $a, b \in \mathbb{Z}_N$.
This is a linear congruence which is incredibly simple to solve, and so RSA with addition has no security at all.

14. Let N be the public RSA modulus and suppose that $\mathbb{Z}_N^*$ is cyclic.
The number of generators of $\mathbb{Z}_N^*$ is $\phi(\phi(N))$. Every generator g is a possible base for the discrete logarithm: consider g' another generator, finding a such that $g'^a = b$ with b fixed can be easier with respect to the similar problem $g^a = b$ (observe that we changed the base into g).
Having $\mathbb{Z}_N^*$ cyclic is a nice property but it does not happen in reality: to be cyclic, N has to be either $2, 4, p^\alpha$ or $2p^\alpha$ with p prime different from 2.
In RSA, we have $N = p \cdot q$, where p, q are both primes different than 2. Observe that N cannot be of any of the mentioned forms. Therefore, we know for certain that in RSA, $\mathbb{Z}_N^*$ is not cyclic.

15. Yes, it is possible. But, even in this case, we need a lot of communication between the three parties and if we scale to n parties, we have an enormous number of message-exchanges between the parties involved, and so it is not usable in practice.
Generalizing the known key-exchange protocols to the case where there are n parties with $n > 3$ is an *open problem* in cryptography!