Université Libre de Bruxelles
Faculté des Sciences

2022-2023

INFO-F-405: Introduction to cryptography

4. Public-key techniques

Going public

Exercise 1

The characters in the text below make use of public-key cryptography. They each prepared a pair of public/private keys and they all know everyone else's public key. Complete the following sentences and scratch the invalid alternatives (as much as possible without looking at the slides or at your notes).

To send a confidential file to Xavier, Yasmina _____ it with her/Xavier's _____ key. When she receives it, she _____ it with his/her _____ key.

Elena sends to her husband Fabrice the list of items to buy at the grocery store. The list is not confidential, but she wants to avoid their daughter Gabi from adding chocolates or other extra items to it. To do so, she _____ the list with her/Fabrice's/Gabi's _____ key before she sends it. When he receives it, Fabrice _____ the _____ with his/Elena's/Gabi's _____ key.

Answer of exercise 1

To send a confidential file to Xavier, Yasmina **encrypts** it with **Xavier's public** key. When she receives it, she **decrypts** it with **her private** key.

Elena sends to her husband Fabrice the list of items to buy at the grocery store. The list is not confidential, but she wants to avoid their daughter Gabi from adding chocolates or other extra items to it. To do so, she **signs** the list with **her private** key before she sends it. When he receives it, Fabrice **verifies** the **signature** with **Elena's public** key.

Exercise 2

Thelma and Louise recently met for the first time and would like to do business together. After their meeting, because of the COVID-19, they are forced to stay at

home and can communicate only by phone or via the Internet. They wish to start discussing their business project, but want to do so confidentially. Hopefully, they already exchanged their phone numbers and email addresses, and are both knowledgeable about cryptography. However, their project is highly sensitive and they need to protect their conversation against even active adversaries, that is, people who can afford to tamper with Internet connections. We nevertheless assume that the adversaries are not going to cut their communication lines, as this would be too obvious and as they would otherwise miss all opportunities of spying on the two business women.

Please explain how Thelma and Louise can set up a secure communication channel using public-key cryptography.

Answer of exercise 2

Let us first identify their problems:

- They want to communicate confidentially.
- They cannot meet in person.
- Someone could intercept and alter their internet communication.
- Someone could intercept their phone communication.

Since Thelma and Louise cannot meet and any of their communications can be heard, they cannot exchange any secret key directly. Instead, they could use public key cryptography, but Thelma and Louise need to exchange their public keys while avoiding man-in-the-middle (MITM) attacks. To be able to encrypt with the other party's public key, they must be sure that the public key that they received indeed belongs to the right person. To ensure this, Thelma and Louise can simply call each other and verify each other's public key fingerprint (i.e., the hash of the public key). We assume that they are able to recognize the each other's voices and that an adversary cannot simulate their voices while they speak out their public key fingerprints.

To exchange the public keys, one could say that Thelma and Louise can rely on a public key infrastructure (PKI). However, it is not so clear how to use this properly. The goal would be that Thelma and Louise obtain a certificate on their public keys, but how to avoid a MITM on that part? This only moves the problem elsewhere: How can the certification authority (CA) ensure that the public key indeed belongs to Thelma or to Louise?

Exercise 3

Alice has recently arrived in Belgium for a one-year Erasmus at ULB/VUB. Just like any other student, she has been granted access to the university supercomputer, *Hydra*, to run experiments for her Master's Thesis.

We will assume that the connection protocol with Hydra consists in a Diffie-Hellman key exchange followed by the symmetric encryption of the communications. Upon the first connection, Alice's computer requests Hydra's long-term public key and stores it for the subsequent connections. Then for each connection, Alice's computer generates an ephemeral key pair and sends the public key to Hydra. Finally, each party generates a secret key from its private key and the public key received from the other party.

Carelessly, Alice connects to Hydra with this protocol from her student residence.

a. Let us assume that an attacker (Charles) has control over the network of Alice's student residence. Charles knows that many students will try to connect to Hydra. How can Charles intercept and read all communications between Alice's computer and Hydra in the clear without being noticed?

b. The next day, Charles does not intercept connections to Hydra anymore. What will Alice notice when she connects to Hydra? Why?

c. In SSH, the first time you connect to a host, you get prompted with a *fingerprint* and asked whether that *fingerprint* matches what you expect from the remote host you are trying to reach. What is this fingerprint? To what is it applied? What kind of attack does it prevent?

d. Let us assume that there is a trusted authority such as the Belgian government in the loop. Could the ULB/VUB prevent attacks such as the one Charles is able to perform via this trusted authority ?

Answer of exercise 3

a. Since Charles has control over Alice's network, he can redirect the traffic going to Hydra to his own machine and thereby perform a man-in-the-middle (MITM) attack.

   Consequently, when Alice will connect to Hydra and ask its public key, Charles will send his own instead, and forward all Alice's requests to the actual Hydra. Since Alice does not know the public key of Hydra, she will not notice anything.

b. Because Charles is not decrypting/re-encrypting anymore, Alice will notice that her messages are not understood by Hydra and vice-versa.

   To see this, remember that Charles' public key was stored on Alice's computer as presumed Hydra's public key. Hence, Alice generates a secret key using her ephemeral key pair and Charles' public key. The Hydra server, on its side, generates a secret key using Alice's ephemeral public key and its actual private key (and not Charles'). The generated secret keys therefore do not match between Alice and Hydra, and the traffic encrypted to the other party will decrypt as garbage.

c. This fingerprint is a hash of the public key. If the user knows the remote host fingerprint, (s)he can cross-check it at first connection. This mechanism prevents MITM attacks provided the fingerprint is securely communicated beforehand.

d. Introducing a trusted authority in the loop enables us to rely on a Public Key Infrastructure (PKI). In a PKI, a trusted authority can provide certificates stating that a given public key indeed belongs to a given user (domain name). As a result, provided Hydra has obtained a certificate signed by the Belgian Government, Alice can request that certificate when connecting and thereby check the identity of the remote host.

   This reasoning only holds assuming that Alice knows the public key of the Belgian government, which is a fair assumption to make.

   However, if we assume that this assumption does not hold, the trusted authority does not solve the problem since Alice would have to retrieve the public key of the Belgian government, which could also end up in a MITM attack...

Exercise 4

Lets say (Gen, E, D) is a semantically secure public-key encryption system. Could the algorithm E be deterministic?

Answer of exercise 4

No, because we want two ciphertexts of the same plaintext to be different.

Rivest-Shamir-Adleman

Exercise 5

RSA Encryption

a. Consider the key generation algorithm of RSA. Let $p = 11$ and $q = 17$ be two given prime numbers. What is the corresponding RSA modulus n? Assuming that the public exponent is $e = 3$, what is the corresponding private exponent d in the private key?

b. Consider the encryption algorithm of the RSA "textbook encryption" scheme. Assuming that the plaintext is $m = 15$ and the public key (n, e) is as in question a, compute the resulting ciphertext c.

c. Consider the decryption algorithm of the RSA "textbook encryption" scheme. Using a computer, check that the decryption of the ciphertext c from the previous question indeed yields the original plaintext $m = 15$.

d. Consider the figure depicting RSA-OAEP in the slides and assume a 16-bit RSA modulus (hence $n = 16$ in the notation of that figure), 8 bits of randomness (hence $k0 = 8$) and 4 bits of redundancy (hence $k1 = 4$).

   The functions $G, H : \{0,1\}^* \to \{0,1\}^*$ used in RSA-OAEP should be instantiated with extendable output functions (or hash functions with the MGF1 mode). For simplicity, however, we will assume in this question that they take a very simple form. With $\kappa = k0 = n - k0 = 8$, G and H are defined as follows:

   $$G : b_0 \dots b_{\kappa-1} \mapsto b_{\kappa-1} \dots b_0 \qquad \text{and} \qquad H : b_0 \dots b_{\kappa-1} \mapsto \overline{b_0} \dots \overline{b_{\kappa-1}},$$

   where $\overline{b_i} = 1 - b_i$ for all $0 \leq i < \kappa$.

   Let the plaintext be $m = 1010$ and the randomness be $r = 1110\,0101$. Compute the resulting input $(X \| Y)$ of the exponentiation and give its numerical value in decimal.

e. Let's say we want to set up an RSA system for k users. How many primes do we have to generate? What security issues we might create if we use less prime numbers?

Answer of exercise 5

a. The RSA modulus is $n = 11 \times 17 = 187$. The relationship between RSA exponents e and d is given by the following equation: $ed \bmod \phi(n) = 1$. In other words d is the (multiplicative) inverse of e modulo $\phi(n)$. We first compute $\phi(n) = \phi(187) = \phi(11)\phi(17) = 10 \times 16 = 160$. Note that since $\gcd(3, 160) = 1$ the existence of this inverse is guaranteed.

   We can compute the inverse of 3 mod 160 by computing the extended GCD algorithm, but this was not seen in this course. Alternatively, we can start

from Bézout's identity: There exist integers x and y such that $3x + 160y = \gcd(3, 160) = 1$, and x is the inverse we are looking for, as $(3x + 160y) \bmod 160 = 3x \bmod 160 = 1$. Since $160 = 3 \times 53 + 1$, we can rewrite it as $3(x + 53y) + y = 1$. Setting $x' = x + 53y$, we have $3x' + y = 1$, which can be solved by taking $x' = 1$ and $y = -2$. Finally, $x' = 1$ means that $x + 53 \times (-2) = 1$ and $x = 107$. We can check that indeed $(3 \times 107) \bmod 160 = 1$.

The corresponding private exponent is therefore $d = 107$.

b. Textbook RSA ciphertexts are computed as $c = m^e \bmod n$. Hence, $c = 15^3 \bmod 187 = 3375 \bmod 187 = 9$.

c. The decryption process of RSA "textbook encryption" requires us to compute $m = c^d \bmod n$. Hence we need to compute $m = 9^{107} \bmod 187 = 15$. One can use for instance Python: `(9 ** 107) % 187`.

d. First we extend m with $k1 = 4$ zeroes and get $m' = 1010\,0000$. Then, we compute $G(r) = 1010\,0111$ and get $X = G(r) \oplus m' = 0000\,0111$. Next, we compute $H(X) = 1111\,1000$ and get $Y = r \oplus H(X) = 0001\,1101$. The result is $X \| Y = 0000\,0111\,0001\,1101$ and this is interpreted as the integer 1821.

e. We need at least $2k$ primes, otherwise the users sharing a factor could easily recover the private key of each other.

Exercise 6

In the RSA "textbook signature" scheme, a message m is signed in the following way under the private key (n, d):

$$m \to (m, s) = (m, m^d \bmod n),$$

Is there any way of creating a forgery?

Answer of exercise 6

Yes, starting from a valid message-signature pair $(m, s) = (m, m^d)$, the adversary chooses any value c and constructs the following new message-signature pair:

$$(m', s') = (m \times c^e, s \times c).$$

To see that (m', s') is a forgery, let us verify the signature with the public key:

$$(s')^e = (s \times c)^e = s^e \times c^e = m^{ed} \times c^e = m \times c^e = m'.$$

In other words, (m', s') would be accepted as a valid message-signature pair, although it was not signed with the private key.

Exercise 7

Alice frequently needs to upload files to Bob's server, and they use RSA and hybrid encryption to keep their files confidential. Please find below the specifications of their protocol, similar to RSA-KEM, to which we added a few mistakes.

One-time setup

1. Bob sets $e = 3$.
2. Bob randomly chooses a private prime number p of 256 bits. He checks that $\gcd(e, p) = 1$; otherwise, he repeats with a new prime p.
3. Similarly, Bob randomly chooses another private prime number q of 256 bits. He checks that $\gcd(e, q) = 1$ and $p \neq q$; otherwise, he repeats with a new prime q.
4. Bob computes $d = e^{-1} \bmod \phi(pq)$ and keeps d private.
5. Bob computes $n = pq$ and sends (e, n) to Alice. They check together that Alice received Bob's public key correctly.

When Alice needs to upload a file F

6. Alice chooses a random string m of 512 bits.
7. Alice computes the 160 bits of output of $\mathrm{SHA1}(m)$ and interprets them as the integer k with $0 \leq k \leq 2^{160} - 1$.
8. Alice computes $c = k^e \bmod n$, and she encrypts her file F with a good symmetric encryption scheme using the secret key k resulting in ciphertext G.
9. Alice uploads (c, G).

When Bob receives an encrypted file (c, G)

10. Bob recovers k by computing $k = c^d \bmod n$.
11. Bob chooses a random string m of 512 bits and computes $k' = \mathrm{SHA1}(m)$. If $k' \neq k$, it outputs an error message and aborts.
12. Bob decrypts G with the same good symmetric encryption scheme, using k as secret key, to recover F.
13. Bob stores F on the server.

Questions

a. What is the size in bits of Bob's modulus n that will result from the setup? Does that give sufficient security in 2021? If so, please justify briefly. If not, please recommend which size the primes should have to achieve about 128 bits of security.

b. On line 4, what is $\phi(pq)$? Can you give a simpler expression? What would happen if $\phi(pq)$ was part of Bob's public key?

c. There is a mistake in the one-time setup procedure that can cause the computation to fail sometimes. What is it? How can you fix it, and why? Before it is fixed, what is approximately the probability that this procedure fails?

d. There is a mistake in the way RSA is used that causes a major security issue. What is it? How can you fix it, and why?

e. There is a silly mistake in Bob's procedure that causes it to fail almost always. What is it? How can you fix it, and why?

f. Can someone other than Alice upload a file onto Bob's server? If so, please sketch briefly how to modify the protocol so that only Alice can upload a file. Otherwise, please explain briefly how the current protocol achieve this restriction.

Answer of exercise 7

a. Because Bob constructs the modulus n as the product of two 256-bit integers, n will be 511 or 512-bit long. This does not give sufficient security nowadays, as a 512-bit integer was factored in 1999 and, at of the time of this writing, the record is the factorization of a 829-bit integer, see `https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`. To meet 128-bit security, the NIST currently recommends a 3072-bit modulus.

b. Here $\phi(n) = \phi(pq) = (p-1)(q-1)$ since p and q are distinct primes.

If $\phi(n)$ was made public, one could easily find d from e since $d = e^{-1} \bmod \phi(n)$. And we saw (see slides) that the knowledge of e and d allows one to factor n easily, hence to break the entire scheme. Alternatively, from n and $\phi(n)$, one could deduce the sum of p and q: $n-\phi(n)+1 = pq-(p-1)(q-1)+1 = p+q$. Finding p and q from their sum and product comes down to solving a quadratic equation.

c. On lines 2 and 3, Bob checks that $\gcd(e, pq) = 1$; since e, p and q are distinct primes, this is always the case. However, he should check that $\gcd(e, (p-1)(q-1)) = 1$ instead; this check is required to make sure that e has a multiplicative inverse modulo $\phi(n)$ and hence that d exists.

   Before it is fixed, the computation fails when $\phi(n) = (p-1)(q-1)$ is a multiple of $e = 3$. We have $\Pr[3 \text{ does not divide } p-1] \approx \frac{2}{3}$, and similarly for q. So $\Pr[3 \text{ does not divide } (p-1)(q-1)] \approx \frac{4}{9}$ and the failure probability is about $1 - \frac{4}{9} = \frac{5}{9}$.

d. There is a short message attack!

   We have $k < 2^{160}$ and therefore $k^e = k^3 < 2^{480}$. But since n is at least 2^{510} (and more likely n is even much bigger as seen in question a), we have that $k^e < n$ and the modular reduction (on line 8) has no effect. An attacker can therefore easily recover k by computing $\sqrt[3]{c}$ in the plain integers.

   The best way to fix this is by following RSA-KEM more closely: 1) Alice chooses m of the same size as n, 2) Alice computes $c = m^e \bmod n$ (instead of k^e) and 3) Bob first recovers m and then gets k by hashing m.

e. Line 11 is pointless and is most likely going to fail. We can simply remove it.

f. Yes, assuming that Bob's public key (e, n) is indeed public, anyone can upload a file onto Bob's server. Bob does not check that the file comes from Alice. Furthermore, Bob has no way to check the authenticity of the file because he has no way to distinguish Alice from someone else.

   Alice could also create a pair of public and private keys, share her public key with him and check with him that he received it correctly (to avoid MITM attacks). This way, Alice could sign the file that she uploads and Bob could verify the signature before storing it onto the server.

Discrete logarithm problem in $\mathbb{Z}_p^*$

Exercise 8

The discrete logarithm problem is based on the multiplicative group $(\mathbb{Z}_p^*, \times)$. What happens if we would use the additive group $(\mathbb{Z}_p, +)$ instead? *Hint:* Start with writing out the problem in that group.

Answer of exercise 8

The discrete "logarithm" problem is easy to solve in the additive group.

To see this, let us fix a generator $g \in \mathbb{Z}_p$. Then, the problem is the following: Given $A = ag \bmod p$, find A. It is sufficient to compute $a = g^{-1}A \bmod p$ with $g^{-1} \bmod p$ the multiplicative inverse of g modulo p.

Exercise 9

ElGamal encryption

a. Let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_p^*$, $p = 23$. The order of $\mathbb{G}$ is $|\mathbb{G}| = 11$. Let $g = 4$ be a generator of $\mathbb{G}$. Consider the key generation algorithm of the ElGamal encryption scheme (and of all DLP-based schemes). Assuming that the private key is $a = 3$, compute the corresponding public key A.

b. Consider the encryption algorithm of the ElGamal encryption scheme. Assume that the plaintext $m = 3$, the random exponent (ephemeral private key) chosen by the encryption algorithm is $k = 2$, and $\mathbb{G}$, g, and A are as in question a. Compute the resulting ciphertext (K, c).

c. Consider the decryption algorithm of the ElGamal encryption scheme. Assume that the ciphertext is $(K, c) = (6, 22)$ and $\mathbb{G}$, g, a, and A are as in question a. Compute the resulting plaintext m.

Answer of exercise 9

a. We have that the public key $A = g^a$ in $\mathbb{G}$. Hence $A = g^a \bmod 23 = 4^3 \bmod 23 = 64 \bmod 23 = 18$.

b. ElGamal ciphertexts (K, c) are computed as follows: $K = g^k$ and $c = A^k m$ in $\mathbb{G}$. We thus compute $K = 4^2 \bmod 23 = 16$ and $c = 18^2 \cdot 3 \bmod 23 = 324 \cdot 3 \bmod 23 = 6$. The ciphertext is thus $(K, c) = (16, 6)$.

c. The decryption process of ElGamal encryption requires us to first compute K^a and its inverse in $\mathbb{G}$. We first compute $K^a = 6^3 \bmod 23 = 9$. Now we need to compute the inverse of 9 mod 23 by using the extended euclidian algorithm and find it is equal to 18, so $K^{-a} = 18$ in $\mathbb{G}$. The next step in the decryption process is the computation of $m = c \cdot K^{-a}$ in $\mathbb{G}$. Hence we can compute $m = 22 \cdot 18 \bmod 23 = 5$.

Exercise 10

What is the additive group notation for $\mathbb{Z}_p^*$? Given a prime number p, let the group $\mathbb{G}$ be the set of integers between 1 and $p-1$ (like $\mathbb{Z}_p^*$) equipped with the group operation $+$ be the multiplication modulo p. (Yes, an "addition" symbol to denote a multiplication in this case!) The neutral element of $\mathbb{G}$ is denoted O, i.e., $O = 1$, and the inverse of an element A is denoted $-A$.

The *scalar multiplication* refers to the process of repeating the group operation onto itself a given number of times. We denote $[n]A$ the group element obtained by repeating the group operation on n copies of A. So, $[0]A = O$, $[1]A = A$, $[2]A = A + A$, $[3]A = A + A + A$, etc, and $[-1]A = -A$, $[-2]A = (-A) + (-A)$, etc.

For instance, let $p = 23$ as above, let A and B be group elements $A = 4$ and $B = 6$. Then $A + B = O$ as $4 \times 6 \equiv 1 \pmod{23}$. So A and B are each other's inverse, i.e., $A = -B$. Also, $[5]A = 12$ since $4^5 \equiv 1024 \equiv 12 \pmod{23}$. And $[-1]A = -A = 6$.

ElGamal encryption in additive group notation Assuming a group $\mathbb{G}$, a generator $G \in \mathbb{G}$, rewrite the key generation process to generate Alice's public-private key pair in additive group notation. Then, given a plaintext message $M \in \mathbb{G}$, rewrite the ElGamal encryption scheme in the same notation. Finally, rewrite the decryption and explain why it correctly recovers the plaintext.

Answer of exercise 10

Alice generates a public-private key pair as follows:

- Privately choose a random integer in $a \in \mathbb{Z}_q \setminus \{0\}$, with $q = |\mathbb{G}|$
- Compute $A = [a]G$

The public key is A and the private key is a.

To encrypt $M \in \mathbb{G}$ with Alice's public key A:

- Choose randomly an integer $k \in \mathbb{Z}_q \setminus \{0\}$
- Compute
$$\begin{cases} K = [k]G \\ C = M + [k]A \end{cases}$$

To decrypt, Alice computes:
$$M = C + [-a]K$$

Why is it correct?

$$[-a]K = [-ak]G = [-k]A = (-[k]A)$$
$$C + [-a]K = M + [k]A + [-a]K = M + [k]A + (-[k]A) = M$$

Exercise 11

Let's say Alice and Bob both choose a planet that they want to visit (in our solar system). They want to check if they choose the same planet without giving out their choice. Let's say Alice choses the planet a and Bob the planet b. Alice and Bob agree on the following scheme:

- They publicly choose a prime p and generator g of $G = \mathbb{Z}_q^*$
- Alice chooses random x and y in $\mathbb{Z}_p$ and sends to Bob $(A_0, A_1, A_2) = (g^x, g^y, g^{xy+a})$
- Bob choose random r and s in $\mathbb{Z}_p$ and sends back to Alice $(B_1, B_2) = (A_1^r \times g^s, \left(\frac{A_2}{g^b}\right)^r \times A_0^s)$

How Alice can check if she had chose the same planet as Bob?

Answer of exercise 11

Check whether B_1^x is equal to B_2.

Exercise 12

Schnorr Signature

a. In contrast to RSA with full-domain hashing, the Schnorr signature scheme is probabilistic, i.e., different executions of the signing algorithm on the same input (a, m) results in different signatures $\sigma = (s, e)$. How many different Schnorr signatures can exist for a single message $m \in \{0,1\}^*$?

b. Let q the size of the group, i.e., $q = p - 1$ if g is a generator of $\mathbb{Z}_p^*$. Assume that during the generation of some Schnorr signature $\sigma = (s, e)$ the random exponent (or ephemeral private key) $k \in \mathbb{Z}_q$ becomes known to an adversary $\mathcal{A}$. How can $\mathcal{A}$ use σ and k in order to forge a Schnorr signature σ' for any message m' of its choice?

c. Assume that the same random exponent (or ephemeral private key) $k \in \mathbb{Z}_q$ used in the signing algorithm of the Schnorr scheme was used multiple times. More precisely, we have two signatures $\sigma_1 = (s_1, e_1)$ and $\sigma_2 = (s_2, e_2)$ on two different messages $m_1 \neq m_2$ that were generated using the same exponent $k_1 = k_2 = k$. How can an adversary $\mathcal{A}$ use σ_1 and σ_2 in order to forge a Schnorr signature σ' for any message m' of its choice?

d. To make k random and unique per signature, the signer generates a random $k \in \mathbb{Z}_q$ upon the first signature, and then increments it ($k \leftarrow k + 1$) for each new signature. Is that secure? If so, please justify. If not, what is the problem and how can you fix it?

e. Propose two ways to pick k in a secure way. *Hint:* One probabilistic and one deterministic.

Answer of exercise 12

a. Take a closer look at the signing algorithm. For each choice of $k \in \mathbb{Z}_q$ value, $r = g^k$ and $s = k - ea$ are uniquely determined by m. Since the mapping $k \mapsto g^k$ is a bijection, i.e., for every $k \in \mathbb{Z}_q$ there exists exactly one group element $r = g^k$ and for every group element g^k there exists exactly one $k \in \mathbb{Z}_q$, we can conclude that there exist exactly $|\mathbb{Z}_q| = q$ different Schnorr signatures $\sigma = (s, e)$ for each message m.

b. $\mathcal{A}$ knows that $s = k - ea \bmod q$ and $e = \text{hash}(r \| m)$. From this $\mathcal{A}$ can compute the private key $a = e^{-1}(k - s) \bmod q$ and start forging signatures for arbitrary m' using the original signing algorithm of the Schnorr scheme.

c. $\mathcal{A}$ knows that $s_1 = k_1 - e_1 a \bmod q$ and $s_2 = k_2 - e_2 a \bmod q$, where hash values e_1 and e_2 are also known to $\mathcal{A}$. Since $k_1 = k_2 = k$, we subtract the two equations and we get the equality $s_1 - s_2 = a(e_2 - e_1) \bmod q$, which can be rewritten as $a = (e_2 - e_1)^{-1}(s_1 - s_2) \bmod q$. Note that $e_1 = e_2$ may occur only with a negligible probability since the hash function used to compute them is assumed to be collision-resistant. Hence, $\mathcal{A}$ can use $\sigma_1 = (s_1, e_1)$ and $\sigma_2 = (s_2, e_2)$ to compute the private key a and start forging signatures for arbitrary m' using the original signing algorithm of the Schnorr scheme.

d. Let us assume that a first message was signed using k followed immediately by a second message signed using $k + 1$. So, $\mathcal{A}$ knows that $s_1 = k - e_1 a \bmod q$ and $s_2 = (k + 1) - e_2 a \bmod q$. Again, we subtract the two equations and we get the equality $s_1 - s_2 + 1 = a(e_2 - e_1) \bmod q$, which can be rewritten as $a = (e_2 - e_1)^{-1}(s_1 - s_2 + 1) \bmod q$. Like in the previous question, the adversary can recover the private key a and start forging signatures for arbitrary messages.

To conclude, not only the ephemeral key k must be secret and unique, it must also be chosen independently upon each signature.

e. We can choose k randomly and uniformly in its range using a physical random bit generator, independently upon each signature, or choose k deterministically by hashing together m and a secret value, e.g., $k = \text{hash}(a \| m)$, to ensure only one possible signature per message.

Elliptic curve cryptography

Exercise 13

Elliptic curve in $\text{GF}(11)$**.** In this exercise, all the arithmetic operations are understood modulo 11, although not explicitly written. Let E be the curve over $(\text{GF}(11))^2$ satisfying the Weierstrass equation

$$y^2 = x^3 - 3x + 7.$$

a) List the points on the curve. To do this, we advise the following steps:

   - Build a table of squares modulo 11. For each value y, compute y^2. Sort the table per value y^2. How many squares modulo 11 are there? For a given value y^2, how many corresponding values y can there be? When more than 1, how do these values relate to each other?
   - For each value x, compute $x^3 - 3x + 7$. Using the table of squares modulo 11, look up the possible value(s) of y (if any) that satisfy $y^2 = x^3 - 3x + 7$.

b) How many points does E have? What is the largest prime-order group we can get? What is the cofactor?

c) Which point has order 1? Which point has order 2? (Hint: think about the elliptic curves over the reals.)

Answer of exercise 13

a) - There are six squares modulo 11. These are 0, 1, 3, 4, 5 and 9. The value 0 has one corresponding square root (itself), while the other squares have two possible square roots (a and b) that add up to 11, i.e., one is the opposite of the other ($a + b = 0 \mod 11$). The other values (i.e., the non-squares) have no square roots in $\text{GF}(11)$.

x	$x^2 \mod 11$
0	0
1	1
2	4
3	9
4	5
5	3
6	3
7	5
8	9
9	4
10	1

- The points on E are $(1, \pm 4)$, $(2, \pm 3)$, $(3, \pm 5)$, $(4, \pm 2)$, $(8, 0)$, $(9, \pm 4)$, $(10, \pm 3)$ plus O the point at infinity.

x	$y^2 = x^3 - 3x + 7$	y
0	7	
1	5	4, -4
2	9	3, -3
3	3	5, -5
4	4	2, -2
5	7	
6	7	
7	10	
8	0	0
9	5	4, -4
10	9	3, -3

b)
- There are thus 14 points.
- The largest prime-order group has size 7.
- The co-factor is 2.

c)
- The point at infinity O has order 1 because it is the neutral element in the group.
- Point $(8, 0)$ has order 2. In the reals, the point with $x = 0$ has a vertical tangeant, and this property carries over to curves over finite fields, hence $[2](8, 0) = O$.

Exercise 14

ElGamal signature with elliptic curves The purpose of this exercise is to translate the ElGamal signature scheme as expressed with modular exponentiation into an elliptic curve-based scheme.

First, we recall the original scheme here. Given a prime p and a generator g over $\mathbb{Z}_p^*$, we proceed as follows.

- **Signature** of message $m \in \mathbb{Z}_2^*$ by Alice with her private key a:
  - Compute $h = \text{hash}(m)$
  - Choose randomly an integer $k \in [1, p-2]$
  - Compute $r = g^k \bmod p$
  - Compute $s = k^{-1}(h - ar) \bmod (p-1)$
    - If $s = 0$, restart with a new k
  - Send (r, s) along with m

- **Verification** of signature (r, s) on m with Alice's public key $A = g^a \bmod p$:
  - Compute $h = \text{hash}(m)$
  - Check $A^r r^s \stackrel{?}{\equiv} g^h \pmod p$

Now, let E be an elliptic curve over $\text{GF}(p)$ and let $G \in E$ be a generator of prime order q. Rewrite the scheme above by replacing operations in $\mathbb{Z}_p^*$ with operations in E. (Hint: to compute the equivalent of r, first compute $R = [k]G$ then let r be the x-coordinate of R.)

What is the equivalent of operations modulo $p - 1$?

Answer of exercise 14

Given an elliptic curve E and a generator $G \in E$ of prime order q, we proceed as follows.

- **Signature** of message $m \in \mathbb{Z}_2^*$ by Alice with her private key $a \in \mathbb{Z}_q$:
  - Compute $h = \text{hash}(m)$
  - Choose randomly an integer $k \in [1, q-1]$
  - Compute $R = [k]G$ then let r be the x-coordinate of R
  - Compute $s = k^{-1}(h - ar) \bmod q$

    - If $s = 0$, restart with a new k
  - Send (R, s) along with m

- **Verification** of signature (R, s) on m with Alice's public key $A = [a]G$:
  - Compute $h = \text{hash}(m)$
  - Let r be the x-coordinate of R
  - Check $[r]A + [s]R \stackrel{?}{\equiv} [h]G$

The equivalent of operations modulo $p - 1$ are operations modulo q, in both case modulo the order of the generator.

Exercise 15

A company that installs and maintains an open-source operating system is creating an automatic update mechanism for its customers. Regularly, the company publishes an *update pack* containing the latest changes to be made to the operating system, and each computer connected to the Internet can fetch it. The company would like to guarantee the integrity of these update packs so as to avoid the installation of malicious software on their customers' computers. Fortunately, confidentiality is not required, as it is open-source software.

In the following, we describe a protocol that the company uses to sign the update packs and for the customers to check that the update packs are genuine. However, we voluntarily added some mistakes (including omissions) to its definition. Some of them are functional, that is, they prevent the protocol from working correctly, while others introduce security flaws.

Please *list all the mistakes* you find, and for each, *justify* why it is incorrect and *propose a correction*.

Let $\mathcal{E}$ be a standard elliptic curve over $\text{GF}(p)$ (for some prime p) that is used by the company and its customers. They also agreed on a base point $G \in \mathcal{E}$ with prime order q, i.e., $[q]G$ is the neutral element. Note that the uppercase letters refer to points on $\mathcal{E}$, while lowercase letters are integers, and `UP` is a string of bits that represents an update pack.

One-time setup at the company:

1. The company secretly generates its secret key c randomly and uniformly in $[1 \dots p - 1]$.

2. The company computes its public key $C = [c]G$, and publishes it via some PKI process. (This aspect is out of scope of this question. In the sequel, we assume that all the public keys are correctly authenticated and can be trusted.)

3. The company secretly generates its secret value n_C randomly and uniformly in $[1 \dots p-1]$.

Company's procedure to sign and post an update pack *UP*

1. Compute $k = \text{hash}(n_C)$.
2. Compute $R = [k]G$.
3. Compute $e = \text{hash}(R \| \mathit{UP})$.
4. Compute $s = k + ec \bmod p$.
5. Post (UP, e, s) on the company's update repository.

Customer's procedure to retrieve and install an update pack

1. Retrieve (UP, e, s) from the company's update repository.
2. Compute $C = [c]G$.
3. Compute $R' = [s]G - [e]C$.
4. Compute $e' = \text{hash}(R' \| \mathit{UP})$.
5. The customer installs *UP*.

Answer of exercise 15

The protocol is essentially a Schnorr signature on elliptic curves.

- A first problem is on line 1 of the setup procedure. The private key must be chosen in the set $[1 \dots q-1]$, with q the size of the group that G generates. The prime p does not play a role as a scalar here, it only plays a role in the coordinates (x, y) of the points on the curve, but this is abstracted away in the notation.

- A second problem is on line 1 of the company's procedure. The value k is computed deterministically from the fixed quantity n_C, hence k is fixed. This is a major security problem, as the signature of two different messages with the same value k allows anyone to recover the private key c.

  There are two possible ways to fix this:

- Draw k randomly in the set $[1 \dots q-1]$. (In this case, line 3 of the setup procedure can be discarded.)
- Derive k from the message and a secret value, e.g., $k = \text{hash}(n_C \| \mathsf{UP})$ similarly to EdDSA. (In this case, n_C does not need to be a number but rather a secret string of bits.)

- A third problem is on line 4 of the company's procedure. The computation must be done modulo q and not p, for the same reasons as above. (Some pointed out that the $+$ should be a $-$, as in the original Schnorr signature, but this expression is consistent with the way the verification is done.)

- A fourth problem is on line 2 of the customer's procedure. The customer does not have access to the private key of the company! Anyway, this step is useless because C is public and assumed to be trusted.

- A fifth and last problem is on line 5 of the customer's procedure. The protocol misses the verification that $e = e'$. The customer must check this before installing the update pack, and abort if $e \neq e'$.

Exercise 16

Projective coordinates. Instead of representing points on the curve with (x, y), called *affine coordinates*, one can use an alternate representation using three coordinates $(X : Y : Z)$ called *projective coordinates*.

When $Z \neq 0$, a point in projective coordinates $(X : Y : Z)$ represents (x, y) in affine coordinates with $x = XZ^{-1}$ and $y = YZ^{-1}$. The projective coordinates are *redundant*: For any $\lambda \neq 0$, the projective coordinates $(X : Y : Z)$ and $(\lambda X : \lambda Y : \lambda Z)$ represent the same point.

In projective coordinates, the Weierstrass equation becomes

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \tag{1}$$

a) How can we represent the point at infinity O in projective coordinates? Check that it satisfies the Weierstrass equation above. (Hint: intuitively, the point at infinity is the vertical direction. Over the reals, it can be viewed as having affine coordinates $(0, \pm\infty)$. In the projective plane, points at infinity have $z = 0$ by definition.)

b) In affine coordinates, the addition of two points $P + Q$ in the general case, i.e., assuming $P \neq Q \neq -P$ and $P \neq O \neq Q$, works as follows. We have

$(x_P, y_P) + (x_Q, y_Q) = (x, y)$ with

$$s = \frac{y_P - y_Q}{x_P - x_Q}$$
$$x = s^2 - x_P - x_Q$$
$$y = s(x_P - x) - y_P$$

Given the projective coordinates of P and Q, say, $(X_P : Y_P : Z_P)$ and $(X_Q : Y_Q : Z_Q)$, write the result of the addition in projective coordinates $(X : Y : Z)$. As the projective coordinates are redundant, the value Z can be freely chosen; you may for instance fix $Z = 1$.

c) The projective coordinates are interesting in practical implementations for their higher efficiency: The operations on the points can be expressed *without inversions*, which are otherwise costly compared to additions, subtractions and multiplications. Rewrite the addition of points above, but using only additions, subtractions and multiplications (no inversions). (Hint: set Z so as to cancel any inversions.)

Answer of exercise 16

a) The point at infinity is defined as the point with $Z = 0$. Thus, we can find the values of X and Y that satisfy the Weierstrass equation. We find that the point at infinity is represented as $(0 : Y : 0)$ with $Y \neq 0$ (otherwise it would be the origin), since Y can take any value to fit the equation. We can check that this point indeed satisfies equation 1.

b) Set $Z = 1$ and just rewrite the equations above with $x_P = X_P Z_P^{-1}$ and $y_P = Y_P Z_P^{-1}$.

c) Set $Z = Z_P Z_Q (X_Q Z_P - X_P Z_Q)^3$, which cannot be zero because of the assumptions on P and Q.