# Homework 3 - RSA Cryptography and Elliptic Curves

*Cryptography and Security 2020*

- You are free to use any programming language you want, although SAGE is recommended.
- Put all your answers **and only your answers** in the provided SCIPER-answers.txt file. This means you need to provide us with all `Q` values specified in the questions below. You can download your **personal** files from the following link: `http://lasec.epfl.ch:80/courses/cs20/hw3/index.php`
- You will find an example parameter and answer file on the moodle. You can use this parameters' file to test your code and also ensure that the types of `Q` values you provided match what is expected. **Please do not put any comment or strange character or any new line** in the .txt file.
- We also ask you to submit your **source code**. This file can of course be of any readable format and we encourage you to comment your code. Notebook files are allowed, but we prefer if you export your code as a text file with a sage/python script.
- The plaintexts of most of the exercises contain some random words. Don't be offended by them and Google them at your own risk. Note that they might be really strange.
- If you worked with some other people, please list all the names in your answer file. We remind you that you have to submit your **own source code** and **solution**.
- We might announce some typos/corrections in this homework on Moodle in the "news" forum. Everybody is subscribed to it and does receive an email as well. If you decided to ignore Moodle emails we recommend that you check the forum regularly.
- The homework is due on Moodle on **Monday the 9th of November** at 23h59.

## Exercise 1 PGPv9.2

From his luxury appartment in Miami, the famous artist Boulbi heard about the PGP encryption system and told his crew about it. They decided to use it to secure their confidential communication and thus each member generated a pair of RSA keys. However, as the original PGP software seemed very old, Boulbi hired the Crypto Apprentice to design a new version (PGPv9.2) which works with 3-prime RSA (instead of 2-prime) and which only encrypts small messages (less than 512 bits).

More specifically, the RSA modulus $N$ is computed as $N = p \cdot q \cdot r$ where $p, q, r$ are prime, and the secret exponent $d$ is computed as $d = e^{-1} \bmod \phi(N)$. At decryption, the exponentation acceleration trick using the Chinese Remainder Theorem is used (slide 271 of the course). The full encryption process for a message $m$ of $\ell$ characters works as follows.

1. Sample a random $x$ from $\mathbb{Z}_N$.
2. Encrypt $x$ as $\mathsf{ct}_{RSA} = \mathsf{enc}_{RSA}((e, N), x)$.
3. Hash $x$ as $\mathsf{key} = \mathsf{SHA512}(\mathsf{str}(x))$.
4. Compute $m'$ as the ascii encoding of $m$ ($m'$ is a $\ell$-long bytestring).
5. Compute $\mathsf{ct}_m = (\mathsf{key})_\ell \oplus m'$ where $(\mathsf{key})_\ell$ takes the $\ell$ leftmost bytes of $\mathsf{key}$ (see an example of code below). The ciphertext is $(\mathsf{ct}_{RSA}, \mathsf{ct}_m)$.

Kari, the eternal rival of Boulbi, is worried that he will not be able to spy on his enemy's conversations aynmore. He managed to recover a list of all the modulus $N_i$ of the participants and a list of past RSA plaintext/decrypted ciphertext pairs on Boulbi's computer (i.e. each tuple contains a value $x$ and $\mathsf{dec}_{RSA}(\mathsf{sk}, \mathsf{enc}_{RSA}(\mathsf{pk}, x)))$, where $(\mathsf{pk}, \mathsf{sk})$ is Boulbi's RSA key pair). In addition, Kari knows that Boulbi is very susceptible and hits his computer whenever it seems the decryption takes too long. This causes some occasional errors in the decryption process, in particular during the exponentiation of the ciphertext modulo one of the primes.

Recently, Kari intercepted the encryption (using Boulbi's public key) $(\mathsf{ct}_{RSA}, \mathsf{ct}_{pwd})$ of the password allowing access to Boulbi's secret nightclub in the basement of some restaurant. As he really wants to challenge his old rival on the dance floor, he asked you to help him recover the password.

In your parameters file, you have access to the list of pairs of plaintext/decrypted ciphertext in `Q1_L_x`, the list of moduli $N_i$ in `Q1_L_mod`, Boulbi's public key $\mathsf{pk} = (e, N)$ in `Q1_e` and `Q1_N` and the encrypted password $(\mathsf{ct}_{RSA}, \mathsf{ct}_{pwd})$ in `Q1_ct_RSA` and `Q1_ct_pwd`. Note that `Q1_ct_pwd` is a python bytestring. Put your answer in the `Q1_pwd` variable.

**Hint 1:** In order to generate all the moduli (i.e. the public keys) $N_i$ and $N$, a song of Boulbi is used as the random seed. The entropy of such a source is very low...

**Hint 2:** What happens if $y^d$ modulo one of the prime returns a random value in decryption?

**Hint 3:** The goal is to factor $N$.

Finally, here is a Sage (v9.1) code example showing how to encrypt a short message, given an integer $x \in \mathbb{Z}_N$.

```
>>> import hashlib
>>> m = 'lanezoestquadrillee'
>>> key = hashlib.sha512(str(x_key).encode('ascii')).digest()
>>> m_bytes = m.encode('ascii')
>>> ct_m = bytes(a ^^ b for (a, b) in zip(key, m_bytes))
```

## Exercise 2 Come Together, Right Now, Over Me

As our beloved Crypto Apprentice is trying to build a group messaging protocol, he decides to use the broadcast RSA scheme to design his system. The description of the protocol is as follows.

A group $G$ is defined as a set of public keys $\{pk_1 = (e_1, N_1), \ldots, pk_k = (e_k, N_k)\}$, where $N_i$ values are RSA modulus and $(e_i, \phi(N)) = 1$ for all $1 \leq i \leq k$. The Crypto Apprentice decides to make a protocol with 2 algorithms $\mathsf{Broadcast}(M, G)$, and $\mathsf{Receive}(C, sk, pk)$. These algorithm are described as follows,

**Algorithm 1:** Broadcast

**Input:** $M \in \mathbb{Z}, G = \{(N_1, e_1) \ldots, (N_k, e_k)\}$
**Output:** $C_1, \ldots, C_k$
**for** $1 \leq i \leq k$ **do**
  $C_i \leftarrow M^{e_i} \mod N_i$ ;
  $\mathsf{SEND}(C_i, \text{member}_i)$;
**return** $C_1, \ldots, C_k$;

**Algorithm 2:** Receive

**Input:** $C, sk, pk = (N^*, e^*)$
**Output:** $M^*$
**return** $C^{sk} \mod N^*$;

In order to make the protocol correct for all users, The Crypto Apprentice decided to add the condition that $M < N_i$ for all $1 \leq i \leq k$, without knowing how much of a problem this condition will cause!

In your parameters file you can find Q2_k, the number of group members, and the public keys of the group members, Q2_G $= \{(N_1, e_1), \ldots, (N_k, e_k)\}$. You can also find a broadcast of a message m, Q2_ciphertexts. You are asked to recover the message Q2_m and write it in your answers file.

**Hint:** Is there a small public-key exponent which is repeated a lot?

## Exercise 3 The pastries of Mr. Copper and Ms. Smith

*Throughout this exercise, an RSA public key is denoted by $(e, N)$, where $e$ is the public exponent and $N$ is an $n$-bit RSA modulus. Given an ASCII message $M$, we denote by $\mathsf{INT}(M)$ the integer value of the hexadecimal representation of $M$. In this exercise, we will assess the security of the textbook RSA cryptosystem.*

During their journey, a traveller found a village where a smith and a pastry chef had weird customs. Instead of paying their goods with money, the buyer would need to solve a challenge of their own. The traveller is in great needs of a new sword and wants to taste those delicious pastries. Would you be able to help them in their quest by solving the following challenges ?

**Challenge I** For this challenge, we give without proof an important result on the representation of rational numbers: *for all rational numbers $\frac{u}{v} \in \mathbf{Q}$, there exists an unique finite sequence $[a_0; a_1, \ldots, a_n]$ of integers with $a_n > 1$ called the continued fraction expansion of $u/v$ such that $\frac{u}{v} = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \ldots}}$*. For all $i \geq 0$, the $i$-th convergent is defined to be the rational number $[a_0; a_1, \ldots, a_i]$ and by convention, its irreducible representation is denoted by $h_i/k_i$.

- ▷ Given an RSA public key $(e, N)$ as (Q3a_e, Q3a_N) for which $(e^{-1} \mod \phi(N)) < \frac{1}{3} N^{\frac{1}{4}}$, recover the prime factors $p$ and $q$ of $N$ such that $q < p < 2q$ and report them as Q3a_p and Q3a_q in the answers file.

  **Hint:** Let $(e, N)$ be an RSA public key and let $(d, \phi)$ be the corresponding secret key.

Assume that $N = pq$ satisfies $q < p < 2q$ and $d < \frac{1}{3}N^{1/4}$. Then, $d$ is *exactly* the denominator of a convergent of the continued fraction expansion of $e/N$, i.e. $k_i = d$ for some index $0 \leq i \leq n$.

**Challenge II** For this challenge, we recall some important algebraic results. Given a commutative ring $R$ with unit, the resultant of nonzero polynomials $f(x)$ and $g(x)$ in $R[x]$, denoted by $res_x(f, g)$, is a polynomial $h(x)$ that satisfies $h(x) = 0$ if and only if $f(x) = g(x) = 0$. For nonzero multivariate polynomials $f, g \in R[x_1, \ldots, x_n]$, the resultant is not unique and is computed with respect to a single indeterminate $x_i$, i.e. $res_{x_i}(f, g) \in R[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$.

**Reminder:** In `SAGE`, the resultant is computed via `f.resultant(g)`. When $R$ is neither a field nor $\mathbf{Z}$, it is required to change the base rings of $f$ and $g$ before computing the resultant via the `change_ring` method. For multivariate polynomials, the resultant with respect to some variable can be computed as follows:

```
R = ... # some base ring
P.<x,y,z> = PolynomialRing(R, 3)
P1.<y,z> = PolynomialRing(R, 2)
P2.<x,z> = PolynomialRing(R, 2)
P3.<x,y> = PolynomialRing(R, 2)
f = P.random_element() # a multivariate polynomial
g = P.random_element() # a multivariate polynomial
A1 = f.change_ring(P1).resultant(g.change_ring(P1), x) # A1 in R[y,z]
A2 = f.change_ring(P2).resultant(g.change_ring(P2), y) # A2 in R[x,z]
A3 = f.change_ring(P3).resultant(g.change_ring(P3), z) # A3 in R[x,y]
```

▷ Let $M$ be a secret human-readable string and let `Q3b_e`, `Q3b_N`, `Q3b_c1`, `Q3b_c2`) be parameters generated by Algorithm 3 with input $M$. Using those parameters, recover the plaintext $M$ as a human-readable string and report it as `Q3b_M` in the answers file.

**Algorithm 3:**

**Input:** A secret message $M$.
**Output:** An RSA public key $(e, N)$ and two ciphertexts $(c_1, c_2)$.

$e \leftarrow 3$
$p \leftarrow \mathsf{random\_prime}(1024)$ `// get a random 1024-bit prime`
$q \leftarrow \mathsf{random\_prime}(1024)$ `// get a random 1024-bit prime`
$N \leftarrow pq$
$k \leftarrow 2^{\lfloor \log_2(N)/e^2 \rfloor}$
$r_1 \leftarrow \mathsf{random\_nbit\_integer}(16)$ `// get a random 16-bit integer`
$r_2 \leftarrow \mathsf{random\_nbit\_integer}(16)$ `// get a random 16-bit integer`
$\mu \leftarrow \mathsf{INT}(M)$
$c_1 \leftarrow (k\mu + r_1)^e \bmod N$
$c_2 \leftarrow (k\mu + r_2)^e \bmod N$
**return** $(e, N, c_1, c_2)$

**Hint:** Let $(e, N)$ be an RSA public key, where $N$ is an $n$-bit integer and let $m = \lfloor n/e^2 \rfloor$.

Although $\mathbf{Z}_N[x]$ is not an Euclidean ring (see Homework 1), the standard Euclidean algorithm can be applied to polynomials in $\mathbf{Z}_N[x]$ and if the latter fails, this means that the factorization of $N$ is exposed.

- ▷ Let $f(x) = ax + b \in \mathbf{Z}_N[x]$ be a linear polynomial such that $b \neq 0$. Given two messages $M_1 \neq M_2 \in \mathbf{Z}_N^\times$ such that $M_1 = f(M_2)$ and their associated RSA encryptions $C_i$, implement[1] an algorithm which, given as inputs $(f, e, N, C_1, C_2)$ outputs the plaintexts $M_1$ and $M_2$ with probability 1 when $e = 3$ and with non-negligible probability otherwise.
  **Hint:** Compute the common roots of $f(x)^e - C_1$ and $x^e - C_2$ when $e = 3$.
- ▷ Let $M \in \mathbf{Z}_N^\times$ be a message of at most $n - m$ bits and let $0 \leq r_1, r_2 < 2^m$ be distinct integers. Let $M_i = 2^m M + r_i \bmod N$ and let $C_i$ be their associated RSA encryptions. Implement an algorithm which, given as inputs $(e, N, C_1, C_2)$ outputs the plaintext $M$.
  **Hint:** Write $g_1(x, y) = x^e - C_1$ and $g_2(x, y) = (x + y)^e - C_2$. If $r_1, r_2 < 2^\delta \ll 2^m$, then $\Delta = r_2 - r_1$ is a "small" root of $h(y) = res_x(g_1, g_2) \in \mathbf{Z}_N[y]$. What can you deduce on the common roots of $G_1(x) = g_1(x, \Delta)$ and $G_2(x) = g_2(x, \Delta)$ as elements of $\mathbf{Z}_N[x]$ and how does it relate to the previous hint ?

[1] For this homework, there is no need to consider $e > 3$ as $e = 3$ in the second challenge