Homework 1 – RSA & Elliptic Curves

Cryptography and Security 2025

- ◇ 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 `[id]-answers.sage` file where `[id]` is an integer derived from the student ID.[1] This means you need to provide us with all `Q`-values specified in the questions below.
- ◇ **Please do not put any comment or strange character or any new line** in the submission file and do **NOT** rename the provided files.
- ◇ Do **NOT** modify the `id` headers in the `[id]-answers.sage` file.
- ◇ **Submissions that do not respect the expected format may lose points.**
- ◇ If you do not answer a question, set the answer parameter in `[id]-answers.sage` as `None`.
- ◇ 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 normal textual files containing Python/SAGE code. If an answer is incorrect, we may grant partial marks depending on the implementation.
- ◇ Be careful to always cite external code that was used in your implementation if the latter is not part of the public domain and include the corresponding license if needed. Submissions that do not meet this guideline may be flagged as plagiarism or cheating.
- ◇ Some plaintexts may contain random words. Do not be offended by them and search them online at your own risk. Note that they might be really strange.
- ◇ You are allowed to work in a group of two. Please list the name of the person you worked with in the `readme` of your source code. Note that each of you must **individually** submit a valid answer file on Moodle.
- ◇ Corrections and revisions may be announced on Moodle in the "News" forum. By default, 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 **12th of November** at 23h59.

[1] Depending on the nature of the exercise, an example of parameters and answers will be provided on Moodle.

Exercise 1 Partial One-wayness of RSA

In this exercise, we study the partial one-wayness of the plain RSA encryption scheme. More precisely, we first define the trapdoor one-way permutation.

Definition 1 (Trapdoor One-way Permutation)**.** A *trapdoor one-way permutation* is a collection of functions $\{f_k : \mathcal{D}_k \to \mathcal{D}_k\}_{k\in\mathcal{K}}$ such that the following holds:

1. There exists a PPT algorithm Gen that on input 1^λ outputs a key $k \in \mathcal{K}$ and a trapdoor t_k.
2. There exists a PPT algorithm Eval such that for any $k \in \mathcal{K}$ and $x \in \mathcal{D}_k$, $\mathsf{Eval}(k, x) = f_k(x)$.
3. There exists a PPT algorithm Inv such that for $(k, t_k) \leftarrow \mathsf{Gen}(1^\lambda)$ and $y \in \mathcal{D}_k$, $\mathsf{Inv}(k, t_k, y) = f_k^{-1}(y)$.
4. (One-wayness) For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that
$$\Pr\left[\begin{array}{cc} (k,t_k) \leftarrow \mathsf{Gen}(1^\lambda) & x \xleftarrow{\$} \mathcal{D}_k \\ y \leftarrow f_k(x) & x' \leftarrow \mathcal{A}(k,y) \end{array} : x' = x\right] \le \mathsf{negl}(\lambda).$$

You may notice that the plain RSA encryption scheme can serve as a trapdoor one-way permutation under the hardness assumption of the RSA Decryption Problem. In particular, we have $\mathcal{D} = \mathbb{Z}_n^*$ and $f_{(n,e)}(x) = x^e \mod n$ for public key (n, e), and $t_k = d$ is the secret key such that $ed = 1 \mod \phi(n)$.

In fact, under the same assumption, the RSA encryption scheme also satisfies a stronger notion called *partial one-wayness*.

5. ((ℓ_0, ℓ_1)-Partial One-wayness.) For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that
$$\Pr\left[\begin{array}{cc} (k,t_k) \leftarrow \mathsf{Gen}(1^\lambda) & x \xleftarrow{\$} \mathcal{D}_k \\ y \leftarrow f_k(x) & x' \leftarrow \mathcal{A}(k,y) \end{array} : x' = x|_{\ell_1}^{\ell_0}\right] \le \frac{1}{2^{l_1 - l_0}} + \mathsf{negl}(\lambda)$$
where $x|_{\ell_1}^{\ell_0}$ is the sub-bitstring of x from the ℓ_0-th bit to the ℓ_1-th bit.

This property is crucial in proving the semantic security of RSA schemes with paddings; e.g., RSA-OAEP.

To show the (ℓ_0, ℓ_1)-partial one-wayness of RSA, we assume there is a PPT algorithm Ext that can give you some bits $m|_{\ell_1}^{\ell_0}$ of the plaintext m when you query it with a ciphertext $c = m^e \mod n$. If we can use Ext to fully recover m; in particular, the rest of the bits of m, it means we break the RSA Decryption Problem. In other words, if the RSA Decryption Problem is hard, realizing Ext should be hard, which implies the partial one-wayness. (We assume Ext always works correctly in this exercise. For a rigorous proof, we should assume a success probability of $1/\mathsf{poly}(\lambda)$ instead.)

Let p, q be two primes of 1024 bits and $(\mathsf{Q1\_N} = pq, \mathsf{Q1\_e})$ be the RSA public key. Your task is to finish Algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ to recover the plaintext, given two different Ext_1 and Ext_2, respectively.

$\mathcal{A}_1(N, e, c)$	
1 :	$r \leftarrow 2^{500}$
2 :	$c' \leftarrow c \cdot r^e \bmod N$
3 :	$m_0 \leftarrow \mathsf{Ext}_1(c)$
4 :	$y_0 \leftarrow \mathsf{Ext}_1(c')$
5 :	Do something to recover m

$\mathcal{A}_2(N, e, c)$	
1 :	$r \leftarrow 2^{500}$
2 :	$c' \leftarrow c \cdot r^e \bmod N$
3 :	$(m_0, m_2) \leftarrow \mathsf{Ext}_2(c)$
4 :	$(y_0, y_2) \leftarrow \mathsf{Ext}_2(c')$
5 :	Do something to recover m

Question 1.1 Recovering Missing Least Significant Bits

▷ Let Ext_1 be an algorithm that can compute the $\lceil \log(m) \rceil - 500$ most significant bits of the plaintext. That is, we can write $\mathsf{Q1a\_m} = \mathsf{Q1a\_m_0} \cdot 2^{500} + \mathsf{Q1a\_m_1}$, where $\mathsf{Q1a\_m_1} < 2^{500}$. You are given the ciphertext $\mathsf{Q1a\_c}$, Ext_1's outputs $\mathsf{Q1a\_m_0}$ in Line 3 and $\mathsf{Q1a\_y_0}$ in Line 4 of Algorithm $\mathcal{A}_1$. Recover the missing 500-bit $\mathsf{Q1a\_m_1}$ and find $\mathsf{Q1a\_m}$.

Remark: You need to submit the full $\mathsf{Q1a\_m}$ instead of just the missing bits $\mathsf{Q1a\_m_1}$.

Hint: Let $y = y_0 \cdot 2^{500} + y_1$ such that $y^e \bmod n = c'$. What is the relation among m_0, m_1, y_0, y_1?

Question 1.2 Recovering Missing Middle Bits

▷ Let Ext_2 be an algorithm that can compute the $\lceil \log(m) \rceil - 750$ most significant bits and 250 least significant bits of the plaintext. That is, we can write $\mathsf{Q1b\_m} = \mathsf{Q1b\_m_0} \cdot 2^{750} + \mathsf{Q1b\_m_1} \cdot 2^{250} + \mathsf{Q1b\_m_2}$, where $\mathsf{Q1b\_m_1} < 2^{500}$ and $\mathsf{Q1b\_m_2} < 2^{250}$. You are given the ciphertext $\mathsf{Q1b\_c}$, Ext_2's outputs $(\mathsf{Q1b\_m_0}, \mathsf{Q1b\_m_2})$ in Line 3 and $(\mathsf{Q1b\_y_0}, \mathsf{Q1b\_y_2})$ in Line 4 of Algorithm $\mathcal{A}_2$. Recover the missing 500-bit $\mathsf{Q1b\_m_1}$ and find $\mathsf{Q1b\_m}$.

Remark: You need to submit the full $\mathsf{Q1b\_m}$ instead of just the missing bits $\mathsf{Q1b\_m_1}$.

Hint: Let $y = y_0 \cdot 2^{750} + y_1 \cdot 2^{250} + y_2$ such that $y^e \bmod n = c'$. What is the relation among $m_0, m_1, m_2 y_0, y_1, y_2$?

Hint: Since it is plain RSA, you can check the correctness of your answer by reencrypting the plaintext.

Exercise 2 Leaking Fast Arithmetic on Elliptic Curves

We have seen in class how to define the group law on an elliptic curve

$$E : y^2 = x^3 + Ax + B.$$

Given two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, their sum $P + Q = (x_3, y_3)$ is computed as

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= (x_1 - x_3)\lambda - y_1, \\ \lambda &= \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2, \\ \dfrac{3x_1^2 + A}{2y_1}, & \text{if } x_1 = x_2. \end{cases} \end{aligned}$$

These formulas, while conceptually simple, are inefficient in practice, as they require modular inversions. In fact, a single modular inversion typically costs between 10 and 100 modular multiplications.

To overcome this problem, one can avoid modular inversions by working in projective coordinates. Instead of viewing elliptic curves as subsets of the affine plane $\mathbb{F}_p^2$ (together with the point at infinity), it is mathematically more correct to see them as subsets of the projective plane $\mathbb{P}^2$.

Definition 2 (Projective space)**.** The projective space $\mathbb{P}^n$ is the set of equivalence classes of $\mathbb{F}_q^{n+1} \setminus \{0\}$ under the relation

$$(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n) \iff \exists \lambda \in \mathbb{F}_q^\times \text{ such that } x_i = \lambda y_i \text{ for all } i.$$

We denote an element of $\mathbb{P}^n$ by $(x_0 : x_1 : \cdots : x_n)$.

We can now define elliptic curves in projective space.

Definition 3 (Elliptic curve)**.** An *elliptic curve* E is the subset of $\mathbb{P}^2$ consisting of all points $P = (X : Y : Z)$ satisfying the *homogeneous Weierstrass equation*

$$Y^2Z = X^3 + AXZ^2 + BZ^3,$$

with $A, B \in \mathbb{F}_q$ such that $4A^3 + 27B^2 \neq 0$. and where the distinguished point at infinity corresponds to $0 = (0 : 1 : 0)$.

Passing between affine and projective coordinates is straightforward via the maps:

$$(X : Y : Z) \mapsto (X/Z,\, Y/Z)$$
$$(x, y) \mapsto (x : y : 1).$$

Definition 4 (Projective addition formulas)**.** Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : Z_2)$ be points on E. Then $R = P + Q = (X_3 : Y_3 : Z_3)$ is given by:

- When $P_1 \neq \pm P_2$,

$$u = Y_2Z_1 - Y_1Z_2, \quad v = X_2Z_1 - X_1Z_2, \quad w = u^2Z_1Z_2 - v^3 - 2v^2X_1Z_2,$$
$$X_3 = vw, \quad Y_3 = u\big(v^2X_1Z_2 - w\big) - v^3Y_1Z_2, \quad Z_3 = v^3Z_1Z_2.$$

- When $P_1 = P_2$,

$$t = AZ_1^2 + 3X_1^2, \quad u = Y_1Z_1, \quad v = uX_1Y_1, \quad w = t^2 - 8v,$$
$$X_3 = 2uw, \quad Y_3 = t(4v - w) - 8Y_1^2u^2, \quad Z_3 = 8u^3.$$

- Finally, when $P_1 = -P_2$, we have $P_1 + P_2 = (0 : 1 : 0)$.

An excellent reference for further reading is Silverman's *The Arithmetic of Elliptic Curves*.

Working in projective coordinates allows efficient computation of the group law without modular inversion.

Let $\mathsf{Q2\_p}$, $\mathsf{Q2\_A}$, and $\mathsf{Q2\_B}$ define an elliptic curve, and let $P = (\mathsf{Q2\_x_P}, \mathsf{Q2\_y_P}, \mathsf{Q2\_z_P}) \in E$.

Algorithm 1 Double-and-Add Algorithm

Require: Point P, scalar $k = (k_{n-1} \dots k_1 k_0)_2$
Ensure: $Q = [k]P$
  $Q \leftarrow 0$
  For $i = n - 1$ downto 0
    $Q \leftarrow 2Q$
    If $k_i = 1$
      $Q \leftarrow Q + P$
  return Q

Question 2.1 Projective Double and Add

▷ Implement the *double-and-add* algorithm (see 1 that, given a point P and an integer n, compute $[n]P$ using projective coordinates. More specifically, for integers $\mathsf{Q2a\_n_j}$, compute

$$(\mathsf{Q2a\_x_j}, \mathsf{Q2a\_y_j}, \mathsf{Q2a\_z_j}) = [\mathsf{Q2a\_n_j}]P, \quad j = 1, \cdots, 20.$$

Note: As the projective addition formula are only commutative up to scalar, for consistency during grading compute $Q + P$ instead of $P + Q$ (though both are valid).

Question 2.2 Leaking Projective Double and Add

▷ Show that the projective double-and-add algorithm leaks information about the scalar. In particular, given

$$Q_j = (\mathsf{Q2b\_x_{Q_j}}, \mathsf{Q2b\_y_{Q_j}}, \mathsf{Q2b\_z_{Q_j}}) = [\mathsf{Q2b\_m_j}]P,$$

recover the parity of the scalar, namely

$$\mathsf{Q2b\_n_j} = \mathsf{Q2b\_m_j} \bmod 2, \quad j = 1, \cdots, 20.$$

Hint: This leakage is a consequence of projective coordinates and would not occur using (x, y) coordinates.