Examination Advanced Cryptography 2024 – 2025

Vanessa VITSE and Léo COLISSON PALAIS

Only handwritten personal notes are allowed, and electronic devices are forbidden. You may assume the results of unanswered questions to proceed to the next ones. Per-question point attributions are provided as an indicator and may change later. Write each part of the assignment on a **different sheet of paper** for easier grading.

Part I (Vanessa Vitse, 10 pts)

Exercice 1: Somewhat-homomorphic encryption from pairings

Homomorphic encryption schemes allow users to evaluate functions on ciphertexts: if c is an encryption of m, then it is possible to compute an encryption of $f(m)$ for some function f, knowing only c. Fully homomorphic schemes (that is, supporting *any* function f) exist but are rather complicated. In this problem, we will study a "somewhat homomorphic" public-key encryption scheme, that allows the computation of a number of additions and a single multiplication.

Let E be a pairing-friendly elliptic curve defined over a finite field $\mathbb{F}_q$. Let P be a point of large prime order p on $E(\mathbb{F}_q)$, and $Q \in E(\mathbb{F}_{q^d}) \setminus E(\mathbb{F}_q)$ another point of order p, where d is the embedding degree of E (with respect to p). We denote by $e : E[p] \times E[p] \to \mathbb{F}_{q^d}^*$ the Weil pairing (or any other non-degenerate pairing).

Consider the following two algorithms:

- `Gen`(P) $\to$`(pk, sk)` : Choose random $a \in \mathbb{Z}/p\mathbb{Z}$ and set $P_a = aP$. Output the public key `pk` $= (P, P_a)$ and the secret key `sk` $= a$.
- `Enc(pk,`m`)`$\to$ `ct` : Given a message $m \in \mathbb{Z}/p\mathbb{Z}$, choose a random $r \in \mathbb{Z}/p\mathbb{Z}$ and output `ct` $= (rP, rP_a + mP)$ where `pk` $= (P, P_a)$.

1. Let B an integer, and assume that $-B/2 \leq m \leq B/2$. Give a `Dec` algorithm that takes a secret key `sk` and a ciphertext `ct` $= (u, v)$ and outputs m, with complexity in $O(\sqrt{B})$ operations in E.
2. Give an algorithm `Add(ct, ct')` $\to$ $\texttt{ct}_{\texttt{sum}}$ that takes as input two ciphertexts `ct` and `ct'`, which are encryptions with the same public key of $m, m' \in \mathbb{Z}/p\mathbb{Z}$ respectively, and outputs an encryption of $m + m' \pmod p$.

Now, let `(pk',sk')`$\leftarrow$ `Gen`(Q) be the public and secret keys obtained by running `Gen` using the group generated by Q. Consider the following algorithm:

- `Mult(ct, ct')` : on input two ciphertexts `ct` = (u, v) $\leftarrow$ `Enc(pk,` m`)` and `ct'` = (u', v') $\leftarrow$ `Enc(pk',` m'`)`, output the tuple $(w_1, w_2, w_3, w_4) \in (\mathbb{F}_{q^d})^4$ where

$$w_1 = e(u, u'), \quad w_2 = e(u, v'), \quad w_3 = e(v, u'), \quad w_4 = e(v, v').$$

3. Let `sk` $= a$ and `sk'` $= a'$. In the above notation, what is the discrete logarithm in base $e(P, Q)$ of $w_4.w_3^{-a'}.w_2^{-a}.w_1^{aa'}$?
4. As in question 1., assume that $-B/2 \leq mm' \leq B/2$. Explain how to recover $mm' \bmod p$ from $w_1, \ldots, w_4$ and the two secret keys `sk`, `sk'`.
5. The `Mult` algorithm thus allows to compute a multiplication on ciphertexts, but the result is a cipertext of a different format : a quadruple of elements in the finite field $\mathbb{F}_{q^d}$, instead of a couple of points in E.
   Give an algorithm $\texttt{Add}_2(\texttt{ct}_2, \texttt{ct'}_2)$ that takes as input two ciphertexts of the second type $\texttt{ct}_2 = (w_1, w_2, w_3, w_4)$ and $\texttt{ct}' = (v_1, v_2, v_3, v_4)$ for the same public key pair `(sk, sk')`, and outputs an encryption (of the second type) of the sum of the two corresponding plaintexts.

Exercice 2: Set accumulators from polynomial commitments

A *set accumulator* is a cryptographic primitive allowing users to perform several of the following tasks:

- Committing to a subset S of a universe $\mathscr{U}$
- Producing / verifying proofs that an element x belongs to S
- Producing / verifying proofs that an element x does not belong to S
- Updating committments and proofs of (non)membership to accomodate addition / removal of elements of S.

The goal of this problem is to show how to construct a set accumulator for sets S of bounded size from the Kate-Zaverucha-Goldberg (KZG) polynomial commitment scheme. Let p be a large prime number and d an integer. We recall that after a trusted setup phase generating points $P, sP, \dots, s^dP, Q, sQ$ of order p in a pairing-friendly curve for a secret parameter $s \in \mathbb{F}_p$, the KZG scheme consists of three algorithms:

- `CommitKZG`(f) outputs the constant-size commitment C of a polynomial $f \in \mathbb{F}_p[X]_{\leq d}$
- `EvalProof`(f, x, y) outputs a constant-size proof π that $f(x) = y$ (for $x, y \in \mathbb{F}_p$)
- `Verify`(C, π, x, y) verifies the correctness of the proof π that $f(x) = y$, knowing only the commitment C of f.

Let S be a subset of $\mathscr{U} = \mathbb{F}_p$ of cardinality at most d.

1. Explain how to compute a polynomial $f \in \mathbb{F}_p[X]_{\leq d}$ such that f vanishes exactly on S.
2. Show that `CommitKZG`(f) is a satistically-hiding, computationally-binding commitment to the set S (you are allowed to use security properties of the KZG scheme without proving them).
3. **Membership proofs.**
   Show that for $x \in S$, the output of `EvalProof`$(f, x, 0)$ gives a proof that x belongs to the set S. Explain how to verify this proof knowing only x and the commitment of S. Is this proof system sound?
4. **Non-membership proofs.** Let $x \notin S$.
   (a) A first idea to produce proofs that x does not belong to S is to compute $\pi =$ `EvalProof`(f, x, y) where $y = f(x)$ and to output (π, y). The verification consists in checking that $y \neq 0$ and that `Verify`$(C, \pi, x, y) =$ `True` where C is the commitment of S.
   Show that the knowledge of d non-membership proofs of this type for S allows to recover the set S.
   (Note that d is much smaller than p. Obviously, the knowledge of d membership proofs for S reveals S completely, but it should not be the case that d non-membership proofs reveal S).
   (b) A better idea is to compute $\pi =$ `EvalProof`(f, x, y) where $y = f(x)$, and then to compute $Z = yP$ as well as a zero-knowledge proof π_{DL} of knowlege of the discrete logarithm of Z in base P. The non-membership proof is then (π, Z, π_{DL}).
   In order to verify this proof knowing $C =$ `Commit`(f) and x, one checks that $Z \neq \mathscr{O}$, that π_{DL} is a valid proof for (Z, P), and that
   $$e(C, Q) = e(\pi, sQ - xQ).e(Z, Q)$$
   Show that this non-membership proof system is correct.
   Is it sound (assuming soundness of the discrete-log proof system)? What information does it leak on S?
5. **Update for element insertion.** Let $S \subset \mathbb{F}_q$ be a subset of cardinality at most $d - 1$, and $f \in \mathbb{F}_p[X]_{\leq d-1}$ a polynomial vanishing exactly on S. When computing the commitment $C =$ `CommitKZG`(f) to the set S, a user can also compute an update token $U =$ `CommitKZG`(Xf).
   (a) Describe a procedure `UpdateCommit`(C, U, a) that takes as input the commitment C for a set S, the update token U for S, and a point $a \in \mathbb{F}_p$, and outputs a valid commitment for $S \cup \{a\}$.
   (b) Let C be a commitment for a set S, and π a membership proof for $x \in S$. Show that $\pi' = C + (x - a)\pi$ is valid proof that $x \in S \cup \{a\}$ for the updated commitment C' of $S \cup \{a\}$.

Part II (Léo Colisson Palais, 10 pts)

Exercice 3: Oblivious Transfer from LWE

We have seen in the course how to obtain Oblivious Transfer (OT) from assumptions based on elliptic curves. The goal of this exercise will be to explore realizations of OT assuming the hardness of the Learning-With-Error (LWE) problem.

1. (0.5 pts) What is typically the main advantage of relying on LWE instead of elliptic curves ? And of elliptic curves over LWE ?

2. (0.25 pts) LWE is proven to be at least as (quantumly) hard as some lattice-related problems. Name one such problem.

3. (1 pts) Is deciding if a vector v belongs to a lattice $\mathcal{L}$ an efficient task to do in general, given a basis $\mathcal{L}$? Let $B = \begin{pmatrix} 4 & 9 \\ 2 & 5 \end{pmatrix}$ be the basis of a lattice $\mathcal{L}$. Which of the following vectors $v_0 \coloneqq \begin{pmatrix} 5 \\ 2 \end{pmatrix}$ and $v_1 \coloneqq \begin{pmatrix} 10 \\ 6 \end{pmatrix}$ belong to $\mathcal{L}$? Justify.

   In case it can be helpful, we remind that for any matrix $B \coloneqq \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, *we have* $\det B = ad - bc$ *and that an (innefficient) formula to compute the inverse of* B *is* $B^{-1} \coloneqq \frac{1}{|\det B|} \begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$

4. (0.5 pts) Remind the definition of the OT ideal functionality, between a sender S and a receiver R.

5. (1 pts) Let us say that one tries to obtain an OT protocol for messages of size $n > 1$ by simply repeating a bit-OT protocol (i.e. the messages belong to $\{0,1\}$) sequentially, bit-by-bit. Show that this protocol is not secure if one party (which one?) is malicious, by describing an explicit attack against the above functionality, i.e. by showing that the malicious party can extract some information that would be impossible to obtain while interacting directly from the ideal functionality.

6. The OT primitive, alone, is of limited interest. We explore here one application of OT:

   (a) (0.25 pts) Cite and explain one major functionality seen in the course based on OT having much broader applications. What do we informally expect in term of security?

   (b) (0.25 pts) Cite the name of one protocol realizing this functionality seen in the course. What is the maximum number of parties in this protocol?

   (c) (0.5 pts) In the protocol cited in the previous question, how many times do you need to run the OT protocol? (Your answer can of course depend on the various parameters of the problem.)

   (d) (0.75 pts) Informally explain how OT is used in this protocol, and why it is required. More precisely, explain *(i)* which party plays the role of the sender and who plays the role of the receiver *(ii)* what are the inputs and outputs of the OT protocol? *(iii)* what attack could be executed if the OT protocol is insecure against a malicious sender, able to learn the input of the receiver.

7. In the part, we will study an LWE-based construction to realize a "branch-based" bit-encryption mechanism. More precisely, a branch-based encryption is determined by the algorithms $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(p_k, s_k) \leftarrow \mathsf{Gen}(\mathsf{crs}, \sigma \in \{0,1\})$, $c \leftarrow \mathsf{Enc}(\mathsf{crs}, p_k, d \in \{0,1\}, m \in \{0,1\})$ and $m \leftarrow \mathsf{Dec}(\mathsf{crs}, s_k, c)$. We also expect the following statements:

   - Correctness: if the branch $\sigma \in \{0,1\}$ chosen to generate the keys is equal to the branch d chosen to encrypt a message, then we can safely decrypt the encrypted message, i.e. for any message $m \in \{0,1\}$ and branch $\sigma \in \{0,1\}$:

$$\Pr_{\substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (p_k, s_k) \leftarrow \mathsf{Gen}(\mathsf{crs}, \sigma)}} [\,\mathsf{Dec}(\mathsf{crs}, s_k, \mathsf{Enc}(\mathsf{crs}, p_k, \sigma, m)) = m\,] = 1 - \mathsf{negl}(\lambda) \tag{1}$$

- Soundness: intuitively, we expect that it should be hard to know which branch can be decoded easily if we do not know the secret key s_k. More formally, we have soundness if p_k can be seen as a secure (IND-CPA) public key[1] encryption of σ when considering Setup as the key generation algorithm. In the following, for simplicity, we will assume that the CRS is used only once per OT, hence we say that an encryption scheme is sound if the following two libraries (using the terminology of *Joy of Cryptography*) are indinstinguishable:

```
$\mathcal{L}_{\text{cpa-L}}$
1   EAVESDROP$(\sigma_L, \sigma_R)$ :
2       $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
3L      $(p_k, s_k) \leftarrow \mathsf{Gen}(\mathsf{crs}, \sigma_L)$
4       return $(\mathsf{crs}, p_k)$
```

$\approx$

```
$\mathcal{L}_{\text{cpa-R}}$
1   EAVESDROP$(\sigma_L, \sigma_R)$ :
2       $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
3R      $(p_k, s_k) \leftarrow \mathsf{Gen}(\mathsf{crs}, \sigma_R)$
4       return $(\mathsf{crs}, p_k)$
```

(2)

- Single branch recovery: Informally, given an encryption of $\mathsf{Enc}(\mathsf{crs}, p_k, d, m_0)$ and $\mathsf{Enc}(\mathsf{crs}, p_k, d, m_1)$ where p_k may be maliciously crafted, there must be at least one d such that no adversary can recover m_d (this can be formalized with simulators but we will not details this here). As this property is more technical to prove, we will not prove any formal statement here.

We consider the following instantiation based on the $\mathsf{LWE}_{n,q,\chi,m}$ problem:

```
$\mathcal{L}_{\text{branch-enc}}$
5   $\mathsf{Setup}(1^\lambda)$ :
6       $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$
7       $\mathbf{v}_0 \xleftarrow{\$} \mathbb{Z}_q^n$
8       $\mathbf{v}_1 \xleftarrow{\$} \mathbb{Z}_q^n$
9       $\mathsf{crs} := (\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1)$
10      return $\mathsf{crs}$

11  $\mathsf{Gen}(\mathsf{crs} := (\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1), \sigma \in \{0,1\})$:
12      $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$
13      $e \leftarrow \chi$
14      $p_k := \mathbf{A}\mathbf{s} + e - \mathsf{v}_\sigma$
15      $s_k := \mathbf{s}$
16      return $(p_k, s_k)$
17  $\mathsf{Enc}(\mathsf{crs} := (\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1), p_k, d \in \{0,1\}, m \in \{0,1\})$:
18      $\mathbf{x} \leftarrow \{0,1\}^m$
19      $c := (\mathbf{x}^T\mathbf{A}, \mathbf{x}^T(p_k + \mathbf{v}_d) + m\lceil \frac{q}{2} \rfloor)$
20      return c
```

(3)

(a) (0.5 pts) crs represents a Common Reference String (CRS). Who is typically supposed to generate a CRS, and therefore run the Setup phase? (ideally and in practice)

(b) (0.75 pts) Detail how a branch-based encryption can be used to obtain an OT protocol. The security should not be formally proven, one or two sentences are enough to justify the security.

(c) (1 pts) How can you instantiate $m \leftarrow \mathsf{Dec}(\mathsf{crs}, s_k, c)$? Prove that the resulting scheme is correct according to the above definition.

(d) (0.75 pts) To prove the soundness of the scheme, start to prove the following useful lemma:

$\forall m \in \mathbb{N},$

```
$\mathcal{L}_{\text{random-add}}$
21  SAMP$(\mathbf{v})$ :
22      $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$
23L     return $\mathbf{b} + \mathbf{v}$
```

$\equiv$

```
$\mathcal{L}_{\text{random-single}}$
21  SAMP$(\mathbf{v})$ :
22      $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$
23R     return $\mathbf{b}$
```

(4)

(e) (2 pts) Assuming the hardness of $\mathsf{LWE}_{n,q,\chi,m}$ (for n, q and χ defined above and for any m):

```
$\mathcal{L}_{\text{LWE-real}}$
24  SAMPLE() :
25      $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$
26      $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$
27      $\mathbf{e} \xleftarrow{\$} \chi^m$
28      return $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$
```

$\approx$

```
$\mathcal{L}_{\text{LWE-rand}}$
24  SAMPLE() :
25      $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$
29      $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_q^m$
30      return $(\mathbf{A}, \mathbf{b})$
```

(5)

[1] Note that the security definition differs slightly from the IND-CPA seen in the course since in a public key setting, the public key is also known to the attacker, but the rest is identical.

formally prove (using the *Joy of Cryptography* framework, ideally detailing all steps) that the instantiated scheme is sound, i.e. that:

$$\mathcal{L}_{\text{cpa-L}} \diamond \mathcal{L}_{\text{branch-enc}} \approx \mathcal{L}_{\text{cpa-R}} \diamond \mathcal{L}_{\text{branch-enc}} \tag{6}$$

For conciseness, when defining hybrid libraries, you can just write the numbers $\textcircled{\textit{i}}$ *identifying the lines to avoid writing them entirely. Feel free to introduce additional custom line numbering (starting from* $\textcircled{\textit{31}}$ *to avoid naming clash), and to name you intermediate libraries to avoid rewriting them completely.*