Exam

Preliminaries:

- A *graph* is a pair (V, E) where V is a set of vertices, and E is a set of pairs of (distinct) vertices, called the edges of the graph. Two edges $\{a, b\}$ and $\{c, d\}$ in E *intersect* if and only if they share a common vertex; that is, if and only if $\{a, b\} \cap \{c, d\} \neq \varnothing$.
- Whenever we talk about *committing* in the exercises, we will use the following simple commitment scheme. To *commit* to a value x, Alice draws a sufficiently long binary string r uniformly at random, and sends $h = H(x, r)$ as her commitment, where H is a public hash function. To *open* her commitment, Alice reveals x and r. We assume the hash function H is collision-resistant: in particular, once h is given, there is only one way for Alice to open her commitment, she cannot find $(x', r') \neq (x, r)$ such that $H(x', r') = H(x, r) = h$ (formally, the commitment is *binding*, as defined in the course). We also assume that h reveals no information about x (formally, we assume the commitment is *hiding*, as defined in the course).
- Throughout the exercises, Enc denotes a modern, probabilistic IND-CCA public-key encryption scheme.
- $\mathbb{Z}_N^*$ denotes the multiplicative group of $\mathbb{Z}_N$ (the elements of $\mathbb{Z}_N$ that are invertible for multiplication).

Exercise 1 (Zero-knowledge identification)**.** Alice wants to be able to prove her identity to Bob, in order to get access to her files stored on Bob's server. To do so, Alice's first idea is to use the public-key encryption scheme Enc. Before starting, Alice picks a private key for Enc, and publishes the corresponding public key. When she needs to identify herself to Bob, Alice proceeds as follows. First, Bob encrypts some random data r with Enc, and sends the encryption to Alice. Then, Alice recovers r by decrypting the data using her private key, and sends back r to Bob. Bob accepts if the decrypted r sent back by Alice is correct.

> **1.a.** Insofar as this identification protocol can be seen as a "proof" of Alice's ability to decrypt, is the proof honest-verifier zero-knowledge? What if Bob is malicious (general zero-knowledge)?

Alice instead considers the following protocol. She picks two large prime numbers p and q, and sets $N = pq$. All computations from now on are modulo N. Alice picks k random values $s_i \in \mathbb{Z}_N^*$, and computes $v_i = s_i^2$ for i in $[1, k]$. She publishes N and the v_i's as her public identification key. Later, when Alice wants to identify herself to Bob, she proceeds as follows. First, Alice picks a random $r \in \mathbb{Z}_N^*$, and sends $x = r^2$ to Bob. Bob then draws some uniformly random bits $a_1, \ldots, a_k$, and sends them to Alice. Alice computes $y = r \prod s_i^{a_i}$, and sends y to Bob. Bob accepts if $y^2 = x \prod v_i^{a_i}$.

> **1.b.** Show that this proof of knowledge is complete (correct), sound, and honest-verifier zero-knowledge.

Exercise 2 (Oblivious compaction)**.** Alice is storing n files on Bob's server. All files are of the same size. Currently, the files are simply stored in an array A of size n, one file after the other, on Bob's server. Each cell in the array A can hold one file. All files are encrypted with the public-key encryption scheme Enc, with Alice holding the private key (but not Bob). Recently, Alice went through her files, downloading and decrypting them one after the other, and tagged some of the files with the "OLD" tag. Each time, after adding the tag or not, she reuploaded a fresh encryption of the file in the same place in the array A (potentially including a tag, which we assume does not change the file size). Now, Alice wants to delete all files tagged with OLD, and reduce her storage cost on Bob's server. But she does not want Bob to know which files are being deleted among the n files. She is okay with Bob learning *how many files* are deleted, but not which ones.
Let $r \leq n$ be the number of files tagged with OLD. What Alice wants is to do is to reorder the files in the array A, so that all files tagged with OLD come first. Once that is done, she can simply tell Bob to delete the first r files, and free up that memory. The problem is, Alice is using her smartphone, which is only capable of holding two files at the same time (plus some auxiliary data, like cryptographic keys, counters, etc), and she does not want Bob to learn which files are being moved to the first r positions.

2.a. Since the files tagged with OLD have been modified before being re-encrypted and re-uploaded, and the others have been re-encrypted and re-uploaded with no difference in content, can Bob deduce from this which files have been tagged with OLD?

2.b. Propose a solution to Alice's problem using oblivious sorting. How many rounds of interaction (roundtrips) are needed between Alice and Bob?

Alice knows oblivious sorting is a little slow, and would prefer a faster solution (her smartphone is not very powerful). She considers instead the following protocol. Assume $n = 2^k$ is a power of two. Bob creates $k + 1$ arrays $(A_i)_{i\in[0,k]}$ of the same size as array A, with $A_0 = A$; the other arrays are initially empty. For each file tagged as OLD, Alice adds an integer tag d denoting how many positions the cell needs to be moved to the left, so that all cells tagged OLD end up in the first r positions (in the same order as in A). Then, Alice scans through each array A_i starting from level $i = 0$. For each cell at position j in A_i whose file is marked as OLD, and tagged with integer d_j, the cell is copied to array A_{i+1} at position $j - (d_j \bmod 2^{i+1})$. The distance tag d_j is at the same time updated as $d_j \leftarrow d_j - (d_j \bmod 2^{i+1})$, since the file has now been moved $d_j \bmod 2^{i+1}$ positions to the left.

2.c. How can Alice perform all these operations obliviously, given the limitations of her smartphone?

2.d. Prove that the new position $j - (d_j \bmod 2^{i+1})$ in array A_{i+1} is equal either to j or to $j - 2^i$.

2.e. Prove that no collision occurs, that is, it is never the case that two files are moved to the same cell in an array.

2.f. How many rounds of interaction are needed between Alice and Bob to carry out the whole operation? How much memory is needed on Bob's server?

2.g. $(*)$ Assume that Alice's smartphone is a little more powerful, and can store $m > 2$ files at the same time. Propose a more efficient variant of the previous oblivious protocol. (To avoid unnecessary complications, the proposal is not required to work for all possible values of m, it is enough if it works for an infinite number of possible m's.)

Exercise 3 (Zero-knowlede Proofs for NP-complete languages)**.** In the course, we have seen a zero-knowledge proof of knowledge of a 3-coloring of a graph. Instead, let us consider the following problem. A *vertex cover* of a graph is a set of vertices S such that every edge in the graph has at least one endpoint in S. Alice wants to prove to Bob that she knows a vertex cover S of size k for some public graph G. She proceeds as follows.

a) Alice draws a uniformly random permutation π of the vertices V. She computes a commitment h_π to the permutation π. Let $G' = (V', E')$ be the permuted graph, with $V' = \pi(V)$ and $E' = \{\{\pi(a), \pi(b)\} : \{a, b\} \in E\}$. Let $S' = \pi(S)$ be the vertex cover in the permuted graph. Alice computes commitments $(h_s)_{s\in S'}$ for each individual vertex in the vertex cover, and also computes commitments $(h_e)_{e\in E'}$ for each individual edge. She sends all commitments h_π, (h_s), (h_e) to the verifier.

b) The verifier picks a bit $b \in \{0, 1\}$, and an integer $i \in [1, |E|]$, uniformly at random. The verifier sends b and i to Alice.

c) If $b = 0$, Alice opens her commitment to π, and her commitments to every edge (h_e), and sends them to the verifier. If $b = 1$, Alice opens her commitment to the i-th value in (h_e), corresponding to some edge e in G', and opens her commitment to one value in (h_s), choosing it so that the corresponding vertex v belongs to e (uniformly at random if there are several such vertices). She sends the openings to the verifier.

d) The verifier accepts the proof iff there were k values in (h_s) (so the vertex cover is the right size), if the open commitments are correct (with the earlier conventions, the opening x, r satisfies $H(x, r) = h$, where h was the original commitment), and additionally: if $b = 0$, the verifier checks that the edges E' are equal to E permuted by π; if $b = 1$, the verifier checks that the opened vertex v belongs to the opened edge e.

3.a. Is this proof of knowledge complete (correct), sound, and honest-verifier zero-knowledge?

3.b. $(*)$ Consider the k-clique problem: Alice wants to prove she knows a k-clique in G, that is, a subset C of vertices of G of size k, such that for every $v \neq w \in C$, $\{v, w\} \in E$ (the subgraph induced by G on those vertices is complete). Sketch a zero-knowledge proof of knowledge for this problem. (Detailed arguments about correctness, soundess and zero-knowledge are not required.)