Exam

Notation.

- Throughout the exam, we fix a cyclic group $\mathbb{G}$ of prime order p. The prime p can be assumed to be very large (exponentially large relative to the security parameter). Fix g and $h = g^\alpha$ two distinct generators of $\mathbb{G}$. Later on, g and h will be public, but not α which is only known to a trusted authority.
- All discrete logarithms will be in base g. Given an element $f \in \mathbb{G}$, write $\log(f)$ for the discrete logarithm of f in base g. That is, $\log(f)$ is the unique element in $\mathbb{Z}_p$ such that $f = g^{\log(f)}$.
- For f_1, f_2 two elements of $\mathbb{G}$, define $f_1 \otimes f_2 = g^{\log(f_1)\cdot\log(f_2)}$. Equivalently, $\otimes$ is defined by: $\forall x, y \in \mathbb{Z}_p, g^x \otimes g^y = g^{xy}$. If $f_1, \ldots, f_n$ are elements of $\mathbb{G}$, define $\bigotimes_{i=1}^n f_i = f_1 \otimes \cdots \otimes f_n$.

Electronic voting via shuffling

Important. The goal of the exam is to build an electronic voting protocol. All exercises are intermediate steps towards that goal. As a consequence, questions and exercises are ordered in a logical sequence to build the voting protocol. They are *not* ordered from easiest to hardest. Feel free to answer questions in any order, skipping questions and coming back to them as needed. The exam is designed in such a way that skipping questions is not a problem to answer subsequent questions. Make sure to indicate clearly the number of the question you are answering.

Exercise 1. Zero-knowledge log products.
Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be public elements of $\mathbb{G}$. Alice knows $x_i = \log X_i$ and $y_i = \log Y_i$ for all i. Alice wants to prove to Bob in zero-knowledge the statement: $\bigotimes_{i=1}^n X_i = \bigotimes_{i=1}^n Y_i$.
Question 1.1 Suppose $\exists i, X_i = 1$, or $\exists i, Y_i = 1$. Propose a zero-knowledge proof protocol for that special case. Sketch an argument that your protocol is zero-knowledge. (Arguing about correctness and soundness is not required.)
Hint: If your argument is not trivial, you are not on the right track.
From now on, we assume $\forall i, X_i \neq 1$, and $\forall i, Y_i \neq 1$. Consider the following protocol.

1. Alice picks independently and uniformly at random values $\theta_1, \ldots, \theta_{n-1}$ in $\mathbb{Z}_p$. She computes:

$$A_1 = Y_1^{\theta_1}, \quad A_2 = X_2^{\theta_1} Y_2^{\theta_2}, \ldots, A_i = X_i^{\theta_{i-1}} Y_i^{\theta_i}, \ldots, A_{n-1} = X_{n-1}^{\theta_{n-2}} Y_{n-1}^{\theta_{n-1}}, \quad A_n = X_n^{\theta_{n-1}}.$$

   Alice sends $A_1, \ldots, A_n$ to Bob.

2. Bob picks a challenge $\gamma \in \mathbb{Z}_p$ uniformly at random, and sends it to Alice.

3. Alice computes $n-1$ values $r_1, \ldots, r_{n-1}$ in $\mathbb{Z}_p$, such that:

$$Y_1^{r_1} = A_1 X_1^{-\gamma}, X_2^{r_1} Y_2^{r_2} = A_2, \ldots, X_i^{r_{i-1}} Y_i^{r_i} = A_i, \ldots, X_{n-1}^{r_{n-2}} Y_{n-1}^{r_{n-1}} = A_{n-1}, X_n^{r_{n-1}} = A_n Y_n^{(-1)^{n-1}\gamma}. \quad (1)$$

   Alice sends $r_1, \ldots, r_{n-1}$ to Bob.

4. Bob accepts the proof if the equation system (1) holds.

Completeness. For once, completeness is not trivial, because we need to argue that, assuming $\bigotimes_{i=1}^n X_i = \bigotimes_{i=1}^n Y_i$ is true, Alice can compute $r_1, \ldots, r_{n-1}$ such that (1) is satisfied.
Question 1.2.a Let $\overline{r}_i = r_i - \theta_i$. Show that (1) is equivalent to:

$$\begin{pmatrix} y_1 & 0 & 0 & \cdots & 0 \\ x_2 & y_2 & 0 & \cdots & 0 \\ 0 & x_3 & y_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & x_{n-1} & y_{n-1} \\ 0 & 0 & \cdots & 0 & x_n \end{pmatrix} \begin{pmatrix} \overline{r}_1 \\ \overline{r}_2 \\ \overline{r}_3 \\ \vdots \\ \overline{r}_{n-2} \\ \overline{r}_{n-1} \end{pmatrix} = \begin{pmatrix} -\gamma x_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ (-1)^{n-1}\gamma y_n \end{pmatrix}. \quad (2)$$

Question 1.2.b Prove that the following determinant is zero:

$$\begin{vmatrix} x_1 & y_1 & 0 & 0 & \cdots & 0 \\ 0 & x_2 & y_2 & 0 & \cdots & 0 \\ 0 & 0 & x_3 & y_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & x_{n-1} & y_{n-1} \\ (-1)^n y_n & 0 & 0 & \cdots & 0 & x_n \end{vmatrix} = 0.$$

Question 1.2.c Deduce that the system (2) must have a solution $\overline{r}_1, \ldots, \overline{r}_{n-1}$.
Question 1.2.d Deduce that Alice can compute a solution $r_1, \ldots, r_{n-1}$ to (1). An exact computation is not required; briefly sketching how the values can be computed is enough.
Zero-knowledge.
Question 1.3 Prove that the protocol is honest-verifier zero-knowledge. Is your proof showing perfect, statistical, or computational zero-knowledge?
Reminder: this means building a simulator that, without knowing any secret information, is able to output a simulated transcript $((A_1, \ldots, A_n), \gamma, (r_1, \ldots, r_{n-1}))$, such that the simulated transcript is indistiguishable from the real transcript (i.e. the sequence of messages between Alice and Bob in a real run of the protocol).
Soundness.
Question 1.4.a Prove that if the statement $\bigotimes_{i=1}^n X_i = \bigotimes_{i=1}^n Y_i$ is false, then it is not possible for Alice to find $r_1, \ldots, r_{n-1}$ such that Bob will accept the proof, except with some small probability. Give an upper bound for that small probability (a.k.a. soundness error).
Question 1.4.b Can the proof be interpreted as a proof of knowledge? If so, knowledge of what?
Hint: this is a difficult question that can be skipped without any issue.

Exercise 2. Zero-knowledge shuffle.
Given two public vectors $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ in $\mathbb{G}^n$, assume we know how to prove that $\bigotimes_{i=1}^n A_i = \bigotimes_{i=1}^n B_i$, with a proof that is complete, sound, and zero-knowledge. (Such a protocol was built in the previous exercise, but in this exercise, we don't care how it works, just that it exists and has the desired properties.) Let us call such a protocol a log-product protocol (LPP).
Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be public elements of $\mathbb{G}$. Let c and d be (secret) elements of $\mathbb{Z}_p$, and let $C = g^c$ and $D = g^d$ be public. Alice wants to prove to Bob in zero-knowledge: "there exists a permutation π of $\{1, \ldots, n\}$, and $c, d \in \mathbb{Z}_p$, such that for all i, $Y_i^d = X_{\pi(i)}^c$". Assume Alice knows c and d, and all discrete logarithms $x_i = \log X_i$ and $y_i = \log Y_i$. Alice uses the following protocol.

1. Bob picks a challenge $t \in \mathbb{Z}_p$ uniformly at random, and sends it to Alice.
2. Alice and Bob execute the LPP protocol for the two vectors:

$$\Phi = (X_1/D^t, \ldots, X_n/D^t, \overbrace{C, \ldots, C}^{n}) \quad \Psi = (Y_1/C^t, \ldots, Y_n/C^t, \overbrace{D, \ldots, D}^{n}).$$

3. Bob accepts the proof if and only if he accepts the proof of the LPP protocol.

Question 2.1 Prove that the protocol is complete: if the statement "$\exists \pi, \forall i, Y_i^d = X_{\pi(i)}^c$" is true, then Bob will accept the proof.
Question 2.2 Prove that the protocol is honest-verifier zero-knowledge.
Question 2.3 Prove that the protocol is sound. Give an upper bound on the soundness error (the probability that Bob will accept the proof even if the statement was false).

DSA signatures. The upcoming discussion will refer to DSA signatures. For this exam, it does not matter how DSA signatures work. The only aspects of DSA signatures that matter are: (1) The signing key (used for signing) is a uniformly random value $x \in \mathbb{Z}_p$; (2) The verification key (used for verifying signatures) is $y = g^x$. DSA signatures can also be used with respect to the generator h, in which case the verification key is $y = h^x$ instead of $y = g^x$.
Electronic voting framework. We want to perform electronic voting, using the following framework. There are n authorized voters, and it is assumed that they all vote. Each voter picks a secret DSA signing key $v_i \in \mathbb{Z}_p$, for $1 \leq i \leq n$. The verification keys g^{v_i} for DSA signatures with respect to g are public, and linked to the identity of the voter. Verification keys h^{v_i} with respect to h are not initially public, and not linked with the identity of the voters. Voting proceeds as follows.

1. Each voter i chooses their vote, and publishes it anonymously on a public board, together with a DSA signature of the vote with respect to generator h, and the corresponding verification key h^{v_i}. Votes are *not* encrypted. On the other hand, the identity of each voter is unknown.

2. A trusted authority that knows α takes all the newly published verification keys h^{v_i}, permutes them randomly, and at the same time, raises each of them to the power $1/\alpha$. The trusted authority then publishes the resulting list of verification keys g^{v_i}, together with a zero-knowledge proof that this list was obtained by randomly permuting the h^{v_i}'s, and raising them to the power $1/\alpha$.

3. Anybody can check that the initial signatures (with respect to h) are correct, that the zero-knowledge proof is correct, and that the final list of g^{v_i}'s matches the public list of verification keys g^{v_i} for authorized voters. The original votes are then accepted as having been signed by authorized voters (even if the person who signed each vote is not known).

Exercise 3. Electronic voting with oblivious shuffles.
Given two public vectors $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$, and $c, d \in \mathbb{Z}_p$, assume we know how to prove in zero-knowledge the statement: "there exists a permutation π of $\{1, \ldots, n\}$, and $c, d \in \mathbb{Z}_p$, such that for all i, $Y_i^d = X_{\pi(i)}^c$" (property $(\star)$). (Such a protocol was built in the previous exercise.) Let us call such a protocol a shuffle protocol.
Question 3.1 Explain how we can use the above protocol to realize the electronic voting framework outlined just before the exercise. Would there be any concrete issues with the voting protocol if the shuffle protocol is not zero-knowledge? If it is not sound? If it is not complete?
Assume that the trusted authority in the voting protocol is actually some sysadmin who is on holidays, and who can only interact with the public board containing the initial votes using his old smartphone, from his hotel room, with terrible wifi. Luckily, the server hosting the public board contains a trusted enclave that the smartphone can connect to, and the trusted enclave can do all the work. (The enclave is a small secure component that only the sysadmin can access.) Unfortunately, the enclave has very limited memory. After receiving all secret information from the smartphone (c, d, a trapdoor to compute x_i's, y_i's at will, and a description of π), the enclave has just enough memory to perform the shuffle protocol on two elements, and not more. As a consequence, the enclave can only provide proofs of property $(\star)$ for permutations of two elements, not all n verification keys at once.
The public board is a simple memory array: it performs no computation, and can only receive queries to read and write memory. The enclave can interact with the public board at will, but keep in mind that everything the enclave sends and receives to and from the public board is publicly visible.
Question 3.2 Propose a way for the enclave to permute all verification keys h^{x_i} on the public board according to π, without breaking the security of the voting protocol (the identity of voters should not leak). At the outcome of the permutation, each h^{x_i} must also be replaced by g^{x_i}, and your solution should provide some sort of proof that the final permutation of $Y_i = g^{v_i}$ satisfies property $(\star)$ with regard to the verification keys $X_i = h^{v_i}$ from the initial votes. (This allows the vote to be publicly verifiable: anybody can check that the original votes were signed by authorized voters only.) You are allowed to adapt the voting protocol as needed. What is the size of your proof, using as unit one element of $\mathbb{G}$ or $\mathbb{Z}_p$ (both count for 1)?
Hint: to hide values of h^{x_i} during intermediate steps of the permutation, it is enough to raise them to the power β_k, where β_k is sampled uniformly for $\mathbb{Z}_p$ by the enclave and changes for each intermediate step. This does not need to be justified. By the end of the protocol, all values should be raised to the power $1/\alpha$, so that the elements of the final permutation are verification keys g^{v_i}. This aspect of the computation (how to raise to different powers to hide intermediate values) is not the focus, and does not need to very detailed. (But bonus points if your solution only uses a polylogarithmic number of distinct powers β_k.)
Hint #2: The question leaves some room for creativity. A full answer is long, but partial answers will earn points.
Question 3.3 Now suppose the enclave is more powerful: it can execute the shuffle protocol for $n/2$ elements. Unfortunately, we still have n votes, so we still cannot shuffle all n verification keys at once. Propose a solution to realize the voting framework with a proof of size $\mathcal{O}(n)$.
Hint: this essentially amounts to doing an oblivious sort on n elements stored on a server, when the client can download $n/2$ elements at once to their local memory (for comparison, a sorting network only ever requires the client to download 2 elements at a time). One hint for a possible solution: use a structure similar to a bubble sort on a constant number of inputs, but each input is actually a block of $n/?$ elements. "?" stands for a small constant that is omitted from this hint.