6.5620 (6.875), Fall 2022

Homework # 5 **Due: Nov 16 2022, 11:59:59pm ET**

- **Typsetting:** You are encouraged to use LaTeX to typeset your solutions. You can use the following template.
- **Submissions:** Solutions should be submitted to Gradescope.
- **Reference your sources:** If you use material outside the class, please reference your sources (including papers, websites, wikipedia).
- **Acknowledge your collaborators:** Collaboration is permitted and encouraged in small groups of at most three. You must write up your solutions entirely on your own and acknowledge your collaborators.

Problems:

1. *(8 points)* **A zero-knowledge proof that a tuple is a DDH tuple.** Let p be a prime such that $p = 2^{O(\lambda)}$, where λ is a security parameter. Let g be a generator of a group $\mathbb{G}$ with order p, and let $(a, b, c) \in \mathbb{G}^3$. Let L be the language

$$L = \{(a,b,c) : \exists (x,y,z) \text{ s.t. } a = g^x, b = g^y, c = g^z \\ \text{and } z = xy \bmod p.\}$$

   (a) *(6 points)* Design a three-message proof system which allows a prover P to prove to a verifier V that a given statement $(a, b, c) \in \mathbb{G}^3$ is in L, and prove that this proof system is complete, sound, and *honest-verifier perfect* zero-knowledge. Your completeness error should be 0 (the honest prover is always accepted by the honest verifier). Your soundness error (maximum probability that a cheating prover can win if $(a, b, c) \notin L$) should be $\frac{1}{p}$.

   Hint. A proof system in which the prover simply reveals x to the verifier, and the verifier checks that $(g^y)^x = g^z$, is obviously complete and sound, and equally obviously not zero-knowledge. A first attempt at designing a proof system which is zero-knowledge might be as follows: the prover 'one time pads' the secret x using a random element $r \leftarrow \mathbb{Z}_p$, and sends $\alpha = r + x$ and $\beta = g^{yr}$ to the verifier in a single message. The verifier can then check that $(g^y)^\alpha = \beta \cdot g^z$. Note that completeness holds because $(g^y)^{r+x} = g^{yr+yx} = g^{yr} \cdot g^z$. Zero-knowledge also holds (convince yourself by constructing a simulator), but soundness does *not* hold: because this is a non-interactive proof system, a cheating prover can run exactly the same strategy as the simulator! Think about how you can fix this

problem using interaction. A natural paradigm for a three-message protocol is the commitment-challenge-response paradigm; how can you combine this paradigm with the broken idea just outlined?

Hint 2. A two-message solution is possible in this part, and perfectly acceptable; however, your life might be easier in the rest of the problem set if you come up with a three-message system with the structure described in problem 2.

(b) *(2 points)* Modify your proof system from (a) so that it has soundness 1/2 and so that it is malicious verifier perfect zero-knowledge, and prove these two properties. (No need to re-prove completeness.)

2. *(4 points)* **A zero-knowledge proof for the OR of two statements.**

Let L be a language. Suppose there is a three-message honest-verifier perfect zero-knowledge protocol $\mathcal{P}$ for proving that $x \in L$ that has completeness error 0 and soundness error s. In addition, suppose that $\mathcal{P}$ has the following structure:

- Message 1: the prover P sends a message a to the verifier.
- Message 2: the verifier sends a challenge c to the prover which is chosen uniformly at random from $\mathbb{G}$, where $\mathbb{G}$ is some group.
- Message 3: the prover sends a message $z(a, c)$ to the verifier.
- The verifier computes a predicate $f(x, a, c, z)$ and accepts iff $f(x, a, c, z) = 1$.

Finally, suppose that, for all $x \in \{0,1\}^*$,

$$\Pr[f(x, a, c, z) = 1 : (a, c, z) \leftarrow \mathsf{Sim}^V(1^\lambda, x)] = 1,$$

where $\mathsf{Sim}^V(1^\lambda, x)$ is the zero-knowledge simulator for $\mathcal{P}$ run on input x and security parameter λ with oracle access to the honest verifier V.

(This is a very common structure for zero-knowledge protocols for NP languages.)

(a) Design a three-message honest-verifier perfect zero-knowledge protocol $\mathcal{P}'$ which allows a prover P to prove to a verifier V that *at least one of* two statements x, x' is in L, assuming nothing more than the existence of the protocol $\mathcal{P}$. Prove that your protocol is complete, sound and zero-knowledge. Your completeness error should be 0. Your soundness error should be $s^2 \cdot |\mathbb{G}|$.

Hint. A naive attempt would be to ask the prover to produce proofs (using $\mathcal{P}$) for both the statements x, x' if they are both in L, and otherwise to produce a proof for the one which is in L. However, we do not want to reveal to the verifier even the bit which says whether both x and x' are in L, or whether only one is in L. What if the verifier sets up the proof in such a way that it appears to be asking for proofs for both x and x', but on purpose allows the honest prover to *simulate* one out of two proofs? How can we arrange things so that the verifier cannot tell

which proof is simulated? (Remember that, in order to simulate a proof transcript for a no-instance, it is critical that the simulator can pick c at the same time that it generates the rest of the transcript.)

(b) (*Optional; ungraded*) Suppose now that $\mathcal{P}$ is malicious-verifier (computationally, statistically, or perfectly) zero-knowledge, and that it retains all the other properties described in part (a). How would you design a protocol $\mathcal{P}'$ that has completeness error 0, soundness error $s^2 \cdot |\mathbb{G}|$, and the same flavour of malicious-verifier zero-knowledge as $\mathcal{P}$, such that $\mathcal{P}'$ allows a prover P to prove that at least one of x, x' is in L, *assuming that x, x' are always drawn independently from a distribution $\mathcal{D}$ such that yes– and no-instances are computationally indistinguishable*? That is, given $x \leftarrow \mathcal{D}$, the probability that a PPT adversary correctly guesses whether $x \in L$ or $x \notin L$ is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$.

While this part is optional, it might help you in problem 3(d) if you understand how this would be done. (One step in the proof is slightly subtle, in a similar way to 3(d); if you get stuck there, you might find it easier to think about this simpler version first.)

3. *(16 points)* **An efficient zero-knowledge proof for CIRCUIT-SAT.**

CIRCUIT-SAT is the problem of deciding, given a circuit C,[1] whether C is satisfiable or not. Formally, the language L associated with CIRCUIT-SAT is the language

$$L = \{C : \exists x \text{ s.t. } C(x) = 1\}.$$

CIRCUIT-SAT is NP-complete. In class, we saw one zero-knowledge proof system (the GMW proof system) for an NP-complete problem, namely graph 3-colouring; however, this proof system has $1 - \frac{1}{|E|}$ soundness error 'out of the box', and in order to achieve constant soundness error we would have to repeat it $\Omega(|E|)$ times. One execution of the proof system requires the prover to send $|E|$ commitments to the verifier in the first step, so the total amount of communication between prover and verifier required in order to achieve constant soundness is $\Omega(|E|^2 \cdot \text{poly}(\lambda))$, where λ is a security parameter that controls how long the commitment strings are. Note that, in the worst case, the number of edges $|E|$ could be as large as $|V|^2$.

In this problem, we will build an honest-verifier zero-knowledge proof system based on DDH for CIRCUIT-SAT that requires just $O(n \cdot \lambda)$ bits of communication between the prover and the verifier, where n is the number of gates in the circuit instance and λ is

[1] In this problem, we assume that circuits are represented as directed acyclic graphs in which each vertex represents a gate, and wherein each vertex can have fan-in at most 2 and unbounded fan-out. Each vertex is labelled with a gate type from a constantly sized gate set. Any such DAG can be represented in $O(n \log n)$ bits, where n is the number of gates in the circuit: for each vertex (gate) in the graph, we simply specify a gate type in addition to the ≤ 2 vertices that point into it.

a security parameter. (Malicious-verifier zero-knowledge is similar, but requires a little more work, so we do not ask you to do it here.)

(a) *(2 points)* We will start by building a commitment scheme from DDH. Fix a generator g of a group $\mathbb{G}$ of prime order p, with $p = 2^{O(\lambda)}$ for security parameter λ. Suppose that the sender wishes to commit to ℓ values $z_1, \ldots, z_\ell \in \mathbb{Z}_p$. Consider the commitment scheme in which the sender chooses $x \leftarrow \mathbb{Z}_p$ and $r_1, \ldots, r_\ell \leftarrow \mathbb{Z}_p$ uniformly at random, and sends g^x along with $\{ c_i = (g^{r_i}, g^{xr_i} \cdot g^{z_i}) \}_{i \in [\ell]}$ to the receiver. (Note that g^x serves as something like a 'public key' in this commitment scheme, and we may refer to it as such informally throughout the problem.) To open a commitment, the sender sends $\{r_i, z_i\}_{i \in [\ell]}$, and the recipient checks that $c_i = (g^{r_i}, (g^x)^{r_i} \cdot g^{z_i})$ for all $i \in [\ell]$. Prove that this commitment scheme is correct, *perfectly binding* and *computationally hiding* (assuming the DDH assumption in $\mathbb{G}$).

Remark 1. Note that this commitment scheme is *additively homomorphic* under componentwise multiplication. That is, if $c_1 = (g^{r_1}, g^{xr_1} \cdot g^{z_1})$ and $c_2 = (g^{r_2}, g^{xr_2} \cdot g^{z_2})$ are valid commitments to z_1 and z_2 respectively under the public key g^x, then $c_1 \cdot c_2 := (g^{r_1} \cdot g^{r_2}\ ,\ (g^{xr_1} \cdot g^{z_1}) \cdot (g^{xr_2} \cdot g^{z_2}))$ is a valid commitment to $z_1 + z_2 \bmod p$.

(b) *(4 points)* Now we will introduce another essential building block. As in part (a), fix a generator g of a group $\mathbb{G}$ of prime order p, with $p = 2^{O(\lambda)}$. Suppose that $c = (g^x, g^r, g^{xr} \cdot g^z) := (a, b, c)$ is a commitment to value $z \in \mathbb{Z}_p$ which has been generated by the sender from the commitment scheme in part (a). Consider the language

$$L = \{(a, b, c) \in \mathbb{G}^2 : \exists x, r \in \mathbb{Z}_p, z \in \{0, 1\} \text{ s.t. } a = g^x, b = g^r, c = g^{xr} \cdot g^z.\}$$

In other words, L is the language of $(a, b, c) \in \mathbb{G}^2$ such that (a, b, c) is a valid commitment to a binary value $z \in \{0, 1\}$. Construct a three-message honest-verifier perfect zero-knowledge proof system for this language that has completeness error 0 and inverse-exponential soundness error. Prove that your proof system has these properties.

Hint. Combine problems 1 and 2 on this set.

(c) *(Optional; ungraded.* ***You can use this subpart without proof in subsequent subparts.****)* Recall that NAND gates are universal. Prove that, for $b_1, b_2, b_3 \in \{0, 1\}$,

$$(b_1 \text{ NAND } b_2) = b_3 \iff (b_1 + b_2 + 2b_3 - 2) \in \{0, 1\}.$$

(Arithmetic on the right-hand-side is normal integer arithmetic, not modulo 2 arithmetic.)

(d) *(10 points)* Now we're ready to go! Assume that $p = 2^{O(\lambda)}$, where λ is our security parameter. Construct a zero-knowledge proof system for CIRCUIT-SAT (based on DDH in the order-p group $\mathbb{G}$) that uses $O(n \cdot \lambda)$ bits of communication between the prover and the verifier, where n is the number of gates in the CIRCUIT-SAT

instance C. For simplicity, you may assume that C is made up entirely of NAND gates. **You may also assume that all the gates in C have fan-out $O(1)$** (and fan-in 2, as usual); this is justified because, in fact, deciding the satisfiability of *formulas* (e.g. 3SAT formulas) is already NP-complete.

Prove that your proof system has *perfect completeness* and *constant soundness error*, and that it satisfies *honest-verifier computational zero-knowledge*.

Hint 1. Think about the canonical reduction from CIRCUIT-SAT to 3SAT (also known as the Cook-Levin reduction, if you haven't seen it before). Similar ideas (in particular, the tableau idea) will be useful here!

Hint 2. How do we make sure the circuit C actually outputs 1? Observe that the commitment $c = (1, g)$ is a valid commitment to $z = 1$ under the commitment scheme from part (a), and that it is very easy to verify that it is a commitment to $z = 1$.

Warning. This is a complex proof with many moving parts. (Security proofs for real cryptographic protocols are usually like this. If you're curious, you might want to try formally proving the security of GMW. You may find that it's more subtle than you expected!) Make sure that, every time you hybrid from one distribution to another, or, equivalently, replace something with something else that's computationally indistinguishable, you *justify the step with a reduction* to the appropriate security assumption. (You can, of course, do reductions to your work from previous parts of this problem set without repeating the proofs, as long as you state what you're reducing to with an appropriate degree of formality.)

Here is an example of a subtlety that you might want to watch out for. Consider a toy zero-knowledge protocol that starts with the prover sending a commitment $c := \mathsf{commit}(z; r)$ to the verifier. Suppose that, after the prover sends c, it proves (using some zero-knowledge protocol) to the verifier that there exists (z, r) such that $c = \mathsf{commit}(z; r)$ and z satisfies some predicate (for simplicity, let's say the predicate is $z \neq 0$). Our first attempt at analysing the zero-knowledge property of this protocol might be as follows: construct a hybrid argument in which the first step is to 'replace' c with a commitment to some fixed string, e.g. 0, using the hiding property of the commitment scheme, and our second step is to replace the zero-knowledge proof with a simulation. More specifically, H_0 is the original distribution produced by the honest prover, H_1 is the distribution in which the prover always commits to 0 but does everything else the same way, and H_2 is the distribution in which the prover always commits to 0 and simulates the zero-knowledge proof. H_2 is clearly simulable, so (naively) we are done. You might find, however, that doing the reduction (to the hiding property of the commitment scheme) which proves that $H_0 \approx_c H_1$, or even *defining* H_1 precisely, is a little tricky. In particular, if the prover always commits to $z = 0$, then the statement that it is supposed to prove in zero-knowledge (that $z \neq 0$) becomes *false*. How, then, is it going to 'do everything else the same way' (i.e. prove the statement it's supposed to prove) in H_1? It can't produce an acceptable proof for a statement

that's false! Or can it?

(In this toy example, the simulator could just pick some other fixed string instead of 0 to commit to. However, you might find that you can't do this so easily in the analysis of your real protocol.)

(e) *(Optional; ungraded.)* You may wish to convince yourself that we're comparing apples to apples in comparing the efficiency of this proof system with the efficiency of GMW. In particular, starting from any 3SAT instance ϕ with n variables and $O(n)$ clauses (this regime is where the NP-complete instances lie), compute how many edges there are in the graph which the canonical reduction from 3SAT to 3COL produces. Then, starting from ϕ, compute how many NAND gates are required to compute ϕ using a circuit made up entirely of NAND gates. Finally, compare the efficiency of GMW (which we stated in terms of $|E|$) to the efficiency of your proof system from (d) (which we stated in terms of the number of NAND gates in the circuit C).

4. *(9 points)* **Circular security.**

In this problem, we consider a security property (of encryption schemes) called circular security: namely, security even when an adversary is given an encryption of the secret key sk. One variant of circular security for encryption schemes is defined as follows.

Definition 1. A secret key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be circular secure if for all p.p.t. oracle algorithms $\mathcal{A}^{\mathcal{O}(\cdot)}$, we have that

$$\mathop{\mathbf{E}}_{sk \leftarrow_R \mathsf{Gen}(1^n)}[\mathcal{A}^{\mathrm{Left}_s k(\cdot)}(c^*)] = \mathop{\mathbf{E}}_{sk \leftarrow_R \mathsf{Gen}(1^n)}[\mathcal{A}^{\mathrm{Right}_s k(\cdot)}(c^*)] + \mathrm{negl}(n),$$

where $c^* \leftarrow \mathsf{Enc}(sk, sk)$, $\mathsf{Left}_s k(\cdot)$ is an oracle which on input (m_L, m_R) outputs an encryption of m_L, and $\mathsf{Right}_s k(\cdot)$ is on oracle which on input (m_L, m_R) outputs an encryption of m_R.

An equivalent definition is as follows.

Definition 2. A public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be circular secure if any p.p.t. algorithm $\mathcal{A}$ wins the following game (interacting with a challenger) with probability at most $\frac{1}{2} + \mathrm{negl}(n)$:

(i.) The challenger samples $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and sends (pk, c^*) to $\mathcal{A}$, where $c^* \leftarrow \mathsf{Enc}(pk, sk)$ is a ciphertext corresponding to the secret key.

(ii.) $\mathcal{A}$ sends two messages (m_0, m_1) to the challenger.

(iii.) The challenger selects $b \leftarrow_R \{0, 1\}$ and sends $c_b \leftarrow \mathsf{Enc}(pk, m_b)$ to $\mathcal{A}$.

(iv.) $\mathcal{A}$ outputs a bit $\tilde{b}$. We say that $\mathcal{A}$ wins if $\tilde{b} = b$.

(a) *(6 points)* It turns out that not every IND-CPA secure public key encryption scheme is also circularly secure. Construct a public-key encryption scheme which is IND-CPA secure but not circularly secure, relying only on the existence of public-key encryption schemes. Prove that your scheme is IND-CPA secure but not circularly secure.

(b) *(3 points)* In this part, you will show that a variant of the LWE-based secret key encryption we saw in class does satisfy circular security (under an LWE-like assumption). In particular, we consider a variant of the LWE problem where the secret s is a uniformly random binary string.

Definition 3 (LWE* assumption)**.** The LWE* assumption with error distribution χ states that the following two distributions are computationally indistinguishable:

$$\{\mathbf{s} \leftarrow_R \{0,1\}^n, \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{e} \leftarrow_R \chi^m : (\mathbf{A}, \mathbf{s}^T\mathbf{A}+\mathbf{e}^T)\} \approx_c \{\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{b} \leftarrow_R \mathbb{Z}_q^m : (\mathbf{A}, \mathbf{b})\}.$$

Under the LWE* assumption (with m any polynomial in n, and with error distribution χ), prove that the n bit encryption scheme defined by

$$\mathsf{Enc}(\mathbf{s}, \mathbf{m} \in \{0,1\}^n; \mathbf{R} \leftarrow_R \mathbb{Z}_q^{n\times n}, \mathbf{e} \leftarrow \chi^n) = (\mathbf{R}, \mathbf{s}^T\mathbf{R} + \mathbf{e}^T + \left\lfloor \frac{q}{2} \right\rfloor \mathbf{m}^T)$$

is a circularly secure secret key encryption scheme.

Hint. Show that it is possible to generate an encryption of s given only an encryption of 0.