6.875: Cryptography and Cryptanalysis **November 10, 2021**
Last updated: November 11, 2021

Problem Set 5

Instructor: Vinod Vaikuntanathan

TAs: Lali Devadas, Aparna Gupte, Sacha Servan-Schreiber

Instructions.

- **When:** This problem set is due on **November 22, 2021** before **11pm ET**.
- **How:** You should use LaTeX to type up your solutions (you can use our LaTeX template from the course webpage). Solutions should be uploaded on Gradescope as a single pdf file.
- **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*.
- **Reference your sources:** If you use material from outside the lectures, you must reference your sources (papers, websites, wikipedia, ...).
- **When in doubt, ask questions:** Use Piazza or the TA office hours for questions about the problem set. See the course webpage for the timings.

Problem 1. CCA transformations. Suppose $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an IND-CCA-secure public key encryption scheme.

> **For each of the following, state whether the resulting scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ is (1) IND-CPA secure and (2) IND-CCA secure.**

(a)
- $\mathsf{Gen}'(1^\lambda) = \mathsf{Gen}(1^\lambda)$ outputs (pk, sk).
- $\mathsf{Enc}'(pk, x\|y) = (\mathsf{Enc}(pk, x)\|\mathsf{Enc}(pk, y))$ where $|x| = |y|$.
- $\mathsf{Dec}'(sk, c_1\|c_2) = \mathsf{Dec}(sk, c_1)\|\mathsf{Dec}(sk, c_2)$.

(b)
- $\mathsf{Gen}'(1^\lambda) = \mathsf{Gen}(1^\lambda)$ outputs (pk, sk).
- $\mathsf{Enc}'(pk, x) = \mathsf{Enc}(pk, r)\|r \oplus x$ where r is a random bit-string.
- $\mathsf{Dec}'(sk, c_1\|c_2) = \mathsf{Dec}(sk, c_1) \oplus c_2$.

Problem 2. Homomorphic encryption from LWE. In this problem we will explore a cool encryption scheme that shares some ideas with "fully homomorphic encryption," and is based on the LWE hardness assumption (first introduced by Oded Regev in 2005). While this scheme is **not** *fully* homomorphic, it is additively homomorphic (two encrypted messages can be efficiently "added" together to produce a ciphertext of their sum) and also supports *one* homomorphic multiplication (two encrypted messages can be efficiently "multiplied" together to obtain a ciphertext encrypting the product of the two messages, but the resulting ciphertext cannot be "multiplied" again).

Learning with Errors (LWE). For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^{n \times 1}$ and an error bound B (which we think of as $\ll q$), let $\mathcal{D}_{\mathbf{s},B}$ be the distribution obtained by choosing a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ uniformly at random and a noise term $\mathbf{x} \xleftarrow{R} \{-B, \ldots, B\}^{m \times 1}$, and outputting

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times 1}.$$

Definition 1 (The LWE problem)**.** *Let n and m be positive integers. For an integer $q = q(n)$ and an error bound $B = B(n)$ over $\mathbb{Z}$, the learning with errors problem $\mathsf{LWE}_{n,m,q,B}$ is defined as follows.*

- ***Computational variant, denoted*** $\mathsf{LWE}_{n,m,q,B}$**.** *For a uniformly random $\mathbf{s} \leftarrow \mathbb{Z}_q^{n \times 1}$, given $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x}) \leftarrow \mathcal{D}_{\mathbf{s},B}$ find $\mathbf{s}$.*
- ***Decisional variant, denoted*** $\mathsf{distLWE}_{n,m,q,B}$**.** *Distinguish between the following two probability distributions: (1) pick a uniformly random $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^{n \times 1}$, and output an instance chosen according to $\mathcal{D}_{\mathbf{s},B}$; and (2) an instance chosen according to the uniform distribution over $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times 1}$.*

The security of our scheme will depend on the hardness of $\mathsf{distLWE}_{n,m,q,B}$.

Key sampling. Before we dive into the encryption scheme, we first describe a way to sample a "special" LWE matrix which we use in our key generation algorithm. Specifically, we sample a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{t} \in \mathbb{Z}_q^{m \times 1}$ as follows.

```
Sample($n, m, q$):
1. $\mathbf{A}' \xleftarrow{R} \mathbb{Z}_q^{(m-1) \times n}$
2. $\mathbf{t}' \xleftarrow{R} \{0,1\}^{(m-1) \times 1}$
3. $\mathbf{t} \leftarrow \begin{bmatrix} \mathbf{t}' \\ 1 \end{bmatrix}$                                  // $\mathbf{t} \in \mathbb{Z}_q^{m \times 1}$
4. $\mathbf{A} \leftarrow \begin{bmatrix} \mathbf{A}' \\ -\mathbf{t}'^{\top}\mathbf{A}' \end{bmatrix}$        // Concatenate $-\mathbf{t}'^{\top}\mathbf{A}' \in \mathbb{Z}_q^{1 \times n}$ to $\mathbf{A}'$
5. return $(\mathbf{A}, \mathbf{t})$                    // $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{t} \in \mathbb{Z}_q^{m \times 1}$

Note: here $\mathbf{v}^{\top}$ denotes the transpose of the vector $\mathbf{v}$.
```

The matrix $\mathbf{A}$ and vector $\mathbf{t}$ that Sample outputs have the following properties:

- $\mathbf{t}^{\top}\mathbf{A} = \mathbf{0} \in \mathbb{Z}_q^{1 \times n}$;
- The entries of $\mathbf{t}$ are small, namely, $\mathbf{t}[i] \in \{0, 1\}$ for all $i \in [m]$.
- $\mathbf{A}$ is *statistically close* to a uniformly random $m \times n$ matrix.

The last property follows by the leftover hash lemma as long as $m = \Omega(n \log q)$ is "sufficiently large"; you can assume this property holds for the rest of this problem.

Parameters. We will let n denote the security parameter. We set $m, q = \mathsf{poly}(n)$ with q an odd prime. Our message space is $\{0, 1\}$. The public keys are matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, the secret keys are vectors $\mathbf{t} \in \mathbb{Z}_q^{m \times 1}$, and ciphertexts are vectors $\mathbf{c} \in \mathbb{Z}_q^{m \times 1}$.

```
$\mathsf{Gen}(1^\lambda)$
    $(\mathbf{A}, \mathbf{t}) \leftarrow \mathsf{Sample}(n, m, q)$
    $pk \leftarrow \mathbf{A}$
    $sk \leftarrow (\mathbf{A}, \mathbf{t})$
    return $(pk, sk)$
```

```
$\mathsf{Enc}(pk, b \in \{0,1\})$
    parse $pk = \mathbf{A}$
    $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^{n\times 1}$
    $\mathbf{x} \xleftarrow{R} \{-B, \dots, B\}^{m\times 1}$
    $\mathbf{m} \leftarrow [0\ 0\ \cdots\ 0\ b]^\top \in \mathbb{Z}_q^{m\times 1}$
    $\mathbf{c} \leftarrow \mathbf{A}\mathbf{s} + 2\mathbf{x} + \mathbf{m} \pmod q$
    return $\mathbf{c}$
```

```
$\mathsf{Dec}(pk, sk, \mathbf{c})$
    parse $sk = (\mathbf{A}, \mathbf{t})$
    $\tilde{b} \leftarrow \mathbf{t}^\top \mathbf{c} \bmod q$
    $b \leftarrow \tilde{b} \bmod 2$
    return $b$
```

Figure 1: LWE-based Homomorphic Encryption Scheme

The encryption scheme. We use the Sample procedure described above to define our encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

(a) **Given n, m, and B, find a suitable modulus q such that the encryption scheme in Figure 1 is correct (with probability 1). Prove correctness of the scheme under these parameters.**

(b) **Prove that the scheme is IND-CPA secure via a reduction to the $\mathsf{distLWE}_{n,m,q,B}$ problem.**

(c) **Prove that the encryption scheme of Figure 1 is *additively homomorphic.* That is, let $\mathbf{c_0}$ and $\mathbf{c_1}$ be encryptions of bits b_0 and b_1, respectively. Then there exists an efficiently computable function $\mathsf{ADD}(\mathbf{c_0}, \mathbf{c_1})$ which outputs ciphertext $\mathbf{c}'$ that is an encryption of $b_0 + b_1 \pmod 2$.**

(d) **Prove that the encryption scheme of Figure 1 is *one-time multiplicatively homomorphic.* That is, let $\mathbf{c_0}$ and $\mathbf{c_1}$ be encryptions of bits b_0 and b_1, respectively. Then, there exists an efficiently computable function $\mathsf{MUL}(\mathbf{c_0}, \mathbf{c_1})$ which outputs a ciphertext $\mathbf{c}'$ that is an encryption of $b_0 b_1 \pmod 2$.**

Hint: the format of the product ciphertext could be different from a fresh encryption, and consequently the decryption algorithm could be different as well. However, you should be able to efficiently recover $b_0 b_1 \pmod 2$ from the ciphertext $\mathbf{c}'$ using the secret key $\mathbf{t}$.

(e) **Show that you can homomorphically evaluate any polynomial of degree 2 on a sequence of encrypted bits $b_0, b_1, \dots, b_N$ using the above scheme. Specifically, show how to set q in terms of n, m, and B such that your scheme is correct and the size of your evaluated ciphertext grows only polylogarithmically with N. For example, consider the polynomial $P(x_1, x_2, x_3) = x_1^2 + x_3^2 + x_1 + x_2 + x_3 + 1 \pmod 2$. Here $N = 3$ and the degree of P is at most 2. You must show that it is possible to evaluate such a P, i.e., generate a ciphertext for $P(x_1, x_2, x_3)$ from *encryptions* of inputs x_1, x_2 and x_3.**

Problem 3. A bad definition of Zero-knowledge?

In 6.875, Alice learned that an interactive proof system $\langle P, V\rangle$ for a language $\mathcal{L}$ is computationally zero-knowledge if there exists an efficient simulator S such that for all $x \in \mathcal{L}$,

$$\mathsf{view}_V(\langle P, V\rangle(1^\lambda, x)) \approx_c S(1^\lambda, x)$$

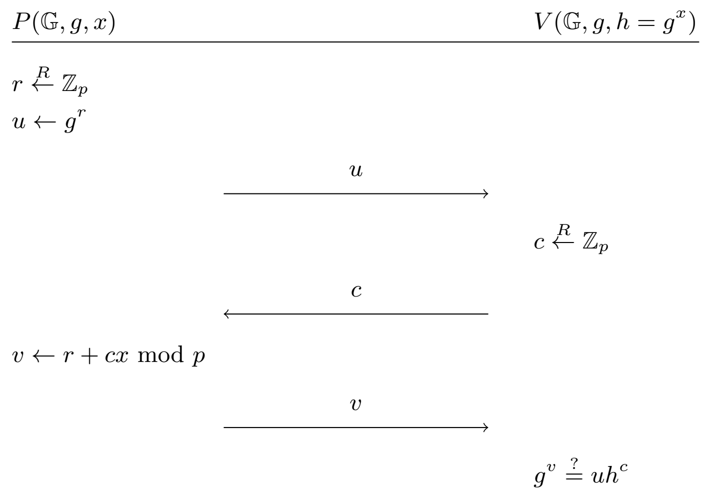
for $\mathsf{view}_V(\langle P, V\rangle(1^\lambda, x)) = (T, \mathsf{coins}_V)$ where T is the transcript of messages between P and V during the protocol and coins_V is the randomness used by V during the protocol.

She explains the definition to Bob, but Bob thinks that including V's randomness in the view is unnecessarily complicated. He claims that it is equivalent to say that a proof system $\langle P, V\rangle$ is computationally zero-knowledge if there exists an efficient simulator S such that for all $x \in \mathcal{L}$,

$$T \approx_c S(1^\lambda, x).$$

Is Bob correct? Either show that his definition is equivalent to Alice's (the one we learned in class) or show a counterexample, i.e., a proof system which satisfies completeness, soundness, and Bob's definition of zero-knowledge but not Alice's definition of zero-knowledge.

Problem 4. Zero-knowledge Proofs of Knowledge (ZKPoK) Let $\mathbb{G}$ be a group of order p (for some large λ-bit prime p) and let g be a generator of $\mathbb{G}$. Given an instance g^x, can Alice convince Bob she knows x without revealing anything about x to Bob? Recall the protocol from lecture (see Figure 2), where Alice acts as the prover and Bob acts as the verifier. For simplicity, we assume that the prover (Alice) always convinces the honest verifier (Bob) with probability 1.



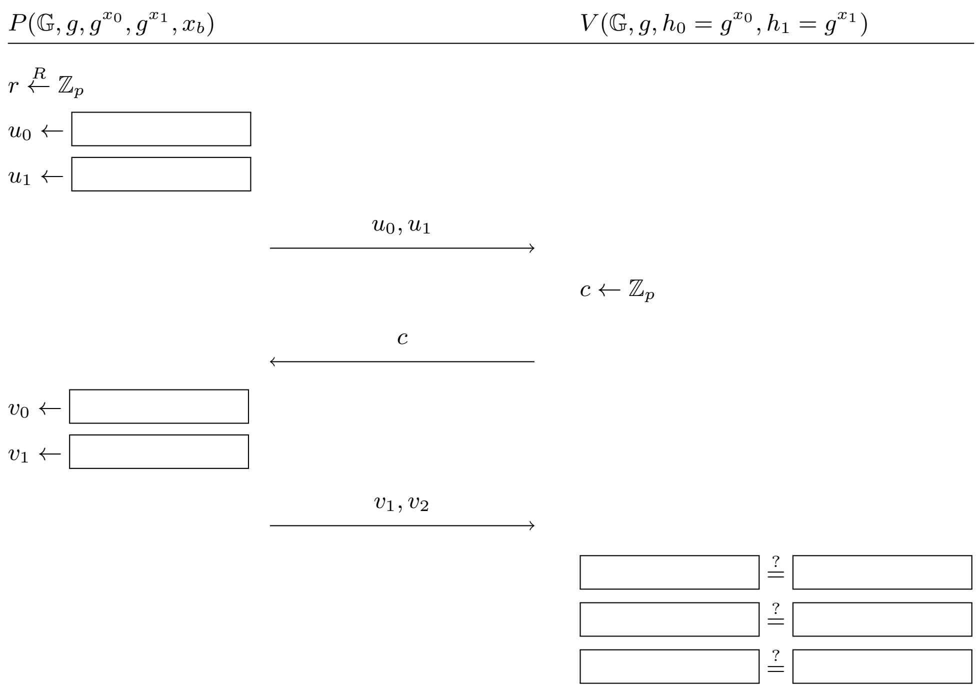
$P(\mathbb{G}, g, x)$
$V(\mathbb{G}, g, h = g^x)$
$r \overset{R}{\leftarrow} \mathbb{Z}_p$
$u \leftarrow g^r$
u
$c \overset{R}{\leftarrow} \mathbb{Z}_p$
c
$v \leftarrow r + cx \bmod p$
v
$g^v \overset{?}{=} uh^c$

Figure 2: Honest-verifier Zero-knowledge proof of Discrete Logarithm Knowledge.

For convenience, we provide the analysis for *completeness*, *proof of knowledge*, and *(honest-verifier) zero-knowledge* from lecture in Appendix A.

Suppose that Alice and Bob have two values g^{x_0} and g^{x_1}. Alice knows x_b for $b \in \{0, 1\}$ but does not know x_{1-b}. Can Alice convince Bob in zero knowledge that she knows one of the two discrete logarithms? In particular, Bob should not learn which of the two discrete logarithms Alice knows.

Design an honest-verifier zero-knowledge proof of knowledge of 1-out-of-2 discrete logarithms (we provide a stencil in Figure 3 to get you started). Remember, a zero-knowledge proof of knowledge must satisfy three properties: *completeness*, *proof of knowledge*, and *zero-knowledge*. In your proof of the proof of knowledge property, you may assume that the prover P convinces the verifier V with probability 1.



$P(\mathbb{G}, g, g^{x_0}, g^{x_1}, x_b)$
$V(\mathbb{G}, g, h_0 = g^{x_0}, h_1 = g^{x_1})$
$r \xleftarrow{R} \mathbb{Z}_p$
$u_0 \leftarrow \boxed{}$
$u_1 \leftarrow \boxed{}$
u_0, u_1
$c \leftarrow \mathbb{Z}_p$
c
$v_0 \leftarrow \boxed{}$
$v_1 \leftarrow \boxed{}$
v_1, v_2
$\boxed{} \overset{?}{=} \boxed{}$
$\boxed{} \overset{?}{=} \boxed{}$
$\boxed{} \overset{?}{=} \boxed{}$

Figure 3: Zero-knowledge proof of 1-out-of-2 Discrete Logarithm Knowledge.

A Analysis of DL ZKPoK from lecture

We briefly go over the proofs of *completeness*, *proof of knowledge*, and *(honest-verifier) zero-knowledge* for Figure 2.

Completeness. The verifier always accepts a valid proof because:

$$g^v = g^{r+cx} = g^r g^{cx} = u(g^x)^c = uh^c.$$

Proof of knowledge. We build an *extractor* $\mathcal{E}$ which runs the prover to recover the discrete logarithm of $h = g^x$ as follows. On input h,

1. Run P to get commitment $u = g^r$.
2. Return challenge $c_0 \overset{R}{\leftarrow} \mathbb{Z}_p$ and receive v_0 from P.
3. Rewind P, send $c_1 \overset{R}{\leftarrow} \mathbb{Z}_p$, and receive v_1 from P.
4. Return $x = (v_0 - v_1)(c_0 - c_1)^{-1} \bmod p$.

Analysis: Because P always returns v_0 and v_1 which satisfy the relation $g^{v_0} = uh^{c_0}$ and $g^{v_1} = uh^{c_1}$, respectively, it holds that $g^{v_0} = g^{r+c_0x} = uh^{c_0}$ and $g^{v_1} = g^{r+c_1x} = uh^{c_1}$. As such, $(v_0 - v_1) = c_0x - c_1x = x(c_0 - c_1)$. Hence, $x = (v_0 - v_1)(c_0 - c_1)^{-1}$, as required, except in the case where $c_0 = c_1$. The probability of sampling $c_0 = c_1$ is $\frac{1}{p}$, which is negligible. Therefore, we get that the extractor succeeds in recovering the discrete logarithm with probability $1 - \mathsf{negl}(\lambda)$, and hence we have negligible extraction error.

Honest-verifier Zero-knowledge. We construct a simulator $\mathcal{S}$ which generates an indistinguishable view by running the protocol "in-reverse." On input $(\mathbb{G}, g, h)$:

1. Sample random $r, c \overset{R}{\leftarrow} \mathbb{Z}_p$.
2. Set $u \leftarrow g^r h^{-c} \in \mathbb{G}$.
3. Set $v \leftarrow r$.
4. Output $\{u, c, v\}$.

Analysis: First, observe that $g^v = g^r = g^{r+xc-xc} = g^{r-xc}h^c = uh^c$. Because the verifier V is honest, c follows the same (uniform) distribution in $\mathbb{Z}_p$ as in the actual protocol. Likewise for the distribution of u and v, which are both uniformly random.