6.875: Cryptography and Cryptanalysis **November 10, 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 $|x| = |y|$.
- $\mathsf{Dec}'(sk, c_1 \| c_2) = \mathsf{Dec}(sk, c_1) \| \mathsf{Dec}(sk, c_2)$.

Solution

This scheme is CPA secure but not CCA secure. The CPA security of this scheme follows from a hybrid argument seen in class (one message security implies many message security for public key encryption). To see it is not CCA secure, one can construct the following adversary, which utilizes mix-and-match attacks.

Let $x_1 \neq x_2$ and $y_1 \neq y_2$. The adversary, on public key pk, outputs $m_1 = (x_1, y_1)$, $m_2 = (x_1, y_2)$ and gets back (c_1, c_2). The adversary also computes $\mathsf{Enc}(pk, (x_2, y_1))$ and gets (c_3, c_4). The adversary then queries the decryption oracle on (c_3, c_2) and gets back (x, y), output 0 when $y = y_1$ and 1 otherwise. This adversary wins the CCA game with probability 1.

(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.
- $\mathsf{Dec}'(sk, c_1 \| c_2) = \mathsf{Dec}(sk, c_1) \oplus c_2$.

Solution
This scheme is CPA secure but not CCA secure. The new scheme is malleable, given a ciphertext of m, the adversary can construct a ciphertext of $m \oplus 1$ simply by xor-ing the second half with a 1. Suppose the new scheme is not CPA secure and there is an adversary A wins the CPA game for the new scheme with non-negligible probability. Consider the hybrid game that the challenge ciphertext is generated by randomly choosing r then outputs $(\mathsf{Enc}(0), r \oplus x)$. The winning probability of $\mathcal{A}$ in the CPA game should be negligible close to the winning probability in the hybrid game due to the indistinguishability of the ciphertexts. However, in the hybrid game, the challenge hides x perfectly, i.e. there is no adversary who can wins the game with probability greater than $1/2$, even for a computationally unbounded adversary.
The new scheme is not CCA secure since we can construct an adversary given ciphertext $c = (c_1, c_2)$ of some unknown message m and access to the decryption oracle (can't access at input c), recovers m. The adversary queries the decryption on $(c_1, c_2 \oplus 1)$ and gets back m , then outputs $m = m \oplus 1$. Note this suffices since an adversary in CCA game can use this adversary to recover the message then compare with m_0, m_1.

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

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

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 $(\mathbf{A}, \mathbf{As} + \mathbf{x}) \leftarrow \mathcal{D}_{\mathbf{s},B}$ 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 $\mathbf{c} \in \mathbb{Z}_q^{m\times 1}$.

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

Gen(1^λ)

1 : $(\mathbf{A}, \mathbf{t}) \leftarrow \mathsf{Sample}(n, m, q)$
2 : $pk \leftarrow \mathbf{A}$
3 : $sk \leftarrow (\mathbf{A}, \mathbf{t})$
4 : **return** (pk, sk)

Enc$(pk, b \in \{0,1\})$

1 : **parse** $pk = \mathbf{A}$
2 : $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^{n\times 1}$
3 : $\mathbf{x} \xleftarrow{R} \{-B, \ldots, B\}^{m\times 1}$
4 : $\mathbf{m} \leftarrow [0\ 0\ \cdots\ 0\ b]^\top \in \mathbb{Z}_q^{m\times 1}$
5 : $\mathbf{c} \leftarrow \mathbf{A}\mathbf{s} + 2\mathbf{x} + \mathbf{m} \pmod q$
6 : **return** $\mathbf{c}$

Dec$(pk, sk, \mathbf{c})$

1 : **parse** $sk = (\mathbf{A}, \mathbf{t})$
2 : $\tilde{b} \leftarrow \mathbf{t}^\top \mathbf{c} \bmod q$
3 : $b \leftarrow \tilde{b} \bmod 2$
4 : **return** b

Figure 1: LWE-based Homomorphic Encryption Scheme

Note about correctness: The computation $\tilde{b} \leftarrow \mathbf{t}^\top \mathbf{c}$ happens mod q, but we interpret $\tilde{b}$ as a value in $(-\frac{q}{2}, \frac{q}{2}]$ instead of in $[0, q)$. Then when we compute $b \leftarrow \tilde{b} \bmod 2$, even if $\mathbf{t}^\top 2\mathbf{x}$ was negative it will still be 0 mod 2.

(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.**

Solution

Finding a suitable value of q. We need q to be large enough to prevent overflow on decryption. We will show that setting $q > 4B$ is sufficient for correct decryption.

Correctness. To see correctness, observe that given an encryption $\mathbf{c}$ of a bit $b \in \{0,1\}$, we recover b as follows:

$$\begin{aligned}\tilde{c} &= \mathbf{t}^\top \mathbf{c} \bmod q \\ &= \mathbf{t}^\top \mathbf{A}\mathbf{s} + \mathbf{t}^\top 2\mathbf{x} + \mathbf{t}^\top \mathbf{m} \bmod q \\ &= \mathbf{0} + \mathbf{t}^\top 2\mathbf{x} + \mathbf{t}^\top \mathbf{m} \bmod q.\end{aligned}$$

Suppose that $2\mathbf{x} \in (-\frac{q}{2}, \frac{q}{2}]$. Then since the entries of $\mathbf{t}$ and $\mathbf{m}$ are all in $\{0,1\}$, we have that $\mathbf{t}^\top \mathbf{c} = \mathbf{t}^\top 2\mathbf{x} + \mathbf{t}^\top \mathbf{m} \in (-\frac{q}{2}, \frac{q}{2}]$, so we can correctly decrypt as follows:

$$\mathbf{t}^\top 2\mathbf{x} + \mathbf{t}^\top \mathbf{m} = 0 + \mathbf{t}^\top \mathbf{m} = b \bmod 2.$$

Setting $q > 4B \implies 2\mathbf{x} \in (-\frac{q}{2}, \frac{q}{2}]$, so it is sufficient for correct decryption.

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

Solution

IND-CPA security. Suppose, towards contradiction, that the scheme is not IND-CPA secure. Then there exists a PPT $\mathcal{A}$ breaking the IND-CPA security game. Construct $\mathcal{B}$ which has non-negligible advantage in the $\mathsf{distLWE}_{n,m,q,B}$ problem. Specifically, $\mathcal{B}$ gets as input a matrix $\mathbf{A}$ and a vector $\mathbf{c}$ and must distinguish between the case where $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q$ for $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^{n\times 1}$ and $\mathbf{x} \xleftarrow{R} \{-B, \ldots, B\}$ and the case where $\mathbf{c} \xleftarrow{R} \mathbb{Z}_q^{1\times m}$.

$\mathcal{B}(\mathbf{A}, \mathbf{c})$

1 : $b \leftarrow \{0,1\}$
2 : $\mathbf{m} \leftarrow [0\ 0\ \cdots\ 0\ b]^\top \in \mathbb{Z}_q^{m\times 1}$
3 : $\tilde{\mathbf{c}} \leftarrow 2\mathbf{c} + \mathbf{m} \bmod q$
4 : $b' \leftarrow \mathcal{A}(\mathbf{A}, \tilde{\mathbf{c}})$
5 : **if** $b = b'$ **then return** 1 (LWE)
6 : **else return** 0 (random)

Analysis: First, we observe that we do not need to let $\mathcal{A}$ pick the messages (as in the general IND-CPA game) given that the message space is $\{0,1\}$ (thus we already know the choice of messages). Second, note that if $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{x}$, then

$$\tilde{\mathbf{c}} = 2\mathbf{c} = \mathbf{A}(2\mathbf{s}) + 2\mathbf{x} + \mathbf{m} \bmod q.$$

Since $\gcd(q, 2) = 1$ and $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^{n\times 1}$, $2\mathbf{s}$ is uniformly random in $\mathbb{Z}_q$. Thus $\tilde{\mathbf{c}}$ is distributed identically to an encryption of $b \in \{0,1\}$ in the above scheme with $pk = \mathbf{A}$. Hence, if $\mathcal{A}$ has non-negligible advantage $\delta(n)$ in distinguishing between an encryption of 0 and an encryption of 1 then $b = b'$ with probability $\delta(n)$ when $\mathcal{A}$ is given an LWE instance. On the other hand, if $\tilde{\mathbf{c}}$ is uniformly random, then $\mathcal{A}$ has no advantage and so $b = b'$ with probability $\frac{1}{2}$. We conclude, therefore, that $\mathcal{B}$'s advantage is at least $\delta(n)/2$ in $\mathsf{distLWE}_{n,m,q,B}$. By contrapositive, if $\mathsf{distLWE}_{n,m,q,B}$ is assumed to be intractable, then the encryption scheme is IND-CPA secure.

(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 $b_0 + b_1 \pmod 2$.**

Solution
We define $\mathsf{ADD}(c_0, c_1) = c_0 + c_1 \bmod q$. To see why this works:

$$\begin{aligned}
\mathbf{c'} &\leftarrow \mathbf{c_0} + \mathbf{c_1} \bmod q \\
&\Rightarrow \mathbf{c'} = \mathbf{A s_0} + 2\mathbf{x_0} + \mathbf{m_0} + \mathbf{A s_1} + 2\mathbf{x_1} + \mathbf{m_1} \pmod q \\
&= \mathbf{A}(\mathbf{s_0} + \mathbf{s_1}) + 2(\mathbf{x_0} + \mathbf{x_1}) + (\mathbf{m_0} + \mathbf{m_1}) \bmod q \\
&= \mathbf{A}(\mathbf{s'}) + 2(\mathbf{x'}) + (\mathbf{m'}) \bmod q
\end{aligned}$$

where $\mathbf{m'} = \mathbf{m_0} + \mathbf{m_1} \bmod q$. Following the decryption procedure results in $b_0 + b_1 \bmod 2$, as required.

(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 $\mathbf{t}$.

Solution

To perform a homomorphic multiplication, we take the outer-product of the ciphertexts. That is, we set $\mathsf{MUL}(\mathbf{c_0}, \mathbf{c_1}) = \mathbf{c_0}\mathbf{c_1}^\top \bmod q$. The resulting ciphertexts will be of the form $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$. The LWE "secrets" will be matrices of the form $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S'} \in \mathbb{Z}_q^{m \times n}$. The error and message terms will be matrices $\mathbf{X} \in \mathbb{Z}_q^{m \times m}$ and $\mathbf{M} \in \mathbb{Z}_q^{m \times m}$, respectively.

$$\begin{aligned}
\mathbf{C'} &\leftarrow \mathbf{c_0}\mathbf{c_1}^\top \bmod q \\
&= (\mathbf{A}\mathbf{s_0} + 2\mathbf{x_0} + \mathbf{m_0})\mathbf{c_1}^\top \\
&= (\mathbf{A}\mathbf{s_0})\mathbf{c_1}^\top + (2\mathbf{x_0} + \mathbf{m_0})\left(\mathbf{s_1}^\top\mathbf{A}^\top + 2\mathbf{x_1}^\top + \mathbf{m_1}^\top\right) \bmod q \\
&= \mathbf{A}\underbrace{(\mathbf{s_0}\mathbf{c_1}^\top)}_{\mathbf{S}} + 2\underbrace{\left(2\mathbf{x_0}\mathbf{x_1}^\top + \mathbf{m_0}\mathbf{x_1}^\top + \mathbf{x_0}\mathbf{m_1}^\top\right)}_{\mathbf{X}} + \underbrace{\mathbf{m_0}\mathbf{m_1}^\top}_{\mathbf{M}} + \underbrace{(2\mathbf{x_0} + \mathbf{m_0})\mathbf{s_1}^\top}_{\mathbf{S'}}\mathbf{A}^\top \bmod q
\end{aligned}$$

By the above, we see that the product ciphertext has the form:

$$\mathbf{AS} + 2\mathbf{X} + \mathbf{M} + \mathbf{S'}\mathbf{A}^\top \bmod q.$$

and $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$ is a matrix ciphertext. We can decrypt this ciphertext using $\mathbf{t}$ as follows:

$$b \leftarrow (\mathbf{t}^\top\mathbf{C}\mathbf{t} \bmod q) \bmod 2.$$

To see correctness, observe that given a product encryption $\mathbf{C}$ of a bit $b \in \{0, 1\}$,

$$\begin{aligned}
\widetilde{\mathbf{C}} &= \mathbf{t}^\top\mathbf{C}\mathbf{t} \bmod q \\
&= \left(\mathbf{t}^\top\mathbf{A}\right)\mathbf{S}\mathbf{t} + \mathbf{t}^\top 2\mathbf{X}\mathbf{t} + \mathbf{t}^\top\mathbf{M}\mathbf{t} + \mathbf{t}^\top\mathbf{S'}\left(\mathbf{A}^\top\mathbf{t}\right) \bmod q \\
&= \mathbf{0} + 2\left(\mathbf{t}^\top\mathbf{X}\mathbf{t}\right) + \mathbf{t}^\top\mathbf{M}\mathbf{t} + \mathbf{0} \bmod q \\
&= 2\left(\mathbf{t}^\top\mathbf{X}\mathbf{t}\right) + \mathbf{t}^\top\mathbf{M}\mathbf{t} \bmod q.
\end{aligned}$$

If we interpret the entries of $\widetilde{\mathbf{C}}$ in $(-\frac{q}{2}, \frac{q}{2}]$ then we have that:

$$2\left(\mathbf{t}^\top\mathbf{X}\mathbf{t}\right) + \mathbf{t}^\top\mathbf{M}\mathbf{t} \bmod 2 = \mathbf{t}^\top\mathbf{M}\mathbf{t} \in \mathbb{Z} = b \bmod 2.$$

(e) **Show that you can homomorphically evaluate any polynomial of degree 2 on a sequence of encrypted bits $b_0, b_1, \ldots, b_N$ using the above scheme. Specifically, show how to set q in terms of N, 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.**

Solution

To evaluate a degree-2 polynomial, we first compute a product ciphertext for each monomial.

- If the monomial has degree 2, e.g. x_1x_2, we multiply the relevant ciphertexts as described in (d), e.g. $\mathbf{c_1 c_2}^\top$, where $\mathbf{c_1}$ is the ciphertext encrypting the bit b_1 and $\mathbf{c_2}$ is the ciphertext encrypting the bit b_2.
- If the monomial has degree 1, e.g. x_2, then we multiply the relevant ciphertext by an encryption of 1 as described in (d), e.g. $\mathbf{c_2 c^*}^\top$, where $\mathbf{c^*}$ is an encryption we generate ourselves of the bit 1.

Now all we need to do is add together all the product ciphertexts for each monomial to create an encryption of the desired output. However, we must first show that a product ciphertext is still additively homomorphic, since in (c) we only showed that regular ciphertexts are additively homomorphic.

Additive homomorphism of product ciphertexts. To see this, recall that a product ciphertext is of the form:

$$\mathbf{AS} + 2\mathbf{X} + \mathbf{M} + \mathbf{S'A}^\top \bmod q.$$

Let $\mathbf{C_0}$ and $\mathbf{C_1}$ be two $m \times m$ ciphertext matrices.

$$\begin{aligned}
&\mathbf{C_0}+\mathbf{C_1} \\
&\quad = (\mathbf{AS_0} + 2\mathbf{X_0} + \mathbf{M_0} + \mathbf{S'_0A}^\top \bmod q) + (\mathbf{AS_1} + 2\mathbf{X_1} + \mathbf{M_1} + \mathbf{S'_1A}^\top \bmod q) \\
&\quad = \mathbf{A}(\mathbf{S_0} + \mathbf{S_1}) + 2(\mathbf{X_0} + \mathbf{X_1}) + (\mathbf{M_0} + \mathbf{M_1}) + (\mathbf{S'_0} + \mathbf{S'_1})\mathbf{A}^\top \bmod q \\
&\quad = \mathbf{AS} + 2\mathbf{X} + \mathbf{M} + \mathbf{S'A}^\top \bmod q
\end{aligned}$$

where $\mathbf{M} = \mathbf{M_0} + \mathbf{M_1}$. Thus, we have that product ciphertexts are additively homomorphic (though they cannot be multiplied again).

Setting parameters. We now look at how to set q such that we can correctly evaluate degree-2 polynomial with N variables. Recall that each monomial's product ciphertext has an error term $\mathbf{X} = 2\mathbf{x_0 x_1}^\top + \mathbf{m_0 x_1}^\top + \mathbf{x_0 m_1}^\top$, where the entries of $\mathbf{x_0}, \mathbf{x_1}$ are in $\{-B, \ldots, B\}$ and the entries of $\mathbf{m_0}, \mathbf{m_1}$ are in $\{0, 1\}$, so the entries of $\mathbf{X}$ are in $\{-2B^2 - B, \ldots, 2B^2 + B\}$. We can upper bound the number of monomials by N^2, since we have N variables, so the final evaluated ciphertext will be the addition of at most N^2 product ciphertexts. Then the entries of the error term in the evaluated ciphertext are in $\{-N^2(2B^2 + B), N^2(2B^2 + B)\}$. For us to be able to interpret this as a value in $(-\frac{q}{2}, \frac{q}{2}]$, we need $q > 2N^2(2B^2 + B)$.
Since the final evaluated ciphertext is in $\mathbb{Z}_q^{m \times m}$, it has can be represented by $m^2 \log q$ bits, which is polylogarithmic in N.

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.

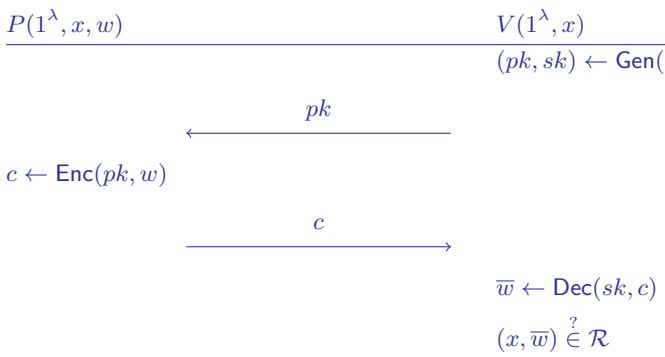
She explains the definition to Bob, but Bob thinks that including V's randomness in the view is unnecessarily complicated. He claims that it is equivalent to say that a proof system $\langle P, V\rangle$ is computationally zero-knowledge if there exists an efficient simulator S such that for all $x \in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

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

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.

Solution

The definitions are not equivalent. Counterexample: Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Consider the following proof system for a language $\mathcal{L}$ where $\exists w \in \{0,1\}^\ell$ s.t. $(x, w) \in \mathcal{R} \iff x \in \mathcal{L}$ and ℓ is the maximum length of a witness:



Completeness. By correctness of the PKE scheme.

Soundness. V accepts $\implies \exists\ \overline{w}$ such that $(x, \overline{w}) \in \mathcal{R} \implies x \in \mathcal{L}$.

Bob's ZK. By IND-CPA security of the PKE scheme. Consider the following simulator.

$S(1^\lambda, x)$

1 : $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$
2 : $c \leftarrow \mathsf{Enc}(pk, 0^\ell)$
3 : **return** (pk, c)

Alice's ZK. V learns a witness for x, so the protocol is clearly not 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.

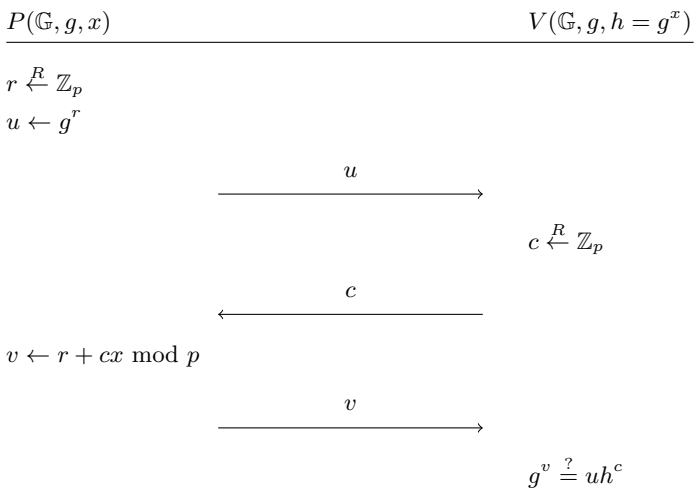


Figure 2: Honest-verifier Zero-knowledge proof of Discrete Logarithm Knowledge.

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 4 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.

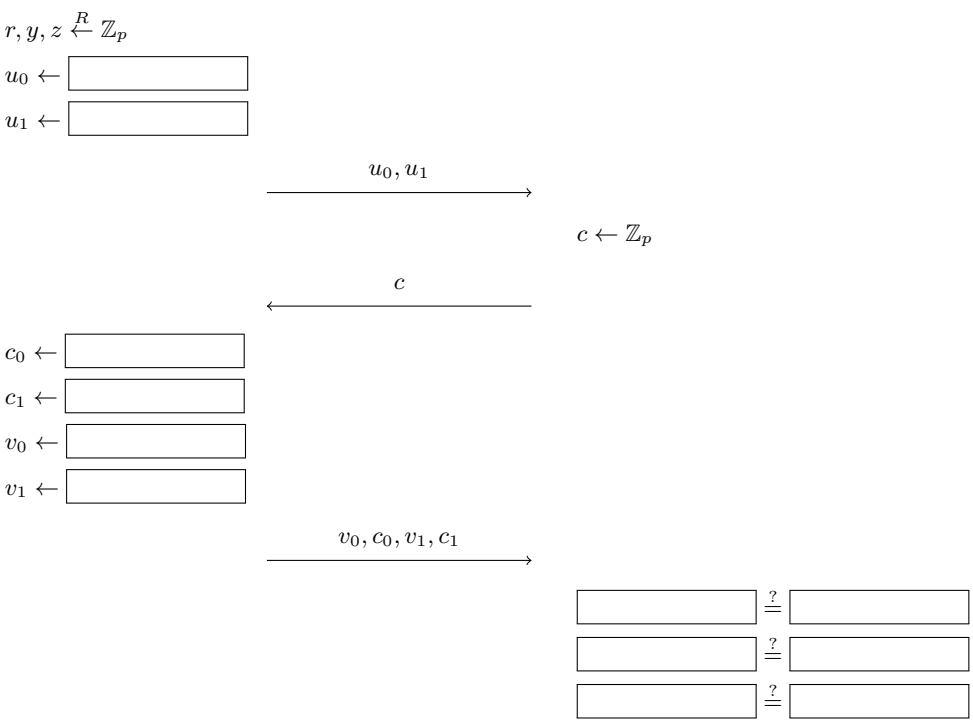


Figure 3: Zero-knowledge proof of 1-out-of-2 Discrete Logarithm Knowledge.

Solution

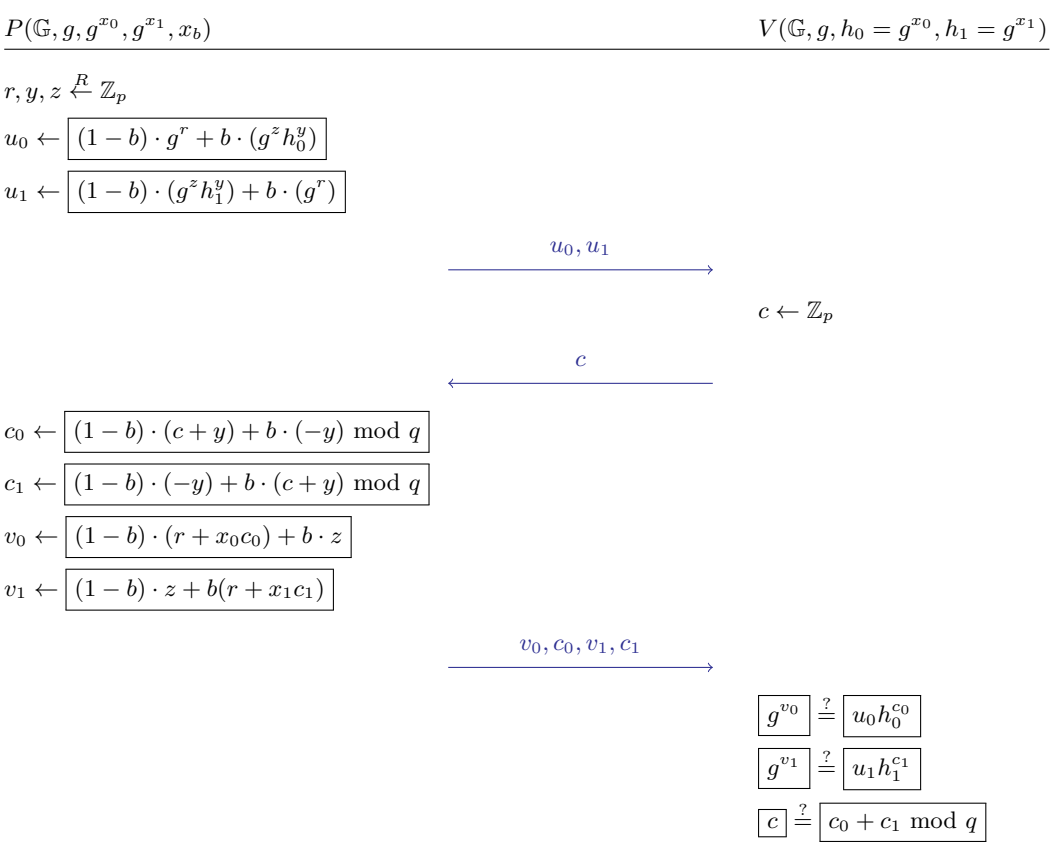


Figure 4: Zero-knowledge proof of 1-out-of-2 Discrete Logarithm Knowledge.

Solution (continued...)

Completeness. The verifier always accepts a valid proof. We show this case by case.
Case $b = 0$:

$$u_0 = g^r, u_1 = g^z h_1^y$$
$$c_0 = c + y, c_1 = -y$$
$$v_0 = r + x_0 c_0, v_1 = z$$

Then we have that:

$$g^{v_0} = g^{r+x_0 c_0} = g^r h_0^{c_0} = u_0 h_0^{c_0}$$
$$g^{v_1} = g^z = g^z g^{x_1 y} g^{x_1(-y)} = g^z h_1^y h_1^{(-y)} = u_1 h_1^{c_1},$$

and $c_0 + c_1 = (c + y) - y = c$ as required.
Case $b = 1$:

$$u_0 = g^z h_1^y, u_1 = g^r$$
$$c_0 = -y, c_1 = c + y$$
$$v_0 = z, v_1 = (r + x_1 c_1)$$

Then we have that:

$$g^{v_0} = g^z = g^z g^{x_0 y} g^{x_0(-y)} = g^z h_0^y h_0^{(-y)} = u_0 h_0^{c_0}$$
$$g^{v_1} = g^{r+x_1 c_1} = g^r h_1^{c_1} = u_1 h_1^{c_1},$$

and $c_0 + c_1 = (c + y) - y = c$ as required.

Solution (continued...)

Proof of knowledge. We build an *extractor* $\mathcal{E}$ which runs the prover P to recover the discrete logarithm of either $h_0 = g^{x_0}$ or $h_1 = g^{x_1}$ as follows. On input h,

1. Run P to get commitments u_0 and u_1.
2. Return challenge $c \xleftarrow{R} \mathbb{Z}_p$ and receive (v_0, c_0, v_1, c_1) from P.
3. Rewind P, send $c' \xleftarrow{R} \mathbb{Z}_p$, and receive (v_0', c_0', v_1', c_1') from P.
4. Return $x_0 = (v_0 - v_0')(c_0 - c_0')^{-1} \bmod p$ and $x_1 = (v_1 - v_1')(c_1 - c_1')^{-1} \bmod p$.

Analysis: P always returns (v_0, c_0, v_1, c_1) and (v_0', c_0', v_1', c_1') which satisfy the relation:

- $g^{v_0} = u_0 h_0^{c_0}$ and $g^{v_0'} = u_0' h_0^{c_0'}$
- $g^{v_1} = u_1 h_1^{c_1}$ and $g^{v_1'} = u_1' h_1^{c_1'}$
- $c_0 + c_1 = c$ and $c_0' + c_1' = c'$.

We analyze two cases.

Case 1: Suppose that the prover sends $c_0 = c + y$ and $c_1 = -y$ (i.e., the prover is choosing to prove knowledge of x_0). Then, $(v_0 - v_0') = c_0 x_0 - c_0' x_0 = x_0(c_0 - c_0')$. Hence, $x_0 = (v_0 - v_0')(c_0 - c_0')^{-1}$, as required, except in the case where $c_0 = c_0'$, which is negligible.

Case 2: Suppose that the prover sends $c_0 = -y$ and $c_1 = c + y$ (i.e., the prover is choosing to prove knowledge of x_1). Then, $(v_1 - v_1') = c_1 x_1 - c_1' x_1 = x_1(c_1 - c_1')$. Hence, $x_1 = (v_1 - v_1')(c_1 - c_1')^{-1}$, as required, except in the case where $c_1 = c_1'$, which is negligible.

Because we rewind the prover to the step in the protocol *after* it sends the commitments u_0 and u_1, the prover cannot alternate between proving knowledge of x_0 and x_1 across the two executions. Moreover, by the check that $c_0 + c_1 = c$ and $c_0' + c_1' = c$ it holds that either Case 1 or Case 2 is true. Therefore, we get that the extractor succeeds in recovering the discrete logarithm with probability $1 - \mathsf{negl}(\lambda)$ of either h_0 or h_1, giving us 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 $(\mathbb{G}, g, h_0, h_1)$:

1. Sample random $r_0, r_1, c_0, c \xleftarrow{R} \mathbb{Z}_p$.
2. $c_1 \leftarrow c - c_0$
3. Set $u_0 \leftarrow g^{r_0} h_0^{-c_0}$ and $u_1 \leftarrow g^{r_1} h_1^{-c_1}$.
4. Set $v_0 \leftarrow r_0$ and $v_1 \leftarrow r_1$.
5. Output $\{u_0, u_1, c, c_0, c_1, v_0, v_1\}$.

Analysis: First, observe that:

- $g^{v_0} = g^{r_0} = g^{r_0 + x_0 c_0 - x_0 c_0} = g^{r_0 - x_0 c_0} h^{c_0} = u_0 h^{c_0}$.
- $g^{v_1} = g^{r_1} = g^{r_1 + x_1 c_1 - x_1 c_1} = g^{r_1 - x_1 c_1} h^{c_1} = u_1 h^{c_1}$.
- $c_0 + c_1 = c$

Because the verifier V is honest, c follows the same (uniform) distribution in $\mathbb{Z}_p$ as in the actual protocol (and hence c_0 and c_1 are also uniformly distributed). Likewise for the distribution of u_0, u_1 and v_0, v_1, which are both uniformly random.

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 $c_0 \xleftarrow{R} \mathbb{Z}_p$ and receive v_0 from P.
3. Rewind P, send $c_1 \xleftarrow{R} \mathbb{Z}_p$, 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 $g^{v_0} = g^{r+c_0 x} = uh^{c_0}$ and $g^{v_1} = g^{r+c_1 x} = uh^{c_1}$. As such, $(v_0 - v_1) = c_0 x - c_1 x = x(c_0 - c_1)$. 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 $r, c \xleftarrow{R} \mathbb{Z}_p$.
2. Set $u \leftarrow g^r h^{-c}$.
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.