Exam – Cryptology 1 – 01410

15.05.2024

With solutions

Instructions and advice

- For all calculations in Part II: explain how you have computed your results. Unexplained results will receive few or no points. If you use a computer (or similar), then you need to be able to explain how the computer arrived at the answer.
- The problems have been created such that it is possible to solve them without the help of a computer or similar.
- Read all the questions first, and begin to work on the ones you find easy.

Part I – See online system (28 points)

You get 2 points for every correct selection and -2 points for every incorrect selection. As for any multiple choice quiz, here only the selection counts. There is no need to describe your reasoning. The questions and answer options are here printed for your convenience, but **you have to answer them using the web-based system!**

1. Which of the following statements about Pseudorandom cryptographic primitives are true? Select all that apply.
    A. A Pseudorandom Permutation (PRP) is also more commonly known as a hash function. false
    B. A PRP of block size B can be constructed from a PRF of input and output length $B/2$ using the Feistel construction. true
    C. A PRP of block length B is indistinguishable from a PRF of output length B up until the Birthday Bound true
    D. Electronic Codebook is a recommended mode of operation to encrypt long messages using a PRP of fixed block length. false, ECB mode is not IND secure for messages longer than one block, and not IND-CPA secure for any message length.
    E. Modes of operation are used to encrypt messages of arbitrary length directly, i.e. without any additional processing of the message before applying the mode of operation. false, padding it needed to make the message length a multiple of the block length, and possibly also for security
    F. Counter mode encryption of a message of k blocks (including any padding) produces ciphertexts of length $k + 1$ blocks true
    G. Counter mode encryption is IND-CPA secure. true
2. Which of the following statements about Hash functions and MACs are true? Select all that apply.

A. CBC-MAC, which applies the same key throughout all invocations of the underlying PRF, is secure against length-extension attacks.false

B. Message authentication Codes (MACs) are used to make encryption schemes IND-CPA secure.false, they are used to make encryption IND-CCA secure.

C. A PRF with output length at least the security parameter can directly be used as a Message Authentication Code, by evaluating the PRF on the MAC input and attaching the PRF output as the MAC.true

D. In practice, as in theory, cryptographic hash functions are always keyed false, in practice hash functions are fixed, keyless functions

E. For a cryptographic hash function of output length 128 bits, we expect to see the first collision with constant probability after querying 2^{64} inputs true

3. Which of the following statements are true about RSA? Select all that apply.

A. RSA uses arithmetic modulo a prime false, RSA uses arithmetic modulo a product of two primes

B. RSA is secure if factoring is hard false, this implication is not known

C. RSA is broken if factoring is easy true, factoring allows to compute the private key from the public key.

D. For the security of RSA, it is important that the public exponent e is picked in a randomized way false. In practice, a fixed value of $e = 2^{16} + 1$ is used.

E. For the security of RSA, it is important that the private exponent d is picked in a randomized way true, this is the private key and should not be guessable

F. RSA encryption is IND-CPA secure, irrespective of any used padding scheme false

G. (plain) RSA signatures are constructed by using the decryption algorithm of RSA encryption as the signing algorithm true

4. Which of the following statements are true about Diffie-Hellman? Select all that apply.

A. The Diffie-Hellman key exchange protocol has 3 protocol messages false, it has 2

B. The Diffie-Hellman key exchange protocol can be completed in one simultaneous round of communication between the participants. true

C. In the Diffie-Hellman key exchange protocol, it is important that Alice sends her first protocol message after receiving Bobs first protocol message. false

D. In the Diffie-Hellman key exchange protocol, Alice and Bob output the same key with probability one (in the absence of active attacks). true

E. The Diffie-Hellman key exchange protocol is secure against active attacks, assuming DDH. false, it is not secure against, for example, Man in the Middle attacks

F. The Diffie-Hellman key exchange protocol is insecure against passive attacks, even assuming DDH. false

G. The Diffie-Hellman key exchange protocol outputs a secure key if at least one of the participants picks their exponent uniformly at random and independent from any other data.true

5. Which of the following statements are true about Post-Quantum Cryptography and LWE? Select all that apply.

A. Quantum computers are known to be in principle able to break all deployed cryptography in time polynomial in the key length. false, not symmetric crypto

B. Quantum computers are a threat to cryptography, as they can be used to completely break the security of block ciphers.false

C. Quantum computers are a threat to cryptography, as they can be used to break the security of RSA.true, using Shor's algorithm

D. Elliptic-curve Diffie-Hellman is believed to be secure against quantum computing attacks. false

E. The Learning With Errors problem is believed to be hard to solve even for quantum computers. true

F. In Regev's encryption scheme, the message is used as the error in the Learning With Errors problem. false, it is added as an additional "big error", not as the small LWE error

Part II (42 points)

1. (10 points) Consider the function $F : \{0,1\}^{n\times\lambda} \times \{0,1\}^n \to \{0,1\}^\lambda$, which does the following:

   1. On input $\mathbf{k}, \mathbf{x}$ it writes the vector $\mathbf{k} \in \{0,1\}^{n\times\lambda}$ as n consecutive vectors $\mathbf{k}_0, \ldots, \mathbf{k}_{n-1}$ where each $\mathbf{k}_i$ is of length λ bits. This means $\mathbf{k}_0$ consists of the first λ bits of $\mathbf{k}$, $\mathbf{k}_1$ consists of the next λ bits etc.
   2. Let $x_0, \ldots, x_{n-1}$ be the bits of $\mathbf{x}$, i.e. x_0 is the first bit of $\mathbf{x}$, x_1 is the 2nd bit etc.
   3. $F(\mathbf{k}, \mathbf{x}) := \bigoplus_{i\in\{0,\ldots,n-1\}, x_i=1} \mathbf{k}_i$, i.e. F is equivalent to the XOR of all substrings of $\mathbf{k}$ where the corresponding input bit is 1.

   If F was a PRF, then we would like that $L^F_{PRF-Real} \approx L^F_{PRF-Rand}$ where

Library $L^F_{PRF-Real}$

Sample $k \leftarrow K$ uniformly at random

Lookup$(x \in \{0,1\}^n)$

1. $y \leftarrow F(k, x)$
2. Output y

and

Library $L^F_{PRF-Rand}$

Let T be an empty array.

Lookup$(x \in \{0,1\}^n)$

1. If $T[x]$ is undefined then sample $T[x] \leftarrow \{0,1\}^\lambda$
2. Output $T[x]$

(a) Describe an algorithm A that can reconstruct the whole key $\mathbf{k}$ by making n queries to the library $L^F_{PRF-Real}$. Then use your algorithm A to build a polynomial-time (in n and λ) distinguisher that can distinguish $L^F_{PRF-Real}$ from $L^F_{PRF-Rand}$ and explain how successful it will be.
A generate the input $x = 1||0^{n-1}$ and calls the library to get the first row of the key. Then A repeats that $n-1$ times, while changing the index of the 1 bit for each call to recover the full key. Construct the distinguisher D as follows:

1. Run A to recover the full key
2. Choose vector $x \in \{0,1\}^n \setminus \{e_0, ..., e_{n-1}\}$, query the library on x to get back y
3. Compute the value $y' = \bigoplus_{\{0,..,n-1\},x_i=1} K'_i$ and check $y == y'$
4. If $y \neq y'$ output Random
5. Esle, output Real

If D is connected to $L^F_{PRF-Real}$ it will always output Real, if D is connected to $L^F_{PRF-Rand}$ it will output Real with probability $\leq \frac{1}{2^\lambda}$

(b) For this PRF candidate F, it turns out that one does not need to recompute the key $\mathbf{k}$ to distinguish $L^F_{PRF-Real}$ from $L^F_{PRF-Rand}$. Show that there exists an input x which, when queried to the library, will almost always succeed in distinguishing (i.e. it fails with probability $2^{-\lambda}$). Call the function with $x = 0^n$, no matter which key was used $L^F_{PRF-Real}$ will always output 0^n. The distinguisher only looses the game if it is connected to $L^F_{PRF-Rand}$ and the library outputs 0^n, which happens with probability $\frac{1}{2^\lambda}$

(c) Both aforementioned attacks would not work if we modify our construction into a function $G : \{0,1\}^{2n\times\lambda} \times \{0,1\}^n \to \{0,1\}^\lambda$ as follows:

1. On input $\mathbf{k}, \mathbf{x}$ it writes the vector $\mathbf{k} \in \{0,1\}^{2n\times\lambda}$ as $2n$ consecutive vectors $\mathbf{k}_0, \ldots, \mathbf{k}_{2n-1}$ of length λ bits each.
2. Again, let $x_0, \ldots, x_{n-1}$ be the bits of $\mathbf{x}$, i.e. x_0 is the first bit of $\mathbf{x}$, x_1 is the 2nd bit etc.
3. $G(\mathbf{k}, \mathbf{x}) := \bigoplus_{i\in\{0,\ldots,n-1\},x_i=1} \mathbf{k}_i \oplus \bigoplus_{i\in\{0,\ldots,n-1\},x_i=0} \mathbf{k}_{n+i}$.

Is this function G now a secure PRF according to the definition? If yes, then argue why. If not, show how you can modify the attacks so that they succeed now.

The function is still insecure. For example, when querying the library on $0^n, 1^n$ and vector x, we can always compute the vector $\tilde{x}$ with all bits flipped. Indeed, by querying the library on $0^n, 1^n$ we get the values $K_n \oplus ... \oplus K_{2n-1}$ and $K_0 \oplus ... \oplus K_{n-1}$ respectively. By querying a value x we get $(\bigoplus_{i\in\{0,\ldots,n-1\},x_i=1} K_i) \oplus (\bigoplus_{i\in\{0,\ldots,n-1\},x_i=0} K_i)$. Thanks to the definition of G and the properties of $\oplus$ we have that $G(K, \tilde{x}) = \bigoplus_{i=0}^{2n-1} K_i \oplus G(K, x)$.

2. (6 points) Let $F : K \times \{0,1\}^B \to \{0,1\}^\lambda$ be a Pseudorandom Function of input length B and consider the following MAC scheme for messages whose length is a multiple of B, i.e. $M = (\{0,1\}^B)^*$

$\mathsf{KG}()$**:**

1. Sample a key $k \leftarrow K$.
2. Output k.

$MAC(k, m)$**:**

1. Write $m = (m_0, \ldots, m_\ell)$ where each m_i is of length B bits.
2. Compute $t := \bigoplus_{i=0}^{\ell} F(k, m_i)$
3. Output t.

If these algorithms form a secure MAC scheme, then we would like that $L_{MAC-Real} \approx L_{MAC-fake}$ where

Library $L_{MAC-real}$

Run $k \leftarrow \mathsf{KG}()$

$Gettag(m \in M)$

1. Return $MAC(k, m)$

$Checktag(m \in M, t \in \{0,1\}^\lambda)$

1. Return $MAC(k, m) == t$

and

Library $L_{MAC-fake}$

Run $k \leftarrow \mathsf{KG}()$ and let S be an empty set

$Gettag(m \in M)$

1. $t = MAC(k, m)$
2. $S \leftarrow S \cup \{(m, t)\}$
3. Return t

$Checktag(m \in M, t \in \{0,1\}^\lambda)$

1. Return $(m, t) \in S$

(a) Show that these algorithms do not form a secure MAC scheme. To show this construct a distinguisher, explain why it works and what its success probability is. (3 points) The main observation should be that the MAC algorithm XORs the ouputs of F together, while the order of input blocks does not matter to the MAC algorithm. There are two simple attacks: 1) Call Checktag on a message $m' = (m||m)$ for some m of length B bits and tag $t = 0^\lambda$, or 2) obtain a MAC t on message $m_0||m_1$ of length $2B$ and then check the MAC t against the message $m_1||m_0$. In both cases, $L_{MAC-fake}$ will return 0 as the MAC has not been generated by the library for this message, while $L_{MAC-real}$ will always return 1 as it is a valid MAC according to the algorithm. Hence, an attacker can use this to distinguish and the distinguishing probability is 1.

(b) Argue why, if messages can only have length B (meaning that $M = \{0,1\}^B$ instead), this is a secure MAC algorithm. Explain how many messages would have to be MACed with this algorithm before one could observe the first collision in the MACs with constant probability. (3 points) Since the message is now of fixed length B, the XOR becomes irrelevant and the MAC algorithm is identical to just evaluating a PRF on an unknown key. This, as mentioned in the lecture and written in the textbook, means that the algorithm is a secure MAC algorithm. The outputs are indistinguishable from random (as F is a PRF). This implies that the attacker needs to send $\approx 2^{\lambda/2}$ *Gettag* requests to the library before finding a collision on the outputs, unless F has less than $\lambda/2$ bits of security.

3. (10 points) Recall the RSA algorithm from the lecture:

$\mathsf{KG}()$**:**

1. Sample two different odd prime numbers p, q and set the RSA nodulus $N = p \cdot q$.
2. Set $\varphi(N) = (p-1) \cdot (q-1)$ and choose e such that $gcd(e, \varphi(N)) = 1$. Then compute $d = e^{-1} \bmod \varphi(N)$.
3. Output $pk = (e, N)$ as public key and $sk = (d, N)$ as private key.

$\mathsf{Enc}(pk, m)$**:** To encrypt the message $m \in Z_N^*$ using public key $pk = (e, N)$:

1. Compute $c = m^e \bmod N$

2. Output c.

$\mathsf{Dec}(sk, c)$: To decrypt a ciphertext $c \in Z_N^*$ using private key $sk = (d, N)$:

1. Compute $m = c^d \bmod N$
2. Output m.

(a) Explain why the public exponent cannot be 2 for any RSA modulus. Because e has to be co-prime to $\varphi(N)$. As $\varphi(N) = (p-1) \cdot (q-1)$ must be an even number and the GCD of an even number and 2 will always be 2.

(b) Explain how the equation

$$1^e = 1 \bmod N$$

implies that plain RSA encryption is not IND-CPA secure. Explain how padded RSA avoids this problem. In the left-right game it is easy to see which message was encrypted by setting $m_L = 1$ and m_R. As $1^e = 1 \bmod N$ message m_L will generate output 1 while m_R generates an output $\neq 1$ (since e cannot be the order of any message with the way it is selected). The attack does not work for padded RSA as the padding is done with random numbers, which will make the output look random as well.

(c) Find the smallest possible value for the public exponent e for RSA modulus $N = 73 \cdot 83$. Explain your reasoning! $\varphi(N) = \varphi(73) \cdot \varphi(83) = (72) \cdot (82)$
$72 = 2 \cdot 26 = 2^2 \cdot 18 = 2^3 \cdot 9 = 2^3 \cdot 3^2$ and $82 = 41 \cdot 2$.
So we have to find the smallest number which is co-prime to $2, 3$ and 41, which is 5.

(d) Let $N = 73 \cdot 83$, $e = 17$, $d = 3473$. Compute the encryption of the message $m = 2$. We can break down the computation to make it easier to compute, $2^{17} \bmod N$ is equal to $(2^{16} \bmod N) \cdot 2 \bmod 6059$. Continue until we reach $(((2^2 \bmod N)^2 \bmod N)^2 \bmod N)^2 \bmod N) \cdot 2 \bmod N$. Using the real numbers as input:

$(2)^2 \bmod 6059 = 4$
$(2^2)^2 \bmod 6059 = 2^4 \bmod 6059 = 16$
$(2^2)^2 \bmod 6059 = 2^8 \bmod 6059 = 256$
$(2^2)^2)^2 \bmod 6059 = 2^{16} \bmod 6059 = 4949$

So $2^{17} \bmod 6059 = 4949 \cdot 2 \bmod 6059 = 2833$.

(e) Let $N = 73 \cdot 83$, $e = 17$, $d = 3473$. Describe how a message m with hash $h = H(m)$ can be signed performing 16 multiplications mod N. We can break the computation down as well:

m^{3473}
$= m^{3472} \cdot m$
$= (m^{1736})^2 \cdot m$
$= ((m^{868})^2)^2 \cdot m$
$= (((m^{434})^2)^2)^2 \cdot m$
$= ((((m^{217})^2)^2)^2)^2 \cdot m$ (5 multiplications)
m^{217}
$= m^{216} \cdot m$
$= (m^{108})^2 \cdot m$
$= ((m^{54})^2)^2 \cdot m$
$= (((m^{27})^2)^2)^2 \cdot m$ (4 multiplications)
m^{27}
$= m^{26} \cdot m$
$= (m^{13})^2 \cdot m$
$= (m^{12} \cdot m)^2 \cdot m$

$= ((m^6)^2 \cdot m)^2 \cdot m$
$= (((m^3)^2)^2 \cdot m)^2 \cdot m$
$= (((m^2 \cdot m)^2)^2 \cdot m)^2 \cdot m$ (7 multiplications)
Which gives an overall of $7 + 5 + 4 = 16$ multiplications.

4. (8 points) Recall the El Gamal encryption scheme where p is a prime, $q = (p-1)/2$ is also a prime and $g \in Z_p^*$ is a generator of the subgroup of Z_p^* of order q.

   $\mathsf{KG}()$**:**

   1. Let $B = g^b \bmod p$ where $b \in Z_q$ is uniformly random.
   2. Output $pk = (B)$ as the public key and $sk = (b)$ as the secret key.

   $\mathsf{Enc}(pk, m)$**:** To encrypt a message $m \in \langle g \rangle$ with public key $pk = (B)$:

   1. Choose $a \in Z_q$, and compute $c_1 = g^a \bmod p$.
   2. Set $c_2 = pk^a \cdot m \bmod p$.
   3. Output ciphertext $c = (c_1, c_2)$.

   $\mathsf{Dec}(sk, c)$**:** To decrypt a ciphertext $c = (c_1, c_2)$ with private key $sk = (b)$:

   1. Set $m' = c_2 / c_1^{sk} \bmod p$ (here, both multiplication and inversion are modulo p).
   2. Output m'.

   We also recap the libraries that are used to define IND-CPA security for a public key encryption scheme KG, Enc, Dec with plaintext space M, namely $L_{pk-cpa-0}$ and $L_{pk-cpa-1}$:

Library $L_{pk-cpa-0}$

Run $(pk, sk) \leftarrow \mathsf{KG}()$

$Getpk()$

1. Return pk

$Challenge(m_0 \in M, m_1 \in M)$

1. Return $\mathsf{Enc}(pk, m_0)$

and

Library $L_{pk-cpa-1}$

Run $(pk, sk) \leftarrow \mathsf{KG}()$

$Getpk()$

1. Return pk

$Challenge(m_0 \in M, m_1 \in M)$

1. Return $\mathsf{Enc}(pk, m_1)$

(a) For RSA, the naive signature scheme is constructed by using the plain RSA decryption algorithm as signing algorithm and the plain RSA encryption algorithm as verification algorithm. Explain why this blueprint will not yield a correct digital signature scheme when applied to El Gamal encryption. The El Gamal encryption algorithm is randomized (a is chosen at random). We can thus not expect Dec(Enc(x))=x, the equation required for correctness of the described signature scheme.

An El Gamal ciphertext consists of 2 group elements, which is twice the size of the message. In the following, we will improve on that.

(b) Consider a variant of El Gamal where a message consisting of two group elements, $m = (m_0, m_1)$, is encrypted by picking a random exponent $a \in Z_q$ and computing $(g^a, pk^a \cdot m_0, pk^a \cdot m_1)$ (modulo p). Show that this variant is not IND-CPA secure. Choose messages $m_i = (m_{i,0}, m_{i,1})$ for $i = 0, 1$ where $m_{0,0} = m_{0,1}$ and $m_{1,0} \neq m_{1,1}$. The calling $Challenge(m_0, m_1)$ will yield output (g^a, c_0, c_0) in $L_{pk-cpa-0}$, but (g^a, c_0, c_1) with $c_0 \neq c_1$ in $L_{pk-cpa-1}$. This allows the attacker to distinguish with probability 1.

In the following, we consider a variant of El Gamal where key generation outputs $sk = (b_0, b_1)$ and $pk = (g^{b_0}, g^{b_1})$. Encryption with a public key $pk = (B_0, B_1)$ samples an exponent $a \in Z_q$ and computes $(g^a, B_0^a \cdot m_0, B_1^a \cdot m_1)$ (all modulo p).

(c) Describe the decryption algorithm for this modified El Gamal scheme, and show that the overall scheme is correct: encryption and subsequent decryption recovers the input message. To decrypt the message m_i c_0 is raised to the secret key component b_i. When multiplying the inverse of that with c_i the original message is recovered.

$$m_0 = (c_0^{b_0})^{-1} \cdot c_1 = (g^{ab_0})^{-1} \cdot B_0^a \cdot m_0 = (g^{ab_0})^{-1} \cdot g^{ab_0} \cdot m_0$$
$$m_1 = (c_0^{b_1})^{-1} \cdot c_2 = (g^{ab_1})^{-1} \cdot B_1^a \cdot m_1 = (g^{ab_1})^{-1} \cdot g^{ab_1} \cdot m_1$$

(d) Prove that this variant of El Gamal is IND-CPA secure if the Decisional Diffie-Hellman problem is hard. To that end, assume there exists an Adversary $\mathcal{A}$ that distinguishes the libraries $L_{pk-cpa-0}$ and $L_{pk-cpa-1}$ for the aforementioned public-key encryption scheme that uses $sk = (b_0, b_1)$ and $pk = (g^{b_0}, g^{b_1})$. Use this adversary to construct an algorithm that solves the decisional Diffie Hellman problem, and describe its runtime and advantage.

Assume an IND-CPA adversary $\mathcal{A}$ and construct a DDH distinguisher $\mathcal{D}$ as follows. First the DDH challenger will do the following

1. $a, b, c \xleftarrow{\$} \mathbb{Z}_{q-1}$
2. $\beta \xleftarrow{\$} \{0, 1\}$
3. $\begin{cases} a \cdot b, & \beta = 0 \\ c, & \beta = 1 \end{cases}$
4. $(g, g^a, g^b, g^{c_\beta})$

And gives $(g, r, s, t) = (g, g^a, g^b, g^{c_\beta})$ to $\mathcal{D}$ who does the following:

1. Pick $\eta \leftarrow \{0, 1\}$ % for choosing where to place the DDH challenge
2. Sample $b' \xleftarrow{\$} \mathbb{Z}_{q-1}$
3. Set $B_\eta = s, B_{1-\eta} = g^{b'}$
4. $\gamma \xleftarrow{\$} \{0, 1\}$ % For choosing $L_{pk-cpa-\gamma}$
5. $\delta \xleftarrow{\$} \{0, 1\}$ % For choosing real or random for the simultated part of the public key
6. $d \xleftarrow{\$} \mathbb{Z}_{q-1}$

7. Set $u = \begin{cases} g^d & \text{if } \delta = 0 \\ r^{b'} & \text{if } \delta = 1 \end{cases}$
8. Run the IND-CPA adversary to obtain γ', simulating library $L_{pk-cpa-\gamma}$ as follows:
   - *Getpk*():
     (a) Return $pk = (B_0, B_1)$
   - *Challenge*($m_0 \in M, m_1 \in M$):
     (a) Parse $m_\gamma = (m_{\gamma,0}, m_{\gamma,1})$
     (b) Compute $c_\eta = t \cdot m_\gamma$ and $c_{1-\eta} = u \cdot m_{1-\gamma}$.
     (c) Return (r, c_0, c_1).
9. Set $\begin{cases} \beta' = 0, & \gamma' = \gamma \\ \beta' = 1 \end{cases}$
10. Output β' to the DDH challenger.

First, observe that the cases $\beta = 1, \delta = 0$ and $\beta = 0, \delta = 1$ are indistinguishable: In either case, one of the message parts (m_η in the former case and $m_{1-\eta}$ in the latter) is multiplied with the right public key part raised to the power a, and the other message part is multiplied with a random value. Also, the two libraries are exactly indistinguishable for $\beta = 1, \delta = 1$ as both message parts are multiplied with a uniformly random group element. We therefore have

$$\begin{aligned}
Adv(\mathcal{D}) =& \frac{1}{2}\left(Adv(D|\delta = 0) + Adv(\mathcal{D}|\delta = 1)\right) \\
=& \frac{1}{2}\left(|\Pr[1 \leftarrow D|\delta = 0, \beta = 0] - \Pr[1 \leftarrow D|\delta = 0, \beta = 1]| + |\Pr[1 \leftarrow D|\delta = 1, \beta = 0] - \Pr[1 \leftarrow D|\delta = 1, \beta = 1]|\right) \\
=& \frac{1}{2}\left(\left|\frac{1}{2} + Adv(\mathcal{A}) - \Pr[1 \leftarrow D|\delta = 0, \beta = 1]\right| + \left|\Pr[1 \leftarrow D|\delta = 1, \beta = 0] - \frac{1}{2}\right|\right) \\
=& \frac{1}{2}\left(\left|\frac{1}{2} + Adv(\mathcal{A}) - \Pr[1 \leftarrow D|\delta = 0, \beta = 1]\right| + \left|\Pr[1 \leftarrow D|\delta = 0, \beta = 1] - \frac{1}{2}\right|\right) \\
\leq& \frac{1}{2} Adv(\mathcal{A})
\end{aligned}$$

Here, $Adv(\mathcal{D}|\delta = 0)$ is the distinguishing advantage of $\mathcal{D}$ conditioned on $\delta = 0$. The runtime of $\mathcal{D}$ is similar to the runtime of $\mathcal{A}$.

5. (8 points) Recall Regev's encryption scheme:

   KG():
   1. Let $\mathbf{s} \in \mathbb{Z}_p^n$ be chosen uniformly at random.
   2. Pick $\mathbf{a}_i \in \mathbb{Z}_p^n$ independently uniformly at random and $e_i \leftarrow D_{\mathbf{Z},\sigma}$ independent as well, for $i = 1, ..., m$.
   3. Set $b_i = \mathbf{a}_i \cdot \mathbf{s} + e_i$.[1].
   4. Output the public key $pk = (\mathbf{a}_i, b_i)_{i=1}^m$ and secret key $sk = (\mathbf{s})$.

   Enc(pk, m): To encrypt a message $m \in \{0, 1\}$ with public key $pk = (\mathbf{a}_i, b_i)_{i=1}^m$:
   1. Sample a vector of random bits $\mathbf{r} \in \{0, 1\}^m$.
   2. Set
   $$\mathbf{c}_0 = \sum_{i=1}^m r_i \mathbf{a}_i \bmod p, \qquad c_1 = \frac{x(p-1)}{2} + \sum_{i=1}^m r_i b_i \bmod p.$$

[1] $\mathbf{x} \cdot \mathbf{y}$ mod p for vectors $\mathbf{x}$ and $\mathbf{y}$ denotes the inner product

3. Output the ciphertext $c = (\mathbf{c}_0, c_1)$.

$\mathsf{Dec}(sk, c)$**:** To decrypt a ciphertext $c = (\mathbf{c}_0, c_1)$ using secret key $sk = \mathbf{s}$:

1. Compute $h = c_1 - \mathbf{s} \cdot \mathbf{c}_0 \bmod p$.
2. Interpret h as a real number and compute $h' = \left\lceil \frac{2h}{p} \right\rfloor$
3. Interpret h' as an integer and output $x' = h' \bmod 2$.

(a) Describe a chosen ciphertext attack against Regev's scheme, exploiting the rounding operation in the decryption algorithm. More precisely, given a ciphertext $c = (\mathbf{c}_0, c_1)$, describe how to modify it to obtain a different ciphertext $c' = (\mathbf{c}'_0, c'_1)$ that decrypts to the same message with high probability. Let $c = (c_0, c_1)$ be a ciphertext. We set the modified ciphertext to $c' = (c_0, c_1 + 1)$. To avoid decryption failures, when decrypting the honest ciphertext c, h is likely not too far from a multiple of $p/2$, i.e., h has at least distance 2 to $p/4$ and $3p/4$ with constant probability. In that case, it follows that $\left\lceil \frac{2(h+1)}{p} \right\rfloor = \left\lceil \frac{2h}{p} \right\rfloor$, showing that decrypting c' yields the same result as decrypting c.

(b) Sometimes it can be an advantage if it is possible to "re-randomize" ciphertexts. Assume the parameters for Regev's encryption scheme allow taking the encryption randomness $\mathbf{r}$ from $\{-2, -1, 0, 1, 2\}^m$ instead of just $\{0, 1\}^m$. Describe an algorithm that, on input a ciphertext $c = (\mathbf{c}_0, c_1)$ for standard Regev (generated with $\mathbf{r} \in \{0, 1\}^m$) and the public key, generates a ciphertext $c' = (\mathbf{c}'_0, c'_1)$ with the following property: There exists $\mathbf{r} \in \{-2, -1, 0, 1, 2\}^m$ such that

$$\begin{aligned} \mathsf{Enc}_{pk}(x; \mathbf{r}) &= c' \text{ for} \\ x &= \mathsf{Dec}_{sk}(c) \end{aligned}$$

where $c \neq c'$.

Here, $\mathsf{Enc}_{pk}(x; \mathbf{r})$ denotes encryption of a message x with public key pk and randomness $\mathbf{r}$.

Note: The main point here is that it is possible to modify c to obtain c' without knowing sk, and without knowing x. Suppose $c = (\mathbf{c}_0, c_1)$ is the result of encrypting message x with randomness r. Pick $\mathbf{r}' \in \{0, 1\}^m$ and set

$\mathbf{c}'_0 = c_0 + \Sigma_{i=1}^m r'_i a_i$
$c'_1 = c_1 + \Sigma_{i=1}^m r'_i b_i$

Clearly, encrypting x with randomness $r + r'$ results in ciphertext $c' = (\mathbf{c}'_0, c'_1)$ as well. As $r, r' \in \{0, 1\}^m$, we have $r + r' \in \{0, 1, 2\}^m \subset \{-2, -1, 0, 1, 2\}^m$.