CS4501 Cryptographic Protocols, Homework 3

Response by: Your Name, (Computing ID)

Total points: 40 awarded maximum. 54 available. Points are noted after each problem.

Instructions. For each problem, typeset your solution in the `answer` environment, and if there are sub-problems, mark them clearly. Feel free to use as much space as you require, and be sure to update your name and computing ID above, and the acknowledgments box at the end.

Policies. In short, you are encouraged to think about the problems on your own, and then discuss them and work toward solutions with your classmates. You must write and submit your own solutions. You may also read any published material that helps you come to an understanding of the problems, but you must acknowledge and/or reference any discussion or published material, with the exception of lecture notes and other official class materials, in-class and in-office-hours discussions, and basic LaTeX help or dictionary lookups. It is a violation of the honor code if any of the following occur:

- You copy text directly from any source.
- You use any material or discussion without acknowledgment or citation, excluding the above special cases.
- You are unable to explain your work orally.

See https://jackdoerner.net/teaching/2026/Spring/CS4501/#policies for more details.

Problem 1 (Proving OT Secure (25pts total))**.** In this question, you will prove that if the Decisional Diffie-Hellman Problem is hard relative to some group sampling algorithm $\mathcal{G}$, then the Oblivious Transfer protocol presented in class securely computes the function $f_{\mathsf{OT}} : [0,\ell]^2 \times \{0,1\} \to \{\lambda\} \times [0,\ell]$ such that $f_{\mathsf{OT}}((m_0, m_1), b) = m_b$, for some constant ℓ. Since we are realizing a deterministic functionality, it is sufficient to use the "even simpler" semi-honest computational SFE definition (Definition 9 from Lecture 14), which allows us to prove correctness and simulation separately. Since the protocol is *asymmetric*, we will have to construct two different simulators. This problem will walk you through each of the major components in the proof. It may help you to read the entire problem before starting.

Remark: The version of the OT protocol originally presented at the end of Lecture 14 had a minor error, and you will not be able to prove it secure. The Lecture 14 slides have been updated to feature a corrected protocol, which was additionally presented at the beginning of Lecture 15. Please use the corrected version.

Remark: Assume that randomness is always provided in whatever form is needed. So if Bob needs to sample a random element of $\mathbb{G}$, his random "coins" will magically be a uniformly distributed element of $\mathbb{G}$, even if $\mathbb{G}$ itself was determined during the course of the protocol. Similarly, when simulating an element that Bob randomly samples from $\mathbb{G}$, your simulator only needs to compute the correct distribution for that element: it does not need to figure out which sequence of random boolean coins would need to be put into a sampling algorithm in order to produce that element.

(a) Correctness (2pts): Prove that if both parties behave honestly, then Bob certainly outputs m_b at the end of the protocol, thereby satisfying the security definition's first requirement.

(b) Simulator for Corrupted Alice (5pts): Construct a simulator algorithm $\mathsf{Sim}_{\mathsf{A}}^{\mathcal{G}}$ that takes as input Alice's inputs 1^κ, m_0, and m_1, and uses them together with $\mathcal{G}$ to *perfectly* simulate the view of a corrupted Alice in the OT protocol. Argue that your simulator's output is identically distributed to Alice's view in the real protocol.

(c) Simulator for Corrupted Bob (5pts): Construct a simulator algorithm $\mathsf{Sim}_{\mathsf{B}}^{\mathcal{G}}$ that takes as input Bob's inputs 1^κ and b and his output m_b and uses them together with $\mathcal{G}$ to simulate the view of a Corrupted Bob in the OT protocol. It is important that your simulator does not use m_{1-b} in *any way*. You do not yet need to argue about the distribution your simulator produces: you will do that in the next two sub-problems.

(d) Hybrid Experiment (5pts): Next, define a *hybrid experiment* which blends aspects of the simulated view you defined in the previous sub-problem and aspects of a corrupted Bob's view in the real world. Your hybrid experiment should use both m_b and m_{1-b}. Use Lemma 1 from Lecture 14 to argue that your hybrid experiment is distributed *identically* to the simulated view you defined previously.

(e) Reduction (5pts): Now you will argue that your hybrid experiment is *computationally indistinguishable* from the real world protocol. Suppose you have a distinguisher $\mathcal{D}$ that can distinguish between your hybrid experiment and the real-world view of a corrupted Bob. Use $\mathcal{D}$ to construct another distinguisher $\mathcal{D}'$ that plays the Decisional Diffie Hellman Games (given Definition 3 of Lecture 14). Argue that the advantage of $\mathcal{D}'$ in the DDH game is exactly the same as the advantage of $\mathcal{D}$ in distinguishing your hybrid experiment from the real-world protocol.

(f) Conclusion (3pts): Combine the pieces you have constructed in sub-problems (b) through (e) to conclude that the second requirement of the security definition is satisfied.

Hint: The solution to this problem uses ideas similar to those use to prove Theorems 1 and 2 in Lecture 14. If you are stuck, try reviewing those proofs for inspiration.

Problem 2 (Homomorphic Encryption (5pts))**.** In this problem, we will prove that the ElGamal encryption function (given in Definition 14 of Lecture 14) is a homomorphism (see Definition 8 of Lecture 13), and the decryption function is its inverse. Let $(\mathbb{G}, g, q)$ be a cyclic, abelian group such that $|\mathbb{G}| = q$ is prime, $q > 2\ell$ for some constant ℓ, and g is a generator of $\mathbb{G}$. Let the group operation be written multiplicatively. Given $x \leftarrow \mathbb{Z}_q$, let $\mathsf{sk} := x$, $X := g^x$, and $\mathsf{pk} := (\mathbb{G}, g, q, X)$. For $i \in [2]$, let $m_i \in [0, \ell]$ be arbitrary, let $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i)$, and let $(R_i, C_i) = c_i$. Finally, let $c_\times := (R_1 \cdot R_2, C_1 \cdot C_2)$. Prove that $\mathsf{Dec}_{\mathsf{sk}}(c_\times) = m_1 + m_2 \in [0, 2\ell]$

Remark: Notice that this means you can perform some kinds of computations on data that is encrypted under ElGamal *without decrypting it.* Doesn't that seem useful?

Problem 3 (Coin Tossing with ElGamal (10pts total))**.** In problem 2b of Homework 1, you were asked to construct a simple coin tossing protocol that was secure against semi-honest adversaries. In this problem, we will have our first encounter with *malicious* adversaries, who might instruct the parties they corrupt to do things other than what the protocol says they should.

Before reading the rest of this problem, think about your solution to problem 2b of Homework 1.

Is there a single party that a malicious adversary can corrupt in order force the output of the coin toss to be a particular value? Think about how you might mitigate this. Is there a way that you could combine encryption and your solution to part 2a? You do not need to write anything down and there are no points for performing this thought experiment in your head (but you may award yourself points in your head, if you like).

Now that you've thought about it, consider the following protocol for flipping a fair coin, which requires two parties (Alice and Bob) and an ideal functionality $\mathcal{F}_{\mathsf{FlipHelper}}$. The protocol makes use of a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (see Definitions 10-12 of Lecture 14) for some message space $\mathcal{M}$ such that $\{0,1\} \subseteq \mathcal{M}$.

1. $\mathcal{F}_{\mathsf{FlipHelper}}$ sends a public key pk to both Alice and Bob.
2. Alice samples a uniform bit $b_{\mathsf{A}} \leftarrow \{0,1\}$ and computes $c_{\mathsf{A}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(b_{\mathsf{A}})$. Alice sends c_{A} to both Bob and $\mathcal{F}_{\mathsf{FlipHelper}}$.
3. After he receives c_{A}, Bob samples a uniform bit $b_{\mathsf{B}} \leftarrow \{0,1\}$, computes $c_{\mathsf{B}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(b_{\mathsf{B}})$, and sends c_{B} to both Alice and $\mathcal{F}_{\mathsf{FlipHelper}}$. Alice and $\mathcal{F}_{\mathsf{FlipHelper}}$ ignore his message if $c_{\mathsf{B}} = c_{\mathsf{A}}$.
4. $\mathcal{F}_{\mathsf{FlipHelper}}$ decrypts both c_{A} and c_{B}. If $b_{\mathsf{A}} \in \{0,1\}$, then $\mathcal{F}_{\mathsf{FlipHelper}}$ sets $b'_{\mathsf{A}} := b_{\mathsf{A}}$; otherwise it sets $b'_{\mathsf{A}} := 0$. If $b_{\mathsf{B}} \in \{0,1\}$, then $\mathcal{F}_{\mathsf{FlipHelper}}$ sets $b'_{\mathsf{B}} := b_{\mathsf{B}}$; otherwise it samples $b'_{\mathsf{B}} := 0$. Finally, $\mathcal{F}_{\mathsf{FlipHelper}}$ sends b'_{A} to Bob and b'_{B} to Alice.
5. Both parties output $b'_{\mathsf{A}} \oplus b'_{\mathsf{B}}$.

It is easy to see that if the adversary semi-honestly corrupts one participant, then this protocol is secure. We are going to investigate what happens if the adversary is *malicious*, meaning that the corrupted party might not follow the protocol instructions. For the following sub-problems, assume that all participants in the protocol are PPT and corrupted parties always send *well-formed* ciphertexts (that is, all ciphertexts sent by corrupted parties efficiently decrypt to messages in $\mathcal{M}$).

(a) (3pts) Suppose a malicious adversary corrupts Alice, but not Bob. Argue that regardless of what she does, Bob's output from the protocol is still uniformly distributed.

(b) (3pts) Suppose that a malicious adversary corrupts Bob, but not Alice. Suppose furthermore that the parties are using ElGamal Encryption (Definition 14 of Lecture 14). Because ElGamal Encryption is CPA secure, we know that Bob's chance of decrypting b_{A} before he sends his own message is negligible. Show that in spite of this, corrupted Bob can always force Alice to output 0.

(c) (3pts) Similarly, show that corrupted Bob can always force Alice to output 1.

(d) (1pt) Can you envision a security property for encryption that's *stronger* than CPA Indistinguishability, which would prevent Bob's attack and ensure that honest parties always output uniform bits in the above protocol? Your answer does not need to be very formal and you do not actually need to prove that the protocol would be secure under your definition. I am looking for intuition. Note that nothing about the protocol should change other than the encryption scheme. Talking through this one with your friends might help, if you're stuck.

Problem 4 (OWF or not? (3pts each))**.** Let $f : \{0,1\}^* \to \{0,1\}^*$ be a OWF. For each of these sub-problems you must determine whether the proposed constructions of g are *necessarily* OWFs or not. Justify your answer by providing either a short proof (if g *must be* a OWF), or disproof (if it *might not be*).

A proof is performed by contraposition: it consists of an algorithm that inverts f with non-negligible probability if there exists some adversary that inverts g with non-negligible probability.

A disproof consists of an algorithm that inverts g with non-negligible probability (unconditionally). A disproof might depend the particular details of f, in which case you should describe the f on which your disproof depends, and argue that it is a OWF.

(a) $g(x\|r) = f(x)\|r$ where $|x| = |r|$ and $\|$ denotes concatenation.

(b) $g(x) = f(x)\|x_1$ where x_1 is the first bit of x.

(c) $g(x) = f(x)||x_1||...||x_{\kappa/2}$ where $\kappa = |x|$ and x_i for $i \in [\kappa/2]$ is the i^{th} bit of x.

Problem 5 (A Conundrum (5pts))**.** Consider again a function $f : \{0,1\}^* \times \{\lambda\} \to \{0,1\}^* \times \{0,1\}^*$ such that $f(x,\lambda) = (g(x), h(x, g(x)))$ where $g(x)$ is some randomized process and h is deterministic.

In Problem 2c of Homework 1, you proved that there exists a *one-message* protocol that *perfectly* securely computes any such f (under Definition 2 of Lecture 3) in the presence of a semi-honest adversary that statically corrupts at most one party.

Prove that if the same one-message protocol you developed before also *computationally* securely computes any f (under Definition 7 of Lecture 14) in the presence of a semi-honest adversary that statically corrupts at most one party, then one-way functions *do not exist*.

Remark: If you didn't solve this problem in Homework 1, you may ask one of your classmates for their solution, but please be sure to cite them appropriately.

Hint: Computational security implies that there must exist a simulator. Use that simulator to win the one-way function game.

Acknowledgments

In this box, you should acknowledge your collaborators and the resources you used, if any. For example:

Problem 1: I discussed this problem with Alice and Bob. In addition, I asked Carol for help understanding Conditional Probability, but we did not discuss the problem further.

Problem 2: I asked ChatGPT "What is a turing machine?", and it gave me the following transcript: https://chatgpt.com/share/68b4dba7-e19c-8013-a137-e8db901493b7.

Problem 3: It helped me to read the proof of [Vad12, Theorem X, Page Y].

Instructor's Acknowledgments

Problems in this homework have been borrowed from or inspired by a number of sources. Citations can be provided on request, after the homework is completed.

References

[Vad12] Salil P. Vadhan. *Pseudorandomness*. 2012. https://people.seas.harvard.edu/~salil/pseudorandomness/pseudorandomness-published-Dec12.pdf.