CS4501 Cryptographic Protocols, Homework 2

Response by: Your Name, (Computing ID)

Total points: 40 awarded maximum. 52 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.

Notation. Throughout this document, when f is an n-ary function with domain $\mathcal{X}_1 \times \ldots \times \mathcal{X}_n$, we use the phrase "$f$-hybrid model" as shorthand for the $\mathcal{F}_{\mathsf{SFE}}(n, f, \mathcal{X}_1, \ldots, \mathcal{X}_n)$-hybrid model. When we have a randomized function $f(x_1, \ldots, x_n)$ that uses randomness from finite domain $\mathcal{R}$, we write $f(x_1, \ldots, x_n; r)$ to denote a *deterministic* version of f that takes the randomly sampled string $r \leftarrow \mathcal{R}$ as an additional input. We will use the functions $\mathsf{Share}_{p,n,t}$ and $\mathsf{Recon}_{p,n,t}$ to refer specifically to degree-t Shamir sharing among n parties, over $\mathbb{F}_p$. For $x \in \mathbb{F}_p$ we will use $\langle x \rangle$ to denote a Shamir sharing of x, and $\langle x \rangle_i$ to denote P_i's component of that sharing. Finally, we use $\mathcal{P}_{p,t,x}$ to denote the set of all degree-t polynomials over $\mathbb{F}_p$ that pass through the point $(0, x)$.

Problem 1 (Protocol Reductions (5pts each))**.** In each sub-problem, you will be given a pair of 2-ary functions f and g and ask you to design a protocol in the f-hybrid model that perfectly securely computes g in the presence of a static semi-honest adversary that corrupts at most one participant.

1. Consider the boolean AND function $f_{\mathsf{and}} : \{0,1\} \times \{0,1\} \to \{0,1\} \times \{0,1\}$ such that $f_{\mathsf{and}}(x_1, x_2) = (x_1 \wedge x_2, x_1 \wedge x_2)$, and the boolean OR function $g_{\mathsf{or}} : \{0,1\} \times \{0,1\} \to \{0,1\} \times \{0,1\}$ such that $g_{\mathsf{or}}(x_1, x_2) = (x_1 \vee x_2, x_1 \vee x_2)$. Design a protocol in the f_{and}-hybrid model and prove that it security computes g_{or}.
2. For some fixed $k \in \mathbb{N}$, consider a *randomized* $g : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k \times \{0,1\}^k$ where the randomness space is also $\{0,1\}^k$. That is, g takes inputs $x_1, x_2 \in \{0,1\}^k$ and internally samples some random value $r \leftarrow \{0,1\}^k$ which it uses to compute $(y_1, y_2) = g(x_1, x_2; r)$.

Define a *deterministic* 2-ary function f, and design a protocol that securely computes g in the f-hybrid model. Prove that your protocol is secure.

Hint: Think of the coin tossing functionality from your last homework as an example of g.

3. Consider a deterministic $g : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k \times \{0,1\}^k$ in which the parties obtain different outputs. That is, if we have $x_1, x_2 \in \{0,1\}^k$ and $(y_1, y_2) = g(x_1, x_2)$, it might be the case that $y_1 \neq y_2$. Note that P_1 is not allowed to learn any more about y_2 than can be inferred from x_1, y_1 (and likewise for P_2 with respect to y_1).

   Define a function $f : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{Y} \times \mathcal{Y}$ in which P_1 and P_2 receive the *same* output. That is, your function must be defined such that for $x_1, x_2 \in \mathcal{X}_1 \times \mathcal{X}_2$ there exists some $y \in \mathcal{Y}$ such that $(y, y) = f(x_1, x_2)$. Then, design a protocol that securely computes g in the f-hybrid model, and prove that your protocol is secure. You are free to choose the (finite) sets $\mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}$ in whatever way is convenient.

   Hint: One-Time Pad might be useful.

Problem 2 (Danish Multiplication (5pts each))**.** Recall that for some prime p, the BGW protocol is defined over $\mathbb{F}_p$, and the functionality $\mathcal{F}_{\mathsf{mul}}$ internally computes the randomized n-ary function $f_{\mathsf{mul}} : (\mathbb{F}_p \times \mathbb{F}_p)^n \to \mathbb{F}_p^n$ such that if $(x_1, \ldots, x_n) \in \text{image}(\mathsf{Share}_{p,n,t})$ and $(y_1, \ldots, y_n) \in \text{image}(\mathsf{Share}_{p,n,t})$ then $f_{\mathsf{mul}}((x_1, y_1), \ldots, (x_n, y_n)) = \mathsf{Share}_{p,n,t}(\mathsf{Recon}_{p,n,t}([n], (x_1, \ldots, x_n)) \cdot \mathsf{Recon}_{p,n,t}([n], (y_1, \ldots, y_n)))$.

In this problem, we will explore another way to securely compute f_{mul}, which makes use of a trusted dealer D that distributes *correlated randomness*. The trusted dealer performs the following steps before the protocol begins:

- The dealer samples $r \leftarrow \mathbb{F}_p$.
- The dealer samples $g \leftarrow \mathcal{P}_{p,t,r}$ and $G \leftarrow \mathcal{P}_{p,2t,r}$.
- For every $i \in [n]$, the dealer computes $r_i := g(i)$ and $R_i := G(i)$ and sends (r_i, R_i) to P_i.

Now we will construct our new multiplication protocol. At the beginning, each party P_i for $i \in [n]$ holds the correlated randomness (r_i, R_i) that it received from the dealer, and the input shares $\langle w_1 \rangle_i, \langle w_2 \rangle_i$ such that $\langle w_1 \rangle$ and $\langle w_2 \rangle$ are degree-t Shamir sharings. The **first** step of the protocol is for every P_i to locally compute $\langle \widehat{w_1 \cdot w_2} \rangle_i := \langle w_1 \rangle_i \cdot \langle w_2 \rangle_i$. Note that as in our other multiplication protocols, $\langle \widehat{w_1 \cdot w_2} \rangle$ is a *non-uniform* degree-$2t$ sharing of $w_1 \cdot w_2$.

1. The **second** step of the protocol is for every P_i to compute $\langle u \rangle_i := \langle \widehat{w_1 \cdot w_2} \rangle_i + R_i$ and send $\langle u \rangle_i$ to P_1. P_1 reconstructs $u := \mathsf{Recon}_{p,n,t}([n], \langle u \rangle)$, and then sends u to all of the other parties.

   Explain how to simulate the view of any single corrupted party P_i, given its inputs $(\langle w_1 \rangle_i, \langle w_2 \rangle_i)$ and correlated randomness (r_i, R_i).

2. Next, in the **third** step, each party P_i must use what it knows to locally compute a share $\langle w_3 \rangle_i$ such that the entire sharing $\langle w_3 \rangle$ is uniform[1] and has degree t, and such that $w_3 = w_1 \cdot w_2$.

   Explain how this can be done. As a reminder, P_i knows $\langle w_1 \rangle_i, \langle w_2 \rangle_i, r_i, R_i$, and u. P_i may not communicate.

[1] That is, it comprises n points on some polynomial that is uniform in the set $\mathcal{P}_{p,n,t}$.

3. What is the communication complexity of the above protocol (i.e. how many rounds does it take, and how many elements of $\mathbb{F}_p$ must be exchanged)? How does it compare to the original BGW multiplication protocol we saw in class? You should account separately for messages exchanged between parties, and messages transmitted from the dealer to the parties.

Problem 3 (From Arithmetic to Boolean (5pts each))**.** In class we argued that circuits over $\mathbb{F}_p$ can compute all functions with inputs and outputs in $\mathbb{F}_p$. In this problem we will generalize slightly further: we will see that circuits over $\mathbb{F}_p$ for $p > 2$ can also compute all functions with *boolean* inputs and outputs. This is important because we require $p > n > 2$ in the context of BGW, meaning we can never work over $\mathbb{F}_2$ directly.

Suppose you are given a protocol (such as BGW) that securely computes any function $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$. You wish to use it to securely compute a function $g : \{0,1\}^n \to \{0,1\}^n$, where each party has one bit as input. Let C be some circuit that computes g, expressed in terms of boolean AND and NOT gates. It is easy to see that if we encode `True` as 1 and `False` as 0 in $\mathbb{F}_p$, then

- If we are given input $x \in \{0,1\}$, a boolean NOT gate can be computed in $\mathbb{F}_p$ as $1 - x$.
- If we are given inputs $x_1, x_2 \in \{0,1\}$, a boolean AND gate can be computed in $\mathbb{F}_p$ as $x_1 \cdot x_2$.

Since AND and NOT are complete for boolean functions [Wik26], this method will allow us to compute any g. However, there is a problem with encoding a boolean circuit as an arithmetic circuit like this: the arithmetic circuit accepts as input all values in $\mathbb{F}_p$, but the encoding of g is not necessarily well-behaved if the parties input values other than 0 and 1.

Consider a case with three parties. Each P_i is supposed to use an input $x_i \in \{0,1\}$, and they wish to compute the boolean function $g(x_1, x_2, x_3) = (y_1, \lambda, \lambda)$ where $y_1 = ((\neg x_1) \wedge x_2) \vee (x_1 \wedge x_3)$. They will use a protocol that works over $\mathbb{F}_5$, and they can encode g in $\mathbb{F}_5$ as $y_1 = (1 - x_1) \cdot x_2 + (x_1 \cdot x_3)$.[2] Notice that if $x_1 = 0$ then $y_1 = x_2$, and if $x_1 = 1$ then $y_1 = x_3$. That is, if P_1 uses one of the intended inputs, then P_1 can learn either x_2 or x_3, but not both.

1. Demonstrate that a cheating P_1 that inputs $x_i \notin \{0,1\}$ can learn both x_2 and x_3.
2. To prevent this type of attack, you can first map all possible input values $x_i \in \mathbb{F}_5$ to a bounded input $x_i' \in \{0,1\}$ and then run the encoded boolean function g on x_1', x_2', x_3'. Create a function over $\mathbb{F}_5$ that performs this kind of mapping, and describe how your function can be generalized to any $\mathbb{F}_p$.

   Hint: Lagrange Interpolation might be useful. If you're more adventurous you can try looking up Fermat's Little Theorem.

Problem 4 (Generalizing an Impossibility (12 pts total))**.** In class, we proved that two parties cannot compute boolean AND with perfect security, if one party is corrupt. In this homework problem, you will prove that it is possible to generalize that result to larger values of n and $t \geq n/2$. In particular, you will prove that the impossibility holds for $n = 4$ and $t \geq 2$; the argument for an arbitrary n is essentially the same.

Let $f^2_{\mathsf{and}}(x_1, x_2) = (y, y)$ where $x_i \in \{0,1\}$ and $y = x_1 \wedge x_2$.

Let $f^4_{\mathsf{and}}(x_1, x_2, x_3, x_4) = (y, y, y, y)$ where $x_i \in \{0,1\}$ and $y = x_1 \wedge x_2 \wedge x_3 \wedge x_4$.

[2] Logical OR can be encoded as $+$ in this special case because we know that at most one of the two inputs to the OR gate will be `True`.

1. (5pts) Let π^4 be a 4-party protocol that computes f^4_{and} when all parties are honest (that is, assume that π^4 is correct, but not necessarily simulatable). Use π^4 to construct a 2-party protocol π^2 that computes f^2_{and} when both parties are honest. For now you only need to argue that your protocol is correct; you will argue for the security of your protocol in the next sub-problem.

   Hint: Since π^4 requires 4 parties, but there are only 2 parties participating in π^2, each of the parties in π^2 will have to play the role of *multiple* parties in π^4. In other words, each party in π^2 will have to emulate multiple π^4 parties in its head. Specify which parties from π^4 are emulated in the head of which parties in π^2, how the inputs of the emulated parties are set, which messages need to be sent in π^2, and how to determine the output.

2. (5pts) Let $4 > t \geq 2$. Prove that *if* π^4 perfectly securely computes f^4_{and} in the presence of a static, semi-honest adversary that corrupts t participants, *then* π^2 perfectly securely computes f^2_{and} in the presence of a semi-honest adversary that corrupts 1 participant.

3. (2pts) Combine what you have proven above with our theorem from class to reach a conclusion the existence of protocols that perfectly securely compute f^4_{and}.

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.

[Wik26] Wikipedia contributors. Functional completeness — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Functional_completeness&oldid=1334929950, 2026. [Online; accessed 16-February-2026].