Mid Semester - Fall, 2026
M. Tech. Cryptology & Security - 2^{nd} Year
Advanced Cryptology
(with Answers)

September 7, 2026

Maximum Marks: 25
Maximum Time: 2 hr
Open note/book exam

Instructions

- Maximum marks is 25. Total marks provided in the paper: 35.
- Be short and precise. For example, a 5 (respectively 3) mark question's answer should not be ideally more than a page (respectively half a page) with standard hand-writing. Partial marking will be provided for a partially correct answer/attempt.
- All parts of the same questions must be done at the same place, and in order. Marks will be deducted for violating this – this will be followed strictly.
- Verbosity (e.g. unnecessary/irrelevant sentences) is highly discouraged, and may lead to deduction of marks.
- You can use any book/notes during the exam, but *not internet*.
- The seating arrangements and other regulations circulated from Dean's office must be followed adequately.

Notations. The set of all integers is denoted by $\mathbb{Z}$, and the ring of integers modulo n by $\mathbb{Z}_n$. Below $\kappa \in \mathbb{N}$ denotes the security parameter. A uniform random sample from a domain D is denoted as $s \leftarrow_{\$} D$. Throughout, $\langle g \rangle = \mathbb{G}$ is a cyclic group of prime order p in which the group operation is written multiplicatively, and all exponent arithmetic is modulo p. We write $X \overset{c}{\approx} Y$ for computational indistinguishability, and $\mathsf{negl}(\kappa)$ for a negligible function.

Questions

1. Recall that the **Decisional Diffie–Hellman** (DDH) assumption over a cyclic group (of prime order) $\mathbb{G}$ states that
$$(g,\ g^x,\ g^y,\ g^{xy}) \quad \overset{c}{\approx} \quad (g,\ g^x,\ g^y,\ g^z), \qquad \text{where } x, y, z \leftarrow_{\$} \mathbb{Z}_p.$$
Consider the following *extended* version of it, which we call 2-DDH: for $x, y_1, y_2, z_1, z_2 \leftarrow_{\$} \mathbb{Z}_p$,
$$(g,\ g^x,\ g^{y_1},\ g^{y_2},\ g^{xy_1},\ g^{xy_2}) \quad \overset{c}{\approx} \quad (g,\ g^x,\ g^{y_1},\ g^{y_2},\ g^{z_1},\ g^{z_2}).$$
Note that the *same* x is paired with both y_1 and y_2.

(a) Show that if DDH holds over $\mathbb{G}$ then so does 2-DDH.

Answer: We use a hybrid argument with two steps, replacing one Diffie–Hellman element at a time. Define

$$\begin{array}{rcl} \mathsf{HYB}_0 & := & (g,\ g^x,\ g^{y_1},\ g^{y_2},\quad g^{xy_1},\ g^{xy_2}) \\ \mathsf{HYB}_1 & := & (g,\ g^x,\ g^{y_1},\ g^{y_2},\quad g^{z_1},\ g^{xy_2}) \\ \mathsf{HYB}_2 & := & (g,\ g^x,\ g^{y_1},\ g^{y_2},\quad g^{z_1},\ g^{z_2}) \end{array}$$

with all exponents uniform and independent. HYB_0 is the left distribution and HYB_2 the right one, so it suffices to show $\mathsf{HYB}_0 \overset{c}{\approx} \mathsf{HYB}_1$ and $\mathsf{HYB}_1 \overset{c}{\approx} \mathsf{HYB}_2$.

Step 1 ($\mathsf{HYB}_0 \overset{c}{\approx} \mathsf{HYB}_1$). Given a distinguisher A for HYB_0 vs. HYB_1, build a DDH distinguisher B_1. On DDH challenge (g, A, B, T) — where $A = g^x$, $B = g^{y_1}$ and T is either g^{xy_1} or uniform — B_1 does the following:

- pick $y_2 \leftarrow_{\$} \mathbb{Z}_p$ *itself*;
- hand A the tuple $\big(g,\ A,\ B,\ g^{y_2},\ T,\ A^{y_2}\big)$ and output whatever A outputs.

The crucial point is that B_1 can compute the last component: $A^{y_2} = g^{xy_2}$, because it *chose* y_2. If $T = g^{xy_1}$ the tuple is exactly HYB_0, and if T is uniform it is exactly HYB_1. Hence B_1's DDH advantage equals A's.

Step 2 ($\mathsf{HYB}_1 \overset{c}{\approx} \mathsf{HYB}_2$). Symmetrically: on DDH challenge (g, A, B, T), the reduction B_2 picks $y_1, z_1 \leftarrow_{\$} \mathbb{Z}_p$ itself, embeds B as g^{y_2}, and outputs $\big(g,\ A,\ g^{y_1},\ B,\ g^{z_1},\ T\big)$. This is HYB_1 when T is real and HYB_2 when T is uniform.

Combining the two steps, $\mathsf{HYB}_0 \overset{c}{\approx} \mathsf{HYB}_2$, which is exactly 2-DDH.

Note: *A* one-shot *reduction also works, via the* random self-reducibility *of* DDH*: from a single challenge $(g, A = g^x, B = g^y, C)$, draw $s_i, t_i \leftarrow_{\$} \mathbb{Z}_p$ and set*

$$B_i \ := \ B^{s_i} g^{t_i}, \qquad C_i \ := \ C^{s_i} A^{t_i} \qquad (i = 1, 2).$$

If $C = g^{xy}$ then $C_i = g^{x(ys_i + t_i)}$, the correct DH element for $B_i = g^{ys_i + t_i}$. If $C = g^z$ with $z \neq xy$, then $(ys_i + t_i,\ zs_i + xt_i)$ is the image of (s_i, t_i) under a matrix of determinant $xy - z \neq 0$, hence uniform on $\mathbb{Z}_p^2$ and independent across i. The same trick manufactures arbitrarily many slots out of one challenge.

The additive terms are not optional. *A tempting shortcut is to re-randomize* multiplicatively *only — $B_i := B^{\rho_i}$, $C_i := C^{\rho_i}$. The real side comes out right, but the random side does not: there $C_i = g^{z\rho_i}$ and $B_i = g^{y\rho_i}$, so* every *slot satisfies $\log C_i / \log B_i = z/y$, whereas in HYB_2 that ratio is fresh in each slot. The two distributions are therefore far apart, and the reduction proves nothing. The failure is starkest in the application of Question 2(a): there this shortcut answers every query as $\mathsf{H}(x_i)^{z/y}$, i.e. as the* genuine *PRF under the key z/y — so the "random" case is again a PRF, and the reduction has advantage exactly 0. It is the g^{t_i} and A^{t_i} terms that break the correlation.*

(b) Consider the following *multi-message* ElGamal scheme, which encrypts two (small) messages M_1, M_2 under a single ciphertext, reusing one randomness r under two keys:

$$\begin{array}{ll} \underline{\mathsf{KGEN}(1^\kappa)}: & x_1, x_2 \leftarrow_{\$} \mathbb{Z}_p; \quad \mathsf{pk} := (h_1, h_2) = (g^{x_1}, g^{x_2}), \quad \mathsf{sk} := (x_1, x_2) \\ \underline{\mathsf{ENC}(\mathsf{pk}, (M_1, M_2))}: & r \leftarrow_{\$} \mathbb{Z}_p; \quad c := \big(c_0,\ c_1,\ c_2\big) = \big(g^r,\ h_1^r \cdot g^{M_1},\ h_2^r \cdot g^{M_2}\big) \\ \underline{\mathsf{DEC}(\mathsf{sk}, c)}: & \text{recover } g^{M_i} = c_i / c_0^{x_i} \text{ and then } M_i \text{ by brute-force discrete log } (i = 1, 2). \end{array}$$

Prove that this scheme is IND-CPA secure. (Here the challenge is a pair of *message vectors* (M_1^0, M_2^0) and (M_1^1, M_2^1).)

Answer: Note first *why part (a) is needed*: the two ciphertext components share the *same* randomness r, so they are not two independent ElGamal ciphertexts and one cannot simply invoke single-message ElGamal CPA-security twice. It is exactly this reuse of r that the extended assumption of part (a) is tailored to.

We proceed by two games.

GAME_0: the real IND-CPA game. By definition $\Pr[\mathsf{GAME}_0 = 1] = \frac{1}{2} + \mathsf{Adv}^{\mathsf{cpa}}(\mathsf{A})$.
GAME_1: identical, except that the challenge ciphertext is computed as

$$c^* := \big(g^r,\ u_1 \cdot g^{M_1^b},\ u_2 \cdot g^{M_2^b}\big), \qquad u_1, u_2 \leftarrow_{\$} \mathbb{G},$$

i.e. the two masks h_1^r, h_2^r are replaced by independent uniform group elements.

Claim 1: $|\Pr[\mathsf{GAME}_0 = 1] - \Pr[\mathsf{GAME}_1 = 1]| \leqslant \mathsf{Adv}^{\text{2-DDH}}$**.** The reduction B receives an 2-DDH challenge $(g, A, B_1, B_2, C_1, C_2)$ and:

- sets $\mathsf{pk} := (h_1, h_2) = (B_1, B_2)$ and gives it to A;
- on receiving (M_1^0, M_2^0) and (M_1^1, M_2^1), picks $b \leftarrow_{\$} \{0,1\}$ and returns the challenge ciphertext

$$c^* := \big(A,\ \ C_1 \cdot g^{M_1^b},\ \ C_2 \cdot g^{M_2^b}\big);$$

- outputs 1 iff A's guess b' equals b.

Observe that B does *not* know $x_1 = \log_g B_1$ or $x_2 = \log_g B_2$ — and it does not need to, since the CPA game gives A no decryption oracle. Writing $A = g^r$: if $C_i = g^{r x_i} = h_i^r$ (the real case), then c^* is distributed exactly as $\mathsf{ENC}(\mathsf{pk}, (M_1^b, M_2^b))$, so A's view is that of GAME_0; if C_1, C_2 are uniform and independent, the view is that of GAME_1.

Claim 2: $\Pr[\mathsf{GAME}_1 = 1] = \frac{1}{2}$**.** In GAME_1, u_i is uniform on $\mathbb{G}$ and independent of everything else, so $u_i \cdot g^{M_i^b}$ is uniform on $\mathbb{G}$ regardless of M_i^b — a one-time pad in the group. Since u_1, u_2 are independent, c^* is a triple of group elements whose distribution does not depend on b at all; and $c_0 = g^r$ is independent of b too. Hence b is information-theoretically hidden and A can do no better than guessing.

Putting the two claims together with part (a),

$$\mathsf{Adv}^{\mathsf{cpa}}(\mathsf{A}) \ \leqslant\ \mathsf{Adv}^{\text{2-DDH}} \ \leqslant\ 2\,\mathsf{Adv}^{\mathsf{DDH}}(\kappa) \ =\ \mathsf{negl}(\kappa).$$

Note*: The identical proof works when h_1, h_2 are the public keys of two* different *receivers — this is multi-recipient ElGamal with reused randomness — as long as neither of the two secret keys is corrupted. Also, the messages must be small, since decryption brute-forces the discrete log — for the bits of part (c) this is trivial.*

(c) Design a Σ-protocol that proves, about a ciphertext $c = (c_0, c_1, c_2)$ of the above scheme, that *both* encrypted messages are bits, i.e. $M_1, M_2 \in \{0,1\}$ — without revealing M_1, M_2 or r. A description of the high-level design is enough; you need not prove its properties.

Answer: **Four instances of the generic protocol.** Let $\mathcal{R}_{\text{LIN}}$ be the relation $\vec{u} = \mathbf{G}^{\vec{\alpha}}$, with instance $(\vec{u}, \mathbf{G})$ and witness $\vec{\alpha}$. For each bit pair $\mathbf{b} = (b_1, b_2) \in \{0,1\}^2$, set

$$\mathbf{G} = [\,g,\ h_1,\ h_2\,] \in \mathbb{G}^{1\times 3}, \qquad \vec{\alpha} = (r), \qquad \vec{u}_{\mathbf{b}} = [\,c_0,\ c_1/g^{b_1},\ c_2/g^{b_2}\,].$$

Thus each branch has $n = 1$ witness variable and $m = 3$ equations, in the notation of the generic protocol:

$$\vec{u}_{\mathbf{b}} = \mathbf{G}^{\vec{\alpha}} \quad\Longleftrightarrow\quad c_0 = g^r, \qquad c_1/g^{b_1} = h_1^r, \qquad c_2/g^{b_2} = h_2^r.$$

The four target vectors are

$\mathbf{b}$	$\vec{u}_{\mathbf{b}}$
$(0,0)$	$[\,c_0,\ c_1,\ c_2\,]$
$(0,1)$	$[\,c_0,\ c_1,\ c_2/g\,]$
$(1,0)$	$[\,c_0,\ c_1/g,\ c_2\,]$
$(1,1)$	$[\,c_0,\ c_1/g,\ c_2/g\,]$

Let $\Sigma_{\mathbf{b}}$ be the generic Σ-protocol instantiated on $(\vec{u}_{\mathbf{b}}, \mathbf{G})$.

Apply the binary OR construction recursively:

$$(\Sigma_{00} \text{ OR } \Sigma_{01}) \text{ OR } (\Sigma_{10} \text{ OR } \Sigma_{11}).$$

High-level design. Use the challenge space $\mathcal{C} = \{0,1\}^\ell$ with $2^\ell \leqslant p$; interpret challenges as integers when used in exponent arithmetic. The prover knows the real branch $\mathbf{b}^* = (M_1, M_2)$ and its witness $\vec{\alpha} = (r)$.

i. For the real branch, choose $\vec{\alpha}_{t,\mathbf{b}^*} \leftarrow_\$ \mathbb{Z}_p^1$ and form the generic first message $\vec{u}_{t,\mathbf{b}^*} = \mathbf{G}^{\vec{\alpha}_{t,\mathbf{b}^*}}$. For each of the other three branches, choose $c_\mathbf{b} \leftarrow_\$ \mathcal{C}$ and simulate the generic protocol:

$$\vec{\alpha}_{z,\mathbf{b}} \leftarrow_\$ \mathbb{Z}_p^1, \qquad \vec{u}_{t,\mathbf{b}} = \mathbf{G}^{\vec{\alpha}_{z,\mathbf{b}}} \odot \vec{u}_\mathbf{b}^{-c_\mathbf{b}}.$$

Send all four first messages $\vec{u}_{t,\mathbf{b}}$ to the verifier.

ii. The verifier sends $c_{\text{OR}} \leftarrow_\$ \mathcal{C}$. The prover sets

$$c_{\mathbf{b}^*} = c_{\text{OR}} \oplus \bigoplus_{\mathbf{b} \neq \mathbf{b}^*} c_\mathbf{b}, \qquad \vec{\alpha}_{z,\mathbf{b}^*} = \vec{\alpha}_{t,\mathbf{b}^*} + c_{\mathbf{b}^*}\vec{\alpha} \pmod p,$$

and sends $(c_\mathbf{b}, \vec{\alpha}_{z,\mathbf{b}})$ for all four branches.

iii. The verifier checks the challenge split and the generic verification equation on every branch:

$$\bigoplus_{\mathbf{b} \in \{0,1\}^2} c_\mathbf{b} = c_{\text{OR}}, \qquad \mathbf{G}^{\vec{\alpha}_{z,\mathbf{b}}} \stackrel{?}{=} \vec{u}_{t,\mathbf{b}} \odot \vec{u}_\mathbf{b}^{c_\mathbf{b}}.$$

Here $\odot$ and vector exponentiation are entrywise. Each branch uses the same witness r in all three equations; the OR proof hides which bit pair is the real branch.

(5+5+5 = 15)

2. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ be a hash function modeled as a random oracle. For a key $K \leftarrow_\$ \mathbb{Z}_p$ define

$$F_K(x) \;:=\; \mathsf{H}(x)^K, \qquad x \in \{0,1\}^*.$$

Recall that a keyed function F with range $\mathbb{G}$ is a **pseudorandom function** (PRF) if it is deterministic and efficiently computable, and no PPT adversary wins the following game with probability noticeably better than $\frac{1}{2}$. Note that here the adversary also gets access to the random oracle.

Game (PRF security): challenger $\mathcal{C}$ versus PPT adversary A.

1. $\mathcal{C}$ samples $K \leftarrow_\$ \mathbb{Z}_p$ and $b \leftarrow_\$ \{0,1\}$. If $b = 1$ it also picks a uniformly random function $\mathcal{R} : \{0,1\}^* \to \mathbb{G}$ (sampled lazily).
2. Throughout the game A may query the random oracle: on input x it receives $\mathsf{H}(x)$.
3. For $i = 1, 2, \ldots, q$:
   (a) A chooses $x_i \in \{0,1\}^*$ and sends it to $\mathcal{C}$;
   (b) if $b = 0$, $\mathcal{C}$ replies with $F_K(x_i) = \mathsf{H}(x_i)^K$; if $b = 1$, $\mathcal{C}$ replies with $\mathcal{R}(x_i)$, i.e. a fresh uniform element of $\mathbb{G}$, kept consistent across repeated queries.
4. A outputs a guess b' and wins if $b' = b$.

(a) Prove that F is a pseudorandom function, based on DDH in the random oracle model.

Answer: **Setting up.** Let A be a PPT adversary in the game above, making at most q PRF queries. We may assume the queries $x_1, \ldots, x_q$ are distinct, and that A queries the random oracle on a point before querying the PRF on it — if not, the reduction below simply makes the missing H-query itself.

The key observation is that a random oracle into $\mathbb{G}$ returns an independent uniform group element on each fresh input, so we may write $\mathsf{H}(x_i) = g^{r_i}$ with $r_i \leftarrow_{\$} \mathbb{Z}_p$ uniform and independent ($r \mapsto g^r$ is a bijection $\mathbb{Z}_p \to \mathbb{G}$). The PRF answers are then $\mathsf{H}(x_i)^K = g^{r_i K}$: each query hands A one Diffie–Hellman pair $\left(g^{r_i}, g^{r_i K}\right)$ under the *same* secret exponent K. Replacing all of them by uniform elements is exactly the assumption of Question 1, with q slots instead of two.

Hybrids. For $j = 0, 1, \ldots, q$ let HYB_j be the game in which the first j queries are answered with fresh uniform elements of $\mathbb{G}$ and the remaining $q - j$ with the real $\mathsf{H}(x_i)^K$. Then HYB_0 is the $b = 0$ game and HYB_q is the $b = 1$ game, so it is enough to show $\mathsf{HYB}_{j-1} \overset{c}{\approx} \mathsf{HYB}_j$ for each j.

The reduction. Fix j. On DDH challenge $(g,\ A = g^K,\ B = g^y,\ C)$, the reduction B runs A and answers as follows, maintaining a table of oracle answers.

- *At the j-th query point, program the oracle to the challenge:*

$$\mathsf{H}(x_j) \ := \ B \ = \ g^y, \qquad \text{and answer the } j\text{-th PRF query with } C.$$

- *At every other point x_i,* draw $r_i \leftarrow_{\$} \mathbb{Z}_p$ and set $\mathsf{H}(x_i) := g^{r_i}$; note that B *knows* this exponent. Then answer the PRF query on x_i with

$$\text{a fresh } u_i \leftarrow_{\$} \mathbb{G} \quad \text{if } i < j, \qquad A^{r_i} \ \left(= \ g^{K r_i} = \mathsf{H}(x_i)^K \right) \quad \text{if } i > j.$$

- Output whatever A outputs.

Two remarks on why this is a faithful simulation. The oracle is perfect: B and the g^{r_i} are independent uniform elements of $\mathbb{G}$, which is exactly what a random oracle into $\mathbb{G}$ returns. And B answers the *real* queries $i > j$ without knowing K at all: it chose r_i, so $\mathsf{H}(x_i)^K = (g^K)^{r_i} = A^{r_i}$ is computable from the challenge element A.

Now read off the two cases:

$C = g^{Ky}$:	the j-th answer is $B^K = \mathsf{H}(x_j)^K$, i.e. real $\implies$ A's view is HYB_{j-1};
$C = g^z$ uniform :	the j-th answer is uniform and independent $\implies$ A's view is HYB_j.

So any advantage of A in separating HYB_{j-1} from HYB_j is a DDH advantage for B. Summing over the q hybrids, A's PRF advantage is negligible.

Note*: Two things worth saying. (i)* The random oracle is doing essential work. *Drop it — i.e. let the input be a group element u that the adversary picks, and set $F_K(u) = u^K$ — and the construction dies at once: query u and u^2, and test whether $F_K(u^2) = F_K(u)^2$. Hashing first is what stops the adversary from choosing algebraically* related *inputs; programming the oracle in the reduction is the same fact seen from the other side. (ii) The reduction hands A the element $A = g^K$, which the PRF game does not require it to know. That is deliberate: it shows F stays a PRF even when g^K is* published *— exactly what part (b) will need. This function is the DH-based OPRF underlying Privacy Pass and OPAQUE.*

(b) Alice holds the key K, and Bob wants Alice to evaluate the PRF for him on a *public* input x: Bob sends x, and Alice returns a value y. But Bob has no reason to believe that the y he receives is really $\mathsf{H}(x)^K$ — Alice could return anything at all. Propose a protocol so that Bob becomes convinced that y was computed correctly, *and* Alice is also assure that her secret key K remains hidden from Bob?

Describing the strategy is enough; you need not write out a full protocol. For instance, if you think a Σ-protocol is needed, it suffices to say *for which relation* it should be run, and why its properties would suffice to achieve both Alice and Bob's goals.

Answer: Two changes are needed, and the first is the one that is easy to forget.

1. Alice must publish a public key, once and for all:

$$\mathsf{pk}_A \ := \ g^K,$$

Publishing g^K costs nothing: by the note in part (a), F remains a PRF even when g^K is public.

2. Alice must attach a proof that y was produced with that same K. Bob can compute $\mathsf{H}(x)$ himself, since x and H are public. So the statement to be proven is

$$\exists\, \beta \in \mathbb{Z}_p : \qquad \mathsf{pk}_A = g^\beta \quad \text{and} \quad y = \mathsf{H}(x)^\beta,$$

i.e. $\log_g \mathsf{pk}_A = \log_{\mathsf{H}(x)} y$: the pair $(\mathsf{pk}_A,\, y)$ has the *same* exponent with respect to the two bases $\big(g,\ \mathsf{H}(x)\big)$. Equivalently,

$$\big(\ \underbrace{\mathsf{H}(x)}_{u},\ \underbrace{\mathsf{pk}_A}_{v},\ \underbrace{y}_{w}\ \big) \quad \text{is a DH-triple,}$$

which is the relation for which **Chaum–Pedersen** is a Σ-protocol, with witness $\beta = K$. Alice runs it as prover and Bob as verifier. Applying Fiat–Shamir with the challenge

$$c \ := \ \mathsf{H}'\big(\mathsf{pk}_A,\ \mathsf{H}(x),\ y,\ a\big)$$

— hashing the whole statement together with the commitment a — makes the proof a single string sent alongside y, and stops it from being replayed in another session.

Why this delivers precisely Bob's two requirements.

- *"y is correct" comes from soundness.* This follows from the special soundness of Chaum–Pedersen. Concretely: if Bob accepts, one can extract a β with $g^\beta = \mathsf{pk}_A$ and $\mathsf{H}(x)^\beta = y$; but in a prime-order group $\log_g$ is a bijection, so pk_A forces $\beta = K$, and therefore $y = \mathsf{H}(x)^K$.
- *"K stays secret" comes from zero knowledge.* Chaum–Pedersen is special HVZK: its transcript can be simulated from the statement $\big(\mathsf{H}(x), \mathsf{pk}_A, y\big)$ alone, which Bob already holds. So the proof tells Bob nothing beyond the truth of the statement — and in particular nothing about K — and this continues to hold as the proofs accumulate over many queries, since each is independently simulatable.

Note*: If Bob wants to hide x from Alice as well — the* oblivious *setting — he blinds before sending: draw $\rho \leftarrow_\$ \mathbb{Z}_p$ and send $\tilde{u} := \mathsf{H}(x)^\rho$; Alice returns $\tilde{y} := \tilde{u}^K$ with a Chaum–Pedersen proof on the triple $(\tilde{u}, \mathsf{pk}_A, \tilde{y})$; Bob unblinds as $y := \tilde{y}^{1/\rho} = \mathsf{H}(x)^K$. Since $\tilde{u}$ is a uniform group element, Alice learns nothing about x. The result is a* **verifiable oblivious PRF***, which is exactly what Privacy Pass and OPAQUE use. Note that the blinding step is the same re-randomization used in the reduction of part (a) — there a proof technique, here a protocol device.*

(5+5 = 10)

3. Fix two generators g, h of $\mathbb{G}$, produced by a setup that guarantees that *nobody* knows $\log_g h$. The **Pedersen commitment** to a message $m \in \mathbb{Z}_p$, with randomness $r \leftarrow_\$ \mathbb{Z}_p$, is

$$\mathsf{COM}(m;\, r) \ := \ g^m h^r \ \in \ \mathbb{G},$$

and to *open* a commitment C the committer reveals (m, r), which the receiver checks by recomputing.

(a) Suppose Bob wants a commitment on a message m from Alice. There is no setup. So, in the first step, Alice computes g, h and sends them to Bob. Then Alice sends the commitment $g^m h^r$. Do you think Alice can later open this commitment to any other value m' and thereby break binding? If yes, then how?

Answer: **Yes: Alice can choose the generators so that she knows their discrete-log relation.** She picks a generator g, chooses $\omega \leftarrow_\$ \mathbb{Z}_p \setminus \{0\}$, sets $h := g^\omega$, and sends (g, h) to Bob

while keeping ω. Since p is prime and $\omega \neq 0$, h is also a valid generator; checking that g, h are generators does not prevent the attack.

Alice commits to m using randomness r, sending

$$C = g^m h^r = g^{m+\omega r}.$$

Later, to open the same C to any chosen $m' \neq m$, she computes

$$\boxed{r' := r + (m - m')\omega^{-1} \pmod p}$$

and reveals (m', r'). The inverse exists because $\omega \neq 0$. Bob accepts, since

$$g^{m'} h^{r'} = g^{m'+\omega r'} = g^{m'+\omega r+(m-m')} = g^{m+\omega r} = C.$$

The two openings (m, r) and (m', r') reveal different messages for the same C, so binding fails. Alice need not solve a discrete logarithm: she already knows $\omega = \log_g h$ because she chose h herself. The setup guarantee in the question's introduction is absent in this part.

Note: *Note that, in this case no setup was done before, as it should have been. So it is very important to have a Setup done either by a trusted authority, or a collaborative protocol. Letting the committer chose the parameters may have devastating consequences.*

(b) Note that Pedersen's commitment is additively homomorphic. Alice now publishes three commitments C_1, C_2, C_3 each computed using independent randomness. Then she claims that the values inside them satisfy $m_3 = m_1 + m_2$. Describe a Σ-protocol with which she convinces a verifier (say, Bob) of this, revealing neither the m_i nor the randomness. (No need to prove its properties)

Answer: **Homomorphism.** With $C_i = g^{m_i} h^{r_i}$,

$$C_1 \cdot C_2 \;=\; g^{m_1+m_2} h^{r_1+r_2} \;=\; \mathsf{COM}\big(m_1 + m_2;\; r_1 + r_2\big),$$

so multiplying commitments adds the committed values and adds the randomness.

Reduce the claim to a commitment to zero. The verifier can compute

$$D \;:=\; \frac{C_1 C_2}{C_3} \;=\; g^{(m_1+m_2-m_3)}\, h^{(r_1+r_2-r_3)}$$

from the public commitments alone. The claim $m_3 = m_1 + m_2$ says exactly that the g-exponent vanishes, i.e. that D is a commitment to 0 — and a commitment to 0 is nothing but a power of h. So, setting $\rho := r_1 + r_2 - r_3$, which Alice knows, the statement to be proven is

$$\exists\, \rho \in \mathbb{Z}_p : \qquad D = h^\rho.$$

This is the Schnorr relation, run with base h in place of g; the witness is $\rho = r_1 + r_2 - r_3$. Explicitly:

$P(\rho,\, D)$		$V(C_1, C_2, C_3)$
$\rho_t \leftarrow_\$ \mathbb{Z}_p, \quad d_t := h^{\rho_t}$	$\xrightarrow{\;d_t\;}$	$D := C_1 C_2 / C_3$
	$\xleftarrow{\;c\;}$	$c \leftarrow_\$ \mathcal{C}$
$\rho_z := \rho_t + c\,\rho$	$\xrightarrow{\;\rho_z\;}$	
		CHECK: $\quad h^{\rho_z} \stackrel{?}{=} d_t \cdot D^c$

Why it is sound — and where binding does the work. Special soundness of Schnorr extracts ρ with $D = h^\rho$, i.e. the opening $(0, \rho)$ of D. Suppose Alice is also bound to openings (m_i, r_i) of the C_i (she will open them later, or has proved knowledge of them). Then D has the

second opening $\big(m_1{+}m_2{-}m_3,\ r_1{+}r_2{-}r_3\big)$. If the claim were false, these two openings would be distinct — and by part (a) that yields $\log_g h$. So under the discrete-log assumption a convincing proof forces $m_1 + m_2 - m_3 = 0$. Zero knowledge is inherited from Schnorr's special HVZK: the transcript is simulatable from D alone, so nothing about ρ, the r_i or the m_i leaks.

Note: *The same trick handles* any *linear relation among committed values: to prove* $\sum_i \lambda_i m_i = \kappa$ *for public* λ_i, κ*, form* $D := g^{-\kappa} \prod_i C_i^{\lambda_i}$ *and prove it is a power of* h*. Composed with Fiat–Shamir this is a single string, and it is the engine behind confidential transactions — amounts stay hidden while "inputs = outputs" is proven in public.*

(c) Can you use a Σ-protocol by which the committer proves *knowledge* of an opening (m, r) of a single commitment C, without revealing it?

Answer: Yes. This is essentially **Okamoto's Σ-protocol** for proving knowledge of a representation of C with respect to (g, h):

$$\mathcal{R} = \big\{\big((g, h, C), (m, r)\big) : C = g^m h^r\big\}.$$

Alice runs it with witness (m, r) and Bob verifies. Special soundness gives a proof of knowledge of an opening, while special honest-verifier zero knowledge hides the opening from an honest verifier.

(4+4+2 = 10)