Mid Semester - Fall, 2024
M. Tech. Cryptology & Security - 2^{nd} Year
Advanced Cryptology
(with Answers)

November 25, 2025

Maximum Marks: 25
Maximum Time: 3 hr
Open note / book exam

Instructions

- Maximum marks is 25. Total marks provided in the paper: 36.
- Be short and precise. Partial marking will be provided for a partially correct answer/attempt.
- 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 are denoted by $\mathbb{Z}$. The ring of all integers modulo n is denoted by $\mathbb{Z}_n$. Below $\kappa \in \mathbb{N}$ denotes the security parameter throughout (for example, if 128 bit security is desired from the system, then κ is set to 128, this is often equal to the key-length). A uniform random sample from a domain D is denoted as $s \leftarrow_{\$} D$. We assume $\langle g \rangle = \mathbb{G}$ to be a cyclic group of prime order p.

1. Recall El-Gamal Encryption Scheme. Consider that as a commitment scheme.
   (a) Prove that it is unconditionally binding.
   (b) Prove that it is computationally hiding.
   (c) What minimal change you can make so that the scheme becomes unconditionally hiding and computationally binding?

   (4+4+2)

   Answer: (a) An exponentiated ElGamal ciphertext has the form $(g^r, \mathsf{pk}^r.g^m)$, where m is a message, and r a randomness. Also $\mathsf{pk} = g^{\mathsf{sk}}$ is the public-key. To see **unconditional binding**, note that: fixing the above ciphertext fixes r and $s = (r \cdot \mathsf{sk} + m)$. In addition, the public key fixes sk. So, together, fixing r, s, sk also fixes m uniquely. This means that the ciphertext fixes unique m and r, and even unbounded (all powerful) adversary can not come up with another $m' \neq m$ which would be consistent with this specific ciphertext.

   (b) **Computational hiding** is the same as CPA-security. This holds assuming DDH. `Elaboration Needed.`

   (c) Just removing the first component makes the ciphertext a Pedersen commitment $\mathsf{pk}^r.g^m$, which is unconditionally hiding and computationally binding. This is because, removing the first component does not fix r, and hence for an unbounded adversary only s, sk are known – this is not sufficient to uniquely determine m (or r).

2. Schnorr's Non-interactive Sigma Protocol for proving knowledge of exponents, for relations of the form $(x, y = g^x)$ is described below:

   - The prover algorithm with input (x, y) works as follows:
     - choose a random $r \leftarrow_\$ \mathbb{Z}_p$ and compute $a := g^r$;
     - use a hash function to compute the challenge $c := \mathsf{H}(a)$;
     - finally compute $z = cx + r$.
     - output (a, c, z) as a proof.
   - The verifier algorithm, with input $(y, (a, c, z))$ checks
     - $c \stackrel{?}{=} \mathsf{H}(a)$;
     - $a \cdot y^c \stackrel{?}{=} g^z$.

   (a) If H is a linear function, such as $\mathsf{H}_{u,v}(\alpha) = u\alpha + v$, ($u, v$ are known to the prover), then will there be any problem in the security (soundness or zero-knowledge) of the protocol? Explain with precise arguments.
   Answer: There are two cases. First, if the values u, v are chosen in the setup, and are given to both prover and verifier. In that case, the forking lemma / rewinding of the random oracle would not work, so the special soundness proof would fail. Although a concrete attack against soundness is not immediate, because to produce a valid transcript without witness, the prover has to program c to a computed from the verification equation – this programming will not be possible since u, v are known to verifier a priori and can not controlled by the prover. However, if u, v are chosen by the prover during the protocol execution, and (then possibly is sent to the verifier along with the proof additionally), then the prover can program any c and a, and thereby can concretely break soundness. **Note**: *Note that, for this to work there should be an efficient hashing from group to field, currently it is not made explicit.*

   (b) What happens if prover chooses $r = 0$ by mistake – will there be an issue in security? Explain with precise arguments.
   Answer: If $r = 0$, then verifier would be able to compute x – this would break zero-knowledge completely. Formally, no simulator can simulate cx correctly without knowing x. Note that, the simulator is only given y as input, so by discrete log hardness it can not learn x. In other words, if there is a simulator that can simulate the transcript without knowing x, it is possible to construct an algorithm (reduction) to solve discrete log. **Note**: *Note that, this holds for any predictable r which is not necessarily 0.*

   (4+4)

3. Let $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function such that for any N, an N-tuple of uniform random $(x_1, x_2, \ldots, x_N) \leftarrow_\$ \mathcal{X}^N$, then the N-tuple $(f_k(x_1), \ldots, f_k(x_N)) \in \mathcal{Y}^N$ is computationally indistinguishable from uniformly random N-tuple $(y_1, \ldots, y_N) \leftarrow_\$ \mathcal{Y}^N$. We call such functions *weak* PRFs (recall that PRFs are stronger primitives, where the same holds over arbitrary, but distinct x_i's values, instead of random x_i's).

   - Given a weak PRF, and a random oracle $\mathsf{H} : \mathcal{X} \to \mathcal{X}$, how can you construct a PRF $f' : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, such that for *any* (possibly adversarially chosen) $(x_1, x_2, \ldots, x_N)$ ($x_i \neq x_j$ if $i \neq j$), then the N-tuple $(f'_k(x_1), \ldots, f'_k(x_N)) \in \mathcal{Y}^N$ is computationally indistinguishable from uniformly random N-tuple $(y_1, \ldots, y_N) \leftarrow_\$ \mathcal{Y}^N$is computationally indistinguishable from uniform random $y \leftarrow_\$ \mathcal{Y}$? Write the proposed construction.
   - Analyze with a reduction that if f is a weak PRF and H is a random oracle, then f' is a PRF.

   (2+4)

   Answer: The answer is given in two parts:

   - Given a random oracle $\mathsf{H} : \mathcal{X} \to \mathcal{X}$, and a *weak* PRF $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we can construct a PRF $f' : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ as $f'(x) = f(\mathsf{H}(x))$.

- To reduce the PRF construction to weak PRF in the random oracles model, we construct a reduction, which breaks weak PRF whenever there is an adversary that breaks PRF. Also the reduction answers the random oracle queries from the PRF adversary. It works as follows:
  - The reduction maintains a list L containing triples (x, x', y), where $x' = \mathsf{H}(x)$.
  - If the reduction receives an RO query x, then it checks whether there is $(x, x', y) \in L_{\mathsf{H}}$, if it does then it replies with x', otherwise it makes a query to its weak PRF to obtain pair (x', y), and reply with x' and program $\mathsf{H}(x) = x'$. Also it appends (x, x', y) into L.
  - When the reduction receives a PRF query on x, then it first checks whether there is $(x, x', y) \in L$. If yes, then reply with y. Otherwise it makes a query to weak PRF challenger to obtain (x', y), and reply with x' and program $\mathsf{H}(x) = x'$. Also it appends (x, x', y) into L.

  Clearly, the reduction simulates the real game for PRF f', when it obtains real pair (x', y) such that $f(x') = y$ from the weak PRF challenger. On the other hand when the reduction obtains random (x, y), then it also simulates the random function to the adversary. So, whenever the PRF adversary breaks the PRF, the reduction breaks the weka PRF. **Note**: *Note that, the number of weak PRF queries the reduction makes to its challenger is the sum of the RO queries and PRF queries it receives from the PRF adversary.*

4. Let $\mathsf{AHE} = (\mathsf{KGEN}, \mathsf{ENC}, \mathsf{DEC})$ be any arbitrary additively homomorphic encryption scheme. Show with a concrete attack that AHE can not be CCA-2 secure. (4)

   Answer: Given an AHE challenge ciphertext $c^* = \mathsf{ENC}_{\mathsf{pk}}(m_b)$, a CCA2 adversary may compute $c^\dagger = ADD(c^*, c_1)$ where $c_1 = \mathsf{ENC}_{\mathsf{pk}}(1)$. Then it queries the CCA2 decryption oracle with $c^\dagger$ to obtain back the decryption $m_b + 1$, from which m_b can be recovered.

5. Assume that the CDH is *hard* in group $\mathbb{G}$, that is: given g, g^x, g^y computing g^{xy} is *hard*, where $(x, y) \leftarrow_\$ \mathbb{Z}_p^2$. Now consider the following assumption, called computational Square-DH (CSDH): given g, g^x computing g^{x^2} is computationally *hard* (again $x \leftarrow_\$ \mathbb{Z}_p$). Show that, these assumptions are *equivalent*, that is, specifically:

   (a) If CDH is *easy*, then so is CSDH. Or in other words, given an adversary (solver) that solves CDH, describe a reduction which can use that adversary to solve CSDH *efficiently*. Note that, a solver for CDH, on input (g, g^x, g^y) (for uniform random $(x, y) \in \mathbb{Z}_p^2$), returns g^{xy}.
   Answer: The reduction, on receiving (g, g^x) from CSDH challenger, works as follows:
   - Choose uniform random $r \leftarrow_\$ \mathbb{Z}_p$.
   - Send (g, g^x, g^{xr}) to CDH challenger.
   - Once the challenger sends back h, return $h^{1/r}$ to the CSDH challenger.

   Now, first note that the distribution of (g, g^x, g^{xr}) is identical to a CDH challenge (g, g^x, g^y), since r is random. Therefore the CDH adversary can not detect the difference. Once CDH adversary returns $h = g^{x^2 r}$, then $h^{1/r} = g^{x^2}$ is the correct solution of given CSDH instance.

   (b) In the other direction, similarly use a reduction to show that if CSDH is easy, then so is CDH. Note that a solver for CSDH (that, on input (g, g^x), returns g^{x^2}), one can *easily* solve CDH. (Hint: Remember that, when you are using any algorithm as oracle, that only gives you correct answer if the input is distributed identically as actual inputs. For example, you can not give (g, g^x, g^x) as input to the CDH solver, because the exponents (x, x) is not uniform random in $\mathbb{Z}_p^2$. The CDH solver will be able to detect that the second and third inputs are identical and may decline to return the answer.)
   Answer: The reduction, on receiving (g, g^x, g^y) from CDH challenger, works as follows:
   - Send $(g, g^{(x+y)})$ to CSDH adversary to obtain back $h_1 = g^{(x+y)^2}$.
   - Then send $(g.g^{(x-y)})$ to CSDH adversary to obtain back $h_2 = g^{(x-2)^2}$.
   - Finally it computes $g^{xy} = (h_1/h_2)^{1/4}$.

   The correctness follows as $xy = 1/4((x+y)^2 - (x-y)^2)$. **Note**: *Note that, it requires two calls to the CDH adversary for solving.*

   (4+4)