UC Berkeley — CS 276 : Cryptography
Prof. Sanjam Garg

Final Solutions

Final
April 30, 2026

Name:

SID:

- **Time: You have *1 hour and 20 minutes* (80 minutes) to complete the exam. The exam runs from 11:10 AM to 12:30 PM. No extra time will be given after 12:30 PM.**
- **After the exam starts, write your name and SID on every odd-numbered page. We reserve the right to deduct points if you do not, and you will not be allowed to do so after time is called.**
- **For short questions, your answers must be written clearly inside the box region. Any answer outside the box will not be graded. For longer questions, if you run out of space, you must clearly mention in the space provided for the question if part of your answer is elsewhere.**
- **You may consult at most *1 double-sided sheet of handwritten notes*. Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted for looking up content.**
- **There are 6 pages on the exam (counting this one). Notify a proctor immediately if a page is missing.**

1. **[10 points] True/False**

   (a) Digital signature schemes can be constructed from IBE. ☐

   True

   (b) Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function. The function $g(x) = f(x) \| f(x \oplus 1^n)$ is also a one-way function. ☐

   False. Consider a OWF that leaks the first half or second half of the input depending on the first bit of the input.

   (c) In the context of private-key encryption, the standard notion of IND-CCA security where the adversary is permitted to make decryption oracle queries after receiving the challenge ciphertext is equivalent to the notion where decryption oracle queries are only permitted before the challenge ciphertext is issued. ☐

   False

   (d) Given two functions $G_1, G_2 : \{0,1\}^k \to \{0,1\}^n$ such that at least one of them is a PRG, $G'(s_1, s_2) = G_1(s_1) \oplus G_2(s_2)$ is always a PRG. ☐

   True. This is a PRG combiner – the output looks pseudorandom as long as either G_1 or G_2 is a PRG.

   (e) Every CRHF is a UOWHF. ☐

   True

2. **[30 points] MAC-then-encrypt insecurity.**

   A **MAC scheme** is a tuple $(\text{KGen}, \text{Mac}, \text{Verify})$ where: $k \leftarrow \text{KGen}(1^n)$, $t \leftarrow \text{Mac}(k, m)$, $\text{Verify}(k, m, t) \in \{0,1\}$, with $k, m, t \in \{0,1\}^*$.

   Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a CPA-secure public-key encryption scheme where keys are generated as $(pk, sk) \leftarrow \mathcal{G}(1^n)$, ciphertexts and messages are bitstrings, and let $(\text{KGen}, \text{Mac}, \text{Verify})$ be an arbitrary secure MAC scheme. We construct a new encryption scheme $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ where $\mathcal{G}' := \mathcal{G}$ and:

$$\mathcal{E}'(pk, m) := \left\{ \begin{array}{l} k \xleftarrow{\$} \text{KGen}(1^n), \\ c \xleftarrow{\$} \mathcal{E}\big(pk,\ (k, m)\big), \\ t \xleftarrow{\$} \text{Mac}(k, c), \\ \text{output } (c, t) \end{array} \right\} \qquad \mathcal{D}'\big(sk, (c, t)\big) := \left\{ \begin{array}{l} (k, m) \leftarrow \mathcal{D}(sk, c), \\ \text{if } \text{Verify}(k, c, t) = 1 \text{ output } m, \\ \text{otherwise output } \textbf{reject} \end{array} \right\}$$

   (a) Design a CPA-secure encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ for which the transformed $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ is not CCA-secure. To construct this scheme, you may use an arbitrary CPA-secure encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$.

$\mathcal{G}(1^n)$:

$(pk, sk) \xleftarrow{\$} \text{Gen}(1^n)$, output (pk, sk)

$\mathcal{E}(pk, m)$:

$c \xleftarrow{\$} \text{Enc}(pk, m)$, output $c\|0$

$\mathcal{D}(sk, c)$:

Parse c as $c'\|b$ where $b \in \{0,1\}$. Parse $\text{Dec}(sk, c')$ as (k, m). If $b = 0$, output (k, m), else output $(0^n, m)$.

(b) Prove that $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ is not CCA-secure by constructing an adversary $\mathcal{A}$ that wins the CCA game with non-negligible advantage.

The adversary $\mathcal{A}$ proceeds as follows:

i. Output two distinct messages m_0, m_1 of the same length to the challenge oracle.
ii. Receive the challenge ciphertext (c^*, t^*) where $c^* = c'\|0$.
iii. Modify the ciphertext component to $\tilde{c} = c'\|1$. Note that $\tilde{c} \neq c^*$.
iv. Compute a valid MAC for $\tilde{c}$ under the fixed key 0^n: $\tilde{t} \leftarrow \text{Mac}(0^n, \tilde{c})$.
v. Query the decryption oracle with the modified ciphertext $(\tilde{c}, \tilde{t})$.
vi. The oracle computes $\mathcal{D}(sk, \tilde{c})$, which parses $\tilde{c}$ as $c'\|1$, decrypts c' to (k, m_b), and since $b = 1$, outputs $(0^n, m_b)$. $\mathcal{D}'$ then verifies the MAC using the key 0^n on the ciphertext $\tilde{c}$ with tag $\tilde{t}$, which succeeds. The oracle returns m_b.
vii. $\mathcal{A}$ outputs 0 if the oracle returns m_0, and 1 otherwise.

This adversary wins the CCA game with probability 1, which is non-negligible, proving the scheme is not CCA-secure.

3. **[30 points] Zero-Knowledge for Graph Hamiltonicity.**

Let $G = (V, E)$ be a graph on n vertices. Consider the following interactive proof system for the language

$$L_{\text{HAM}} = \{G : G \text{ contains a Hamiltonian cycle}\}.$$

The common input is the adjacency matrix A of G. The prover P knows a Hamiltonian cycle H in G. We use a computationally hiding and perfectly binding commitment scheme Com.

A *cycle graph* on n vertices is a graph obtained by choosing a permutation π of the vertices and including exactly the edges $\{\pi(i), \pi(i+1)\}$ for $i = 1, \ldots, n-1$ together with $\{\pi(n), \pi(1)\}$; equivalently, its edges form a single Hamiltonian cycle. A *random* cycle graph is sampled by choosing π uniformly at random.

Protocol:

(a) **Prover's first message:** P picks a random cycle graph C on n vertices. The prover then computes the adjacency matrix M of C, and sends commitments to all entries of M.

(b) **Verifier's challenge:** V sends a uniformly random bit $b \xleftarrow{\$} \{0,1\}$.

(c) **Prover's response:**

- **If $b = 0$:** P decommits to M. V verifies all decommitments are correct and form a valid cycle graph.
- **If $b = 1$:** Let $\sigma(H) = C_n$ and $C = \pi(C_n)$. P reveals $\phi = \pi \circ \sigma$ and decommits to entries in M for non-edges of $\phi(G)$. V verifies these decommitments are correct and equal 0.

(a) Construct an honest-verifier zero-knowledge (HVZK) simulator S that, on input G, outputs a transcript (prover's first message, verifier's challenge, prover's response) without knowing the Hamiltonian cycle H.

S on input G picks $b \xleftarrow{\$} \{0,1\}$.

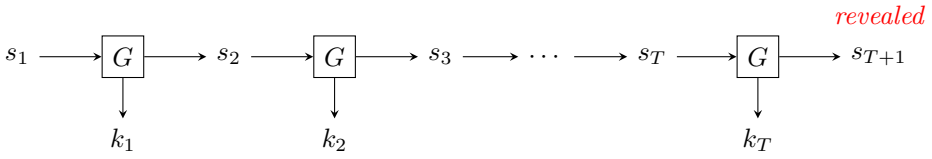
- $b = 0$: Output commitments to a random cycle graph C, challenge $b = 0$, and decommitments to all entries.
- $b = 1$: Pick a random permutation ϕ. Output commitments to an all-zeros matrix, challenge $b = 1$, permutation ϕ, and valid decommitments to 0 for non-edges of $\phi(G)$.

(b) Prove that the output of your simulator S is computationally indistinguishable from the real transcript of the protocol.

- For $b = 0$, the simulated transcript is identical to the real protocol, as both output valid commitments and decommitments to a random cycle graph.
- For $b = 1$, both real and simulated transcripts provide a uniformly random permutation ϕ and decommit to 0 for non-edges of $\phi(G)$. The only difference is the unopened commitments (zeros in simulation, ones and zeros in real). By the computational hiding property of Com, these unopened commitments are indistinguishable.

4. **[30 points] Forward-Secure Encryption.**

Forward Security. Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a secure pseudorandom generator, and let (E, D) be a CPA-secure symmetric encryption scheme with key space $\{0,1\}^n$. Consider a stateful symmetric encryption scheme operating over T time periods where the state is updated via $(s_{t+1}, k_t) := G(s_t)$, and the key k_t is used for encryption during time slot t, as illustrated below.



We say this scheme is *forward-secure* if for every PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible:

(a) The challenger samples a random bit $b \xleftarrow{\$} \{0,1\}$ and an initial state $s_1 \xleftarrow{\$} \{0,1\}^n$.

(b) For each time period $t = 1, \ldots, T$:

- The challenger computes $(s_{t+1}, k_t) := G(s_t)$.

- $\mathcal{A}$ can adaptively query an encryption oracle with messages m to receive $c \xleftarrow{\$} E(k_t, m)$.
- *Only during the last time period* (i.e., $t = T$), $\mathcal{A}$ can additionally query a challenge oracle with pairs of equal-length messages (m_0, m_1) to receive $c^* \xleftarrow{\$} E(k_T, m_b)$.
- The old state s_t is then erased.

(c) At the end of time T, $\mathcal{A}$ is given the compromised state s_{T+1} (the seed from which all keys for time $T+1$ and beyond would be derived).

(d) $\mathcal{A}$ outputs a guess b', and wins if $b' = b$.

To prove forward security, it is sufficient to show that the derived keys are pseudorandom even given the compromised state. In the remainder of this problem, we will prove that the following two distributions are computationally indistinguishable:

$$\left\{ (k_1, \ldots, k_T,\ s_{T+1}) \ :\ s_1 \xleftarrow{\$} \{0,1\}^n;\ (s_{i+1}, k_i) := G(s_i) \text{ for } i = 1, \ldots, T \right\}$$

$$\approx_c$$

$$\left\{ (k_1, \ldots, k_T,\ s_{T+1}) \ :\ s_{T+1} \xleftarrow{\$} \{0,1\}^n;\ k_1, \ldots, k_T \xleftarrow{\$} \{0,1\}^n \right\}$$

(a) Define a sequence of hybrids $H_0, H_1, \ldots, H_T$ to prove forward security. For each hybrid H_i, specify how $k_1, \ldots, k_T$ and s_{T+1} are generated. H_0 and H_T are given below; fill in H_i for general $1 \le i \le T-1$.

H_0: Sample $s_1 \xleftarrow{\$} \{0,1\}^n$. Compute $(s_{j+1}, k_j) := G(s_j)$ for all $j = 1, \ldots, T$. Output $(k_1, \ldots, k_T,\ s_{T+1})$.

H_i (for $1 \le i \le T-1$):

Sample $k_1, \ldots, k_i \xleftarrow{\$} \{0,1\}^n$ and $s_{i+1} \xleftarrow{\$} \{0,1\}^n$ independently and uniformly at random. Compute $(s_{j+1}, k_j) := G(s_j)$ for all $j = i+1, \ldots, T$. Output $(k_1, \ldots, k_T,\ s_{T+1})$.

H_T: Sample $k_1, \ldots, k_T \xleftarrow{\$} \{0,1\}^n$ and $s_{T+1} \xleftarrow{\$} \{0,1\}^n$ independently and uniformly at random. Output $(k_1, \ldots, k_T,\ s_{T+1})$.

(b) Prove that for any $i \ge 1$, hybrid H_{i-1} is computationally indistinguishable from H_i by constructing a reduction algorithm $\mathcal{B}$ that uses a distinguisher between the two hybrids to break the security of G.

Assume there is a distinguisher D that distinguishes H_{i-1} and H_i with non-negligible advantage. We construct an adversary $\mathcal{B}$ against the PRG G:

i. $\mathcal{B}$ receives a challenge string $y \in \{0,1\}^{2n}$. It parses y as $(s^*, k^*) \in \{0,1\}^n \times \{0,1\}^n$.
ii. $\mathcal{B}$ sets $s_{i+1} = s^*$ and $k_i = k^*$.
iii. $\mathcal{B}$ samples $k_1, \ldots, k_{i-1} \xleftarrow{\$} \{0,1\}^n$ independently and uniformly at random.
iv. For $j = i+1, \ldots, T$, $\mathcal{B}$ computes $(s_{j+1}, k_j) := G(s_j)$.
v. $\mathcal{B}$ invokes D on the input $(k_1, \ldots, k_T,\ s_{T+1})$ and outputs whatever D outputs.

If the challenge y was generated as $G(s)$ for a random seed $s \xleftarrow{\$} \{0,1\}^n$, then (s_{i+1}, k_i) is correctly distributed as $G(s_i)$ for a random s_i, and the view provided to D exactly matches the distribution of H_{i-1}. If the challenge y was drawn uniformly at random from

$\{0,1\}^{2n}$, then s_{i+1} and k_i are uniformly random and independent, and the view exactly matches the distribution of H_i. Thus, $\mathcal{B}$ breaks the PRG security of G with the same non-negligible advantage as D.