CS 388H: Cryptography **Fall 2021**

Homework 2: Symmetric Cryptography

Due: October 7, 2021 at 11:59pm (Submit on Gradescope) **Instructor:** David Wu

Instructions. You **must** typeset your solution in LaTeX using the provided template:

`https://www.cs.utexas.edu/~dwu4/courses/fa21/static/homework.tex`

You must submit your problem set via Gradescope (accessible through Canvas).

Collaboration Policy. You may discuss your general *high-level* strategy with other students, but you may not share any written documents or code. You should not search online for solutions to these problems. If you do consult external sources, you must cite them in your submission. You must include the names of all of your collaborators with your submission. Refer to the official course policies for the full details.

Problem 1: CBC Padding Oracle Attack [15 points]. Recall that when using a block cipher in CBC mode, the message must be an even multiple of the block size. When encrypting messages whose length is not an even multiple of the block size, the message must first be padded. In the TLS protocol (used for securing traffic on the web), if v bytes of padding are needed, then v bytes with value $(v-1)$ are appended to the message. As a concrete example, if 1 byte of padding is needed, a single byte with value 0 is appended to the message before applying CBC encryption. In TLS, the record layer is secured using an approach called "MAC-then-Encrypt[1]" (which as we will soon see, is not the ideal combination). At decryption time, the ciphertext is first decrypted (and the padding verified) *before* checking the MAC. In older versions of OpenSSL, the library reports whether a decryption failure was due to a "bad pad" or due to a "MAC verification failure." One might think that it was beneficial to provide an informative error message on decryption failure. As you will show in this problem, this turns out to be a disaster for security.

Suppose an adversary has intercepted a target ciphertext ct encrypted using AES-CBC. Let ct_i be any non-IV block in ct. Let m_i be the associated message block. Show that if the adversary is able to submit ciphertexts to a CBC decryption oracle and learn whether the padding was valid or not, then it can learn the last byte of m_i *with probability 1* by making at most 512 queries. Here, the CBC decryption oracle only says whether the ciphertext was properly padded or not; it does *not* provide the output of the decryption if successful. Then, show how to extend your attack to recover *all* of m_i. **Hint:** Start by showing how to test whether the last byte of m_i is some value t by making 2 queries to the decryption oracle.

Remark: Are there settings where the server would repeatedly decrypt ciphertexts of the user's choosing? It turns out that when using IMAP (the protocol email clients use to fetch email) over TLS, the IMAP client will repeatedly send the user's password to the IMAP server to authenticate. With the above padding oracle (implemented using a "timing channel"), an adversary can recover the client's password in *less than* an hour! This problem shows that if a decryption failure occurs, the library should provide *minimal* information on the cause of the error. This type of "padding oracle" attack was the basis of the "Lucky 13" attack on TLS 1.0 (2013)—many years after they were first discovered (2002) and thought to be patched!

[1] In MAC-then-encrypt, the encryption algorithm first computes a MAC t on the message m, and the ciphertext is the encryption of the message-tag pair (m, t).

Problem 2: Cryptographic Combiners [15 points]. Suppose we have two candidate constructions Π_1, Π_2 of a cryptographic primitive, but we are not sure which of them is secure. A cryptographic combiner provides a way to use Π_1 and Π_2 to obtain a new construction Π such that Π is secure if at least one of Π_1, Π_2 is secure (*without* needing to know which of Π_1 or Π_2 is secure). Combiners can be used to "hedge our bets" in the sense that a future compromise of one of Π_1 or Π_2 would not compromise the security of Π. In this problem, we will study candidate combiners for different cryptographic primitives.

(a) Let $G_1, G_2 \colon \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be arbitrary PRG candidates. Define the function $G(s_1, s_2) := G_1(s_1) \oplus G_2(s_2)$. **Prove or disprove:** if at least one of G_1 or G_2 is a secure PRG, then G is a secure PRG.

(b) Let $H_1, H_2 \colon \{0,1\}^* \to \{0,1\}^\lambda$ be arbitrary collision-resistant hash function candidates. Define the function $H(x) := H_1(H_2(x))$. **Prove or disprove:** if at least one of H_1 or H_2 is collision-resistant, then H is collision-resistant.

(c) Let $(\mathsf{Sign}_1, \mathsf{Verify}_1)$ and $(\mathsf{Sign}_2, \mathsf{Verify}_2)$ be arbitrary MAC candidates[2]. Define $(\mathsf{Sign}, \mathsf{Verify})$ as follows:

- $\mathsf{Sign}((k_1, k_2), m)$: Output (t_1, t_2) where $t_1 \leftarrow \mathsf{Sign}_1(k_1, m)$ and $t_2 \leftarrow \mathsf{Sign}_2(k_2, m)$.
- $\mathsf{Verify}((k_1, k_2), (t_1, t_2))$: Output 1 if $\mathsf{Verify}_1(k_1, m, t_1) = 1 = \mathsf{Verify}_2(k_2, m, t_2)$ and 0 otherwise.

Prove or disprove: if at least one of $(\mathsf{Sign}_1, \mathsf{Verify}_1)$ or $(\mathsf{Sign}_2, \mathsf{Verify}_2)$ is a secure MAC, then $(\mathsf{Sign}, \mathsf{Verify})$ is a secure MAC.

Problem 3: Adaptive vs. Non-Adaptive Security [25 points]. Let $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF. Recall that in the PRF security game, the adversary issues queries to the challenger on inputs $x_1, \ldots, x_Q \in \mathcal{X}$, and the challenger replies with $y_1 = f(x_1), \ldots, y_Q = f(x_Q)$, where $f = F(k, \cdot)$ is either a pseudorandom function or a truly random function $f \xleftarrow{R} \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$. In this case, we say that the adversary makes *adaptive* queries since x_i can depend on the challenger's responses $y_1, \ldots, y_{i-1}$ to its previous queries.

We can also consider a weaker *non-adaptive* notion of security where the adversary has to submit all of its queries upfront. Namely, at the beginning of the security game, it gives $x_1, \ldots, x_Q$ to the challenger and receives $y_1, \ldots, y_Q$. This is sometimes referred to as a "selective" notion of security since the adversary is "selecting" its queries in advance.

(a) If F satisfies adaptive security, it is clear that it also satisfies non-adaptive security. Show that the converse is not true. Namely, using $F \colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, construct a new PRF F' that satisfies non-adaptive security if F is a secure PRF (according to the standard adaptive definition), but does not satisfy adaptive security even against adversaries that make just 2 queries. You should prove that (1) F' satisfies non-adaptive security (assuming security of F); and (2) F' does not satisfy adaptive security against an adversary that makes two queries. To simplify things, you may assume that $\mathcal{X} = \mathcal{Y} = \{0,1\}^\lambda$ where λ is the security parameter. **Hint:** Try changing the value of F on a single input. To prove non-adaptive security of F', consider a hybrid structure where you first replace all invocations of $F(k, \cdot)$ (in the evaluation of F') with $f(\cdot)$ where f is a uniform random function. Then, show that this experiment is indistinguishable from the experiment where the challenger answers all of the adversary's queries using a truly random function.

While a selective notion of security may seem unrealistic or overly restrictive on the capabilities of the adversary, if oftentimes implies some form of the usual adaptive notion of security if we make a stronger

[2] Namely, you can assume that they are correct (but could be arbitrarily broken).

hardness assumption. This technique of converting a selectively-secure scheme into an adaptively-secure one is called *complexity leveraging*. In many settings in cryptography, it is much easier to prove security against selective adversaries, and complexity leveraging allows us to show that the same (or a closely-related) construction is essentially adaptively secure just under a stronger assumption. In this problem, we will develop this approach for PRFs.

(b) Suppose $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a PRF with security against non-adaptive adversaries. Let $\mathcal{A}$ be an efficient adversary for F that makes up to Q *adaptive* queries to F. Moreover, suppose we restrict $\mathcal{A}$ to only make queries on inputs $x \in S$ where $S \subseteq \mathcal{X}$. The set S is *public* and *fixed* (and there is an efficient algorithm to sample elements from S). (We can set $S = \mathcal{X}$, in which case this is a vacuous restriction, but being able to consider smaller subsets $S \subset \mathcal{X}$ gives a tighter bound). Show that for every $\mathcal{A}$ that makes up to Q adaptive queries (where every query must be an element of the set S), there exists an efficient non-adaptive adversary $\mathcal{B}$ for F such that

$$\mathsf{NonAdaptivePRFAdv}[\mathcal{B}, F] \geq \frac{1}{|S|^Q} \mathsf{PRFAdv}_{Q,S}[\mathcal{A}, F].$$

Here, $\mathsf{PRFAdv}_{Q,S}[\mathcal{A}, F]$ is the advantage of $\mathcal{A}$ in the standard PRF security game restricted to Q queries from the set S, and $\mathsf{NonAdaptivePRFAdv}[\mathcal{B}, F]$ is the advantage of $\mathcal{B}$ in the non-adaptive PRF security game. **Hint:** Without loss of generality, you can assume that algorithm $\mathcal{A}$ always makes *exactly* Q queries (since any adversary making fewer than Q queries can be converted into one that makes exactly Q queries). Observe that the number of possible query sequences $\mathcal{A}$ can submit in the adaptive security game is exactly $|S|^Q$.

(c) Let $\{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a collection of functions where $F_\lambda\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$. Suppose that there exists a positive constant $\varepsilon < 1$ such that for all security parameters $\lambda \in \mathbb{N}$ and all adversaries $\mathcal{A}$ running in time $\mathrm{poly}(\lambda)$,

$$\mathsf{NonAdaptivePRFAdv}[\mathcal{A}, F_\lambda] \leq 2^{-\lambda^\varepsilon}.$$

This is an example of a *sub-exponential* hardness assumption. Show how to use $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ to construct a new family of PRFs $F' = \{F'_\lambda\}_{\lambda \in \mathbb{N}}$, where each $F'_\lambda\colon \{0,1\}^{\ell_{Q,\varepsilon}(\lambda)} \times \{0,1\}^\lambda \to \{0,1\}$ is defined over the same domain and range as F_λ. Moreover, your PRF family F' should be secure against all *adaptive* PRF adversaries making up to Q queries (where Q is a fixed polynomial in λ): namely, for all security parameters $\lambda \in \mathbb{N}$ and all efficient adversaries $\mathcal{A}$ for F' that makes up to Q queries,

$$\mathsf{PRFAdv}_Q[\mathcal{A}, F'_\lambda] \leq \mathrm{negl}(\lambda).$$

Your construction should specify your choice of the key length $\ell_{Q,\varepsilon}(\lambda)$. Use the result from Part (b) to prove security of F'. **Hint:** Define F'_λ using $F_{\ell_{Q,\varepsilon}(\lambda)}$, where $\ell_{Q,\varepsilon}(\lambda)$ is a function of λ, Q, and ε.

Remark: In the case of PRFs, complexity leveraging only allowed us to argue security against adversaries making an *a priori* bounded number of queries rather than full adaptive security.

(d) Suppose you apply the complexity leveraging transformation from Part (c) to your non-adaptive PRF from Part (a). Does your adaptive attack (using 2 queries) still break the resulting scheme? Give a brief (and *informal*) explanation (e.g., 2-3 sentences).

Problem 4: Encrypting Twice, Revisited [20 points]. Let $(\mathsf{Encrypt}, \mathsf{Decrypt})$ be a symmetric authenticated encryption scheme. For each of the following constructions $(\mathsf{Encrypt}', \mathsf{Decrypt}')$, state whether they are authenticated encryption schemes. If so, give a proof (you need to show *both* CPA-security and ciphertext integrity); otherwise, give an attack.

(a) Define $(\mathsf{Encrypt}', \mathsf{Decrypt}')$ as follows:

$$\mathsf{Encrypt}'(k, m) = \big(\mathsf{Encrypt}(k, m), \mathsf{Encrypt}(k, m)\big)$$

$$\mathsf{Decrypt}'(k, (c_1, c_2)) = \begin{cases} \mathsf{Decrypt}(k, c_1) & \mathsf{Decrypt}(k, c_1) = \mathsf{Decrypt}(k, c_2) \\ \perp & \text{otherwise.} \end{cases}$$

(b) Define $(\mathsf{Encrypt}', \mathsf{Decrypt}')$ as follows:

$$\mathsf{Encrypt}'(k, m) = \big(c, c\big) \text{ where } c \leftarrow \mathsf{Encrypt}(k, m)$$

$$\mathsf{Decrypt}'(k, (c_1, c_2)) = \begin{cases} \mathsf{Decrypt}(k, c_1) & c_1 = c_2 \\ \perp & \text{otherwise.} \end{cases}$$

Problem 5: Time Spent [3 extra credit points]. How long did you spend on this problem set? This is for calibration purposes, and the response you provide does not affect your score.

Optional Feedback. Please answer the following *optional* questions to help us design future problem sets. You do not need to answer these questions. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) What was your favorite problem on this problem set? Why?

(b) What was your least favorite problem on this problem set? Why?

(c) Do you have any other feedback for this problem set?

(d) Do you have any other feedback on the course so far?