CS 355: Topics in Cryptography **Spring 2022**

Problem Set 1

Due: Friday, 8 April 2022 (submit via Gradescope)

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

https://crypto.stanford.edu/cs355/22sp/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **862WDX** to sign up. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Bugs: We make mistakes! If it looks like there might be a mistake in the statement of a problem, please ask a clarifying question on Ed.

Problem 1: True/False [7 points].

(a) Which of the following are true in a world where $\mathsf{P} = \mathsf{NP}$.

i Secure PRFs exist in the standard model.

ii Secure PRFs exist in the random oracle model.

iii The one-time-pad cipher is secure.

(b) If there exists a PRG with 1-bit stretch, there exists a PRG with n^{800}-bit stretch (where n is the length of the PRG seed).

(c) Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a pseudorandom permutation. Then:

i $f_{k=0}(x) := F(0,x)$ is (always) a one way function.

ii $f_{k=0}(x) := F(0,x)$ is (always) a one way permutation.

iii $f_{x=0}(k) := F(k,0)$ is (always) a one way permutation.

Problem 2: Merkle Puzzles [15 points]. In lecture, you saw Merkle's key-exchange protocol. That protocol uses a hash function $H\colon \mathbb{Z}_{\lambda^2} \to \{0,1\}^{\log^2 \lambda}$, where $\lambda \in \mathbb{Z}^+$ is the security parameter. We model H as a random oracle.

Merkle's key-exchange protocol works as follows:

1. Alice picks integers $a_1, \ldots, a_\lambda \xleftarrow{R} \mathbb{Z}_{\lambda^2}$ and publishes $H(a_1), \ldots, H(a_\lambda)$.
2. Bob picks integers $b_1, \ldots, b_\lambda \xleftarrow{R} \mathbb{Z}_{\lambda^2}$ and publishes $H(b_1), \ldots, H(b_\lambda)$.
3. Alice and Bob find the least $i, j \in \{1, \ldots, \lambda\}$ such that $H(a_i) = H(b_j)$. If no such pair exists, output "fail."
4. Alice outputs a_i as her shared secret with Bob. Bob outputs b_j as his shared secret with Alice.

While Alice and Bob each make λ queries to H, we argued in class that any successful eavesdropping attacker must make $\Omega(\lambda^2)$ queries to H.

(a) There is some chance that Alice and Bob agree on indices i and j in Step 3 such that $H(a_i) = H(b_j)$ but $a_i \neq b_j$. First, explain why this is problematic in the context of a key-agreement protocol. Next, show that the probability of this bad event is negligible.

(b) Prove that Alice and Bob successfully agree on a shared secret with probability at least $1/100$. Make sure that your argument accounts for the failure event from part (a).
[**Hint:** You may use the "Birthday Bound" in Appendix B.1 of the Boneh-Shoup book.]

(c) Show that it is possible to reduce the failure probability to some quantity *negligible* in λ while still keeping the total communication between Alice and Bob $\widetilde{O}(\lambda)$. (Recall that $\widetilde{O}(\lambda)$ is notation for $\lambda \cdot \text{polylog}(\lambda)$.)

(d) As described, Merkle's scheme requires total communication $2\lambda \log^2 \lambda$ bits. Describe how to reduce the total communication to $\lambda \log^2 \lambda + o(\lambda)$ bits, without changing the protocol's failure probability or security properties.

(e) **Research problem [+100 points].** If we take $\lambda \approx 2^{30}$ to get security against attackers running in time $\lambda^2 \approx 2^{60}$, then Merkle's protocol requires a huge amount of communication—around ten *gigabytes*. Show that it is possible to reduce the communication of Merkle's scheme to $\widetilde{O}(1)$ while: (a) Alice and Bob still run in time $\widetilde{O}(\lambda)$ and (b) the protocol maintains security against attackers running in time $o(\lambda^2)$. Even constructing a scheme with communication complexity $\widetilde{O}(\lambda^\epsilon)$ for any $\epsilon < 1$ would be very interesting. Your solution must somehow skirt the known impossibility results.

(f) **Research problem [+1000 points].** Construct a key-agreement protocol from a one-way function (e.g., AES) in which Alice and Bob run in time λ and the best attack runs in time *super-polynomial* in λ. We have no idea how to construct such a protocol, but we also have no way to rule out the existence of such a protocol either. A famous result of Impagliazzo and Rudich implies that any such protocol would likely not make "black-box" use of the one-way function.

Problem 3: Key Leakage in PRFs [5 points]. Let F be a secure PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{K} = \mathcal{X} = \mathcal{Y} = \{0,1\}^n$. Let $\mathcal{K}_1 = \{0,1\}^{n+1}$. Construct a new PRF F_1, defined over $(\mathcal{K}_1, \mathcal{X}, \mathcal{Y})$, with the following property: the PRF F_1 is secure; however, if the adversary learns the last bit of the key then the PRF is no longer secure. This shows that leaking even a *single* bit of the secret key can completely destroy the PRF security property.
[**Hint:** Try changing the value of F at a single point.]

Problem 4: P and NP [5 points]. Show that the existence of any one-way function implies $\mathsf{P} \neq \mathsf{NP}$.
Recall: if $\mathsf{P} = \mathsf{NP}$*, then for any an deterministic, efficient algorithm* $\mathcal{A}(x, w) \to \{0,1\}$ *there is a deterministic, efficient algorithm* $\mathcal{B}(x) \to \{0,1\}$ *that computes whether* $\exists w.\ \mathcal{A}(x, w) = 1$

Problem 5: Vector Commitments [5 points]. In this problem we consider *vector commitments*: commitments to vectors which can be opened to one index at a time.

The classic construction vector commitment scheme is the *Merkle tree*, which is parameterized by a hash function $H : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$. The core of this construction is a hash tree, as shown in Figure 1. Each internal node in the tree represents an evaluation of the hash function over the child nodes. The Commit procedure evaluates the hash tree, returning the root at the commitment. The IdxOpen procedure

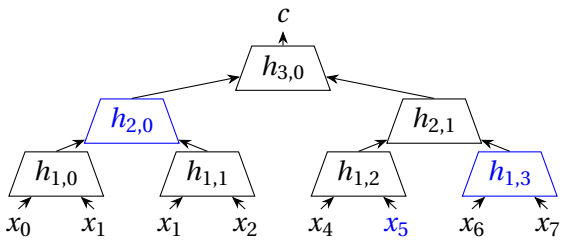

Figure 1: A Merkle tree

produces the siblings along a path (e.g., the blue nodes, for x_4), and the IdxVerify checks the hash evaluations along the path.

More precisely, the Merkle tree vector commitment scheme is defined by three algorithms:

- $\mathsf{Commit}(d \in \mathbb{N}, x \in \mathcal{X}^{2^d}) \to \mathcal{X}$:

  for $i \in \{0,1,\ldots,2^d-1\}$: $h_{0,i} \leftarrow x_i$

  for $\ell \in \{1,2,\ldots,d\}$, for $i \in \{0,1,\ldots,2^{d-\ell}-1\}$: $h_{\ell,i} \leftarrow H(h_{\ell-1,2i}, h_{\ell-1,2i+1})$

  return $h_{d,0}$

- $\mathsf{IdxOpen}(d \in \mathbb{N}, i \in \mathbb{N}, x \in \mathcal{X}^{2^d}) \to \mathcal{X}^d$:

  compute $h_{\ell,i}$ as in Commit

  for $\ell \in \{0,\ldots,d-1\}$: $p_i \leftarrow h_{\ell,((i \gg \ell) \oplus 1)}$ [1]

  return $(p_0,\ldots,p_{d-1})$

- $\mathsf{IdxVerify}(d \in \mathbb{N}, i \in \mathbb{N}, x, \in \mathcal{X}, c \in \mathcal{X}, p \in \mathcal{X}^d) \to \{0,1\}$:

  $h_0 \leftarrow x$

  for $\ell \in \{1,\ldots,d\}$:

  $j \leftarrow i \gg (\ell-1)$

  if j is odd: $h_\ell \leftarrow H(p_{\ell-1}, h_{\ell-1})$

  else: $h_\ell \leftarrow H(h_{\ell-1}, p_{\ell-1})$

  return $h_d \stackrel{?}{=} c$

A vector commitment scheme is *index binding* if for all $d \in \mathbb{N}$ and for all efficient adversaries $\mathcal{A}$,

$$\Pr\left\{\mathsf{IdxVerify}(d,i,x,c,p) = 1 \wedge \mathsf{IdxVerify}(d,i,x',c,p') = 1 \wedge x \neq x' : (c,i,x,p,x',p') \leftarrow \mathcal{A}(d,\lambda)\right\} = \mathrm{negl}(\lambda)$$

where λ is the security parameter of H. A hash function H is *collision resistant* if for all efficient adversaries $\mathcal{A}$,

$$\Pr\left\{H(x,y) = H(x',y') \wedge (x,y) \neq (x',y') : (x,y,x',y') \leftarrow \mathcal{A}(\lambda)\right\} = \mathrm{negl}(\lambda)$$

(a) Please prove that Merkle tree vector commitments are index binding if H is collision-resistant.

[1] Here, $\gg$ is logical right shift and $\oplus$ is bitwise XOR, as in C. That is, $x \gg y$ denotes $\lfloor x/2^y \rfloor$ and $x \oplus y$ denotes the integer with binary representation equal to the bitwise XOR of the binary representations of x and y.

(b) **Extra Credit [2 points].** Is a Merkle tree commitment hiding in the random oracle model? If so, prove it. If not, briefly explain why, modify the scheme to make it hiding in the random oracle model, and prove that the modification is hiding in the random oracle model. *Note: A few different "hiding" properties can be formalized for vector commitments. You should consider whether the commitment itself is hiding—you need not consider opening proofs.*

Problem 6: Our Favorite PRF [10 points]. In class we saw the PRF $F(k,x) = H(x)^k$ and were told that it has many useful properties. In this problem, we will explore two applications of this PRF.

(a) Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF defined over groups $(\mathcal{K}, +)$ and $(\mathcal{Y}, \otimes)$, where $+$ and $\otimes$ are the respective group operations in those groups. We say F is *key-homomorphic* if it holds that

$$F(k_1 + k_2, x) = F(k_1, x) \otimes F(k_2, x).$$

Is the PRF $F(k,x) = H(x)^k$ defined with a random oracle $H : \mathcal{X} \to G$ (where G is a group of prime order p) a key-homomorphic PRF? Please prove your answer one way or the other.

(b) *Key rotation* is a common problem encountered in cloud storage: how to change the key under which data is encrypted without sending the keys to the storage provider? A naive solution is to download the encrypted data, decrypt it, re-encrypt it under a new key, and re-upload the new ciphertext. We will now see how this process can be made more efficient with a key-homomorphic PRF.

Suppose you have a ciphertext c made up of blocks $c_1, ..., c_N$ that corresponds to a message $m = (m_1, ..., m_N)$ encrypted under a key k_1 using a key-homomorphic PRF F in counter mode, i.e., $c_i = m_i \otimes F(k_1, i)$. Now you want to rotate to a key k_2. It turns out you can send the storage provider a single element $k_{\mathsf{update}} \in \mathcal{K}$ which it can then use to generate c', an encryption of m under k_2. Please tell us how you can compute k_{update} and how the storage provider can use k_{update} and c to compute c'.

(c) An *oblivious PRF* is an interactive protocol between a client who holds a message x and a server who holds a key k. The protocol allows the client to learn the PRF evaluation $F(k,x)$ without the server learning anything about x. Oblivious PRFs are used in many advanced crypto protocols.

It turns out that there is an oblivious PRF protocol for the PRF $F(k,x) = H(x)^k$. Please show us how a client holding x and a server holding k can interact so that the client learns $H(x)^k$ while the server learns nothing about x. You don't need to prove security, we would just like to see the protocol.