Massachusetts Institute of Technology
6.5610: Applied Cryptography and Security — February 6, 2024
Professors Henry Corrigan-Gibbs and Yael Tauman Kalai — **Due:** February 16, 2024 at 4:59pm

Problem Set 1

Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate page.*

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza. See the course website for our policy on collaboration. Each group member must independently write up and submit their own solutions.

Homework must be typeset in LaTeX*and submitted electronically!* Each problem answer must be provided as a separate page. Mark the top of each page with your group member names, the course number (6.5610), the problem set number and question, and the date. We have provided a template for LaTeX on the course website (see the *Psets* tab at the top of the page).

Problem 1-1. One-way functions and collision resistance

A function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* (OWF) if it is computable in polynomial-time and for any polynomial-time adversary A there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[f(x) = f(x') \colon \begin{array}{c} x \xleftarrow{R} \{0,1\}^\lambda \\ x' \leftarrow A(f(x)) \end{array}\right] \leq \mu(\lambda).$$

In other words, given $f(x)$ it is difficult to find x' such that $f(x') = f(x)$.

A family of functions $\{f_\lambda\}_{\lambda\in\mathbb{N}}$ is said to be *collision resistant* if it is polynomial-time computable, for every $\lambda \in \mathbb{N}$, $f_\lambda : \{0,1\}^* \to \{0,1\}^\lambda$, and for all polynomial-time adversaries A there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[f_\lambda(x) = f_\lambda(x') \wedge x \neq x' \ : \ (x, x') \leftarrow A(1^\lambda)\right] \leq \mu(\lambda).$$

In other words, it is difficult to find distinct x, x' such that $f_\lambda(x) = f_\lambda(x')$.

For each of the following functions g determine if g is necessarily a one-way function (OWF). If so, explain in a few sentences why it is a OWF, and if not, provide an attack.

For simplicity, in what follows we define f and g for a given input length, and omit the subscript λ from the collision resistant hash functions.

- **(a)** Let $f : \{0,1\}^n \to \{0,1\}^\lambda$ be a OWF, and let $g : \{0,1\}^n \to \{0,1\}^{2\lambda}$ where $g(x) = f(x)||0^\lambda$.
- **(b)** Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a OWF, and let $g : \{0,1\}^{\lambda/2} \to \{0,1\}^\lambda$ where $g(x) = f(x||0^{\lambda/2})$ and we assume for simplicity that λ is even.
- **(c)** Let $g : \{0,1\}^* \to \{0,1\}^\lambda$ be collision resistant. Is g necessarily a OWF?

For each of the following functions g determine if g is necessarily a collision resistant function. If so, explain in a few sentences why it is collision resistant, and if not, provide an attack.

- **(d)** Let $f : \{0,1\}^n \to \{0,1\}^\lambda$ be collision resistant, and let $g : \{0,1\}^{2n} \to \{0,1\}^\lambda$ where $g(x) = f(x_1||x_3||x_5||...||x_{2n-1})$.
- **(e)** Let $f : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ be collision resistant, and let $g : \{0,1\}^{4\lambda} \to \{0,1\}^\lambda$ where $g(x_1||x_2) = f(f(x_1)||f(x_2))$.
- **(f)** Let $g : \{0,1\}^* \to \{0,1\}^\lambda$ be a OWF. Is g necessarily collision resistant?

Problem 1-2. Short integer solutions

We will need the following definition:

> **Definition** (Short integer solutions (SIS))**.** The short-integer-solutions problem is parameterized by positive integers n, m, q, and B. For a random matrix $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$, the problem is to find a nonzero vector $\mathbf{e} \in \mathbb{Z}^m$ such that:
>
> (1) $\mathbf{A} \cdot \mathbf{e} = \mathbf{0} \in \mathbb{Z}_q^n$ and
>
> (2) $\|\mathbf{e}\|_\infty \leq B$, where $\|\cdot\|_\infty$ denotes the L_∞-norm where $\|\mathbf{x}\|_\infty = \max |x_i|$

To be fully formal, we can treat n as the security parameter and then let m, q, and B be functions of n. (Often, we will take all of these parameters to be polynomials in n.) An example parameter setting might be $n = 1024$, $q = 2^{32}$, $m = 4n \log q$, and $B = 1$.

Then we have:

Definition (SIS assumption)**.** The SIS assumption on parameters (n, m, q, B) is that for all p.p.t. adversaries $\mathcal{A}$, there is a negligible function $\mu(\cdot)$ such that

$$\Pr\left[\mathbf{A} \cdot \mathbf{e} = \mathbf{0} \wedge \|\mathbf{e}\|_\infty \leq B \ : \ \begin{array}{c} \mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m} \\ \mathbf{e} \leftarrow \mathcal{A}(\mathbf{A}) \end{array}\right] \leq \mu(n).$$

(a) For the SIS assumption to hold, the parameters (n, m, q, B) need to satisfy certain conditions. We will list a few insecure settings of the SIS parameters. For each, explain why we do not instantiate the SIS problem with this parameter setting:

(a) $(n, 2n, 2, 2)$

(b) $(n, n/1000, q, 1)$, for a prime $q \approx n$ (*Hint:* Count the number of possible inputs and outputs.)

(c) $(n, 10n \log n, q, 2)$ for a prime $q \approx n$, except rather than sampling $\mathbf{A}$ at random from $\mathbb{Z}_q^{n\times m}$, we sample a random matrix $\mathbf{A}$ that has at most one non-zero element in each row and at most one non-zero element in each column.

(b) For SIS parameters (m, n, q, B) and a random matrix $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$, let $H_\mathbf{A} \colon \{0,1\}^m \to \mathbb{Z}_q^n$ be a hash function, defined as $H_\mathbf{A}(\mathbf{e}) := \mathbf{A} \cdot \mathbf{e}$. Explain why $H_\mathbf{A}$ is collision resistant under the SIS assumption with parameters $(m, n, q, 2B)$. In other words, given an efficient algorithm $\mathcal{A}$ that finds a collision in $H_\mathbf{A}$, produce an efficient algorithm that breaks the SIS assumption with the given parameters.

(c) State two reasons (in at most one sentence each) why we might not use $H_\mathbf{A}$ as a collision-resistant hash function in practice. Again, think of the parameters necessary to achieve 128-bit security as being something like $n = 2^{10}$, $q = 2^{32}$, $m = 2^{17}$, and $B = 1$.

Problem 1-3. Finite fields and polynomials In this problem we consider polynomials over finite fields and finite rings. Specifically, let $\mathbf{Z}_n$ be the ring of elements $\{0, 1, \ldots, n-1\}$ where addition and multiplication are done modulo n. When n is a prime this is a field (where every non-zero element has a multiplicative inverse) and when n is not a prime it is a ring (where not all elements have a multiplicative inverse). In this problem we consider degree d polynomials $f : \mathbf{Z}_n \to \mathbf{Z}_n$. Such polynomials can be represented as $f(x) = \sum_{i=0}^{d} a_i x^i$ where $a_0, \ldots, a_d \in \mathbf{Z}_n$ and where addition and multiplication are done modulo n.

(a) Argue that a non-zero degree-d polynomial modulo a prime p has $\leq d$ roots. (Hint: There are multiple ways to do this. One way is by induction. Another way uses the fact that the Vandermonde matrix over a field has full rank.)

(b) Argue that for every prime p, every distinct $x_1, \ldots, x_{d+1} \in \mathbf{Z}_p$ and every $y_1, \ldots, y_{d+1} \in \mathbf{Z}_p$, there exists a unique degree-d polynomial $f : \mathbf{Z}_p \to \mathbf{Z}_p$ such that $f(x_i) = y_i$ for every $i \in \{1, \ldots, d+1\}$. (Hint: You can assume that the Vanderonde matrix has full rank.)

(c) Give an example of a (non-prime) n and a degree-d polynomial $f : \mathbf{Z}_n \to \mathbf{Z}_n$ that has more than d roots.

Problem 1-4. Breaking AES without S-box

On piazza, you can find a zip file `pset1.zip` that contains the file needed for this problem.

`gen.py` contains an AES encryption implementation, except the substitution operations are omitted. Using this wrong implementation, it encrypts the secret message (`secret.txt`) and 150 random blocks. Your goal is to recover the secret.

- `secret.txt` is the secret you want to recover and is not contained in the zip file.
- `gen.py` reads from `secret.txt` and generates `ciphertext.txt` and `data.txt`. You don't have to read the details of AES encryption as it's meant to be a correct implementation (except for the substitution); it should be enough to check the code in the main function.
- `ciphertext.txt` is the encrypted secret. It contains non-ASCII characters, so don't be surprised if it looks garbled.
- `data.txt` has 150 lines, each containing a block and its encryption in hex.
- `hint.pdf` provides hints and guidance, but please try it without checking the hints first!

Please submit the secret message and the code on Gradescope.