Modern Cryptology | Spring 2023

Problem Set 2

Ron Rothblum | Due: 18/5/2023

Reminder: please type your answers (LaTeX is encouraged but not mandatory).

1 PRGs

In each of the following sections, assume that $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a PRG ($n < \ell(n)$). You need to prove or disprove that G' is a PRG. When disproving, it suffices to construct some PRG G such that G' is not a PRG (and you may use the assumption that some PRG G_0 exists to construct G).
Notation: $x \circ y$ is the concatenation of the strings x, y.

1. Denote $[\ell] = \{1, ..., \ell\}$. Let $\sigma : [\ell] \to [\ell]$ be a fixed permutation. Define $G'(s) = G_{\sigma(1)}(s) \dots G_{\sigma(\ell)}(s)$, where $G_i(s)$ is the i^{th} bit of $G(s)$.
2. $G'(s_L \circ s_R) = G(s_R) \circ G(s_L)$, where $|s_L| = |s_R|$.
3. $G'(s) = G(s) \circ s_1$, where s_1 is the first bit of s.

2 PRG implies OWF

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG. Prove that G is a one-way function.

Remark 2.1. *For simplicity, you may assume that G is a $1:1$ function. That is, for any distinct $x, x' \in \{0,1\}^n$ it holds that $G(x) \neq G(x')$.*

3 Bad Primes for Discrete Log

The discrete log problem is believed to be hard in $\mathbb{Z}_p^*$, for prime p. In this question however, we show that we should be careful when choosing the prime p. Specifically, we show that discrete log is easy for $p = 2^k + 1$ for any positive integer k. Let g be a generator of $\mathbb{Z}_p^*$.

1. Show how to efficiently compute the least significant bit of any $x \in \mathbb{Z}_{p-1}$ given $y = g^x$. That is, devise an algorithm that returns 0 if x is even and returns 1 if x is odd.

   Hint 3.1. *Consider the value $y^{\frac{p-1}{2}}$.*

2. Assuming x is even, show how to efficiently compute the second least significant bit of x.
3. Generalize the ideas from the previous two sections and devise an algorithm that computes the discrete log of any $y \in \mathbb{Z}_p^*$.
4. **Food for thought (not for submission):** why would you algorithm fail for general primes?

4 Information Theoretic MAC

In this question we will show how to construct a MAC that is secure even against unbounded adversaries, and without relying on any unproven assumption. While we allow the adversary unbounded computational power, we will restrict it so that it can see at most *one* authenticated message (and based on that needs to forge a new tag).

4.1 Naive Adversary

Suppose the length of the tag is t bits. Show an adversary that successfully forges a tag (of any message of its choice) with probability at least 2^{-t}.

4.2 Pairwise Independent Hashing

Loosely speaking, a collection of functions is called pairwise independent, if the behaviour of a *random* function from the family on any two distinct points in the domain, is like that of a totally random function.

Formally, a collection of hash function $H = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is called *pairwise independent* if for every distinct $x_1, x_2 \in \{0,1\}^n$ and every $y_1, y_2 \in \{0,1\}^m$ it holds that:

$$\Pr_{h \leftarrow H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 2^{-2m}.$$

Remark 4.1. *Make sure you understand why we chose to put 2^{-2m} on the RHS.*

Let $\mathbb{F}$ be the finite field of size 2^n.[1] Consider the collection of functions $\{h_{a,b} : \mathbb{F} \to \mathbb{F}\}_{a,b \in \mathbb{F}}$, where $h_{a,b}(x) = a \cdot x + b$. Show that this collection is pairwise independent. How can you extend it to handle the case that the domain and range are not the same?

4.3 One-Time MAC

Use the pairwise independent hash functions of Section 4.2 to construct a MAC, with tag length t, in which an adversary that sees at most one tagged message can forge a tag with probability at most 2^{-t}.

5 Not for submission

Let $n_1, \dots, n_k$ be co-prime and x be an integer such that $\forall i \in \{1, \dots, k\} : n_i | x$. Prove that $\prod_{i=1}^{k} n_i | x$.

[1] For those who don't remember all the details about finite fields - don't worry, we basically just need to use the fact that there is such a field. A more advanced comment - you can assume that we have a nice representation of the field (via an irreducible polynomial) and that field operations can be performed in time poly(m).