0368.3049 **Fall 2013**

Introduction to Modern Cryptography

Benny Chor and Nir Bitansky

Assignment 1

Published October 31, 2013. Due November 14, in mailbox 372 (Schreiber building). Submission in pairs is encouraged (submission in threes or more is not allowed). A 5-point bonus will be given to typed (as opposed to handwritten) submissions.

Problem 1: Substitution ciphers are easy to break. Decrypt the substitution cipher given in: `http://tau-crypto-f13.wdfiles.com/local--files/home-assignments/cipher.txt`. Explain the techniques you use, and describe the steps of your solution. (The original text is in English; punctuation marks and whitespace have been removed.) Language statistics may come in handy; can use for example `http://www.simonsingh.net/The_Black_Chamber/substitutioncrackingtool.html`.

Problem 2: Perfect ciphers. Let p be a prime. Consider the following encryption scheme. The secret key is a pair (a, b) sampled uniformly at random from $\mathbb{Z}_p^* \times \mathbb{Z}_p$. An encryption of a message $M \in \mathbb{Z}_p$ is defined as:

$$E_{a,b}(M) = a \cdot M + b \bmod p \ .$$

(a) **Warmup:** Show that for any $b \in \mathbb{Z}_p$, if u is distributed uniformly in $\mathbb{Z}_p$, then $b + u$ is also distributed uniformly in $\mathbb{Z}_p$. Show that for any $a \in \mathbb{Z}_p^*$, if u is distributed uniformly in $\mathbb{Z}_p$, then $a \cdot u$ is also distributed uniformly in $\mathbb{Z}_p$.

(b) Recall that, in a *perfect cipher*, the encryptions of any two messages M and M' have the same distribution, over a random choice of secret key. Show that E is a perfect cipher.

(c) Is it true that the encryptions of any two pairs of messages (M_1, M_2) and (M_1', M_2') have the same distribution, over a random choice of a secret key, where the same secret key is used to encrypt both M_1, M_2 (or M_1', M_2'). (Prove your answer.)

(d) What if we additionally assume $M_1 \neq M_2$ and $M_1' \neq M_2'$? (Prove your answer.)

Problem 3: Indistinguishability. Let A be an algorithm with a single output bit. Recall that two distributions (D_0, D_1) on $\{0,1\}^n$ are ϵ-indistinguishable for A, if

$$\left| \Pr_{d \leftarrow D_0}[A(d) = 1] - \Pr_{d \leftarrow D_1}[A(d) = 1] \right| \leq \epsilon \ .$$

Also recall that a sequence $\{(D_0(n), D_1(n))\}_{n \in \mathbb{N}}$ is ϵ-computationally-indistinguishable (respectively, ϵ-statistically-indistinguishable) if for any large enough $n \in \mathbb{N}$, $(D_0(n), D_1(n))$ are ϵ-indistinguishable for *any* poly-time (respectively, unbounded-time) A, and this is denoted by $D_0 \approx_{c,\epsilon} D_1$ (respectively, $D_0 \approx_{s,\epsilon} D_1$).

(a) Show that (D_0, D_1) are ϵ-indistinguishable for A iff

$$\left| \Pr_{\substack{b \leftarrow U_1 \\ d \leftarrow D_b}}[A(d) = b] - \frac{1}{2} \right| \leq \frac{\epsilon}{2} \ ,$$

where U_1 denotes the uniform distribution on $\{0,1\}$.

(b) Let $\{(D_0(n), D_1(n))\}_{n\in\mathbb{N}}$ be such that

$$\max_{d^*\in\{0,1\}^n} \left| \Pr_{d\leftarrow D_0}[d = d^*] - \Pr_{d\leftarrow D_1}[d = d^*] \right| \leq 2^{-n} \ .$$

Does this imply that $D_0 \approx_{c,\frac{1}{2^n}} D_1$? Does it imply that $D_0 \approx_{c,\frac{1}{10}} D_1$?

(c) Let $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{n+s}$ be an efficiently computable function. Prove that there exists an unbounded A that distinguishes the distribution $\mathsf{PRG}(U_n)$, given by the output of PRG from the uniform distribution U_{n+s} with probability at least $\epsilon := 1 - 2^{-s}$.

Problem 4: PRGs imply OWFs. In this problem, we will show that any pseudo-random generator $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{n+s}$ is a one-way function. Let A be a good inverter; specifically,

$$\Pr_{v\leftarrow\mathsf{PRG}(U_n)} \left[\begin{array}{l} A(v) = u' \\ \mathsf{PRG}(u') = v \end{array} \right] \geq \delta \ .$$

(a) Show how to construct a distinguisher A' that *efficiently* uses A to distinguish $\mathsf{PRG}(U_n)$ from U_{n+s} with probability at least $\epsilon := \delta - 2^{-s}$. How large does s have to be compared to δ for the above to be meaningful?

(b) Show how to construct a distinguisher A' that *efficiently* uses A to distinguish $\mathsf{PRG}(U_n)$ from U_{n+s} with probability at least $\epsilon := \delta(1 - 2^{-s})$.

Problem 5: Multiplicative generators in $\mathbb{Z}_m^*$. In this problem, we will make a mild usage of programming to explore the existence and abundance of multiplicative generators in three groups $\mathbb{Z}_m^*$.

Let $(G, \cdot)$ be a finite group with k elements, and denote by 1 its unit element. Recall that G is called *cyclic* if there is an element $g \in G$ such that $\langle g\rangle = G$, namely $\{g, g^2, \ldots, g^{k-1}, g^k\} = G$. Such g is called a *multiplicative generator* of G. Testing if a given g is a multiplicative generator using the equality above is feasible for small groups, but infeasible for large ones.

Suppose we know the factorization of k: $k = p_1^{e_1} p_2^{e_2} \ldots p_\ell^{e_\ell}, e_i \geq 1$ (in particular, k has ℓ distinct prime factors). It is known that in such case, g is a generator iff $g^{k/p_1} \neq 1, g^{k/p_2} \neq 1, \ldots g^{k/p_\ell} \neq 1$ (if you haven't already tried to prove it, try now using Lagrange theorem).

(a) Write a *short and readable* code in your favorite programming language (even Scheme, if you insist) that tries all elements in G and determines if each of them is a multiplicative generator or not. You should state if G is cyclic or not. If G is cyclic, your code should output the list of all multiplicative generators in G. Otherwise, the code should output the list of all elements in G having maximum order. Submit the code and the requested output lists.

Recall that the elements in the group $(\mathbb{Z}_m^*, \cdot \bmod m)$ are all integers in the set $\{1, \ldots, m-1\}$ that are relatively prime to m. There are $\phi(m)$ many elements in $\mathbb{Z}_m^*$. Run your code for $m = 35, 37, 38$.

Hint: if you use Python, `pow(a,b,m)` computes $a^b \pmod m$.

(b) It is known that if m is a prime, $\mathbb{Z}_m^*$ has a multiplicative generator. In fact, such groups have many multiplicative elements. For $m = 2^{61} - 1$ (which is a prime), it is not feasible

to try all $g \in G$. Instead, write a code that samples N elements $g \in G$ *uniformly at random*, and for each of them, tests if it is a multiplicative generator. You may choose N to be as large as you'd like, but at least as large as $100,000$. Count the number of multiplicative generators, A, and output A, N, and the first 10 multiplicative generators your code finds.

Use A and N to estimate the number of multiplicative generators in $\mathbb{Z}^*_{2^{61}-1}$. Compare your estimate to $\phi(\phi(2^{61}-1))$, which is the exact number of such generators (the function $\phi(m)$ was defined in class). How good would you say your estimate is. Submit the code and the requested outputs.

If you use Python, call `import random` (once). Then `g=random.randint(1,2**61-2)` is a pseudo random generator that produces a new `g` in `G` each time it is invoked. Serious distinguishers will probably tell it apart from a truly random sequence, but it is just fine for our needs.

Oops: To solve the problem, the prime factorization of $m-1$ is required. Well,

$$2^{61} - 2 = 2 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 41 \cdot 61 \cdot 151 \cdot 331 \cdot 1321 \ .$$

Problem 6: 10-Point bonus. A student in class proposed the following variant of Naor's bit commitment:

- The receiver Bob sends two random strings $w_0, w_1 \leftarrow \{0,1\}^n$ to the sender Alice.
- To commit to a bit $b \in \{0,1\}$, Alice chooses a random $s \leftarrow \{0,1\}^n$, and sends to Bob $G(s, w_b)$, where $G : \{0,1\}^{2n} \to \{0,1\}^{10n}$ is a pseudo random generator.
- To decommit to b, Alice sends (b, s).

Show:

(a) There exists a PRG G, such that the above scheme is not binding.

(b) There exists a PRG G, such that the above scheme is not hiding.

(Can assume that for any polynomial length-functions $\ell(n) < \ell'(n)$, there exist PRGs that stretch ℓ to ℓ'.)