Computer Science and Engineering, UCSD | Fall 09
CSE 207: Modern Cryptography | **Instructor:** Mihir Bellare
Problem Set 6 Solutions | November 9, 2009

Problem Set 6 Solutions

Problem 1. [25 points] Let $G = \langle g \rangle$ be a cyclic group of order m, and let $k = \lceil \log_2(m) \rceil$. The group G as well as g, m, k are public and known quantities. Suppose you are given a (possibly randomized) algorithm B such that $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(B) \geq 1/2$. You are also given a positive integer s. Design an algorithm A that uses B as a subroutine to achieve $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) \geq 1 - 2^{-s}$. The running time T_A of A should be $sT_B + \mathcal{O}(skT_G)$ where T_B is the running time of B and T_G is the time to do a group operation.

Let $X = g^x$ be the input, so that the algorithm succeeds if it returns x. Our first thought is likely to be the algorithm A on the left of Fig. 1. $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A)$ is the probability that there is some i such that $y_i = x$. The assumption $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(B) \geq 1/2$ means that line 02 returns $y_i \neq x$ with probability at most $1/2$. So the probability that *all* y_i are different from x is at most $(1/2)^s = 2^{-s}$, whence $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) \geq 1 - 2^{-s}$. Right?

Wrong. It is true that line 02 returns $y_i \neq x$ with probability at most $1/2$. But the events $y_i \neq x$ are not independent as i ranges from 1 to s, because the probability is over the random choice of X, which is the same for all i. (Whether B is randomized or not makes no difference.) So we can't conclude that the probability that *all* y_i are different from x is at most $(1/2)^s = 2^{-s}$. For example it could be that there is a set $S \subseteq G$ of size $|S| = |G|/2$ such that $B(X) = \mathrm{DLog}_{G,g}(X)$ when $X \in S$ and $B(X) \neq \mathrm{DLog}_{G,g}(X)$ when $X \notin S$. If our input X is in S then A will succeed and else it will not, so $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A)$ is only $1/2$, not $1 - 2^{-s}$.

To get around this, we want to invoke B each time on random, independent inputs. Yet the result must tell us something about $\mathrm{DLog}_{G,g}(X)$ for our one, given X. The algorithm on the right of Fig. 1 illustrates how to do this. Let $x_i = \mathrm{DLog}_{G,g}(X_i)$ for $1 \leq i \leq s$. Then $x_i \equiv r_i + x \pmod m$ for $1 \leq i \leq s$. The assumption $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(B) \geq 1/2$ means that line 13 returns $y_i \neq x_i$ with probability at most $1/2$. But due to the random choices of $r_1, \ldots, r_s$, the points $X_1, \ldots, X_s$ are uniformly and independently distributed in G, so the events $y_i \neq x_i$ are independent as i ranges from 1 to

algorithm $A(X)$
01 for $i = 1, \ldots, s$ do
02 $\quad y_i \xleftarrow{\$} B(X)$
03 $\quad$ if $g^{y_i} = X$ then return y_i
04 return $\perp$

algorithm $A(X)$
11 for $i = 1, \ldots, s$ do
12 $\quad r_i \xleftarrow{\$} \mathbf{Z}_m$; $X_i \leftarrow g^{r_i} X$
13 $\quad y_i \xleftarrow{\$} B(X_i)$
14 $\quad$ if $g^{y_i} = X_i$ then return $(y_i - r_i) \bmod m$
15 return $\perp$

Figure 1: Incorrect and correct algorithms for Problem 1.

s. The probability that $y_i \neq x_i$ for *all* i is thus at most $(1/2)^s = 2^{-s}$. But $y_i = x_i$ means that $x \equiv x_i - r_i \equiv y_i - r_i \pmod m$, and thus $g^{y_i - r_i} = X$. So $\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) \geq 1 - 2^{-s}$.

Each of the s iterations of the "for" loop runs B once and performs an exponentiation. The former has cost T_B and the latter $\mathcal{O}(kT_G)$. Additionally, the iteration might do a subtraction modulo m, which has cost $\mathcal{O}(k)$, dominated by $\mathcal{O}(kT_G)$. The overall cost is thus $sT_B + s \cdot \mathcal{O}(kT_G)$.

Problem 2. [25 points] Let $G = \langle g \rangle$ be a cyclic group of order m. Let $k = \lceil \log_2(m) \rceil$ and let w be a positive integer dividing k. The group G as well as g, m, k, w are public and known quantities. An *exponentiation with pre-processing scheme* is a pair (P, E) of algorithms. The first takes no inputs and outputs a table T. The second takes input T and any $x \in \mathbf{Z}_m$ and outputs g^x. Design such a scheme so that T consists of at most $(k/w)2^w$ group elements and E uses at most k/w group operations.

Let $B = 2^w$ and $\ell = k/w$. For $x \in \mathbf{Z}_m$ let $\mathrm{Expand}_B(x)$ be the vector $(x_0, x_1, \ldots, x_\ell)$ of points in $\mathbf{Z}_B$ for which

$$x \;=\; \sum_{i=0}^{\ell-1} x_i B^i \;=\; x_0 + x_1 B + x_2 B^2 + \cdots + x_{\ell-1} B^{\ell-1} \,.$$

We will exponentiate via

$$g^x \;=\; g^{\sum_{i=0}^{\ell-1} x_i B^i} \;=\; \prod_{i=0}^{\ell-1} g^{x_i B^i} \,.$$

This costs $\ell - 1$ group operations if the quantities $g^{x_i B^i}$ are known. This gives us an idea for what to precompute. Let

$$T[s,i] \;=\; g^{s 2^{iw}} \;=\; g^{s B^i}$$

for all $s \in \mathbf{Z}_B$ and $i \in \mathbf{Z}_\ell$. This table of ℓB group elements will be computed by algorithm P. Note it does not depend on x. Now, algorithm E can be defined via

> algorithm $E(T, x)$
> $(x_0, x_1, \ldots, x_\ell) \leftarrow \mathrm{Expand}_B(x)$
> $Y \leftarrow \mathbf{1}$
> for $i = 0, \ldots, \ell$ do
> $Y \leftarrow Y \cdot T[x_i, i]$
> return Y

This type of exponentiation speedup via precomputation is very significant in practice. Designers trade-off the table size and speed gains by appropriate choices of w.
