ICS 268: Cryptography and Communication Security 10/14/2004

Homework 3

Due **Thursday**, *10/21/2004*

1 Shank's Discrete Logarithm Algorithm : Modification

Describe how to modify Shank's "Baby Step - Giant Step" algorithm so that to compute the discrete logarithm $DL_{g,p}(y)$ for elements y of the form $y = g^x \bmod p$ where x is known to lie in an interval $[s,t]$ s.t. $0 \leq s < t < q$ where $q = ord_p(g)$ is the order of element g in $\mathbb{Z}_p^*$. We want an algorithm which runs in time $O(\sqrt{t-s})$, or, if you want to be more exact, in time $O(\sqrt{t-s} * |p|^c)$ for some small constant c. Prove that your algorithm is correct.

2 Boosting a "Non-negligible Probability" Attack

Show that if there is a constant d s.t. some probabilistic polynomial time algorithm A computes the discrete logarithm $DL_(g,p)(y) = x$ s.t. $g^x = y \bmod p$ with a (non-negligible) probability $\epsilon_A(\tau) \geq \frac{1}{\tau^d}$ for *every* input y, then this algorithm can be used to obtain a different probabilistic polynomial time algorithm A' which computes DL for every y with probability $1 - \delta$ for any constant δ. (Hint: You can approximate $(1 - \frac{1}{a_n})^{a_n}$ as $\frac{1}{e}$ for any sequence a_n growing to infinity, where $e = 2.718....$)

If A runs in time $T_A(\tau)$ then what's the time $T_{A'}(\tau)$ of A'?

2.1 Bonus Question: Failure to Boost a "Negligible" Attack

Show that such boosting strategy fails if $\epsilon_A(\tau)$ is a (negligible) function $2^{-\tau}$ instead.

3 Boosting a DL Attack using DL Self-Reducibility

Imagine that someone creates an efficient (probabilistic polynomial time) algorithm A which computes the DL function perfectly (i.e. with probability 1) but A works only if the input y is of the form $y = g^x \bmod p$ for x whose first 20 bits are all zeroes (here "first bits" can mean either the MSBs or the LSBs, it does not matter).

What's the advantage of A in breaking the one-wayness property of the exponentiation function? Is it negligible? Does the existence of A imply that *Exp* is not a one-way function?

It seems that one could counteract such an attack by simply making sure that such x's are always avoided (in whatever application the discrete logarithm is used: encryption, signatures, authentication, etc). However, as you will show below, such modification would be useless because if computing discrete logarithm on a costant-fraction subset of the domain is easy then computing discrete logarithm *everywhere* must be easy too.

Namely, show how to use algorithm A to construct, for any constant δ, a PPT algorithm A' which breaks the discrete logarithm problem completely by computing $DL_{g,p}(y)$ for every $y \in < g >$ with probability $1 - \delta$. (If you prefer, you can give instead an algorithm which computes the discrete log *always* and runs in *expected* polynomial time.)

What's the running time of A' compared to the running time of A?

3.1 Bonus Question: Generalization to any OWF

How would you describe the property of the Exp function that enables this phenomenon, i.e. that ability to compute its inverse on some (non-negligible) fraction of the space enables computation on the whole space? Try to define the required property, state the general theorem, and prove it.