MTAT.07.003 Cryptology II
Spring 2012 / Exercise session ?? / Example Solution

Exercise (Hardness of discrete logarithm bits). *Let $\mathbb{G} = \langle g \rangle$ be a finite group of an odd order q generated by the powers of an element g. Then the Discrete Logarithm (DL) problem is following. For any element y find a power $x \in \mathbb{Z}_q$ such that $g^x = y$.*

1. *Show that if there exists an efficient procedure that can always compute the highest bit of discrete logarithm then the DL problem is easy.*
2. *Show that if there exists an efficient procedure that can always compute the lowest bit of discrete logarithm then the DL problem is easy.*
3. *Show that if there exists an efficient procedure that can always compute two consecutive bits at positions i and $i+1$ then the DL problem is easy.*

($\star$) *Show that if there exists an efficient procedure that can always compute the ith bit then the DL problem is easy. If it is not possible show the counterexample.*

Solution. By construction the discrete logarithm is $\lceil \log_2 q \rceil$ bit number, further denoted by ℓ. Before going any further, we make the following crucial observation. Let x be the discrete logarithm of y. If the highest bit $x_\ell = 0$ then squaring of y leads to a group element with discrete logarithm $2x$. If $x_\ell = 1$ then we get $2x + \kappa$ where $\kappa = 2^{\ell+1} - q$. All our reductions uses this property to recover the discrete logarithm bit by bit.

Highest bit. Let $\mathcal{A}$ is the magic algorithm that can always output the highest bit of discrete logarithm, i.e., $\mathcal{A}(y)$ outputs the ℓth bit of $\log y$ for any $y \in \mathbb{G}$. Then we can construct an adversary $\mathcal{B}$ can solve the DL problem as follows:

```
$\mathcal{B}^{\mathcal{A}}(y)$
    $z \leftarrow g^{2^\ell}$
    For $i \in \{1, \ldots, \ell\}$ do
        $\alpha_i \leftarrow \mathcal{A}(y)$
        $y \leftarrow (y \cdot z^{-\alpha_i})^2$
    return $\alpha_1\alpha_2 \ldots, \alpha_\ell$ .
```

Since $\mathcal{A}$ is always correct the update first eliminates the ℓth bit and then shifts the power one position to left. As a consequence, we restore the discrete logarithm of y bit by bit. If the running time of $\mathcal{A}$ is $t_{\mathcal{A}}$ then the running time of $\mathcal{B}$ is $\mathrm{O}(\ell \cdot t_{\mathcal{A}} + \ell)$.

Lowest bit. If $\mathcal{A}$ can reconstruct the lowest bit then we can use squaring to detect the highest bit. Indeed, if the highest bit is zero then the lowers bit after squaring is also zero. If the highest bit is one then the lowers bit is one after squaring as $\kappa \equiv 1 \pmod 2$. This leads to the following reconstruction algorithm:

```
$\mathcal{B}^{\mathcal{A}}(y)$
    $\kappa \leftarrow 2^{\ell+1} - q$
    For $i \in \{1, \ldots, \ell\}$ do
        $y \leftarrow y^2$
        $\alpha_i \leftarrow \mathcal{A}(y)$
        $y \leftarrow yg^{-\kappa\alpha_i}$
    return $\alpha_1\alpha_2 \ldots, \alpha_\ell$ .
```

Two consecutive bits.