# Introduction to Cryptography
# Solution to Maman 15

Yehuda Lindell

November 11, 2007

**Solution 1:** Let the public-key for El-Gamal be $(p, \alpha, \beta)$ where $\beta = \alpha^a$. Then, an encryption of $x$ is a pair $(y_1, y_2)$ where $y_1 = \alpha^k$ and $y_2 = \beta^k \cdot x$. (All computations here are of course modulo $p$.) Now, given $(y_1, y_2)$ it holds that $(y_1, 2y_2)$ is an encryption of $2x$. This is because $2y_2 = 2\beta^k \cdot x = \beta^k \cdot 2x$. Thus, $y_2/y_1^a = 2x$ as required.

**Solution 2:** Diffie-Hellman is only secure for an eavesdropping adversary. If there is an active adversary $\mathcal{A}$ who can carry out a man-in-the-middle attack, then it can do the following. When Alice sends $\alpha = g^a$ it intercepts this, chooses its own $a'$ and sends Bob the value $\alpha' = g^{a'}$. (As usual, all computations are modulo $p$.) When Bob receives $\alpha'$ he chooses a value $b$ and sends $\beta = g^b$ to Alice. However, once again, the adversary $\mathcal{A}$ intercepts $\beta$, chooses its own $b'$, computes $\beta' = g^{b'}$ and sends $\beta'$ to Alice.

Now, Alice computes $K_A = \beta'^a = g^{ab'}$ and Bob computes $K_B = g^{a'b}$. However, the adversary $\mathcal{A}$ can compute $\alpha^{b'} = g^{ab'} = K_A$ *and* $\beta^{a'} = g^{a'b} = K_B$. Therefore, the adversary knows both keys that Alice and Bob generate. If Alice sends $E_{K_A}(m)$ for Bob, then $\mathcal{A}$ can decrypt it (learning $m$) and if it wishes, it can re-encrypt it under $K_B$ so that Alice and Bob will not even know that anything happened.

**Solution 3:** Let $y = x^\alpha \bmod p$. Then, given $\alpha$, $y$ and $p$, the first thing to do is to compute the inverse of $\alpha \bmod p-1$. We know that such an inverse exists because $\alpha \in \mathsf{Z}^*_{p-1}$ (see page 163 of the course book at the bottom). Let $\beta$ denote this inverse of $\alpha$. Then, just like in the RSA cryptosystem, it holds that

$$(x^\alpha)^\beta \bmod p = x^{\alpha \cdot \beta \bmod \phi(p)} \bmod p = x^{\alpha \cdot \beta \bmod p-1} \bmod p = x^1 \bmod p = x$$

Since finding the inverse of $\alpha$ can be done efficiently, and raising $y$ to the power of $\beta$ can be done efficiently, we have that this problem is not at all hard.

**Solution 4:**

1. Let OracleDDH be an oracle that solves the DDH problem. Let $(\alpha, \beta)$ be the public-key. Given an El-Gamal ciphertext $y = (y_1, y_2)$, we wish to know if $y$ is an encryption of $x_1$ or $x_2$. In order to do this, we give OracleDDH the tuple $(\beta, y_1, y_2/x_1)$. (We assume that $\alpha$ and $G$ are known and fixed to the oracle.) If the oracle returns YES (meaning that this is Diffie-Hellman tuple), then we return $x_1$. Otherwise we return $x_2$.

   In order to see that this is correct, notice that $y_1 = \alpha^k$ and $y_2 = \beta^k \cdot x_i$ for $i = 1$ or $i = 2$, where $\beta = \alpha^a$ for some value $a$. Thus, when $i = 1$ (and so $y$ is an encryption of $x_1$, we have

given OracleDDH a tuple of the form $(\alpha^a, \alpha^k, \alpha^{ak})$ which is a Diffie-Hellman tuple. We therefore reply with $x_1$ which is correct. In contrast, when $i = 2$ (and so $y$ is an encryption of $x_2$, we have given OracleDDH a tuple of the form $(\alpha^a, \alpha^k, \alpha^{ak} \cdot x_1/x_2)$ which is *not* a Diffie-Hellman tuple (the last value equals $\alpha^b$ for some $b$ that is not equal to $ak$). We therefore reply with $x_2$ which is also correct. We conclude that given OracleDDH we can distinguish an encryption of $x_1$ with an encryption of $x_2$.

2. We now show that we can solve the Decisional Diffie-Hellman problem using OracleDistinguish that distinguishes encryptions of $x_1$ from encryptions of $x_2$. Given a generator $\alpha$ and an input $(\beta, \gamma, \delta)$ for the Decisional Diffie-Hellman problem, we define the public-key to be $pk = (\alpha, \beta)$. Next, we compute a ciphertext $y = (\gamma, \delta \cdot x_1)$ and hand OracleDistinguish the public-key $pk$ and the ciphertext $y$. If OracleDistinguish replies $x_1$ we answer yes; if OracleDistinguish replies $x_2$ or that the ciphertext is not an encryption of $x_1$ or $x_2$, we answer no.

   In order to see that this is correct, notice that if the input $(\beta, \gamma, \delta)$ is a Diffie-Hellman tuple, it holds that OracleDistinguish receives a correct encryption of $x_1$. Therefore, the oracle returns $x_1$ (because it correctly distinguishes), and we answer yes. In contrast, if the input is not a Diffie-Hellman tuple, we have that the ciphertext is not a valid encryption of $x_1$. Thus it is either an encryption of $x_2$ or not a valid encryption of either value. Thus, we answer no as required.