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

Exercise (IND-FPA $\Rightarrow$ IND-CPA). *In a fixed plaintext attack (FPA), an adversary has to fix the queried messages ahead of other interactions. Consequently, it might be possible to achieve a security goal against fixed plaintext attacks (CPA) that is infeasible against chosen ciphertext attacks. This separation manifests already if we consider indistinguishability as a security goal. Recall that a cryptosystem is (t, ε)-IND-FPA if for all t-time adversaries*

$$\mathsf{Adv}^{\mathsf{ind\text{-}fpa}}(\mathcal{A}) = |\Pr[\mathcal{G}_0^{\mathcal{A}} = 1] - \Pr[\mathcal{G}_1^{\mathcal{A}} = 1]| \leq \varepsilon$$

where

```
$\mathcal{G}_0^{\mathcal{A}}$
    $(m_0, m_1) \leftarrow \mathcal{A}$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}$
    return $\mathcal{A}(\mathsf{Enc}_{\mathsf{pk}}(m_0))$

$\mathcal{G}_1^{\mathcal{A}}$
    $(m_0, m_1) \leftarrow \mathcal{A}$
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}$
    return $\mathcal{A}(\mathsf{Enc}_{\mathsf{pk}}(m_1))$
```

Give a corresponding construction that converts any IND-FPA secure encryption scheme to the IND-CPA secure encryption scheme. What is the corresponding overhead in communication and computation if we want to preserve the size of the message space?

Solution. t ?? Now we construct and adversary $\mathcal{A}$ that plays IND-FPA game. At first $\mathcal{A}$ only outputs two different messages. And later when public key is known, it initializes $\mathcal{B}$ and then, because $\mathcal{B}$ can give out different messages m_0 and m_1, $\mathcal{A}$ needs to check the first one and just flip the output of $\mathcal{B}$. To be more specific, when $\mathcal{B}$ gives out $m_0 = m_1$ then $\mathcal{B}$ is not a good adversary. When $\mathcal{B}$ gives out $(0, 1)$ then it matches the output of $\mathcal{A}$, and when $\mathcal{B}$ gives out $(1, 0)$ then we need to flip the output of $\mathcal{B}$.

```
$\mathcal{A}$
    return $(0, 1)$

$\mathcal{A}(c)$
    $(\bar{m}_0, \bar{m}_1) \leftarrow \mathcal{B}(\mathsf{pk})$
    if $(\bar{m}_0 \overset{?}{=} 0)$ then
        return $\mathcal{B}(c)$
    else
        return $!\mathcal{B}(c)$
```

Construction of $\mathcal{A}$ is valid. The running time of $\mathcal{A}$ is equal to the running time of $\mathcal{B}$ plus 1 for comparison. So for a t-time $\mathcal{B}$ the running time of $\mathcal{B}$ is $t + 1$. The advantage of $\mathcal{A}$ is still the same as the advantage of $\mathcal{B}$. More formally: $\mathsf{Adv}^{\mathsf{ind\text{-}fpa}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}(\mathcal{B})$. This means that (t, ε) IND-FPA security implies $(t - 1, \varepsilon)$ IND-CPA security when message space is $\{0, 1\}$.

A similar construction can be used to show that same applies to encryption tuples. The IND-FPA and IND-CPA games are the same as before, but new instead of $\mathsf{Enc}_{\mathsf{pk}}$ a tuple encryption $\overline{\mathsf{Enc}_{\mathsf{pk}}}$ is used instead. Lets say that we have a good $(t_{\mathcal{B}}, \varepsilon)$ adversary $\mathcal{B}$ against IND-CPA. For it to be good, it makes sense for it to always choose m_0 and m_1 such that they are distinct at every position. Now I give a construction of $\mathcal{A}$ that is good against IND-FPA.

```
$\mathcal{A}$
    return $((0_1, \ldots, 0_n), (1_1, \ldots, 1_n))$

$\mathcal{A}((c_1, \ldots, c_n))$
    $(\bar{m}_0, \bar{m}_1) \leftarrow \mathcal{B}(\mathsf{pk})$
    $b \xleftarrow{u} \{0, 1\}$
    $(x_1, \ldots, x_n) = \bar{m}_b$
    for $i \leftarrow \{1, \ldots, n\}$
        if $(x_i \overset{?}{=} !b)$ then
            $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(!b)$
    return $\mathcal{B}((c_1, \ldots, c_n))$
```

Basically $\mathcal{A}$ chooses bit b to guess if it is playing IND-FPA game $\mathcal{G}_0$ or $\mathcal{G}_1$. Then it encrypts the other bits itself. And returns what $\mathcal{B}$ does with this new encryption tuple.

The advantage of A is by definition: $|\Pr\left[\mathcal{G}_0^{\mathcal{A}} = 1\right] - \Pr\left[\mathcal{G}_1^{\mathcal{A}} = 1\right]|$. Lets look at first what is the advantage that $\mathcal{A}$ wins in $\mathcal{G}_0$. If b is guessed correctly then, $\mathcal{A}$ receives an valid encryption of $\bar{m}_0$. Then the probability is equal to $\Pr\left[\mathcal{Q}_0^{\mathcal{B}} = 1\right]$. On the other hand if b is wrong then $\mathcal{B}$ does not get the encryption of $\bar{m}_0$ and probably not the encryption of $\bar{m}_1$ either. And in that case $\mathcal{B}$ can just run into an infinite loop, give out bottom, or just make a wild guess, which is kind of bad, so I ignore that. I can ignore that because I am giving a lower bound on the advantage and the wild guess from the other game cancels that part mostly out. In total: $\Pr\left[\mathcal{G}_0^{\mathcal{A}} = 1\right] = \Pr\left[b \text{ is correct}\right]\Pr\left[\mathcal{Q}_0^{\mathcal{B}} = 1\right] = \frac{1}{2}\Pr\left[\mathcal{Q}_0^{\mathcal{B}} = 1\right]$. The same argumentation can be applied in the case of game $\mathcal{G}_1^{\mathcal{A}}$, and from there we get that $\Pr\left[\mathcal{G}_1^{\mathcal{A}} = 1\right] = \frac{1}{2}\Pr\left[\mathcal{Q}_1^{\mathcal{B}} = 1\right]$. We put those two together and we get that

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-fpa}}(\mathcal{A}) &= \left|\frac{1}{2}\Pr\left[\mathcal{Q}_0^{\mathcal{B}} = 1\right] - \frac{1}{2}\Pr\left[\mathcal{Q}_1^{\mathcal{B}} = 1\right]\right| \\
&= \frac{1}{2}\left|\Pr\left[\mathcal{Q}_0^{\mathcal{B}} = 1\right] - \Pr\left[\mathcal{Q}_1^{\mathcal{B}} = 1\right]\right| \\
&= \frac{1}{2}\mathsf{Adv}^{\text{ind-cpa}}(\mathcal{B}) = \frac{1}{2}\varepsilon \ .
\end{aligned}$$

Construction of $\mathcal{A}$ is valid. Running time of $\mathcal{A}$ is $t_{\mathcal{B}} + nt_{\mathsf{Enc}}$. That is, the time to run $\mathcal{B}$, and at-most n times to encrypt.

In conclusion (t, ε) IND-FPA security implies $(t - nt_{\mathsf{Enc}}, 2\varepsilon)$ IND-CPA security.

??

The solution is to use a encryption tuple from the previous task. For that to work, we need to be able to represent the message space in binary. For all practical things, this is doable, because we can represent almost anything in computers, and they all work with binary encoding. One way to give a binary representation for message space is to create a linear ordering on the message space and then label the messages with integers. One extra assumption that we must make is that the message space is finite. So lets assume that there are n elements in the message space.

The overhead in computation is that, you need to convert the object into binary, which can be done in $\mathrm{O}(\log_2 n)$ by using binary search on the prepared and sorted table of elements from the message space. Then you need to run encryption. Because n elements can be represented in binary with $log_2 n$ bits, that means that you have to run encryption $\log_2 n$ times. So the total complexity of encryption now is $\mathrm{O}(\log_2 n) + \log_2 nt$, where t is the time needed for the original encryption. On decryption the operations are done in the other way, but we still need to run decryption $\log_2 n$ times, and then use a binary search to lookup the corresponding message.

The communication needed is also $\log_2 n$.