8.1 CPA-secure encryption

8.1.a Σ' CPA\$-secure?

We consider the following distinguisher:

Distinguisher A
pick $k \in \{0,1\}^\lambda$
pick $m \in \{0,1\}^\lambda$
$y = \Sigma'(k,m)$
return $first\ two\ bits\ of\ y \overset{?}{=} 00$

First we will pick k and m at random. Then the return value of Σ' is checked whether the two starting bits are both 0 or not.

It is obvious that for $L^{\Sigma'}_{CPA-real}$ the distinguisher will always output 1. For $L^{\Sigma'}_{CPA-rand}$ the distinguisher will only output 1 if the random algorithm outputs a bit-string with the first two bits being 0. The probability for this will be $\frac{1}{4}$.
For the advantage we will then get:

$$\begin{aligned} Bias(A) &= \mid Pr(A \diamond L^{\Sigma'}_{CPA-real}) - Pr(A \diamond L^{\Sigma'}_{CPA-rand}) \mid \\ &= \mid 1 - \frac{1}{4} \mid \\ &= \frac{3}{4} \neq 0 \end{aligned}$$

Therefore Σ' is not CPA\$-secure.

8.1.b Σ' CPA-secure?

1. We will show that $\Sigma' \diamond \Sigma(k, m_L)$ is indistinguishable from $\Sigma' \diamond \Sigma(k, m_R)$:

$L^{\Sigma'}_{CPA-L}$
pick $k \in \{0,1\}^\lambda$
<u>EAVESDROP(m_L, m_R)</u>
<u>if</u> $\mid m_L \mid \neq \mid m_R \mid$
<u>then</u> return ERROR
$c := 00 \| \Sigma.Enc(k, m_L)$
return c

This is our starting library $L^{\Sigma'}_{CPA-L}$.

$L^{\Sigma'}_{CPA}$
pick $k \in \{0,1\}^\lambda$
<u>EAVESDROP(m_L, m_R)</u>
<u>if</u> $\mid m_L \mid \neq \mid m_R \mid$
<u>then</u> return ERROR
$c := 00 \| \Sigma.Enc(k, (m_L, m_R))$
return c

$\diamond$

L^{Σ}_{CPA-L}
<u>Enc($k, (m_L, m_R)$)</u>
<u>if</u> $\mid m_L \mid \neq \mid m_R \mid$
<u>then</u> return ERROR
$c := \Sigma.Enc(k, m_L)$
return c

With adding L^{Σ}_{CPA-L} we create a hybrid-library.

$L^{\Sigma'}_{CPA}$
pick $k \in \{0,1\}^\lambda$

EAVESDROP(m_L, m_R)
if $|m_L| \neq |m_R|$
then return ERROR
$c := 00 \| \Sigma.Enc(k, (m_L, m_R))$
return c

$\diamond$

L^{Σ}_{CPA-R}
Enc($k, (m_L, m_R)$)
if $|m_L| \neq |m_R|$
then return ERROR
$c := \Sigma.Enc(k, m_R)$
return c

Because we know that Σ has CPA\$-security (especially also CPA-security $L^{\Sigma}_{CPA-L} \approx L^{\Sigma}_{CPA-R}$

$L^{\Sigma'}_{CPA-R}$
pick $k \in \{0,1\}^\lambda$

EAVESDROP(m_L, m_R)
if $|m_L| \neq |m_R|$
then return ERROR
$c := 00 \| \Sigma.Enc(k, m_R)$
return c

Therefore we can inline this subroutine to end with the library $L^{\Sigma'}_{CPA-R}$. Therefore we showed that $L^{\Sigma'}_{CPA-L} \approx L^{\Sigma'}_{CPA-R}$.

8.2 From a PRP to CPA-secure encryption

8.2.1 Corresponding Decoder

8.2.1.a

$Enc(k, m)$:
$r \leftarrow \{0,1\}^\lambda$
$z := F(k, m) \oplus r$
return (r, z)

$Dec(r, z)$:
$c = z \oplus r$
$m' = F^{-1}(c)$
return m'

8.2.1.b

$Enc(k, m)$:
$r \leftarrow \{0,1\}^\lambda$
$s := r \oplus m$
$x := F(k, r)$
return (s, x)

$Dec(r, z)$:
$r = F^{-1}(x)$
$m' = r \oplus s$
return m'

8.2.1.c

$Enc(k, m)$:
$s_1 \leftarrow \{0,1\}^\lambda$
$s_2 := s_1 \oplus m$
$x := F(k, s_1)$
$y := F(k, s_2)$
return (x, y)

$Dec(r, z)$:
$s_1 = F^{-1}(x)$
$s_2 = F^{-1}(y)$
$m' = s_2 \oplus s_1$
return m'

8.2.2 CPA-secure?

8.2.2.a

No, it is not *CPA-secure*, because:

Proof:

$$\left(\begin{array}{|l|}\hline L^{\Sigma}_{CPA-L}: \\ k \leftarrow \{0,1\}^{\lambda} \\ \\ \underline{\textsc{Eavesdrop}(m_L, m_R)} \\ \quad \underline{\text{if}}\ |m_L| \neq |m_R| \\ \quad \underline{\text{then}}\ \text{return } \textsc{Error} \\ \quad c := \Sigma.Enc(k, m_L) \\ \quad \textbf{return } c \\ \hline \end{array} \overset{?}{\approx} \begin{array}{|l|}\hline L^{\Sigma}_{CPA-R}: \\ k \leftarrow \{0,1\}^{\lambda} \\ \\ \underline{\textsc{Eavesdrop}(m_L, m_R)} \\ \quad \underline{\text{if}}\ |m_L| \neq |m_R| \\ \quad \underline{\text{then}}\ \text{return } \textsc{Error} \\ \quad c := \Sigma.Enc(k, m_R) \\ \quad \textbf{return } c \\ \hline \end{array} \right) \diamond \begin{array}{|l|}\hline Enc(k,m): \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad z := F(k,m) \oplus r \\ \quad \textbf{return } (r, z) \\ \hline \end{array}$$

We will consider the following distinguisher:

$$\begin{array}{|l|}\hline Distinguisher\ A \\ \textbf{pick } x \in \{0,1\}^{\lambda} \\ \textbf{pick } y \in \{0,1\}^{\lambda} \\ (r_1, c_1) = \textsc{Eavesdrop}(x, x) \\ (r_2, c_2) = \textsc{Eavesdrop}(x, y) \\ \textbf{return } c_1 \oplus r_1 \overset{?}{=} c_2 \oplus r_2 \\ \hline \end{array}$$

First we will pick two messages x and y at random. Then the return values of $\textsc{Eavesdrop}(m_L, m_R)$ are fetched. In the end the XOR-step which the encoder has done, will be reversed by using the return values. The distinguisher will therefore output whether both $F(k,m)$ have returned the same value.

It is obvious that if the L^{Σ}_{CPA-L} is used the adversary will output 1 because in both cases $F(k,x)$ is returned. If the L^{Σ}_{CPA-R} is used one will output $F(k,x)$ and the other one will output $F(k,y)$ which will not be the same and therefore the output value of A will be 0. So the adversary can distinguish between L^{Σ}_{CPA-L} and L^{Σ}_{CPA-R} and therefore this encoder is not *CPA-secure*.

8.2.2.b

Yes, it is *CPA-secure*, because:

Proof (We will show that the encoder has *CPA\$-security* and is therefore also *CPA-secure*):

$$\begin{array}{|l|}\hline L^{\Sigma}_{CPA\$-real} \\ k \leftarrow \{0,1\}^{\lambda} \\ \\ \underline{\textsc{Ctxt}(m)} \\ \quad c := \Sigma.Enc(k, m) \\ \quad \textbf{return } c \\ \hline \end{array} \diamond \begin{array}{|l|}\hline Enc(k,m) \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad s := r \oplus m \\ \quad x := F(k,r) \\ \quad \textbf{return } (s, x) \\ \hline \end{array} \overset{?}{\approx} \begin{array}{|l|}\hline L^{\Sigma}_{CPA\$-rand} \\ \underline{\textsc{Ctxt}(m)} \\ \quad r \leftarrow (\{0,1\}^{\lambda})^2 \\ \quad \textbf{return } r \\ \hline \end{array}$$

$$\begin{array}{|l|}\hline L^{\Sigma}_{CPA\$-real} \\ k \leftarrow \{0,1\}^{\lambda} \\ \\ \underline{\textsc{Ctxt}(m)} \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad s := r \oplus m \\ \quad x := F(k,r) \\ \quad \textbf{return } (s, x) \\ \hline \end{array} \approx \begin{array}{|l|}\hline L^{\Sigma}_{CPA\$-hybrid} \\ \underline{\textsc{Ctxt}(m)} \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad s := r \oplus m \\ \quad x \leftarrow \{0,1\}^{\lambda} \\ \quad \textbf{return } (s, x) \\ \hline \end{array}$$

Because we know that F is a secure PRP/PRF we can inline the subroutine of a random PRF.

$\mathcal{L}^{\Sigma}_{CPA\$-hybrid}$		$\mathcal{L}^{\Sigma}_{CPA\$-hybrid}$	
$\underline{\text{CTXT}(m)}$ $r \leftarrow \{0,1\}^{\lambda}$ $s := r \oplus m$ $x \leftarrow \{0,1\}^{\lambda}$ **return** (s, x)	$\approx$	$\underline{\text{CTXT}(m)}$ $s \leftarrow \{0,1\}^{\lambda}$ $x \leftarrow \{0,1\}^{\lambda}$ **return** (s, x)	Because "XORing" a random bitstring with the message bitstring is also the same as generating a random bitstring, we can change this in the hybrid.

$\mathcal{L}^{\Sigma}_{CPA\$-hybrid}$		$\mathcal{L}^{\Sigma}_{CPA\$-rand}$	
$\underline{\text{CTXT}(m)}$ $s \leftarrow \{0,1\}^{\lambda}$ $x \leftarrow \{0,1\}^{\lambda}$ **return** (s, x)	$\approx$	$\underline{\text{CTXT}(m)}$ $r \leftarrow (\{0,1\}^{\lambda})^2$ **return** r	Because s and x are making up a pair which is in C we can inline such a subroutine to end up with $\mathcal{L}^{\Sigma}_{CPA\$-rand}$.

8.2.2.c

No, it is not *CPA-secure*, because if whether one of m_L and m_R is the 0-bit-string, the s_2 bit string will be the same as the s_1 bit string and therefore the return value x and y will be bitwise equal.

8.3 Modes of operation

8.3.1 CBC

It will affect the block which is corresponding with the encrypted block with the error as well as the following block.

8.3.2 OFB

It will only affect the block which is corresponding with the encrypted block with the error.

8.3.3 CTR

It will only affect the block which is corresponding with the encrypted block with the error.