

Homework 2

1. Our KeyGen function is as follows:

KeyGen:
do
$k \leftarrow \{0, 1\}^\lambda$
until $k \neq 0^\lambda$
return k

We will use these EAVESDROP functions:

$\mathcal{L}_{\text{ots-L}}^\Sigma$	$\mathcal{L}_{\text{ots-R}}^\Sigma$
EAVESDROP(m_L, m_R):	EAVESDROP(m_L, m_R):
$k \leftarrow \Sigma.\text{KeyGen}$	$k \leftarrow \Sigma.\text{KeyGen}$
$c := k \oplus m_L$	$c := k \oplus m_R$
return c	return c

This calling function will call an EAVESDROP function with the values 0^λ and 1^λ .

A
EAVESDROP(m_L, m_R):
$c := \text{EAVESDROP}(0^\lambda, 1^\lambda)$
return $c = ? 1^\lambda$

$A \diamond \text{ots-L}$ produces c as an encryption of m_L . This produces a uniformly random result out of the 15 possibilities ($1/(2^\lambda - 1)$). Those 15 possibilities will never generate 0^λ .

$A \diamond \text{ots-R}$ produces c as an encryption of m_R . This produces a different result, because you will never get 1^λ back from ots-R.

Every possible input to EAVESDROP will produce a different set of possible outputs with the same probability. Incidentally, every possible input to EAVESDROP will never produce that input in the set of possible outputs.

$$\Pr[A \diamond \text{ots-L} \Rightarrow \text{true}] = 1/(2^\lambda - 1).$$

$$\Pr[A \diamond \text{ots-R} \Rightarrow \text{true}] = 0.$$

These two probabilities are different, so the modified KeyGen does not satisfy one-time secrecy.

2. Encryption Scheme:

$\mathcal{K} = (\Sigma.\mathcal{K})^2$ $\mathcal{M} = \Sigma.\mathcal{M}$ $\mathcal{C} = \Sigma.\mathcal{C}$	$\text{Enc}((k_1, k_2), m):$ $c_1 := \Sigma.\text{Enc}(k_1, m)$ $c_2 := \Sigma.\text{Enc}(k_2, m)$ $\text{return } (c_1, c_2)$	$\text{Dec}((k_1, k_2), (c_1, c_2)):$ $m_1 := \Sigma.\text{Dec}(k_1, c_1)$ $m_2 := \Sigma.\text{Dec}(k_2, c_2)$ $\text{if } m_1 \neq m_2 \text{ return err}$ $\text{return } m_1$
$\text{KeyGen}:$ $k_1 \leftarrow \Sigma.\mathcal{K}$ $k_2 \leftarrow \Sigma.\mathcal{K}$ $\text{return } (k_1, k_2)$		

It is given that:

$\mathcal{L}_{\text{ots-L}}^\Sigma$ $\text{EAVESDROP}(m_L, m_R):$ $k \leftarrow \Sigma.\text{KeyGen}$ $c := k \oplus m_L$ $\text{return } c$	\equiv	$\mathcal{L}_{\text{ots-R}}^\Sigma$ $\text{EAVESDROP}(m_L, m_R):$ $k \leftarrow \Sigma.\text{KeyGen}$ $c := k \oplus m_R$ $\text{return } c$
--	----------	--

We will start with ots-L encrypt twice (ots-L2):

$\mathcal{L}_{\text{ots-L2}}^\Sigma$ $\text{EAVESDROP}(m_L, m_R):$ $k_1 \leftarrow \Sigma.\text{KeyGen}$ $k_2 \leftarrow \Sigma.\text{KeyGen}$ $c_1 := k_1 \oplus m_L$ $c_2 := k_2 \oplus m_L$ $\text{return } c_1, c_2$
--

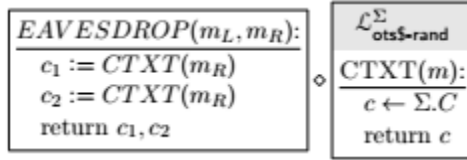
Now we will factor out two separate sets of statements into one subroutine, changing nothing about the behavior of the scheme:

$\text{EAVESDROP}(m_L, m_R):$ $c_1 := \text{CTXT}(m_L)$ $c_2 := \text{CTXT}(m_L)$ $\text{return } c_1, c_2$	\diamond	$\mathcal{L}_{\text{ots\$-real}}^\Sigma$ $\text{CTXT}(m):$ $k \leftarrow \Sigma.\text{KeyGen}$ $c := k \oplus m$ $\text{return } c$
--	------------	---

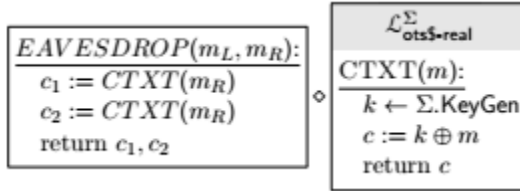
Next, we will replace ots\$-real with ots\$-rand, changing nothing about the behavior of the scheme:

$\text{EAVESDROP}(m_L, m_R):$ $c_1 := \text{CTXT}(m_L)$ $c_2 := \text{CTXT}(m_L)$ $\text{return } c_1, c_2$	\diamond	$\mathcal{L}_{\text{ots\$-rand}}^\Sigma$ $\text{CTXT}(m):$ $c \leftarrow \Sigma.C$ $\text{return } c$
--	------------	--

Next, we will change the statements in EAVESDROP to use m_R instead of m_L , again changing nothing about the behavior of the scheme:



Now, we can make a series of changes to get back to our original scheme, but using m_R instead of m_L :



Finally, we can inline our $\text{ots}\$-real$ subroutine to make ots-R2 :

