CS-E4340 Cryptography: Exercise Sheet 10

Submission deadline: November 28, 2022, 11:30, via MyCourses

Each exercise can give up to two participation points, 2 for a mostly correct solution and 1 point for a good attempt. Overall, the exercise sheet gives at most 4 participation points. We encourage to **choose** exercises which seem **interesting** and/or adequately challenging to you.

Exercise Sheet 10 is intended to help...

(a) ...understand oblivious transfer (Ex. 1).
(b) ...analyze secret-sharing protocols (Ex. 2).
(c) ...understand how to run a secret-sharing protocol (Ex. 3).
(d) ...familiarize yourself with simulation-based definitions (Ex. 4).

Exercise 1 (Simple OT)**.** One very useful primitives from secure multi-party computation is *oblivious transfer (OT)*, where receiver and sender have the following inputs and outputs

Sender Inputs:	secret strings s_0, s_1	**Receiver inputs:**	bit b
Sender Output:	none	**Receiver output:**	bit s_b

Security of an OT protocol guarantees that

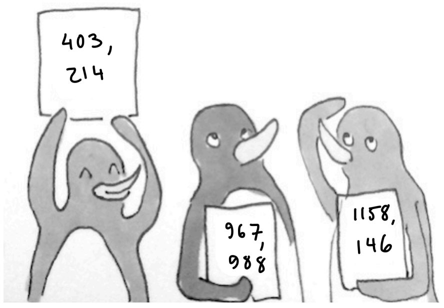
(1) the receiver learns nothing about s_{1-b} and that
(2) the sender learns nothing about the receiver's bit b.

Figure 1 describes a simple OT protocol which is similar to the Diffie-Hellman key exchange (first described in `https://eprint.iacr.org/2015/267.pdf`).
Task: Explain why (2) holds information-theoretically, i.e., the sender learns nothing about the receiver's secret bit. In addition, elaborate on the problems which need to be difficult for (1) to hold, i.e., that the receiver does not learn any information about the sender's secret bit.

Exercise 2 (Penguin C Attack)**.** In the lecture, we discussed a secure communication protocol where penguin A, B, and C each hold a secret bit b_A, b_B, and b_C, respectively, and want to compute $b = b_A \wedge b_B \wedge b_C$ without revealing to each other anything except for the result b.
Task: Explain how penguin C can distinguish between the case that $b_A = b_B = 0$ and $b_A = b_B = 1$, regardless of whether C's own bit b_C is 0 or 1.

Exercise 3 (Shamir's Secret Sharing)**.** Study the section on Shamir's Secret Sharing in the lecture notes and answer the following:

- why the adversary cannot learn the secret when only $t-1$ shares are leaked?
- is the scheme secure only against polynomial time (polynomial in the bit-length of the secret) adversaries or can we allow exponential runtime adversary? Why?
- Consider Shamir's Secret Sharing with threshold $t = 3$ and $q = 1361$. Suppose Penguin A has share (403, 214), Penguin B has share (967, 988) and Penguin C has share (1158, 146), where the first value of the pair is x and second value is y.



What is the secret? Please explain all the steps.

Hint: You may use any technique for polynomial interpolation that you are familiar with and you may tools such as Wolfram Alpha to invert the Vandermonde Matrix or use some other tool to solve Gaussian elimination.

Hint: The secret is our most favorite code.

Exercise 4 (Simulation-Based Security of Symmetric Encryption)**.** We want to write IND-CPA security of a symmetric encryption scheme using the simulation-paradigm. Below, we give the outline of such a simulation-based definition. Decide which inputs the simulator gets and explain why. Explain why you think that the resulting notion is (or is not) equivalent to the standard notion of IND-CPA security. Optional: Sketch a reduction argument.

Remark: In the definition sim-IND-CPA security, the game maintains the state variable *st* to keep the state of the simulator.

Definition 1 (sim-IND-CPA). *A symmetric encryption scheme* se *is sim-IND-CPA-secure if there exists a PPT simulator* $\mathcal{S}$ *such that the real game* $\mathtt{Gsimind\text{-}cpa}^0$ *and the ideal game* $\mathtt{Gsimind\text{-}cpa}^1$ *are computationally indistinguishable, that is, for all PPT adversaries* $\mathcal{A}$*, the advantage*

$$\begin{aligned}&\mathbf{Adv}_{\mathcal{A}}^{\mathtt{Gsimind\text{-}cpa}^0,\mathtt{Gsimind\text{-}cpa}^1}(\lambda)\\ &:= |\Pr\left[1 = \mathcal{A} \to \mathtt{Gsimind\text{-}cpa}^0\right] - \Pr\left[1 = \mathcal{A} \to \mathtt{Gsimind\text{-}cpa}^1\right]|\end{aligned}$$

is negligible in λ.

```
$\mathtt{Gsimind\text{-}cpa}^0_{\mathsf{se}}$

Parameters
  $\lambda$:   sec. parameter
  $\mathsf{se}$:  sym. enc. sch.

Package State
  $k$:   key

$\mathsf{ENC}(x)$
  if $k = \bot$ :
      $k \leftarrow\$ \{0,1\}^\lambda$
  $c \leftarrow\$ \mathsf{se.enc}(k, x)$
  return $c$
```

```
$\mathtt{Gsimind\text{-}cpa}^1_{\mathcal{S}}$

Parameters
  $\lambda$:   sec. parameter
  $\mathcal{S}$:   simulator

Package State
  $st$:  simulator state

$\mathsf{ENC}(x)$
  —your code here—
  $(c, st) \leftarrow\$ \mathcal{S}($—your input here$, st)$
  return $c$
```

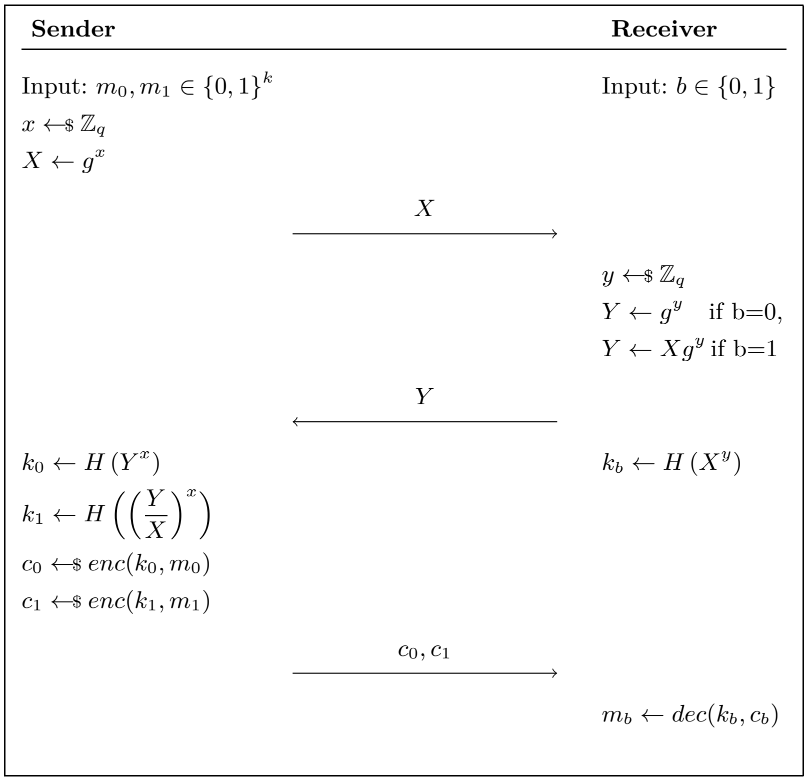

Sender
Receiver
Input: $m_0, m_1 \in \{0,1\}^k$
Input: $b \in \{0,1\}$
$x \leftarrow\!\!\$\ \mathbb{Z}_q$
$X \leftarrow g^x$
$\xrightarrow{X}$
$y \leftarrow\!\!\$\ \mathbb{Z}_q$
$Y \leftarrow g^y$ if b=0,
$Y \leftarrow Xg^y$ if b=1
$\xleftarrow{Y}$
$k_0 \leftarrow H\left(Y^x\right)$
$k_b \leftarrow H\left(X^y\right)$
$k_1 \leftarrow H\left(\left(\frac{Y}{X}\right)^x\right)$
$c_0 \leftarrow\!\!\$\ enc(k_0, m_0)$
$c_1 \leftarrow\!\!\$\ enc(k_1, m_1)$
$\xrightarrow{c_0, c_1}$
$m_b \leftarrow dec(k_b, c_b)$

Fig. 1: Simple OT protocol