Institute of Computer Science, University of Bern
Cryptographic Protocols, FS 2021

Prof. Christian Cachin
Jovana Mićić

Exercise 1

1.1 Calculation in finite fields (4pt)

We define two-party secure computation protocols as protocols for evaluating a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, where $x \in \mathcal{X}$ is the input of Alice, $y \in \mathcal{Y}$ is the input of Bob, and $z = f(x, y) \in \mathcal{Z}$ is the common output.

The function f is either represented as a *Boolean circuit* composed of wires and logical gates, with $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ being binary strings, or as an *arithmetic circuit*. An arithmetic circuit takes inputs in a finite field $GF(q)$, also written as $\mathbb{F}_q$, and its operations are the addition and multiplication operations of $GF(q)$.

There are q distinct elements in $GF(q)$. If q is a prime, then calculation in $GF(q)$ corresponds to addition and multiplication modulo q. We also write $\mathbb{Z}_q$ for addition on $\{0, \ldots, q-1\}$ modulo q and $\mathbb{Z}_q^*$ for multiplication on $\{1, \ldots, q-1\}$ modulo q. If you are not familiar with the arithmetic of finite fields, check [Sec. 1.2, Smart16] or any text on algebra.

If $q = p^k$ for some prime p and $k > 1$, then the elements of $GF(q)$ are represented by the set of polynomials in $GF(p)[X]/P$, where $P \in GF(p)[X]$ is an irreducible polynomial of degree k over $GF(p)$.

The addition and multiplication operations of $GF(4)$, for instance, are as follows:

$+$	$\mathbf{0}$	$\mathbf{1}$	α	$\mathbf{1}+\alpha$
$\mathbf{0}$	0	1	α	$1+\alpha$
$\mathbf{1}$	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$\mathbf{1}+\alpha$	$1+\alpha$	α	1	0

$*$	$\mathbf{0}$	$\mathbf{1}$	α	$\mathbf{1}+\alpha$
$\mathbf{0}$	0	0	0	0
$\mathbf{1}$	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$(\mathbf{1}+\alpha)$	0	$1+\alpha$	1	α

Tasks:

a) Evaluate the polynomial $r(X) = 3X^3 + 2X^2 + X \in GF(5)[X]$ at $X = 2$.

b) Evaluate the polynomial $s(X) = (1+\alpha) \cdot X^3 + \alpha \cdot X^2 + X \in GF(4)[X]$ at $X = \alpha$.

1.2 Trivial functions for secure computation (3pt)

A function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is *trivial* for secure computation whenever the output together with one input determines the other input. In other words, Alice may infer Bob's input because for every x and z, there is a unique y such that $z = f(x, y)$, and the same holds for Bob, who can determine x from y and z. These functions may be computed "in plain" and without the help of any cryptography or secure computation protocols.

$\Longrightarrow$

Determine which ones among the following functions are trivial. All of them except for d are defined on $GF(p)$ for some prime $p > 2$ (e.g., think of $p = 5$) and function d is defined on $GF(p) \setminus \{0\}$.

a) $a(x, y) = x + y$.

b) $b(x, y) = \max\{x, y\}$.

c) $c(x, y) = x \cdot y$.

d) $d(x, y) = x \cdot y$, where domain and range do not include 0.

e) $e(x, y) = \begin{cases} 0 & \text{if } x < y \\ 1 & \text{otherwise} \end{cases}$

1.3 Non-trivial functions and an embedded OR (3pt)

It has been shown that any two-argument function that is *not* trivial for secure computation contains a so-called *embedded OR*[1]. This is a pair of inputs x_1, x_2 for Alice and a pair of inputs y_1, y_2 for Bob such that

$$\begin{aligned} f(x_1, y_1) = f(x_1, y_2) = f(x_2, y_1) &= c \\ f(x_2, y_2) &\neq c \end{aligned}$$

for some c in the range of f. For instance, $(x_1, x_2) = (1, 0)$ and $(y_1, y_2) = (2, 0)$ form an embedded OR in the equality function *eq* over $GF(3)$:

eq	**0**	**1**	**2**
0	1	0	0
1	0	1	0
2	0	0	1

This means, e.g., that when Alice inputs $x = 1$ and $eq(x, y) = 0$ results, she cannot infer whether Bob's input was $y = 0$ or $y = 2$.

Consider again the five functions from problem 1.2. Show an embedded OR for all those that are not trivial (and use $GF(5)$).

[1] The notion might equivalently be called *embedded AND*, *embedded NOR*, or *embedded NAND*.