Institute of Computer Science, University of Bern
Cryptographic Protocols, FS 2023

Prof. Christian Cachin
Jovana Mićić

Exercise 12

MPyC

MPyC is a library for secure multiparty computation in Python, developed by Berry Schoenmakers and team from Eindhoven University of Technology.

Home page: `https://www.win.tue.nl/~berry/mpyc/`

Some background: `https://www.win.tue.nl/~berry/mpyc/TPMPC2018.pdf`

Source: `https://github.com/lschoe/mpyc`

Install it using

```
$ pip3 install git+https://github.com/lschoe/mpyc
```

to get the latest version from GitHub or from the source (ensure you have Python version at least 3.7) and study some of the samples in the `demo` directory. Recommended are:

```
demos/SecretSantaExplained.py
demos/sort.py
```

In preparation of the project, we also provide in ILIAS sample code for computing the maximum (`ex12-samplemax.py`). Please read the instructions in the file to learn how to run it for a different number of correct/corrupted parties.

12.1 Fair allocation (+10pt)

Fair division considers the problem of splitting up goods among two or more people such that everyone gets a "fair" share according to their own taste. An illustrative example considers an inhomogeneous cake, which is to be divided among two people: the first cuts it in two pieces and the second chooses a piece. Both receive a piece that is at least half in their valuation. The problem has been studied in economics, social sciences, and mathematics[1].

Here we consider the *fair allocation* of a set of m indivisible items[2] among n parties (e.g., furniture pieces after a divorce, an inheritance following a death, or cities and territories after an armistice). Our goal is to compute a fair division securely, such that every party inputs a sealed bid that must stay secret, using a simple allocation scheme. In general, defining a criterion that makes an allocation fair can be difficult and computing such an allocation is a complex optimization problem.

More precisely, denote the items by $\mathcal{I} = \{1, \ldots, m\}$. Every party $P_1, \ldots, P_n$ inputs a list $V_i = (v_{i1}, \ldots, v_{im})$ containing its valuation, where $\sum_{j=1}^{m} v_{ij} = B$ and v_{ij} denotes the preference

[1]`https://en.wikipedia.org/wiki/Fair_division`
[2]`https://en.wikipedia.org/wiki/Fair_item_allocation`

of P_i for item j. The number B is fixed. An allocation $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ consists of n sets with $\mathcal{A}_i \subseteq \mathcal{I}$ and gives items worth $\sum_{j \in \mathcal{A}_i} v_{ij}$ to P_i. A *maximal* allocation achieves the highest total worth, summed over all parties.

Implement an algorithm for finding the maximal allocation using MPyC that keeps all valuations secret. It should use exhaustive search, enumerate all allocations, and return some maximal allocation.