Make Crypto Great Again : Task4 README

Sofiane Azogagh, Aubin, Birba, Victor Delfour, Marc-Olivier Killijian

16 septembre 2022

1 Introduction

This document is a user guide regarding our participation to the IDASH PRIVACY & SECURITY WORKSHOP 2022 - secure genome analysis competition task 4.

2 Running the code

The code runs under python 3 latest stable release as of 12/09/2022.
Several modules are required : json ,time, numpy, pandas, hashlib, secrets, sympy, socket, PyCryptodome, argparse. All these packages are installed in the Docker containers party_a and party_b.

Our solution is based on two programs, one for hospital A and one for hospital B. The hospitals run a protocol in order to achieve the record linkage. Hospital B has to be launched before hospital A which connects to B.

There are several arguments and parameters that must be given to both mainA.py (in A's container) and mainB.py (in B's container) for the protocol to work correctly : the datasets filenames, the other party/hospital IP address and port number but most importantly the number of tuples that we want to use. This number of tuples represent the trade-off between accuracy and efficiency, it can vary from 1 (least accurate, most efficient) to 14 (most accurate, least efficient). By default it has been set to 3 as it achieves on the learning set an accuracy of 99.01% and according to the task's evaluation criteria, it doesn't make sense to go beyond that. However, if one's want to achieve more accuracy, because the evaluation rules are modified, or because the test dataset is very different to the training set or has been generated differently, do not hesitate to try different values.

Our solution is only intended to run on one server. However, the container's images create two different working containers without access to the other

container's data. The protocol can run on two separate servers with a proper deployment.

Docker and Docker compose are needed are required to run the protocol.

Here is how to launch the protocol, run :
— docker compose build
— docker compose up
To obtain OutputA do :
— Restart the container for $party_a$ and only this one
— Go to the root folder. The result is in the the OutputA.csv file.
To obtain OutputB do :
— Close the container for $party_a$
— Restart the container for $party_b$ and only this one
— Go to the root folder. The result is in the the OutputB.csv file.

3 Our solution

Our record linkage solution consists in matching tuples of different attributes. Indeed, most attributes, taken as a singleton, cannot be used to identify reliably an individual, there are too many Peters for example to use the first name as an identifier. Yet, by combining several quasi-identifiers we can achieve much better performance for the matching, for example by combining name and zip-code or surname, name and email address. However, the data contains both errors and missing values so we cannot rely on a just a tuple combining all the attributes. In order to mitigate the missing values and the errors, we use different tuple combinations chosen among those that give very little false positives. If one tuple from an individual in dataset A is the same as for an individual in dataset B, then those individuals are considered as linked.

A tuple is created by concatenating the different fields values and then by hashing the resulting value with a SHA256.

For example, we create the tuple (First Name, Last Name, Birthday). A computes a list of hashes Hash(First Name||Last Name||Birthday) for all the individuals in its dataset, B does the same and they both enter a Private Set Intersection protocol in order to determine the linked individuals. This protocol is then repeated for the other tuples, e.g. (ZipCode, Email Address), etc. Some links are discovered by several tuples, some other by only one. The protocol outputs two files OutputA.csv and OutputB.csv that contain only one column, which value is either True or False, depending if the individual has been matched by at least one tuple or not.

4 Security of our solution

We use a basic Private Set Intersection (PSI) presented in [1] to compute the matching between the 2 parties. The security of this PSI is based on the Elliptic Curve Discrete Logarithm (ECDL) problem. The elliptic curve used is the NIST-P256 standardized one and it fits with the 128 bits security level required. We communicate the tuples and protect them by using the private set intersection from [1] with the curve P-256 and without the cuckoo filters optimisation.

B is the one computing the intersection between the two sets and sends back to A its linked IDs. We protect this data the following way :

— First A shuffles its data and retains the information to be able to inverse the permutation. This way A's real IDs are hidden behind the permutation. For instance, A has individuals (1,2,3), the permutation is (2,3,1), so A's set becomes (2,3,1)

— Then A follows the PSI protocol, sends its set tuples in the order of the permutation (for example, the set is (2,3,1), so A sends ((Tuple(2),Tuple(3),Tuple(1)) to B)

— B computes the intersection between its own tuples and A's tuples and sends back to A the linked IDs regarding to the position of the clients in A's set (for example, imagine that (Tuple(2),Tuple(1)) are both linked from A's set. Then, B will send back to A the list (1,3) and A will reverse the permutation for 1 that becomes 2 in the original A's ID list and 3 that becomes 1 in the original A's ID list).

This solution for IDs is sufficient for several reasons :
First, it is not possible to reverse a random permutation without knowing the permutation itself.
Second, the entries are protected by the shared key used for the hash of the data (The data from the competition is already hashed with a private key shared by A and B) so an observer cannot infer anything related to the data, even when knowing which tuples are linked.
Third, if A wanted to do the same protocol with another hospital C, the hash key would change (So every element is different than what it looked like during the protocol between A and B) and the permutation is different so no information could be gathered on shared elements between A, B and C.

5 References

Références

[1] Amanda Cristina Davi Resende and Diego de Freitas Aranha. Faster unbalanced private set intersection. Cryptology ePrint Archive, Paper 2017/677, 2017. https://eprint.iacr.org/2017/677.