***INFO-F405 – Computer Security  
Project 1: Rainbow Tables***

1. Presentation of the project

We have to create a software which has to find one of the passwords corresponding to a fingerprint with the help of a rainbow table based attack. The stolen fingerprint has a size of 24 bits and the password to find has a size of 12 bits. The rainbow table has to use must be complete, so the password should be found at the end of the program execution. Four different reduction functions have to be used to create the table.

According to the size of the password, there are no more than 212 passwords to create, hash multiply times and store – with their fingerprint – Into the rainbow table.

The software can be realized in Java or C++.

1. Presentation of the work

We find out that the project can be split up into some steps which allow us to divide the work fairly into our group of three people:

* The creation of the rainbow table : the dictionary and the fingerprint corresponding to each password
* The password hash
* The creation of reduction functions and the selection of the best of them

According to the fact that each stolen fingerprint – or a modified fingerprint, depending where we are into the algorithm – is used to find a similar into the rainbow table, the search into it has to be the fastest available. To do that, we decide to use the dichotomic search with a rainbow table which has been sorted at the end of its creation.

The password hash has been realized thanks to the library crypto++ as we decided to use C++ as software language considering it is the best language for the bits manipulation.

The creation of reductions functions has been made randomly in the sense that there are millions of possibilities and it is not humanly possible to test them all. The main purpose during the creation to the reduction functions we wanted to use was to have a function returning a 12 bits set based on the maximum number of available bits.

The selection of the best reduction functions has been made that way:

* We create a table with the reduced fingerprint of every password with a specific reduction function for the complete dictionary
* We count the number of collisions and save
* We repeat the operation for another reduction function until we test all the reduction functions that we previously created.
* We choose the four reduction functions that creates the lowest number of collisions.