Report LAB 2

Stefano Leggio, Aleksandra Matla

December 20, 2021

1 Task 1

The protocol is implemented in three python files: *prover.py*, *verifier.py* and *authentication.py*. Executing *authentication.py* you can see that everything is working correctly. Below a plot of the protocol running time vs p from 10^3 to 10^7, the code script is available in *plot_p_time.py*.
We can observe that the trend is exponential.

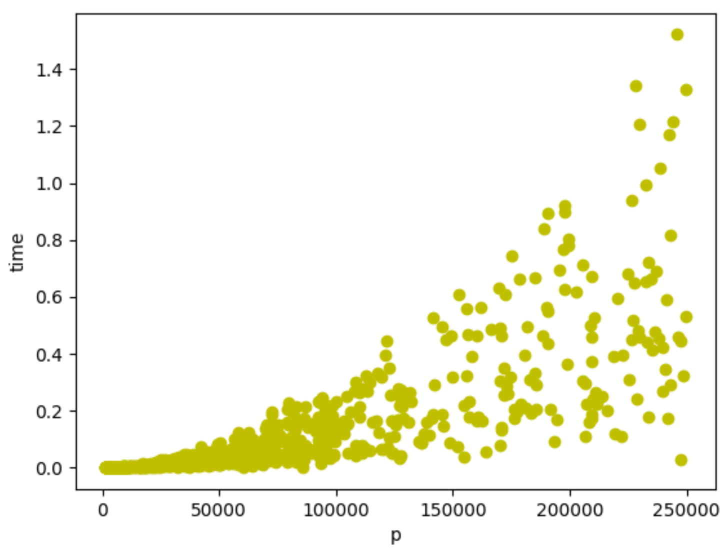

Figure 1: Plot of the protocol running time vs p

2 Task 2

We noticed that two pairs of t has tha same t_1, so we used the following formula to get k:

$$c_a = 1323262 \ t_a = (610187, 1018690)$$
$$\mathrm{c}_b = 2511813 \ t_b = (610187, 1684559)$$
$$\mathrm{k} = (\mathrm{t}_{b,2} c_a - t_{a,2} c_b)(t_{a,1}(t_{b,2} - t_{a,2}))^{-1} mod (p-1)$$

Key founded: 1346536
Below the output of *attack_task2.py*

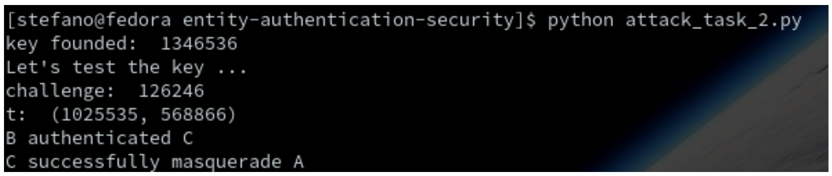

Figure 2: Output of *attack_task2.py*

3 Task 3

Since the probability of r is not uniform we were able to create a list of its values ordered from the lowest to the highest probability.
For the one attempt case we just used the last value of this list, so the most probable value of r. Then, we tested the algorithm keeping progressively all the values of the list.
Once selected the highest value of r we were able to get k with the following formula:

$$k = (c - rt_2)t_1^{-1} mod(p-1)$$

Notice that this formula works only if the modular moltiplicative inverse of t_1^{-1} can be done, so if and only if $gcd(t_1^{-1}, p-1) = 1$, this will influence the success probability.
For $p = 773$ the success probability of just one attempt averaged over 10000 iterations is 0.0831.

```
[*] Success probability of a single masquerade attempt (checking only the most probable value of r)
Running  10000  simulations...
Success probability:  0.0831  ( 8.31 % )
```

Figure 3: Output of *attack_task3.py*

Incrementing progressively the number of attempts we can reach the success probability of 0.4927.

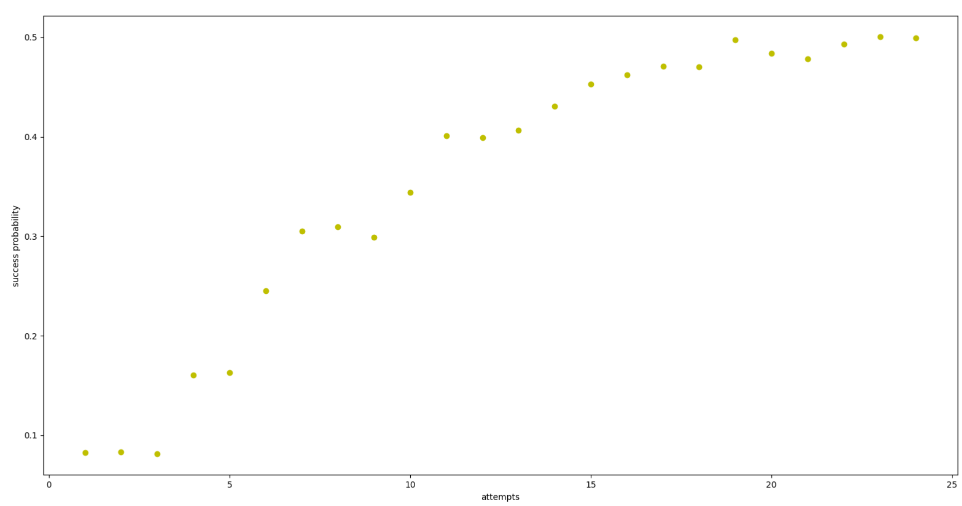

Figure 4: Plot of the success probability vs the number of attempts

4 Task 4

We used the following values of t to force a success response:

$q \in Z_{p-1}$

$$\begin{cases} t'_1 = k' \alpha^q mod\ p \\ t'_2 = -t'_1 mod(p-1) \end{cases}$$

If we substitute in the verifier we have:

$s = \tilde{s} \Leftrightarrow \alpha^u = \alpha^{q t'_2}$

In our case we picked $q = 0$:

$$\begin{cases} t'_1 = k' mod\ p \\ t'_2 = -t'_1 mod(p-1) \end{cases}$$

$s = \tilde{s} \Leftrightarrow n = -c\ mod\ p$

Using these formulas in a python script we authenticated successfully without knowing the key.

```
[stefano@fedora entity-authentication-security]$ python attack_task_4.py

Choosen variables:
 p: 1702079
 k_1:  948347
 alpha: 7
 c: 732904
B authenticated C
C successfully masquerade A
```

Figure 5: Output of *attack_task4.py*