University of Padua
Department of Information Engineering

Information Security Report
Team Blue

Implementation and weakness evaluatation of a challenge-response scheme for entity authentication

Authors:
Sara Bardi
Simone Favaro
Alessandro Rizzo
Enrico Tiozzo

Teacher:
Nicola Laurenti

20th December 2020

Tasks implementation

Task 1

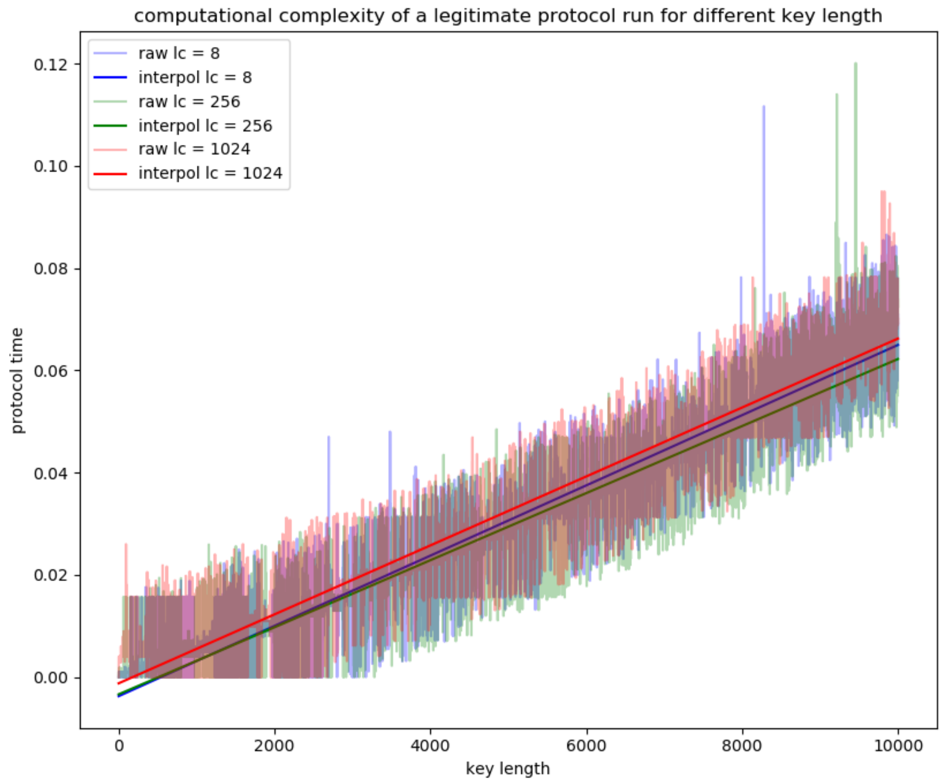
Protocol implementation

`task1.py` contains the entity authentication scheme. The main idea behind our implementation is using two different data structures in order for A and B to store the messages and the relevant variables they exchange during the communication steps. In order to accomplish that, we initialize two different python dictionaries containing the following keys: *key, challenge, n_counter, id_A, u2, r.*
Moreover, the script is design in such a way that the setup and each other protocol step is implemeted by a different method:

- `setup(n,lk)` initializes the `n` value, the key length `lk` and the id of A `id_A`, then it generates a random binary key which is stored in A's and B's dictionaries. Finally, `n` is stores in B and `id_A` in A. It outputs `id_A` and the key.
- `step1()` simulates the sending of $u_1 = id\_A$ from A to B. It outputs u_1,
- `step2(lc)` first updates the `n` value in B, then uses the initialization of the challenge length `lc` to generate a random binary challenge. Finally the method simulates the sending of $u_2 = (c, n)$ from B to A. It outputs u_2,
- `step3()` uses the util function `r_calc` which follows step 3 instructions in order to compute $u_3 = r$. Then it simulates the sending of u_3 from A to B. It outputs r.
- `step4()` let B compute $\hat{r}$ using the same `r_calc` method. It outputs the $\hat{r}$ value.

Protocol implementation

Concerning the computational complexity of the protocol implementation, the following plot represents the execution time for different values of key length (from 1 bit to 10000 bits). Each curve of the same color is composed by the raw data sampled by the `time_protocol` method (which uses the `time` function to measure the correponding executions) and the line interpolation of the raw data. This is repeated for three different values of `lc`: 8, 256 and 1024.



From the plot analysis, follows these observations:

- the curves follow a linear trend, to it is safe to say that the execution time is linear with respect to the key length.
- even if the noise is relatively high on this scale, we think that it does not depend on `lk`, but on the machine computations; and that for bigger key values it becomes more and more insignificant with respect to the execution time.
- we observe that for similar `lc` values the interpolation lines are almost congruent (actually, because of the randomness nature and the bit impact of the noise, the `lc`=8 and `lc`=256 curves have the same biases and the former has even a steeper slope). However, we notice that for `lc`=1024, the bias is higher than the other two curves: this means that even the `lc` has influence on the execution time, since all time measures are higher than the corresponding ones with a less `lc` value.

Task 2 - First attack implementation

Task 3 - Second attack implementation