PV204 Security Technologies

2nd assignment – Square and Multiply (blinded) – Side Channel Attack

Supervisor: Dr.Petr Švenda

Student: Pedro Gomes, 490830

Due date: 21th March

Spring Semester 2019

FACULTY OF INFORMATICS
Masaryk University

Histograms, 4 scenarios and discussions:

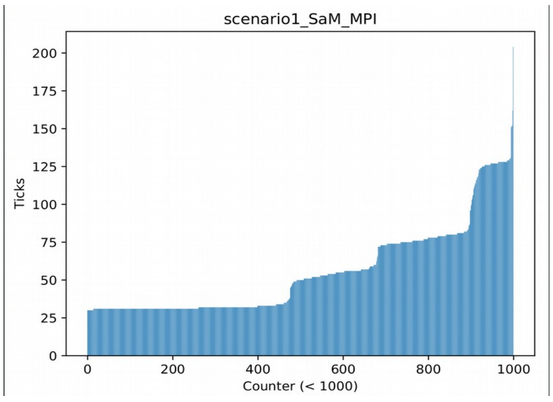


Illustration 1: scenario1 without blind

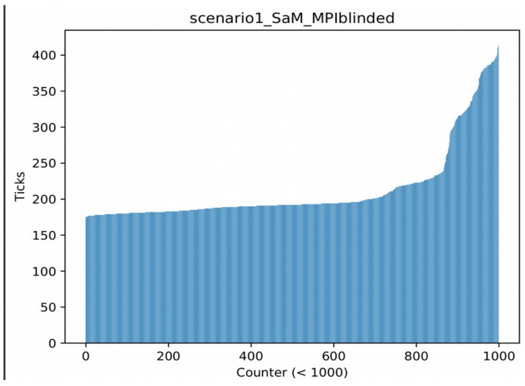


Illustration 2: scenario1 blinded

We clearly see that the blinded version requires a lot more CPU ticks in order to be achieved. This is the case for all subsequent scenarios, and the logical explanation is that the blinded version requires more data and computations, including but not limited to: Generation of a random value ***r (r^ -1 invert mod N must hold)*** and decryption of the blinded message. The blinding operation, specially the generation of the random value r makes sure that an adversary will not be capable of fully make use of the time taken to decrypt the blinded message(after the standard RSA decryption), as he does NOT know what the random value r is. This solves the problem raised in slide 8[1] of seminar2, because it is not possible to use time measures to know if the j-th bit of the private exponent is either a “1” or a “0” .

For the sake of space saving, the plots of the scenarios 2 and 3 are not going to be posted here, as the results and conclusions are pretty much similar to what was already mentioned in the previous paragraph. Nevertheless, they can be found at [2].

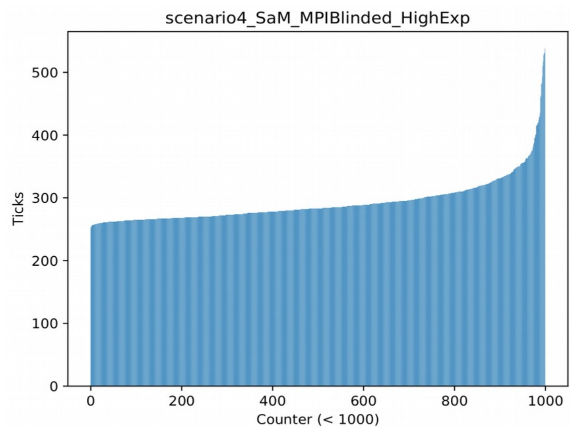


Illustration 3: Scenario 4 same exponent High HW

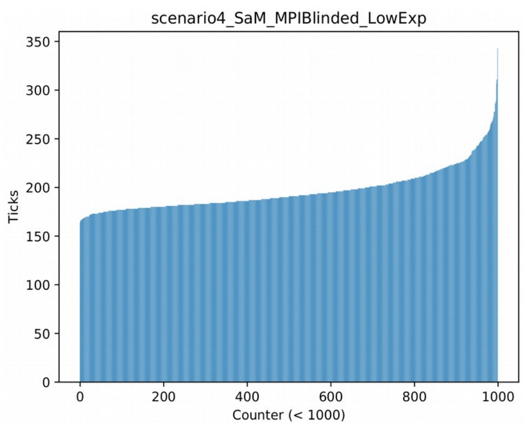


Illustration 4: Scenario 4 same exponent Low HW

Initially, I was not sure to which exponent: Private or public, apply the scenario 4. I finished to apply it to the public exponent as it dictates how efficient the encryption of some data X will be.

Surprisingly, we can see that in illustration 4 the ticks are not too far away from the scenario 1, in illustration 2. When it comes to the High hamming weight in illustration 3, we clearly see that ticks relatively increase. Out of all the plots I did, illustration 3 is the one the highest number of ticks.

According to this description[3] of RSA keys generation, a public exponent with a small HW should provide a more efficient encryption. After reading[4], the reason for this is because: The number of multiplications required for an exponent, in this case the public one, is: ***log2 e + weight(e)*** *;* hence the smaller the weight, the more efficient the computation is.

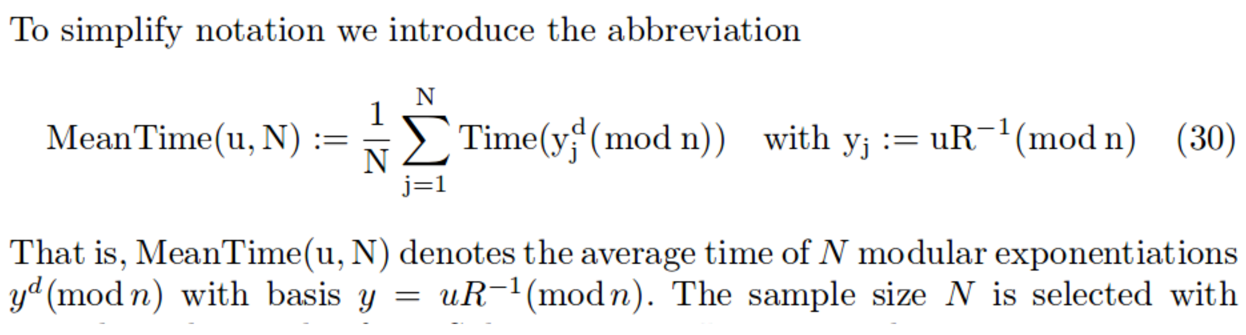
We can say that: Less expensive computations done with an exponent with low HW are more efficient than computations done with an exponent with high HW for this specific scenario. Please note that [3] mentions that a very low HW on the exponent has been proved to be less secure under certain settings.

One vector of attack that can be exploited for the blinded version is try to guess the random value r.

The randomness of the algorithm used to generate the random value r is crucial. If the algorithm used has a flaw which can be exploited to quickly guess the random value generated, it would be trivial for an attacker to guess some information out of the time it took to decrypt the blinded message, since now the random value r is known. The blind operation would loose all its value.

Another attack vector that can probably be more difficult to implement, but still possible, is to sniff for the generated random value r. If the attacker is able to access the specific part of memory where r is saved, he might just see it and use it, again, to make time analysis for the blinded message.

It is worth saying that I found several papers on the web that state and mathematically show that blinding the message is NOT enough. Unfortunetely, the explanations on almost all of them require a great level of mathematical knowledge that I do not have yet. Nevertheless, I was able to understand out of one of them[4] that:



To simplify notation we introduce the abbreviation

$$\mathrm{MeanTime}(\mathrm{u},\mathrm{N}) := \frac{1}{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{Time}(\mathrm{y}_{\mathrm{j}}^{\mathrm{d}}(\mathrm{mod}\,\mathrm{n})) \quad \text{with } \mathrm{y}_{\mathrm{j}} := \mathrm{u}\mathrm{R}^{-1}(\mathrm{mod}\,\mathrm{n}) \qquad (30)$$

That is, MeanTime(u, N) denotes the average time of N modular exponentiations $y^d (\mathrm{mod}\, n)$ with basis $y = uR^{-1} (\mathrm{mod}\, n)$. The sample size N is selected with

Means that, one possible attack, according to[4] is based on the mean of the time of the sum of all modular exponentiations.

References:
[1] : Dr. Svenda Petr, CroCS_MUNI, slides of seminar2, PV204_03_SideChannelAttacks_LABS_2019.
[2] : Pedro Gomes, plots generated for the different scenarios of seminar2, https://github.com/OblackatO/PV204-Security-Technologies/tree/master/Seminar2/plots
[3] : Unknow author, Wikipedia, description of what Hamming weight, and some RSA references of its usage, https://en.wikipedia.org/wiki/Hamming_weight
[4] : Werner Schindler, Bundesamt f ̈ur Sicherheit in der Informationstechnik (BSI)Godesberger Allee 185–189, Timing attack, RSA, CRT, exponent blinding, https://eprint.iacr.org/2014/869.pdf