Machine learning techniques to break AES

Leonardo Corsini

Side-Channel Attack (SCA)

- Hardware implementation of cryptographic algorithms present vulnerabilities.
- High risk scenario: Smart cards, IoT and medical devices.
- Modern devices implement security measures (masking, random jitter...).
- Artificial Intelligence is needed to find the key.

Project intent

- Perform a Side-Channel Attack against a hardware implementation of AES protected with masking.
- Retrieve one byte of the key, analyzing current leakage during the S-BOX execution.
- Compare classic Machine Learning (Random Forest and Gaussian Naive Bayes) with Deep Learning (Multi-layer perceptron) in this task.

Data Acquisition

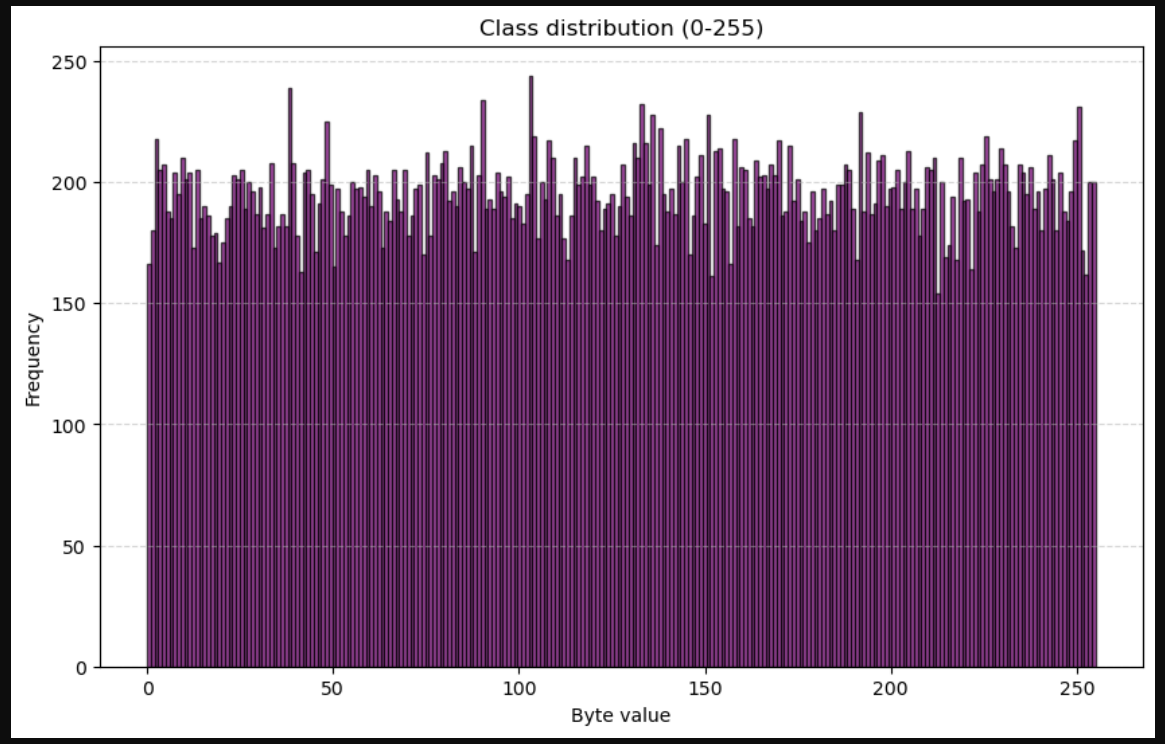
ASCAD Dataset:

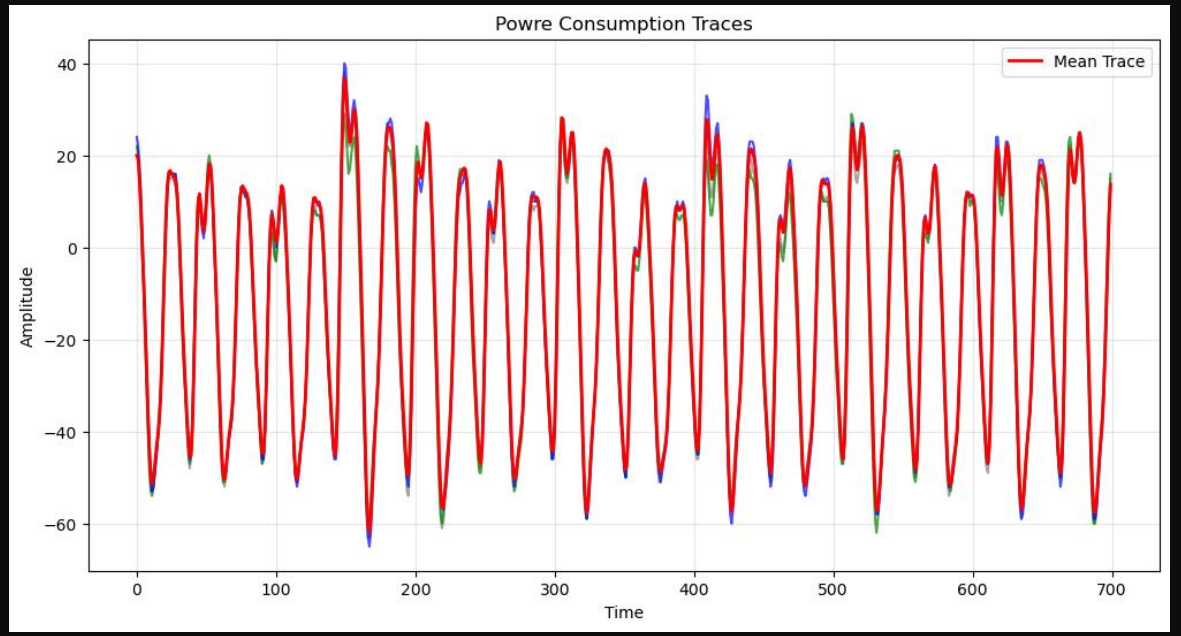
A public reference dataset containing electromagnetic measurements extracted from an 8-bit microcontroller executing a masked AES-128 algorithm.

- 50.000 training traces
- 10.000 test traces
- 700 features per trace
- Fixed encryption key
- Traces protected by masking
- Randomly different mask every trace

Data Exploration

- Quite balanced distribution of classes
- Aligned power traces with different amplitudes
- No missing or infinite values



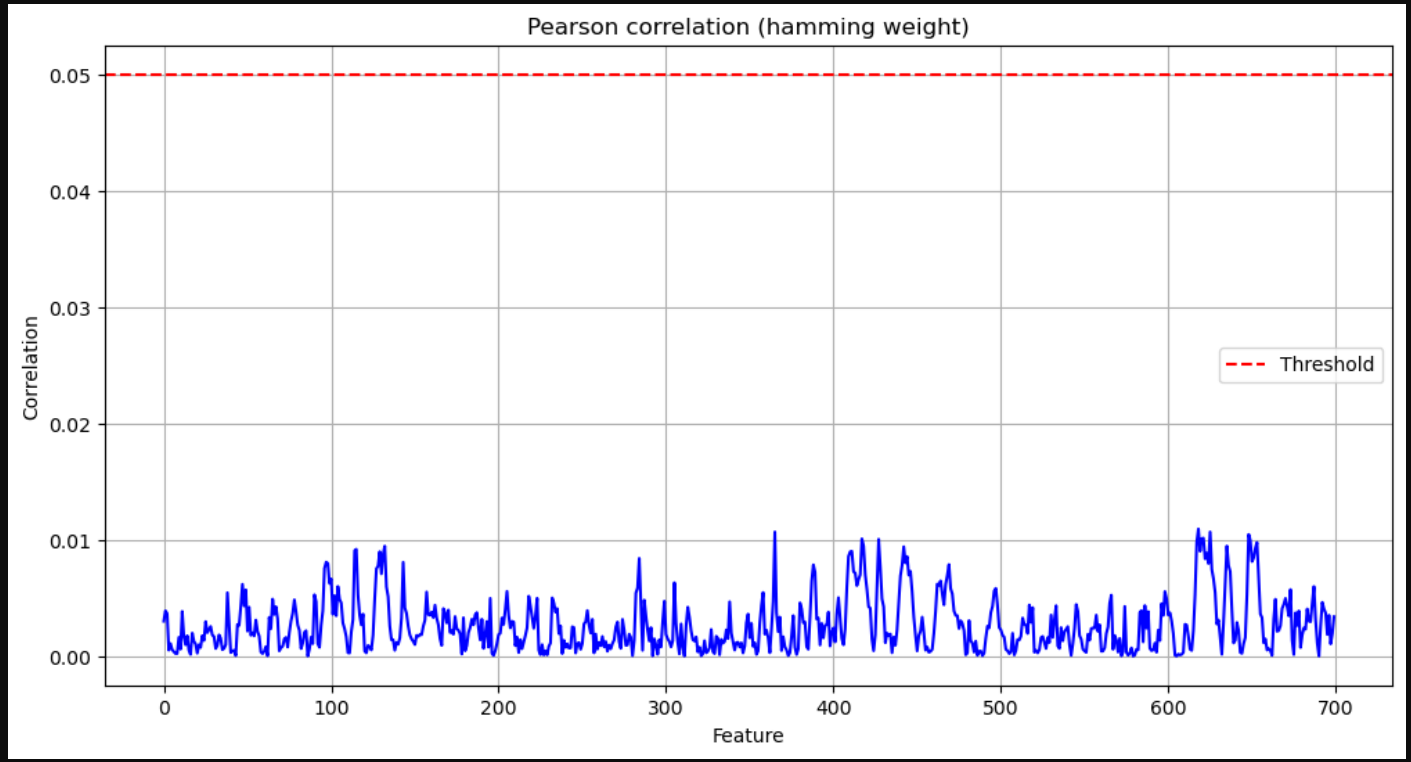


Deal With Masking

- Signal *s* in the dataset is protected with masking *m*:

  $masked\_signal = s \oplus m$

- Power consumption modeled with Hamming Weight (HW) of the unmasked signal
- Pearson Correlation computed on all 700 features
- No correlation peak on first-order leakage



Deal With Masking

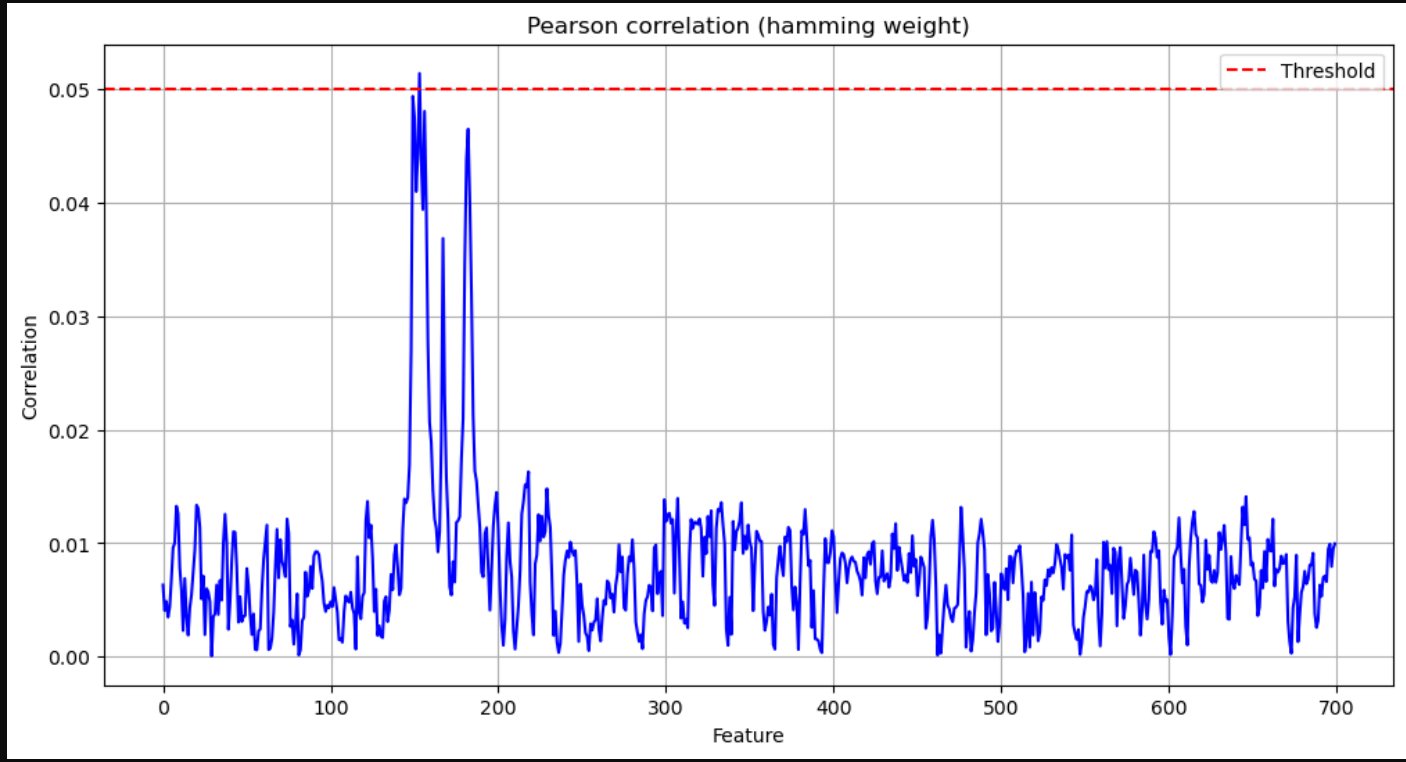
- 2nd order preprocessing to bypass masking:
- Mean of each feature *j*:

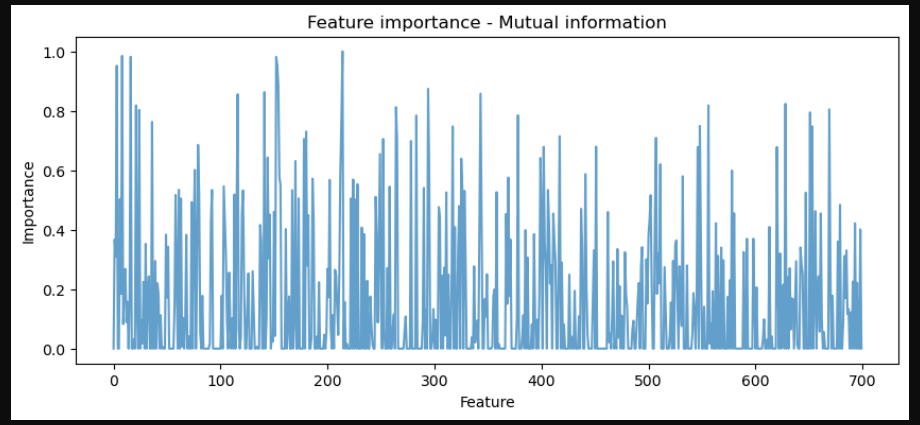
$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

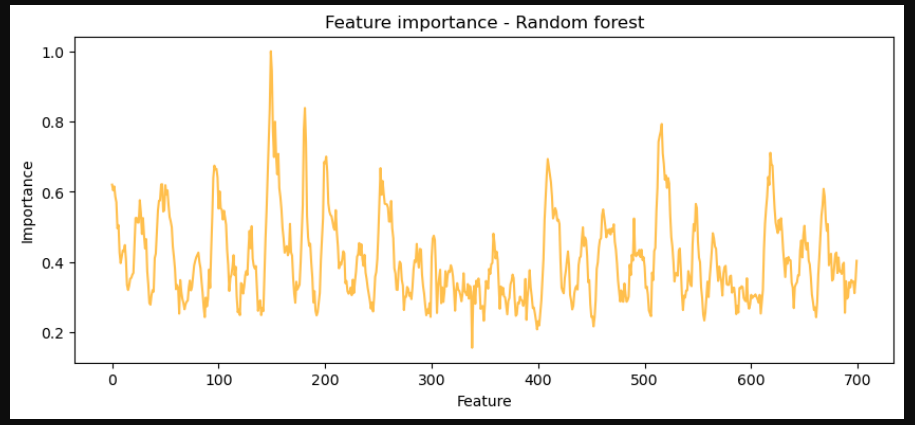
- Transformation applied to each sample *j* of trace *i* in both training and attack sets

$$x'_{i,j} = (x_{i,j} - \mu_j)^2$$

- Correlation is now visible







Dimensionality Reduction

- Two different techniques of feature selection were tested: Mutual information and Random forest.
- Both chosen for their ability to find non-linear patterns.
- 100 best features selected for the training of a Random forest model.
- 20 best features selected for the training of a Gaussian Naïve Bayes model.

Key Rank and Success Rate

- Accuracy is not enough to evaluate the correctness of the attack:
- Key Rank:

$$S_t(k) = \sum_{i=1}^{t} \log\big(P(k \mid x_i)\big)$$

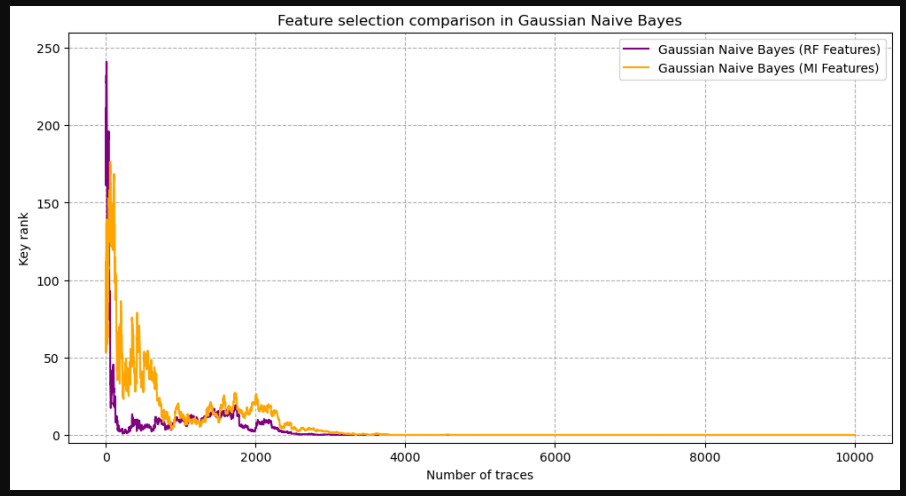
$$Rank_t(k^*) = |\{k \in \mathcal{K} \mid S_t(k) > S_t(k^*)\}|$$

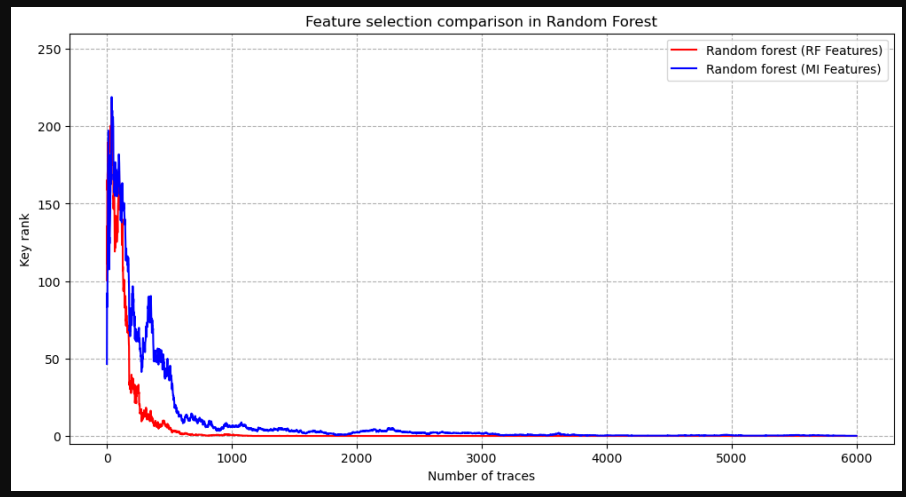
- Success Rate:

$$SR_t = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\{Rank_{t,i}(k^*)=0\}}$$

Feature selection and ML models comparisons

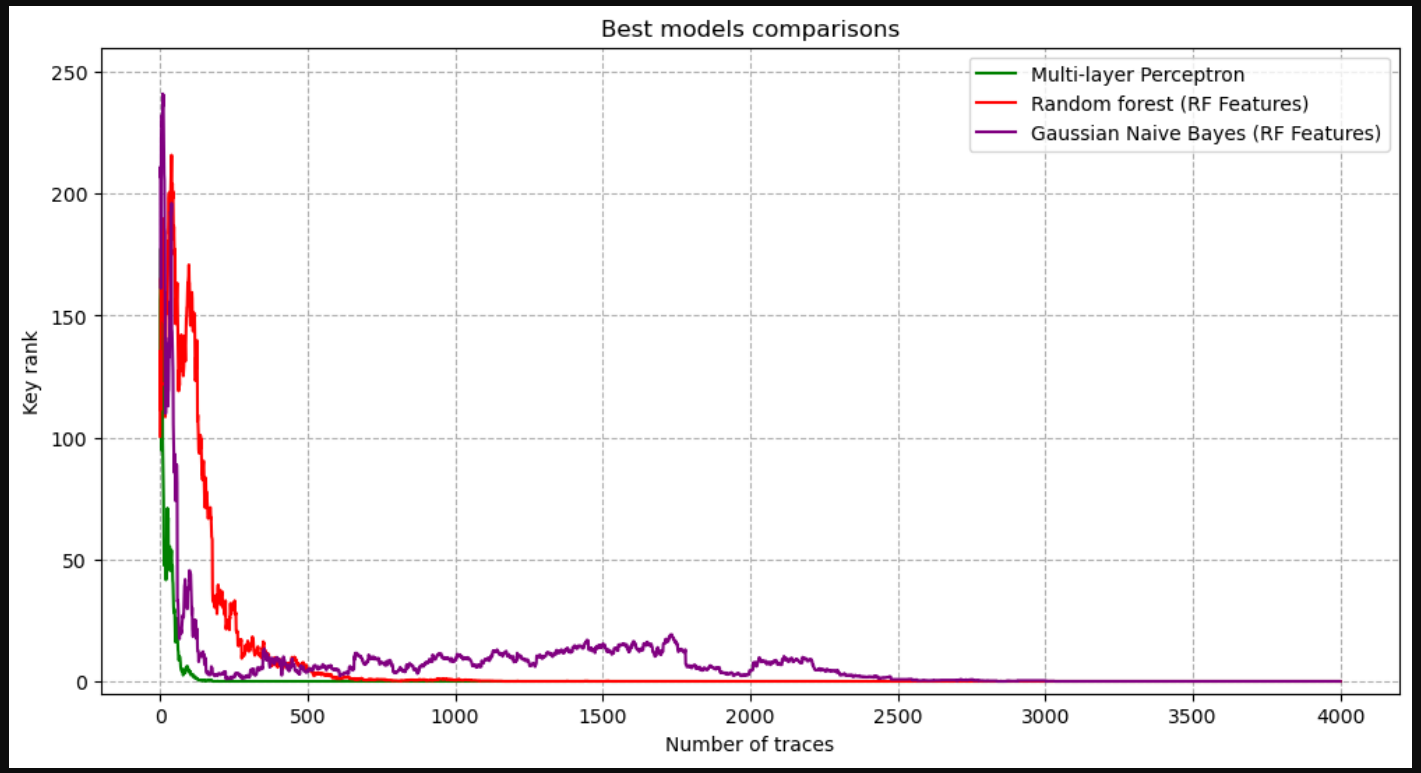
- Mutual information feature selection is slower to reach a stable *rank 0* compared to Random forest, but still allows retrieving the key.
- In Gaussian Naïve Bayes pipeline, PCA was used after feature selection. This aligns with the assumption of independent features, leading to a performance boost.
- Random forest was opportunely pre-pruned to combat overfitting as much as possible.





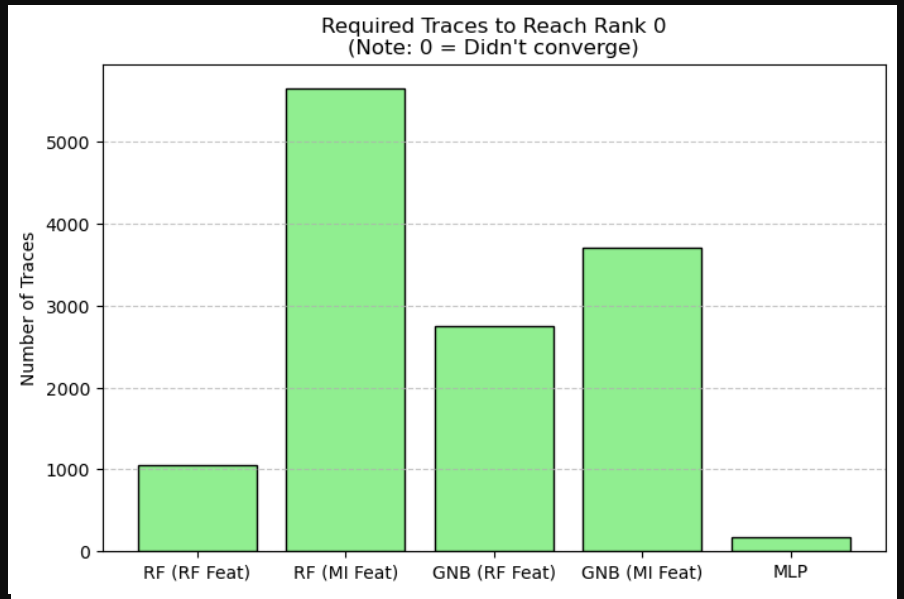
Model comparisons

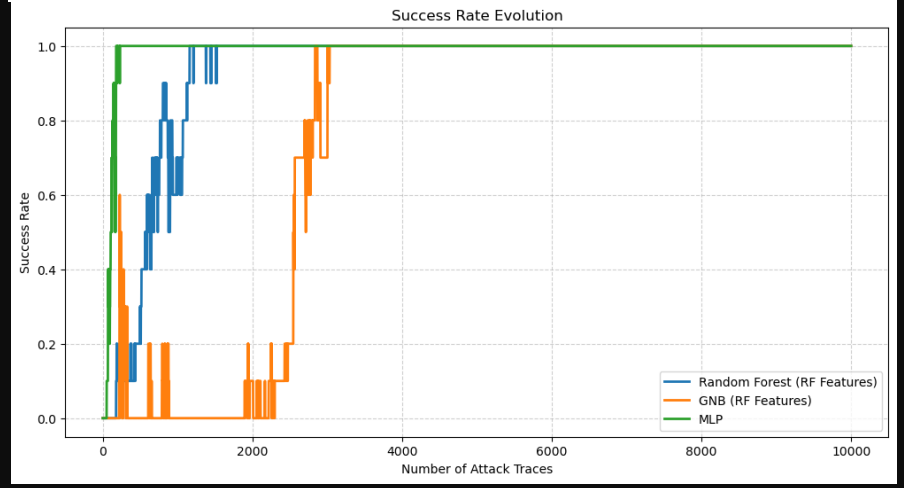
- Multi-Layer Perceptron was trained to be compared with classical Machine Learning approaches.
- No preprocessing operation was applied.
- This implementation can retrieve the Key in 100-200 traces on average.
- Obtained results of this model are comparable with the literature.



Final considerations

- Cross validation used to evaluate models, but test implemented with the test traces
- Best obtained implementation of Random forest and Gaussian Naïve Bayes were able to reach a stable rank 0 with ~ 1000 traces and ~ 2700 on average.
- Random forest feature selection was the best.
- Machine Learning models are anyway outperformed by Deep-Learning in this scenario.
- Accuracy is <1% due to the very low SNR, but the probability to find the key in one guess is still higher than random guessing (≈39%).





Model	Top-1 Acc (%)	Top-5 Acc (%)
RF (RF features)	0.40	2.17
GNB (RF features)	0.41	2.11
GNB (MI features)	0.46	2.11
RF (MI features)	0.45	2.14
MLP	0.71	3.40

Confusion matrices and learning curves