Hacking AES

Timothy Chong, Kostis Kaffes
Stanford University
Email: {ctimothy, kkaffes}@stanford.edu

Problem Overview

AES[1] is an encryption specification used in a large variety of applications from encrypting classified documents in government agencies to allowing secure data transfer to everyday transfer protocols, such as HTTPS and FTPS.

In this work, we explore machine learning techniques towards side channel analysis using power-trace to recover AES-128 keys. These techniques include multinomial Gaussian distribution modeling[3] and Support Vector Machine[4].

AES-128

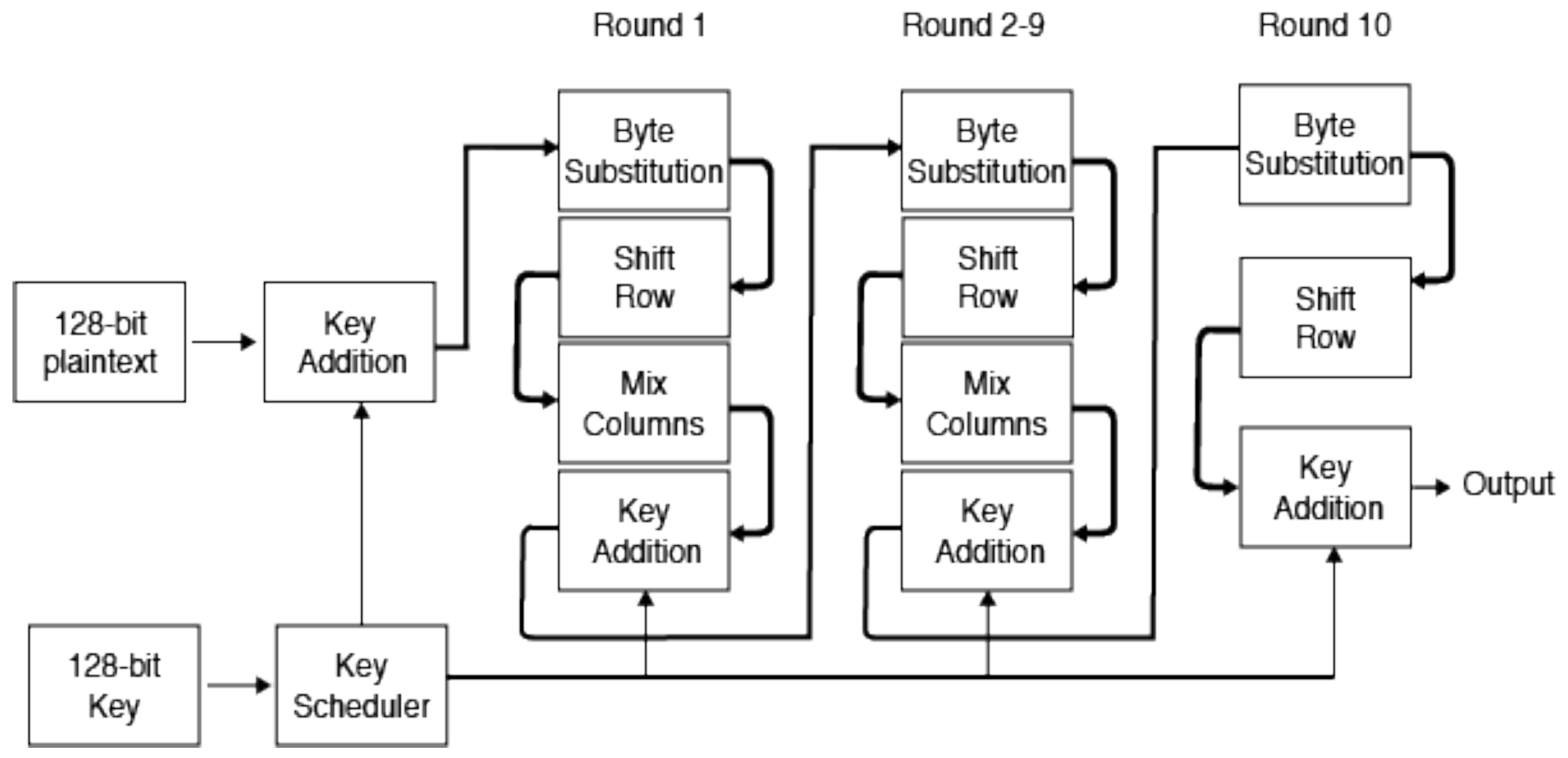

Fig. 1. AES-128 algorithm flow

10 Rounds of encryptions with operations:

- Key addition
- Byte substitution
- Row shifting
- Column mixing

Inputs:

- 128-bit key
- 128-bit plain text

Output:

- 128-bit cipher text

Data set

Our data set contains 20k power traces collected from a hardware security module while it is actively encrypting. Each power trace refers to a different plain text input but to an identical key. We divided our input set into a 17.5k training set and a 2.5k testing set.

Goal and Assumption

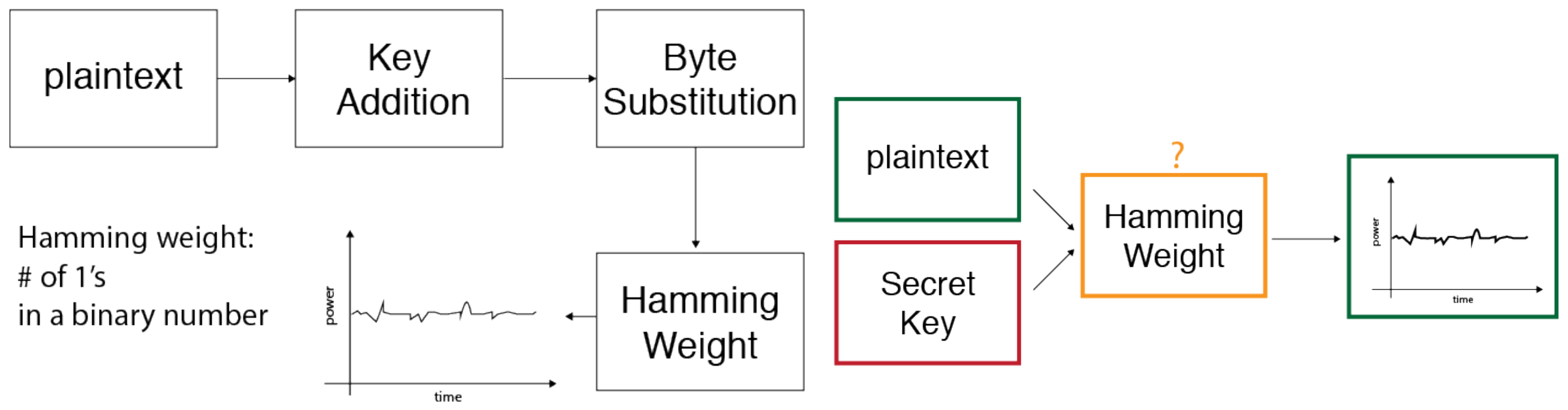

Fig. 2 We assume a relationship between the hamming weight of the output from the first key substitution unit and the power trace. If we can predict this hamming weight, we can predict the key of a system, given power traces and plain texts. There are only 9 possible hamming weights for an 8 bit sub-key.

Template Attack

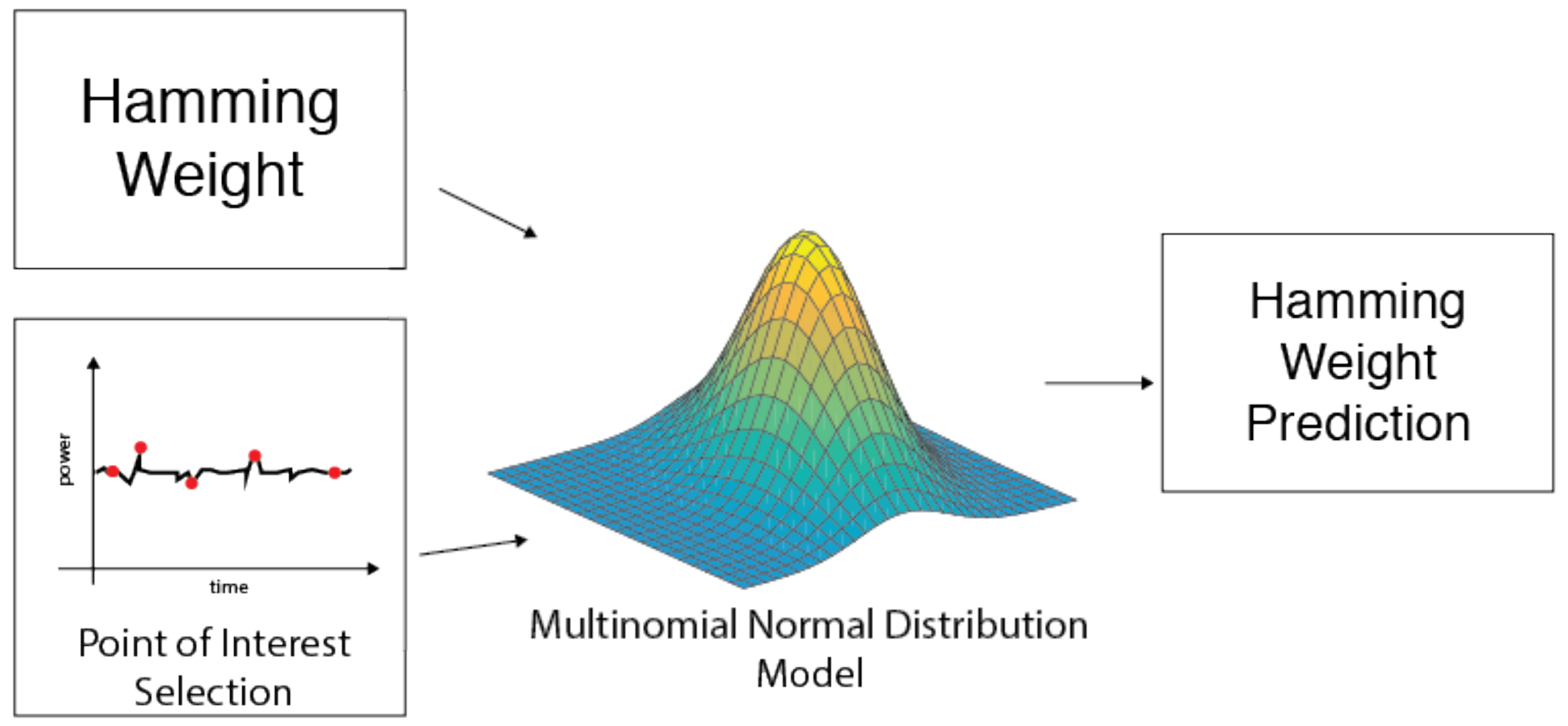
Hamming Weight
power
time
Point of Interest Selection
Multinomial Normal Distribution Model
Hamming Weight Prediction

Fig. 3. Template attack flow

Point of interest selection:
Method 1: Find points whose power values vary the most across hamming weight group pairs.
Method 2: Points with highest correlation coefficient between power values and hamming weight.

Support Vector Machine

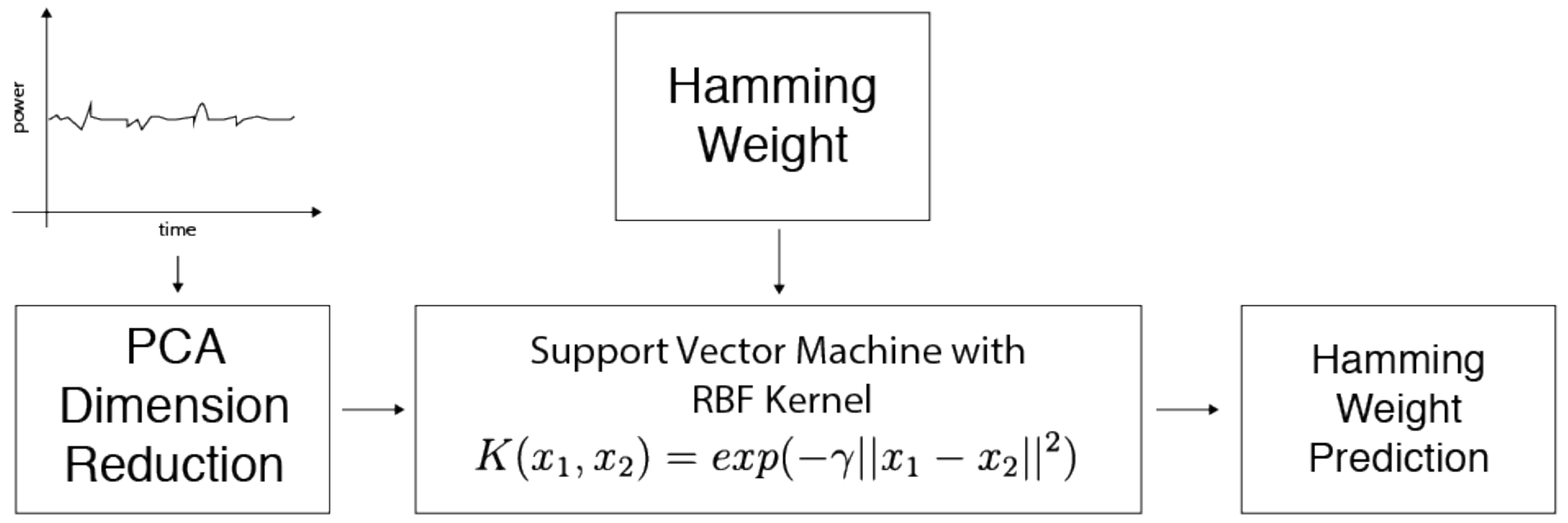
power
time
Hamming Weight
PCA Dimension Reduction
Support Vector Machine with RBF Kernel
$K(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2)$
Hamming Weight Prediction

Fig. 4. SVM attack flow

Balancing skewed hamming weight distribution:
The hamming weights of our data sets are not equally distributed because there is an inherent skewing with the hamming distribution for binary numbers given a set length. To address that, we assign weights inversely proportional to class frequencies in the input data.

Experimental Results

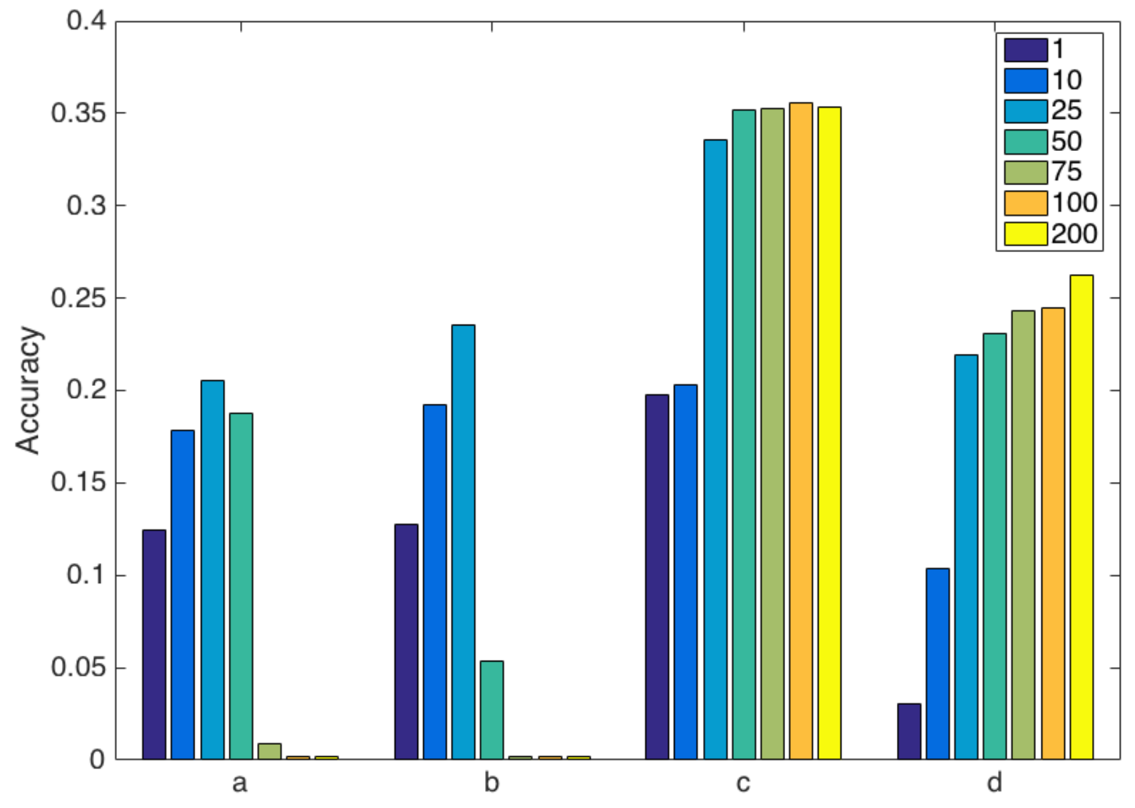

Fig. 5. Accuracy of each method with different numbers of POI's and PCA components.

a) Template attack method 1 POI's.
b) Template attack with method 2 POI's.
c) SVM without class balancing
d) SVM with class balancing

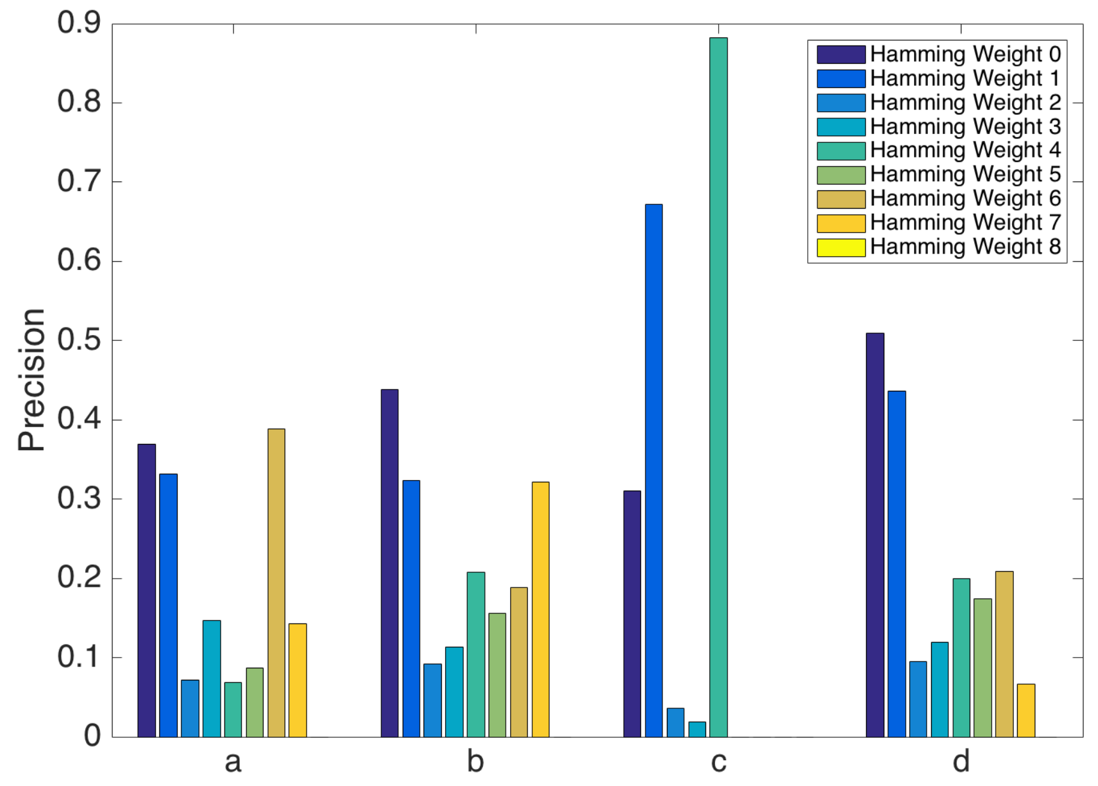

Fig. 6. Precision rate of each hamming weight category.

a) Template attack with 25 POI chosen with method 1.
b) Template attack with 25 POI's chosen with method 2.
c) SVM without class balancing
d) SVM with class balancing

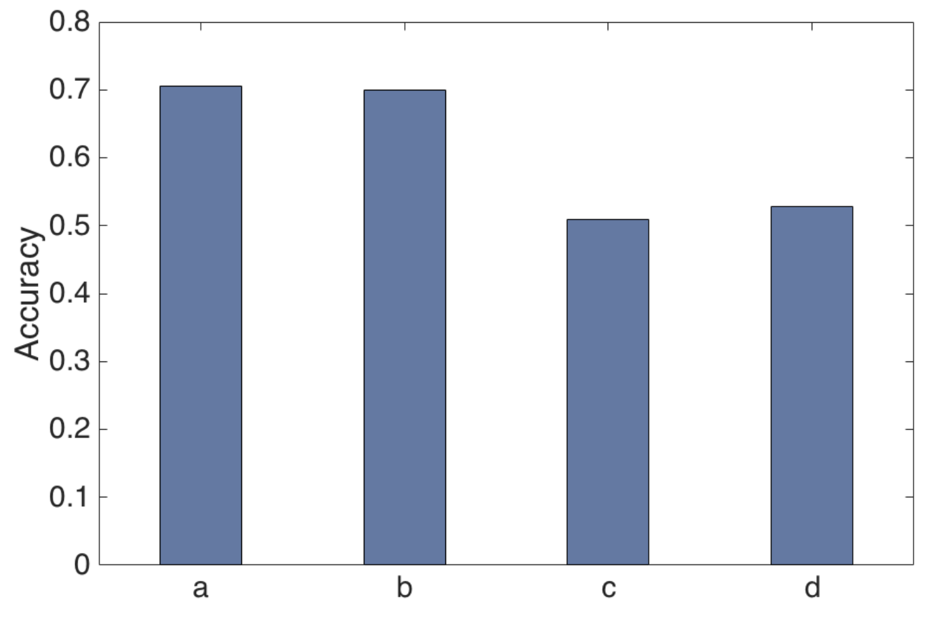

Fig. 7. Accuracy of a classifier for the first bit of the output of the first phase.

a) SVM with RBF kernel
b) SVM with RBF kernel and load balancing
c) SVM with linear kernel
d) SVM with linear kernel and load balancing.

Conclusion and Future work

Our work shows it is not a trivial task to predict the hamming weight using a single attack trace. It is worth noting that the inherent skewing of the hamming weight distribution makes it harder to sustain high accuracy while taking into account low frequency hamming weights. The one-bit prediction could alleviate this problem because it does not have to deal with hamming weights.
Several possible improvements include:

- Attack phase isolation on power trace.
- Using Random Forest and/or Neural networks to improve prediction accuracy

References

[1] J. Daemen and V. Rijmen, The Design of Rijndael. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.
[2] I. Jolliffe, Principal Component Analysis. Springer Verlag, 1986.
[3] S.Chari,J.R.Rao,andP.Rohatgi,TemplateAttacks. Berlin,Heidelberg:Springer Berlin Heidelberg, 2003, pp. 13–28.
[4] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018.
[5] H. He, J. Jaffe, and L. Zou, "Side channel cryptanalysis using machine learning," CS229 Project.