A Machine Learning Method For Sensor Authentication Using Hidden Markov Models

Julian Murphy*†, Gareth Howells*
*School of Engineering and Digital Arts
University of Kent
Canterbury
United Kingdom (UK)
Email: j.murphy-2060@kent.ac.uk, w.g.j.howells@kent.ac.uk

Klaus D McDonald-Maier†
†School of Computer Science and Electronic Engineering
University of Essex
Colchester
United Kingdom (UK)
Email: kdm@essex.ac.uk

***Abstract*—A machine learning method for sensor based authentication is presented. It exploits hidden markov models to generate stable and synthetic probability density functions from variant sensor data. The principle, and novelty, of the new method are presented in detail together with a statistical evaluation. The results show a marked improvement in stability through the use of hidden markov models.**

I. Introduction

Recent security trends have led to a need to be able to authenticate and identify correctly embedded devices used in the Internet-of-Things (IoT). Such devices typically feature a 32-bit processor, multiple sensors and a moderate level of software intelligence in-built. While authentication is achieved by extracting a stable and predictable sequence of n ID bits (e.g., 128-bits) from the device.

A n-bit ID can be formed in two ways:

1) from outputs of dedicated integrated circuits (ICs) or from on-chip hardware macro blocks; or,
2) by inferring it from data extracted from the sensors, memory and processor, which contains hidden information local and unique to that device.

We presented a solution to the first approach (bullet point 1 above) in [1]. In this work, a quaternary ID circuit for used in hardware macro blocks was discussed. And which, 1) exploits the fact lithography fabrication differences—known as process variability—cause variations in circuit-level switching threshold voltages (V_{th}) due to transistor doping imbalances; and, 2) quaternary metastability, instead of using the common approach of binary metastability. It attempted to address the fact certain bits will be unstable and different each time a n-bit ID is generated, rather than giving the same n-bit ID each time (e.g., 20 bits will be unstable for a 128-bit ID). This problem, termed stability, occurs due to environmental factors, such as temperature changes, and the natural non-linear properties of transistors.

The second approach (bullet point 2 above) to form an n-bit ID—where the authors have conducted work on this approach using probability density functions (PDFs) in [2] and termed ICMetrics—also suffers from the exact same stability problem. Therefore, knowing this, based on the previous work and after evolving a number of ideas, we have researched and developed a method to act as the basis for a new solution, and as follows.

Given stability affects the consistency of the sensor data extracted from a device, we first attempted various approaches to stabilize the data (pre-processing) before it was used as a PDF to generate an ID e.g., filtering and various DSP techniques. And, later cutting out the analysis as a PDF and instead generating some visual fingerprint using image processing and geometric algorithms to generate an ID e.g., voroni diagrams and line sweeping. This led to novel solutions, but no real or appreciable improvement in stability was observed, indeed it often worsened.

However, what we did note, which stood out when combining and permutating the previous work and the above attempts to improve stability, was that: machine generated PDFs, such as instruction cache signatures, were usually perfectly stable. This is because they are effectively synthetic PDFs, while the PDFs from sensors are naturally analogue and influenced by their environment; and obviously intuitive since one is digitally created and one is not. But it was not clear that perhaps something similar could be exploited to improve stability, and which spawned the following idea: if synthetic PDFs can be generated from sensor data, and thus making them machine generated, stability should improve (or at least by an appreciable percentage).

In researching this, we also noted a similar process, conceptually, to this exists in hardware via opto-couplers, which take in a dirty signal and infer a clean digital signal from light induced variations. This method is also often used in side-channel analysis of smartcards to crack their secure IDs from cleaned up power signals. Therefore, we have investigated methods to emulate an opto-coupler's operation with the purpose of generating synthetic PDFs from raw sensor data, and led to a method of using Hidden Markov Models (HMMs) to output synthetic PDFs as follows.

A HMM is initially trained on a data set of sensor data (e.g., 1000 samples from an ambient light sensor). Its parameters are stored and then reused to rebuild the HMM later, then a new data set of sensor data is used with it to make a set of state predictions to form a synthetic PDF. More precisely, as the new sensor data is passed through the HMM the probability

of being in a certain state changes over time—for example, moving from state 1 to state 4 to state 2—which naturally leads to a count for each state for the sensor data set, and which can be used to build a synthetic PDF. This new digital PDF can then be used in the regular ICMetrics approach to generate an ID.

We present in this paper in detail the method behind the idea, since we have not found any other works that are related. The scientific novelty is using HMMs to improve stability of IDs from synthetic PDFs.

II. Related Work

The technology and novelty of ICMetrics is that it is designed to be similar to what can be found in real life and in a human that is unique and can be used to generate digital IDs e.g., DNA, a thumb fingerprint or an eye's iris. Other technologies from the literature that work in a similar way include: Physical Unclonable Functions (PUF) [3], hard-wired digital keys [4], biometrics [5], dynamic encryption and passwords [6]. However, ICMetrics can be considered technologically more of a hybrid approach, given that it exploits data which can be extracted from hardware (or software) as might be available or best suited to the application at hand. This is then used to form PDFs whose parameters are used to form unique IDs.

In previous publications, namely [7] and [8], it was explored how viable and if at all possible it was to apply the core technology to generate stable IDs from a processors software execution signature. Here, the main idea was to exploit the program counter (PC) as the data source, given that the PC signature yields distinct PDFs from different programs, in a similar way malware analysis. The main findings of the work are presented in [9]. It can be observed of this approach, that the PDFs formed are purely digital and synthetic, and thus effectively noise free and highly stable.

Conversely, any PDF formed from reading sensor values will have a high amount of noisy and changing values simply due to environmental effects. For example, an ambient light sensor's values require light to be at exactly the same level to get repeatable readings. Therefore, a means to generate a stable synthetic from analogue data is desirable. We have not found anything to this end in the literature, only an approach to go the opposite way from PDFs to synthetic time-series instead in [10].

III. Methodology

A. Hidden Markov Models

A Hidden Markov model is a stochastic signal model which was first introduced by [11] and based on the following assumptions:

1) an observation at t was generated by a hidden state;
2) the hidden states are finite and satisfy the first-order Markov property;
3) the matrix of transition probabilities between these states is constant;
4) the observation at time t of an HMM has a certain probability distribution corresponding with one of the possible hidden states.

Although HMMs were developed in the 1960s, a maximization method was not presented until the 1970s in [12] to calibrate the model's parameters. Since, more than one observation can be generated by a hidden state the authors in [13] introduced a maximum likelihood estimation method to train HMMs with multiple observation sequences, assuming that all the observations are independent. From which two main types of hidden Markov models can be built: discrete HMMs and continuous HMMs.

The actual parameters of an HMM are the constant matrix A, the observation probability matrix B and the vector p, as follows:

$$\lambda \equiv \{A, B, p\}$$

If we have infinite symbols for each hidden state, the symbol v_k will be omitted from the model, and the conditional observation probability b_{ik} is:

$$b_{ik} = b_i(O_t) = P(O_t|q_t = S_i)$$

If the probabilities are continuously distributed, we have a continuous HMM. In this work, we assume that the observation probability is a Gaussian distribution. And, therefore, $b_i(O_t) = \mathcal{N}(O_t = v_k, \mu_i, \sigma_i)$, where μ_i and σ_i are the mean and variance of the distribution corresponding to the state S_i, respectively. Such that the parameters of an HMM are:

$$\lambda \equiv \{A, \mu, \sigma, p\}$$

where μ and σ are vectors of means and variances of the Gaussian distributions.

Three main questions can be answered when applying a HMM to solve a real-world problem as follows:

1) Given the observation data $O = O_t, t = 1, 2, ..., T$ and the model parameters $\lambda = A, B, p$, calculate the probability of observations, $P(O|\lambda)$.
2) Given the observation data $O = O_t, t = 1, 2, ..., T$ and the model parameters $\lambda = A, B, p$, find the "best fit" state sequence $Q = q_1, q_2, ..., q_T$ of the observation sequence.
3) Given the observation sequence $O = O_t, t = 1, 2, ..., T$, calibrate HMM's parameters, $\lambda = A, B, p$.

These problems can be solved by using the main HMM algorithms as below:

1) Find the probability of observations: Forward or backward algorithm.
2) Find the "bet fit" hidden states of observations: Viterbi algorithm.
3) Calibrate parameters for the model: Baum–Welch algorithm.

The most important of the HMM's algorithms is the Baum–Welch algorithm, which calibrates the parameters of a HMM given the observation data.

Generating PDFs from HMMs

HMMs have been widely used in mathematics to predict economic cycles and for speech/text recognition. However, in this paper we propose a method of using them with the purpose of generating stable PDFs by emulating the operation of opto-couplers.

In this method, we start by training a HMM using a raw sensor data set of a fixed length for a given sensor:

$$O = O_t, t = 1, 2, ..., T$$

where where O_t is the sensor sample at time t.

While the number of HMM states used should correspond to the number of bins of the raw training data set's PDF. We also assume that the distribution corresponding with each hidden state is a Gaussian distribution.

Next given a trained HMM for a sensor we simply predict the PDF given another raw data set from the same sensor. The prediction is executed by simply running the data set through the HMM and recording the predicted state changes. A synthetic PDF can then be constructed from the tally of states. If the correct sensor has been used the resultant PDF should correspond to the trained HMM PDF from which parameters can be extracted to generate an ID.

IV. Evaluation

To develop and evaluate the method we have used extensively a Arty FPGA board together with a variety of PMOD sensors to extract data from. For ease of analysis sensor data was recorded straight into a database and file storage under various conditions to exercise the full range of sensor data ranges as per a given sensor's usage. For example, an accelerometer sensor was moved through each axis and the ambient light sensors data was extracted under varying lighting conditions. This approach was taken to ease back-end analysis, rather than trying to perform it all in real-time and potentially missing points or data anomalies of interest. Once the batch sensor data sets had been collected, training of the HMMs and prediction was performed in the open-source statistical analysis software known as R.

Algorithm 1 Building a HMM model and histogram in R

```
1| dataDiffT = as.numeric( diff( dataT ) )
2| hmmT <- depmix( dataDiffT ~ 1,
     family = gaussian(), nstates = 3,
     data=data.frame(dataDiff=dataDiff) )
3| hmmfitT <- fit( hmmT, verbose = FALSE )
4| post_probsT <- posterior( hmmfitT )
5| hist( post_probsT$state )
```

Prior to settling on R various other software solutions were investigated for their HMM capabilities (such as Ruby, Python and Matlab), however the availability and open source nature of R's HMM packages proved the most reliable and easiest to work with. Various HMM solutions are available in R, so the most practical was chosen and discussed here, namely *depmixS4* with R version 3.4.4. To use any of code presented the reader will need to first install version 3.4.4 and also load depmixS4 as a R library. For the experiments relevant sensor data was simply read from file as a CSV then used with the following core code listings.

Algorithm 2 Predicting a HMM model and histogram in R

```
1| dataDiffP = as.numeric( diff( dataP ) )
2| hmmP <- depmix( dataDiffP ~ 1,
     family = gaussian(), nstates = 3,
     data=data.frame(dataDiffP=dataDiffP))
3| hmmP <- setpars(hmmP, getpars(hmmfitT))
4| hmmfitP <- fit( hmmP, verbose = FALSE )
5| post_probsP <- posterior( hmmfitP )
6| hist( post_probsP$state )
```

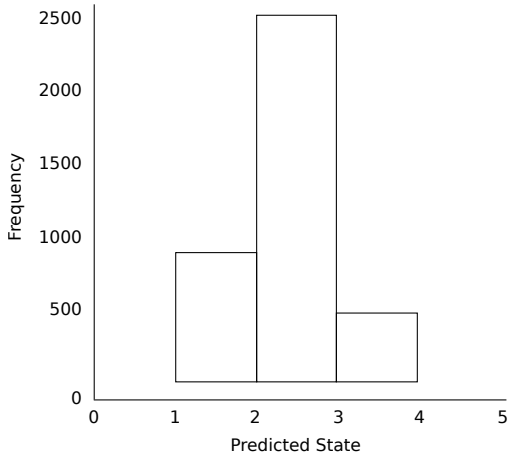

Fig. 1. Ambient sensor HMM trained PDF

The first, Algorithm 1, on line 1 takes the difference between all the samples to form a training data set named *dataDiffT*. From which a Gaussian HMM is built with three states; the state count was varied over differing ranges during the experiments, three is just a placeholder in the code. We found it is not guaranteed that a HMM can be built given the data and number of states, or even that the resultant histogram is actually useful, all input data can map to just one HMM state for instance. To address this, initializing R's seed setting, for example as *"set.seed(1)"*, to different values helps mitigate this issue and to choose states that matched the number of bins in the raw data set PDF. Next a HMM is trained and fit to the input training sensor data in line 3 and the posterior probabilities extracted in line 4 into variable *post_probsT*. Lastly, a histogram is constructed by extracting the state tally from this variable with *post_probsT$state*. All the variable names in the code are appended with letter 'T' to signify that they are training data.

Once a HMM had been trained on a sensors raw data set it was just a matter of saving the HMM parameters. They are

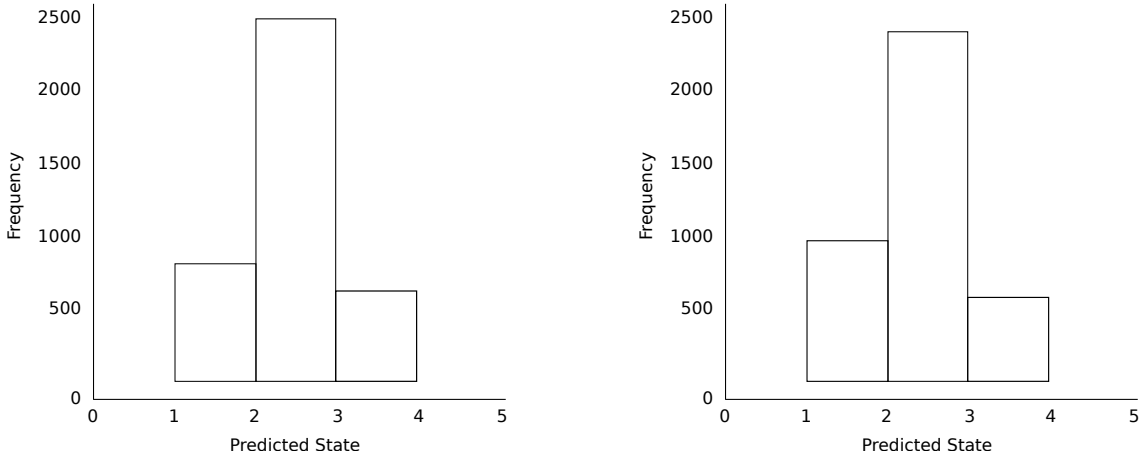

Fig. 2. Two ambient sensor HMM prediction PDFs

simply reloaded as required to make a prediction with new sensor data to output a synthetic PDF of the HMM states. The code listing to accomplish this in R is listed in Algorithm 2. The differences of the sensor data to be used for prediction are read stored into *dataDiffP* on line 1. Then a placeholder HMM is constructed in line 2, whose parameters are updated with the trained HMM in line 3. Next a HMM is fit to the data using the parameters of the trained HMM and the posterior probabilities extracted into the variable *post_probs*P in line 4. A prediction PDF is then built by extracting the states stored in variable *post_probsP$state*.

As example of the method, Figure 1 shows a trained HMM PDF for three states for an ambient sensor under room lighting conditions. Using two other data sets from the same sensor also under room lighting two synthetic PDFs constructed using the trained HMM but making a prediction is shown in Figure 2. For each of the synthetic PDFs there is a clear consistency and self-evident stability in comparison to the trained HMM PDF.

V. Conclusions

We have presented a method and its details of how to generate synthetic PDFs from raw sensor data using HMMs in an endeavor to increase stability. Initial experimental results show a marked improvement in stability. In conducting the work we also explored other structures and algorithms in machine learning as possible solutions for generating synthetic PDFs. One area, which is currently of great interest regardless, which we thought might be suitable was neural network processing. While a normal neural network is not fitting, a certain subclass of neural networks which seem to be appropriate are Restricted Boltzman Machines which we have noted can be used in a similar fashion as HMMs, and often used in speech processing and prediction. An which, will form the starting point of future work.

Acknowlegment

This work has been supported by the Euro-pean CHIST-ERA SPIRIT Project funded in the UKby the Engineering and Physical Sciences ResearchCouncil (EPSRC) [grant numbers EP/P016006/1 andEP/P015956/1].

References

[1] J. Murphy, G. Howells, and K. D. McDonald-Maier, “On Quaternary 1-of-4 ID Generator Circuits”, AHS, pp. 323-326, 2018.

[2] Y. Kovalchuk, K. D. McDonald-Maier, and G. Howells, “Overview of ICmetrics technology-security infrastructure for autonomous and intelligent healthcare system,” International Journal of u- and e-Service, Science and Technology, vol. 4, pp. 49-60, 2011.

[3] G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in 44th ACM/IEEE Design Automation Conference, 2007, pp. 9-14.

[4] H. Handschuh, G.J. Schrijen, and P. Tuyls, “Hardware Intrinsic Security from Physically Unclonable Functions,” in Towards Hardware-Intrinsic Security, A.-R. Sadeghi and D. Naccache, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 39-53.

[5] A. K. Jain, P. Flynn, and A. Ross, Handbook of Biometrics: Springer US, 2008.

[6] W. Sheng, G. Howells, M. C. Fairhurst, F. Deravi, and K. Harmer, “Consensus Fingerprint Matching with Genetically Optimised Approach,” Pattern Recognition, vol. 42, pp. 1399-1407, 2009.

[7] Y. Kovalchuk, W. G. J. Howells, H. Hu, D. Gu, and K. D. McDonald-Maier, “A practical proposal for ensuring the provenance of hardware devices and their safe operation,” in 7th IET International Conference on System Safety, incorporating the Cyber Security Conference, 2012, pp. 1-6.

[8] Y. Kovalchuk, W. G. J. Howells, H. Hu, D. Gu, and K. D. McDonald-Maier, “ICmetrics for low resource embedded systems,” in the 3rd International Conference on Emerging Security Technologies, 2012, pp. 121 - 126.

[9] Y. Kovalchuk, H. Huosheng, G. Dongbing, K. McDonald-Maier, D. Newman, S. Kelly, et al., “Investigation of Properties of ICmetrics Features,” in the 3rd International Conference on Emerging Security Technologies (EST) 2012, pp. 115-120.

[10] M. Sinhuber, E. Bodenschatz, M. Wilczek, “A probability distribution approach to synthetic turbulence time series,” APS Division of Fluid Dynamics, 2016.

[11] L. E. Baum and T. Petrie, “Statistical inference for probabilistic functions of finite state Markov chains,” The Annals of Mathematical Statistics, 37, pp. 1554–63, 1966.

[12] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,” The Annals of Mathematical Statistics, 41, pp. 164–71, 1970.

[13] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, “An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition,” The Bell System Technical Journal, 62, pp. 1035–74, 1983.