Using Deep Learning Techniques to augment Side Channel Analysis

Nimar Blume

***Abstract*—At the moment, cryptography is everywhere, allowing endpoints to transmit data securely over an insecure network. A multitude of encryption algorithms are available but even those which have no proven weakness theoretically can be attacked by attacking the algorithm's implementation. Therefore, leaking side channel data such as a chip's power consumption can be used to infer parts of or the entire secret key. Currently, the Template Attack (TA) is the most powerful side channel attack (SCA), requiring the least amount of side channel data in the attack phase. However, recently Machine Learning (ML) based TAs were shown to supersede traditional TAs in efficiency. Since Deep Learning (DL) techniques have beaten ML techniques in for example object recognition in images, this paper examines viability and performance of DL techniques in SCAs. In the end, especially Convolutional Neural Networks have proven to be more effective than traditional SCA techniques such as TAs.**

***Index Terms*—deep learning, neural network, side channel analysis, side channel attack, encryption, AES**

I. Introduction

Data encryption is commonly employed to restrict data access to selected people or endpoints. Proper encryption provides privacy, integrity and authenticity of data exchanged between two or more parties over an insecure connection. An encryption key or a pair of keys is generated with which the chosen data is encrypted, so that the data can only be read or modified by an endpoint who possesses the appropriate secret key. For symmetric encryption, where the same secret key is used to encrypt and decrypt data, the key has to be shared over a secure connection beforehand, whereas asymmetric encryption uses key pairs, one of which is used to encrypt and one is used to decrypt data. Therefore, asymmetric encryption enables two parties to exchange data securely over an insecure connection by publishing the key used for encryption and keeping the key used for decryption private. This paper will only consider attacking symmetric encryption, although techniques discussed could also partly be applied to asymmetric encryption algorithms. Encryption algorithms have been attacked in multiple ways, and attacking the algorithms implementation has proven to be effective against cryptographically sound algorithms. In the past, machine learning (ML) has been used to attack implementations as part of the profiling stage in Template attacks (subsubsection II-A4). As deep learning (DL) recently has made a comeback and consistently outperforms ML techniques in areas such as image recognition (XXXX INSERT SOURCE), applying DL techniques to breaking cryptographic algorithms is attractive to investigate. In the end, consistently with the improvement seen in image recognition, DL techniques improve ML based attacks and require fewer traces to select the correct secret key in a side channel attack on AES (page 1).

A. The Advanced Encryption Standard

While there are many encryption algorithms available, this paper focuses on the most popular symmetric block cipher: the Advanced Encryption Standard (henceforth AES) with a $128\,\mathrm{bit}$ key. The AES algorithm encrypts data in blocks of $128\,\mathrm{bit}$ or $16\,\mathrm{B}$. Should a block be smaller than $16\,\mathrm{B}$, padding will be appended until the block size reaches $16\,\mathrm{B}$. Furthermore, AES expands a single $128\,\mathrm{bit}$ key to 11 round keys which are then used in the subsequent 11 rounds to encrypt the given data. It is important to note, that a compromise of any of the 11 round keys enables an attacker to reconstruct the initial $128\,\mathrm{bit}$ key and thus decrypt the whole encrypted data set.

B. Side Channel Analysis

Side Channel Analysis (henceforth SCA) refers to a technique used to break encryption schemes by using data indirectly generated by the implementation of the cryptographic algorithm, instead of attacking the algorithm itself. Even a theoretically perfectly safe cryptographic algorithm can be subject to SCA and be broken by it. SCA uses side channels such as the power consumption of a microprocessor, electro magnetic emissions or even emitted sound to reconstruct the entire or parts of the secret key used to encrypt the data. It can be possible to infer the secret key form side channel data due to data dependant program flows. Specifically, the code contains conditional statements acting on the secret key data. For example a certain loop will only execute if the currently processed key bit is zero, therefore the power consumption of the microprocessor will measurably increase. Furthermore, power modelling is used to predict a microprocessor's power consumption based on the secret key data. Power models such as Hamming Weight (a "1" consumes more energy to be processed than a "0") or Hamming Distance (bitwise comparison resulting in the number of different bits) are popular choices. Therefore, to protect against SCA it is vital to pay attention not only on the theoretical safety of an encryption algorithm but also on its implementation.

II. State of the Art

State of the art of crypto attacks

A. The state of Side Channel Analysis

Side channel attacks were first developed in 1996 by Paul Kocher [1] in the form of timing attacks. Timing attacks can exploit an implementation's execution flow dependence on secret key data. Later, SCAs have been extended to infer secret key data by analysing a processor's power consumption and its dependence on secret key data.

1) Simple Power Analysis: Simple Power Analysis (henceforth SPA) is a SCA based on the power consumption of the target device. The target device (e.g. a smart card) performs multiple encryption or decryption operations during which the attacker measures the device's power consumption. The attacker does not know the plaintext, which makes it a ciphertext only attack (COA). Then, the attacker calculates a hypothetical power consumption for each possible key combination using for example the Hamming Weight model. Finally, he correlates the hypothetical power consumption to the measured power traces and then ranks the hypothetical keys accordingly. The correct key should now be among the top ranked hypothetical keys. However, SPA is very reliant on clean power traces. Modern computers perform many operations in parallel and thus generate a lot of noise, which is why SPA is best used on simple devices like smart cards.

2) Differential Power Analysis: Differential Power Analysis (henceforth DPA) is similar to SPA because it also tries to extract the secret key from a device by using power traces taken during a operation. However, in contrast to the SPA, multiple power traces are taken during different device states. Power traces of the target device are also taken while the device is idle to determine the background noise. The noise level is then subtracted from the power traces taken during the cryptographic operation, resulting in a more distinct signal. Still, concurrent operations running parallel to the cryptographic operations worsen the quality of the power traces, but that can be compensated by taking a large number of traces and averaging over them. Thus, DPA can also be used on more complex devices than smart cards.

3) High-order Differential Power Analysis: High-order Differential Power Analysis (henceforth HO-DPA) is similar to DPA but additionally, further power traces are taken at multiple steps of the cryptographic operation. HO-DPA allows to attack implementations using masking to hide cryptographic operations, by taking power traces at the mask generation thereby demasking the final encryption operation. However, due to taking many different traces and correlating them with one another, HO-DPAs are highly complex and not used when a DPA or a SPA could also be used to mount a viable attack on an implementation.

4) Template Attacks: A Template Attack is a very powerful, targeted attack on an implementation of a cryptographic algorithm. The premise is, that the attacker has restricted access to the target machine, but ubiquitous access to similar machines containing a varying secret key. The attacker can now execute many encryption operations while varying the secret key and recording power traces. The goal is to acquire a large labelled data set, which maps power consumption to key bits on the specific type of machine. Afterwards, the attacker trains a machine learning model using the previously acquired labelled dataset. For example, a Support Vector Machine (SVM) is trained to rank key hypotheses given power traces as data input. Finally, the attacker obtains a small number of power traces from the target machine containing the secret key during an encryption or decryption operation and feeds the power trace along with the key hypotheses into the SVM for analysis. The SVM will now rank the potentially correct key the highest, resulting in a successful attack.

III. Theory

A. Motivaiton for using Deep Learning over Machine Learning

DL techniques differentiate themselves from ML techniques in that ML relies on features selected by the attacker manually, for example power traces are correlated with hypothetical key values and subsequently points of interest (POIs) chosen based on the correlation. Using DL techniques, the input consists of raw data and the NN chooses the POIs itself. In fact, applying a principal component analysis before feeding the data into the NN for dimensionality reduction reduces the NN's effectiveness as shown in [2, p. 15] due to possibly removing vital information to achieve lower dimensional data.

B. Multi layer perceptron

A Multi layer perceptron (MLP) is a representation of a single neuron used in computer science. It consists of one or multiple inputs which are multiplied by respective weights and then fed into an activation function. The activation function then determines the output of a single MLP. A popular choice for an activation function is the rectified linear unit (ReLU): $f(x) = max(0, x)$. When training MLPs, the output is compared to the desired output and the error is calculated: $MLPERRORFORMULAR$. Using the gradient descend algorithm, the weights are then adjusted step by step to minimise the error.

C. Convolutional Neural Networks

Convolutional Neural Networks (henceforth CNN) are Neural Networks (NN) which have recently risen in popularity due to their strength in classifying images. They are good at extracting features from input data represented in a 2nd order tensor invariant from the features scale, rotation or vicinity. Feature maps are created from the input through stacked convolutional layers until a reasonably small and thus dense feature map is created. The extracted feature maps do not represent features which commonly are extracted manually.

1) Convolutional layer: The Convolutional layer of a CNN is responsible for extracting features from the inputs. A feature map is extracted from the input data by convoluting a filter tensor with the input tensor. The filter is a 2nd order tensor of a smaller size than the input and usually initialised with small random variables which are adjusted in the training phase. As displayed in XXXX TODO. After one convolution the output size decreases but still represents the same information, thus extracting features, a feature map.

2) Fully connected layer: A fully connected layer (FCL) is a layer of multiple Multi layer perceptrons (MLP, subsection III-B) of which each is connected to every MLP in the subsequent layer. FCL are commonly used as output layer in a CNN because it is very easy to tune it to the desired number of outputs per calculation cycle.

3) Pooling layer: "Downsampling", Max Pooling

4) Normalisation layer: "Softmax" A normalisation layer is used in a CNN to

D. *Stacked Auto-Encoders*

Stacked Auto-Encoders (AE) consist of an encoder layer followed by multiple FCL (subsubsection III-C2) and at the end feature a decoder layer. During training the goal of an AE is to reproduce the given input at the output. Its weights are then trained by gradually adding noise to the input data while the network should still output the original, denoised input. The FCLs are generally sized smaller than the number of inputs as to extract a more compact feature set from which the input data can then be recreated. Finally, after training multiple encoding layers, the decoding layer at the end is removed and the trained encoding layers are stacked onto one another to produce a NN.

E. *Recurrent Neural Networks*

Good for for data which has connections to itself in i.e. different points in time.

F. *Long and Short Term Memory (RNN)*

G. *Using Deep Learning to improve Side Channel Analysis*

Deep Learning assists Side Channel Analysis in that it provides a new method of profiling, and subsequently a new engine for attacking. DL techniques can be used in Template Attacks (subsubsection II-A4) in the profiling phase instead of the gaussian assumption of the leakage. A NN is then trained instead using the available labelled data set of power traces with ZUGEHÖRIGEN key data. First, power traces are recorded during cryptographic operations which use the key data. In this case, 1000 piece traces per key byte are recorded.

Afterwards, the Side Channel Attack itself is performed in the same manner as a Template Attack subsubsection II-A4. The trained NN can be of different types as discussed above and

The output of a NN in assisting SCA is a vector assigning a probability to each key hypothesis.

Regular research papers need at least two additional sections here. One section for contributions and methods and one section for the results. For seminar papers these sections can be omitted.

IV. Experimental results

Deep Learning (DL) techniques can improve SCA significantly if employed correctly. It is however essential to pick the appropriate DL technique for the working data set to get the best or even a better result than without using DL techniques. Show the results from the paper which covered the same topic.

V. Conclusion

Put the conclusions of the work here. The conclusion is like the abstract with an additional discussion of open points.

References

[1] "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," Cryptography Research, Inc., 607 Market Street, 5th Floor, San Francisco, CA 94105, USA., Tech. Rep., 1996.

[2] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," *IACR Cryptology ePrint Archive*, vol. 2016, p. 921, 2016. [Online]. Available: http://eprint.iacr.org/2016/921