Machine Learning for Side Channel Attacks

Literature Review and Project Specification

Abstract: The rise of fledgling technologies such as cloud computing and the Internet of Things and the proliferation of cryptographic algorithms and protocols has raised concerns around the security of these; notably that side channel attacks would be particularly effective against poorly implemented cryptography, potentially affecting a myriad of systems. Side channel attacks are a class of attacks where a third party can extract extra information and secrets from the implementation details of an algorithm or protocol, as opposed to finding flaws in the algorithm itself (as is the case in traditional cryptanalysis). Examples include timing and power monitoring attacks against cryptosystems, many of which have been shown to be effective in a variety of contexts and environments. Machine learning is the process of using computers to learn from data, and has strong applications to side channel attacks for automating analysis and increasing effectiveness compared with previous techniques. This paper covers the uses of machine learning in side channel attacks against cryptosystems and proposes a project to demonstrate their efficacy.

I certify that all material in this dissertation which is not my own work has been identified.

1 Introduction and Motivation

The rapid expansion of the internet, in terms of users, devices and applications, has led to the necessary adoption of modern cryptography: understanding how data can be securely transmitted such that only the intended recipient(s) can understand the information being sent [12]. This field is essential to ensuring that communications over untrusted networks (such as the internet) are secure and trustworthy, and a significant portion of research is dedicated to analysing cryptographic systems and algorithms to discover how they can be broken. The intention is to understand their flaws and weaknesses, and use these to design systems which offer better security and are more effective; the use of offensive techniques to evaluate security is commonplace in the field of computer security and is considered best practice to ensure that an attacker cannot violate the confidentiality, integrity or authenticity of a system.

Side channel attacks (SCAs) refer to a class of attacks which focus on finding flaws in the implementation details of an algorithm or protocol, which may leak extra information about the computation being performed [42]. Examples of side channels which can be leveraged to attack a computer system include timing and execution speed, cache hits/misses, power consumption, electromagnetic radiation (such as light, radio waves and microwaves), and acoustic information (such as sound and vibration). SCAs can be used in conjunction with cryptanalysis and software security testing to uncover flaws in modern computer systems which may not be found using traditional techniques. A well known example of such flaws are the Spectre and Meltdown CPU bugs [30, 23], which were discovered using cache and timing attacks, and allow a process to read all memory. As of 2018, these vulnerabilities affect nearly every computer in the world [34].

Machine learning (ML) is the study of how computers can find patterns in data and learn to perform tasks automatically without being explicitly programmed to do so [27]. Typical machine learning tasks include: predicting values (either discrete or continuous) for some unseen data based on a previous dataset with labelled values (supervised learning [35]); identifying patterns or "clusters" within a dataset without being given any information a priori (unsupervised learning [16]); and finding optimal behaviours or routines to maximise some reward in an environment (reinforcement learning [44]). ML has applications in a wide variety of fields, including medicine, computer vision, email filtering and large language modelling, and is highly useful where an analytical algorithm may be too costly to develop or may not exist at all. Machine learning also strong applications to security, in particular to side channel attacks, and often results in attacks which are more effective than their non-ML counterparts, as shown later in the literature review.

The motivation of this paper is to understand how machine learning can be applied to side channel attacks to make them more effective in comparison with traditional techniques. While several non-ML approaches currently exist for different classes of SCAs (e.g., template attacks for power analysis, etc.), these are difficult to implement and often require a large amount of effort with diminishing returns. Furthermore, the typical workflow for carrying out these attacks – collecting a large number of sample observations and using analysis to discover secrets – naturally lends itself to ML techniques. Lastly, the diversity of algorithms, protocols, devices and platforms that currently exist is of concern to the security community at large: analysing and understanding the flaws which may exist in all of these is a significant, if not impossible undertaking. Emerging technologies such as the Internet of Things and cloud computing have exacerbated these fears, and consequently it is necessary to find approaches for discovering vulnerabilities which are effective and efficient. This paper proposes a project to demonstrate the efficacy of machine learning for side channel attacks, and discover new attacks on existing cryptographic systems.

2 Literature Review

This section covers the background material of this paper necessary to understand the concepts discussed and the project proposed, including a survey of research being conducted in this area.

2.1 Introduction to Side Channel Attacks

As described previously, side channel attacks refer to attacks which exploit information leakage across side channels (such as timing/execution speed, power consumption, etc.) which are not typically considered when implementing an algorithm or protocol. These side channels can be used to deduce secret information in a system, as described in table 1.

Side Channel Attack	**Description**
Timing Attacks	Attacks focused on measuring the execution times of a computation.
Cache Attacks	Attacks which monitor the cache of a computer and use this to infer secret information based on cache hits and misses.
Power Monitoring Attacks	Attacks which monitor the power consumption of a device during certain operations and uses power traces to deduce secrets.
Electromagnetic Attacks	Attacks which measure the radio waves and/or microwaves emitted from a device during computation and can often directly lift plain text data from secure devices.
Optical Attacks	Attacks which monitor the activity of a computer using video recording and uses this information to find secrets.
Acoustic Attacks	Attacks which correlate fan speed/noise and low frequency noise emitted by the CPU with power consumption, and uses audio recordings to guess secrets.
Fault Attacks	Attacks which deliberately introduce faults into a system to evaluate the current internal state.

Table 1: Side channel attacks and their descriptions.

Side channel attacks can vary by their application, environment and requirements. Some SCAs can be performed as black-box attacks against unknown systems, while others require deeper knowledge about the system being attacked including the algorithm or protocol being used and technical details around the implementation of these. Furthermore, some SCAs are only effective when the attack has physical access to the system (as is the case with power monitoring, electromagnetic and acoustic attacks) while others can take place on remotely, over multiple networks and even across the internet (such as timing and cache attacks). Finally, different side channels will require unique approaches to actually perform the attack. Nonetheless, they all follow the same basic process:

1. Gather a (potentially large) number of sample observations, usually based on some input.
2. Perform analysis on the results to (sometimes partially) infer secret information.
3. Repeat the previous steps till the secret is completely discovered, usually adjusting the input according to the results.

It should be noted that side channel attacks differ from traditional cryptanalysis and software security testing. Cryptanalysis focuses on the mathematical aspect of security and cryptography, using techniques such as model checking and cryptographic attacks to analyse the security of algorithms and protocols. Software security testing examines a system to discover flaws which an attacker can exploit to breach its security, with prime examples including SQL injections and buffer overflow attacks. Side channel attacks instead focus on how an algorithm or protocol can leak information across external channels which are not generally considered during cryptanalysis or security testing but still pose a significant threat. Nonetheless, it is used in conjunction with these fields to ensure security in computer systems, and eliminate potential attack vectors which malicious third parties can exploit. It should also be noted that extracting secrets by analysing and influencing human behaviour is not considered a side channel attack – this falls under social engineering.

SCAs are of particular concern to researchers as they are typically more fruitful and easier to implement compared with traditional cryptanalysis and consider attack vectors which are not usually taken into account with security testing. Furthermore, given the widespread use and reliance on cryptography, it is paramount that these kinds of flaws are identified so they can be eliminated. Current research suggests that modern web applications [10], mobile and IoT devices [32], cloud systems [1] and cryptographic systems in general [49] are at high risk of being vulnerable to side channel attacks, with cryptographic failures being the second highest vulnerability listed as part of the OWASP Top 10 for 2021 [37]. It is therefore necessary to develop techniques for efficiently identifying if a particular device or system is vulnerable to a side channel attack or not.

This paper focuses on timing and monitoring attacks as examples of SCAs which could have machine learning applied to them.

2.2 Timing Attacks

In a timing attack, the attacker measures the amount of time it takes for an algorithm to complete execution [26]. Each operation on a computer takes a certain amount of time to execute, and the amount of time may vary depending on the input. An attacker can examine the times taken for a particular operation to be performed with different inputs and use this information to deduce a secret, for example the password of a user. Consider the Python program given in figure 1:

```python
# Compare 2 strings. Return True if they
# are the same and False if not.
def compare_strings(str_1, str_2):
    if len(str_1) != len(str_2):     # Compare string lengths.
        return False

    str_length = len(str_1)
    for i in range(str_length):      # Iterate through each character.
        if str_1[i] != str_2[i]:     # Compare individual characters.
            return False

    return True

secret_string = "0123456789"

# Operation 1:
compare_strings(secret_string, "abcdefghij")

# Operation 2:
compare_strings(secret_string, "012345678X")
```

Figure 1: An insecure string comparison function.

The function terminates much quicker in the first operation compared with the second, due to the linear manner in which comparisons are made. An adversary can supply this function with multiple strings and compare the execution times for each, and use this information to deduce what the value of `secret_string` is. This attack can be extended to more complicated operations, provided that timing variations for execution are dependent on the input given to the algorithm. External noise (such as network traffic or CPU scheduling) can cause the attack to be slightly more difficult, however statistical correlation analysis can be performed to fully recover the value of a secret. This attack is powerful as it is simple to understand and implement and has low computational demands, but requires knowing the internal mechanisms of the algorithm implementation being attacked to understand where data-dependent timing variations can be introduced.

In 1996, Kocher presented a timing attack against RSA and Diffie-Hellman along with several other cryptographic schemes [26]. This attack was the first of its kind, and demonstrated that an attacker could factor RSA keys, discover exponents for Diffie-Hellman, etc., without performing complicated mathematics or brute force attacks, which are extremely difficult and expensive operations to carry out. Kocher stresses that while the attack is computationally efficient, it is also highly dependent on the details around the implementation of an algorithm: in Kocker's case, the square-and-multiply algorithm used for modular exponentiation introduces timing variations depending on the input parameters.

Kocher describes the following scenario: an attacker is attempting to guess some secret s which

is used with some algorithm implementation P to perform some cryptographic computation (e.g. decryption, digital signing, etc.) on some input m. It is given that M is the set of possible inputs and S is the set of possible secrets. Additionally, attacker knows the following functions:
$P : M \times S \to \theta : (m, s) \to P(m, s)$, the implementation of the algorithm.
$T : M \times S \to \tau : (m, s) \to T(m, s)$, the time taken for $P(m, s)$ to be computed.

Given that the attacker has already guessed the first $i - 1$ bits of s, the attacker can construct two messages m_0 and m_1 where the first bits are the correctly guessed bits and the last bit of m_0 is 0 and the last bit of m_1 is 1. The attacker can then find $t_0 = T(m_0, s)$ and $t_1 = T(m_1, s)$ and, based on appropriate statistical analysis of t_0 and t_1, accept either m_0 or m_1 as the message containing the ith bit of s. This process is repeated until the entire secret has been recovered. This method can be trivially extended to other types of secrets (e.g. binary strings, ciphertexts, etc.). The actual statistical analysis will depend on the algorithm being used, the properties of the inputs/outputs being computed, the environment the system takes place in (to account for noise), etc. but the core principle remains the same.

As part of their research, Kocher also suggested several defence mechanisms against this attack, including:

- Forcing all operations to run at a constant time quantum: this method is flawed as it does not take into account multiple platforms which may cause variations in timing (e.g. due to cache hits/misses, etc.) and unwanted compiler optimisations.
- Adding random noise to the algorithm: while feasible, defeating this method is simply a case of gathering more sample observations. In their paper, Kocher suggests that the number of samples required would increase from 10^6 to 10^7 – while not feasible at the time, it does not provide a great amount of security assurance.
- Using blinding methods to introduce randomness to the function [9]: this method is generally preferred despite the penalty performance as it more reliable and introduces noise in the data being computed, rather than simply increasing the time of computation, meaning an attacker cannot accurately guess the actual secret.

Kocher's attack, while limited in application and highly theoretical, gave rise to several other attacks which follow the same principles. In 2005, Brumley and Boneh demonstrated that OpenSSL's implementation of RSA was vulnerable to timing attacks, despite using Chinese Remainder Theorem and Montgomery Reductions to carry out the algorithm (which was previously considered impervious to timing attacks) [5]. Notably, the attack they carried out was over a network connection, which had been thought to be impractical due to the noise that may be introduced when transmitting data across a network. This attack led to the widespread adoption of blinding techniques to prevent such attacks.

Several similar attacks have been demonstrated to be highly effective against a wide variety of algorithms, protocols, devices and systems. A practical implementation of Kocher's attack was shown to be highly effective against the CASCADE smart card system by Dhem et al. who were able to recover a 512 bit key in a few hours using 300,000 sample observations and able to recover a 128 bit key in just a few seconds using 10,000 sample observations [13]. Similar to Brumley, this work demonstrated that timing attacks are highly practical and have real world impacts. Bernstein showed that timing attacks were effective against AES encryption in 2005 [2], which was followed by another attack by Bonneau and Mironov in 2006 following on from Bernstein's findings by being able to recover a 128 bit key using only 2^{13} samples, an improvement by a factor of almost four [3]. These findings led to timing attack resistant implementations of AES, first presented by Käsper and Schwabe, which claimed to be a constant time implementation of AES-GCM with a speed of 21.99 cycles per byte [22]. Brumley demonstrated again in 2011 OpenSSL was still vulnerable to remote timing attacks, this time focusing on the ECDSA algorithm [4]. Timing attacks were also shown to be effective at recovering user keystrokes and passwords from SSH sessions (as shown by Song et al. [41]), and modern algorithms based on elliptic curve cryptography have been shown to be continually vulnerable to timing attacks [20].

2.3 Power Monitoring Attacks

Power monitoring attacks, also known as power analysis attacks, are a class of SCAs which focus on monitoring the power consumption of a device while it performs computation. The principle is similar to that used in timing attacks: the amount of power a processor uses at any given time is related to and influenced by the operation currently being performed. An attacker can observe the power consumption of a particular system and use this information to deduce the data being currently passed through it [24], including secrets such as keys.

External tools such as oscilloscopes and resistors are typically used to monitor and record the current flowing through devices over time, known as traces. Traces sample the current during an operation; a 1 millisecond operation sampled at 3 MHz yields 3,000 sample points. Multiple traces may need to be collected for certain attacks to be performed (see below for details). Consequently, this class of attacks requires (at least) close physical proximity to the device being attacked, and are typically more invasive compared with timing attacks (which may be performed remotely).

Power monitoring attacks can be broadly split into two kinds of attacks: those which make use of profiling techniques (such as template attacks and stochastic techniques) and those which do not (including simple power analysis, differential power analysis and correlation power analysis). Profiling techniques typically make use of a device identical to the system being attacked, and are useful when the number of samples that an adversary can gather is limited. Non-profiling attacks are generally used when an attacker can gather as many samples as needed, though careful consideration must be made as to the specific approach that is used depending on the specific scenario.

The first examples of power monitoring attacks can be traced back to Kocher's 1999 paper, introducing simple power analysis (SPA) and differential power analysis (DPA) [24]. Both these attacks are non-profiling attacks which use traces to identify patterns during encryption, which can be then used to infer secrets. Simple power analysis is the most basic kind of power monitoring attack. The attacker simply observes a trace as a cryptographic operation is being executed and uses this to directly infer secrets. An example of this is shown in figure 2, where the multiple rounds of the DES encryption are clearly visible.

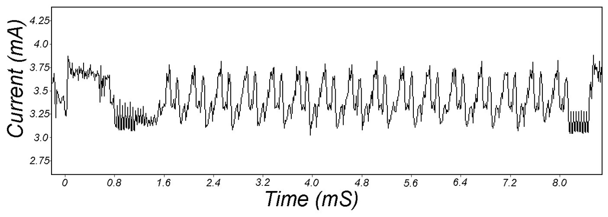

Figure 2: A trace of the DES algorithm being executed. Credit to Kocher et al. [24].

SPA is simple to implement and execute, and specific operations (such as the square and multiply operations performed during RSA or individual rounds performed during AES) can be easily identified, allowing an attacker to easily deduce what data is being computed. Kocher shows that SPA is effective against naïve implementations of DES, along with other algorithms which do not contain protections against SPA. Some suggested defences against SPA include:

SPA is best used against simple machines such as smart card readers, which only perform basic operations. Against more complex machines (such as desktop computers or FPGAs) however, SPA is not sufficient to perform an attack as it can be very difficult to identify a single signal from a processor which may be performing multiple operations in parallel. Additionally, the increased processing power and additional noise from external process can make sampling extremely challenging and a single sample will not be enough to recover a secret. These challenges make implementing a practical attack very difficult in a realistic scenario.

To counter these issues, Kocher proposes differential power analysis (DPA). Multiple traces are recorded and these are analysed to guess some secret. The method suggested by Kocher is as follows [25]:

1. Collect a (usually large) number of traces, including the inputs used to produce them.
2. Split the traces into subsets based on their inputs. (Note: this is the guessing stage of the attack – the aim is to find the input which has the highest correlation with the power output under the assumption the output will be highest when the correlation is strongest).
3. For each subset of traces:
   (a) Compute the average trace of the subset.
4. Compute the difference between the average traces of the subsets and note the result.
   - If the average traces show little to no difference then the input value was not in the secret.
   - If the average traces show a clear difference then the input value was in the secret.

The rationale here is that if a sufficient number of traces are analysed in this manner, even miniscule differences between the outputs of the algorithm can be guessed. A demonstration of DPA is shown in figure 3 and the method described above can be used to guess individual bits used in DES and AES encryption, as shown by Kocher et al. [24, 25].

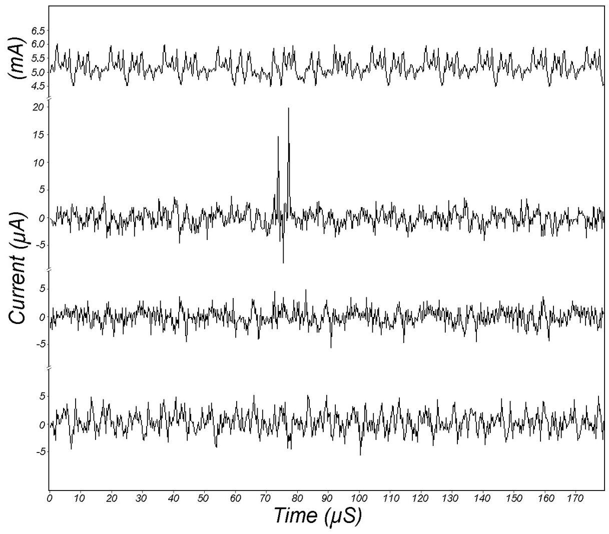

Figure 3: Traces showing DPA being executed against DES. The first trace is a reference, the second is a correct guess while the last two are incorrect guesses. Credit to Kocher et al. [24].

Kocher recommends the use of blinding methods again to prevent these sorts of attacks. Additionally, they also recommend increasing the signal-to-noise ratio: this can be done either by increasing the amount of noise in a sample so attacks are more challenging to carry out, or by reducing the level of leakage using techniques such as balancing. Both of these are highly effective at creating algorithms that are resistant to DPA.

To combat DPA resistant algorithms and for scenarios where the number of traces that can be extracted is very limited (e.g. due to physical security measures, etc.), template attacks were proposed by Chari et al. [8]. Here, an adversary uses an identical copy of the victims device to create a profile of the attack. The attacker uses the device copy to simulate the attack, creating potentially thousands of power traces with various keys and plaintexts to produce a “template”. This template can then be analysed along with a number of traces from the victims device to complete the attack – given enough preprocessing, it is possible to recover a key using just one victim trace. The steps to carry out this attack are described below:

1. Using the copy of a device, create a large dataset of power traces.

2. Create a template of the device by measuring certain points of interest during the cryptographic operation.

3. Using the traces from the victims device, apply the template and use this to find keys/subkeys. If necessary, repeat till the entire secret is recovered.

The number of victim traces required varies depending on the quality of the template, which can increase processing times. This attack has the advantage that only a very small number of traces (potentially just one) are needed to break a cryptosystem; conversely a large amount of time must be spent setting up the attack, including creating traces and templates alongside choosing the appropriate points of interest.

Both DPA and template attacks have been shown to be highly effective at breaking cryptographic systems, both from their original papers alongside the research that has stemmed from them. In 1999, Goubin and Patarin demonstrated a practical DPA attack against DES, and also showed several practical methods which can be used to increase DPA resistance [15]. Velegalati et al. showed a DPA attack against AES running on an FPGA system in 2008 [45]. FPGAs are resistant to SPA attacks given the amount of noise that they produce – this paper demonstrates that DPA is able to counter the noise-to-signal ratio and find the secret. The Trivium and Gain stream ciphers of the eSTREAM cipher project were shown to be susceptible to DPA, with Fischer et al. demonstrating a novel technique to remove all the noise and leave the pure signal [14]. Attacks against ECDSA using DPA were shown by Itoh et al. and separately by Wu et al. illustrating that DPA is effective against elliptic curve cryptography [19, 47]. DPA was also shown to be effective against implementations of RSA, as exemplified by Yen et al. [48].

In 2004, Oswald and Rechberger published a paper which implemented template attacks [39]. This paper was particularly important as it covers several additional details vital to ensuring the success of the attack, including how to preprocess data and how to select points of interest, which are not covered by Chari et al. and shows that using preprocessing techniques improves classification by a significant factor. In 2006, Oswald showed that masking techniques used as a defense mechanism against DPA could be circumvented using a combination of DPA and template attacks, and showed an attack against AES encryption [36]. An attack on ECDSA was shown by Medwed in 2008, demonstrating again that elliptic curve cryptography (which is designed to be resistant to side channel attacks, in particular timing attacks) is vulnerable to template attacks and power monitoring side channels [33].

2.4 Application of ML to SCAs

Previous work has been done to apply machine learning techniques to side channel attacks. The earliest examples of this date back to 2011, when Hospodar et al. used Least Squares Support Vector Machines to attack the AES encryption scheme [18]. It was found that LS-SVMs had a very high success rate, dependent mostly on the parameters used for the machine learning technique rather than on the number of traces used for training the SVM. It was noted that when using an extremely small number of traces (500), the SVM performed poorly, however this was also true of template attacks. The paper concludes showing that future work in this area could prove to be highly fruitful for the future – while SVMs are heavier compared with standard template attacks, they could pave the way for alternative machine learning methods.

In 2015, Lerman et al. revisited the idea of machine learning being used for template attacks, covering the effectiveness of ML techniques in this area to date [29]. This paper was particularly concerned with the so-called "curse of dimensionality": the idea that data with a large number of features is more difficult to analyse and extract results from despite seemingly having more detail due to the range of features. Lerman suggested that while template attacks are theoretically better, ML based techniques may prove to be more practical. This is reinforced by their earlier research, where a machine learning approach against AES reduced the number of traces required to carry out an efficient and correct attack [28]. Future survey work including Picek et al. were more in favour of machine learning techniques, going so far as to suggest that traditional methods such as template attacks are no longer sufficient for identifying security vulnerabilities in systems [38].

With the diversity of systems and devices that can be used to perform attacks, researchers were interested in how machine learning can help alleviate the gap between different devices. Wang et al. showed that using different devices can decrease the performance of MLPs used for by up to 75%, even if identical chips are used [46]. To counter this, deep learning models can be used, as demonstrated by Das et al. who showed that accuracies greater than 99.9% can be achieved despite multiple devices being used for side channel attacks [11].

A survey paper from 2020 by Hettwer et al. reviewed the current state of the art with respect to machine learning techniques being applied to side channel attacks [17]. The main findings were as follows:

- ML can be used to achieve various outcomes depending on the scenario, including direct key recovery, recovering intermediate key states, mask recovery (which can then be used to perform direct key recovery later) and aligning traces.
- ML techniques can also be used in many parts of the general SCA workflow, including preprocessing and feature engineering to remove noise and increase success rates.
- Deep learning techniques enjoy great success but without hyperparameters being specified it can be difficult to reproduce research.

While the majority of research in this area focuses on supervised learning techniques used for power analysis side channel attacks, there is a growing interest ML methods used for timing attacks. A recent paper from Luo et al. demonstrates a novel reinforcement learning technique used for performing cache-timing attacks called AutoCAT [31]. This method is unique as it poses the problem of timing attacks as a game where the algorithm must guess the secret. This approach could prove highly promising in developing novel algorithms for attacking systems remotely.

Finally, in 2021, researchers at Google published SCAAML: a deep learning framework used for attacking cryptosystems using side channel attacks [6, 7]. This library can be used to carry out attacks based around machine learning and encourage further research.

3 Project Specification

This section covers the specification of the project, including experimental design, success criteria, project contributions, legal/ethical issues, risks and mitigations, and a timeline of deliverables.

3.1 Aims and Objectives

The broad aim of this project is to carry out an empirical analysis to compare SCAs which employ ML techniques with traditional SCA approaches. To achieve this, a series of experiments around timing and power analysis attacks on the OpenSSL [43] implementations of RSA [40] and ECDSA [21] will be performed. The traditional (non-ML) techniques that will be used are standard timing attacks for the timing side channel and DPA for the power consumption side channel, on both algorithms. The ML techniques used will be reinforcement learning for timing attacks and deep learning for power monitoring attacks. These attacks will then be used to carry out a series of experiments to evaluate the overall efficacy of ML-based SCAs (see section 3.2 for more details).

The following list specifies the success criteria that the project will be evaluated against:

- An implementation of a non-ML timing attack against OpenSSL's RSA and ECDSA algorithm.
- An implementation of a non-ML DPA attack against OpenSSL's RSA and ECDSA algorithm.
- An implementation of a ML-based timing attack against OpenSSL's RSA and ECDSA algorithm.

- An implementation of a ML-based power analysis attack against OpenSSL's RSA and ECDSA algorithm.
- A final report describing a series of experiments carried out with these attack implementations, including a discussion of the results.

For the project to be considered successful, it must complete these success criteria.

3.2 Experimental Design

This section describes a series of experiments that will be performed to understand how machine learning techniques can be used to improve non-ML techniques with SCAs. The proposed scenario is as follows: some data C is signed with some algorithm A and some private key k to produce some signature S: $A(C, k) = S$. The attacker knows the data C, the signature S and the algorithm A (which is either RSA or ECDSA), and the objective is to find the secret key k.

To achieve this, the attacker will use timing attacks (similar to Kocher and Brumley [26, 4, 5]) and DPA attacks (similar to Kocher [24]). These will be implemented as part of the project, and represent the non-ML versions of timing and power monitoring attacks respectively. Additionally, the attacker will also use machine learning based SCAs to attack the cryptosystem, specifically reinforcement learning (similar to Luo et al. [31]) and deep learning (similar to SCAAML [6]). The attacker will then implement these attacks against the OpenSSL implementations of these attacks using an unknown key and some data of their choice. Versions of OpenSSL which are vulnerable to these attacks may be used: the aim is not to discover new attacks, instead it is to compare machine learning with analytical techniques. To do this, the experiment will measure the time taken to train the different models, whether or not the model could successfully carry out an attack, the precision, accuracy, recall and F1 score of the models and compare this with the number of traces and amount of time taken to carry out analytical techniques. Finally, the experiment will use these results to carry out an evaluation of ML techniques compared with traditional SCA methods.

3.3 Project Contributions

This project contributes to the wide scientific community by performing comparative empirical analysis of machine learning techniques for side channel attacks, which has not previously been done before (as shown in section 2). While individual analysis have been done when exploring novel ideas, a study on the effectiveness of ML based SCAs with traditional SCAs has not been conducted. Are ML techniques well suited to SCA problems? Can they be used to advance the field and are they practical to implement? Do they help find vulnerabilities that are not discoverable with traditional techniques? Are they more effective when compared with traditional techniques or are there trade-offs to consider? It is these sorts of questions that the study proposed above will help to answer, and help direct the future course of this field.

3.4 Legal and Ethical Issues

This project contains no significant legal issues. The algorithms to be worked with (ECDSA and RSA) and their implementation software (OpenSSL) are publicly available. The algorithms for RSA and ECDSA are well known to the wider security community, are not encumbered by patents, and are commonly implemented in multiple libraries and experimented on by security researchers. The OpenSSL software implementation is licensed under the permissive Apache 2 licence or the OpenSSL and SSLeay dual licence depending on the version release. Both of these licences permit the user to perform scientific experiments on the software and provide the software "as is", along with including some terms and conditions regrading redistribution – this project will not be redistributing the OpenSSL software so this is not of concern.

As the project does not process any personal information, there are no ethical concerns around ensuring privacy. There are some ethical concerns that the project could be used to discover vulnerabilities in the OpenSSL software or the RSA/ECDSA algorithms, and that these vulnerabilities could be used in unethical ways such as to steal secret user information. This is discussed in more detail in section 3.5.

3.5 Risks and Mitigations

This project is purely a scientific project involving experimentation on open source software, so there are no environmental, health, safety, privacy or financial risks. There are no legal risks as the software and algorithms being used are open source. There are some project and security risks, which are described in table 2 along with some mitigation strategies.

Risk	**Likelihood**	**Impact**	**Mitigation**
The project could be used for malicious purposes.	Low	High	The project will be released under the MIT public license to encourage security research around this area and to ensure that the playing field is even: both the security community and potential malicious actors will have access to the same resources. Additionally, the project focuses on vulnerabilities which have already been discovered and mitigated. Finally, if any new vulnerabilities are discovered, an appropriate and ethical disclosure strategy will be followed to be discussed with the supervisor as needed.
The project may not be completed on time.	Low	High	A well defined success criteria specifying the expected outcomes of this project has been given in 3.1 along with a Gantt chart specifying when they are expected to be completed in section 3.6. Additionally, regular updates in the form of weekly meetings or emails will be provided to the supervisor along with a log that will be regularly updated to ensure that progress is tracked and followed.

Table 2: Identified risks and their mitigations.

3.6 Timeline of Deliverables

A Gantt chart describing a timeline of expected tasks is shown in figure 4.

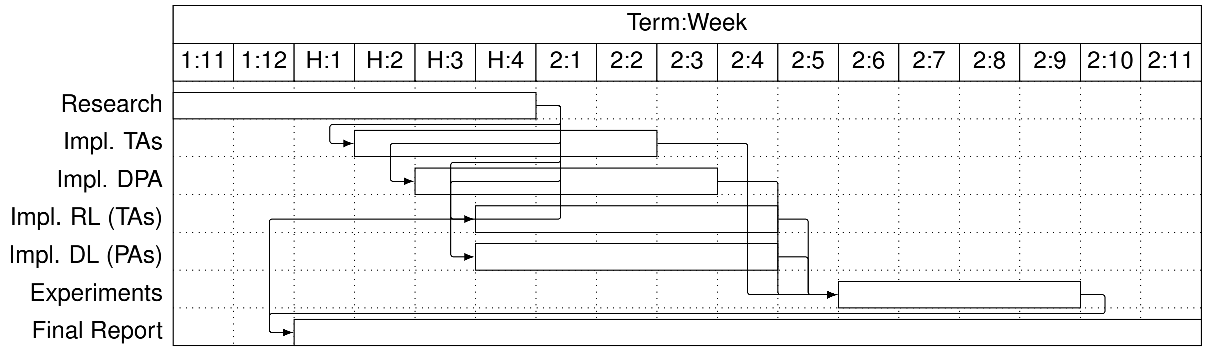

Figure 4: A Gantt chart showing the timeline of deliverables.

The tasks shown in figure 4 correlate to the expected deliverables listed in the success criteria: an implementation of a timing attack, a DPA attack, a deep learning attack and a reinforcement learning attack.

References

[1] Bazm, Mohammad-Mahdi and Lacoste, Marc and Südholt, Mario and Menaud, Jean-Marc. Side channels in the cloud: Isolation challenges, attacks, and countermeasures, 2017. URL: https://inria.hal.science/hal-01591808/document. Retrieved November 29, 2023.

[2] D. J. Bernstein. Cache-timing attacks on AES, 2005. URL: https://mimoza.marmara.edu.tr/~msakalli/cse466_09/cache%20timing-20050414.pdf. Retrieved November 29, 2023.

[3] J. Bonneau and I. Mironov. Cache-Collision Timing Attacks Against AES. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 201–215. Springer, 2006. DOI: 10.1007/11894063_16.

[4] B. B. Brumley and N. Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011. DOI: 10.1007/978-3-642-23822-2_20.

[5] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005. DOI: 10.1016/j.comnet.2005.01.010.

[6] E. Bursztein et al. SCAAML: Side Channel Attacks Assisted with Machine Learning, 2019. URL: https://github.com/google/scaaml. Retrieved December 1, 2023.

[7] E. Bursztein, L. Invernizzi, K. Král, D. Moghimi, J.-M. Picod, and M. Zhang. Generic Attacks against Cryptographic Hardware through Long-Range Deep Learning. *arXiv preprint arXiv:2306.07249*, 2023. DOI: 10.48550/arXiv.2306.07249.

[8] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 13–28. Springer, 2003. DOI: 10.1007/3-540-36400-5_3.

[9] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983. DOI: 10.1007/978-1-4757-0602-4_18.

[10] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *2010 IEEE Symposium on Security and Privacy*, pages 191–206. IEEE, 2010. DOI: 10.1109/SP.2010.20.

[11] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-DeepSCA: Cross-device deep learning side channel attack. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6. IEEE, 2019. DOI: 10.1145/3316781.3317934.

[12] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. Springer, 2015, pages 1–10. ISBN: 9783662479742. DOI: 10.1007/978-3-662-47974-2_1.

[13] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In *Smart Card Research and Applications: Third International Conference, CARDIS'98, Louvain-la-Neuve, Belgium, September 14-16, 1998. Proceedings 3*, pages 167–182. Springer, 2000. DOI: 10.1007/10721064_15.

[14] W. Fischer, B. M. Gammel, O. Kniffler, and J. Velten. Differential power analysis of stream ciphers. In *Topics in Cryptology–CT-RSA 2007: The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007. Proceedings*, pages 257–270. Springer, 2006. DOI: 10.1007/11967668_17.

[15] L. Goubin and J. Patarin. DES and differential power analysis the "Duplication" method. In *Cryptographic Hardware and Embedded Systems: First InternationalWorkshop, CHES'99 Worcester, MA, USA, August 12–13, 1999 Proceedings 1*, pages 158–172. Springer, 1999. DOI: 10.1007/3-540-48059-5_15.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2009, pages 485–585. ISBN: 9780387848587. DOI: 10.1007/978-0-387-84858-7_14.

[17] B. Hettwer, S. Gehrer, and T. Güneysu. Applications of machine learning in side-channel attacks: a survey. *Journal of Cryptographic Engineering*, 10:135–162, 2020. DOI: 10.1007/s13389-019-00212-8.

[18] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302, 2011. DOI: 10.1007/s13389-011-0023-x.

[19] K. Itoh, T. Izu, and M. Takenaka. Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 129–143. Springer, 2003. DOI: 10.1007/3-540-36400-5_11.

[20] J. Jancar, V. Sedlacek, P. Svenda, and M. Sys. Minerva: The curse of ECDSA nonces: Systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):281–308, 2020. DOI: 10.13154/tches.v2020.i4.281-308.

[21] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001. DOI: 10.1007/s102070100002.

[22] E. Käsper and P. Schwabe. Faster and timing-attack resistant AES-GCM. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–17. Springer, 2009. DOI: 10.1007/978-3-642-04138-9_1.

[23] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020. DOI: 10.1145/3399742.

[24] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology-CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999. DOI: 10.5555/646764.703989.

[25] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1:5–27, 2011. DOI: 10.1007/s13389-011-0006-y.

[26] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology-CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996. DOI: 10.1007/3-540-68697-5_9.

[27] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*. Springer, 1996, pages 151–170. ISBN: 9789400902794. DOI: 10.1007/978-94-009-0279-4_9.

[28] L. Lerman, G. Bontempi, and O. Markowitch. A machine learning approach against a masked AES: Reaching the limit of side-channel attacks with a learning model. *Journal of Cryptographic Engineering*, 5:123–139, 2015. DOI: 10.1007/s13389-014-0089-3.

[29] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *Constructive Side-Channel Analysis and Secure Design: 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers 6*, pages 20–33. Springer, 2015. DOI: 10.1007/s13389-014-0089-3.

[30] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, et al. Meltdown: Reading kernel memory from user space. *Communications of the ACM*, 63(6):46–56, 2020. DOI: 10.1145/3357033.

[31] M. Luo, W. Xiong, G. Lee, Y. Li, X. Yang, A. Zhang, Y. Tian, H.-H. S. Lee, and G. E. Suh. AutoCAT: Reinforcement Learning for Automated Exploration of Cache-Timing Attacks. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 317–332. IEEE, 2023. DOI: 10.1109/HPCA56546.2023.10070947.

[32] Y. Lyu and P. Mishra. A survey of side-channel attacks on caches and countermeasures. *Journal of Hardware and Systems Security*, 2:33–50, 2018. DOI: 10.1007/s41635-017-0025-y.

[33] M. Medwed and E. Oswald. Template attacks on ECDSA. In *International Workshop on Information Security Applications*, pages 14–27. Springer, 2008. DOI: 10.1007/978-3-642-00306-6_2.

[34] Meltdown and Spectre Contributors. Meltdown and Spectre, 2018. URL: https://meltdownattack.com. Retrieved November 28, 2023.

[35] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018, pages 1–8. ISBN: 9780262039406.

[36] E. Oswald and S. Mangard. Template attacks on masking—resistance is futile. In *Cryptographers' Track at the RSA Conference*, pages 243–256. Springer, 2007. DOI: 10.1007/11967668_16.

[37] OWASP Foundation. OWASP Top Ten, 2021. URL: https://owasp.org/www-project-top-ten. Retrieved November 29, 2023.

[38] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4095–4102. IEEE, 2017. DOI: 10.1109/IJCNN.2017.7966373.

[39] C. Rechberger and E. Oswald. Practical template attacks. In *International Workshop on Information Security Applications*, pages 440–456. Springer, 2004. DOI: 10.1007/978-3-540-31815-6_35.

[40] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. DOI: 10.1145/359340.359342.

[41] D. X. Song, D. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *10th USENIX Security Symposium (USENIX Security 01)*. USENIX Association, 2001. URL: https://www.usenix.org/conference/10th-usenix-security-symposium/timing-analysis-keystrokes-and-timing-attacks-ssh. Retrieved November 29, 2023.

[42] F.-X. Standaert. *Secure Integrated Circuits and Systems*. Springer, 2010, pages 27–42. ISBN: 9780387718293. DOI: 10.1007/978-0-387-71829-3_2.

[43] The OpenSSL Project. OpenSSL, 1998. URL: https://www.openssl.org. Retrieved December 1, 2023.

[44] M. van Otterlo and M. Wiering. *Reinforcement Learning*. Springer, 2012, pages 3–42. ISBN: 9783642276453. DOI: 10.1007/978-3-642-27645-3_1.

[45] R. Velegalati and P. S. Yalla. Differential power analysis attack on FPGA implementation of AES. *ECE 746 Statistical Signal Processing*, 2008. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dfd6d12f2e840c58a919b9604eaa7c01d3c055b4. Retrieved December 1, 2023.

[46] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova. How diversity affects deep-learning side-channel attacks. In *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, pages 1–7. IEEE, 2019. DOI: 10.1109/NORCHIP.2019.8906945.

[47] K. Wu, H. Li, T. Chen, and F. Yu. Simple Power Analysis on Elliptic Curve Cryptosystems and Countermeasures: Practical Work. In *2009 Second International Symposium on Electronic Commerce and Security*, volume 1, pages 21–24. IEEE, 2009. DOI: 10.1109/ISECS.2009.7.

[48] S.-M. Yen, W.-C. Lien, S. Moon, and J. Ha. Power analysis by exploiting chosen message and internal collisions–vulnerability of checking mechanism for RSA-decryption. In *Progress in Cryptology–Mycrypt 2005: First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005. Proceedings 1*, pages 183–195. Springer, 2005. DOI: 10.1007/11554868_13.

[49] Y. Zhou and D. Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *Cryptology ePrint Archive*, 2005. URL: https://eprint.iacr.org/2005/388. Retrieved November 29, 2023.