A Survey on Side Channel Attacks

Suhas Thalanki - PES1UG19CS563

Sai Ramya Paturi - PES1UG19CS419

Side-Channel attacks are based on reading unintended signals of a computer rather than weaknesses in the security of the Cryptographic systems. This can be done by measuring various physical parameters such as power consumption, execution time, electromagnetic radiation among others.

Classical attacks on cryptographic systems depended on either the plaintext, ciphertext or both along with, other attacks such as brute force or Man-In-the-Middle attacks which directly depend on the actual cryptosystems themselves.

Side Channel attacks do not depend on these cryptosystems directly, but depend on them indirectly. They work on the unintended outputs of these cryptosystems such as changes in power consumption, execution time, electromagnetic radiation among others which act as hints or clues to these attacks. They also give hints about the internal states of block ciphers by using statistical tests that make the key slightly non-random.

These attacks are of a huge concern as they lead to a huge breach in security and can often be implemented with ready-made hardware that ranges from a few hundred to a few thousand dollars. Some attacks such as **Simple Power Analysis** (SPA) take only a few seconds whereas **Differential Power Analysis** (DPA) takes a few hours.

Types

Some types of Side Channel Attacks are:

- **Timing Attack**: Attacker aims to gain information by analysing the time taken to execute cryptographic algorithms or relevant operations.
- **Power-Monitoring Attack**: Based on the various levels of power draw by the hardware while performing cryptographic operations or executing cryptographic algorithms.
- **Acoustic Cryptanalysis**: Exploits the amount and variation of sound emitted by a computer during computation. Mainly focuses on the sounds produced by keyboards and internal components.
- **Electromagnetic Attack:** These attacks are based on the electromagnetic radiation that is being leaked, and can result in directly providing the plaintext. Can also be used to infer keys.

Timing Attack

It is a Side-Channel Attack in which the attacker aims to exploit a system by analysing the time taken to execute cryptographic algorithms.

Every option takes time to execute in a computer. This time also differs based on the input. An attacker can find the input by using precise measurements and also by looking at various inputs to notice trends in the time taken to execute certain logical operations.

Timing attack can be a more efficient and easier method to find out secrets/keys when compared to cryptanalysis of known plaintext, ciphertext pairs.

Even though Timing Attacks work well in research conditions, they are very rare in the wild. In addition, one must execute these attacks multiple times to gather samples that reveal the timing difference and give clues regarding the plaintext and the key. Hence, they tend to be very slow and not discreet. They also are **all-encompassing**, i.e., timing measures leak **everything** a system is doing, not just cryptographic algorithms.

Example

```
# A function to compare two strings
def compare(s1, s2):
    for i in range(len(s1)):
        if s1[i] != s2[i]:
            return False
    return True
```

In the above example, the number of iterations the program executes for is not fixed.

Let us take two strings `abcde` and `bcdef` and the user **ONLY knows the second string** `bcdef`. In this case, the first characters in each string do not match, and as a result the function exits on the first iteration of the loop. Let us assume the time taken to be `t`.

Now take the second string to be `abdef`. Here, the first two characters in each string match, but the third characters do not. Hence, the loop exists on the third iteration. The time taken in this case would be `t + y` where `y` is the time taken for the second and third iterations to run.

The difference in the time can be taken to estimate how many characters have matched compared to the first pair. Similarly, this procedure can be repeated to guess the remaining digits by observing `y`. This helps in **significantly** reducing the number of guesses required to find the string when compared to Brute Force, and signifies a leak in information and hence is **NOT perfectly secret**.

The above security flaw can be fixed by having a `flag variable` to ensure that the execution time does not vary regardless of what the second string is.

```
# Patched function
def compare(s1, s2):
    flag = True
    for i in range(len(s1)):
        if s1[i] != s2[i]:
            flag = False
    return flag
```

Constant-Time Cryptography

They are pieces of code that do not leak information via timing analysis, i.e., they are **not** vulnerable to timing attacks. There are two main approaches to Constant-Time Cryptography:

- Execution time does not depend on the `secret` elements that are to be protected. Either the execution time is constant, or it is not correlated to the secret elements, i.e., execution time can vary but it is not

dependent on the secret elements.
- The other method is **masking** where you introduce a random integer `r modulo n` and add this to encrypting a plaintext. This results in the attacker not knowing what the encryption was performed on, thus depriving him the chance of correlating execution time with a plaintext or a key.

Example

In RSA, encryption is $m^d \bmod n$. This can be converted to $r^{-1}(mr^e)^d \bmod n$.

The ideal method to ensure Constant-Time cryptography is to write code in assembly as modern languages and compilers aren't built for constant-time cryptography. Instructions that are Constant-Time can also vary from chipset to chipset and architecture to architecture.

The most practical method is to write code that **looks like** Constant-Time. This is not guaranteed to work as if the compiler detects a better way to execute the code, it will, thus not ensure constant-time cryptography.

There is research going on to make compilers to be more resistant to Timing attacks and till then, the mentioned practices must be taken to ensure security.

Power Monitoring Attack

Power Monitoring is a Side Channel attack is based on the power consumptions of cryptographic devices. Capturing data is as simple as monitoring power consumption while a cryptographic or critical operation is being performed.

Power Variation

Integrated Circuits are built using Transistors which act as switches controlled by voltage. Current flows across the transistor when charge is applied to/removed from the transistor. This leads to a flow of current from this transistor to other transistors, wires among other electrical components. This movement of current consumes power and emits electromagnetic radiation which can be observed externally.

 Transistor

As microprocessers have regular transistor switching actions, the change in power consumption can be analysed to identify activity.

Simple Power Analysis (SPA)

In SPA, attacker directly observes power consumption over a period of time. Different operations draw different amounts of power. Using this, one can distinguish what operations are being executed at a particular instance in time. For example, multiplication and addition can be identified separately as multiplication draws more power than addition. In addition, during reading of data, the ratio of 1s when compared to 0s will be reflected in the power consumption.

 Simplified SPA Graph The above graph is a simplified graph of SPA.

Large features and operations such as DES Rounds and RSA operations can be identified as they have the operations performed during various stages of these vary to a large extent.

SPA can be used to break RSA by looking at the difference in power consumption during multiplication and addition as stated earlier. It can also be used to break DES due to visible differences in power consumption during permutations and shifts.


DES SPA

The above figure represents power consumption during DES. The **upper curve** represents the **entire encryption process**, including initial permutation, 16 rounds and the final permutation. The **lower curve** represents the **second and third rounds** of the encryption process.

It is not practical when there is a lot of noise in the power consumption. In addition, it is very easy to prevent a SPA-resistant device as well.

Differential Power Analysis (DPA)

It is a statistical method to apply data-dependent correlations and is much more powerful than SPA, and as a result, much more difficult to prevent.

There are two major phases to DPA:

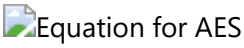
- **Data Collection**: Sampling a device's power consumption while executing cryptographic operations as a function of time.
- **Data Analysis**: Analysing the collected data to find correlations between cryptographic operations and power consumption.

You require two sets of data, multiple traces of each, for this approach. Now one must compute the difference between the average of these traces. If this value is zero, there is no correlation. If the difference is non-zero, these sets of data are correlated. Even tiny correlations can be seen given enough traces as noise in a system is eliminated during the averaging.


DPA

The average of two sets of traces in DPA are shown on the first two lines of the above picture. The difference of those two traces is shown in the third line. The trace on the fourth line is the same but magnified by a factor of 15. There is a clear non-zero value being shown in the fourth line and hence it can be inferred that these two straces are correlated. If there was no correlation, the difference would be approximately zero.

Application of DPA to break AES


Equation for AES

The above picture contains the equation of AES where S is a look-up table, which is used on the XOR of a known input X_n and an encryption key K_n. To determine K_n, several guesses are made.
The first set of traces fall into the set where LSB of Output is 0. The second set of traces fall into the set where LSB of Output is 1. As stated earlier, the difference of the average of the two traces is computed and examined. In the below figure, there are traces of five different K_n values.


DPA AES

The correct key is in the **third** trace. As the encryption key is usually a 128-bit value, the number of attempts to test every single value would be 2^{128}. This key can be broken into `16-bit` blocks where each byte can be solved individually. It will take only 16 * 256 attempts or 4096 attempts to break the entire key.

Countermeasures

SPA

SPA can be prevented by just adding noise to the system. In addition, one can also perform random operations to obscure the power consumption done by other operations. The design should **not** have conditional branches.

DPA

Preventing DPA is much more challenging. A few common methods are:

- Decrease the **signal-to-noise** ratio. Lower the ratio, greater the number of traces required to perform the attack. Adding random wait states and dummy operations also helps.
- Balancing the amount of power used for given data value or operation. Balancing power consumption will result in reducing the amplitude of trace, making detecting correlations that much harder.
- The **most effective** and **least difficult** way to prevent side-channel attacks is to limit the number of transactions that can be performed with a particular key, like a password timeout.

Electromagnetic side channel attack

Introduction

- According to Kerchoff's regulations, cryptographic security should be based exclusively on the secrecy of the key, not on the concealment of the encryption technique.
- However, a cryptosystem that employs a specific encryption method may have flaws in its physical implementation.In that circumstance, one or more leaks of any kind could offer useful information to an attacker.
- Physical signals are frequently employed as a source of leakage in side channel cryptanalysis.
- For example, time, power consumption, or electromagnetic radiation can be used.
- Leakage of electromagnetic radiation has been recognised for a long time, and it is also the topic of modern research.
- When looking at cryptographic implementations, the near and far field of cryptographic processors can be a source of leakage that should be considered.

Principles

- Radiation emitted by electronic components can be investigated while they are performing a delicate computation involving a secret. As a result, measuring the chip's electromagnetic radiation during computing is a viable option.
- **Simple Power Analysis (SPA)** and **Differential Power Analysis (DPA)** are based on consumption changes that occur during computation as a function of the value of a key bit, which can be utilised to recover the key.

Use / Example

- The figure depicted below shows two curves, representing the initialisation of a smart card, the execution of a DES and the stopping of the card.
- Both curves have signal to noise ratios that are very close, but the curve that details the current through the capacitor, requires 25 times less traces than the one that represents the current measurement.

 EM

Countermeasures

- At the hardware level, there are numerous countermeasures, although they are not all well suited to all instances and must occasionally be locally tweaked.
- The first of these countermeasures is, of course, the employment of a Faraday cage to prevent radiation leakage of any kind.
- Although this countermeasure is excellent, it is also the hardest to implement.
- When employing a Faraday cage, there are numerous and significant limits that cannot always be disregarded.
- A tiny metal covering (preferably ferromagnetic) can occasionally serve to decrease radiation and make measurements more difficult.
- When a Faraday cage cannot be employed due to binding wires, power supply lines, or mechanical limits (the thickness of a smart card is set at 0.76 mm), a protection zone should be defined around the device.

Acoustic cryptanalysis

- Due to vibration in certain of their electronic components, many computers make a high-pitched noise while operating.
- We demonstrated that various RSA keys produce different sound patterns in an early presentation (Eurocrypt'04 rump session), but it was unclear how to extract individual key bits.

Introduction

- Cryptanalytic side channel attacks target implementations of cryptographic algorithms that, while mathematically secure, mistakenly leak secret information through indirect channels such as power usage, electromagnetic emissions, timing fluctuations, CPU resource competition, and so on.

- Another potential avenue is acoustic emanations, but this has only been utilised to listen in on slow electromechanical components like keyboards and printers thus far. -Mechanical noise from fans and storage devices like hard discs clearly indicates system activity, but it appears to carry only very coarse information that is of little use for cryptanalysis.

Acoustic Attack scenarios

An Acoustic Attack app

- Mobile phones are ubiquitous, and they all have integrated microphones with enough bandwidth and sensitivity to mount key extraction assaults, as we show.
- Furthermore, they have extensive signal processing capabilities and can finish the adaptive chosen-ciphertext loop in real time (thanks to their wireless data link).

- As a result, the entire attack might be packaged as a software "programme" that does not require any particular hardware or knowledge.
- Under some pretext, an attacker would install this programme, go physically close to the target computer, and position the phone suitably for the length of the attack.

Self eavesdropping

- If taken to its logical conclusion, a device with a microphone may be used to spy on itself.
- The attacker is in control of an unprivileged process with only microphone recording permissions and network connectivity in this situation (eg, a web page using th eHTML media capture features or a flash app, as in existing web based video conferencing services).
- The attacker can use these to record and analyse cryptographic actions taking place in another process, or even a different virtual machine, on the same computer.

Eavesdropping Bugs

- The use of acoustic eavesdropping "bugs" is common in espionage.
- For under $30, you can have a matchbox-sized battery-operated bug with built-in microphones and cellular network connectivity.
- These have traditionally been employed for eavesdropping on conversations, but they could now be used for cryptanalysis as well.
- Other traditional eavesdropping devices, such as phone bugs and laser microphones that can listen through windows from distance, might be repurposed as well.

Targeted Bugs

- Acoustic bigs can be buried where they will be placed in close proximity or contact with the computer for the greatest signal acquisition.
- When laptop computers are placed on a charging station, a presentation podium, or a crowded table, for example, they are placed in a pretty predictable way.
- An attacker may use this to set up a cave or a Kensington lock that is easily accessible to visitors and has hidden microphones that will be perfectly aligned when the laptop is plugged in.

Conclusion

- Side-channel assaults are a type of cryptanalysis technique that is widely used.
- Although less generic than traditional cryptanalysis, they are often far more powerful because they target a specific implementation rather than an abstract algorithm.
- Such assaults are applicable to the majority of current circuit technologies and should be regarded as a substantial danger to the security of real-world embedded devices.
- From an operational standpoint, security against side-channel assaults can be achieved through a well-balanced set of countermeasures.
- However, in order to adequately analyse the security of any cryptographic device and exchange it with implementation efficiency, significant attention must be made to the fair evaluation of these countermeasures.