Side-Channel Attacks

Nicholas Brown
07/17/2018

Side Channel Attacks are attacks based on side channel information. Side channel information is information that can be retrieved that is neither the plaintext, nor the ciphertext. Information such as the implementation of the system, timing, power consumption, electromagnetic leaks and even sound can be a source of side channel information.

1.0 Purpose

This document describes the most common Side Channel Attacks, timing attacks, simple power analysis (SPA) attacks, differential power analysis (DPA) attacks and differential fault analysis attacks. It also describes some methods that can be implemented to help mitigate, or prevent, these attacks.

2.0 Introduction to Side Channel Attacks

Encryption devices, or modules, are often perceived as a sort of black box that receives plaintext input and produces a ciphertext as output. In the past, attacks were therefore based on either knowing the ciphertext, both the ciphertext and plaintext or on the ability to define the plaintext and observing the results of the encryption. Today we know that encryption devices also have additional output and often additional input which is neither the plaintext, nor the ciphertext. These devices produce timing information, radiation of various kinds, power consumption information and more. Side channel attacks are attacks that make use of any of this information to recover sensitive information, such as the key the device is using.

Side channel attacks are of great concern because a lot of the attacks can be implemented quickly and with readily available hardware costing between a few hundred and a few thousand dollars. The amount of time required for the attacks depends on the type of attack. For example, SPA attacks on smartcards can take as little as a few seconds per card, while a DPA attack could take several hours.

We can consider the entire internal state of a block cipher to be all the intermediate values that are never included in the output. For example, DES has 16 rounds, so we can consider rounds 1 through 14 as a secret internal state. Side channels usually give information about these internal states, or the operations used in the transition of the internal state from one round to another. Of course, the type of side channel will determine what information is available about these states. The attacks typically work by finding some kind of information out about the internal state of the cipher, which can be learned by guessing part of the key and checking the value directly, and by some statistical property of the cipher that makes that checkable value slightly non-random.

3.0 Timing Attacks

Timing attacks focus on measuring the time it takes for an encryption device to perform operations in order to gain information about the secret keys. Careful measuring of the amount of time it takes to perform private key operations can allow an attacker to find fixed Diffie-Hellman exponents, factor RSA keys and break other cryptosystems. If the device is vulnerable, the attack is simple and usually only requires knowing the ciphertext.

Cryptosystems usually take slightly different lengths of time to process different inputs. Performance optimizations to bypass unnecessary operations, branching and conditional statements, RAM cache hits and instructions that run in non-fixed time (like multiplication and division) are just some examples. The encryption key and input data typically effect the performance characteristics of a cryptosystem. Simple timing characteristics may appear to reveal only a small amount of information from a cryptosystem; however, attacks do exist which can find entire secret keys from exploiting timing measurements.

By checking the correlations between time measurements, a statistical model can provide a guessed key bit with some degree of certainty.

Computing the variances is relatively easy and provides a good way to identify the correct exponent bit guesses. The properties of the noise and the signal determines the number of samples needed to gain a meaningful amount of information to allow the recovery of the key. The more noise there is, the more samples that are required. Error correction will increase the memory and processing requirements for the attack but can greatly reduce the number of samples required.

4.0 Power Consumption Attacks

Power consumption attacks are based on analyzing the power consumption of the device while performing the encryption operation. Either simple or differential analysis of the power consumed can allow an attacker to learn about the processes that are occurring and gain information that can assist in the recovery of the secret key, when used alongside other cryptanalysis techniques.

Integrated circuits are built out of individual transistors, which act as simple voltage-controlled switches. Applying, or removing charge to the gate causes current to flow across the transistor’s substrate. The current then delivers charge to other transistor gates, interconnecting wires and other circuit loads. The motion of this electric charge consumes power and produces electromagnetic radiation, both of which can be externally detected.

To measure the power consumption of a circuit, a small resistor is placed in series with the power or ground input. The voltage difference across the resistor divided by the resistance yields the current used. A well-equipped lab can have equipment that can digitally sample the voltage differences at very high rates with excellent accuracy.

4.1 Simple Power Analysis (SPA) Attacks

Generally, SPA attacks are based on looking at just the visual representation of the power consumption of the device while an operation is being performed. It involves direct

interpretation of the power consumption measurements collected during the cryptographic operations. This can yield information about the device's operation as well as key material.

SPA attacks work because the amount of power consumed varies depending on the instruction performed by the microprocessor. Larger features like DES rounds or RSA operations can be identified because the operations performed vary significantly during different parts of the operations. For example, SPA can be used to break RSA implementations by revealing the differences between the multiplication and the squaring operations. Also, some DES implementations have visible differences within permutations and shifts and can therefore be broken using SPA.

Since a SPA attack can reveal instruction sequences, it can be used to break cryptographic implementations where the execution path is dependent on the data that being processed, like DES permutations, comparisons and multipliers.

4.2 Differential Power Analysis (DPA) Attacks

Differential Power Analysis attacks are the harder of the Power Consumption attacks to prevent because they usually consist of not only visual but also statistical analysis and error-correction statistical methods to obtain information about the keys. DPA is typically comprised of data collection and data analysis stages that make extensive use of statistical functions for filtering noise, as well as for gaining some additional information about the processes performed.

Along with the large-scale power variations of the instruction sequences, there are effects correlated to the data values that are being manipulated, although these variations are usually smaller and can be overshadowed by errors and other noise. However, it is still possible to break the system by using statistical functions tailored to the target algorithm. Since DPA attacks automatically locate correlated regions in the device's power consumption, the attacks can be automated with little or no information about the target implementation.

Several improvements can be made to data collection and the DPA analysis processes to reduce the number of samples needed or to circumvent countermeasures. For instance, it can be helpful to apply corrections for the measurement variance, generating the significance of the variations instead of their magnitude. One variation of this is automated template DPA which can find DES keys using less than 15 traces for most smart cards.

For asymmetric ciphers, public key algorithms can be analyzed using DPA by correlating the candidate values for computation intermediates with power consumption measurements. It is also possible to test exponent bit guesses by testing whether the predicted intermediate values are correlated to the actual computation for modular exponentiation operations. By defining selection functions over the Chinese Remainder Theorem (CRT) reduction or recombination processes some CRT RSA implementations can also be analyzed. Generally, the signals leaded during asymmetric operations tend to be much stronger than those from some symmetric algorithms because of the higher computational complexity of the multiplication operations.

5.0 Differential Fault Analysis (DFA) Attacks

Fault analysis is the ability to investigate ciphers and extract keys by either generating faults in a system or by the natural faults that occur. Most often these faults are caused by changing the voltage, tampering with the clock or applying various types of radiation.

The attacks consist of encrypting the same plaintext, not necessarily known by the attacker, twice and comparing the results. A single bit difference indicates a fault in one of the operations and a short computation can be applied for DES, for example, to identify which round the error occurred in. An attacker can run a whole set of operations to recover a DES sub-key, which is also the sub-key of the last round. Then the attacker can either try guessing the missing 8 bits, 256 possible options, or simply subtract the last round for which they know the sub-key and rerun the attack on the reduced DES. The latter of which can also be used against Triple-DES.

6.0 Preventing Side-Channel Attacks

There are some known techniques that can be used for cryptographic modules to help increase the ability to defend against side-channel attacks.

6.1 General Mitigations Against All Attacks

6.1.1 Data-Independent Calculations

Operations performed by the module should all be data-independent in the time consumption. Input data or key data should not affect the time that operations take. Also, any sub-operations performed should take the same number of clock cycles.

Data-independent calculations prevents against all timing attacks because these attacks are dependent upon the variations in the computation times based on input and key bits. There is one property that will still have an effect on the timing of operations, the length of the exponent in exponentiation operations; however, this exposes no meaningful information and is usually known anyway.

6.1.2 Avoiding Conditionals Branching

Conditional execution of portions of code can reveal properties of the data if the attacker is measuring the time or power consumption to perform certain procedures. Calculations should be performed using elementary operations, like AND, OR and XOR, and not using branching and conditional executions of parts of code. This can make it very difficult to guess input or key values using a timing attack or power consumption. When all lines of code are always running irrespective of the input or key bits, the timing and power consumption will not fluctuate depending on the data and therefore protects against all timing attacks on asymmetric cyphers, as well as some power consumption attacks.

6.2 Mitigations Against Timing Attacks

6.2.1 Adding Delays

Designing a cryptographic module that makes all operations take the same amount of time is the most obvious way to protect against a timing attack, but this can be

difficult. Even if a timer is used to return results at a pre-specified elapsed time, the system responsiveness or power consumption can still have detectable changes with the operation finishes. Also, a fixed time cryptographic module is likely to be relatively slow since some performance optimizations cannot be used because all operations are required to take as long as the slowest operation.

However, when random delays are used, even though they do increase the number of ciphertexts required, an attacker can simply compensate by increasing the number of measurements. The number of samples needed roughly increases as the square of the timing noise, so random delays can still make an attack more difficult to perform, but still possible.

6.2.2 Timing Equalization of Multiplication and Squaring

If the amount of time the cryptographic module takes to perform multiplication and to perform exponentiation are set to be similar, attackers will not be able to determine how many multiplications and how many exponentiations are made. One way to accomplish this is to always perform both operations and discard the result of the unnecessary operation. Time equalization prevents timing attacks against the exponentiation operations of asymmetric encryption, which are the focus to the most common attacks.

6.3 Power Analysis Attack Countermeasures

6.3.1 Power Consumption Balancing

Dummy registers and gates should be added where useless operations are performed with the goal of reducing the power consumption to a constant value. Such as, whenever an operation is performed in the hardware, another is performed on a dummy element in order to maintain the total power consumption balance of the device. With the power consumption kept constant and independent of the input and key bits, all sorts of power consumption attacks, like SPA and DPA, can be prevented.

6.3.2 Addition of Noise

Another way to protect against DPA attacks is by introducing more noise into the power consumptions measurements. Adding noise increases the number of samples required for an attack; however, if the increase in noise is high enough this could make the sampling unfeasible due to the number of samples required. The main goal should be to add enough noise to stop an attack while adding only minimal overhead.

6.4 Fault Attack Mitigations

6.4.1 Encrypting Twice

Running the encryption twice on the same input and only output the results if the two results are identical is one possibility in preventing DFA attacks. However, this does increase computational time and the probability that the fault will not occur twice is not adequately small enough. Because the fault may still occur twice, this mitigation

only makes the attack harder to implement by requiring more samples, but not impossible.