Side-channel analysis: why it matters?

Side-channel attacks are a physical type of threat that aims to recover sensitive data manipulated by your code without using cryptanalysis, or taking advantage of software vulnerabilities.

In practice, algorithms are implemented and run on real-world devices (your laptop, your server, your smartcard, your phone…). In other words, code needs a hardware reality in order to function. Naturally, physical devices have physical properties. For example, your processor will take an amount of time to process data, will consume more or less power, emit electromagnetic waves, produce heat, or even sound as it operates.

All these physical effects are the direct result of the operations being performed, and on the underlying operands the device may be manipulating. It is possible to passively observe these physical phenomena, and use them to deduce the data being processed. This kind of attack is referred to as a side-channel attack, or side-channel analysis (SCA).

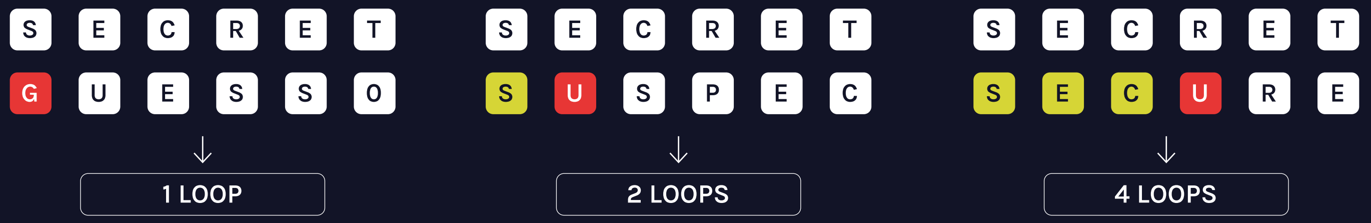
Researchers and practitioners have repeatedly demonstrated the practical threat of SCA - compromising cryptography libraries, FIDO devices, cryptocurrency wallets, smart cards, TPMs, IoT devices, and smart vehicles.

Timing analysis

This function compares a user input with a secret password. It does so character by character, and makes use of an early stopping mechanism whenever it encounters a mismatch. However, the problem with this early stopping is that it introduces a side-channel weakness: the function execution time depends on the secret. Indeed, the number of executed loops is directly related to the number of leading characters that are correct in the user input. The number of executed loops has a direct influence on the running time of the function.

```
bool check_pwd(str secret, str input){
    for (int i=0; i < secret.len(); i++){
      if (input[i] != secret[i]){
         return false;
      }
    }
  return true;
}
```

Instead of having to search for the whole space of possible passwords, a smart adversary could submit different inputs while varying the first letter, and measure the execution time. The submission producing the longest running time will correspond to the instance where the input contains the correct first letter of the password. The function can then be called again, this time varying the second letter, and so on until full recovery of the password has been achieved.

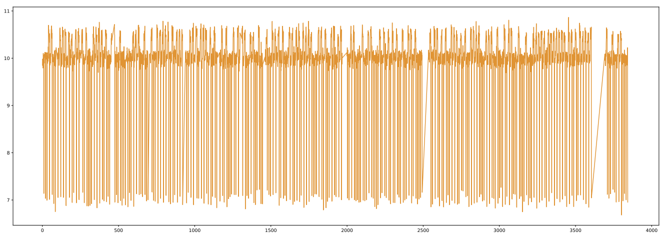


The property that this function did not verify is called constant-time. Non-constant time functions and mechanisms allow for timing attacks, and have been the root cause of many security issues in recent years. For example, Spectre, Meltdown, Lucky13, or TPM-fail attacks: all of them exploited constant timing issues.

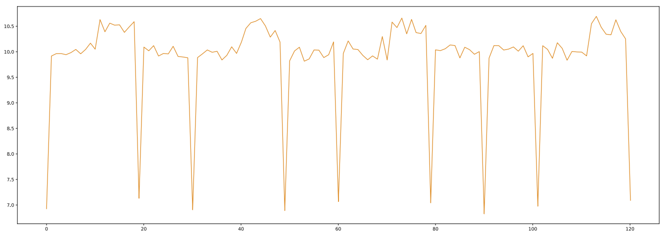
Power analysis

With physical access to the device running code, an attacker can use cheap tools to measure power consumption or electromagnetic emanations, and get a very clear visualisation of the running algorithm.

Let's consider a device running an ECDSA signature. The figure below shows a typical reading of power consumption, as viewed through an oscilloscope.



At first glance, we only see peaks and troughs, and the detail of what's occurring could be unclear. But let's zoom in to a small section, and try to make sense of the apparent chaos.

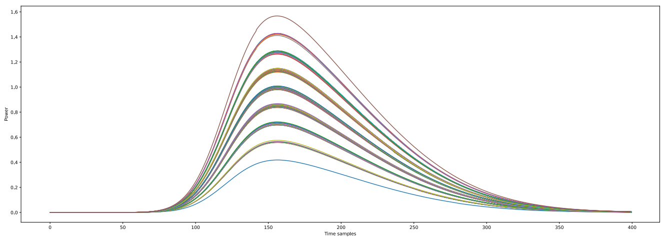


In this view, the trace is marginally noisy, but we can now clearly distinguish two types of pattern: the small, narrow signals, and the taller, wide peaks in consumption. These two patterns correspond to a different operation being performed. In the case of an operation on an elliptic curve, we can simply infer that one of these patterns corresponds to a point-doubling, and the other one to a point-addition. This knowledge allows for a direct recovery of the secret data being manipulated: the small part of the curve above shows a sequence "wide-narrow-wide-narrow-wide-narrow-narrow-wide" which can easily translate as the bit sequence "1-0-1-0-1-0-0-1".

As we just saw, simple power analysis can be a very powerful tool to distinguish between operations. In some cases, such as the example of ECDSA, this can lead to a trivial recovery of the secret. In some other cases, this may serve as an effective means of reverse-engineering.

However, we can also go one step further and distinguish between the values themselves. The following figure illustrates how different values actually have different power consumption leakages whenever manipulated by a smart card, as was originally observed by Kocher et al. in 1999.

State of the art techniques can exploit this relationship between leakage and manipulated data. Simple observations in early discoveries led to statistical correlations, and in recent years the use of AI and machine learning could be used more effectively to determine these values.



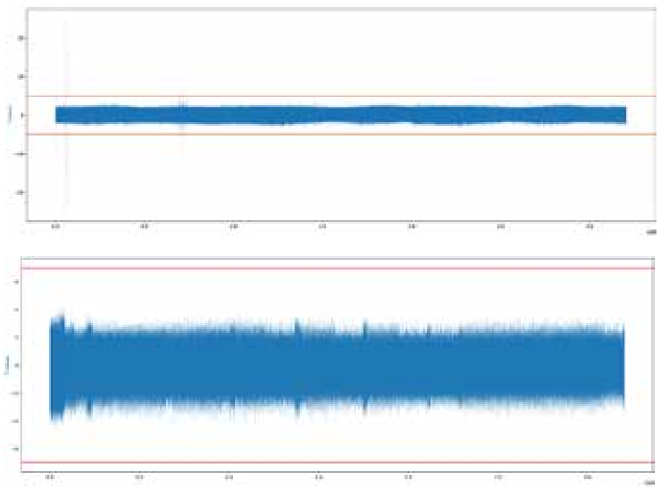
State of the art techniques can also exploit this relationship between leakage and manipulated data. For example, simple observations in early discoveries led to statistical correlations, and in recent years the use of AI and machine learning.

Leakage detection

To obtain a certain level of security against these attacks, it is important to ensure that observed leakage does not relate to manipulated sensitive values. This can be achieved by several methods (masking, hiding), all of which have been well studied.

It is extremely important to verify this independence in practice. To this end, PQShield uses a methodology defined in ISO17825 and dubbed TVLA (Test Vector Leakage Assessment). In short, this approach comprises the collection of a large number of traces under different values, and the computation of Welch's t-test on this data. TVLA detects statistical dependencies between observations and underlying data with a strong confidence.

The following figures illustrate the TVLA results of two ML-KEM (Kyber) implementations. The first one is unprotected and clearly shows peaks indicating dependencies at several time indices which might be exploited by attackers. The second example uses countermeasures that prevent observing these dependencies.



At PQShield, we constantly push boundaries to make sure that all our products execute in constant time, and are automatically tested using the TVLA methodology. We also use extensive fuzzing, mount bespoke SCA key recovery attacks, and test for fault injection resistance. This allows us to offer our products at the best security protection level - Cloud, Edge, Government - for any application.

PQ SHIELD

Ready to learn more?

Get in touch: contact@pqshield.com | www.pqshield.com

Products

Publications

Careers