Defending against side-channel attacks - Part I

Gilbert Goodwill, Cryptography Research, Inc. - March 06, 2013

***Editor's Note:** This article was originally presented at ESC Boston 2011.*

Part One of this three-part series provides a brief introduction to side-channel analysis, including timing analysis and simple and differential power analysis (SPA and DPA). The article also appeared for the first time on EE Times' Military and Aerospace Designline.

1. Introduction
Cryptography is a fundamental building block for securing systems and communications and is widely deployed in embedded systems used for commercial and defense applications. Basic cryptographic operations such as encryption/decryption, message-authentication and digital signatures rely on secret keys that must be kept securely within a device and protected from disclosure. Modern cryptographic algorithms, when used with appropriate-sized keys, are designed to resist all known attacks where the attacker can observe (or manipulate) the inputs or outputs of the algorithm, but does have any other information about the secret key or about the execution of the algorithm.

In practice, however, an attacker who has access to a device that is performing a cryptographic operation can easily obtain additional information about the operation, beyond just the inputs and outputs. For example, even a remote attacker can obtain a (noisy) estimate of the time taken to perform cryptographic operations. An attacker who is physically close to the device could also measure the power consumed by the device or its EM emissions while it is performing the operation. These additional sources of information about cryptographic operations are known as side-channels, and in the mid-1990s Kocher et al [1,4] showed that side-channels such as timing and power consumption contained enough information to easily extract the secret key from naÃ¯ve implementations of all cryptographic algorithms. They also proposed several fundamental techniques for protecting cryptographic implementations from such attacks.

Subsequently, substantial R&D activity has been directed towards understanding side-channel attacks and implementing defenses. Many industry and government standards as well security certifications now require tamper resistant devices to defend against side-channel attacks. Non-invasive side-channel attacks such as timing attacks, and simple and differential power analysis (SPA and DPA), should be addressed by all systems that require any significant degree of tamper resistance since these attacks can be carried out by attackers with modest skill and resources, and timing and power measurements can be collected easily.

This paper provides a brief introduction to side-channel analysis, including timing analysis and simple and differential power analysis (SPA and DPA). It then discusses CRIâ€™s recent side-channel analysis of popular mobile devices, in which cryptographic keys are extracted from the devices using EM emissions from the processor as it performs certain cryptographic calculations. (These are unintended emissions from the devices, and not related to the emissions from the devicesâ€™ ordinary communications channels.) Also, we propose a new suite of standardized tests intended to help analysts look for potential problems in their devices. These tests have been designed to enable consistent testing by validation labs, as well as help developers find problems in their devices without the need for custom tests. **2. Side-channel analysis**
This section provides a very brief introduction to some of the more commonly exploited side-channel analysis techniques. While the majority of the examples in this paper use power analysis, the techniques presented are largely agnostic to how the data is collected. Other physical measurements used in side-channel analysis include, but are not limited to, RF signals and E-Field data. Other sources such as sound, heat and photon emissions have been proposed and researched.

2.1. Timing attacks
Timing attacks exploit small differences in execution time to extract secret information from systems. Commonly discovered sources of timing leaks include, but are not limited to:

- Data dependent differences in instruction times;
- Early exit;
- Data dependent code branches;
- Cache access times.

A very commonly seen timing attack exploits the early exit from loops comparing passwords or message authentication codes (MACs). This allows the attacker to use the verifier as an oracle to extract the secret key. Although these attacks are decades old, they are still found in systems that are both currently deployed and still under development.

There are many examples of practical timing attacks against cryptographic algorithms. Timing attacks against standard cryptosystems such as Diffie-Hellman, RSA, and DSS were introduced by Kocher in [**1**]. Cache collision timing attacks against AES executing on modern processors were demonstrated by Bonneau and Mironov in [**2**]. Brumly and Boneh [**3**] demonstrated that practical remote timing attacks against networks were possible. These examples show that cryptographic algorithms and protocols may be vulnerable to timing attacks, even when operating on modern hardware.

2.2 Simple power analysis
Simple power analysis (SPA) is a set of techniques for analyzing large-scale changes in power consumption which occur due to changes in the algorithm being performed or the data being processed. It is often used to help analyze devices by:

- Determining what operations are taking place;
- Determining timelines for the sequence of operations performed;
- Identifying changes in power consumption due to changing operation or data;
- Extracting secrets non-invasively.

SPA leaks can usually be exploited using just one or a few power measurements. In general, the same sources of timing leaks also result in SPA leaks, since power traces can be used to determine the precise timing of the different operations.

Public key algorithms tend to be particularly susceptible to SPA, since the big-number arithmetic required by these algorithms often uses a lot of power relative to other operations, and can vary greatly with the data being processed. For example, the most straightforward method for implementing a modular exponentiation is a loop similar to the following:

```
Algorithm MODULAR_EXPONENTIATION (exponent e, base b, modulus n)
     Let A = 1
     Let X = b mod n
     Let k be the number of bits in e.  Then,
     For i = k downto 0, do
          A = (A * A) mod n                          ; SQUARE
          Let e_i be the i'th bit of exponent e      ; bit 0 is LSB
          if (e_i == 1), do
               A = (A * X) mod n                     ; MULTIPLY
          Done
     Done
     Return A
End Algorithm
```

Click on image to enlarge.

Figure 1: Straightforward implementation of a modular exponentiation

Note that in the above loop, a modular square is always performed, but whether a modular multiplication occurs depends upon the current bit of the exponent being processed. ***Figure 2*** below is a power trace obtained from a device performing modular exponentiation using the RSA decryption exponent.

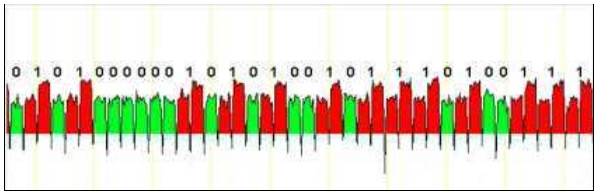

Figure 2: Power trace of device performing modular exponentiation using RSA decryption exponent. In this device, the squaring operation was an optimized version of the general multiplication routine, leading to an SPA attack.

In this device, the squaring routine is an optimized version of the general multiplication routine. As a result, the squaring routine uses less power than the general multiplication routine. Combined with the fact that a square can follow either a square or a multiply, but a multiply must always follow a square, it is then a simple matter to read off the secret decryption exponent from a single trace.

Section 3 shows how this type of attack works against cryptographic applications running on several different popular mobile devices. **2.3. Differential power analysis (DPA)**
Differential power analysis (DPA) is a set of techniques for analyzing changes in power consumption that are too small to be observed directly. It was first published in the late 1990's by Kocher, Jaffe, and Jun in [**4**], and has become one of the primary tools for non-invasive analysis. There is now a very large body of literature on the subject.

To give a brief example of the how DPA can be used, ***Figure 3*** below shows a power trace from a device performing a single AES encryption. If multiple traces were collected, data dependent differences in the traces would be too small to observe directly.

Figure 3: Power trace of device performing a single AES encryption.

However, suppose many traces are collected from a device performing AES on random input data, and the traces partitioned into 2 sets. In the first set, bit 18 of the input (18 is an arbitrary choice here) is 0, and in the other set bit 18 is 1. Since the input was random, the two sets will each contain about half of the original traces. Now compute the average of the traces in each set. Visually, the resulting power traces would look the same as the trace in ***Figure 3***. However, as ***Figure 4*** below shows, taking the difference between the averages of the two sets reveals the influence of bit 18 on the power consumption of the device.

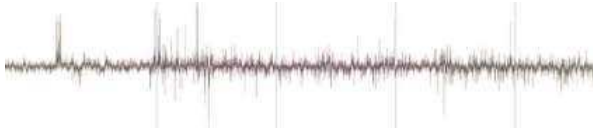

Figure 4: Difference between averages of AES power traces sorted on input bit 18 vertically magnified by a factor of 25. The spikes show the influence of bit 18 on the power consumption of the device.

Similar techniques can be used to extract secret keys. For example, consider the construction shown in ***Figure 5*** below.

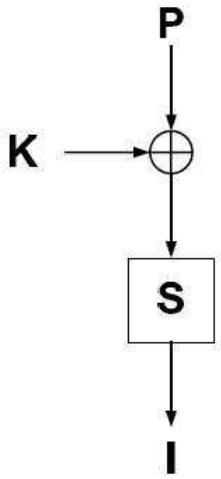

Figure 5: Common construction in many modern block ciphers.

In the above construction, a sub-block of plaintext P is XORed with a sub-block of key K, and the result is transformed by the nonlinear function S, which outputs intermediate value I. This type of construction appears in many modern block ciphers. In AES, P, K and I are all 8-bit bytes, and S is an invertible nonlinear lookup table. The small size of the sub-blocks helps enable a divide and conquer strategy in which individual sub-blocks of key are extracted independently until the entire key is recovered.

To see how the basic DPA attack proceeds, consider an attack against AES-128 in which a device encrypts a moderate number of, say 1000, random blocks of plaintext, during which the power consumption is monitored. Then the key can often be extracted using the following steps.

1. Select a byte of key to recover.
2. Select a bit of the intermediate I (say bit 3) on which the traces are to be sorted.
3. Guess a value for the byte of key.

4. For each block of plaintext:
a. Compute the value of I using the (known) plaintext and (guessed) byte of key.
b. Assign each power trace to one of two sets, depending upon whether bit 3 of I has the value 0 or 1. Since the plaintext is randomly chosen, there should be roughly 500 power traces in each set.

5. Average the power traces in each of the two sets, and compute the difference of the two.
6. Looks for large spikes in the difference trace, which shows that the influence of the value of bit 3 of I on power consumption. See ***Figures 6*** through ***8*** below for an example difference traces for correct and incorrect key byte guesses.
a. If large spikes are not present, return to step 3 and guess a different value for the key.
b. If no guesses for the key byte result in large spikes in the difference trace, try sorting on a different bit of the intermediate I. It is often the case that some bits will exhibit stronger leaks than others.
7. If a key byte recovered in step 6, go to step 1 and select a new byte of key to recover.

Figure 6: Mean trace.

Figure 7: Difference trace for correct key byte guess.

Figure 8: Difference trace for incorrect key byte guess.

Part One discussed a basic DPA attack against AES. Many different variations of DPA attacks exist, depending upon the type of algorithm and how it is being implemented. **Part Two** will discuss a DPA attack against AES using EM emissions from the devices.

References

[1] Paul Kocher, "*Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*", Advances in Cryptology – Crypto '96 Proceedings, Lecture Notes in Computer Science Vol. 1109, Neal Koblitz (Ed.), Springer-Verlag, 1996, pp. 104–113.
[2] Joseph Bonneau and Ilya Mironov, "*Cache-Collision Timing Attacks against AES*", Cryptographic Hardware and Embedded Systems – CHES 2006, Lecture Notes in Computer Science, Vol. 4249, L. Goubin and M. Matsui (Ed.), Springer-Verlag, 2006, pp. 201-215.
[3] D. Brumly and D. Boneh, "*Remote Timing Attacks are Practical*", Proceedings of the 12th USENIX Security Symposium, August 4–8, 2003. (Paper available at **http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf**).
[4] Paul Kocher, Joshua Jaffe, Benjamin Jun, "*Differential Power Analysis*," Advances in Cryptology - Crypto 99 Proceedings, Lecture Notes In

Computer Science Vol. 1666, M. Wiener, (Ed.), Springer-Verlag, 1999, pp. 388–397. (Whitepaper available at **http://www.cryptography.com /resources/whitepapers/DPATechInfo.pdf**)

See related links:
Using MISRA C and C++ for security and reliability. Part I
Using MISRA C and C++ for security and reliability. Part II
Using MISRA C and C++ for security and reliability. Part III
How secure is AES against brute force attacks?
Public key cryptography and security certificates

If you found this article to be of interest, visit Military/Aerospace Designline where you will find the latest and greatest design, technology, product, and news articles with regard to all aspects of military, defense and aerospace. And, to register to our weekly newsletter, click here.