# Hardware Attacks on Cryptographic Devices

## *Implementation Attacks on Embedded Systems and Other Portable Hardware*

Jem Berkes

University of Waterloo

Prepared for ECE 628, Winter 2006

## 1. Introduction to hardware attacks

Most research in cryptography examines the mathematics of cryptographic algorithms, ciphers, and protocols. Since the classical focus of cryptography has been communication security, more attention has been given to attacks on the information flowing over a channel rather than the endpoint hardware. However, the hardware which implements the cryptography is of course an important part of the overall system and deserves just as much scrutiny.

Rapid advancements in computer hardware technology have resulted in smaller, faster and cheaper devices which have become commodity items. As a result, cryptographic hardware is now widely deployed in everything from pay TV units to cell phones to prepaid cards and access cards (smartcards). These devices often store private keys or other sensitive data, so compromise of this private data or the hardware which guards it can have disastrous implications including loss of privacy, forged access, or direct monetary theft.

Because the cryptographic devices are easily obtainable, attackers can study the internal structure of the hardware to learn specific details about the implementations. Knowledge of the implementation may then be used to carry out attacks on the system without directly attacking the mathematics of the algorithms. In other words, even when totally secure algorithms and protocols are employed, the attacker might still be able to learn valuable secret information due to the implementation of the hardware. Even if secret information can not be learned, attackers may be able to disrupt the hardware or deny service leading to other kinds of failures in the security system.

This paper examines practical "implementation attacks" on cryptographic

hardware, with a focus on embedded systems and portable hardware. Attack feasibility and difficulty, along with basic countermeasures are also compared.

## 1.1. Context: embedded systems and portable hardware

While there are many kinds of computer hardware that use cryptographic processing, embedded systems and portable hardware pose some unique challenges. In the following discussion of attack styles, embedded systems and other small, portable hardware will be the focus. Consider the following two examples of hardware which have been the targets of implementation attacks:

Smartcards. Thin credit card-like cards with embedded ICs. The cards do not carry their own power source, as the contacts on the card allow the card readers (ATMs, pay telephones, Points of Sale) to both power and communicate with the card. The cards typically have sensitive information such as private keys in non-volatile storage, and communicate with a card reader using standard protocols to encrypt and authenticate.

Cell phones and PDAs. These devices have more computational power and wireless communications capabilities. In order to obtain network service, they must authenticate securely over an insecure and easily manipulated channel. The devices often store and communicate private information belonging to consumers, service providers and manufacturers.

The noteworthy aspect of security as it relates to embedded systems and portable hardware is the extremely hostile environment in which the hardware is used. The designer can not assume any physical security exists, as is the case with most other kinds of computer hardware. Not only can the hardware itself fall into the hands of an attacker, but other computer equipment which connects to the embedded system might be under the control of an attacker. For instance, a smartcard carrying financial information might be connected to a card reader (for instance, Point of Sale terminal or ATM) that is under the control of an attacker.

The hostile environment is made more complicated by the fact that there are potentially numerous attackers or threats depending on the viewpoint. In some applications (for instance, multimedia content distribution) the customer and owner of the device is treated as a threat since they may want to use digital content in a way that is not permitted. In the application of smartcards for financial transactions, neither the holder of the card nor the card reader can be trusted by the bank. A cellphone provider's primary concern is restricting access to paid subscribers, while the end user's concern of communication privacy is a

different consideration completely.

The way in which embedded systems are deployed for commercial use also adds a practical complication for security. Because much of the hardware (for instance, card readers) have already been deployed, constraints on backwards compatibility mean that users often have to settle for less-than-optimal security. While crypto algorithms and protocols continually evolve, it is no simple matter to deploy millions of new embedded units to subscribers.

Finally, the hardware resource limitations of embedded systems lead to difficult security design considerations. Because of the restrictions on size, cost, and battery power, these computers have limited computational power and storage space. The software which implements cryptography has to be efficient and fit in minimal storage. Because cryptographic algorithms are very power-hungry, designers do not have the freedom to implement very computationally intensive crypto processing. There is a tradeoff between computational security requirements and the resources of the embedded system. Figure 1 illustrates the serious energy burden of cryptography in an example wireless device.

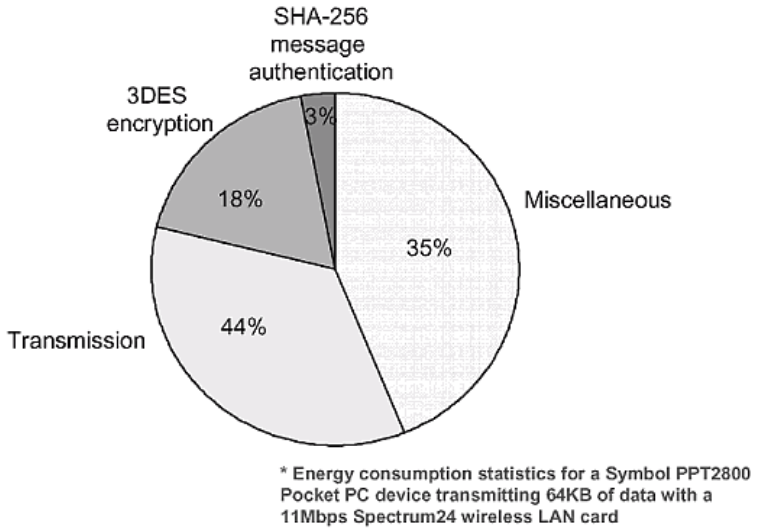


*Figure 1: Energy consumption profile from a sample wireless device (source: Karri and Mishra, 2002)*

## 1.2. Degree of security

The types of implementation attacks which will be discussed do not involve breaking mathematical algorithms. While weaknesses in common algorithms undoubtedly exist, the implementation attacks exploit other weaknesses which are easier targets. So while overall security can be compromised by the "weakest link", including weak math, the security of underlying algorithms and mathematical theory will be omitted from this discussion.

The focus of implementation attacks is on physical security of the device. Unfortunately, there can not be total security when the cryptographic hardware (and the keys that are stored in it) are physically in the hands of an attacker. While cryptographic hardware such as smartcards are manufactured to be tamper resistant, they can never be totally tamper proof. Given sufficient resources, an attacker who has access to semiconductor test equipment can directly observe the private information stored on the device. The cost might be millions of dollars, making this option infeasible for the majority of attackers.

The degree of security, within the scope of this discussion, therefore refers to the time and cost (difficulty) of attacking the system. Generally speaking, less invasive attacks are easier to carry out and pose more of a threat in practise.

## 1.3. Goals of an attacker

The goals of an attacker can vary substantially depending on the particular application. We can simplify the goals down to three categories: (1) learning secrets, (2) taking advantage of victim hardware without learning private information, (3) disrupting normal operation or denial of service.

Compromising private information, usually secret keys, has the most serious implications and has been the focus of much research. In a secure cryptosystem, an attacker should not be able to learn anything about the secret key or message by observing or manipulating the inputs/outputs to the "black box" crypto unit. The only method available to the attacker must be an exhaustive search of the full key space. If any additional information is available which reveals even partial information about the secret (for instance, the first few bits) then the security has been weakened, as an attacker's search has become simplified. Learning just one bit of a secret key halves the search time!

The attacks described below are mostly of this nature. For example, physical tampering of bits stored in EEPROM memory (invasive) might reveal

some portion of a secret key and dramatically reduce system security. Other non-invasive techniques, such as measuring power consumption of the device over time, might reveal partial information about the key. Such "side-channel attacks" exploit the fact that some clues about the secrets are revealed through a side-channel (power consumption, electromagnetic signals, or even sound). If any clues relating to the secret are learned, the attacker has an advantage.

## 2. Types of hardware attacks

The following discussion examines three categories of hardware attacks on embedded systems: side-channel attacks, fault attacks, and physical tampering. While an attacker may have any number of goals in practice (see 1.3) the literature in this area typically emphasizes key compromise. After introducing the attacks, the difficulty from the attacker's perspective and countermeasures will be examined in section 3.

Note that all of the described attacks are of a practical nature, leading to compromise of commonly used cryptographic hardware. Several example attacks are provided to illustrate by example, but many details have been omitted for brevity in this broad review.

### 2.1. Side-channel attacks (non-invasive)

A side-channel refers to an avenue of information provided by the implementation of cryptographic hardware. Examples of side-channels are sound, infrared radiation, time delays, power consumption, and electromagnetic radiation. This "leaked information" may be statistically related to the underlying computations or keys, giving clues that are useful to an attacker. Unlike the more invasive techniques that will be covered later, these side-channel attacks can be carried out on unmodified hardware while it is performing normal computations.

### 2.1.1. Side channel attacks: Timing attacks

This simple side-channel attack, documented by Kocher in 1996, demonstrates how measuring computation time can reveal vital information about keys. Kocher's attack shows how measurements of the time required to perform private key operations can reveal fixed Diffie-Hellman exponents, RSA factors, and other secret parameters of cryptosystems. It is assumed that the attacker knows implementation details of the cryptosystem, and the attack is highly dependent on the specific implementation. One simplified example is presented here for a modular exponentiation algorithm used in RSA computations, with $m = c^d \bmod n$ where the attacker wants to find the private key *d.*

The attack can be crafted to exploit any implementation that does not run in fixed time. For example, a modular exponentiation algorithm which tests one bit of a key and branches to either a fast operation (if key bit is 0) or a very slow operation (multiplication if key bit is 1) reveals information about the secret key along the way.

An attacker can start by guessing the first key bit as 0 or 1, and seeing which hypothesis results in the strongest correlation between predicted and actual computational time. This is then repeated until the attacker learns all the key bits by observing the strongest time correlations among many samples and key bit choices. At the very least, the search space for possible keys is reduced. In Kocher's experiments with the RSAREF software toolkit on a 120-MHz Pentium computer, only a few thousand trials revealed the 256-bit secret exponent used in modular exponentiation. The attack was described as "computationally quite easy".

### 2.1.2. Side channel attacks: Power analysis

The category of power analysis attacks (which includes simple power analysis and differential power analysis) involve physical measurements, via probe, of the device's current consumption versus time. The power analysis attacks rely on a correlation between the current being drawn by the processor and the instructions or data being processed. Assuming a CMOS implementation as shown in Figure 2, these changes in current result mainly from the charging and discharging of a load capacitance during switching.

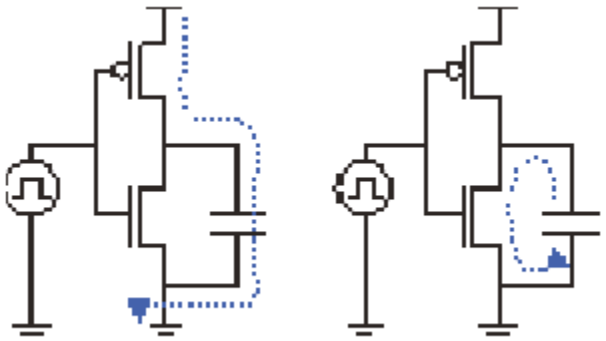

*Figure 2: Current through CMOS inverter (source: Peeters et al., 2006)*

In Simple Power Analysis (SPA) attacks, the attacker observes the trace of current consumption over time and tries to directly apply it to the underlying cryptographic processing. Figure 3 shows a SPA trace from a smartcard performing a DES operation. The 16 rounds are clearly visible in this device, and higher resolution traces would even reveal additional characteristics such as register rotations or even conditional jumps (not shown).

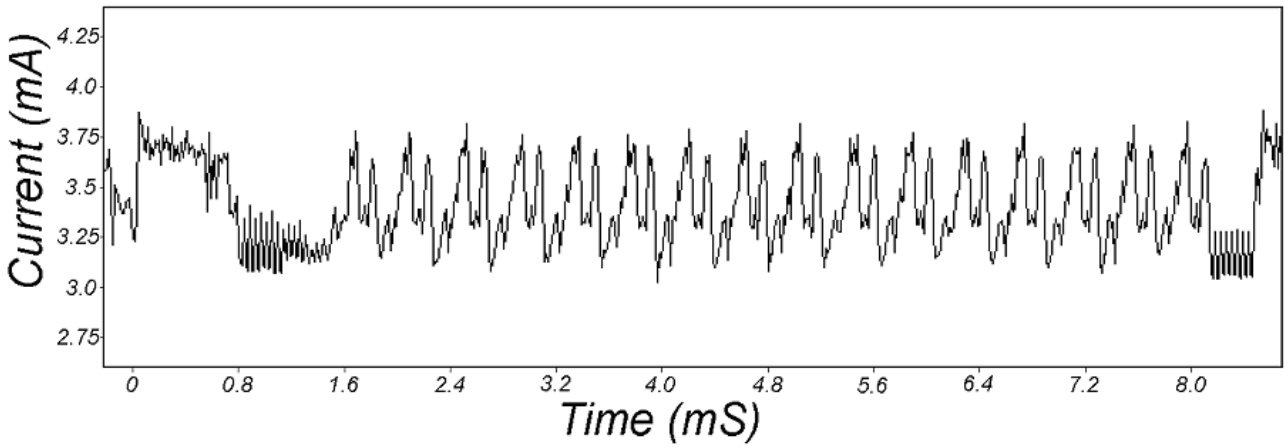


*Figure 3: SPA trace from typical smartcard showing 16 rounds of DES operation (source: Kocher et al, 1999)*

Consider an example analogous to the timing attack described in 2.1.1. In the timing attack, the time taken to compute revealed information about branch choices, and therefore secret key bits. In the SPA example shown in Figure 4, the two traces are from identical computations until a branch decision which is visible in cycle 6. This reveals information about the sequence of instructions being executed, and could (depending on implementation) lead to key compromise. This is a more detailed view of the Figure 3 trace, showing

individual clock cycles on the processor.

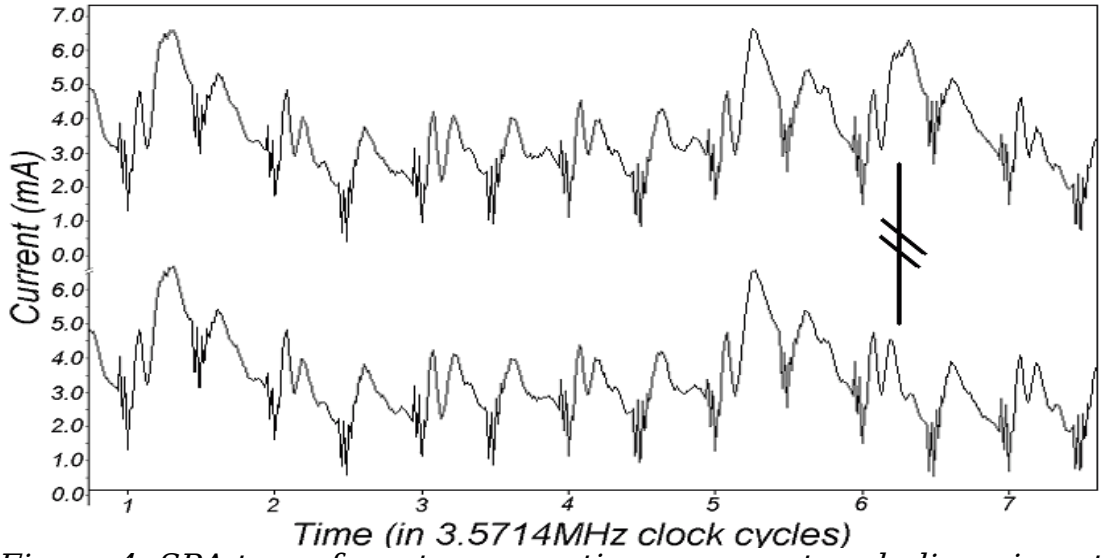


*Figure 4: SPA trace from two executions on smartcard, diverging at cycle 6 upon conditional branch (source: Kocher et al., 1999)*

SPA attacks have been used to break implementations of RSA by showing the differences between multiplication and squaring operations used in modular exponentiation. SPA attacks can also provide other clues to attackers to augment other attacks, even if the SPA attack alone can not directly reveal a secret key.

A more powerful kind of power analysis attack is Differential Power Analysis (DPA) which relies on statistical tests to isolate a signal of interest from noisy and complex power signals on a device. The power of DPA lies in its ability to discover useful information whenever there is a correlation between power traces and processed data, even if the relationship between power use and instruction execution is too complex to link directly (as is done with SPA). Although the attacker does not need to know details of how an algorithm is programmed, they still must know which algorithm to attack since the differential attack requires a known model of cipher behaviour. Using a DPA attack, the attacker can discover a group of key bits at once and dramatically reduce the key search space.

The attacker first collects a large number of power traces for thousands of encryptions using high speed equipment such as modern digital oscilloscopes with high speed A/D capture. The attacker makes a hypothesis about the key (for instance, guesses a subset of the key bits in a certain round) and uses this hypothesis to calculate the corresponding bits in the "next stage" of computation. (This could be the inputs to the next round, in a Feistel-class cipher). If this hypothesis is correct, the corresponding bits at the "next stage" will be as

predicted. If the hypothesis is incorrect, the resulting bits will appear to be random and therefore only match the predicted bits in roughly 50% of test cases.

Various methods can then be used to statistically compare the prediction to measurements. This allows an attacker to discover when their hypothesis is correct. Take for example the original DPA attack described by Kocher, Jaffe, and Jun is to find DES keys on smartcards. The attacker guesses the 6 bit input into s-boxes at a specific round and calculates the resulting 4 bits in the next round.

In this attack, some arbitrary bit from the 4 bits in the next-round is used to divide the collected power traces into two subsets. If the original guess was wrong, then the two subsets have been randomly selected and the mean power will be no different from the main set. However, if the original guess was correct, then the choice of subsets correlates to the actual computation. When the difference between the average trace of the two subsets is plotted, there will be a statistically significant peak where the target bit has an effect on power consumption. Measurement errors and power consumption due to other operations on the chip are uncorrelated.

Figure 5 shows four traces from a DPA attack on a smartcard running DES. On top is the reference power trace (as in SPA). The three traces below this are differential traces computed as described above, from two subsets formed by a guess of subkey bits. Of these three guesses, the first is correct and the other two are incorrect. From only 1000 samples and even without further statistical processing, the signal (correct guess) stands out among the noise.

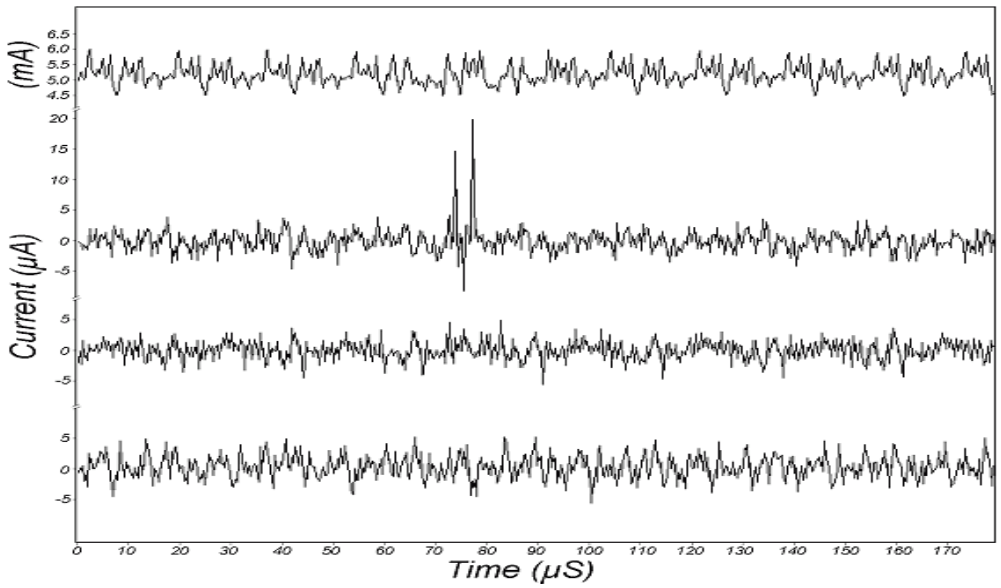


*Figure 5: DPA traces against top reference, showing one correct guess out of three (source: Kocher et al, 1999)*

Once the attacker learns one part of the key, they can proceed to testing other parts of the key or switch to an exhaustive search once the keyspace becomes feasibly small. This type of attack is particularly potent because, other than some basic knowledge of the encryption algorithm, the attacker requires little or no knowledge of the target implementation. A search for correlation between power traces and internal data can be automated, and more samples can always be taken to compensate for noise or measurement error.

### 2.1.3. Side-channel attacks: Electromagnetic analysis

While power analysis attacks (SPA and DPA) are based on measured power consumption, electromagnetic (EM) attacks are based on measured electromagnetic signals due to currents flowing in microelectronics. Once the measurement is acquired, the types of attacks are very similar to power analysis attacks and can generally be called Simple Electro-Magnetic Analysis (SEMA) and Differential Electro-Magnetic Analysis (DEMA).

There are some notable differences between power and EM attacks, however. While power analysis is generally limited to measurements of overall device power consumption, electromagnetic analysis can target specific areas of the chip by positioning a small antenna. EM attacks can also be carried out by an attacker that is far from the hardware, meaning physical access it not strictly required. Early attacks could use simple AM demodulators even a few meters away from the chip, for instance.

On the other hand, without the luxury of direct probes connected to physical wires, EM attacks suffer complications due to noise, RF interference, and measurement error.

## 2.2. Fault attacks (semi-invasive)

A variety of fault attacks exist, where some hardware fault (an unexpected condition or defect) leads to a processing mistake that is beneficial to the attacker. Fault attacks might overlap with physical tampering. Methods of inducing faults include: supplying noisy power or clock signals, incorrect voltage, excessive temperature, radiation or high energy beams such as UV, laser, etc.

Some algorithms are particularly vulnerable to hardware faults. For

instance, an RSA implementation using the Chinese Remainder Theorem (CRT) has frequently been the target of experimental fault attacks. If a smartcard computes an RSA signature modulo n = p*q by first computing it modulo p and q separately and then using CRT to combine the results, the overall computation depends critically upon the sub-computations. If there is a computational error in either, an attacker can then easily factor *n* and compromise the private key.

If *e* is the public exponent, and the generated RSA signature $S = M^d \pmod{pq}$ is correct modulo p but incorrect modulo q due to a hardware fault, then the attacker can compute $p = \gcd(n, S^e - M)$

Two other practical fault attacks are described below.

Pay-TV smartcard attack: The CPU can be made to execute wrong instructions by applying a rapid transient to the clock or power supply. A portion of code is targeted; in the example described by Anderson and Kuhn, the critical code is a loop which writes the content of a limited memory range to the serial port. Systematically, faults are introduced until the CPU fails to properly execute a conditional branch. As a result, the entire contents of memory can be dumped to the serial port, allowing an attacker to learn all smartcard contents. This is an interesting demonstration of inducing errors in instruction code, rather than data.

Attacks on DES: If the attacker can choose to make a specific instruction fail, then the normal execution of DES can be corrupted to the attacker's benefit. For instance, the number of DES rounds can be reduced by corrupting the appropriate loop variable or conditional jump. The attacker can then learn the DES key by inspection.

## 2.3. Physical tampering (invasive)

Attacks that involve physical tampering are far more invasive than the previous attacks, and generally require more expensive equipment to carry out. Methods that involve probing the electronics of the device first require chip depackaging, to remove protective layers and gain access to chip internals. Part of the difficulty with any physical attack is determining the precise location of important units such as memory, registers, etc.

If many implementation details are known, such as the exact code executed

by the processor and the location of instructions in ROM, then the behaviour of the device can be modified by selectively overwriting the device's ROM. Using a laser cutter microscope, bits can be modified such that vital instructions are altered to produce desirable affects (such as skipping rounds of DES). In the case of EEPROM, two microprobing needles can be used to set or clear a target bit.

Another approach is using the above probing tools to modify data, instead of code. Assuming the location of the key is known in EEPROM, the probes can still be used to determine the key even if the values can not be directly read. The approach would be to arbitrarily set the first bit to a 1 or 0. Operating the device, the attacker can observe whether the behaviour is the same or different than before. If the operation is the same, the key bit was guessed correctly and if the operation fails, the key bit was the opposite. Setting the key bit back to the correct value and continuing, an attacker can eventually learn the entire key.

Finally, other low-level hardware observation can reveal useful information. It turns out that both static and dynamic RAM “remember” values that have been stored for a long time, even after power is cut-off (memory remanence). Even though tamper resistant devices might, for instance, cut power when a cover is removed it is possible that secret information still lingers in memory cells with some probability of a bit being represented correctly. Since any probabilistic bias gives the attacker an advantage, even disabled and “cleared” memory can provide useful information. Anderson and Kuhn describe a tamper-resistant cryptosystem used in bank ATMs that had this weakness.

## References

Anderson, R., and Kuhn, M.

Karri, R. and Mishra, P. 2002. Minimizing energy consumption of secure wireless session with QoS constraints. In *Proceedings of the International Conference on Communications*. 2053-2057.

Peeters, E., Standaert, F-X., and Quisquater, J-J. 2006. Power and Electromagnetic Analysis: Improved Models, Consequences and Comparisons. To appear in: *Integration, the VLSI Journal, Elsevier*.