Total: 1000 to 3000 words, all including title, authors, affiliations, references, but not figures.

Security adds an extra design dimension to integrated circuit design

Ingrid Verbauwhede, KU Leuven, COSIC, Belgium and UCLA, USA.

Short quote: For the last 20 years we have been designing for low power and low energy: for the next 20 years we will design for security in addition.

Abstract:

Internet of things devices, including medical implants, sensor nodes, wearables, automotive devices, and more, are typically immersed in the environment and extremely resource constraint. They operate from batteries or scavenge energy: thus optimizing for power and/or energy is of utmost importance. To protect privacy, to authenticate data or source of information, and to add resistance to physical manipulation, we add security, cryptographic capabilities and other countermeasures to the integrated circuits. The implementations of these cryptographic algorithms and protocols also need to be compact and energy efficient. But at the same time, they should resist physical attacks and sensitive information should not leak from the device during computations. Adding countermeasures increases the overall cost and energy budget and adds an extra design dimension. Attackers will always go for the weakest link. Thus security needs to be considered at every design step and requires its own testing strategies.

In this short paper, we motivate the need for security in embedded devices. We start from the 'attacker model', discuss the associated design and test methods and we illustrate this with several examples. We mainly focus on digital hardware implementations for cryptographic algorithms, protocols and security devices. We do not address hardware solutions to protect against software attacks.

It is difficult to measure security as there are no common agreed units for it. Therefore, security is based on an attacker model and security is evaluated against the assumptions made in the attacker model. For typical digital circuit design we distinguish between the following broad categories: the black box model mostly of the past, the gray box model of current designs and the immersed model of the future. Also white box cryptography exists.

Black box attacker model: the past

Modern cryptography started in the 1970s with the first US standard for the symmetric key algorithm DES, the invention of public key cryptography and the development of the public key RSA algorithm. These developments appeared together with novel electronic communication means and was pushed by the

electronic revolution. The DES algorithm was developed with efficient hardware implementation in mind. Indeed, it performs poorly in software. Cryptography in this case, is applied to protect the information flow between two communicating parties. As attacker model, we assume that the devices of Alice and Bob (the two main characters in every cryptographic protocol) operate in black boxes, meaning that only the cryptographic inputs and outputs of the devices can be observed by Eve, the attacker. This is shown in Figure 1, where every device has a root of trust indicated with the green dot. Examples are computers and servers in computer rooms or offices. Typical attacks are performed on the network connections between devices, e.g. this could be an LAN, the internet, or any wireless link between devices. In the black box model, Eve collects input/output pairs, i.e. plain text/ cipher text pairs, which she uses to guess the secret key. She is also allowed to modify input data or adaptively supply her own input data. The security strength in this model is purely based on the computational complexity of the underlying cryptographic algorithms. If Eve succeeds in guessing the key faster than brute-force, i.e. faster than trying all possible keys, then the algorithm is considered broken by the cryptographic community.

In this context, Moore's law is important for the algorithm designer. Indeed, Moore's law helps the attacker as it gives him more computational power for brute-forcing cryptographic algorithms. The consequence is that key lengths for cryptographic algorithms keep growing. First the European network ECRYPT and later ENISA publishes documents on required key lengths for near term and long term security [ENI14]. For long term security, it suggests key lengths of 256 for symmetric key size, 512 of Hash output size; 15360 for the RSA modulus size and 512 bit elliptic curves.

As a result, the digital hardware designer has to design for often very large word lengths in combination with unusual arithmetic. The most critical component of secret key algorithms such as DES or AES are the 'substitution boxes' or Sboxes. Special efforts are made to make them either compact or very fast [SMS+16]. The most critical components of public key algorithms such as RSA or Elliptic curve based crypto is the implementation of the finite field arithmetic, and more specifically the exponentation algorithms and the finite field multipliers. The focus of digital hardware designer in the black box context is thus *efficiency*: small area, high throughput, low power and low energy. In this sense, the integrated circuit design process is no different from optimizations in other fields, such as image, video or communications applications.

Testing of black box security circuits focuses therefore on functional correctness, and on measuring throughput, power and energy. Care has to be taken that sensitive registers, such as key registers are not

included on scan chains or that these scan chains are disconnected before shipment. Fortunately, because cryptographic algorithms aim at maximum confusion and diffusion, fault coverage is typically very high for testing cryptographic algorithms.

Gray box attacker model: the current

Electronic devices and computations are more dispersed and distributed. In the internet-of-things context, we assume that electronics are in the hands of the user. They use mobile devices, smart cards, game consoles and other gadgets. Sensor nodes are distributed in the environment and medical devices are implanted into our bodies. Electronics are everywhere in the car, to monitor sensors and drive actuators, to provide an entertainment system, to enable autonomous driving and more. Security, reliability, safety are of upmost importance. In this context, we assume an attacker model where the attacker has access both to the communication channel and the devices itself, as shown on Figure 2. Instead of only using input-output plaintext/ciphertext pairs, the attacker obtains extra information by observing (passive attack) or manipulating (active attacks) this one integrated circuit *while* it is performing the calculations on the embedded device. This extra side-channel information can take on many forms.

Timing attacks:

The attacker will observe differences in execution time dependening on the values of the key or other sensitive data. Cache attacks are a well-known example: they are effective for table based implementations of the Sboxes of symmetric key algorithms: if data being in the cache depends on the key, then timing differences leak information [B04]. As a reaction, native AES instructions were added to high-end x86 processors: it improves both performance and runs in data-independent time [SG12]. Public key algorithms, such as RSA or elliptic curve public key schemes, typically scan the key bits in a bit-serial fashion. A sequential implementation that contains key-dependent if-then-else structures will leak information: when the if-branch takes a different execution time than the else-branch, a difference in execution time again leaks information of the key. Therefore, a huge effort is spent to make sure that a cryptographic implementation in hardware or software runs in constant time. For instance, it is of crucial that finite-state-machines consume exactly the same amount of cycles independent of the data being processed. Unfortunately, most hardware optimization and synthesis tools and also software compilers will remove the dummy code or logic that was added to make a circuit or implementation constant time, as it is considered redundant or dead code. Hence there is a need for tools that check or create constant time implementations [RJV17]. Note that timing attacks can also be executed remotely and hence they are also a concern for cloud and server implementations.

Side-channel attacks: simple and differential

Similarly, data-dependent variations in power consumption or electro-magnetic radiation can reveal sensitive data. Collectively they are called side-channel attacks. It requires that the attacker has close proximity to the device and can monitor the power consumption or pick-up the EM radiation. It is a passive attack, and it monitors the device without disturbing its normal operation. The danger of passive attacks is that the device itself might not be aware that it is being monitored, hence also the need to develop circuits to detect observation [HHM+17].

Simple Power Attacks (SPA) rely on one or a few power or EM measurements to obtain the secret key. Template attacks are one example of simple power attacks, where a template of the device under attack is available. Creating the template could take a huge amount of measurements but the attack on the actual device itself could only take one or a few measurements. The name SPA is misleading as in general SPA attacks are not that simple to mount.

Differential power and EM attacks (DPA) are typically based on multiple, hundreds or sometimes millions of power or EM traces to guess cryptographic secrets. In a typical set-up, the attacker creates a model for the power consumption behavior of the device. E.g. the attacker assumes that the power consumption is related to the Hamming distance between current and previous data values stored in flip-flops or registers. Indeed, this closely corresponds to the dynamic power consumption profile in standard CMOS design where power is consumed only when the state of the flip-flop changes. Up till now, we typically ignore leakage current and other noisy effects because dynamic power is still dominant. DPA attacks are quite robust and resilient to noise: noisy uncorrelated effects can be reduced by increasing the number of measurements. To protect symmetric key algorithms, such as AES, two main classes of counter measures have been developed. One class is a set of circuit styles that makes the switching activity of gates independent of the data that is being processed: it is based on dynamic differential logic styles [TV06]. Masking techniques are a second major class of countermeasures: it is based on randomly splitting sensitive data into multiple shares and obtaining a subset of shares does not disclose that actual data [RBG+15]. Public key algorithms are usually protected at the algorithm level, by adding randomness to the calculations, with techniques such as key or scalar blinding or the use of projective coordinates, so that the attacker has difficulty in obtaining multiple power or EM measurements.

In practice, side channel information is used to reduce the computational complexity of a brute-force attack. As an example, the popular correlation power attack (CPA) reduces the effort of attacking an AES key from

2^128 to the size of the AES Sbox, 2^8, the input data to the Sbox being correlated to the key. The attacker makes 256 possible Hamming distance power models and looks for the model that results in a maximum correlation with the measured power traces. This is then repeated for every key byte. Hence a correlation attack is a divide-and-conquer attack: it combines side channel information with brute-forcing 256 possible key byte guesses.

Fault attacks:

Fault attacks are an active attack but not necessarily an invasive attack. Faults can be triggered by playing with the power supply, by introducing clock glitches, by freezing or heating the device typically applied to random number generators, or any other means that triggers faulty results [KSV13]. IC designers have developed many test strategies to address random faults in integrated circuits. There are test strategies based on scan chains, built-in-self-test, and more. Many of the countermeasures, such as adding redundancy, could be reused to detect fault attacks. There are however two main differences between an attacker aiming at injecting faults and testing for random faults. The first one is the simple fact that knowing if a fault occurred yes or no, could already disclose sensitive information and needs to be avoided [BML01]. The second one is that the designer has to avoid that key registers and other sensitive data are sitting on the scan chains. This would otherwise provide a very nice backdoor.

Side note on white box cryptography:

'white-box' cryptography also exists: it is used in the context of secure software distribution where a cryptographic algorithm with associated key is converted into a key specific tamper resistant piece of software. It is used when no hardware cryptographic module or trusted element is available. Mathematically, most white box schemes are broken, but specific applications such as Digital Rights Management (DRM) still make use of white box cryptography. It will be not further discussed in this paper.

Attacker model of the future:

In the future, we expect that the capabilities of the attackers will improve and that from a gray box model we will evolve into an immersed model, as shown in Figure. 3. The internet of things is a complex system with light-weight, resource constraint devices at the edge. These devices communicate with each other and with mobile portable devices, such as phones, PC's, laptops, etc. At the center, cloud computing and servers provide extraordinary compute power to all. So far, in the gray box model, we assume that an attacker observes one device or one communication link at a time. In the immersed model, we assume that multiple nodes supported by malicious servers or botnets can conspire to attack one poor IOT device. Thus besides

providing protection with strong cryptographic algorithms and strong physical protection against side-channel attacks, we also need strong cryptographic protocols that protect a system even if individual nodes fail. In this context, we see many interesting research challenges. Since, electronics are immersed into the environment, we need strong roots of trust. These roots of trust will be in hardware, they need to be minimal in size and there should be some explicit proof that one can trust them. Including in this set will be Physically Unclonable Functions (PUFs) to derive device specific keys and embedded true random number generators that can resist a wide range of attacks. Secondly, there is a need to develop more hardware circuits to support software security to provide for instance control flow integrity, software integrity, remote attestation, and so on. A third challenge is the development of quantum computers. Major classes of current public key algorithms based on RSA and elliptic curves will be broken if/when quantum computers appear. Mathematicians are developing new generation of algorithms, but the challenge for the digital hardware designer is to implement them including all current software, side-channel and fault attack countermeasures. Also lacking is support by design methods and tools for security.

To conclude: the future of secure digital hardware design will require a strong interaction between digital circuit design, computer arithmetic, cryptography and technology evolutions. And while in the past, we only worried about energy and power minimization, we now have to face the additional constraint that our circuits need to be resistant to a wide range of attacks.

References:

[B04] Daniel J. Bernstein. "Cache-timing attacks on AES." 37pp, 2004.11.11, available from https://cr.yp.to/papers.html#cachetiming.

[BML01] D. Boneh, R. DeMillo, R. Lipton, “On the importance of Eliminating Errors in Cryptographic Computations,” Journal of Cryptology, Vol. 14, issue 2, pp. 101-119, March 2001.

[ENI14] “ENISA Algorithms, key size and parameters report 2014,” available from https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014 .

[HHM+17] N. Homma, Y. Hayashi, N. Miura, D. Fujimoto, M. Nagata, T. Aoki, “Design Methodology and Validity Verification for a Reactive Countermeasure Against EM Attacks,” Journal of cryptology, 2017/4/1, Vol.30, Issue 2, pp. 373-391, 2017.

[KSV13] D. Karaklajic, J. Schmidt, and I. Verbauwhede, "Hardware Designer's Guide to Fault Attacks," IEEE Transactions on Very Large Scale Integration (VLSI) Systems 21(12), pp. 2295 - 2306, 2013.

[MGC+17 P. Maene, J. Goetzfried, R. de Clercq, T. Mueller, F. Freiling, and I. Verbauwhede, "Hardware-Based Trusted Computing Architectures for Isolation and Attestation," IEEE Transactions on Computers, 14 pages, 2017.

[RJV17] O. Reparaz, J. Balasch, and I. Verbauwhede, "Dude, is my code constant time?," In Design, Automation and Test in Europe (DATE 2017), IEEE, 6 pages, 2017.

[RBG+15] O. Reparaz, B. Bilgin, B. Gierlichs, S. Nikova, and I. Verbauwhede, "Consolidating Masking Schemes," In Advances in Cryptology - CRYPTO 2015, Lecture Notes in Computer Science 9215, R. Gennaro, and M. J. Robshaw (eds.), Springer-Verlag, pp. 764-783, 2015.

[RRV16] V. Rozic, O. Reparaz, and I. Verbauwhede, "A 5.1μJ per point-multiplication elliptic curve cryptographic processor," International Journal of Circuit Theory and Applications 45(2), pp. 170-187, 2016.

[SG12] Shay Gueron, " Intel Advanced Encryption Standard (Intel AES) Instructions Set - Rev 3.01," available from https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set/, last revision August 2, 2012

[SMS+16] Sudhir Satpathy, Sanu Mathew, Vikram Suresh, Mark Anders, Himanshu Kaul, Amit Agarwal, Steven Hsu, Gregory K. Chen, Ram Krishnamurthy: "250mV-950mV 1.1Tbps/W double-affine mapped Sbox based composite-field SMS4 encrypt/decrypt accelerator in 14nm tri-gate CMOS. VLSI Circuits 2016.

[TEE16] Global Platform, the Standard for Secure Digital Services and Devices, https://www.globalplatform.org/home.asp

[TV06] K. Tiri, and I. Verbauwhede, "A Digital Design Flow for Secure Integrated Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(7), pp. 1197-1208, 2006.