Fault-based Attacks on Cryptographic Hardware

(Embedded Tutorial)

Ilia Polian
Faculty of Computer Science and Mathematics
University of Passau
Innstr. 43, Passau, Germany
Email: ilia.polian@uni-passau.de

Martin Kreuzer
Faculty of Computer Science and Mathematics
University of Passau
Innstr. 43, Passau, Germany
Email: martin.kreuzer@uni-passau.de

***Abstract*—Mobile and embedded systems increasingly process sensitive data, ranging from personal information including health records or financial transactions to parameters of technical systems such as car engines. Cryptographic circuits are employed to protect these data from unauthorized access and manipulation. Fault-based attacks are a relatively new threat to system integrity. They circumvent the protection by inducing faults into the hardware implementation of cryptographic functions, thus affecting encryption and/or decryption in a controlled way. By doing so, the attacker obtains supplementary information that she can utilize during cryptanalysis to derive protected data, such as secret keys. In the recent years, a large number of fault-based attacks and countermeasures to protect cryptographic circuits against them have been developed. However, isolated techniques for each individual attack are no longer sufficient, and a generic protective strategy is lacking.**

I. Introduction

Sensitive data that need to be protected from unauthorized access play an increasingly important role in today's mobile and embedded applications. These applications include, on the one hand, smart-logistics and smart-shop systems, next-generation payment infrastructures and ambient assisted living technology that process personal financial and/or health data. On the other hand, technical systems also depend on protected data, one example being car engine parameters that are prone to manipulation in context of unauthorized tuning [1], [2]. The relevance of data protection will likely further increase in emerging cyber-physical systems that are characterized by increased interconnection, environment interaction and integration in existing infrastructures, most notably the Internet.

Protection of sensitive data in mobile and embedded systems is based on *cryptographic functions* [3]. The most important cryptographic functions are *ciphers*, which *encrypt* information such that access is only possible for users with appropriate authorization, for instance, knowledge of a *secret key*. One can distinguish between asymmetric and symmetric ciphers, and further between block and stream ciphers. *Symmetric ciphers* use the same secret key for encryption and decryption, while *asymmetric ciphers* use two different keys: a *public key* for encryption and a *private key* for decryption. *Block ciphers* process chunks of data with a well-defined length (e.g., 128 bits), whereas *stream ciphers* process data continuously bit-by-bit.

A block cipher takes a *plaintext* m of fixed length and a key k and maps this pair to the *ciphertext* $E(m, k)$. Many block ciphers follow one of the three basic schemes: substitution-permutation networks (AES [4], Serpent [5], Feistel networks (DES [6], MISTY [7]) and addition-rotation networks (FEAL [8], Threefish [9]). Stream ciphers generate a *key stream* that can be blended (usually, XORed) with the plaintext. In *synchronous* stream ciphers (Trivium [10]), the key is independent of plaintext or ciphertext, while *self-synchronized* stream ciphers (Grain [11]) incorporate plaintext or previously calculated ciphertext bits into the key calculation. A relatively new trend are *lightweight ciphers* that are optimized towards low area cost and power consumption of their hardware implementation [12]. They are intended for use in mobile and embedded applications with heavy cost pressures and strictly limited energy supply. Examples of lightweight ciphers are PRESENT [13], PRINCE [14], LED [15], HIGHT [16] and Piccolo [17].

Cryptanalysis [18], [19] is a discipline that investigates the security of ciphers to withstand attempts to circumvent their protection, explores their vulnerabilities to various types of attacks and develops techniques to counter such attacks. Classical cryptanalysis concentrates on making sure that an unauthorized attacker who does not know the key cannot perform decryption with a realistic effort. This is achieved by requiring decryption methods that depend on mathematical calculations for which no efficient algorithms are known, such as factoring of large numbers or computation of discrete logarithm.

More recently, cryptanalysis was extended to methods taking information from *side-channels* into account. Such methods require access to the hardware on which the cryptographic function is executed. *Passive side-channel cryptanalysis* techniques monitor execution time [20], power consumption [21] or electromagnetical emissions [22] and incorporate these data into their effort to derive protected information, such as the secret key or its individual bits. In contrast, *active* techniques, also called *fault-based attacks*, manipulate the circuit on which the cryptographic function is executed in a controlled way [23], [24]. Faults are injected by sudden changes in power-supply voltage or clock frequency, or by inducing parasitic currents in structures such as memory cells, registers or logic gates using laser irradiation [25]. Most fault-based attacks perform a cryptographic function repeatedly, in presence and in absence of faults, and run *differential cryptanalysis* on the observed outputs [26]. Fault-based analysis approaches have been published for block ciphers [27], [28], stream ciphers [29], [30] and asymmetric ciphers [31], [32].

This paper is intended as a rather informal introduction to the basic principles of fault-based attacks and their relation to neighboring fields. It is not meant to provide a comprehensive

978-1-4673-6136-1/13/$31.00

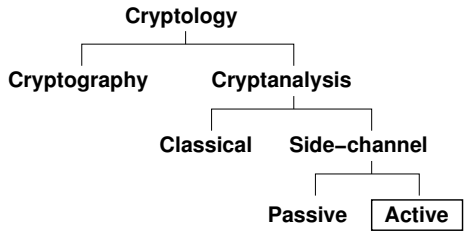

Fig. 1. Fault-based attacks in the broader context

overview of this active scientific area that experienced sustainable growth in the last decade. The remainder of the paper is structured as follows. Section II puts fault-based attacks into the broader context of side-channel cryptanalysis and reviews fundamental techniques which are also used in fault-based attacks. Section III provides more background material on fault-based attacks. Section IV demonstrates one particular fault-based attack developed by the authors for the new LED-64 block cipher [33]. Section V discusses implications of fault-based attacks to circuit design: possible countermeasures and their trade-offs with the traditional design objectives such as area, performance, power consumption and testability. Section VI concludes the paper.

II. Side-channel Cryptanalysis

Figure 1 illustrates the broader context of cryptanalysis, including side-channel cryptanalysis. Cryptology, the science of secure information and communication, comprises cryptography, a discipline concerned with design of protected information systems, and cryptanalysis, which investigates vulnerability of such systems to attacks. Attacks may aim at decryption of protected information (ciphertext) without possessing the appropriate authorization (the secret key), and/or at identification of the secret key which will allow the attacker to access the protected information in the future. Classical cryptanalysis methods include brute-force analysis (trying all possible keys), algebraic approaches (construction of mathematical equations that include variables representing the secret key, and solving these equations), and differential techniques (studying how slight differences in inputs of a cryptographic function manifest themselves at its outputs and deriving secret key information from these observations).

State-of-the-art cryptographic functions are designed with these (and further) methods in mind. For example, the secret key always consists of a number of bits that is large enough to make brute-force search impractical on the fastest existing computers. Cryptographic functions are typically based on mathematical transformations that are difficult to analyze using known methods. Therefore, even though the attacker may formulate a system of equations which could yield the secret key, solving these equations will require inefficient algorithms that are unlikely to terminate in practical time on today's computers.

Side-channel cryptanalysis targets the implementation of the cryptographic function (which may be an application-specific circuit, a mapping on a reconfigurable platform such as an FPGA, or software that runs on a microprocessor) rather than the algorithm itself. One can distinguish between passive techniques which observe the circuit behavior and active techniques which, in addition, manipulate the circuit. Fault-based attacks, which are the focus of this tutorial, are active cryptanalysis techniques. However, a basic understanding of passive techniques is useful, as they are often employed to support fault-based attacks.

Suppose that executing a cryptographic function requires some calculation if bit k_0 of the secret key k equals to 1 and no such calculation if $k_0 = 0$. By observing the circuit, a number of indications may be used to determine whether the subroutine implementing the calculation has been executed. If the circuit has executed the subroutine, it has

1) required more time
2) consumed more power
3) emitted more electromagnetic noise

than a circuit that has not executed the subroutine. If the expected values for any (or all) of these three parameters are known in advance for $k_0 = 1$ and $k_0 = 0$, it is possible to measure the actual values using appropriate equipment and to match the measured data with the precomputed ones. If the measurements indicate that the subroutine has been executed, then k_0 must be 1, otherwise k_0 must be 0.

It may be possible to derive further bits of the key using the same technique. However, it is not necessary to reconstruct the whole key. It is sufficient to narrow down the number of possible key candidates to a value practical for a brute-force attack. For example, if 30 out of 64 secret key bits are known, trying out all combinations for the remaining bits will take 2^{34} attempts, which is feasible on today's computers. Instead of determining the values of the key bits, side-channel analysis may yield relationships between them. For instance, the measured power consumption may indicate the number of 1's in the complete key, which reduces the number of possible candidates. Several such key space restrictions may make exhaustive enumeration feasible.

Most published techniques are based on the follwoing side-channels:

1) Timing [20] (measured using an external clock or mechanisms integrated into the operated system)
2) Power and energy consumption [21] (measured at the power-supply connections of the device)
3) Electromagnetic noise [22] (for example, current fluctuations)
4) Cache contents or memory response time [34]
5) Values observed through infrastructure to facilitate manufacturing test, such as scan chains [35], [36].

However, new unexpected side-channels are frequently discovered.

Side-channel analysis is subject to a number of limitations. Performing measurements on a circuit executing a complex cryptographic function will result in a substantial amount of data which may not be easy to relate to the individual computation steps. For example, suppose that after the conditional execution of a subroutine dependent on key bit k_0, there are similar subroutines executed dependent on key bits k_1, k_2, k_3 and so forth. In order to accurately derive the values of individual key bits, side-channel analysis must be able to distinguish between the different subroutines based on the measured data.

A further complication is the inherent inaccuracy in measurements themselves. On the one hand, the instruments used for performing measurements are imperfect but require a very high precision to detect very small differences in, e.g., power consumption, and accurately synchronize the start and the end of the measurement. On the other hand, the circuits exhibit variability in many ways. For example, the power consumption of circuits in state-of-the-art manufacturing technologies may vary considerably between manufactured instances of a circuit. This variability (which may stem from the parts of the circuit not involved in the calculation of the cryptographic function) could dominate the subtle difference in power consumption used by side-channel analysis. In the example above, $k_0 = 0$ may result in a larger absolute power consumption on a circuit instance which happens to be more power-hungry than $k_0 = 1$ on a different instance with a lower intrinsic power consumption. An other source of the variability is present in complex microprocessors that automatically reorder the individual machine instructions and execute instructions belonging to multiple tasks simultaneously. At a given time, such a microprocessor may execute the cryptographic function and some other, unrelated software in parallel, and it is difficult to know which part of the cryptographic function is being executed.

In addition, a number of countermeasures exist. For example, it would be possible to execute the subroutine independent of k_0 and to simply ignore its result if $k_0 = 0$. Such countermeasures are associated with costs (in the example, wasted resources for an unnecessary computation), and can reduce, but not completely eliminate the vulnerability to side-channel attacks. More information on countermeasures is provided in Section V.

III. Fault-based Attacks

Fault-based attacks are active cryptanalysis methods which manipulate the circuit while it executes a cryptographic function. A number of techniques to inject faults into the circuit have been suggested [23], [24]:

1) Increasing the operating frequency of the circuit, or producing glitches on the clock signal within the clock cycle. This erroneously causes the write operation on the memory elements of the circuit: the values at their inputs, which may be incorrect because the time to calculate them was insufficient, are written into the memory elements.
2) Lowering the power-supply voltage. This increases the delay of the gates in the circuit and leads to the similar effect as increasing the frequency. A related technique is introducing noise on the power supply, which also increases the delay of the gates [37], or inducing electromagnetic disturbances by a spark generator.
3) Depackaging the circuit and its irradiation by intense light. The parasitic currents modify the voltage at the irradiated circuit structures and ultimately induce bit flips. Light sources may range from a low-cost ultraviolet lamp or camera flash to high-precision lasers that can be pointed to a specific register or logic gate.
4) Using a focused ion beam to modify the circuit structure. This requires sophisticated equipment, expensive consumables, and highly skilled personnel.

All fault-based attacks must make an assumption about the capability of the attacker, often formalized as a *fault model*. In general, a fault model must define the temporal and the spatial resolution of the fault injection. *Temporal resolution* refers to the capability of the attacker to inject a fault at a time point of his/her choice. Many published fault-based attacks require fault injection in a specific step of the procedure, e.g., after the 30^{th} of 32 rounds of an encryption algorithm. This requires that the attacker is able to identify the clock cycle when the 30^{th} round finishes and can inject the fault in this cycle and not earlier or later. In practice, this can be achieved by leveraging side-channel information, namely monitoring power consumption. Power consumption over time during a single round tends to follow a specific pattern, and counting 30 such patterns indicates that round 30 has finished.

Spatial resolution is the ability of the attacker to pinpointedly manipulate the desired information. Frequency and voltage manipulation provide a low degree of control, in particular when targeting circuits that exhibit variability. One can assume that some of the values in the memory elements will become erroneous, but it is uncertain which memory elements will be affected. Moreover, if the circuit has a control and a data part, frequency and voltage manipulation will likely impair both parts. However, errors in the control part may simply terminate the algorithm, put it into an infinite loop or create other conditions that deliver no results which are useful for the attacker.

If the attack relies on manipulating a specific structure, such as a register, a memory element, or a logic gate, well-controlled laser irradiation is the most effective technique. For example, some attacks specify a variable that needs to be modified. To implement such an attack, the attacker must determine the register in which the variable is stored, point the laser on this register, adjust its energy such as to induce bit flips in the register without causing errors in the neighboring structures, and trigger it at the required clock cycle. One can further distinguish between the ability of the attacker to flip some random bits of the register (such that the register assumes an erroneous value that cannot be controlled), or to flip just the desired bits (that is, overwrite the register with a specific value). In the former case, it is of interest if the attacker can find out which bits have flipped, and which value the register finally assumed.

Temporal and spatial resolution are ultimately determined by the capabilities of the equipment used by the attacker. Better resolution requires more sophisticated (and therefore more expensive) equipment but also enables more powerful attacks. The fault model, which formalizes this capabilities, is an integral part of a sound attack description.

IV. Fault-based Attack on LED-64

This section illustrates a fault-based attack on the lightweight block cipher LED-64 [33]. The attack has been inspired by a similar work on AES [28]. LED-64 is well suited as an example because it is sufficiently simple and yet it exhibits many representative properties of ciphers based on substitution-permutation networks.

LED-64 [15] encodes a 64-bit plaintext P into a 64-bit ciphertext C using a 64-bit secret key K.[1] The *state* of the

[1]There is a 128-bit version of LED not considered here.

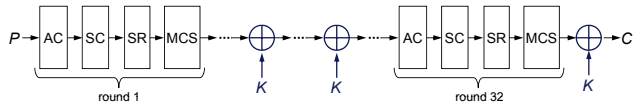

Fig. 2. LED-64 encryption process

algorithm is a 64-bit value set to P in the beginning. LED-64 applies a number of transformations to the state, and in the end of the process the state is the ciphertext C. The 64-bit state is organized in 16 four-bit chunks, called *nibbles*. A four-bit nibble can be represented as a hexadecimal number, and the complete 16-nibble state is typically written as a 4×4 matrix of hexadecimal numbers.

The LED-64 algorithm consists of 32 rounds outlined in Figure 2. In each round, the following four transformation are applied to the state:

- AC (Add constant): A constant value is added (i.e., XORed) to the state. Different constants are used in different rounds, but their exact value is irrelevant for the understanding of the attack.
- SC (SubCells): For every four-bit nibble of the state, its value x is replaced by a value $SBox[x]$ according to a look-up table $SBox$. This is the only non-linear operation in LED-64.
- SR (ShiftRows): Circular shift is applied to each row of the 4×4 state matrix. The four nibbles in the first row of the matrix are unchanged, while the nibbles in the second, third and fourth row are shifted to the left by one, two and three positions, respectively.
- MCS (MixColumnsSerial): The state is multiplied with a matrix M:

$$M = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix} \quad (1)$$

We omit the values of the constants for the AC step and the SBox for the SC step. They can be found in the LED-64 specification.

After every 4 rounds, the secret key K is added (XORed) to the state. Therefore, a total of 8 key additions is performed during the 32 rounds.

The attacker aims at determining the value of the secret key K. It is assumed that he can apply a plaintext P of his/her choice to the inputs of a circuit implementing LED-64 and observe the calculated ciphertext C at the outputs of the circuit. Then the calculation is repeated with the same P as the input, and a fault injection is performed. The fault is injected into the state at the beginning of round 30 (i.e., three rounds before the termination of the algorithm).

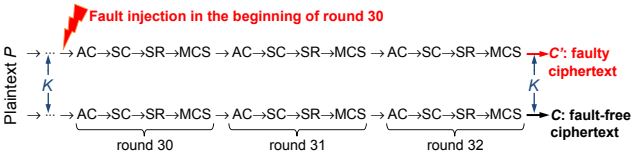

Fig. 3. Fault injection during LED-64 encryption

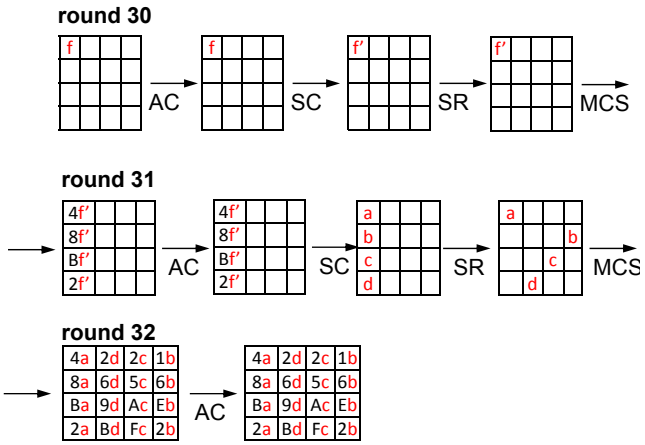

Fig. 4. Propagation of the injected fault

The following fault model is used: the attacker can manipulate exactly one of the 16 four-bit nibbles of the state. It is irrelevant which nibble and how many bits within that nibble are affected, but it is important that the fault effect is restricted to just one nibble. In the following, we will assume that the fault affects the first nibble (top-left in the 4×4 state matrix), but the procedure is similar for different nibbles. Moreover, we assume that the attacker does not know which bits in the affected nibble have flipped. He knows that the fault replaced the value x of the nibble by the value $x \oplus f$, but he does not know the value of f.

Figure 3 illustrates the fault injection. The fault-affected and the fault-free calculations are shown on the top and the bottom, respectively. Since the same P and the same K is used in both cases, the states before fault injection are identical. The fault will propagate to the output, resulting in the faulty ciphertext C' that differs from the correct ciphertext C. The attack requires C and C' to obtain K by differential cryptanalysis. For this purpose, the propagation of the fault effect, shown in Figure 4, is discussed next.

The figure shows the differences $s' \oplus s$ between the fault-affected state s' and the fault-free state s of the cipher. Let $s'_1, \ldots s'_{16}$ and $s_1, \ldots, s_{16}$ denote the individual nibbles of these states. As discussed above, we assume that the fault has been injected into the first nibble: $s'_1 \oplus s_1 = f$ (where the value of f is unknown according to the fault model) and $s'_i \oplus s_i = 0$ for all other nibbles. In the AC step, the same constant is added to s' and s, such that the difference between s' and s remains the same. In the SC step, the non-linear SBox mapping is applied to all nibbles. The difference $SBox[s'_1] \oplus SBox[s_1]$ cannot be represented as function of f; we denote this difference by f' (which is also unknown). However, for all other nibbles $s'_i = s_i$ implies $SBox[s'_i] \oplus SBox[s_i] = 0$.

Step SR does not affect the first row and induces identical changes on the other rows of s' and s. Step MCS, due to the linearity of matrix multiplication, leads to the following differences: $M \cdot s' \oplus M \cdot s = M \times (s' \oplus s) = M \cdot (f', 0, \ldots, 0)$ (where M is the matrix from Eq. 1). The difference matrix at the beginning of round 31 is shown in the second row of Figure 4. Note that 4, 8, B and 2 are hexadecimal scalars. As discussed above, step AC does not lead to any changes in the difference. Step SC maps the four non-zero differences to new, unknown

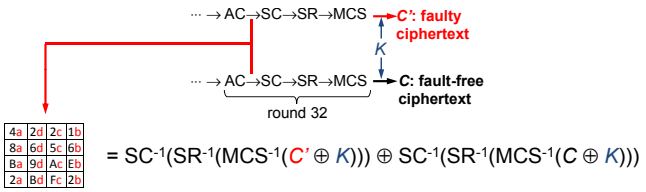

Fig. 5. Equation construction for the attack on LED-64

values a, b, c and d. The SR step shifts the differences within the rows, and the MCS step multiplies the difference matrix by M. The resulting matrix not changed by step AC. It turns out that the relationship between four unknown variables a, b, c and d described by the entries of the difference matrix is sufficient for cryptanalysis. Note that applying the SC step now would replace all differences by unknown values and destroy all residual information.

The attack is performed by creating equations for the cipher state after step AC of round 32, in which the propagation analysis terminated. The general procedure is outlined in Figure 5. Recall that one fault-free encryption and one encryption with fault injection were performed and the respective ciphertexts C' and C have been recorded. The state of the fault-affected cipher after step MCS in round 32 is $C' \oplus K$, the state after step SR is $\mathrm{MCS}^{-1}(C' \oplus K)$, and the state after step AC is $\mathrm{SC}^{-1}(\mathrm{MCS}^{-1}(C' \oplus K))$. Note that the secret key K is unknown; therefore it is represented by 16 variables $k_1, \ldots, k_{16}$, each representing a four-bit portion of K. The same equation is constructed starting with the fault-free ciphertext C. The difference of the resulting states after step AC of round 32 must match the result of the propagation analysis in Figure 4.

The approach outlined above yields a system of equations with unknown variables a, b, c and d (which describe the intermediate effects of the fault injection) and $k_1, \ldots, k_{16}$ (the secret key). Note that neither the plaintext P nor the actual fault value f are present in the equation. The equations are over GF(16) in which all transforms except $SBox$ are linear. The equations can be solved in two ways. It is possible to combine the equations that describe parts of the key and apply filtering techniques which eliminate a large number of possible key candidates, such that the number remaining candidates is small enough for brute-force search [33]. Alternatively, the system of equations can be mapped to a Boolean-satisfiability instance and solved using a SAT solver [38]. The latter is an example of an *algebraic attack* [39].

In summary, the relationships introduced into the cryptographic function by a single fault injection are sufficient to find the secret key in practical time using today's computers, as illustrated by comprehensive results in [33], [38].

V. Countermeasures and Design Considerations

Protection against fault-based attacks can work on multiple abstraction levels: technology, circuit, information, or application. Technology-level approaches focus on physical protection of circuits (to hinder the attacker from accessing the device by storing sensitive information in parts of the circuit that are difficult to access, or shielding these parts of the circuit), or on identifying tampering attempts by providing sensors that detect light irradiation, power glitches, changes in clock frequency, and so forth. If a sensor identifies suspicious activity, the circuit can react by self-deactivation, requesting a new, uncompromized secret key, producing a random output to mislead the attacker, and/or generating a system-wide alarm. Techniques at the circuit level are more focused on the passive side-channel analysis. For example, there are design styles for which the switching activity (and therefore the power consumption) is independent from the data being processed, complicating power analysis. However, these techniques tend to incur large area and power overheads.

A large class of countermeasures is based on different forms of redundancy, similar to techniques used to detect transient faults during the operation of the circuit [40]. One can distinguish between hardware redundancy (providing multiple copies of the same block along with voters, comparators, error-detection and reconfiguration circuitry and the like), time redundancy (repeating potentially erroneous calculations), information redundancy (using special codes to detect and/or correct errors) and software redundancy (using machine instructions for errors in microprocessors). All these methods are associated with considerable area, performance and/or power overheads.

A fundamental difference of fault-based attacks from transient faults is their deterministic nature. For example, circuit duplication is highly effective in detect a random transient fault, as the probability that exactly the same fault will occur in both copies is very small. On the other hand, the malicious attacker will deliberately attempt to inject two identical faults into both copies in order to escape detection. When deploying information redundancy based on the simple parity code, the attacker will target circuit locations which propagate to an even number of outputs. There are special non-linear *robust codes* [41] that have been developed to avoid undetected manipulations.

It is possible to develop specific protection mechanisms for a given cryptographic function or its subroutine. For example, the overview article [24] considers a number of implementations of the RSA algorithm with varying degree of protection against active and passive side-channel analysis. While many such approaches are effective and elegant, they are often restricted to a small class of algorithms and attack scenarios. With over 700 attacks published so far, the need for generic approaches is obvious [42].

A generic design flow to harden the circuits against fault-based attacks will have to take the trade-off between robustness and various types of costs (area, performance, power consumption) into account. This is particularly important for lightweight secure devices used in ultra-low-cost applications with very limited available energy. There are also non-trivial interactions with passive side-channel analysis: for instance, adding redundancy may increase the correlation between the data and the power consumption and make the circuit more susceptible to a power attack. A systematic approach also requires metrics to quantify the vulnerability of a circuit to attacks; some first considerations on a definition of such metrics can be found in [43].

One difficulty in doing research on fault-based attacks is the lack of reliable information on vulnerabilities, countermeasures and their effectiveness from the manufacturers of actual secure circuits. They follow the *security-by-obscurity* paradigm, which confronts the attacker with the additional burden to identify the countermeasures before mounting the actual attack. In the area of cipher design, this approach

has long been replaced by making new developments public, such that any vulnerabilities can be identified and published by the scientific community before an attacker secretly finds and exploits them. Departing from the security-by-obscurity principle would give new impulses to the field and increase the confidence of end-users that the designers did not overlook some subtle side-channel through which protected information can leak to the unauthorized attacker.

VI. Conclusions

Secure handling of protected data is a prerequisite for regulatory approval, societal acceptance and ultimately commercial success of new applications. Fault-based attacks constitute a substantial threat that needs to be understood, quantified, and eliminated. Today's countermeasures which address individual attack scenarios must be subsumed by generic approaches that considered fault-based attacks simultaneously with passive side-channel cryptanalysis. Most realistic applications will be subject to stringent area, performance and power constraints. Practical approaches to harden a circuit against fault-based attacks will offer trade-offs between robustness and costs, combining techniques on multiple abstraction levels.

References

[1] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *IEEE Symp. on Security and Privacy*, 2010.

[2] E. Boehl and P. Duplys, "Nonlinear compression functions using the misr approach for security purposes in automotive applications," in *IEEE Int'l On-line Test Sysmp.*, 2009, pp. 55–60.

[3] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering*. Wiley Publishing, Inc., 2010.

[4] "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards, Tech. Rep. 197, 2001.

[5] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A proposal for the Advanced Encryption Standard," http://www.cl.cam.ac.uk/~rja14/serpent.html.

[6] "Announcing the Data Encryption Standard (DES)," Federal Information Processing Standards, Tech. Rep. 46-3, 1999.

[7] M. Matsui, "Block encryption algorithm MISTY," in *Int'l Workshop on Fast Software Encryption (LNCS 1267)*, 1997, pp. 64–74.

[8] S. Miyaguchi, "The FEAL cipher family," in *CRYPTO*, 1990, pp. 627–638.

[9] N. Ferguson *et al.*, "The Skein hash function family," http://www.skein-hash.info/sites/default/files/skein1.3.pdf.

[10] C. D. Canniere and B. Preenel, "Trivium specifications," http://www.ecrypt.eu.org/stream/ciphers/trivium/trivium.pdf.

[11] M. Hell, T. Johansson, and W. Meier, "Grain – A stream cipher for constrained environments," http://www.ecrypt.eu.org/stream/ciphers/grain/grain.pdf.

[12] "ECRYPT lightweight cryptography lounge," http://www.ecrypt.eu.org/lightweight/index.php/ECRYPT_Lightweight_Cryptography_Lounge.

[13] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 4727)*, 2007, pp. 450–466.

[14] J. Borghoff *et al.*, "PRINCE: A low-latency block cipher for pervasive computing applications," in *ASIACRYPT*, 2012.

[15] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED block cipher," *LNCS*, vol. 6917, pp. pp. 326–341, 2011.

[16] D. Hong *et al.*, "HIGHT: A new block cipher suitable for low-resource device," in *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 4249)*, 2006, pp. 46–59.

[17] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An ultra-lightweight blockcipher," in *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 6917)*, 2011, pp. 342–357.

[18] G. Bard, *Algebraic Cryptanalysis*. Springer, 2009.

[19] E. Biham and O. Dunkelman, *Techniques for Cryptanalysis of Block Ciphers*. Springer, 2011.

[20] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems," in *Annual Int'l Cryptology Conf. (LNCS 1109)*, 1996, pp. 104–113.

[21] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual Int'l Cryptology Conf. (LNCS 1666)*, 1999, pp. 388–397.

[22] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Int'l Workshop on Cryptographic Hardware and Embedded Systems (LNCS 2523)*, 2003, pp. 29–45.

[23] D. Boneh, R. DeMillo, and R. Lipton, "On the importance of elimination errors in cryptographic computations," *Jour. Cryptology*, vol. 14, pp. 101–119, 2001.

[24] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice and countermeasures," *Proc. IEEE*, vol. 99, 2012.

[25] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerers apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, pp. 370–382, 2006.

[26] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Annual Int'l Cryptology Conf. (LNCS 1294)*, 1997, pp. 513–525.

[27] D. Mukhopadhyay, "An improved fault based attack of the Advanced Encryption Standard," in *AFRICACRYPT, (LNCS 5580)*, 2009, pp. 421–434.

[28] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the Advanced Encryption Standard using a single fault," in *Workshop in Information Security Theory and Practice (LNCS 6633)*, 2011, pp. 224–233.

[29] M. Hojsik and B. Rudolf, "Differential fault analysis of Trivium," in *Int'l Workshop on Fast Software Encryption (LNCS 5086)*, 2008, pp. 158–172.

[30] M. Mohamed, S. Bulygin, and J. Buchmann, "Using SAT solving to improve differential fault analysis of Trivium," in *Int'l Conf. on Information Security and Assurance*, 2011, pp. 62–71.

[31] J. Blömer and M. Otto, "Wagner's attack on a secure CRT-RSA algorithm reconsidered," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (LNCS 4236)*, 2006, pp. 13–23.

[32] P.-A. Fouque, R. Lercier, D. Real, and F. Valette, "Fault attack on elliptic curve Montgomery ladder implementation," in *Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2008, pp. 92–98.

[33] P. Jovanovic, M. Kreuzer, and I. Polian, "A fault attack on the LED block cipher," in *Int'l Workshop on Constructive Side-channel Analysis and Secure Design (LNCS 7275)*, 2012, pp. 120–134.

[34] Z. Wang and R. Lee, "Covert and side channels due to processor architecture," in *Comp. Security Applications Conf.*, 2008, pp. 473–482.

[35] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," *IEEE Trans. CAD*, vol. 25, no. 10, pp. 2287–2293, 2006.

[36] J. DaRoit, G. D. Natale, M.-L. Flottes, and B. Rouzeire, "Scan attacks and countermeasures in presence of scan response compactors," in *IEEE European Test Symp.*, 2011, pp. 19–24.

[37] I. Polian, "Power supply noise: causes, effects, and testing," *ASP Jour. Low-Power Electronics*, vol. 6, no. 2, p. 326, 2010.

[38] P. Jovanovic, M. Kreuzer, and I. Polian, "An algebraic fault attack on the LED block cipher," in *Int'l Conf. on Symbolic Computation and Cryptography*, 2012.

[39] P. Jovanovic and M. Kreuzer, "Algebraic attacks using SAT-solvers," *Groups – Complexity – Cryptology*, vol. 2, no. 2, pp. 247–259, 2010.

[40] I. Koren and M. Krishna, *Fault-Tolerant Systems*. Morgan Kaufmann, 2007.

[41] M. G. Karpovsky, K. J. Kulikowski, and Z. Wang, "Robust error detection in communication and computation channels," in *Int'l Workshop on Spectral Techniques*, 2007.

[42] M. Wagner, "700+ attacks published on smart cards: The need for a systematic counter strategy," in *Int'l Workshop on Constructive Side-channel Analysis and Secure Design (LNCS 7275)*, 2012, pp. 33–38.

[43] V. Tomashevich, S. Srinivasan, F. Foerg, , and I. Polian, "Cross-level protection of circuits against faults and malicious attacks," in *IEEE Int'l On-Line Test Symp.*, 2012.