Precise Fault-Injections using Voltage and Temperature Manipulation for Differential Cryptanalysis

Raghavan Kumar[$], Philipp Jovanovic[€] and Ilia Polian[€]
[$]University of Massachusetts, 01002 USA
[€]University of Passau, 94032, Germany
rkumar@ecs.umass.edu, {philipp.jovanovic|ilia.polian}@uni-passau.de

***Abstract*—State-of-the-art fault-based cryptanalysis methods are capable of breaking most recent ciphers after only a few fault injections. However, they require temporal and spatial accuracies of fault injection that were believed to rule out low-cost injection techniques such as voltage, frequency or temperature manipulation. We investigate selection of supply-voltage and temperature values that are suitable for high-precision fault injection even up to a single bit. The object of our studies is an ASIC implementation of the recently presented block cipher PRINCE, for which a two-stage fault attack scheme has been suggested lately. This attack requires, on average, about four to five fault injections in well-defined locations. We show by electrical simulations that voltage-temperature points exist for which faults show up at locations required for a successful attack with a likelihood of around 0.1%. This implies that the complete attack can be mounted by approximately 4,000 to 5,000 fault injection attempts, which is clearly feasible.**

I. Introduction

Fault-based cryptanalysis is an effective technique to extract the secret information from cryptographic devices [1], [2]. A number of highly effective fault-based attacks on classical and new ciphers have been recently introduced [3]–[7]. This latest generation of attacks is able to recover the secret key using a very low number of fault injections (one for AES-128 [3] and LED-64 [4], three to four for PRESENT [6], LED-128 [5] and PRINCE [5], [7]. The flip side of the coin from the attacker's point of view are the high requirements on the precision of the fault injection: Values in specific registers must be manipulated at an exactly known point in time, in order to mount a successful attack. It is commonly believed that such precision is achievable only using complex and invasive injection techniques based on irradiation by intense laser light [8].

Low-cost manipulation of clock frequency or power supply voltage (V_{DD}) tends to result in fault effects that are unpredictable and cannot be exploited for advanced attacks. For example, lowering V_{DD} will lead to failures because the delay of some paths in the combinatorial logic of the circuit will increase beyond the cycle time. However, the paths that will fail when V_{DD} is slightly reduced are typically determined by process variations, ambient temperature and power-supply noise and cannot be controlled. When V_{DD} is reduced significantly, a large number of paths will be delayed and the failing registers will most likely not correspond to the precise requirements of advanced fault-based attacks.

In this paper, we introduce a low-cost manipulation technique that can provide the accuracy that is sufficient to mount last-generation fault-based attacks on hardware implementations of state-of-the-art ciphers. The technique combines lowering V_{DD} to the point when individual logic gates are not able to switch with increasing ambient temperature. We demonstrate that a small but significant percentage of such fault injections (around 0.1%) are exploitable, that is, consistent with the requirements of the fault attack. This percentage is sufficient to make an attack practical. For example, if four exploitable fault injections are required for successful recovery of the key and every $1{,}000^{\mathrm{th}}$ injected fault is exploitable, a total of about 4,000 fault injections are required, which is clearly a feasible number.

The fundamental reason for the success of our fault-injection technique is the existence of small-scale variations. Exploitable faults typically require a fault effect in a given location (e.g. a cell of a register) and the absence of fault effects elsewhere in the circuit. By substantial lowering of V_{DD}, fault effects are induced on individual logic gates rather than accumulated over entire paths. The fault is exploitable if the desired location is flipped before other gates. The first gate that will flip is determined by random variations, and varying the temperature further increases the chance that one of the desired locations will be flipped in one of the fault injection runs.

We demonstrate our technique for a very new fault-based attack [5], [7] on the recently introduced block cipher PRINCE [9]. The attack requires a variable number of fault injections in rounds 8 and 9 of the PRINCE cipher. Each exploitable fault injection reduces the number of secret key candidates, and the exact number of required fault injections is adaptively decided based on the reduction obtained so far. The attack is successful, as soon as the number of remaining key candidates becomes sufficiently small for brute-force search.

For our experiments, we developed an ASIC design of PRINCE, which has comparable characteristics to the reference implementation in [9]. We implemented both a combinatorial and a sequential design of the cipher, as they have different fault injection statistics. The sequential design shows higher percentage of exploitable faults than the combinatorial design due to the presence of precise timing for fault injection. We perform technology-aware full-chip simulation for selected values of power-supply voltage and temperature. The outputs of the faulty circuit are handed over to a software routine that implements the cryptanalysis and ultimately recovers the secret key if the fault was exploitable. We investigate the probability of exploitable faults and validate, that the software routine indeed calculates the correct secret key from the responses of the simulated circuit.

The remainder of the paper is organized as follows. Section II provides some background information on attacks on cryptographic blocks. Our voltage and temperature based fault

injection techniques on cryptographic hardware are presented in Section III. Experimental results of fault-based attacks on PRINCE are shown in Section IV. Finally, the concluding remarks are discussed in Section V.

II. Background

Cryptanalysis is the science of analyzing cryptographic primitives with the aim to measure their level of security. A sub-field of cryptanalysis is the so-called implementation attacks, which target concrete implementations of cryptographic primitives (e.g. on an FPGA, ASIC or micro-controller) and consider additional implementation dependent information. Those attacks can be further divided into active and passive categories.

Passive attacks predominantly use some *side-channel* leakage information for cryptanalysis purposes. Possible side-channel techniques include timing [10], power consumption [11], EM (electromagnetic) noise [12] etc. Most of the side-channel attacks are based on some statistical analysis over the data obtained from physical measurements. For example, a simple timing based attack involves monitoring the data movement in the cryptographic hardware. By observing the duration that the hardware takes to perform cryptographic operations, an attacker might be able to obtain the whole or parts of the secret key.

In *active attacks*, such as *fault-based attacks*, the adversary manipulates the operation of a cryptographic circuit. By injecting faults, it is often possible to reconstruct the key by performing differential cryptanalysis on the observed outputs [13]. Fault injection can be either *transient* or *permanent* [2]. Transient faults are more practical than permanent faults, as they are flexible and repeatable. They can be injected through various methods: Voltage manipulation [14], [15], clock frequency manipulation [16], laser and EM-induced faults [17], etc. Among these methods, voltage and clock frequency manipulation are low-cost and do not require sophisticated equipment unlike laser- and EM-based fault injection. However, the precision of injected faults is low in voltage and frequency manipulation and may require fault injection over an extended period of time to get the desired faults.

Voltage manipulation based attacks can involve either introducing a well-timed glitch into the power supply or under-powering the device. Glitch based fault injection alters the state of the latches or flip-flops, thereby affecting the control and data path logic of the circuit. A successful attack on a RSA implementation using this technique was proposed in [18] and a similar attack on AES implemented in an SRAM based FPGA was demonstrated in [19]. In under-powering attacks, the propagation delays of some combinatorial gates may increase and lead to timing errors. This can cause a flip-flop to capture the erroneous value. One such attack to break a smart card implementation of AES was shown in [20]. Most of the fault-based attacks do not demonstrate very high-precision fault injection depending on the cryptanalysis in effect. However, we demonstrate that very high precision fault-injection is possible through voltage and temperature manipulation, as required by some of the block ciphers such as PRINCE. There are several techniques and emulators available to test and simulate the transient faults injected into secure circuits [21], [22].

As described in [23], fault-based attacks are based on assumptions about an attacker's capability (also known as *fault model*), which defines the spatial and temporal resolution of fault injections. Temporal resolution refers to the ability of an attacker to inject fault(s) into a circuit at a specific point of time (in terms of clock cycle), whereas spatial resolution identifies the capacity to inject fault(s) in the desired location of the circuit. Most of the published fault-based attacks require injection of faults at a specific location of the circuit. For example, the multi-stage attack framework used to attack PRINCE [5] requires fault injections in the 8^{th} and 9^{th} rounds (temporal resolution). Also, the faults must be confined to a single nibble of the 64-bit state (spatial resolution). The spatial and temporal resolutions vary with attacks and the vulnerable locations of a cryptographic circuit.

III. Fault Injection Techniques on Cryptographic Hardware

In this section, we describe fault injection in cryptographic hardware using voltage and temperature manipulation. We use the block cipher PRINCE [9] as a target, to demonstrate the technique. First, we begin with the description of PRINCE. Then we introduce voltage and temperature manipulations based fault injection in PRINCE and show the statistics of the induced faults. Finally, we also show the multi-stage attack framework, adapted to PRINCE, which is able to reconstruct key candidates using the results of our fault injections for differential cryptanalysis.

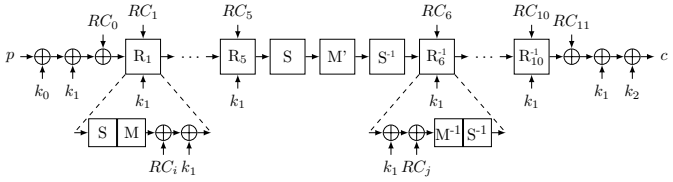

Fig. 1. Layout of PRINCE.

A. Specification of PRINCE

PRINCE takes as input a 64-bit plain-text p and a 128-bit secret key k, that consists of two 64-bit sub-keys k_0 and k_1, with $k = k_0 \parallel k_1$ and computes from these a 64-bit cipher-text c. This process is outlined in Figure 1. The encryption (and likewise decryption) of PRINCE can be roughly divided into five phases: Input whitening (with keys k_0 and k_1), application of the round functions R_1 to R_5, a middle layer, application of the inverse round functions R_6^{-1} to R_{10}^{-1}, and finally output whitening (with keys k_1 and $k_2 := (k_0 \ggg 1) \oplus (k_0 \gg 63)$). In the following we provide some details on the particular phases.

After the initial state has been loaded with p, the keys k_0 and k_1 and the round constant RC_0 are XORed onto the state. For more information on the round constants, see [9]. Then the round functions R_i, with $1 \leq i \leq 5$, are applied, which consist of four operations each: First the 16 nibbles (4-bit words) of the state are substituted using a 4-bit SBox S. The SBox is the only non-linear operation in PRINCE. It is required to prevent an adversary from modeling the primitive as a system of linear equations and then solving it for the key bits, thereby breaking the cipher. The next operation is the multiplication of the state

by a 64×64 binary matrix M, which accomplishes diffusion on bit-level. The last two operations consist of addition of the round constant RC_i and the key k_1. Details on the SBox S, diffusion matrix M and round constant RC_i can be found in [9].

The middle layer consists of the application of the SBox S, followed by the multiplication of a 64×64 binary matrix M' ($\neq M$) and finally by the application of the inverse SBox S^{-1}. Details on the matrix M' can be found in the specification of PRINCE. In the next phase the inverse round functions R_i^{-1}, with $6 \leq i \leq 10$, are applied. Basically each of those consists of the inverse operations as done in the normal rounds, applied in reversed order: First the key k_1 is XORed to the state, followed by a round constant RC_i. Then the inverse diffusion matrix M^{-1} and the inverse SBox S^{-1} are applied to the state.

Once all inverse rounds are processed, the final round constant RC_{11} and the keys k_1 and k_2 are XORed to the state, which ultimately results in the 64-bit cipher-text c. Again more details on the round constants $RC_6, \ldots, RC_{11}$ can be retrieved from the specification document of PRINCE.

B. ASIC Implementation of PRINCE

We designed an ASIC that implements the PRINCE function using the 45nm Nangate Open Cell Library. The circuit is combinatorial and performs one full encryption per clock cycle. Its gate count is 8,320 and its power dissipation was estimated as 41.2 mW, compared to a gate count of 8,260 and a power consumption of 38.5 mW of the reference implementation. For obtaining the fault injection statistics shown in this section, we performed full electrical-level simulation over 40 different instances of PRINCE using NanoSim™. To generate different instances, process variations in terms of threshold voltage variations were assigned from a Gaussian distribution with 3σ deviation of 150mV in order to be consistent with ITRS specifications [24].

C. Voltage & Temperature Manipulation Based Fault Injection

To inject faults in PRINCE, we consider low cost techniques like voltage and temperature manipulation [2]. In [2] the cost of fault injection using the above techniques is identified to be less than $3,000. For fault injection, we varied the supply voltage in terms of 2mV (although more fine tuning is possible [14]) and temperature by 5°. We assume the ideal values for supply voltage and temperature to be 1.1V and 25° in all computations.

We investigated the impact of under-powering the device implementing the PRINCE circuit. As the supply voltage is reduced, transient faults are injected into the circuit ranging from single to multiple bit faults based on the voltage reduction. As the faults are transient in nature, the attacker can perform the computation again with varied settings when the required faults are not injected. The number of correct and faulty computations in PRINCE with different $V_{\rm dd}$ values is shown in Figure 2. There exists a point at which the probability of faulty and correct computations is same. Below this point, the probability of faulty computation increases with reducing $V_{\rm dd}$ and multiple faults may be injected. We performed this experiment at various temperatures and observed an increase in faulty computation probability at high temperatures. For the

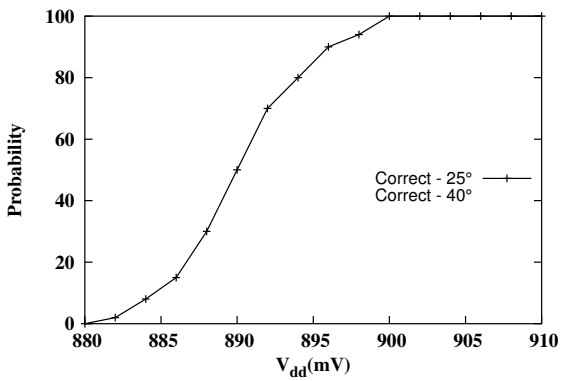

Fig. 2. Percentage of correct and faulty computations as a function of $V_{\rm dd}$ and temperature

sake of brevity, we show the probabilities of faulty and correct computations for two different temperatures (25° and 40°). We can also infer that more faults can be injected at slightly higher supply voltages by increasing the temperature from Figure 2.

The multi-stage attack framework for PRINCE [5] requires that the faults are confined to a single nibble. Therefore, we evaluated the interval of supply voltage over which the required faults are injected to the circuit. The distributions of fault injection in a nibble as a function of supply voltage for three different temperatures (25°, 35° and 45°) are shown in Figure 3. Note that the supply voltage interval shrinks with increasing temperature, as the faults are likely to be injected in multiple bits of a nibble at high temperatures. However, beyond certain temperature, the faults spread to multiple nibbles and they are not exploitable in our attack. Through simulations, we observed a rapid increase of the likelihood for faults in multiple nibbles with temperatures beyond 60°.

Given that there exist successful supply voltage and temperature points for fault injection, we analyzed further fault injection statistics in PRINCE. The probability of fault injection in the state over various rounds in PRINCE is shown in Figure 4. We can observe that the fault injection is uniformly distributed over all the rounds. Figure 5 shows the distribution of the number of faulty nibbles in the 64-bit state of PRINCE. It can be observed that the probability of fault injection over all

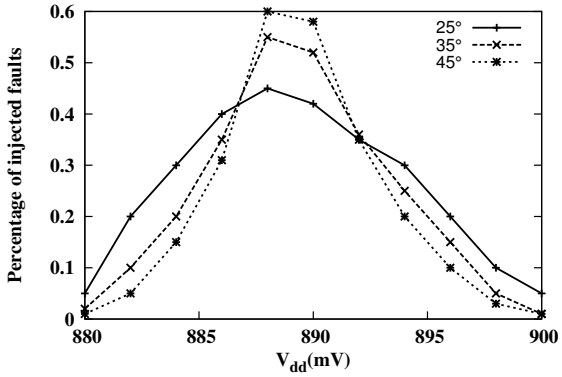

Fig. 3. Percentage of injected faults in a nibble as a function of $V_{\rm dd}$ and temperature

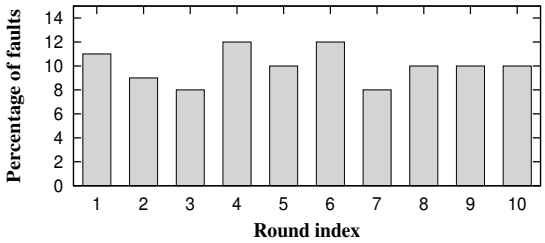

Fig. 4. Percentage of injected faults per round in PRINCE

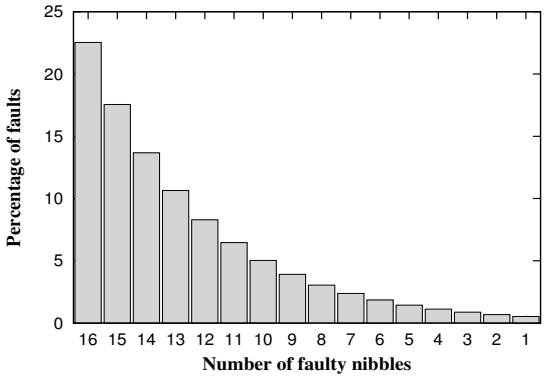

Fig. 5. Percentage of injected faults as a function of number of faulty nibbles

the nibbles is higher than the fault injection in reduced number of nibbles. The fault injection in a single nibble happens with the lowest probability over the supply voltage and temperature interval shown in Figure 3. By observing Figures 4 and 5, the probability of exploitable faults as required by the multi-stage attack framework can be estimated. As the faults in rounds 8 and 9 are exploitable, the probability of successful fault injection is around 0.1% ($= 2 \times 0.005 \times 0.1$, where 0.005 is the percentage of single-nibble faults from Figure 5 and 0.1 is the percentage of faults in nibble 8 or 9 among these faults, as seen in Figure 4). If a fault is not exploitable, the cryptanalysis will not yield the correct secret key as the solution, and the attacker will have to perform further fault injections. This means that around 1000 fault injections are required to get one exploitable fault. Please note that the total fault injections required to get a single exploitable fault might require different voltage and temperature points within the interval shown in Figure 3. We also evaluated the percentage of fault injection in a single nibble and the statistics are shown in Figure 6. The probability of 4-bit errors in a nibble was found to be higher than 3-, 2-, or 1-bit errors. It is important to note that even 4-bit errors in a nibble are useful from our attack perspective.

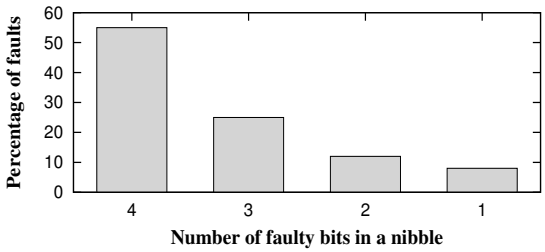

Fig. 6. Percentage of faulty bits in a nibble

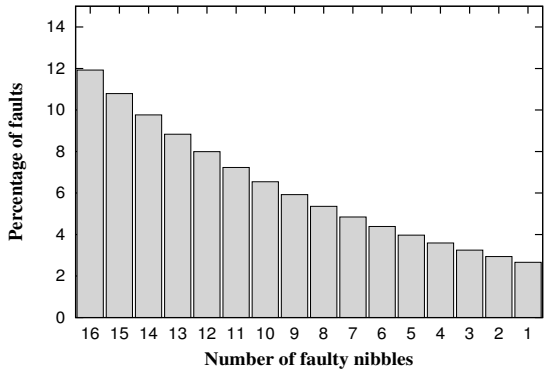

Fig. 7. Percentage of injected faults as a function of number of nibbles in a sequential design

D. Fault Injection in Sequential Implementations

All the discussions above are for a combinatorial design of PRINCE. But, it is interesting to analyze the fault injection statistics in a sequential cipher. As it is feasible to under-power the device over a particular clock cycle, the percentage of exploitable faults can be increased. To analyze this effect, we implemented a sequential design of PRINCE using the same Nangate 45nm library. The sequential design takes one clock cycle for every round and hence one encryption/decryption takes 10 clock cycles. Although the number of clock cycles is higher, the frequency of operation is much higher ($\sim$ 1.7GHz) when compared to the combinatorial design ($\sim$ 150MHz).

Figure 7 shows the percentage of injected faults as a function of the number of nibbles of the 64-bit state. The statistics correspond to both rounds 8 and 9, as faults in those rounds are exploitable for our framework. As switching the supply-voltage takes a considerable amount of time (for example switching the supply voltage by $\pm$ 0.1V takes about 20ns [25]), some faults are injected into rounds 8 and 9 simultaneously. Such double faults are not exploitable by our framework and hence those faults are discarded. From Figure 7, we can infer that the percentage of such exploitable faults is about 2.5%, which is substantially higher than for the combinatorial design.

E. Multi-Stage Fault Attack

The fault attack used here for validation of the fault-injection technique was introduced in [5], [7]. While details can be found in these publications, this section focuses on its generic principles that are necessary for understanding how the fault injection interacts with the mathematical analysis.

Before we describe the structure of the attack we want to remark that the secret key $k = k_0 \parallel k_1$ and the derived key k_2 are used internally during encryption, but are not directly observable by the attacker. In other words, the implementation is treated as a black-box, where only in- and outputs can be observed and recorded. Additionally, we allow an attacker to inject faults during a query to the black-box, as described in Sections III-C and III-D, in order to obtain the faulty cipher-texts. The purpose of the attack is to reconstruct the values for k from the retrieved in- and (faulty) outputs.

In the beginning the adversary queries the black-box to encrypt a plain-text p, and records the resulting cipher-text

c. Then, the encryption is repeated while a fault (a physical disturbance) is injected during the calculation. The faulty cipher-text c' is observed at the output and recorded. During cryptanalysis the relation between c and c' is used to restrict the number of key candidates. Each additional fault injection restricts the number of key candidates even further. Once the number of remaining key candidates falls below a certain threshold (e. g., 2^{25}), an exhaustive search over the remaining keys is performed using the PRINCE circuit and a recorded plain-text, cipher-text pair (p, c) with the aim to determine the correct key k.

It is important that the attack assumes a mathematical model of fault injection that is, in general, not electrically accurate. The attack discussed here requires a fault that affects one of the 16 four-bit nibbles of a particular state in the PRINCE circuit whereas no further faults occur in any other nibble of that state nor in any other state. This is not necessarily consistent with the effects of manipulating supply voltage and/or temperature: they may lead to faults in multiple nibbles simultaneously. Such faults are not exploitable for the attack. We show by simulations that a sufficiently high percentage of faults are exploitable.

The attack on PRINCE is organized in two stages. In stage 0, the expression $(k_1 \oplus k_2)$ is treated as the secret key and a small number of candidates for $(k_1 \oplus k_2)$ are generated. For this purpose, a fault is injected into the state in the beginning of round R_9^{-1}. Let this state be s_9 in the fault-free circuit and s'_9 in the fault-affected circuit. Note that the attacker cannot observe s_9 nor s'_9, yet if the fault is exploitable, then the Boolean difference $(s_9 \oplus s'_9)$ must equal the injected fault f, with 1-bits of f restricted to one nibble, according to the assumptions on the fault injection.

While applying operations from rounds R_9^{-1} and R_{10}^{-1} perturbs the states s_9 and s'_9 in an unpredictable way, some properties of the difference of these states can be derived. For example, the first operations applied to s_9 and s'_9 are XORs with k_1 (unknown to the attacker) and RC_6 (known to the attacker). Since neither k_1 nor RC_6 are affected by fault injection, the Boolean difference of the fault-free and the faulty states after the XOR operation will perfectly match the prior difference f. More complex propagation rules have been derived for other operations (M^{-1} and S^{-1}), see [5]. Using these rules, equations that describe the difference of states s_{10} and s'_{10} before the SBox application in round R_{10}^{-1} are generated.

The restriction of the set of key candidates is achieved by matching the Boolean difference $s_{10} \oplus s'_{10}$ described by the above-mentioned equations with the Boolean difference between the same states calculated backwards from the circuit outputs. Since c is the observed fault-free cipher-text, the correct state s_{10} is given by $s_{10} = S(c \oplus k_2 \oplus k_1 \oplus RC_{11})$. Here, c and RC_{11} are known to the attacker and S is the inverse of the last SBox S^{-1} of round R_{10}^{-1}. Similarly, the fault-affected state is $s'_{10} = S(c' \oplus k_2 \oplus k_1 \oplus RC_{11})$. The resulting system of equations $s_{10} \oplus s'_{10} = S(c \oplus k_2 \oplus k_1 \oplus RC_{11}) \oplus S(c' \oplus k_2 \oplus k_1 \oplus RC_{11})$ restricts possible values of $(k_1 \oplus k_2)$. A further fault injection would result in a different faulty cipher-text c'' and a smaller set of key candidates.

The second stage of the attack is applied for every $(k_1 \oplus k_2)$ candidate determined in the first stage. It aims at determining the sub-key k_1 based on the fault injection in the beginning of round R_8^{-1}. Let $\tilde{s}_8$ denote the fault-free state and $\tilde{s}'_8$ denote the faulty state in that position, and let $\tilde{s}_9$ and $\tilde{s}'_9$ denote the fault-free and faulty states before the SBox in round R_9^{-1}, respectively. Similarly to the first stage, the Boolean difference $e = \tilde{s}_8 \oplus \tilde{s}'_8$ is propagated to yield equations that describe $(\tilde{s}_9 \oplus \tilde{s}'_9)$. Let the fault-free cipher-text be $\tilde{c}$ and the faulty cipher-text be $\tilde{c}'$. Since $(k_1 \oplus k_2)$ is fixed, the fault-free state $\tilde{s}_{10}$ directly after the XOR operation with k_1 in round R_{10}^{-1} is obtained by $\tilde{s}_{10} = RC_{10} \oplus M(S(RC_{11} \oplus (k_1 \oplus k_2) \oplus \tilde{c}))$. The corresponding fault-affected state is $\tilde{s}'_{10} = RC_{10} \oplus M(S(RC_{11} \oplus (k_1 \oplus k_2) \oplus \tilde{c}'))$. By further backwards propagation, the system of equations $(\tilde{s}_9 \oplus \tilde{s}'_9) = S(k_1 \oplus \tilde{s}_{10}) \oplus S(k_1 \oplus \tilde{s}'_{10})$ is obtained and can be used to restrict the key candidate set for k_1.

IV. Experimental Results

In this part we report on the experimental results of the differential fault analysis of PRINCE using our proposed voltage and temperature based fault injection techniques. The analysis was executed on a workstation with an AMD Opteron Processor 6172 operating at 2.1 GHz. The attack was executed 10000 times, in each case on a data set of 10 triples (C, C', C'') where C denotes the correct and C' and C'' the faulty cipher-texts of stage 0 and 1, respectively. The cipher-texts were generated from plain-texts and keys chosen uniformly at random, but the key remained fixed for each set of 10 triples. To generate faulty cipher-texts, full electrical-level simulation was performed under the required supply voltage and temperature points using Nanosim. Table I summarizes the results of the attack and shows that on average between 4 and 5 faults are necessary to successfully reconstruct the 128-bit key.

TABLE I. Overview on the number of required faults

Stage	Min	Max	Avg	Median
0	2	3	2.06	2.0
1	2	10	2.85	3.0
$Total$	4	13	4.91	5.0

We modified the thresholds for the Multi-Stage Fault Attack algorithm to rather low values: 2^{12}, for stage 0 and 2^{16} for stage 1. In general, higher thresholds lead to less required fault injections but increase the complexity of subsequent post-processing. Fault injections based on voltage and temperature modifications are very cheap, thus we decided to use low threshold values to improve the running time of the cryptanalysis algorithm. The downside of these low thresholds is, that we require, on average, one more fault injection as compared to [5]. With this configuration, the run time of post-processing (implemented in Python, with no parallelization) was around 11 seconds.

We also analyzed the distributions of the number of remaining keys after one and two fault injections and compared them to the theoretical results. Note that the fault injections mentioned here in the context of cryptanalysis refer to exploitable faults, each of which requires several (around 1,000) fault injections as explained above. The distributions are shown in Figure 8. The upper diagram shows the results for stage 0 and the lower for stage 1. The x-axis denotes the logarithms with respect to base 2 of the number of key candidates and

the y-axis shows the (rounded) rate how often a particular number of key candidates occurred. There are cases where much more faults (up to 10) are required, but 2 faults were the minimum for every instance. As every exploitable fault requires a reasonable number, around 1,000 fault injections, a complete attack using 4 – 5 exploitable faults is feasible.

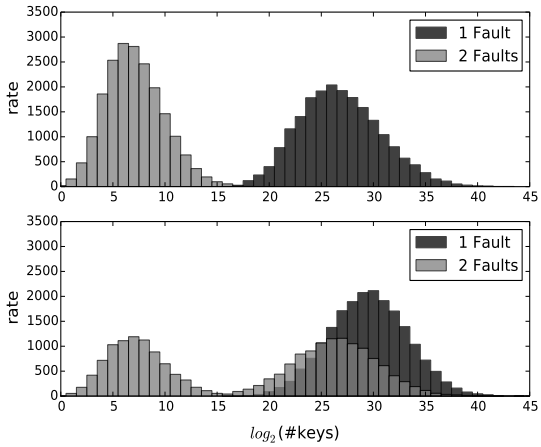

Fig. 8. Analysis results for stage 0 (upper) and 1 (lower)

V. Conclusion

We demonstrated that low-cost voltage and temperature manipulations can be used for high-precision fault injection required to break recent ciphers. The attack requires significant lowering of V_{dd} and is thus detectable by a sensor that monitors the integrity of the power supply. Moreover, several thousands encryptions are required for a successful attack. While this is feasible, some applications may regularly exchange the secret key. This would be an effective (though not perfect) countermeasure if the number of encryptions between key exchanges are small, e. g., less than 1,000. In circuits without V_{dd} integrity sensor and secret key exchange, the attack is highly effective and can be mounted using cost-effective equipment. We showed that the PRINCE cipher requires around 5 exploitable fault injections, which translates to roughly 5,000 total fault injections, for key recovery. The approach is easy to generalize to other lightweight ciphers. Since their circuit implementations are comparable in size to PRINCE, and attack schemes with same or lower number of required fault injections are known for them, similar complexity can be expected for a successful attack.

References

[1] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Elimination Errors in Cryptographic Computations," *Jour. Cryptology*, vol. 14, pp. 101–119, 2001.

[2] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault Injection Attacks on Cryptographic Devices: Theory, Practice and Countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[3] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, ser. LNCS, vol. 6633, 2011, pp. 224–233.

[4] P. Jovanovic, M. Kreuzer, and I. Polian, "A Fault Attack on the LED Block Cipher," in *COSADE*, ser. LNCS, W. Schindler and S. Huss, Eds., vol. 7275. Springer Berlin Heidelberg, 2012, pp. 120–134.

[5] P. Jovanovic, M. Kreuzer, and I. Polian, "Multi-Stage Fault Attacks on Block Ciphers," Cryptology ePrint Archive, Report 2013/778, 2013.

[6] N. Bagheri, R. Ebrahimpour, and N. Ghaedi, "New Differential Fault Analysis of PRESENT," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, pp. 1–10, 2013.

[7] L. Song and L. Hu, "Differential Fault Attack on the PRINCE Block Cipher," Cryptology ePrint Archive, Report 2013/043, 2013.

[8] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The Sorcerers Apprentice Guide to Fault Attacks," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[9] J. Borghoff *et al.*, "PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications," in *Advances in Cryptology – ASIACRYPT 2012*, ser. LNCS, X. Wang and K. Sako, Eds., vol. 7658. Springer Berlin Heidelberg, 2012, pp. 208–225.

[10] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology CRYPTO 96*. Springer Berlin Heidelberg, 1996, vol. 1109, pp. 104–113.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology CRYPTO 99*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, vol. 1666, pp. 388–397.

[12] D. Agrawal, B. Archambeault, J. Rao, and P. Rohatgi, "The EM sidechannel(s)," in *Cryptographic Hardware and Embedded Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2523, pp. 29–45.

[13] R. Leveugle, "Early analysis of fault-based attack effects in secure circuits," *IEEE Transactions on Computers*, vol. 56, no. 10, pp. 1431–1434, 2007.

[14] A. Barenghi, G. Bertoni, E. Parrinello, and G. Pelosi, "Low voltage fault attacks on the RSA cryptosystem," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009 Workshop on*, 2009, pp. 23–31.

[15] A. Barenghi, C. Hocquet, D. Bol, F.-X. Standaert, F. Regazzoni, and I. Koren, "Exploring the feasibility of low cost fault injection attacks on sub-threshold devices through an example of a 65nm AES implementation," in *RFID. Security and Privacy*, 2012, vol. 7055, pp. 48–60.

[16] F. Amiel, C. Clavier, and M. Tunstall, "Fault analysis of DPA-resistant algorithms," in *Fault Diagnosis and Tolerance in Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4236, pp. 223–236.

[17] J. marc Schmidt and M. Hutter, "Optical and EM fault-attacks on CRT-based RSA: Concrete results," pp. 61–67, 2007.

[18] J. Schmidt and C. Herbst, "A practical fault attack on square and multiply," in *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC '08. 5th Workshop on*, 2008, pp. 53–58.

[19] G. Canivet, P. Maistri, R. Leveugle, J. Cldire, F. Valette, and M. Renaudin, "Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA," *Journal of Cryptology*, vol. 24, no. 2, pp. 247–268, 2011.

[20] N. Selmane, S. Guilley, and J.-L. Danger, "Practical setup time violation attacks on AES," in *Dependable Computing Conference*, 2008, pp. 91–96.

[21] L. Antoni, R. Leveugle, and B. Feher, "Using run-time reconfiguration for fault injection in hardware prototypes," in *IEEE Defect and Fault Tolerance in VLSI Systems*, 2000, pp. 405–413.

[22] C. Lopez-Ongil, M. Garcia-Valderas, M. Portela-Garcia, and L. Entrena, "Autonomous fault emulation: A new FPGA-based acceleration system for hardness evaluation," *IEEE Transactions on*, vol. 54, no. 1, pp. 252–261, 2007.

[23] I. Polian and M. Kreuzer, "Fault-based attacks on cryptographic hardware," in *Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2013, pp. 12–17.

[24] "International technology roadmap for semiconductor (ITRS)," 2006.

[25] W. Kim, M. S. Gupta, G. yeon Wei, and D. M. Brooks, "Enabling OnChip Switching Regulators for Multi-Core Processors Using Current Staggering," in *In Proceedings of the Work. on Architectural Support for Gigascale Integration*, 2007.