Too Easy Fault Injection Attacks on Learning with Rounding (LWR)

Francesco Berti
EnICS, Selecsys Labs
Bar-Ilan University
Ramat-Gan, Israel
francesco.berti@biu.ac.il

Sasha Petri
TU-Darmstadt
Darmstadt, Germany
sasha.petri@outlook.de

Itamar Levi
EnICS, Selecsys Labs
Bar-Ilan University
Ramat-Gan, Israel
itamar.levi@biu.ac.il

***Abstract*—We present an extend-and-prune fault Injection attack on serial implementations of Learning With Rounding that drop the least-significant bits. By iteratively isolating and recovering progressively larger key portions via faults, the attack recovers the secret key.**

***Index Terms*—cryptography, hardware security, fault injection, learning with error, LWR**

I. Introduction

One of the most popular countermeasures against side-channel attacks (SCA) [1] is *masking* [2]–[5], which splits a sensitive value into several shares and performs the computation on these shares. Unfortunately, such countermeasures can be very expensive for block-ciphers, as the overhead is typically quadratic in the number of shares (or the security level). Alternatively, there are cryptographic primitives which can be used for (e.g.,) encryption that can be masked more easily. For example Learning Parity with Noise (LPN) [6], and its physical counterparts, Learning Parity with Physical Noise (LPPN) [7]–[11], or Learning with Rounding (LWR) [7]. In these primitives, the output is a "noisy" version of the inner product $\mathbf{k} \cdot \mathbf{x}$ of the key $\mathbf{k}$ with a random vector $\mathbf{x}$. The output is made noisy, either by the addition of a random bit, that equals 1 with probability $\epsilon < 1/2$ (LPN), or by making the computation incorrect with probability ϵ by running the circuit computing $\mathbf{k} \cdot \mathbf{x}$ at too low voltages or too high frequency (LPPN [8]), or by giving a rounded version of the output (LWR). For typical security levels LWR constructions require computing dot-product on *thousands* of elements, *each* represented by 16/32-bits. Note that in the context of this work which focuses on serial implementations, the motivation is clear as parallelism in such constructions would be quite limited in most software and hardware platforms.

An adversary having physical access to a device performing a cryptographic computation, can obtain useful information by injecting faults [12]. Fault injection (FI) attacks can be highly precise using localized injections. On advanced process nodes, attackers can even target individual registers or gates, for example via Laser L-FI [13] or electromagnetic EM-FI [14], [15]. Berti and Standaert [16] have proved that it is possible to attack a serial implementation of LPN by setting some bits to 0 (whereas faults that flip are useless). Their idea is to isolate a key nibble, then, use a maximum-likelihood strategy to recover it. Then, they apply an extend-and-prune strategy, that is, they isolate a bigger key piece containing the key nibble recovered, and they use a maximum likelihood strategy to recover it, exploiting the key nibble already recovered. This attack remains effective even under imprecise fault locations (although performance degrades).

Our contributions: We consider the case where the key $\mathbf{k} \in \mathbb{Z}_{2^q}^n$ and the rounding outputs is the p most significant bits of $\mathbf{k} \cdot \mathbf{x}$. We prove that it is possible to apply the same strategy as Berti et al. [16] against a serial implementation of LWR [1]. We show that an attack is possible *even when only one bit can be faulted per computation*.

II. Preliminaries

We denote by $x \xleftarrow{\$} \mathcal{X}$ sampling x uniformly from $\mathcal{X}$. Let $\mathbf{x}$ be a vector in $\mathbb{Z}_{2^q}^n$, and we denote its components by $\mathbf{x} = (x_1, \ldots, x_n)$. Let $x_{i,j}$ be the j^{th} bit of the i^{th} component of $\mathbf{x}$. Note that $x_i = \sum_{j=0}^{q-1} 2^j x_{i,j}$. We denote with $(\mathbf{k}, \mathbf{x})$, often simplified with $\mathbf{k} \cdot \mathbf{x}$, the inner product.

Learning with Rounding (LWR): Given $q', p' \in \mathbb{N}, q > p$, the *rounding function* $\lfloor \cdot \rceil : \mathbb{Z}_{q'} \to \mathbb{Z}_{p'}$ is defined as $\lfloor x \rceil := x \cdot p'/q'$. When $q' = 2^q$ and $p' = 2^p$, the rounding function has a very easy description: $\lfloor x \rceil := x_{,q-1} \cdots x_{,q-p}$, where x is the string $x_{,q-1} \ldots x_{,0}$. Note that we use x_i to denote the ith coordinate of the vector $\mathbf{x}$, while $x_{i,j}$ denotes the jth most significant bit of x_i (to avoid confusion we use $x_{,j}$ to denote the jth most significant bit of x).

Given $q, p, n \in \mathbb{N}$ with $q > p$, given a random $\mathbf{k} \xleftarrow{\$} \mathbb{Z}_{2^q}^n$, the LWR *distribution* $\mathsf{LWR}_{n,p,q,\mathbf{k}}$ is defined as $(\mathbf{x}, \lfloor \mathbf{k} \cdot \mathbf{x} \rceil)$ with $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_{2^q}^n$. The *decision problem* is to distinguish the $\mathsf{LWR}_{n,p,q,\mathbf{k}}$ from a random distribution and the *search problem* is to find $\mathbf{k}$, given $\mathsf{LWR}_{n,p,q,\mathbf{k}}$ [7].

III. The reduced LWR

First, exploiting a ring homomorphism, we can assume that $q = p + 1$. In fact, given the ring homomorphism $\phi_{q,p+1}$:

I. Levi and F. Berti funded by Israel Science Foundation, ISF grant 2569/21.

[1]Serial implementations are expected to dominate in practice, as parallel LWR designs incur significant hardware cost and most target platforms are resource-constrained.

$\mathbb{Z}_{2^q} \to \mathbb{Z}_{2^{p+1}}$ that keeps the $p+1$ least significant bits, if we can recover the $p-1$ least significant of each component of $\mathbf{k}$, then we can recover the full key $\mathbf{k}$ by simply solving a linear system. Fig. 1 illustrates the behaviour of the inner product, the ring homomorphism, and the roundings. Assuming $q = p+1$

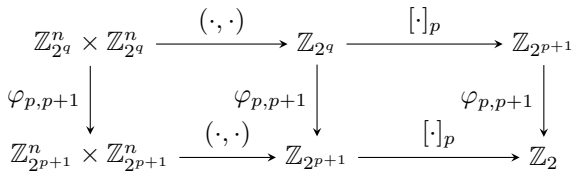


Fig. 1. This commutative diagram represent how the inner product, the rounding and ring homomorphism are connected.

means that $\lfloor \mathbf{k} \cdot \mathbf{x} \rceil_p$ consists of a single bit.

IV. Attack setting q bits to 0

Let us suppose that $\mathbf{k} \cdot \mathbf{x}$ is computed in a serial way. Let $s_i := \sum_{j=i}^{n} k_j x_j$: we compute first s_n, then we compute s_{n-1} by $s_{n-1} = s_n + k_{n-1}x_{n-1}$. Consequently $\lfloor \mathbf{k} \cdot \mathbf{x} \rceil_p := \lfloor s_1 \rceil_p$. We depict this in Fig. 2. Let us suppose that we can set s_2 to 0 by a fault, that is we set the q bits of s_2 to 0 by q bit setting faults (see Fig. 3). Then the faulted LWR distribution will give us $\lfloor k_1 x_1 \rceil_p$, therefore we have isolated a key nibble. To recover k_1 we can collect many samples of $\lfloor k_1 x_1 \rceil_p$ from different $\mathbf{x}$. By exhaustive search, we can recover k_1 because after $O(q)$ fault queries the number of possible x_1 that are consistent with the faulted output will shrink to one.

After having recovered k_1, we can go on recovering k_2 as follows: suppose that we set to 0 all bits of s_3, then we obtain $\lfloor k_2 x_2 + k_1 x_1 \rceil_p$, therefore we have isolated a bigger key nibble that contains k_1, the key nibble that we have already recovered. Since we know x_2 and x_1 ($\mathbf{x}$ is given as outputs of LWR), and k_1 we can recover k_2 by exhaustive search on all possible candidates. After $O(q)$ sample, we should recover it.

Then, we proceed according to the extend-and-prune approach until we have recovered the complete key.

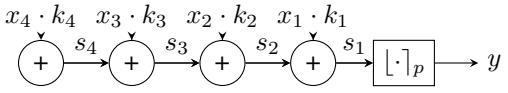


Fig. 2. A serial implementation an LWR instance implemented in a serial way (with $n = 4$).

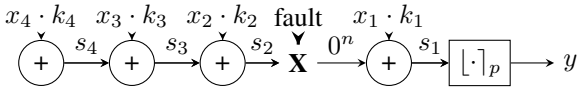


Fig. 3. The first step of our attack setting q bits to 0. We set s_2 to 0^n at $\mathbf{X}$.

V. Attack setting 2 bits to 0

Now, we suppose that we are not able to set q bits to zero, but only 2 bits. Let us suppose that we set to 0 $s_{2,q-1}$ and $s_{2,q-2}$. Now, we consider the output of the faulted LWR. We observe that for all candidate keys k_1^* if $k_1^* x_1$ has its second most significant bit equal to 0(that is, $k_1^* x_1 = *0 * \ldots *$) , then the faulted output is equal to $\lfloor k_1 x_1 \rceil_p$ because the fact that both $s^*_{2,q-2}$ and $(k_1^* \cdot x_1)_{,q-2}$ are equal to 0, (s_2^* is s_2 modified with faults), implies that there is no carryover for the $q-2$ bits in $s^*_{2,q-2} + (k_1^* \cdot x_1)_{,q-2}$. Therefore, for each different x_1 we can assess the correctness of at most half candidates key. As x_1 is random, the correctness check is done each time on a random subset of key candidates, therefore we should be able to recover k_1 with exhaustive search using twice the number of samples as setting q bits to 0.

We can apply the extend-and-prune strategy in a similar way with respect to the q bits fault. Simply, after having recovered $k_1, \ldots, k_{i-1}$, we set to 0 the two most significant bits of s_{i+1}. Therefore, for each candidate key k_i^*, we check whether $(\sum_{j=1}^{i-1} k_j x_j) + k_i^* x_i = *0 * \ldots *$. Again, if the candidate key k_i^* is s.t. $k_i^* x_i = *0 * \ldots *$, we can use the previous idea to assess whether it can be the key or not. After $O(2q)$ samples, we should be able to recover k_i.

Iterating, we can recover the full key following our extend-and-prune strategy.

VI. Attack setting one bit to 0

Suppose that we can now set a single bit to 0. Let us suppose that we can set $s_{2,q-1}$ to 0. Then, assuming that s_2^* (s_2 modified by the fault) is a random value, then, we can assume that the faulted output is equal to the most significant bit of $k_1 \cdot x_1$ with probability proportional to the value of $x_1 k_1$ modulo 2^{q-1}. This probability is the *carry propagation* [2]. That is, the bigger $x_1 k_1$ modulo 2^{q-1} is, the larger the probability that in the sum $x_1 k_1 + s_1^*$ there is a carryover in the position of the corresponding to the 2^{q+1} values.

Therefore, we can use a maximum likelihood approach to recover k_1.

Having recovered $k_1, \ldots, k_{i-1}$, we can apply the extend-and-prune strategy similarly to the previous case.

VII. Attacks on the parallel implementations

Unfortunately, we cannot extend our attack to the parallel implementation because setting to 0 a partial sum isolates completely only half of the key [3]. Therefore, the parallel implementation, as for LPN [16] seems more secure against fault attacks.

VIII. Conclusions and Prospects

We showed that LWR can be attacked with fault injection with some flavors depending on the locality and controllability of the adversary if it is implemented in a *serial / partially serial way*, by setting bits faults. We plan to both simulate and implement this attack on concrete implementations to draw further conclusions and physical limitations in the near future.

[2]The carry propagation is $(x_1 k_1 \text{ modulo } 2^{q-1})\ /2^{q-1}$.

[3]In a parallel implementation, assuming that $n = 2^N$, we have that $s_{0,i} = x_i k_i$, $s_{j,l} = s_{j-1,2l-1} + s_{j-1,2l}$, with $j = [1, \ldots, N]$ and $j = [1, \ldots, 2^{N-j}]$. Therefore, $s = s_{N,1}$

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*, pp. 388–397, Springer, 1999.

[2] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Annual International Cryptology Conference*, pp. 398–412, Springer, 1999.

[3] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: From trivial composition to full verification," *IEEE Transactions on Computers*, vol. 70, no. 10, pp. 1677–1690, 2020.

[4] I. Levi and O. Keren, "Consolidated linear masking (clm): Generalized randomized isomorphic representations, powerful degrees of freedom and low (er)-cost," *Cryptology ePrint Archive*, 2024.

[5] F. Berti, S. Faust, and M. Orlt, "Provable secure parallel gadgets," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2023, no. 4, pp. 420–459, 2023.

[6] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *EUROCRYPT*, 2012.

[7] S. Dziembowski, S. Faust, G. Herold, A. Journault, D. Masny, and F. Standaert, "Towards sound fresh re-keying with hard (physical) learning problems," in *CRYPTO*, 2016.

[8] D. Kamel, F. Standaert, A. Duc, D. Flandre, and F. Berti, "Learning with physical noise or errors," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 5, pp. 957–971, 2020.

[9] D. Bellizia, N. E. Mrabet, A. P. Fournaris, S. Pontié, F. Regazzoni, F. Standaert, É. Tasso, and E. Valea, "Post-quantum cryptography: Challenges and opportunities for robust and secure HW design," in *36th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2021, Athens, Greece, October 6-8, 2021* (L. Dilillo, L. Cassano, and A. Papadimitriou, eds.), pp. 1–6, IEEE, 2021.

[10] D. Bellizia, C. Hoffmann, D. Kamel, P. Méaux, and F. Standaert, "When bad news become good news towards usable instances of learning with physical errors," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 4, pp. 1–24, 2022.

[11] D. Kamel, D. Bellizia, F. Standaert, D. Flandre, and D. Bol, "Demonstrating an LPPN processor," in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, ASHES@CCS 2018, Toronto, ON, Canada, October 19, 2018* (C. Chang, U. Rührmair, D. E. Holcomb, and J. Guajardo, eds.), pp. 18–23, ACM, 2018.

[12] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults (extended abstract)," in *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding* (W. Fumy, ed.), vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, Springer, 1997.

[13] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers* (B. S. K. Jr., Ç. K. Koç, and C. Paar, eds.), vol. 2523 of *Lecture Notes in Computer Science*, pp. 2–12, Springer, 2002.

[14] M. Dehbaoui, J. Dutertre, B. Robisson, and A. Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of aes," in *FDTC*, pp. 7–15, 2012.

[15] S. Ordas, B. Zussa, and P. Maurine, "Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller," in *FDTC*, pp. 77–86, 2014.

[16] F. Berti and F. Standaert, "An analysis of the learning parity with noise assumption against fault attacks," in *CARDIS*, vol. 10146 of *Lecture Notes in Computer Science*, pp. 245–264, Springer, 2016.