Photonic Side Channel Attacks Against RSA

Elad Carmon*, Jean-Pierre Seifert†, Avishai Wool‡
*‡Tel-Aviv University, Tel-Aviv 69978, Israel
†Security in Telecommunications, Technische Universität Berlin and FhG SIT, Darmstadt, Germany

Email: *eladca@gmail.com, †jpseifert@sec.t-labs.tu-berlin.de, ‡yash@eng.tau.ac.il

***Abstract*—This paper describes the first attack utilizing the photonic side channel against a public-key crypto-system. We evaluated three common implementations of RSA modular exponentiation, all using the Karatsuba multiplication method. We discovered that the key length had marginal impact on resilience to the attack: attacking a 2048-bit key required only 9% more decryption attempts than a 1024-bit key. We found that the most dominant parameter impacting the attacker's effort is the minimal block size at which the Karatsuba method reverts to naive multiplication: even for parameter values as low as 32 or 64 bits our attacks achieve 100% success rate with under 10,000 decryption operations. Somewhat surprisingly, we discovered that Montgomery's Ladder—commonly perceived as the most resilient of the three implementations to side-channel attacks—was actually the most susceptible: for 2048-bit keys, our attack reveals 100% of the secret key bits with as few as 4000 decryptions.**

I. Introduction

A. Background

While the phenomena of photonic emission from switching transistors in silicon is actually a very old one, cf. [8], [18], the role of photons in cryptography as a practical side channel source has just recently emerged as a novel research direction, cf. [20], [17], [16], [21], [5], [6]. Thus, it is important to include photonic side channels in future hardware evaluations of security ICs.

However, so far only the first steps within this direction have been successfully achieved. The work of [20], [17], [16], [21], [5], [6], showed that the required equipment to carry out successful SPEA (Simple Photonic Emission Analysis) or DPEA (Differential Photonic Emission Analysis) attacks against real world ICs is comparable in price to that of normal Power Analysis equipment and showed their validity in the AES case. The current paper continues the current state of the art of the Photonic Side Channel and utilizes it to attack the RSA encryption scheme for different (real-world) exponentiation implementaions. In this respect the present paper is the first which considers the RSA case.

The relevance of the photonic side channel is even more important due to new research in this field. A common argument against the validity of the photonic side channel is the claim that it will vanish with smaller technologies due to reduced optical emissions. However, [22] recently performed a photonic emission attack against modern technology IC. Indeed, the gist of their work as relevant to the present paper is the following result: A refined setup of [21] already has much *better* technological capabilities than needed for a photonic side-channel attack against commercially available Altera Cyclone IV FPGAs, manufactured in a modern 60nm process. Thus, it is not surprising that a very recent research tender from the BSI -the Federal Office for Information Security in Germany (cf. [3]) demands a Photonic Emission Analysis of a USB-based FIDO chip [1].

B. Related Work

One of the first uses of PEA in CMOS in a cryptographic application was presented in 2008 [10]. However, the authors increased the voltage supply to 7V operating voltage, which is above the chips maximum limit for voltage. It took the authors 12 hours to recover a *single* potential key byte [10]. In 2011, an integrated PEA system and laser stimulation techniques were used to attack a DES implementation on an FPGA [9]. However, the analysis strongly relied on a specific implementation of DES in which registers were always zeroed before their use. As the authors note, the use of equipment valued at more than 2,000,000 Euros does not make such an analysis particularly relevant.

Nevertheless, recently, a real breakthrough was achieved by [20], [21]. This work presented a novel low-cost optoelectronic setup for time- and spatially resolved analysis of photonic emissions. The authors also introduced a corresponding methodology, named Simple Photonic Emission Analysis. They successfully performed such analysis of an AES implementation and were able to recover AES-128 keys by monitoring memory accesses. The same research group also introduced Differential Photonic Emission Analysis and presented a respective attack against AES-128 [17]. They successfully revealed the entire secret key with their DPEA. In 2015 an enhanced simple photonic emission attack of AES was introduced by [6] where the authors designed a photonic emission simulator which they calibrated against the equipment of [20], and used signal processing and cryptographic post processing in order to apply an attack against AES using less data and got better results. Bertoni et al. [5] also offered an improved Simple Photonic Emission Analysis, monitoring a different section of the SRAM logic. However, they assumed a specific SRAM structure which contains only a single byte in every row. Their simulations model a theoretical case in which the value of every bit can be identified. They also described an attack against a masked AES, however the attack is unrealistic

Algorithm 1 The Binary Method

```
1: Input: $c, d, n$
2: Output: $M = c^d \pmod n$
3: if $d_{k-1} == 1$ then
4:     $M = c$
5: else
6:     $M = 1$
7: end if
8: for $i = k-2$ downto $0$ do
9:     $M = M \cdot M \pmod n$
10:    if $d_i == 1$ then
11:        $M = M * c \pmod n$
12:    end if
13: end for
14: return $M$
```

since it assumes monitoring the photonic emission of a single experiment.

C. Contributions

We consider (attack) three common implementations of the modular multiplication of RSA: the binary ("square and multiply") method, the fixed-window method, and Montgomery's Ladder, all using the Karatsuba multiplication method, for various key sizes and implementation variations.

We first developed an auto-calibrating method to decode the photonic traces, eliminating the need to manually set thresholds. Our decoder has attributes of a "soft decoder": it returns "ambiguous" for bit values where the number of photonic emissions is to too close to the threshold, drastically reducing the post-decoding error-correcting effort.

Then we conducted an extensive evaluation of our attacks against the RSA implementations. We discovered that the key length had marginal impact on resilience to the attack: attacking a 2048-bit key required only 9% more decryption attempts than a 1024-bit key to reach an equivalent success rate. The parameter that has the most dominant impact on the attacker's effort turns out to be the minimal block size at which the Karatsuba method switches to the naive multiplication: even for very small values of this parameter (32 or 64 bits) that are recommended for embedded 8-bit RSA implementations, our attacks achieve 100% success rate under 10,000 decryption operations.

Somewhat surprisingly, we discovered that Montgomery's Ladder—commonly perceived as the most resilient of the three implementations to side-channel attacks—was actually the most susceptible: for 2048-bit keys, the attack reveals 100% of the secret key bits with as few as 4000 decryptions.

Organization. The organization of the present paper is as follows. Section II introduces the modular exponentiation methods and the phototonic emissions side-channel. Section III describes how to attack RSA using the photonic side channel. Section IV explains how we decode the photonic traces. Section V describes our performance evaluation, and we conclude in Section VI. A full version of this paper can be found in [7].

II. Preliminaries and Background

In this paper we focus on 3 exponentiation methods: The Binary method (Algorithm 1), the m-ary method (Algorithm 2)

Algorithm 2 The m-ary

```
1: Input: $c, d, n$
2: Output: $M = c^d \pmod n$
3: Pre-Compute and store $c^\omega \pmod n$ for all $\omega = 2, 3, 4, \ldots, m-1$.
4: Decompose $d$ into $r$-bit windows $F_i$ for $i = 0, 1, 2, \ldots, s-1$.
5: $M = c^{F_{s-1}} \pmod n$
6: for $i = s-2$ downto $0$ do
7:     $M = M^{2r} \pmod n$
8:     if $F_i \neq 0$ then
9:         $M = M \cdot c^{F_i} \pmod n$
10:    end if
11: end for
12: return $M$
```

Algorithm 3 The Montgomery's Ladder

```
1: Input: $c, d, n$
2: Output: $M = c^d \pmod n$
3: $M_1 = c$
4: $M_2 = c^2$
5: for $i = k-2$ downto $0$ do
6:     if $d_i == 0$ then
7:         $M_2 = M_1 * M_2 \pmod n$
8:         $M_1 = M_1^2 \pmod n$
9:     else
10:        $M_1 = M_1 * M_2 \pmod n$
11:        $M_2 = M_2^2 \pmod n$
12:    end if
13: end for
14: return $M_1$
```

and Montgomery's Ladder (Algorithm 3). Common to all exponentiation methods is the need to perform many large number multiplications. The naive multiplication algorithm for two n bit numbers requires $\Omega(n^2)$ digit multiplications. A more efficient multiplication algorithm is the Karatsuba algorithm [15] requiring at most $n^{\log_2 3}$ digit multiplications. In order to calculate $a \cdot b$ the algorithm splits the multiplicands, each of length n, into two halves each. The product can then be formed using only three recursive multiplications of length $n/2$ and some additions and subtractions

A crucial parameter of the Karatsuba algorithm is the recursion depth, and the minimal size B below which the naive multiplication is preformed. According to [12], the implementation in the GnuPG library [2] stops the recursion at a block size of $B = 512$ bits. Hutter and Schwabe [14] report that on the 8-bit ATMega architecture Karatsuba multiplication is faster than the naive method for surprisingly small inputs, starting at 48 or 64 bits, which in our terminology implies stopping the recursion at a block size of $B = 24$ or $B = 32$.

III. Attacking RSA with photonic side channel

The temporary values of the modular exponentiation are stored in an internal RAM array of the IC performing the RSA decryption. In each step of the modular exponentiation, the temporary value is read from the memory, some computations are made involving other RAM locations, and the temporary result is stored. By monitoring the row access transistor of a memory row containing bytes involved in the calculation, we can deduce the course of the algorithm and find the private exponent d.[1]

When using Karatsuba multiplication, the algorithm recursively splits the multiplicands until it reaches the minimal size B, where the algorithm performs a naive multiplication between partial parts of the multiplicands. For this naive multiplication the B-bit parts of the multiplicands need to be read from the memory, generating accesses to the relevant memory section. After conducting the naive multiplications of size B the temporary results are not rewritten to the same RAM addresses—instead they are kept in the stack, i.e., in locations not controlled by the monitored row transistor. So the only accesses to the monitored RAM memory storing the multiplicands takes place during the naive multiplication phase of the minimal size B.

A. *Attacking the binary method*

For the binary method algorithm (Algorithm 1), we monitor the memory section storing c. During the loop operation we shall observe accesses for every private key bit $d_i = 1$. Thus whenever we detect accesses we can deduce that the current private key bit is 1, whereas for time periods lacking accesses— the private key bit is 0. This way we can recover the private key.

When the Karatsuba algorithm is used, the multiplication step ($M*c \pmod n$) is conducted recursively and the memory area storing c is only accessed in "limbs" of size B at a time. So, as stated above, when monitoring a single memory row, storing several bytes of c, the number of expected row accesses depends on the size of B. Note that the ATmega328P processor has an 8-bit data bus, hence every RAM byte that is read during the naive multiplication potentially generates an observable memory access.

B. *Attacking the m-ary method*

For the m-ary method (Algorithm 2) we place our detector over the row access transistor of some row containing one of the precomputed variables c^ω (line 3 in Algorithm 2), for example c^2 for the $m = 4$ case (2-bit windows). Now, during the loop operation, for every window of private key bits equaling '10', the operation $M * c^2 \pmod n$ (line 9) will take place, generating accesses to the memory section we are monitoring and revealing the corresponding 2 bits of the private key.

C. *Attacking Montgomery's Ladder*

When attacking Montgomery's Ladder (Algorithm 3), we can monitor either of M_1 or M_2. For example when we monitor M_1, for $d_i = 0$ we would have M_1 multiplied, squared and reassigned (lines 7-8 in Algorithm 3) which will generate more accesses than when $d_i = 1$, in which case M_1 is only multiplied and reassigned (line 10). By differentiating between these two cases based on the amount of memory accesses, we can reveal the entire private key.

[1]Note that all known embedded RSA hardware-coprocessors also map their long internal registers to the internal RAM array, for easier software handling, and also that existing modern smartcards are still manufactured in 90nm where PEA is known to work.

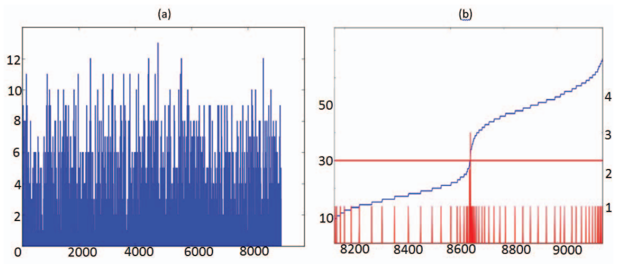

Fig. 1. (a) A photonic trace received from the simulator during RSA decryption for $T = 1,000$. Each point in the x axis corresponds to 220μsec. The y axis counts photonic detections per time step. (b) The sorted trace in blue for $T = 9,000$. The derivative plot is in red where we can spot a peak. The corresponding threshold is shown as a horizontal line.

IV. Decoding an RSA Decryption's Photonic Traces

We activate the IC (or, in our case, the photonic emission simulator) T times to decrypt the ciphertext. For each activation we count the number of detected photons per time step, while the detector is fixed at some SRAM row as described above. The time step corresponds to the specific algorithm and matches the execution time of the operations inside the loop. We summarize the detection counts per time step, to obtain a "photonic trace" (an example of a trace can be found in Figure 1 (a)). Following [19], [20] we assume an IC instruction cycle of 800 ns[2]. For multiplication of 2 n-bit multiplicands M_1, M_2 stored in SRAM memory with a row width of r bytes and minimal Karatsuba size of B bits, the naive multiplication executes $(B/8)^2$ 1-byte multiplications. Assuming a 1-byte memory bus, of the $(B/8)^2$ accesses, only an $8 \cdot r/B$ fraction is observable by a detector placed over one of the rows of one multiplicand, giving:

$$\#\text{observable-row-accesses} = B \cdot r/8 \tag{1}$$

Note that equation (1) differs qualitatively from the effects seen in SPEA against AES [19], [20]: RSA multiplications access *all* the bytes in the row, multiple times, thus having a larger SRAM row width r makes the attack more efficient as it increases the number of observable accesses. In contrast, SPEA against AES with $r = 16$ (as in the ATXMega128A1 IC) requires much more effort than when $r = 8$ since the attacker must identify the specific byte that was accessed in the row.

We now need to decode the trace to distinguish between the two cases of the if statement in order to reveal the value of the private key bit d_i. A natural decoding rule is to use a threshold: if the number of detected activations during the time step exceeds the threshold, the if statement is True.

A. *Threshold Calibration*

A crucial task is calibrating the threshold to reliably distinguish between true detections and noise. Instead of a using

[2]Note that this clock frequency is a slow 1.25MHz. This clock frequency was calibrated to the real lab setup of [19], [20].

a heuristic trial-and-error process, we calibrate the threshold automatically to an optimal value.

When monitoring accesses to an SRAM row containing a variable of the exponentiation algorithm, we have two cases: (i) "some-to-zero": distinguish between SRAM accesses versus no accesses. This is the needed distinguisher when attacking the binary method and monitoring c or when attacking the m-ary method. (ii) "many-to-few": distinguish between many SRAM accesses versus few accesses: this is needed when attacking the binary method and monitoring M, or attacking Montgomery's Ladder.

For the first case, if we sort the "photonic trace" in ascending order, we expect to see a steep rise, a "jump", in the sorted trace (when the SNR is sufficient), between samples containing SRAM accesses and samples containing only noise. By looking at the derivative plot of the sorted photonic trace, we can detect this jump according to a distinct peak in the derivative plot (see Figure 1 (b)). The value of the threshold is set to be the value of the sorted photonic trace where the derivative plot has a peak (the mid-point of the "jump").

For the second case ("many-to-few") we expect to find *two* steep rises in the sorted photonic trace since there will be one jump between samples containing only noise and samples containing the lesser amount of accesses, and another jump between samples containing the lesser amount of accesses and samples containing more SRAM accesses. For this case we take the threshold value to be the value of the sorted trace where the derivative trace has its second peak value.

B. *The Ambiguous Range and Handling Errors*

We refer the reader to our full technical report [7] for details on handling errors.

V. Practical Results

We used the photonic emissions simulator of [6] to simulate an ATmega328P running at 1.25MHz. The experiments were run on an Intel Core Duo T2450 2GHz, 2GB RAM PC running Windows Vista. We simulated the ATmega328P IC with SRAM row width of $r = 8$ and generated the RSA keys according to the PKCS standard.

In order to evaluate the performance of the attack we performed an extensive set of experiments. The experiments were done on the three modular exponentiation methods, with various key sizes, using various various stopping sizes B for the Karatsuba multiplication. We define the success rate as the percentage of private key bits we decode successfully.

A. *Attacking the Binary method and Montgomery's Ladder*

In Figure 2(a) we can see the success rate for the Binary method (recall Algorithm 1) where the detector monitors the memory storing c, as a function of the number of decryptions performed. We can see that with $B = 64$ the success rate approaches 1 around $T = 10,000$ decryptions, while $B = 32$ has a lower success rate and requires twice as many decryptions to achieve the same success rate—as expected by Equation 1.

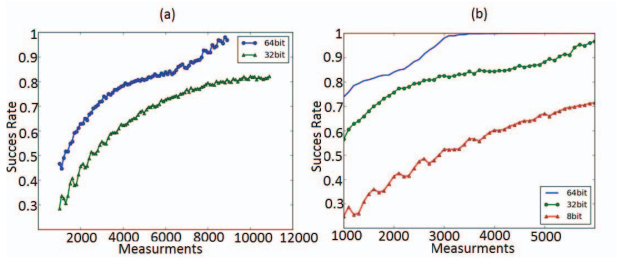

Fig. 2. (a) The success rate for the Binary method as a function of the number of decryptions for a key size of 1024 and a Karatsuba stopping size of $B = 32$ bits (bottom curve) and $B = 64$ bits (top curve). (b) The success rate for Montgomery's Ladder as a function of the number of decryptions for a key size of 1024 and a Karatsuba stopping size of $B = 8$ bits (bottom curve) $B = 32$ bits (middle curve) and $B = 64$ bits (top curve).

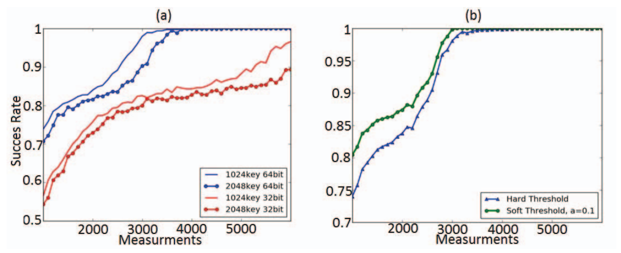

Fig. 3. (a) The success rate for Montgomery's Ladder as a function of the number of decryptions for a key size of 1024 and 2048 and a Karatsuba stopping size B of 32 and 64 bits. (b) The success rate for the Montgomery Ladder using the normal threshold (bottom curve) and the ambiguous range (top curve) as a function of the number of decryptions for a key size of 1024 and $B = 64$ bits.

Figure 2(b) shows the success rate in attacking Montgomery's ladder (Algorithm 3). Again we see the same effect—larger values of the Karatsuba minimal block size B help our attack. Further, comparing Figure 2(a) and (b) we see that our attack is more efficient against Montgomery's ladder, where more memory accesses are generated during the decryption process. This improves the SNR, and hence, allows fewer measurements to achieve the same success rate.

Figure 3(a) examines the effect of the key size n on the success rate. We can see that for a larger key size more decryptions are needed to achieve the same success rate since there are more secret exponent bits to discover—but the difference is not dramatic. E.g., to achieve a 99% success rate against Montgomery's ladder with $B = 64$ we need 3300 decryptions for 1024-bit keys, and 3600 for 2048-bit keys: 9% more.

B. *Attacking the m-ary method*

For the m-ary method (Algorithm 2), when $m = 4$, using one detector we can monitor access to only one out of the four precomputed values stored in memory, e.g., the value c^2. Hence for every 2-bit window in the secret exponent d, for 1/4 of cases there is an observable memory access and we can learn that the bits' value is '10'. For 3/4 of cases there is no observable memory access, so we can learn that the bits' value is *not* '10'. Thus, ignoring what we can learn when there is no observable memory access, the ideal success rate we can hope for when $m = 4$ is 25% (and $1/m$ in general). From an information-theory point of view we learn more: for $n = 1024$ we have 512 2-bit windows, and in 3/4 of these windows we can only have 3 possible values, so the remaining entropy is $512 \cdot 0.75 \cdot \log_2 3 = 608$ bits, i.e., we learn 40.6% of the secret exponent.

Our results show that when attacking the m-ary method, the number of required measurements is as large as in the Binary method, since the number of SRAM accesses is low. Furthermore, as expected, the success rate approaches 25% once sufficiently many decryption measurements become available (graphs omitted).

Thus we see that the m-ary method is much more resilient to our attack than Montgomery's Ladder—in contrast to the resilience to timing or power-analysis side channels, against which Montgomery's Ladder is generally superior.

Note that despite learning only 25% of the secret exponent bits, the attack can perhaps be extended to reveal the missing bits. If the RSA implementation uses the CRT, then after our attack we know the public key n, e exactly, we discover 25% of the bits of d_p and d_q, and these bits are in arbitrary (but known) positions. This situation is close to that described by Heninger and Shacham [13], where the authors developed a solver that reveals the entire key with high probability from a partial key with only 24% randomly located bits. In their case they also assumed a partial knowledge of p and q, which we do not have. On the other hand, we know that the missing bits are constrained: every missing 2-bit window in d_p and d_q *cannot* equal, e.g., the value '10'. Perhaps a similar solver can be designed for our scenario—we leave this as an open question.

An alternative approach is by using multiple detectors: continuing with the example of $m = 4$, if we had 3 detectors, we could place them over the memory areas holding c^1, c^2, and c^3 respectively. By repeating our attack 3 times in parallel we would learn 3/4 of the bits (for all the windows in which one of the detectors observes accesses), and we would reveal the remaining 25% by identifying windows in which none of the 3 detectors observed memory accesses. Using more than one photonic detector was recently demonstrated in [11]. We leave the practical evaluation of the multi-detector attack for future work.

VI. Conclusions and Countermeasures

In this paper we demonstrated, for the first time, that the photonic side channel can be successfully used against the RSA public-key crypto-system. We discovered that the key length had marginal impact on resilience to the attack, while the minimal Karatsuba block size had a dominant effect. Somewhat surprisingly, we discovered that Montgomery's Ladder—commonly perceived as the most resilient of the three implementations we evaluated to side-channel attacks—was actually the most susceptible, while the m-ary method only allowed us to reveal a $1/m$ fraction of the secret exponent bits. This means that resilience to memory-access side channels in general, and the photonic emission side channel in particular, should be considered as important aspects of public-key crypto-system implementations.

Thus, to protect RSA against the photonic side channel, we need to eliminate secret-key-dependent branches from the implementation—and there are ways to do so [4]. More general countermeasures against photonic side channel attacks include varying the memory locations of central variables used by the decryption process, adding dummy operations involving the same memory space access patterns.

References

[1] FIDO alliance. https://fidoalliance.org/about/overview/.

[2] Gnu privacy guard. http://https://www.gnupg.org/.

[3] Projekt 267: Analyse nicht-zertifizierter IT-Sicherheitselemente - Teilprojekt 1: Untersuchung eine JAVA-Karte mit FIDO U2F applet (German). BSI tender: https://www.evergabe-online.de/tenderdetails.html?0&id=122064.

[4] D. J. Bernstein. Personal communication, 2016.

[5] Y. M. Bertoni, L. Grassi, and F. Melzani. Simulations of optical emissions for attacking AES and masked AES. In *Security, Privacy, and Applied Cryptography Engineering (SPACE), LNCS 9354*, pages 172–189. Springer Verlag, 2015.

[6] E. Carmon, J.-P. Seifert, and A. Wool. Simple photonic emission attack with reduced data complexity. In *7'th Constructive Side-Channel Analysis and Secure Design (COSADE'16)*, pages 35–51, Apr. 2016.

[7] E. Carmon, J.-P. Seifert, and A. Wool. Photonic side channel attacks against RSA. Cryptology ePrint Archive, Report 2017/108, 2017. http://eprint.iacr.org/2017/108.

[8] A. Chynoweth and K. McKay. Photon emission from avalanche breakdown in silicon. *Physical Review*, 102(2):369, 1956.

[9] J. Di-Battista, J.-C. Courrege, B. Rouzeyre, L. Torres, and P. Perdu. When failure analysis meets side-channel attacks. In *Cryptographic Hardware and Embedded Systems(CHES)*, pages 188–202. Springer, 2010.

[10] J. Ferrigno and M. Hlavác. When AES blinks: introducing optical side channel. *Information Security*, 2(3):94–98, 2008.

[11] F. Ganji, J. Krämer, J.-P. Seifert, and S. Tajik. Lattice basis reduction attack against physically unclonable functions. In *Proc. 22Nd ACM Conference on Computer and Communications Security (CCS'15)*, pages 1070–1080, 2015.

[12] D. Genkin. personal communications, 2016.

[13] N. Heninger and H. Shacham. Reconstructing RSA private keys from random key bits. In *Advances in Cryptology-CRYPTO 2009*, pages 1–17. Springer, 2009.

[14] M. Hutter and P. Schwabe. Multiprecision multiplication on AVR revisited. *Journal of Cryptographic Engineering*, 5(3):201–214, 2015.

[15] A. A. Karatsuba. The complexity of computations. *Proceedings of the Steklov Institute of Mathematics-Interperiodica Translation*, 211:169–183, 1995.

[16] J. Krämer, M. Kasper, and J.-P. Seifert. The role of photons in cryptanalysis. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 780–787. IEEE, 2014.

[17] J. Krämer, D. Nedospasov, A. Schlösser, and J.-P. Seifert. Differential photonic emission analysis. In *Constructive Side-Channel Analysis and Secure Design*, pages 1–16. Springer, 2013.

[18] R. Newman. Visible light from a silicon pn junction. *Physical Review*, 100(2):700–703, 1955.

[19] A. Schlösser. *Hot electron Luminescence in silicon structures as photonic side channel* (in German). PhD thesis, Faculty of Mathematics and Natural sciences, Berlin Institute of Technology, 2014.

[20] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert. Photonic emission analysis of AES. *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2012.

[21] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert. Simple photonic emission analysis of AES. *Journal of Cryptographic Engineering*, 3(1):3–15, 2013.

[22] S. Tajik, E. Dietz, S. Frohmann, H. Dittrich, D. Nedospasov, C. Helfmeier, J.-P. Seifert, C. Boit, and H.-W. Hübers. Photonic side-channel analysis of arbiter PUFs. *Journal of Cryptology*, pages 1–22, 2016.