IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–10. DOI:XXXXXXXX

Side-channel Attacks on SIMON Block Cipher: CPA and Leakage Assessment on unprotected and masked implementations

Nico Makowe, Kandarp Lalit and Majdi Maraqa

Hochschule Albstadt-Sigmaringen, Albstadt-Ebingen, Germany
{makoweni,lalitkan,maraqama}@hs-ablsig.de

Abstract. This paper investigates implementation-level vulnerabilities of the lightweight block cipher SIMON by means of power-based side-channel analysis. A software implementation on an embedded ARM Cortex-M4 target is analyzed, demonstrating that measuring power consumption enables key recovery using Correlation Power Analysis (CPA). To mitigate these vulnerabilities, boolean masking is implemented. Welch's t-test was performed for statistical leakage assessment. The results show high leakage in the unprotected implementation and significantly lower t-values in the masked implementation. These findings emphasize the importance of careful countermeasure design in lightweight cryptographic implementations.

Keywords: lightweight cryptography, side-channel analysis, correlation power analysis, SIMON, masking, implementation security, ChipWhisperer

1 Introduction

Lightweight cryptography plays a crucial role in securing resource-constrained devices in the Automotive and IoT industry, where limitations in memory, power consumption, and computational capabilities must be carefully considered. The SIMON block cipher, introduced by the National Security Agency in 2013, is a family of lightweight block ciphers designed to achieve high performance in hardware while remaining efficient in software implementations.

SIMON, like many cryptographic algorithms, is susceptible to implementation-level attacks. In particular, side-channel attacks exploit physical leakage such as power consumption or electromagnetic radiation to recover secret information without breaking the underlying cryptographic primitive.

This paper investigates Correlation Power Analysis (CPA) applied to a software implementation of SIMON. Furthermore, the masking countermeasure is analyzed and its effectiveness in increasing resistance against power-based attacks is evaluated. The presented work follows a systematic attack-and-evaluation methodology commonly used in the analysis of cryptographic implementations.

The Simon Block Cipher Simon is a family of lightweight block ciphers based on a Feistel structure. The cipher supports multiple configurations with block sizes of 32, 48, 64, 96, and 128 bits, and key sizes ranging from 64 to 256 bits. [BSS+15]

This study focuses on Simon-64/128, which operates on 64-bit blocks using a 128-bit key over 44 rounds. The cipher is designed for optimal performance in both hardware and software implementations, featuring only bitwise XOR, AND, and rotation operations.

The round function for Simon-64/128 is defined as:

$$R_k(x, y) = (y \oplus f(x) \oplus k, x)$$

where

$$f(x) = (S^1(x) \& S^8(x)) \oplus S^2(x)$$

Here, S^j denotes left circular shift by j bits, $\oplus$ represents bitwise XOR, & denotes bitwise AND, and k is the round key derived from the master key through a key schedule.

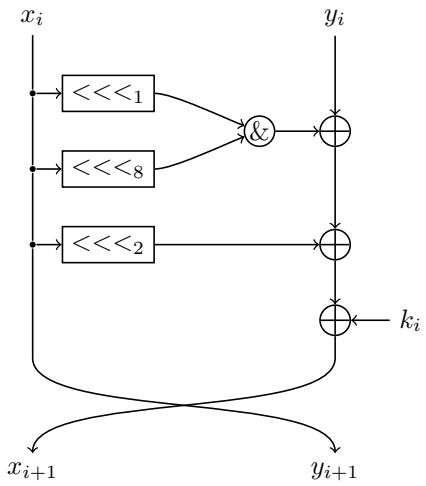


Figure 1: One round of the Simon block cipher

The key schedule for Simon-64/128 generates 44 round keys $k_0, \ldots, k_{43}$ from the 128-bit key k using a linear recurrence relation. The first 4 round keys k_0, k_1, k_2 and k_3 form the original key k.

Side-Channel Attacks Side-channel attacks exploit physical leakage from a device during cryptographic operations. Correlation Power Analysis (CPA) is a powerful statistical attack that correlates hypothetical power consumption models with measured power traces to recover secret keys.

The success of CPA depends on an accurate power model that approximates the device's actual power consumption. Common models include:

- Hamming Weight (HW): Power consumption proportional to the number of 1s in a data word
- Hamming Distance (HD): Power consumption proportional to the number of bit changes between consecutive states

2 Attack on the unprotected Implementation

The experimental setup utilized a 32-bit ARM microcontroller running at 7.38 MHz. An unprotected implementation of Simon-64/128 was written in C. Power traces were collected using the ChipWhisperer-Lite platform with a sampling rate of 29.5 MS/s.

For each measurement, 50.000 traces with random plaintexts were recorded. 5 independent measurements with different keys were taken.

Correlated Power Analysis A CPA was performed on the first 4 rounds of Simon-64/128to extract the round keys k_0, k_1, k_2 and k_3. The CPA is based on the Hamming weight model. Two independent attacks based on different intermediate states were realized.

- The first attack is based on the state x_{i+1} which is the result of adding the round key k_i via XOR. Attacking this state makes it easy to split the attack into smaller parts because every bit of k_i only influences 1 bit of x_{i+1}. The disadvantage is that the XOR-operation is symmetric which will result in the same absolute correlation for each key guess and its inverse.
- The second attack is based on the result of the AND-Operation in the subsequent round. The AND-Gate eliminates symmetry which leads to different correlation values for inverted key guesses. One difficulty here is that each bit of the intermediate state is influenced by multiple key bits. Therefore, the position of the guessed bits must be chosen wisely. (See Figure 2 and Figure 3)

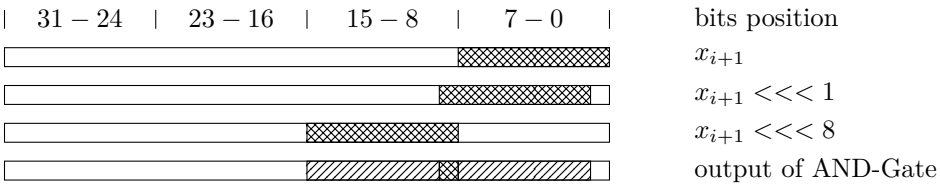


Figure 2: In this example, the 8 right-most bits were guessed. This is a bad choice when attacking the AND-Gate because only 1 bit in the attacked state purely depends on the guessed key bits. The 14 light gray bits depend partially on the guessed key bits but they are also influenced by key bits which are not guessed. This will result in low correlation values.

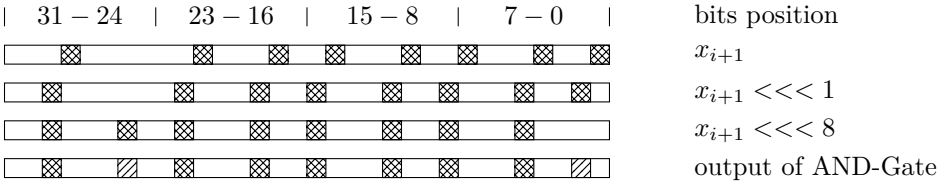


Figure 3: Good choice for guessing 8 bits when attacking the AND-Gate. The guessed bits are distributed with a distance of 7 and therefore mostly line up at the AND-Gate.

Attack steps The CPA attack was executed according to the following procedure:

1. Collect n power traces with m samples each. The whole measurement is a tuple (T, P, C). T is a Matrix of size (n, m) which contains all traces. P contain the plaintexts p_i and C contains the ciphertexts c_i.

$$T = \begin{pmatrix} t_{1,1} & \cdots & t_{1,m} \\ \vdots & & \vdots \\ t_{n,1} & \cdots & t_{n,m} \end{pmatrix} \quad P = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad C = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

2. Create 2^b key candidates K^*. b tells how many bits are guessed.

$$K^* = \left(k_1^* \dots k_{2^b}^*\right)$$

3. For all combinations of plaintexts and key candidates, calculate the attacked intermediate state x. Calculate the hamming weight hw of x but only consider the bits which are influenced by the guessed key bits. The result is the matrix HW.

$$HW = \begin{pmatrix} hw_{1,1} & \cdots & hw_{1,2^b} \\ \vdots & & \vdots \\ t_{n,1} & \cdots & t_{n,2^b} \end{pmatrix}$$

4. Calculate the Pearson Correlation matrix between HW^T and T. The results is a matrix ρ with size $(2^b, m)$. Compute the matrix $\hat{\rho}$ by selecting the the element with the largest absolute value in each row. Compute the value $\tilde{\rho}$ by selecting the largest value of all $|\hat{\rho}_i|$.

$$\rho = \begin{pmatrix} \rho_{1,1} & \cdots & \rho_{1,m} \\ \vdots & & \vdots \\ \rho_{2^b,1} & \cdots & \rho_{2^b,m} \end{pmatrix} \quad \hat{\rho} = \begin{pmatrix} \hat{\rho}_1 \\ \vdots \\ \hat{\rho}_{2^b} \end{pmatrix}$$

5. Based on a threshold τ, find all values $\hat{\rho}_i$ where $|\hat{\rho}_i| + \tau \geq \tilde{\rho}$ The corresponding key candidates k_i^* are marked as promising.

6. For all promising key candidates, create sub candidates and repeat the process from step 3. With this approach, all possible key candidates form a tree where only promising paths are further processed.

7. After attacking all 4 rounds, go through the remaining key candidates and check if encrypting the plaintext p_1 results in the corresponding ciphertext c_1.

Attack Results The power traces clearly show the execution of the SIMON round functions (see Figure 4). This is useful for the attack because the trace can be shortened to the first few rounds. In our experiments only the first 800 samples were included into the attack.

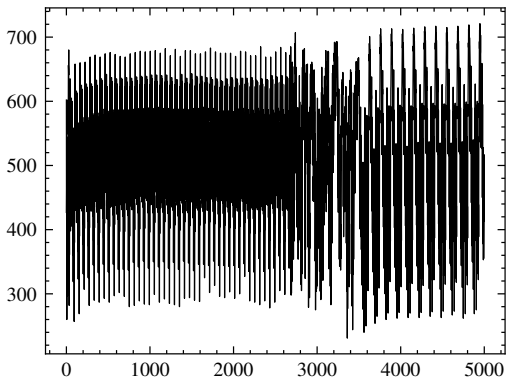
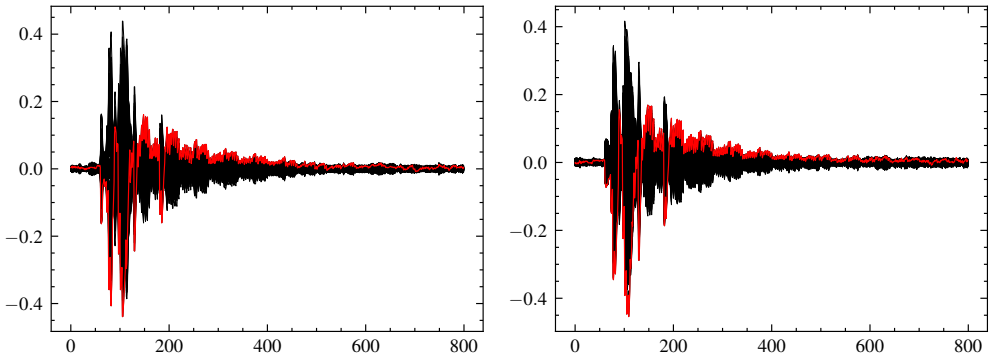


Figure 4: Power consumption graph for the unprotected Simon implementation shows repeating patterns for the 44 rounds from sample 0 to approximately 2600.

Both attacks successfully recovered the full 128-bit key from the unprotected implementation. Figure 5 shows the correlation traces for the correct key candidate and the 255 incorrect ones. For both attacks, the correct candidate has the highest correlation values. In the first attack, the inverse of each key candidate results in a mirrored correlation line.

For the second attack, the graph is not symmetric. In the shown example, the correct key guess has a significantly higher correlation than the inverted one (-0.471 vs. 0.422).

However, this observation was not always true. In some rare cases, the inverted key had higher correlation values than the correct one. The root cause of this phenomenon is not investigated deeper in this work.



(a) Attack 1 (After adding key via XOR) (b) Attack 2 (After AND-Gate)

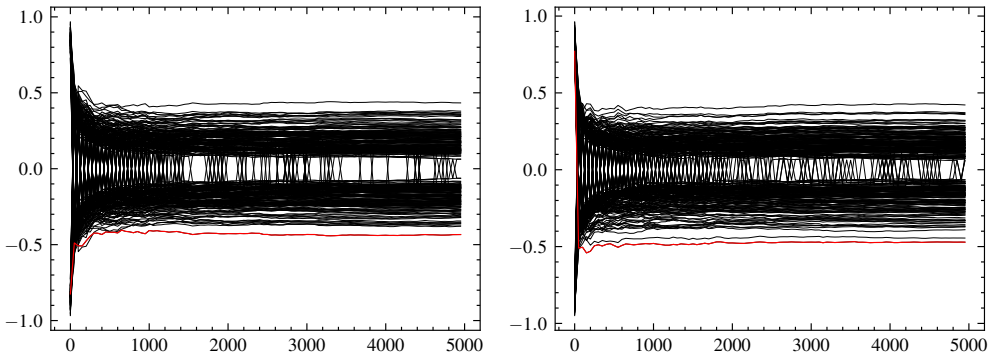
Figure 5: Correlation values over time for 256 key candidates with 50.000 traces.

It was also analyzed how the correlation changes with an increasing number of captured traces. Figure 6 shows the evolution for one measurement campaign and 8 bits being guessed. The experiments have shown that 1000 traces are enough to reliably extract the key.

For attack model 1, the correct key guess was always located within the Top 8 of all key guesses. Because of the symmetry, there are always at least 2 candidates that will be kept for the next attack round. The correlation threshold τ for marking a key candidate as promising was set to 0.02.

For attack model 2, the correct key guess was often found as the top candidate, which is a slight advantage compared to model 1. However, in some cases, the correct key was only found within the Top 10. This might be caused by the fact, that instead of 8 bits, only 7 bits are included in the Hamming weight calculation (see chapter 2.2 and Figure 3 for explanation). The threshold τ was therefore increased to 0.03 because otherwise the correct key might not be found.

The best performance was achieved with $4 \leq b \leq 8$, meaning 4 to 8 key bits are guessed in each attack step. With the chosen parameters, both attacks are equally reliable and have a similar performance. The correct key can be extracted on a standard computer within a few seconds.



(a) Attack 1 (After adding key via XOR) (b) Attack 2 (After AND-Gate)

Figure 6: Maximum correlation values as the number of power traces increases.

3 Masking as a Countermeasure

Boolean masking with two shares was implemented as a countermeasure. 184 random bytes are generated before the encryption using a PRNG based on HMAC (see [BK15]). 8 bytes of randomness are used to mask the plaintext via XOR. 4 additional bytes are required in each of the 44 rounds to perform the AND-operation. The binary rotations and XOR-operations within the round do not require additional masks. The operations can be performed independently on both shares.

Figure 7 shows the updated round function. The blue content is newly added compared to the original round function. The AND-Gate cannot be performed on both shares independently. The Trichina AND-Gate was implemented to avoid any leakage. The inputs of this AND-Gate are the shares of a and b. Additionally, a new mask m_c is brought in.

$$a_m = a \oplus m_a \qquad b_m = b \oplus m_b$$

The AND-Gate must generate the share c_m such that:

$$c = a \cdot b \quad \rightarrow \quad c_m \oplus m_c = (a_m \oplus m_a) \cdot (b_m \oplus m_b)$$

This equation is rearranged to the following form. It can be shown that all intermediate values in this formula are independent from the sensitive values a, b and c. [Tri03], [TKL04]

$$c_m = (((a_m b_m \oplus m_c) \oplus a_m m_b) \oplus b_m m_a) \oplus m_a m_b$$

Avoiding Pitfalls The masked implementation was written in Assembly language (ASM). Potential hamming distance leakage was taken into account during implementation. Literature shows that updating a CPU register will leak the hamming distance between old and new value. Also, performing two arithmetic instructions in a row leaks the hamming distance between input/output operands, even if different CPU registers are used in the two instructions. [SR15], [BCO04]

To avoid leaking confidential information, following precautions were taken:

- Masked values and the masks itself shall be stored in separated registers. In our implementation, even register numbers were used for masked values and odd registers for masks.

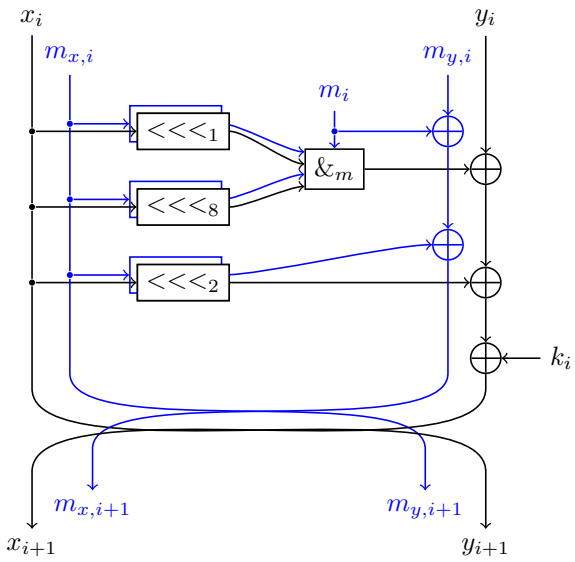


Figure 7: One round of the masked Simon implementation

- Masked values and their corresponding masks shall not be processed in two consecutive instructions. Therefore, dummy instructions were added to clear values within the arithmetic logical unit (ALU).
- Intermediate values which occur, e.g., during the Trichina-AND-Gate, shall be cleared before being reused again.

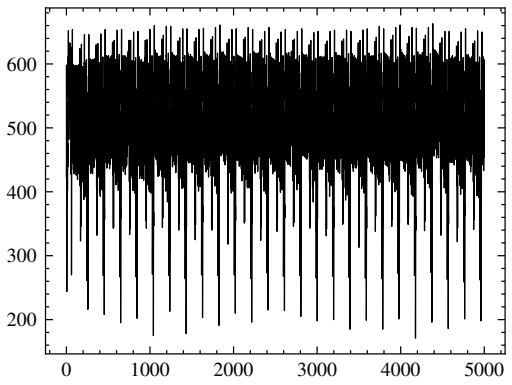
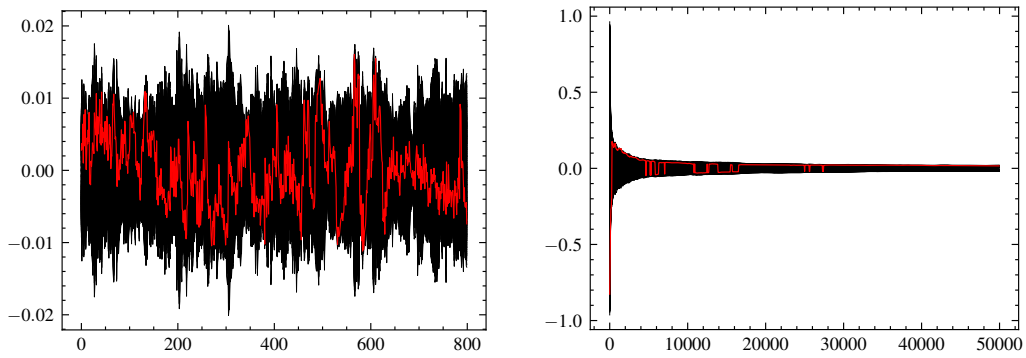


Figure 8: In this graph, 25 of the 44 rounds are visible. Each round takes approximately 4 times as long compared to the unmasked implementation.

Attack on the masked implementation The attacks were repeated on the masked implementation using 50,000 traces. At no point in time, the correct key candidate stands out from the other key candidates. With an increasing number of traces, the maximum correlation goes close to zero. (see Figure 9)



(a) Correlation over time during the first 4 rounds with 50.000 traces.

(b) Maximum correlation values as the number of traces increases.

Figure 9: Attack results on masked Simon implementation.

4 Statistical Leakage Assessment

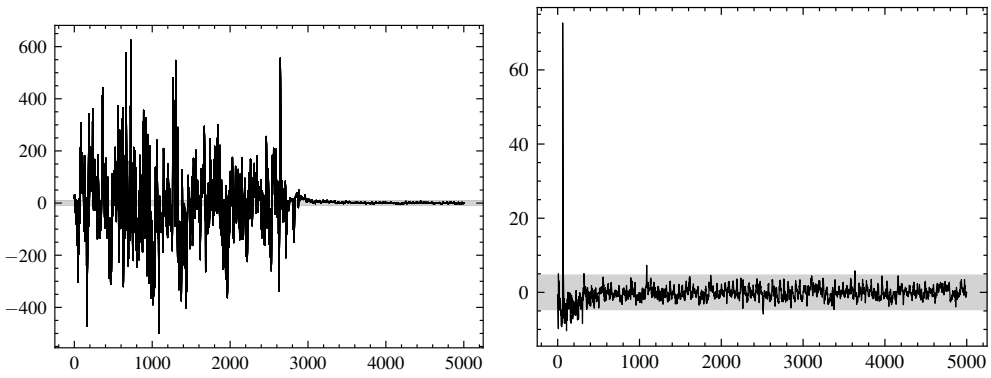
Methodology Welch's t-test (Test Vector Leakage Assessment, TVLA) was employed to detect leakage in both unprotected and masked implementations. Two sets of traces were collected in an alternating way:

- **Set 1:** Fixed key, fixed plaintext
- **Set 2:** Fixed key, random plaintexts

The t-value was computed for each time sample between set 1 and set 2. A threshold of $|t| > 4.5$ indicates that the mean power consumption in set 1 is significantly different from set 2 with 99.999% confidence [GGJR+11]. In a perfect implementation, the mean values of set 1 and 2 should be equal at every point in time. Therefore a high t-value is a indicator that confidential information might be leaked.

Results The t-value over time can be seen in Figure 10.

- **Unprotected Simon:** Strong leakage was detected with t-values exceeding 600 at multiple time points, confirming high vulnerability to CPA attacks.
- **Masked Simon:** The t-values remained below the threshold for most of the samples. However, at the beginning of round 1, the t-values are outside the expected range. Additionally, there are some samples at later rounds where $|t|$ is larger than 4.5. The t-values at the beginning are likely caused by hamming distance leakage at the beginning of round 1. The unmasked plaintext is still present in a CPU register before it is cleared in explicitly in an assembly instruction. This is uncritical for a CPA because no confidential values are being processed at that time. This hypothesis is supported by the fact that later SIMON rounds use the identical code but do not show such a high t-value.



(a) Unprotected Implementation (b) Masked Implementation

Figure 10: T-Value over time in the unprotected and masked Simon implementations

5 Conclusion

This study demonstrates that an unprotected software implementation of SIMON-64/128 is highly vulnerable to Correlation Power Analysis, enabling full key recovery with 1000 power traces. The application of Boolean masking made the original attack fail and drastically reduced leakage. However, the t-values are not completely within the desired threshold of 4.5.

Our results demonstrate a clear trade-off between security and performance in lightweight cryptography implementations. This highlights several important considerations:

- **Implementation quality:** The effectiveness of masking depends heavily on proper implementation, including secure random number generation avoiding hamming distance leakage.
- **Other Attack methods:** While the original attack was not successful on the masked implementations, other approaches like Higher-Order CPA might still be possible.
- **Performance overhead:** The masking implementation increased the encryption time by factor 4. Additional time is required for generating the random numbers. This may be acceptable for many IoT applications but could be prohibitive for extremely resource-constrained devices.

Future work should investigate higher-order masking schemes and their practical applicability to resource-constrained devices, as well as combining masking with alternative countermeasures.

The performance of generating the random numbers was not in the scope of this paper. For real-world use cases, one should investigate if faster random number generators have an impact on the security and if masks can be reused in multiple rounds.

When implementing lightweight cryptography devices, we recommend:

- Evaluate the overall security of the hardware. Securing a device against side-channel attacks might not make sense if the device offers simpler attack vectors (e.g. unlocked JTAG interface)
- Use widespread techniques like T-Test to evaluate leakage. Do not classify a implementation as secure just because the original attack is not working anymore.

- Consider the trade-offs between security, performance, and resource constraints when selecting countermeasures.
- Stay updated on new attack techniques and countermeasures.

A Measurement Parameters

Table 1: Experimental measurement parameters

Parameter	**Value**
Microcontroller	STM32F303 (ARM Cortex-M4)
Clock frequency	7.38 MHz
Sampling rate	29.5 MS/s
Samples per trace	5,000
Traces per Measurement	50,000
TVLA threshold	± 4.5

References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Berlin, Heidelberg, August 2004.

[BK15] Elaine B Barker and John Michael Kelsey. *NIST Special Publication 800-90A Revision 1: Recommendation for random number generation using deterministic random bit generators*. US Department of Commerce, National Institute of Standards and Technology, 2015.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd annual design automation conference*, pages 1–6, 2015.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[SR15] Hermann Seuschek and Stefan Rass. Side-channel leakage models for risc instruction set architectures from empirical data. In *2015 Euromicro Conference on Digital System Design*, pages 423–430. IEEE, 2015.

[TKL04] Elena Trichina, Tymur Korkishko, and Kyung Hee Lee. Small size, low power, side channel-immune aes coprocessor: Design and synthesis results. In *International conference on advanced encryption standard*, pages 113–127. Springer, 2004.

[Tri03] Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. Cryptology ePrint Archive, Report 2003/236, 2003.