Can Machine Learn Pipeline Leakage?

1st Omid Bazangani
Radboud University
Nijmegen, The Netherlands
omid.bazangani@ru.nl

2nd Parisa Amiri Eliasi
Radboud University
Nijmegen, The Netherlands
parisa.amirieliasi@ru.nl

3rd Stjepan Picek
Radboud University
Nijmegen, The Netherlands
stjepan.picek@ru.nl

4th Lejla Batina
Radboud University
Nijmegen, The Netherlands
lejla.batina@ru.nl

***Abstract*—Side-channel attacks cause a significant threat to security implementations in embedded devices. Accordingly, an automated framework simulating side-channel behaviours can offer invaluable insights into leakage origins and characteristics, helping developers improve those devices during the design phase. While there has been a substantial effort towards crafting leakage simulators, earlier methods either necessitated significant manual work for reverse engineering the micro-architectural layer or depended on Deep Learning (DL) models where the neural network's complexity increased considerably with the addition of pipeline stages.**

This paper presents a novel modelling approach using Recurrent Neural Networks (RNNs) to construct instruction-level power models that exhibit enhanced performance in detecting pipeline leakage. Our findings indicate that with memory-based machine learning models, it becomes unnecessary to input data accounting for the pipeline effect. This strategy reduces feature dimensionality by at least one-third for a three-stage pipeline, albeit at a modest compromise in model performance. This reduced feature set underscores our model's scalability, making it a preferred choice for analyzing microprocessors with extended pipeline stages. Importantly, our methodology accelerates the micro-architectural profiling phase in side-channel simulator design.

When evaluated on an expansive dataset, the performance of our memory-based model closely matches that of the Multilayer Perceptron (MLP) with an R^2 value of 0.79. On a reduced dataset (removing the pipeline effect), our model achieves an R^2 value of 0.65, outperforming the MLP, which reaches an R^2 value of 0.39. Moreover, our model is designed with scalability in mind, making it suitable for profiling microcontrollers with advanced pipeline stages.

For the practical realisation of our approach, we employed the open-source `ABBY-CM0` dataset from the ARM Cortex-M0 microcontroller, which has three pipeline stages. To provide a detailed analysis, we also consider a Convolutional Neural Network (CNN) besides two RNN architectures (Long Short-Term Memory and Gated Recurrent Unit).

***Index Terms*—Side-channel attack, Micro-architectural leakage, Leakage simulators, Deep neural networks**

I. Introduction

The theoretical security of cryptographic primitives deployed within embedded systems is well-established. Nevertheless, when transitioning from theory to physical realisation, these algorithms become vulnerable to a powerful class of attacks known as a Side-Channel Attack (SCA) [1], [2]. Consequently, it becomes the responsibility of the hardware designer to safeguard the system against SCAs. To achieve this robustness, the designer must tailor the algorithm for the specific platform and thoroughly evaluate it. This involves conducting various side-channel analyses and ensuring no sensitive data leakage occurs.

The complexity of the system and the nature of the side-channel attack can make the analysis challenging. Not only does the attacker's proficiency come into play, but the quantity and integrity of the acquired measurements also substantially influence the efficacy of the attack or analysis. Repeatedly procuring measurements for each new implementation to ensure its resilience against side-channel attacks can be labour-intensive, potentially inhibiting the product's swift market entry. The burgeoning Internet of Things (IoT) landscape, characterised by numerous start-ups operating on limited budgets, highlights the imperative for automatic tools that streamline the analytical process.

A framework for side-channel leakage profiling, adept at characterising the micro-architectural layer, holds the potential to trim production costs significantly. This framework transforms high-level code into traces nearly identical to those extracted from the real device. Consequently, designers can modify and evaluate implementations based on the data provided by the profiling framework. Throughout the developmental phase, the framework also affords designers the insight to distinguish if a particular modification enhances the system without compromising its security and without substantial costs. Beyond merely detecting side-channel vulnerabilities, the leakage profiling framework offers insights into the origins of such leaks.

The significant efforts aimed at developing tools that emulate side-channel leakage by simulating the real-time power consumption of devices underscore their importance [3]. SILK, recognized as the pioneering open-source side-channel leakage simulator, generates power traces using both a cryptographic algorithm's source code and user-specified parameters, such as the leakage function and the number of leakage points [4]. Given its inability to represent a distinct hardware architecture, it remains best suited for the preliminary stages of cryptographic algorithm design.

Instruction-level simulators (ISA) estimate side-channel leakage by monitoring the machine code tailored for a designated architecture, preventing the need for detailed processor or process technology specifications. One such ISA simulator, SAVRASCA, leverages compiled binary code combined with the tracing capabilities of the SimulAVR tool to produce simulated power traces for the AtmelAVR family [5]. This simulator produces a power sample for each executed instruction, disregarding the memory unit's address. On another front, ASCOLD [6] identifies violations of the independent leakage

979-8-3503-4859-0/DATE24/© 2024 EDAA

assumption (ILA) [7] for the AVR architecture. It operates by using the assembly file of the masked implementation and a system configuration file, pinpointing the specific line number and the rule violated by the program. However, the underlying reasons for ILA breaches remain device-specific, posing difficulties in generalizing them, especially when compared to the instruction-set description of the target.

A subset of simulators emphasizes the micro-architectural effects of the target. Specifically, MAPS, designed for ARM Cortex-M3, is notable for synthesizing power traces from the source code of masked implementations [8]. It utilizes an HDL file of the target architecture, focusing on pipeline leakage effects. Nevertheless, obtaining the corresponding HDL files often proves challenging.

ELMO is a fine-grained leakage simulator for the ARM Cortex-M0/M4 series, modelling power using a linear combination of values and transitions [9]. Notably, it does not demand deep knowledge of the target processor or its hardware description. ELMO* enhances this by addressing multi-cycle interactions, improving the ELMO leakage model [10].

The code rewrite engine called ROSITA introduced in [10], serves as an autonomous code-patching engine, detecting and fixing leakage. Starting with a masked cryptographic algorithm, it produces assembly and binary outputs. Leveraging the ELMO* framework, ROSITA identifies and substitutes leaky instructions with non-leaky counterparts in an iterative manner until no leakage remains.

ABBY [11] is a framework for side-channel leakage profiling that automates the creation of transition-based leakage models, obviating the need for reverse engineering micro-architectural implementations. Central to ABBY is an MLP, which considers salient micro-architectural features, including instruction interactions, pipeline influences, operand values, and memory dynamics. While refining the ELMO model, the power consumption of Cortex-M0 microprocessors was found to be affected by not only individual instructions but also their surrounding instructions. ABBY captures this intricate pipeline and memory interaction. As highlighted in [11], considering a 3-stage pipeline with 56 Cortex-M0 specific Thumb-instructions results in a feature space of 56^3. The MLP's complexity in ABBY directly correlates with the pipeline stages. As stages increase, MLP complexity rises, making pipeline effect modelling progressively more challenging.

In this study, we explore the potential of memory-based machines in understanding pipeline leakage by capitalizing on their memory cells. We confirm this to be possible and subsequently discuss the associated performance cost.

By harnessing the capabilities of memory-based models, the feature dimensionality for microarchitectural profiling remains constant and proportional to the number of pipeline stages. This reduction in feature dimensionality accelerates the profiling process for microcontrollers. A reduced feature set can lead to a decrease in the required training samples and concurrently decrease the risk of model overfitting. Furthermore, the feature reduction process can offer a more profound comprehension of the pipeline architecture, proving invaluable for reverse-engineering endeavours related to the pipeline structure. Additionally, utilizing memory-based models for feature reduction supports the scalability of this approach, particularly when profiling microcontrollers with an extensive pipeline.

Inspired by ABBY, this work introduces a novel profiling model to determine micro-architectural leakage effects. Unlike the traditional MLP employed in ABBY, our approach leverages an RNN. For a microprocessor with a 3-stage pipeline and only 49 features, our model's performance on the `ABBY-CM0` dataset is comparable to ABBY's MLP, requiring 391 features. This streamlined feature set highlights the scalability of our model, positioning it as an optimal selection for examining microprocessors with advanced pipeline stages.

The structure of this paper is as follows. Section 2 lays the groundwork by introducing a metric for contrasting our models against existing ones. A brief overview of the defining characteristics of the ARM Cortex architecture is provided, followed by a concise introduction to RNNs. Section 3 investigates the contemporary state-of-the-art methods for pipeline leakage modelling. Our proposed model is introduced in Section 4. The outcomes and insights from our research are elaborated upon in Section 5. Finally, we draw conclusions in Section 6, alongside suggesting future directions.

II. Background

A. Side-channel Analysis

Side-channel analysis relies on unintended emissions or data leakage from electronic devices. Modern electronics predominantly employ semiconductor logic gates constructed from transistors. Charging or discharging the gates of these transistors not only results in power consumption but also creates electromagnetic radiation [1]. Studies reveal that such power consumption or electromagnetic emissions correlate with the data under process and the instructions in execution. This observation has catalysed the development of numerous statistical techniques focused on extracting sensitive data through analyses of power consumption, electromagnetic radiation, and other side channels. These channels include timing [12], shared micro-architectural components [13], [14], as well as more unusual sources like acoustic and photonic emanations [15]–[17].

B. Evaluation of Models

Machine learning techniques can be utilized for regression or classification. While side-channel analysis with DL typically pertains to classification tasks, leakage simulators employ predictive models for power consumption estimation as a regression task. We aim to forecast power based on the binary code from software implementations. For evaluating our regression models, we adopt the *coefficient of determination* (R^2) and *cross-validation*, which are established metrics for this purpose [18].

The coefficient of determination, R^2, quantifies the fraction of variance in the dependent variable explained by the independent variable(s). An R^2 close to 1 signifies a strong fit between predicted and observed values. It is determined using

the residual sum of squares, which captures the unexplained variance by the model derived from the squared discrepancies between the actual y_i and the predicted values $\hat{y}_i$,

$$\texttt{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1}$$

where n is the number of samples. The explained sum of squares measures the variation of the explanatory variables and is calculated as

$$\texttt{ESS} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2. \tag{2}$$

The total sum of squares is the sum of the above terms:

$$\texttt{TSS} = \texttt{RSS} + \texttt{ESS} = \sum_{i=1}^{n}(y_i - \bar{y})^2. \tag{3}$$

Thus, the coefficient of determination R^2 can be calculated by

$$R^2 = \frac{\texttt{ESS}}{\texttt{TSS}} = 1 - \frac{\texttt{RSS}}{\texttt{TSS}}. \tag{4}$$

As a model's number of explanatory variables increases, the R^2 value tends to rise. To counteract potential overfitting and penalize extra variables, we use the adjusted R^2, denoted as R^2_{adj}.

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{(n - p - 1)}. \tag{5}$$

If there are n samples and p factors (or explanatory variables) to consider, and if the number of samples n is much larger than the number of factors p, then R^2 closely approximates R^2_{adj}.

C. Recurent Neural Networks

RNNs are well-known for modelling sequential data due to their capacity to handle variable-length sequences, distinguishing them from feed-forward networks [19]. Consequently, they excel in time-series analysis [20] and have found applications in domains like natural language processing [21], [22], speech recognition [20], [23], image captioning [24], and video analysis [25].

RNNs can suffer from the vanishing gradient problem, where gradients diminish during backpropagation across multiple time steps. Various solutions, including gradient clipping [26], non-saturating activation functions [27], alterations in gradient propagation paths [28], and architectures like Long Short-Term Memory (LSTM) [29] and Gated Recurrent Unit (GRU) [30], have been proposed to counteract this challenge. As such, this study considers LSTM and GRU architectures for pipeline profiling. LSTM and GRU architectures selectively manage previous inputs using gated mechanisms. LSTM employs three gates and a memory cell, whereas GRU uses two gates for this purpose.

III. Classic Approaches for Pipeline Profiling

Initial research highlighted that the power consumption of a specific instruction is influenced by preceding instructions [31]. For a three-stage pipeline, the power of the current instruction, I_c, is dependent on the previous, I_p, and subsequent instruction, I_s. This insight forms the basis for the ELMO simulator and the ABBY framework.

The quantity of pipeline stages dictates the number of preceding and subsequent instructions that must be considered during the feature generation process, which aims to model the pipeline effect before inputting these features into the deep learning framework.

A. Instruction Coverage

ELMO seeks variance in the leakage behaviour of instructions, considering their context in code sequences. Given the vast set of Thumb instructions and neighbouring code sequence effects, complete profiling becomes challenging. ELMO targets specific Thumb instructions prevalent in symmetric key cryptographic algorithms, applicable to both Cortex-M0 and M4 processors, as depicted in Table I. By categorising these instructions into five consumption behaviour groups, ELMO streamlines pipeline profiling and gathers 1k traces for each instruction triplet combination (5^3). This strategy substantially reduces the number of measurements, albeit at the expense of some profiling accuracy.

The feature reduction embedded within our methodology facilitates the coverage of a more extensive instruction set without significantly complicating the process. This is achievable, given that our model does not require the triplet combination of instructions.

Type	Mnemonic
Memory access instructions	`ldr, ldrb, ldrh, str, strb, strh`
General data processing instructions	`lsls, lsrs, rors, muls eors, and, adds, adds #imm, subs, subs #imm, orrs, cmp, cmp #imm, movs, movs #imm`

TABLE I
ELMO'S PROFILING THUMB INSTRUCTIONS

ABBY extended the Thumb instructions set for pipeline effect profiling on the ARMV6-M architecture, encompassing 44 out of 56 Thumb instructions, including all 21 instructions profiled by ELMO. The complete list is detailed in Table II. Additionally, ABBY replaces omitted instructions in the code sequence with the `NOP` (No Operation) instruction, resulting in a total of 45 profiled instructions.

ABBY utilizes "one-hot encoding" for the binary representation of mnemonic assembly instructions. For the ARMv6-M's three-stage pipeline, ABBY gives a $45 \times 3 = 135$-dimension feature vector. However, this method inflates feature dimensionality and model complexity, especially for architectures with more pipeline stages. Achieving high accuracy also necessitates numerous measurements. Therefore, current profiling solutions become impractical for complex architectures like the Cortex-M7's six stages or Cortex-M85's scalar seven and up to ten

Type	Mnemonic
Memory access instructions	`ldr, ldrb, ldrh, str, strb, strh`
General data processing instructions	`lsls, lsls #imm, lsrs, lsrs #imm, eors, ands, add, adds, adds #imm, rors, muls, sub, subs, subs #imm, orrs, cmp, cmps, cmp #imm, cmn, mov, movs, movs #imm, rev, rev16, revsh, adcs, asrs, asrs #imm, cpy, bics, negs, mvns, sbcs, sxtb, sxth, tst, uxtb, uxth`
Other	`nop`

TABLE II
ABBY'S PROFILING THUMB INSTRUCTIONS

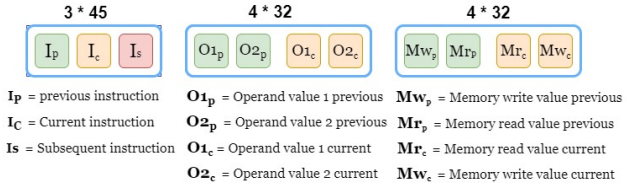

Fig. 1. The ABBY dataset comprises a total of 391 distinct features.

for vector or floating point instructions. Consequently, this paper presents RNN-based models to curtail complexity while retaining accuracy.

B. Operand Value

The ELMO study reveals that operand values influence an instruction's power consumption [9]. Both ELMO and ABBY, visualized in Figure 1, incorporate operand values of current (I_C) and previous (I_p) instructions. ABBY represents these as a 32-bit binary, expanding the operand value feature dimension to $4 \times 32 = 128$ (Figure 1).

C. Memory Transaction

Instructions for memory transactions (`LDR` and `STR`) may lead to side-channel leakage during memory read/write operations [32], [33]. Both ABBY and ELMO account for values in current and previous memory stages, addressing potential memory bus leakage. This incorporation adds four 32-bit binary features, extending the memory-related feature dimension to $4 \times 32 = 128$ (Figure 1).

IV. TRADITIONAL LEAKAGE MODEL CONSTRUCTION

In the present study, we re-examine the ABBY model to both reconstruct the ABBY model and evaluate the performance of alternative models. Utilizing the same MLP architecture employed in the ABBY model, we achieved an R^2 value consistent with that reported in the original ABBY publication. Beyond the MLP, we conducted assessments of LSTM and GRU networks on an identical, non-reduced feature set. Furthermore, our study recognises the potential of CNNs to identify translation-invariant patterns within time series data and leverage the parameter efficiency derived from convolutional weight sharing. Thus, we also incorporated a CNN into our evaluation. Notably, utilising varied kernel sizes in CNNs facilitated the detection of patterns across multiple resolutions. A comprehensive summary of our results can be found in Table III.

TABLE III
MODEL PERFORMANCE FOR ABBY-CM0 DATASET WITH 391 FEATURE

	Mean Square Error	R^2_{adj}
MLP	$7.963E-6$	0.78
LSTM	$9.543E-6$	0.75
GRU	$8.908E-6$	0.76
CNN	$8.564E-6$	0.77

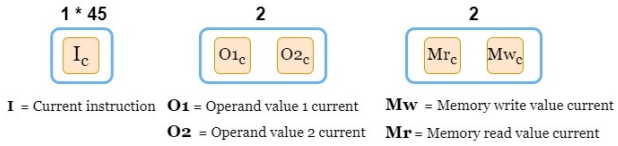

Fig. 2. RNNs dataset features

V. NEW APPROACH FOR PIPELINE PROFILING

In our methodology, we operate under the assumption that a memory-based deep learning approach can encapsulate the effects of the pipeline. As such, our focus is narrowed exclusively to simultaneously executing micro-architectural features. We have opted for DL networks, particularly RNNs, to tackle the inherent complexities. This mitigates complexities and provides a robust foundation for profiling increasingly complex pipeline architectures.

By refining our feature dimensionality, we now emphasize the instruction and operand values of the current execution in conjunction with the memory bus value for it. This refinement results in a streamlined set of 49 distinct features, a significant reduction from the initial 391 (Figure 1. Such an approach greatly simplifies micro-architectural profiling, especially compared to the methods outlined in the classical approach. The input features of our methodology are depicted in Figure 2.

A. Instruction Coverage

Utilizing RNNs allows for the integration of both preceding and subsequent instruction effects without the necessity of directly expanding the feature dimensionality. The memory cells in the network address the details of pipeline stages and their related power consumption effects, resulting in a concise set of 49 features. Notably, 45 of these features pertain to the instructions for pipeline profiling, a significant reduction from the initial 135.

B. Operand Values and Memory Transaction

The operand values and memory transactions need explicit representation in the proposed methodology. In our analysis of the ABBY-CM0 dataset and considering the infrequency of large values in cryptographic algorithms, we chose to normalize operand values within the interval $[0, 1]$ instead of transposing them into binary representations. Consequently, this introduces a mere four dimensions to our feature set, in contrast to

the potential 128. Such an optimization curtails the feature dimensionality by a factor of ≈ 8 and substantially reduces the computational burden on the learning model. This, in turn, enhances the profiling's adaptability to microcontrollers equipped with advanced pipeline stages.

VI. New Approach for Leakage Model Construction

In this section, we conduct a performance assessment of various models on our refined dataset from our updated approach. The goal is to measure the proficiency of memory-based DL techniques in capturing the pipeline effects of the microcontroller. For a baseline comparison, we employ an adapted version of the MLP model akin to what ABBY utilized. This is followed by evaluations using CNN, LSTM, and GRU models. Moreover, we investigate a composite model, integrating features from all aforementioned models due to its observed incremental performance benefits.

MLP. To facilitate a comparative analysis, we evaluate the performance of the MLP model previously employed by ABBY. This model is trained on the dataset, which has been truncated in line with our novel approach. The results indicate a significant decline in the R^2 metric, registering a value of 0.39. Concurrently, as anticipated, the Mean Square Error (MSE) exhibited an increase. This can be attributed to the absence of memory mechanisms within such a network, rendering it incapable of accounting for the pipeline effect.

CNN. We employ a CNN model, motivated by its intrinsic ability to distinguish translation-invariant patterns within time series data and its efficient parameter usage deriving from convolutional weight sharing. We selected a kernel size for the model corresponding to the number of pipeline stages, aiming to capture the pipeline effect. The CNN model achieved an R^2 value of 0.64, underscoring its proficiency in learning the pipeline effect from the dataset.

LSTM, GRU. The network is intrinsically balanced by harnessing the LSTM/GRU architecture to remember previous processing features and influence subsequent operations. Therefore, selecting a window size commensurate with the number of pipeline stages is reasonable. In our configurations, this translates to a window length of three. The R^2 values for the LSTM and GRU models are closely aligned, registering at 0.63 and 0.64, respectively.

Combined Model. Incorporating the strengths of CNN, LSTM, and GRU provides a robust framework for analyzing hybrid data with spatial and temporal characteristics. This integrated approach facilitates the extraction of intricate spatial and temporal patterns and presents significant architectural flexibility. Furthermore, this model has the potential to enhance performance metrics and is particularly adept at addressing challenges inherent in multi-modal tasks.

Given that the micro-architectural implementation of the pipeline remains proprietary and is protected as the intellectual property of the company, we lack comprehensive access to the specifics of this implementation. Moreover, understanding the influence of various micro-architectural attributes on the chip's power consumption remains inaccessible. To possibly reveal more about the micro-architectural implementation, we adopted a hybrid architectural model and analyzed its performance.

Our implementation of this hybrid architecture is showcased in Table IV. While the coefficient of determination, R^2, substantially aligns with the foundational elements of this model, further exploration represents an interesting research direction.

Layer(Type)	Output Shape	# Parameters
conv1d (Conv1D)	(None, 130, 128)	1280
conv1d (Conv1D)	(None, 130, 128)	49280
max_pooling1d (MaxPooling1D)	(None, 65, 128)	0
conv1d (Conv1D)	(None, 65, 32)	12320
max_pooling1d (MaxPooling1D)	(None, 32, 32)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 16)	0
dense (Dense)	(None, 1)	0

TABLE IV
Architecture of the combined model

Table V represents the result based on the R^2 and MSE values.

VII. Results and Discussion

Based on the data presented in Table V, the MLP model, which lacks memory integration, exhibits an inferior R^2 value and a greater Mean Squared Error (MSE) compared to models incorporating memory. Given its inherent characteristics, the CNN aligns more closely with the memory-based models in terms of performance, as reflected by the R^2 values. The marked performance improvement observed in memory-based models (LSTM and GRU) over non-memory models such as MLP (evidenced by the increase in R^2 from 0.39 to 0.64) underscores the aptitude of memory-based models to encapsulate the effects of the pipeline. Notably, the combined model demonstrates a marginally superior performance.

When comparing the novel approach with traditional models, it is evident that the former utilizes a more compact dataset, encompassing only 49 features, as opposed to the latter's 391. Additionally, the parameter count of the new methodology is markedly reduced, standing at 6k in contrast to the traditional model's substantial 311k. These disparities render the new approach more suitable for architectures with increased pipeline stages, albeit at a moderately compromised performance. Specifically, the performance diminishes from an R^2 = 0.78 in the conventional model to an R^2 = 0.65 in the novel approach.

TABLE V
Model performance result for the new approach on reduced dataset

	Mean Square Error	R^2_{adj}
MLP	$2.098E-5$	0.39
LSTM	$1.347E-5$	0.63
GRU	$1.313E-5$	0.64
CNN	$1.321E-5$	0.64
Combined	$1.302E-5$	0.65

VIII. Conclusions and Future Work

We have introduced a new methodology for micro-architectural profiling of pipelines utilizing memory-based machine learning models. This novel strategy offers a marked reduction in complexity compared to its predecessor, the ABBY profiling framework. Our exposition explains how our approach reduces feature dimensionality across various micro-architectural characteristics, including instruction sets, operand values, and memory interactions. Our proposed model is scalable and versatile enough to cater to architectures with an augmented number of pipeline stages. In contrast, existing methodologies tend to escalate in complexity, eventually verging on impracticality.

This research explains the potential of memory-augmented DL networks in addressing the complexities inherent to side-channel simulator design. Furthermore, a thorough exploration into the mechanisms by which the model profiles pipeline leakage presents a promising avenue for future endeavours in reverse engineering of the targeted system. Our future work aims to improve performance by integrating more sophisticated artificial intelligence methodologies, such as attention-driven or transformer-based techniques.

IX. Acknowledgements

Lejla Batina and Stjepan Picek are partially funded by PROACT (NWA.1215.18.014), a project financed by the Netherlands Organisation for Scientific Research (NWO).

References

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*. Springer, 1999, pp. 388–397.

[2] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*. Springer, 2004, pp. 16–29.

[3] I. Buhan, L. Batina, Y. Yarom, and P. Schaumont, "SoK: Design tools for side-channel-aware implementations," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 756–770.

[4] N. Veshchikov, "SILK: high level of abstraction leakage simulator for side channel analysis," in *PPREW@ACSAC*, 2014, pp. 3:1–3:11.

[5] N. Veshchikov and S. Guilley, "Use of simulators for side-channel analysis," in *EuroS&P Workshops*, 2017, pp. 104–112.

[6] K. Papagiannopoulos and N. Veshchikov, "Mind the gap: Towards secure 1st-order masking in software," in *COSADE*, ser. Lecture Notes in Computer Science, vol. 10348. Springer, 2017, pp. 282–297.

[7] M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30*. Springer, 2011, pp. 109–128.

[8] Y. Le Corre, J. Großschädl, and D. Dinu, "Micro-architectural power simulator for leakage assessment of cryptographic software on ARM Cortex-M3 processors," in *COSADE*, 2018, pp. 82–98.

[9] D. McCann, E. Oswald, and C. Whitnall, "Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages," in *USENIX Security Symposium*, 2017, pp. 199–216.

[10] M. A. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, and Y. Yarom, "ROSITA: Towards automatic elimination of power-analysis leakage in ciphers," in *NDSS*, 2021.

[11] O. Bazangani, A. Iooss, I. Buhan, and L. Batina, "ABBY: Automating leakage modelling for side-channel analysis," in *Proceedings of the 2024 ACM Asia Conference on Computer and Communications Security*, 2024.

[12] D. Page, "Theoretical use of cache memory as a cryptanalytic side-channel," *IACR Cryptol. ePrint Arch.*, p. 169, 2002. [Online]. Available: http://eprint.iacr.org/2002/169

[13] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *Journal of Cryptographic Engineering*, vol. 8, pp. 1–27, 2018.

[14] E. Tromer, D. A. Osvik, and A. Shamir, "Efficient cache attacks on AES, and countermeasures," *Journal of Cryptology*, vol. 23, pp. 37–71, 2010.

[15] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, C. Sporleder *et al.*, "Acoustic side-channel attacks on printers." in *USENIX Security symposium*, vol. 9, 2010, pp. 307–322.

[16] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, CA, USA, August 17-21, 2014, Proceedings, Part I 34*. Springer, 2014, pp. 444–461.

[17] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES: photonic side channel analysis for the rest of us," in *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*. Springer, 2012, pp. 41–57.

[18] S. Gao and E. Oswald, "A novel completeness test for leakage models and its application to side channel attacks and responsibly engineered simulators," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022, pp. 254–283.

[19] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The MIT press, 2016, 800 pp," *Genetic programming and evolvable machines*, vol. 19, no. 1-2, pp. 305–307, 2018.

[20] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "Speech recognition with deep Recurrent Neural Networks," *IEEE Signal Processing Magazine*, vol. 29, pp. 141–152, 2013.

[21] T. Mikolov, M. Karafi'at, L. Burget, J. Cernock'y, and S. Khudanpur, "Recurrent Neural Network based language model," *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 1045–1048, 2010.

[22] D. Tang, B. Qin, T. Liu, and J. Yang, "Document modeling with gated Recurrent Neural Network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1422–1432.

[23] R. Srivastava, V. Patel, J. Huang, and B. Raj, "Speaker recognition using Recurrent Neural Network based time-frequency representations," in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1695–1699.

[24] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 2048–2057, 2015.

[25] Y. Fan, X. Lu, D. Li, and Y. Liu, "Video-based emotion recognition using CNN-RNN and C3D hybrid networks," in *Proceedings of the 18th ACM international conference on multimodal interaction*, 2016, pp. 445–450.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," in *International conference on machine learning*. Pmlr, 2013, pp. 1310–1318.

[27] S. Chandar, C. Sankar, E. Vorontsov, S. E. Kahou, and Y. Bengio, "Towards non-saturating recurrent units for modelling long-term dependencies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3280–3287.

[28] D. Arpit, B. Kanuparthi, G. Kerg, N. R. Ke, I. Mitliagkas, and Y. Bengio, "h-detach: Modifying the LSTM gradient towards better optimization," *arXiv preprint arXiv:1810.03023*, 2018.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[31] V. Tiwari, S. Malik, A. Wolfe, and M. T. Lee, "Instruction level power analysis and optimization of software," in *VLSI Design*, 1996, pp. 326–328.

[32] G. Yee, Y. Zheng, M. Franz, and G. Portokalidis, "Leveraging arm trustzone for reducing memory-based side channels," in *ACM Conference on Computer and Communications Security (CCS)*, 2014, pp. 20–31.

[33] S. Bhasin, M. Toussaint, and M. Hutter, "A systematic analysis of memory side-channel leakage on arm-based systems," in *Workshop on Cryptography and Security in Computing Systems (CS2)*, 2012, pp. 15–20.