First-Order and Higher-Order Power Analysis: Computational Approaches and Aspects

Petr Socha, Vojtěch Miškovský, Martin Novotný
Czech Technical University in Prague
Faculty of Information Technology
{sochapet,miskovoj,novotnym}@fit.cvut.cz

***Abstract*—Side-channel analysis pose a serious threat to many modern cryptosystems. Using Correlation power analysis, attacker may be able to recover the cipher key and therefore jeopardize the whole cryptosystem, which is why many countermeasures are being developed. These countermeasures are typically effective against first-order attacks. However, protected implementations may still be vulnerable to higher-order analysis. In this paper, we compare different approaches to the higher-order analysis regarding their mathematical and performance properties. We focus on Correlation power analysis attack and the test vector leakage assesment using Welch's t-test, we optimize and accelerate discussed algorithms using CPU and GPU, and we present our experimental results and remarks.**

***Index Terms*—Side-channel attack, Higher-order analysis, Test vector leakage assesment, Differential power analysis, Correlation power analysis, Pearson correlation coefficient, Welch's t-test**

I. Introduction

In the past few decades, computers and communication networks have become an essential part of our everyday lifes. Our society depends on things such as credit/debit cards, smartphones, or security RFID keychains. With the rise of the IoT era, many other smart (and internet-connected) devices emerge, including personal assistants, smart cars, cities and even medical devices such as peacemakers.

Given this environment, our privacy is more endangered than it ever was and security of these devices becomes a major issue. In the case of IoT devices, this issue gets even more significant given the fact, that an attacker may often easily gain a physical access to the device. Even when appropriate authentification, authorization and encryption mechanisms are employed, side-channel attacks still may be succesfully mounted. These attacks exploit the implementation-dependent information leakage rather than the formal properties of the cipher itself, e.g. by examining power consumption of the device.

Differential Power Analysis (DPA) [1] and Correlation Power Analysis (CPA) [2], [3] are common examples of side-channel attacks applicable even to ciphers considered mathematically secure, such as the Advanced Encryption Standard (AES) [4], PRESENT [5] or SERPENT [6]. These attacks reveal the cipher key and thus jeopardize the whole cryptosystem. Various countermeasures were proposed in order to prevent this kind of side-channel attacks, including hiding using dual-rail logic [7], masking [8] or threshold cipher implementations [9]. While these countermeasures are effective against first-order analysis, the implementations often remain vulnerable against higher-order attacks.

In this paper we present different approaches to the univariate first-order and higher-order analysis. We focus on the Correlation Power Analysis attack and the Test Vector Leakage Assesment using non-specific Welch's t-test [10], and we examine the computational and memory demands of the approaches presented earlier using our open-source CPU and GPU accelerated implementations.

II. First-Order Analysis

In the context of side-channel security, first-order analysis examines the raw measured data. As mentioned earlier, we will focus on the Correlation Power Analysis [3] attack, which examines the relationship between a power consumption of the device and a hypothetical leakage. This attack is briefly explained in Subsection II-A. Various methodologies exist in order to quantify information leakage of the device; we will focus on the non-specific Welch's t-test [10], which is briefly explained in the Subsection II-B. Different computational approaches and their characteristics are then presented in Subsections II-D, II-E and II-F.

A. Side-Channel Attack: Correlation Power Analysis

The first, active phase of the Correlation Power Analysis [3] attack consists of measuring the power consumption during the encryption and capturing the plaintext or the ciphertext. This process results in capturing N power traces, each consisting of S power samples.

The second, analytical phase consists of two parts. First, different power consumption predictions are created using a specific leakage model. These predictions are based on a small part of the cipher key (e.g. a byte or a nibble). In case of AES128 [4], where CPA aims at a byte of the key at a time, this results in 256 key candidates K, i.e. 256 predictions for each power trace measured ($N \times K$ in total). Second, the correlation between the measured power traces and the power predictions is computed, resulting in a matrix of correlation coefficients of dimensions $S \times K$. The rows of this matrix represent the correlation coefficient progression in time, i.e. correlation traces, for each key candidate. The correct key candidate is selected e.g. visually by the attacker or algorithmically using coefficient magnitude or by other characteristics [11]. The analytical phase of the attack is

repeated 16 times in order to obtain the whole 128-bit long cipher key.

To examine the correlation, Pearson correlation coefficient gets estimated, which is defined as follows:

$$\rho_{X,Y} = \frac{\operatorname{cov}[X,Y]}{\sqrt{\operatorname{Var}[X]\operatorname{Var}[Y]}}, \tag{1}$$

with $\operatorname{cov}[X,Y]$ being the covariance, and $\operatorname{Var}[X], \operatorname{Var}[Y]$ being variances of variables X, Y respectively.

B. Side-Channel Leakage Assesment: Welch's t-test

In order to quantify the first-order leakage of the device, non-specific Welch's t-test can be used as described in [10], [12]. First, two sets of power traces are measured, using either random plaintext data or preselected constant plaintext data. Traces from both sets must be measured together in a randomly interleaved fashion: for every power trace measurement, the device is fed randomly (with the same probability) either random or preselected constant plaintext.

Then the Welch's t statistic is computed (for each sampling point), which is defined as follows:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{2}$$

with μ_1, μ_2 being sample means, s_1^2, s_2^2 being sample variances and N_1, N_2 being cardinalities of both measured power sets, respectively. In order to compute the associated degrees of freedom, following formula can be used:

$$d.o.f. = \frac{(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2})^2}{\frac{(\frac{s_1^2}{N_1})^2}{N_1 - 1} + \frac{(\frac{s_2^2}{N_2})^2}{N_2 - 1}}. \tag{3}$$

The Welch's t-test statistic examines the null hypothesis, that samples in both sets were drawn from the same population. In other terms, it quantifies the leakage of the device by "measuring the distinguishability" of random and constant data encryption. The t statistic and $d.o.f.$ are computed for every sampling point.

C. Computational Aspects

In order to perform the Correlation Power Analysis attack, we need to estimate the variance of power traces at each sampling point (i.e. S variances based on N measurements), as well as variance of our power predictions for each key candidate (i.e. K variances based on N predictions). Additionally, a covariance matrix $S \times K$ must be computed, based on all N measurements and N predictions.

Welch's t-test requires variances to be estimated too, for both traces sets (N_1 constant data traces and N_2 random data traces) and for each of the S sampling points. Besides that, the sample means need to be estimated as well.

Since this computation is performed independently at each sampling point, there is a clear potential for data parallelism. Following subsections briefly explain various approaches to the mean, variance and covariance computation.

D. Multiple-Pass Algorithm

Well established approach to these statistical tasks is a multiple-pass algorithm. In the first pass, the sample mean $\mathrm{E}[X]$ is estimated for every sampling point. Then, in the second pass, the variance is straightforwardly computed directly by recalling its definition: $\operatorname{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2]$, and finally, so is the covariance: $\operatorname{cov}[X,Y] = \mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]$.

The primary drawback of this approach is the inability to add new data to the statistical set (i.e. add more power trace measurements) without running the whole computation again from the beginning, and therefore the need for storing all measured data. Another drawback is in the multiple passes themselves, since they present a time performance penalty. Approaches presented in the following two subsections solve both these issues.

E. Naïve One-Pass Algorithm

A naïve approach to a one-pass computation of the variance is expanding its definition using the linearity of the expectation operator: $\operatorname{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - (\mathrm{E}[X])^2$. Similarly, the covariance definition can be expanded as well: $\operatorname{cov}[X,Y] = \mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])] = \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y]$.

To perform a t-test, only the raw moments $\mathrm{E}[X]$, $\mathrm{E}[Y]$ and $\mathrm{E}[X^2]$, $\mathrm{E}[Y^2]$ need to be estimated in a single pass through the data. Furthermore, in a case of CPA attack, the $\mathrm{E}[XY]$ value needs to be estimated too.

This approach allows for the addition of more power traces to the set without running the whole computation from the beginning. However, it may lead to serious numerical instabilities [13], which turns out to be an issue when analyzing a large number of power traces.

F. Stable One-Pass Algorithm

Numerically stable alternative to the naïve one-pass variance computation was presented in [14]. This approach is based on computing a mean-free sum M_2, where $\operatorname{Var}[X] = \frac{1}{n-1} M_2$. Formula for updating this sum to express a variance of a set with a newly added sample is presented in [14], as well as formula allowing to obtain a variance of a set created by merging two sets with known variances together.

Similar formula for the covariance estimation is presented in [15], based on the computation of a mean-free sum C_1, where $\operatorname{cov}[X,Y] = \frac{1}{n-1} C_1$.

This approach allows to add newly measured power traces any time and does not suffer from the numerical instability. Also, using these updating formulas for the variance and covariance estimation present no significant time penalty [16]. Therefore, only this approach to the first-order analysis will further be explored.

III. Higher-Order Analysis

As mentioned earlier, some countermeasures prevent an attacker from sucessfully mounting a first-order attack. These countermeasures include masking techniques [8], [17], which

are typically based on xor-ing a random mask value to the processed data and on altering the cryptosystem so that it still produces valid results. However, implementations protected with masking are often still vulnerable to higher-order attacks [18].

Higher-order Differential Power Analysis was first mentioned by Kocher et al. [1] as an attack that combines one or more samples within a single power trace. This kind of attack is useful when related intermediate cipher values are processed at a different time, e.g. when masking AES S-box in software implementations; it is reffered to as Multivariate Higher-Order Analysis and it will not be further discussed in this paper.

Different situation may occur in hardware implementations, where the intermediate values or their shares may be masked e.g. by parallel processing [9], i.e. they manifest themselves at the same time. Attacker can overcome this by performing a Univariate Higher-Order Analysis, which is performed at each sample independently. For univariate second-order analysis, every power sample gets mean-free squared [19]:

$$t'_i = (t_i - \mu_t)^2, \tag{4}$$

where μ_t is a sample mean at given sampling point. For univariate d-order analysis, where $d > 2$, every power sample gets additionally standardized:

$$t'_i = (\frac{t_i - \mu_t}{s_t})^d, \tag{5}$$

where s_t is a sample standard deviation at given sampling point.

Different approaches to the univariate higher-order analysis are described in Subsections III-A and III-B.

A. Naïve Multiple-Pass Preprocessing

The most straightforward way to perform any kind of higher-order analysis is to preprocess the power traces according to the expressions mentioned above, and then run the first-order analysis algorithm (e.g. CPA or t-test). The biggest drawback of this approach is the same as in case of any multiple-pass statistical algorithm: adding new samples to the statistical set requires launching the whole computation from the beginning (since the sample mean, used for the preprocessing, changes with new data).

B. Stable One-Pass Algorithm

Formulas allowing for robust one-pass higher-order analysis are given in [19], [12]. These take advantage of the fact that **mean and variance of the higher-order preprocessed power traces** can be expressed using **central moments of the raw power traces**. To estimate the n-th central moment, formulas similar to the ones mentioned in Subsection II-F are presented, allowing to estimate the n-th central moment of the set created by adding one or more new traces to an existing set. These formulas operate with mean-free values as well, and therefore avoid numerical instabilities.

Unfortunately, to update the estimate of n-th central moment, all the i-th central moments, $2 <= i < n$, are required.

Table I: Running Time of the First-Order CPA One-Pass Computation for a Various Number of Power Traces and 2,000 Samples Per Trace, in Seconds

# of power traces	2,000	4,000	8,000	16,000	32,000
1 CPU thread	18	41	72	143	289
2 CPU threads	10	20	41	85	169
4 CPU threads	6	12	25	51	105
GTX 1050	2	5	10	19	40

To express the variance of the d-th-order power traces, the $2d$-th and d-th central moments are necessary. In consequence, for the d-th order analysis, all the $2d$ central moments need to be kept and updated with every new power trace.

To estimate the covariance between the higher-order power traces and power predictions, similar formulas are given as well. For a d-th order analysis, d "adjusted central sums" need to be kept and updated.

These facts represent both time and memory penalties in comparison with the preprocessing approach – a cost for not being required to run the whole computation again with new power traces available. In case of very large datasets, this approach would be a preferred option. Unlike the preprocessing approach, these formulas allow for an on-line processing of power traces. It is also notable, that correlation matrices/t-values from central moments can be derived from first-order up to the order of the computation; unlike the preprocessing approach, where only the selected order is computed.

IV. Experimental Results

In this section, we evaluate the running time (wall time) of the first- and higher-order analysis, using some of the approaches described earlier. The CPU running times were measured on Intel Xeon E312xx (Sandy Bridge) machine, OpenMP was used for acceleration. For the GPU computations, nVidia GeForce GTX 1050 and OpenCL were used. Power traces are signed 16-bit integers, power predictions unsigned 8-bit integers, the computation is done in double precision floating point. All the algorithms are implemented in C/C++ for the sake of the best performance.

A. First-Order Analysis

Comparison of presented first-order analysis algorithms is given in [16], including acceleration on CPU using OpenMP. In this paper, we only focus on the Stable One-Pass Algorithm and we furthermore present our OpenCL accelerated version, capable of running the computation on GPU.

Table I presents running time of a First-Order CPA attack on AES128 (i.e. 256 different key candidates, the whole attack repeated 16 times), using 2,000 samples per power trace, and with various number of power traces. Running time of the GPU-accelerated version includes data transfers between RAM and GPU RAM.

As can be seen, the algorithm is well scalable and the running time grows linearly with the number of power traces.

Table II: Running Time of the Preprocessing of Power Traces, with 2,000 Samples Per Trace, in Order to Perform Arbitrary-Order Analysis, in Seconds

# of power traces	2,000	4,000	8,000	16,000	32,000
preprocessing 1 CPU thread	<1	1	2	4	8
preprocessing + CPA 1 CPU thread	18	42	74	147	297
preprocessing + CPA 4 CPU threads	6	13	27	55	113

Since the GPU acceleration is done using OpenCL with multiplatformity on mind, there are no vendor-specific optimizations used. It is fair to assume that with such optimizations present (e.g. using cuBLAS, the nVidia computational library optimized at assembler level for the maximum performance), the running time would be lower. Also, low-end GPUs are not well suited for double precision computation; single precision computation may be much faster, however, numerical instabilities may occur for large datasets.

First-order Welch's t-test is computationaly negligible in comparison with the CPA attack. This is because CPA attack requires computation of the covariance matrix, which is the primary bottleneck. Running time of first-order t-test computation is presented in Table V, alongside the higher-order analysis.

B. Higher-Order Analysis

Both presented approaches to the higher-order analysis will be discussed and examined in this subsection: using preprocessing, and using one-pass higher-order formulas.

1) Preprocessing: The preprocessing approach to the higher-order analysis consists of preprocessing the power traces according to the order of the attack, as described in Subsection III-A, and then running the first-order analysis, as is evaluated in Subsection IV-A. Order of the attack does not have effect on the complexity of preprocessing. Running time for 2,000 samples and various number of power traces is presented in Table II.

As mentioned before, obtaining results for a different order, as well as adding new power traces to the dataset, requires running the preprocessing and analysis again from the beginning. This drawback is solved by higher-order one-pass formulas.

2) One-Pass Formulas: Using formulas in [19], side-channel analysis up to arbitrary order may be performed in a way that allows for simple addition of new power traces, and for merging results based on processing different power traces sets together. Unlike the first-order attack, and the higher-order preprocessing approach, this algorithm requires $2d$ central moments to be computed for each sampling point. Furthermore, in case of CPA attack, d "adjusted central sums", which serve to compute the covariance between higher-order power traces and power predictions, must be computed as well. These facts represent a significant performance penalty, as can be seen in Table III and Figure 1, where higher-order CPA

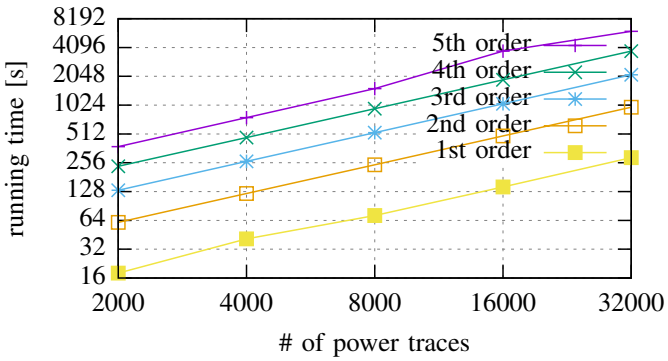

Figure 1: Comparison of the time performance of the CPA for different orders of the attack, using one-pass formulas.

Table III: Running Time of the Higher-Order CPA One-Pass Computation, With 2,000 Samples Per Trace, in Seconds

(a) 1 CPU Thread

# of power traces	2,000	4,000	8,000	16,000	32,000
1st order	18	41	72	143	289
2nd order	61	122	244	489	975
3rd order	132	265	529	1056	2124
4th order	235	469	941	1881	3758
5th order	377	759	1523	2995	6043

(b) 4 CPU Threads

# of power traces	2,000	4,000	8,000	16,000	32,000
1st order	6	12	25	51	105
2nd order	19	37	77	152	304
3rd order	43	79	158	310	618
4th order	69	138	277	547	1101
5th order	103	204	418	846	1633

running time is presented. The running time grows linearly with the number of power traces. Running time for different number of samples per trace is in Table IV.

Higher-order Welch's t-test, unlike the CPA attack, does not require covariance, i.e. d "adjusted central sums" matrices, to be computed. Since this is the bottleneck of the CPA computation, the t-test computation is much faster for the same number of power traces. Running time of higher-order t-test computation on 1 CPU thread is presented in Table V.

Table IV: Running Time of the Higher-Order CPA One-Pass Computation, for 32,000 Power Traces and Various Number of Samples Per Trace, 1 CPU Thread, in Seconds

# of samples per trace	500	1,000	2,000
1st order	79	152	289
2nd order	243	485	975
3rd order	517	1038	2123
4th order	908	1803	3758
5th order	1411	3797	6043

Table V: Running Time of Higher-Order Welch's t-test One-Pass Computation, With 2,000 Samples Per Trace and Various Number of Power Traces, 1 CPU Thread, in Seconds

# of power traces	32,000	64,000	128,000	256,000	512,000
1st order	<1	1	3	6	12
2nd order	1	3	6	12	25
3rd order	2	5	11	23	46
4th order	4	9	19	41	80
5th order	7	14	28	57	117

V. Conclusion

We have compared different approaches to first-order and higher-order side-channel analysis regarding their mathematical, performance and usage properties. Evaluated computations are crucial for Correlation Power Analysis attack, as well as for test vector leakage assesment using Welch's t-test.

Computation time required to perform side-channel analysis, both first-order and higher-order, proved to be linear with the number of power traces available. The time-consuming computations are well scalable, allowing us to utilize parallelism using both CPU and GPU devices.

Naïve approach to the higher-order analysis, i.e. preprocessing, performs nearly as fast as in case of the first-order analysis and the computation time does not depend on the selected order. However, for very large datasets, this approach may become troublesome because of its memory demands and usage properties. Addition of new power traces requires running the whole computation from scratch.

Using one-pass formulas for the same purpose allows for simple addition of new power traces to the dataset. Unfortunately, both memory demands and time complexity of this approach grow with the order of the analysis. On the other hand, all orders up to the specified order can be easily derived from the algorithm working variables. These usage properties make the one-pass approach a better choice for very large power trace datasets and for experimental work.

Higher-order analysis is necessary for attacking cryptographic devices secured against side-channel attacks. It is also necessary for evaluating the leakage of cipher implementations and success rate of countermeasures. Our optimized implementations allow us to choose the right approach for specific task, and significantly speed up these computations. Discussed implementations are part of our open-source side-channel toolkit SICAK [20], allowing anyone for further usage and improvements.

Acknowledgement

Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures".

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS17/213/OHK3/3T/18.

References

[1] P. Kocher, J. Jaffe, and B. Jun, *Differential Power Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[2] B. den Boer, K. Lemke, and G. Wicke, "A dpa attack against the modular reduction within a crt implementation of rsa," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 228–243.

[3] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.

[4] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[5] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 450–466.

[6] E. Biham, R. Anderson, and L. Knudsen, "Serpent: A new block cipher proposal," in *International Workshop on Fast Software Encryption*. Springer, 1998, pp. 222–238.

[7] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2005, pp. 172–186.

[8] J. Blömer, J. Guajardo, and V. Krummel, "Provably secure masking of aes," in *International Workshop on Selected Areas in Cryptography*. Springer, 2004, pp. 69–83.

[9] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: a very compact and a threshold implementation of aes," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2011, pp. 69–88.

[10] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136.

[11] P. Socha, V. Miškovský, H. Kubátová, and M. Novotný, "Correlation power analysis distinguisher based on the correlation trace derivative," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 565–568.

[12] T. Schneider and A. Moradi, "Leakage assessment methodology," *Journal of Cryptographic Engineering*, vol. 6, no. 2, pp. 85–99, 2016.

[13] T. F. Chan and J. G. Lewis, "Computing standard deviations: accuracy," *Communications of the ACM*, vol. 22, no. 9, pp. 526–531, 1979.

[14] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," in *COMPSTAT 1982 5th Symposium held at Toulouse 1982*. Springer, 1982, pp. 30–41.

[15] P. Pébay, "Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments," *Sandia Report SAND2008-6212, Sandia National Laboratories*, vol. 94, 2008.

[16] P. Socha, V. Miškovský, H. Kubátová, and M. Novotný, "Optimization of pearson correlation coefficient calculation for dpa and comparison of different approaches," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2017 IEEE 20th International Symposium on*. IEEE, 2017, pp. 184–189.

[17] M. Rivain and E. Prouff, "Provably secure higher-order masking of aes," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 413–427.

[18] T. S. Messerges, "Using second-order power analysis to attack dpa resistant software," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2000, pp. 238–251.

[19] T. Schneider, A. Moradi, and T. Güneysu, "Robust and one-pass parallel computation of correlation-based attacks at arbitrary order," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2016, pp. 199–217.

[20] P. Socha, "Sicak: Side-channel analysis toolkit." [Online]. Available: https://petrsocha.github.io/sicak/