World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:5, 2014

Accelerating Side Channel Analysis with Distributed and Parallelized Processing

Kyunghee Oh, Dooho Choi

***Abstract*—**Although there is no theoretical weakness in a cryptographic algorithm, Side Channel Analysis can find out some secret data from the physical implementation of a cryptosystem. The analysis is based on extra information such as timing information, power consumption, electromagnetic leaks or even sound which can be exploited to break the system. Differential Power Analysis is one of the most popular analyses, as computing the statistical correlations of the secret keys and power consumptions. It is usually necessary to calculate huge data and takes a long time. It may take several weeks for some devices with countermeasures. We suggest and evaluate the methods to shorten the time to analyze cryptosystems. Our methods include distributed computing and parallelized processing.

***Keywords*—**DPA, distributed computing, parallelized processing.

I. Introduction

MOST computing devices including communication devices contain hardware or software cryptographic modules in them and the security relies on the modules. But, although there is no theoretical weakness in a cryptographic algorithm, the physical implementation of the cryptosystem may leak some information that makes the cryptosystem vulnerable to side channel attacks [1].

Side channel analysis (SCA) is a cryptanalysis which analyzes the side channel information such as timing information, power consumption, electromagnetic leaks or even sound of devices. If cryptosystems do not have countermeasures against SCA, they may face serious risks. SCA on devices is a necessary procedure to verify the safety of computing devices.

Differential Power Analysis (DPA) is a kind of SCA. To perform a DPA test for a device, it is needed to calculate statistics from huge data, which are the side channel information. It may take several weeks for some devices with countermeasures.

Our work suggests and evaluates the methods to shorten the time to analyze cryptosystems. It includes distributed computing over multiple computers and parallelized processing.

II. Side Channel Analysis

When a computing device performs some calculations, it consumes electric power, and the consumption varies with the values that it is calculating. So an attacker can use the power consumption data as the leakage information for SCA.

In a broad sense of SCA, there are passive and active analyses. Passive SCA is based on only the leakage information from a cryptosystem. Active SCA is based on some physical impacts to the system. Fault injection is an active analysis, which injects voltage glitch or laser to make devices malfunction. But usually, SCA means only passive SCA and our research focuses on it.

There are two kinds of passive SCAs of Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Both analyses use the power traces of power consumption or electromagnetic radiation from the analyzing device.

SPA interprets the visual graph of power traces. The operations of a cryptographic algorithm are composed of repeating similar arithmetic operations, so the power traces show repeating patterns over time. For some cases, the pattern may mirror the key value.

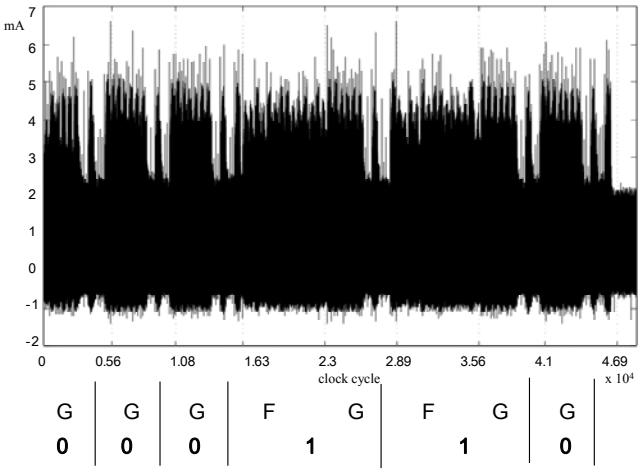


Fig. 1 SPA for ECC

Fig. 1 is the graph of a power trace during the point multiplication of elliptic curve cryptography [2]. Only doubling calculation occurs when the key bit is 0, and doubling and addition occur when the key bit is 1. The patterns are apparent in the graph and we can easily find out the key.

Although the visual graphs of power traces do not show any useful information to human eyes, that is, SPA is not available DPA may succeed to find the keys.

The attacker of DPA computes the hypothetical changes of intermediate values within cryptographic operations, and compares them with the multiple power traces collected from cryptographic operations. The correct key makes strong correlations between the hypothetical values and the real power consumptions.

The attacker collects multiple power traces from the

K. Oh and D. Choi are with the Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: {khoh,dhchoi}@etri.re.kr).

cryptographic calculations with random plaintexts. Usually there are many useless noises in the raw power traces, the traces need to be refined by various preprocessing such as frequency filtering or time alignment. Finally, to find the key, the correlation values are calculated between the plaintext and each key of all available intermediate key for the whole traces. The correct key will result in high correlation. The operation quantity of these processes is very huge and takes much time.

```
for each trace T[L], P
    for each k, gk (where 0≤k<K, 0≤gk<GK)
        data[k, gk] = hypothetical values of P
        d[k, gk] += data[k, gk]
        d2[k, gk] += data[k, gk]^2
    for each l'th point of T (where 0≤l<L)
        s[l] += T[l]
        s2[l] += T[l]^2
    for each l, k, gk
        sd[l, k, gk] += T[l] · data[k, gk]

finally, for each l, k, gk
    R[l, k, gk] = (L·sd[l, k, gk] – d[k, gk]·s[l])
                / sqrt{(L·d2[k, gk] - d[k, gk]^2) · (L·s2[l] – s[l]^2)}
```

where

L	: number of points in a power trace
T[L]	: power values of each point of a trace
N	: number of power traces
P	: plain text or cipher text
K	: number of bytes of the key
GK	: number of available guessing keys
s[L]	: sum of power values for each point of traces
s_2[L]	: sum of squared power values for each point of traces
d[K, GK]	: sum of hypothetical values (hamming weight) for each guessing key
d_2[K, GK]	: sum of squared hypothetical values for each guessing key
sd[L, K, GK]	: sum of multiplications of power values and hypothetical values
R[L, K, GK]	: correlation of hypothetical values and power values for each guessing key and power trace

Fig. 2 DPA pseudo codes

Fig. 2 shows the procedure to get the correlation values R[L, K, GK]. If there are some points of which the correlation values are relative high for a guessing key of each key byte, the key is found. The calculation complexity is O(N·L·K·GK) and the required data memory size for s, s2, d, d2, and sd is about L·K·GK. For the case of AES128, K is 16, GK is 256. The value of N or L are various from thousands to millions by test cases.

III. Accelerating SCA Operations

As the time cost for SCA is not negligible, reducing the calculation time of SCA is important. One of accelerating method is using graphics processing units (GPU). GPU has powerful parallel processing function for cryptanalysis [3]. And [4] has implemented the GPU accelerating function for SCA. But there is the limit of the GPU memory size, and I/O overhead increases rapidly when the required memory is bigger than that of the GPU.

This work proposes distributed computing and parallelized processing for acceleration. With distributed processing it also has the benefit that cooperating computers divide the required memory size.

A. Distributed Computing

The performance can be improved by distributing the analysis operations to multiple computers. But it is needed to manage data and resources properly. The overall performance differs as how to structure the distributed system.

Our distributed SCA system is composed of one master system that manages overall distributed processing and multiple slave systems that cooperate on the analysis.

In most trace preprocessing operations, processing a trace is independent to other trace data. So it is easy to distribute operations. Traces are assigned randomly to slave systems, and merged again to the master system after processing.

Distributing DPA operations is a little complicated. A unit DPA operation has to process the data of multiple traces, and the intermediate analysis data should be retained until overall analysis is completed. As a DPA operation of Fig. 2 can be divided by the l index, which is the point position of traces, we divided a trace by the number of computers and allocated each segment to each slave system. Fig. 3 depicts the allocation of trace data to the slaves.

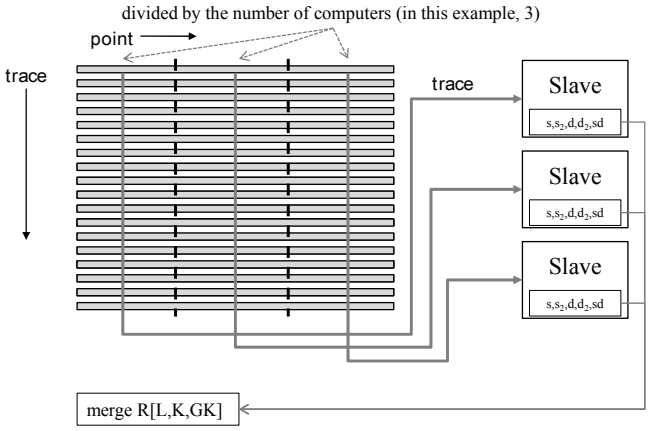


Fig. 3 Distributing power traces to slave systems

Dividing traces also have benefits in memory usage. As L, the number of points in a trace, is divided, the required memory, L·K·GK, is also divided by slaves.

Fig. 4 depicts the DPA performance evaluation of our distributed SCA implementation. On this test, we analyzed 2,000 AES power traces, and each of traces has 5,000 points. It took 109 seconds with a standalone system. As the number of computers increases, the analyzing time decreases. With 4 computers, it took only 34% of the standalone system. This result shows 15-27% of computing resource is the overhead of management for distributed computing.

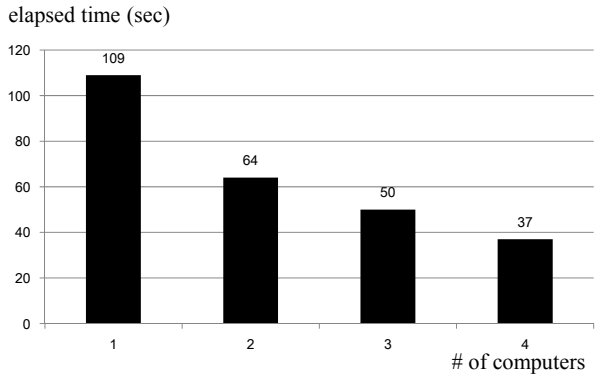
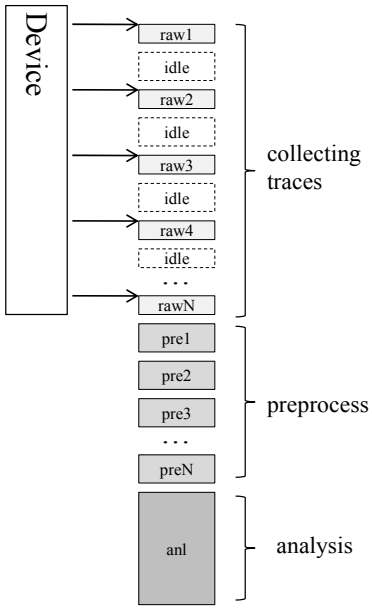
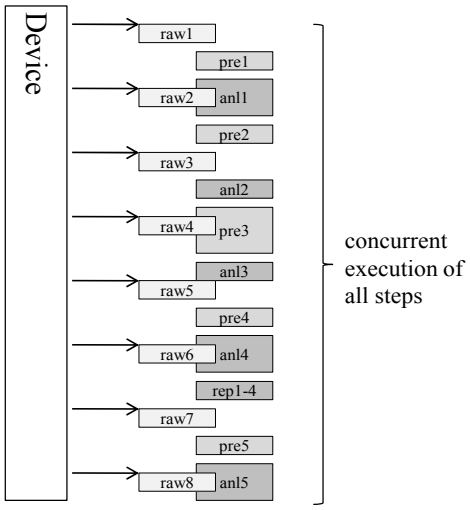


Fig. 4 The number of computers and the time required for a DPA test



(a) Sequential procedure



(b) Parallelized procedure

Fig. 5 Sequential and Parallelized SCA procedure

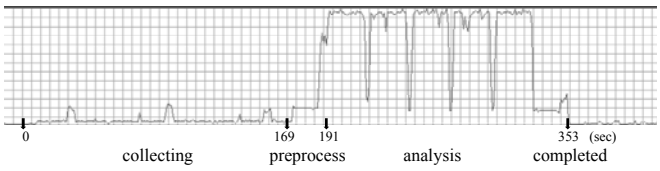
B. Parallelized Processing

SCA procedures include 3 steps of collecting power traces, preprocessing with signal processing, and analyzing of calculating the correlation values. The collecting step which makes raw power traces does not require much computing resource, but the main time factor in collecting traces is related to the performance of the target device. The cryptographic operations of many security devices including smart cards takes much time. The signal processing and statistical computing for the analysis can be huge calculations and it may also take much time although with high performance analyzing systems.

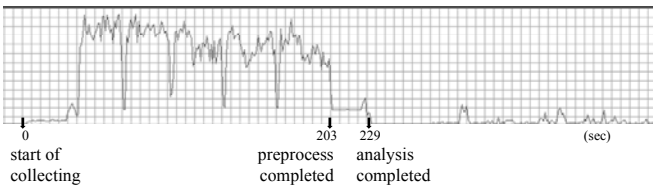
These procedures are common to the commercially available SCA tools [4], [5], and they proceed the steps sequentially as Fig. 5 (a).

We enhanced overall performance by executing other steps concurrently during the collecting step. We designed our implementation to process preprocessing step and analyzing step simultaneously when each new power trace is available. Fig. 5 (b) depicts the parallelized procedure.

Fig. 6 is our comparison result for each procedure. The CPU utilization of the analyzing system with sequential procedure of Fig. 6 (a) is not effective during the collecting step. But the parallelized processing of Fig. 6 (b) utilizes the CPU more effectively as it proceeds the preproccess and analysis steps while collecting power traces and shortens the time to finish the test.



(a) Sequential procedure



(b) Parallelized procedure

Fig. 6 Comparison of CPU utilization and elapsed time

IV. Conclusion

Safety against SCA became a common requirement for various security devices. Many standards including CC, EMV, and FIPS140-3 require the safety verification against SCA for security hardware. Our accelerated SCA tools can enhance the verification tests more efficiently as it shortens the time for the test. And our methods are applicable not only DPA but other advanced analyzing method such as High Order DPA.

Acknowledgment

This work was supported by the K-SCARF project, the ICT R&D program of ETRI (Research on Key Leakage Analysis

and Response Technologies)

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, Differential power analysis, Advances In Cryptology - CRYPTO' 99, LNCS 1666 Springer-Verlag, pp. 388-397, Santa Barbara, USA, August 1999
[2] S. B. Örs, E. Oswald, B. Preneel, Power-Analysis Attacks on an FPGA - First Experimental Results, CHES 2003.
[3] Hans ChristophHudde, GPU assisted Mutual Information Analysis Attacks on AES, Bachelor Thesis, Ruhr-Universität Bochum, April 2010
[4] Riscure Inspector, https://www.riscure.com/security-tools/inspector-sca/, 2014.3
[5] Brightsight Sideways, http://www.brightsight.com/tools/sideways, 2014.3