Thomas Eisenbarth
Yannick Teglia (Eds.)

LNCS 10728

Smart Card Research and Advanced Applications

16th International Conference, CARDIS 2017
Lugano, Switzerland, November 13–15, 2017
Revised Selected Papers

Springer

Lecture Notes in Computer Science 10728

Thomas Eisenbarth · Yannick Teglia (Eds.)

Smart Card Research and Advanced Applications

16th International Conference, CARDIS 2017
Lugano, Switzerland, November 13–15, 2017
Revised Selected Papers

Springer

Editors

Thomas Eisenbarth
University of Lübeck and WPI
Lübeck
Germany

Yannick Teglia
Gemalto
La Ciotat
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-75207-5 ISBN 978-3-319-75208-2 (eBook)
https://doi.org/10.1007/978-3-319-75208-2

Library of Congress Control Number: 2018931376

LNCS Sublibrary: SL4 – Security and Cryptology

Preface

These proceedings contain the papers selected for presentation at the 17th International Conference on Smart Card Research and Advanced Applications (CARDIS 2017), held in Lugano, Switzerland, during November 13–15, 2017. The conference was organized by the ALaRI institute on the campus of Universita della Svizzera italiana in Lugano.

CARDIS has provided a space for security experts from industry and academia to exchange ideas on security of smart cards and related applications. Those objects have been part of our daily life for years: banking cards, SIM cards, electronic passports, etc. But the world is changing; the smartcard, as a secure element, is more and more often the hardware root of trust of bigger systems. Their applications and use cases are also increasing through M2M and massive IoT. As such, smartcard security is key since the security of entire systems relies on it. With the growing use of smartcard technology, the attack surface is also increasing, from physical attacks to logical attacks, from local attacks to remote attacks. Combined attacks are also to be considered. It is more important than ever that we understand how smart cards, and related systems, can be secured.

This year, CARDIS received 48 papers from 22 countries. Each paper was reviewed by three independent reviewers. The selection of 14 papers to fill the technical program was accomplished based on 142 written reviews. This task was performed by the 32 members of the Program Committee with the help of 35 external reviewers. The technical program also featured three invited talks. The first invited speaker, Pierre Girard, from Gemalto, France, presented “Security for IoT”; the second speaker, Matthieu Rivain, from CryptoExperts, presented “White Box Cryptography”; and the third speaker, Thomas Pöppelmann, from Infineon Technologies, Germany, presented “Post Quantum Cryptography.”

We would like to thank the general chair, Francesco Regazzoni, for the great venue and smooth operation of the conference. We would also like to thank the Program Committee and the external reviewers for their thorough work, which enabled the technical program to be of such high quality, and the Steering Committee for giving us the opportunity to serve as program chairs at such a prestigious conference. The financial support of all the sponsors was highly appreciated and greatly facilitated the organization of the conference. We would like to thank the platinum sponsor Hasler Foundation, the gold sponsor Infineon Technologies, the silver sponsors Gemalto, Idemia, Micron Foundation, and Rambus, and the Fondazione Politecnico di Milano for the collaboration. Furthermore, we would like to thank the authors who submitted their work to CARDIS 2017, without whom the conference would not have been possible.

December 2017

Thomas Eisenbarth
Yannick Teglia

CARDIS 2017

16th International Conference on Smart Card Research and Advanced Applications Lugano, Switzerland November 13–15, 2017

Oliver Mischke	Infineon Technologies, Germany
Amir Moradi	Ruhr University Bochum, Germany
Yossi Oren	Ben-Gurion University, Israel
Pedro Peris-Lopez	University of Madrid, Spain
Axel Y. Poschmann	DarkMatter, UAE
Emmanuel Prouff	ANSSI, France
Patrick Schaumont	Virginia Tech, USA
Mike Tunstall	Cryptography Research, USA
Carolyn Whitnall	University of Bristol, UK

Steering Committee

Aurélien Francillon	EURECOM, France
Marc Joye	NXP Semiconductors, USA
Edouard de Jong	n-Count Technology, The Netherlands
Jean-Louis Lanet	University of Limoges, France
Stefan Mangard	University of Graz, Austria
Konstantinos Markantonakis	Royal Holloway University of London, UK
Amir Moradi	Ruhr Uni Bochum, Germany
Svetla Nikova	Katholieke Universiteit Leuven, Belgium
Pierre Paradinas	Inria and CNAM, France
Emmanuel Prouff	Safran Identity and Security, France
Jean-Jacques Quisquater	Université Catholique de Louvain, Belgium
Vincent Rijmen	Katholieke Universiteit Leuven, Belgium
Pankaj Rohatgi	Cryptography Research, USA
François-Xavier Standaert	Université Catholique de Louvain, Belgium

Additional Reviewers

Jean-Philippe Aumasson
Yang Cao
Liron David
Danny De Cock
Wieland Fischer
Si Gao
Hannes Gross
Karine Heydemann
Eliane Jaulmes
Anthony Journault
Elif Bilge Kavun
Houssem Maghrebi
Filippo Melzani
Thorben Moos
Jungmin Park
Hervé Pelletier
Peter Pessl
Romain Poussier
Julien Proy
Oscar Reparaz
Bastian Richter
Michael Rodler
Yolan Romailler
Niels Samwel
Manuel San Pedro
Joern-Marc Schmidt
Tobias Schneider
Johanna Sepulveda
Francois-Xavier Standaert
Ruggero Susella
Thomas Unterluggauer
Karine Villegas
Mario Werner
Brecht Wyseur
Xiaolin Xu

Abstracts of Invited Talks

White-Box Cryptography

Matthieu Rivain

CryptoExperts, Paris, France
matthieu.rivain@cryptoexperts.com

The goal of white-box cryptography, as introduced in 2002 by Chow *et al.*, is to protect a secret key used in a cryptographic software against an adversary that has full access to the underlying execution environment [CEJvO02, CEJv03]. The basic approach is to design an obfuscated cryptographic implementation embedding the key so that, even for such a powerful adversary, key recovery is made difficult. Originally targeting DRM applications, white-box cryptography has recently gained momentum with the advent of mobile payment, and especially for solutions without hardware secure elements (see for instance the HCE technology). More generally, the development of smart applications running on untrusted environments is very appealing for white-box cryptography as a building block of wider security solutions. Despite the industrial deployment of white-box cryptography, the advances have been very limited from the scientific point of view. As of today, all the proposed white-box implementations in the public literature have been broken, which has driven the industry to develop home-made solutions relying on obscurity (*i.e.* secrecy of the underlying obfuscation techniques). The theory of *cryptographic obfuscation* has yet known some progress and many candidate obfuscation schemes have recently been proposed in the crypto literature. However, the security of these schemes is still under investigation, and their performances are far from meeting any practical requirement. This talk[1] presents an overview of the current state of white-box cryptography (as of the end of 2017) in terms of theory and practice, and with a focus on the recent ECRYPT competition [ECR17].

White-box cryptography theory. In the first part of this talk, we introduce some basic definitions and security notions. After introducing the concepts of *obfuscator* and *white-box compiler*, we review the seminal work of Barak *et al.* [BGI+01]. Specifically, we recall the notion of *virtual black box* obfuscation – stating that an obfuscated program should reveal nothing more that its input/output behaviour – and see why this notion cannot be achieved by a general-purpose obfuscator. We then recall the alternative (weaker) notion of *indistinguishability obfuscation* – stating that the obfuscations of two functionally equivalent programs should be computationally indistinguishable – and discuss the meaning of these notions when applied to a white-box compiler. This stresses the need for further security notions dedicated to white-box cryptography. We then review some previous works on the subject and in particular the notions of *unbreakability*, *one-wayness*, *incompressibility* and *traceability* as formalised in [DLPR14].

[1] The slides are available at www.matthieurivain.com/files/slides-cardis17.pdf.

White-box cryptography practice. In the second part of this talk, we focus on practical white-box designs and attacks. We recall the principle of the original white-box implementations proposed by Chow *et al.* – with a closer look at the AES use case [CEJv03] – and we review different attacks against these designs. We then consider some *generic attacks* targeting white-box implementations for which the design is kept secret. Specifically, we focus on *fault analysis* – such as *e.g.* the attack of [PQ03] – and *differential computation analysis* which applies side-channel attack techniques to the white-box context [SMdH15, BHMT16]. We give some insights into why these attacks are so effective at breaking white-box cryptography. We then discuss the application of standard countermeasures to the white-box context and exhibit some related issues.

White-box cryptography competition. The last part of this talk focuses on the recent white-box cryptography competition organised by the ECRYPT CSA project as the CHES 2017 CTF challenge [ECR17]. During this competition, developers were invited to submit their white-box AES implementations (without revealing the underlying design) and attackers were challenged to break the submitted implementations (*i.e.* extract the embedded keys). After giving a wrap up of the rules and the results of the competition, we describe the break of the winning implementation, *i.e.* the one that remained unbroken the longest time (28 days). We depict the different steps of the break: reverse engineering, circuit minimization, data dependency analysis, and key recovery [GPRW17].

References

[BGI+01] Barak, B. et al.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–8. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1

[BHMT16] Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential computation analysis: hiding your white-box designs is not enough. In: Gierlichs, B., Poschmann, A. (eds.) CHES 2016. LNCS, vol. 9813, pp. 215–236. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_11

[CEJv03] Chow, S., Eisen, P., Johnson, H., Van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_17

[CEJvO02] Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 1–15. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-540-44993-5_1

[DLPR14] Delerablée, C., Lepoint, T., Paillier, P., Rivain, M.: White-box security notions for symmetric encryption schemes. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 247–264. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_13

[ECR17] ECRYPT CSA Consortium: The WhibOx Contest – An ECRYPT White-Box Cryptography Competition. CHES 2017 Capture the Flag Challenge (2017). https://whibox.cr.yp.to/

[GPRW17] Goubin, L., Paillier, P., Rivain, M., Wang, J.: How to reveal the secrets of an obscure white-box implementation. Cryptology ePrint Archive (2017). https://eprint.iacr.org/

[PQ03] Piret, G., Quisquater, J.-J.: A differential fault attack technique against SPN structures, with application to the AES and Khazad. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp 77–88. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45238-6_7

[SMdH15] Sanfelix, E., Mune, C., de Haas, J.: Unboxing the white-box – practical attacks against obfuscated ciphers. Black Hat 2015 (2015)

Post-quantum Cryptography

Thomas Pöppelmann

Infineon Technologies AG, Germany
thomas.poeppelmann@infineon.com

Abstract. Currently, signature schemes or public-key encryption based on RSA or elliptic curve cryptography (ECC) are used in various internet and industry standards like transport layer security (TLS), PGP/GPG, IKE, or S/MIME. They protect communication between smart cards, smart phones, computers, servers, the cloud, or industrial control systems. However, in case one of the numerous attempts and approaches to build a powerful enough quantum computer turns out to be successful, Shor's algorithm could be used to break RSA and ECC in polynomial time. At the moment big enterprises like Google, IBM, Intel, and Microsoft as well as research institutions worldwide are working on the development of quantum computers. Some experts forecast that the development of quantum computers capable to run Shor's algorithm might be realistic within a timeframe of 15–20 years. More evidence that concerns about quantum attacks are realistic is also given by an announcement of the NSA Information Assurance Directorate that it will initiate a transition to quantum resistant algorithms in the not too distant future. In addition, the US National Institute of Standards and Technologies (NIST) is currently running a process to standardize new quantum resistant algorithms. Such new quantum resistant algorithms that are supposed to withstand attacks by quantum computers are also called post-quantum cryptography (PQC). PQC algorithms are executed on classical computers and are supposed to be based on mathematical problems and foundations that are hard to solve, even for quantum computers. Usually, PQC schemes have the same or similar high-level behavior as currently available cryptosystems so that they can act as a drop-in replacement to RSA and ECC. The five common hardness assumptions or families of algorithms from the literature that are used to realize asymmetric post-quantum cryptography are hash-based signatures, code-based cryptography, multivariate cryptography, lattice-based cryptography, and isogeny-based cryptography. Currently, in the field of PQC some challenges and opportunity for further optimization exist. Examples are the reduction of the size of signatures, ciphertexts, and keys as well as cryptanalysis of currently proposed parameter sets to increase confidence in the security of the new schemes. Additionally, for practical realization on smart cards or other adverse environments countermeasures against side-channel and fault attacks are required. While first works are promising, it appear that the field is in an early stage and that more work is required to fully understand secured implementation of PQC. However, already today it seems that PQC will have a disruptive impact on the security industry and that industry and academia have to act on it now to be prepared.

Contents

Opening Pandora's Box: Effective Techniques for Reverse Engineering IoT Devices

Omer Shwartz(✉), Yael Mathov(✉), Michael Bohadana(✉), Yuval Elovici, and Yossi Oren(✉)

Ben-Gurion University of the Negev, Beersheba, Israel
{omershv,yaelmath,bohadana}@post.bgu.ac.il,
{elovici,yos}@bgu.ac.il

Abstract. With the growth of the Internet of Things, many insecure embedded devices are entering into our homes and businesses. Some of these web-connected devices lack even basic security protections such as secure password authentication. As a result, thousands of IoT devices have already been infected with malware and enlisted into malicious botnets and many more are left vulnerable to exploitation.

In this paper we analyze the practical security level of 16 popular IoT devices from high-end and low-end manufacturers. We present several low-cost black-box techniques for reverse engineering these devices, including software and fault injection based techniques for bypassing password protection. We use these techniques to recover device firmware and passwords. We also discover several common design flaws which lead to previously unknown vulnerabilities. We demonstrate the effectiveness of our approach by modifying a laboratory version of the Mirai botnet to automatically include these devices. We also discuss how to improve the security of IoT devices without significantly increasing their cost.

1 Introduction

In the early days of computing, low-cost ubiquitous devices were generally powered by simple microcontrollers. These microcontrollers typically ran a very limited software stack, ranging from a fixed-function program running in a busy loop to a limited functionality real-time operating system (RTOS). As technology matured, it became more cost-effective to create these devices around a fully-featured operating system such as Linux, taking advantage of the existing code base and of the relative ease of development and debugging. This is especially the case in the Internet of Things (IoT), which can be defined as a network of smart electronic devices with internet connectivity. In the past years we have been witnessing a dramatic rise in the amount of connected devices and recently, wireless connected devices. The number of IoT devices is estimated to reach 50 billion by 2020 [17].

O. Shwartz, Y. Mathov and M. Bohadana contributed equally to this paper.

T. Eisenbarth and Y. Teglia (Eds.): CARDIS 2017, LNCS 10728, pp. 1–21, 2018.
https://doi.org/10.1007/978-3-319-75208-2_1

The task of the device security engineer has also evolved with the move from ASICs and simple microcontrollers to complete Linux devices. Traditional hardware attack methods which target ICs are less effective in this modern situation, since the hardware can be assumed to be generic and even shared between different vendors. Ubiquitous network connectivity also changes the attack model, making it interesting to examine the vulnerability of devices to remote attacks, or the ability of an attacker to translate a single instance of physical access to widespread damage to many devices. Indeed, the introduction of these small, embedded devices unto the web and into residencies and businesses was quickly followed by emerging security challenges [39]. The rapid growth in the quantity and variety of IoT devices created a scenario where millions of devices [27] are deployed while the consumers may know very little about their composition and security. This is especially crucial since IoT devices are often equipped with a wide array of sensors, are connected to private networks and control a variety of physical systems, from entry gates and door locks to HVAC systems (Heating, Ventilation and Air Conditioning) [29].

In this work we present a general methodology for "black-box" reverse engineering of complete stack IoT devices. The techniques presented should answer many use cases and can be used as a tutorial for accessing new devices. While most of the techniques we use are generally well known, this is to the best of our knowledge the first time they are applied systematically to many different IoT devices. This allows us to make quantitative arguments about the state of IoT security today.

In detail, our paper makes the following contributions: We present a systematic reverse engineering workflow appropriate for complete-stack IoT devices in a detailed and tutorial-like manner. We apply this workflow to sixteen IoT devices produced by different manufacturers and discuss their common characteristics and security flaws. Finally, we offer some guidance to implementors interested in making these devices more secure.

1.1 Related Work

Mahmoud et al. [28] present a survey of the current concerns for IoT security. The authors describe the general architecture of IoT devices and the security challenges rising from this design, corresponding to the security principles of confidentiality, integrity, availability and authentication. Sicari et al. [36] claim that the network communication characteristics of IoT devices, combined with the increase in exchanged information, multiplies the potential for attacks on the system privacy leaks. Similar concerns were also raised by Alqassem and Svetinovic [6] and by Zhang et al. [40]. Interestingly, most of this analysis centers on security threats to the **user** of the IoT device (i.e. loss of confidentiality and availablility) and less on the risks to the **device itself** (e.g. counterfeiting).

Patton et al. [31] studied the extent of vulnerabilities found in network-accessible IoT devices. They reviewed several network scanners and focused on Shodan [35], a publicly available search engine for internet connected services. Using Shodan, the authors discovered many vulnerable IoT systems including a large number of SCADA (Supervisory Control And Data Acquisition) systems. Similar techniques can also be found in the work of Bodenheim et al. [10].

Tellez et al. [37] focused on WSN (Wireless Sensor Networks) and elements of their security. For their research, the authors chose the MSP430 MCU and investigated it. The BSL password (Bootstrap Loader) that protects the MCU from unauthorized access was presented as a main security feature of the MSP430 MCU. A flaw detected in the BSL password mechanism through reverse engineering techniques allowed the researches to easily break into a secured MCU. The authors also suggest ways for designing a Secure-BSL that can improve the MCUs protection.

Gubbi et al. [20] offered an all-in-one review of the WSN terrain along with the terminology that exists within it. Halderman et al. [21] showed techniques for recovering secrets from DRAM (Dynamic Random Access Memory) modules by transferring the modules into a new machine while minimizing data decays. Lanet et al. [24] showcase methods for reverse engineering EEPROM data of java memory cards. They describe forensics methods which enable the researcher to locate critical data within the memory image, account for errors and eventually rebuild the original applet code that is stored in the card.

Obermaier and Hutle [30] employs reverse engineering techniques on several wireless security cameras and shows how these are vulnerable to remote attackers with no physical access to the surroundings of the device. The authors show various encryption and communication faults that may allow and attacker to impersonate a camera and eavesdrop or sabotage its communication.

1.2 Embedded Device Software Architectures

Software architecture determines many of a device's properties and limitations, the architecture may include an OS (Operating System) or not. We differentiate between three main types of software architectures present in embedded devices. **Full-stack OS based devices** contain a modern operating system, such as Linux, that separates execution into kernel mode and user mode. While traditionally this architecture was preferred only when versatility and high performance was needed, [22], more and more low-cost devices are now based on Linux due to falling component costs and the ease of developing for this operating system. In particular, many of the cameras we surveyed had a complete-stack Linux implementation. **Partial stack OS based devices** are devices with a special-purpose real-time operating system (RTOS) such as VxWorks or vendor-provided OS implementation. These devices are generally very specifically crafted for their task [16], and tend to omit some of the features of complete-stack OS. Some lower-end or single tasked IoT devices use this architecture, with the RTOS handling WiFi, web protocols, and added vendor code in charge of gathering sensor data. Finally, **devices with no operating system** are embedded devices which directly execute compiled instructions, without any OS support for functionalities such as threading or interrupts. Devices with no OS can offer better raw performance and higher run-time predictability than other architectures, but tend to have increased difficulty of development, causing a longer time-to-market.

While it is the authors' belief that *partial stack OS devices* have the potential for security vulnerabilities, *full-stack OS devices* were chosen as the target for reverse engineering in this paper. So far, all of the IoT attacks seen in the wild, and known to the authors, had targeted *full-stack OS devices*, as these are more generic and make use of many drivers and open source components that may have vulnerabilities.

2 Reverse Engineering Methodology

Following is a description of the flow of actions performed in order to gain access to the software of IoT devices, run foreign applications on it and extract secrets such as credentials used for accessing the device. This section focuses on reverse engineering "black-box" devices where no previous knowledge about the device is required. The tools used for assessing these techniques can be seen in Table 3 in Appendix.

Our black-box reverse engineering process follows a standard workflow that can be seen in Fig. 1:

1. Physical inspection of the device.
2. Extraction of the device firmware image and file system:
   (a) Bypass boot-time security and recover the firmware image.
   (b) Recover the data with out-of-band means.
3. Analysis the firmware image and recovery of the secrets inside.

Inspect Device
Extract Data
Analyze Firmware

Fig. 1. The building blocks of black-box reverse engineering

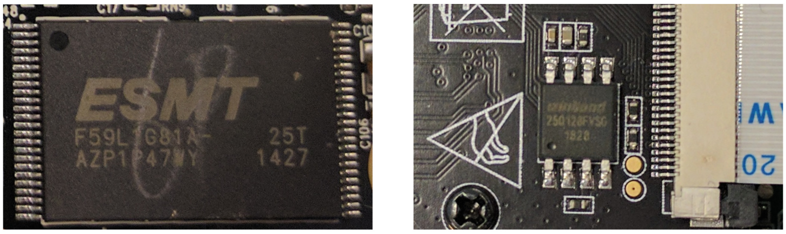
2.1 Inspection of the Device

Most of the devices can be carefully opened without damaging neither the exterior of the device nor the internal components.

Locating and Identifying Memory Components. Smart devices that run the Linux operating system require enough non-volatile memory for storing the kernel and additional mandatory file system components. A cheap and efficient way for engineering such devices is placing the memory module outside of the main processor package. Devices engineered in such configurations usually employ a processor that is capable of loading and running instructions directly from SPI (Serial Parallel Interface) Flash memory or EEPROM (Erasable Programmable Read-Only Memory) devices.

Understanding the memory technology is crucial for performing firmware extraction when there is no capability to run commands on the tested device, see more details in Subsect. 2.2.

It is common to find a memory module that uses technology consistent with the required capacity inside devices. Common examples are: 25XX\26XX series eight-pin SPI flash memory with up to 32 MB of storage space; larger SPI Flash devices with sixteen or thirty-two pins; NAND Flash devices that come in various capacities and shapes and are usually coupled with a 24XX EEPROM module for holding initial configuration; eMMC (embedded Multi-Media Controller) modules or cards usually containing more than a GB of data. Examples of memory modules can be seen in Fig. 2.



(a) F59L1G81A 1GB NAND Flash module inside Xtreamer Cloud Camera

(b) W25Q128FVSG 16MB SPI Flash module inside Ennio SYWIFI002 Wireless Doorbell

Fig. 2. Examples of onboard memory

Identification of the memory module can be performed by searching of the engraved device codes on the IC (Integrated Chip) package. In most cases the modules used are commonly known and available off-the-shelf with public datasheets.

Locating UART Terminals. UART (Universal Asynchronous Receiver/Transmitter) ports can be found on many smart devices. UART ports are commonly used for development and maintenance via a Linux console that the port is bound to. UART ports' communication is based on a specified protocol in predetermined baud rate, typically 9600, 57000, or 115200 bits-per-second.

In many cases, UART terminals are embedded into the PCB (Printed Circuit Board) in the prototyping stages of a product's life and are kept in the design during production either to reduce costs of redesign or maintain access for future maintenance. In certain cases, UART terminals are placed in a visible and accessible locations, occasionally marked with their purpose. In other cases the terminals are purposefully or unpurposefully hidden between many other test points exposed on the boards for post-production testing. Connecting to

UART terminals allows easy access for communication with the OS, and may also form a beachhead for the effort of reverse engineering.

Basic UART communication requires only three electrical lines: TX (Transmit), RX (Receive) and GND (Ground). A typical UART terminal has two to four exposed copper pads aligned in a row; when having two pads, the TX pad is pulled electrically towards +1.8V, +3.3V or +5V and the RX pad might not be pulled to either directions; when having three pads, the additional pad is usually the GND pad and should have continuity to the ground plane of the PCB; when having four pads, the last pad is generally VCC and shows up as +1.8V, +3.3V or +5V when powered on.

By using the known properties and appearances of UART terminals it is possible to locate suspected terminals using a Multimeter and verify them by attaching a Digital Analyzer capable of analyzing UART communication. Figures showing various placements of UART terminals can be seen in Fig. 4.

UART Discovery Assistant Module. In order to assist with the detection of UART terminals on PCBs that contain a large amount of test points, a small device that generates audible beeps when probing an active UART TX line was designed. The device is composed of an ATtiny13A [9] programmable micro-controller along with auxiliary electronics with custom code that switches between three popular UART baud rates, and it beeps when encountering a threshold amount of English printable ASCII characters (characters larger than 0×20 and smaller than $0 \times 7F$). The device can be seen in Fig. 3 in Appendix. The source code for the module is publicly available in the authors' github repository [8].

2.2 Extraction of Firmware and Data

Handling Bootloader and Linux Passwords. While booting, the bootloader loads the kernel and passes over the boot arguments to the kernel. Commonly, within the boot argument is the path for a user-mode process that starts when the kernel completes booting.

After booting, the Linux kernel transfers control of the console to the user-mode process. Traditionally, after executing a list of scripts, the init process may transfer control either to the login or the shell process. When the login process is started, it requests and verifies the user's credentials and instantiates a shell process for the user to control. The login process is protected from brute-force attempts and employs a delay between consecutive password guessing attempts.

When encountered with a login request in an embedded device, a simple technique is to replace the init part of the boot argument with a path to /bin/sh or any other process that can assist with gaining access to the system. This change can be done from within the bootloader terminal, that can be accessed when the boot process begins.

Access to the bootloader is usually done by pressing some key at the first stages of boot. In certain cases the bootloader is protected by password. Since the bootloader has a very small memory footprint, it usually lacks the infrastructure for password hashing and only performs string comparison against a hard-coded password. The password string may be recovered from memory blobs obtained via out-of-band methods.

Using Physical Attacks for Bypassing Passwords or Recovering Passwords. Fault injections have a significant role in reverse engineering [32]. The usage of fault injections allows the researcher to generate a hardware fault at any given time and manipulate the underlying software. Countermeasures for fault injection attacks are under constant research [19], but they are rarely implemented in devices that are not designed to be tamper-proof. We discovered that hardware faults which cause the initialization process to fail can cause the system to fall back into a highly-privileged shell process. This can be done by disconnecting or shorting various hardware components. For example, shorting the GND and MISO pins of an SPI Flash module will cause any reads from the device to be malformed. Of course, this procedure carries the risk of damaging the device or its memory.

While side-channel attacks can also be used for recovering passwords [15], they tend to be better suited to systems with a simpler design such as ASICs or FPGAs. They are more difficult in our black-box scenario which includes a fully-featured multitasking operating system. Many other physical attacks exist for the determined researcher, some of which are even effective against tamper-resistant devices [7], but none of the devices we investigated required these methods.

Uploading Additional Tools into the Device. Embedded systems are often designed with the minimal set of features and components required for their task, as such, their software is designed similarly. Embedded Linux may contain only a small subset of the Linux utilities and features that desktop Linux users are used to having. BusyBox [38] provides many known Linux utilities in reduced size and pre-compiled for many common architectures. Using common utilities such as FTP (File Transfer Protocol utility), TFTP (Tiny File Transfer Protocol utility), Wget or NetCat can mediate data and file transfer to and from the device and over the network.

When network utilities are unavailable, data can be infiltrated through crude methods such as scripting the use of the Unix Bash Echo command for writing binary data into files. A simple python script that uses Echo for transferring files over UART is publicly available in the authors' github repository [8].

Obtaining the Firmware. Extracting a copy of the firmware and file system is an important stage for reverse engineering since analysis of the firmware can reveal secrets and vulnerabilities. Firmware analysis is further discussed in Sect. 4.

When network connection and console access are available, Flash memory MTD (Memory Technology Device) partitions can be streamed into NetCat and sent to a remote computer. A copy of the file system may also be compressed using the Tar utility and streamed using NetCat. Doing so will eliminate the need for unpacking the file system, which is not always a trivial task.

If a network connection is unavilable, memory contents can be read over UART from bootloader or Linux console. Bootloaders consoles often contain memory read/write/display primitives and can be used to slowly dump an image of the memory into the UART console. A script on the receiving end can convert the hexadecimal-displayed data into binary format; such script is publicly available in the authors' github repository [8].

When the bootloader and Linux console are inaccessible, flash memory contents can be dumped via out-of-band methods. There are several ways in which the researcher can gain access to partial or complete data belonging to the device's memory. A minimally intrusive option is connecting a logic analyzer to the pins of the memory module and recording the signals while the device is booting up. Partial memory images can be extracted from the communications on the memory bus, depending on the actual addresses that were accessed during the recording. A simple script can convert the logic analyzer output to usable binary, such script is publicly available in the authors' github repository [8].

In order to gain a full and accurate image of the device memory, it is possible to desolder the memory chip and connect it to off-the-shelf memory readers such as the CH341A. If the memory module is not compatible with off-the-shelf readers, a custom reader can be built using a general purpose USB adapter such as FT2232H or a programmable micro-controller.

More advanced techniques have been proposed [13] but are outside the scope of this paper due to their costs and effort requirements.

2.3 Analyzing the Firmware

Unpacking Memory Images. Once a memory image had been obtained, it is necessary to unpack it in order to view the data it holds. The community-maintained Binwalk utility has the ability to unpack and extract most common embedded file systems, and even some proprietary file systems. When used with the '−Z' argument, Binwalk detects raw compression streams that may be hidden from default scans and is able to extract them. A collection of utilities named firmware-mod-kit [2] contains several file formats and variations that Binwalk does not support.

Brute-Forcing Passwords. One of the more interesting feats of reverse engineering is password extraction. Native Linux passwords are used by default over SSH (Secure Shell) and Telnet (Telecommunication Network) connections and in cases also for other services such as HTTP and FTP. An observation about the Mirai IoT malware is that the infection method was connecting to IoT devices over SSH/Telnet with default credentials. Many devices today have credentials that are not as trivial as 'root', 'admin' or '123456' but are still not complex enough to withstand exhaustive password search.

Linux user passwords are usually stored in the special file '/etc/passwd' or its companion '/etc/shadow' in a hashed format, using the crypt(3) [1] utility. The password hash files can be read freely by users with sufficient credentials and can also be extracted from the firmware.

crypt(3) supports several hashing algorithms, but two are the most observed in IoT devices: **Descrypt -** A DES (Data Encryption Standard) based password hashing algorithm. A modern high-end GPU (Graphical processing using) is capable of calculating over 9 * 10ˆ8 descrypt hashes per second. **Md5crypt -** An MD5 (Message-Digest Algorithm 5) based password hashing algorithm. A modern high-end GPU is capable of calculating over 10ˆ6 (Ten million) md5crypt hashes per second.

While simple passwords can be recovered using a generic password recovery tools such as John the Ripper [4], advanced password cracking can be done with Hashcat [3]. Hashcat supports advanced rules and patterns and is designed for GPU hashing. The usage of Hashcat requires more knowledge than using John the Ripper ans it is widely used for recovery of difficult passwords.

In order to perform efficient password cracking, a word-list or pattern file is required. Many patterns and word-lists are available online but none had proved effective enough against hard to guess IoT device passwords. A few observations by the authors about known and newly discovered passwords allowed the creation and sorting of a password pattern list that proved effective against IoT device passwords. The pattern generation rules consist of: up to two symbol characters; up to two three uppercase characters; any amount of digits and lowercase characters; up to 8 characters total.

Another observation was that many elements of password difficulty inversely correlates with password selection. For example: symbol characters are expensive to search and used less often than other characters; digits are easy to search and are widely used; uppercase characters are used less than lowercase characters. This allowed sorting the pattern list according to increasing difficulty levels while expecting to guess passwords in the early stages of testing the list. More on the results of password cracking in Sect. 3. A python script for generating and sorting the pattern list is publicly available in the authors' github repository [8].

Detecting Vulnerabilities Within the Firmware. As firmware images contain the operating system and code controlling the device behavior, further analysis may expose underlying vulnerabilities. While in-depth reverse engineering techniques of the firmware are beyond the scope of this paper, there are many previous researches done in this field [11,12,25,26,30].

3 Results

3.1 Devices Under Inspection

Table 1 describes 16 IoT devices that were subjected to reverse engineering. As shown in the Table, the survey included devices from many different vendors and with prices which varied by an order of magnitude. Most of the devices with the properties selected for this work contain cameras. Additionally there are two smart doorbells that are capable of streaming video, audio, initiating VOIP sessions and also opening an entry door or a gate. A smart thermostat was also analyzed. This device can control an entire household's HVAC systems. A list of all of the devices and their properties can be seen in Table 1. All of the devices contained the embedded Linux operating system.

Table 1. List of devices reverse engineered

Device ID	Device type	Manufacturer	Model	Video recording	Additional capabilities	Price (USD)
1	IP camera	Xtreamer	Cloud camera	Yes	None	84
2	IP camera	Simple home	XCS7_1001	Yes	None	54
3	IP camera	Simple home	XCS7_1002	Yes	None	47
4	IP camera	Simple home	XCS7_1003	Yes	None	142
5	IP camera	Foscam	FI9816P	Yes	None	70
6	IP camera	Foscam	C1	Yes	None	58
7	IP camera	Samsung	SNH-1011N	Yes	None	68
8	IP camera	Xiaomi	YI Dome	Yes	None	40
9	IP camera	Provision	PT-838	Yes	None	163
10	IP camera	Provision	PT-737E	Yes	None	102
11	IP camera	TP-Link	NC250	Yes	None	70
12	Baby monitor	Phillips	B120N	Yes	None	46
13	Baby monitor	Motorola	FOCUS86T	Yes	None	145
14	Doorbell	Danmini	WiFi Doorbell	Yes	Open door/gate	80
15	Doorbell	Ennio	SYWIFI002	Yes	Open door/gate	119
16	Thermostat	Ecobee	3 (golden firmware)	No	HVAC control	170

3.2 Techniques Used on Devices

Table 4 shows a sample of the devices inspected along with the properties that allow or hinder reverse engineering. In the table there are also the techniques shown effective against these devices.

3.3 Discoveries Made During the Evaluation

Login Credentials. One of the most significant steps of reverse engineering an IoT device is to identifying all of the user accounts within the device. Every device contains at least one effective account which is the root account. The root account is the most privileged account on a Unix system. The root account has the ability to carry out all facets of system administration, including adding accounts, changing user passwords, accessing the file system, and installing software. Once a hashed password is recovered and its underlying plaintext password revealed, the ability of logging into the device with root user privileges is achieved. As can be seen in Table 2, eight of the devices contained password hashed with the descrypt algorithm, while the other eight devices employed md5crypt. The selection of hashing algorithm is critical for resisting password cracking, descrypt hashing can be as much as ninety times faster than md5crypt, as described in Subsect. 2.3.

Table 2. Discovered device properties

Device ID	Similar products	Password hash type	Open services for remote access	Password complexity	Contains private keys
1	Closeli simplicam	descrypt	-	Medium	Yes
2	-	md5crypt	Telnet	Very low	-
3	-	descrypt	Telnet	Low	-
4	Tenvis TH692	md5crypt	Telnet	Low	-
5	-	md5crypt	FTP	Unknown	Yes
6	-	md5crypt	FTP	Unknown	-
7	-	md5crypt	-	Unknown	-
8	-	md5crypt	-	None	-
9	VStarcam D38	descrypt	-	Low	-
10	VStarcam C23S	descrypt	Telnet	Low	-
11	-	md5crypt	-	Very low	-
12	-	descrypt	SSH	Medium	Yes
13	-	md5crypt	-	Unknown	-
14	-	descrypt	Telnet	Very low	-
15	-	descrypt	Telnet	Very low	-
16	-	descrypt	-	Low	-

The pattern based password recovery described in Subsect. 2.3 was used against all of the extracted password hashes. Figure 5 in Appendix shows the theoretical duration of password recovery using the proposed 48,820 patterns that cover all of the password possibilities previously mentioned. The patterns were sorted in order of rising complexity. For example, the pattern for six consecutive digits contains 1,000,000 possibilities and was sorted before the pattern for five consecutive English characters that has 11,881,376 possibilities. As the figure shows, most observed non-empty passwords were recovered within the first 5,000 patterns, after testing only 5.22e+11 passwords. The theoretical bound for testing that many passwords on a strong GPU server is 2.4 min for descrypt hashes and 217 min for md5crypt. Actual password recovery can have significant overheads over the theoretical bounds.

Eleven non-empty passwords were recovered, one device contained an empty password. Four passwords were not yet recovered to the time of writing this paper and are expected to be revealed within several weeks. Table 2 shows password complexities that varied between very low complexity (e.g. “abcd”) to medium complexity (e.g. “AbC123de”), undiscovered passwords were given the complexity rating “Unknown”. All the discovered passwords were verified as the credentials in multiple devices of the same model. Two devices made by the same manufacturer were discovered to have the same passwords but different hash values due to random salt.

Remote Access. A simple port scan using Nmap [18] revealed that many of the tested devices have administration services bound to open ports such as SSH or Telnet, which allows a remote access. Remote access allows a user to log-in to a device as an authorized user without being in the proximity of the device, depending on the network topology. Six of the devices maintain a Telnet service, one device has an accessible SSH port and two devices allow communication to open FTP ports as can be seen in Table 2. Although some of the devices do not allow communication through an administration port, by accessing the UART console it is possible to set up network services performing the desired functions.

WiFi Credentials. IoT devices must be connected to the internet in order to function properly. In order to maintain wireless connection persistency across reboots and power shortages, a configuration file that holds the WiFi credentials is located in all of the tested devices. The configuration file is located in the mounted file system, usually under the “config” or the “NetworkManager” paths, and contains all of the WiFi settings, including the SSID (Service Set Identifier) and non-encrypted password. Retrieval of the correct file from an extracted file system can be done simply by searching for relevant keywords.

Embedded Private Keys. A private key is an object that is used by an encryption algorithm for encrypting and decrypting messages and plays an important role in asymmetric cryptography. In three of the devices a hard-coded private key used for secure communication were found, as shown in Table 2. With the private key exposed, secure communication may be rendered insecure and exposed to violations such as man-in-the-middle attack.

Rebranded Devices. In the IoT market, a rebranded device is one where the internal design, architecture and file system are purchased from one manufacturer, and cosmetic modifications link the device to a new brand and manufacturer. Identifying rebranded devices means that discovered private keys, hashed passwords, account credentials and even the application vulnerabilities may be identical across several devices. Four devices inspected were found to share a non-trivial password or hashed password with products from different manufactures, strongly implying a similarity between them. The devices were found using a simple web search for the passwords and hashes and encountering forum posts that specify hashes and passwords of other devices.

4 Analysis

The techniques that were shown in Sect. 2 may be used for both malicious and benign activities. This section serves to demonstrate and discuss some of the possibilities that emerge from making the reverse engineering process more generic and streamlined, and considering the results seen in the last section.

4.1 Extension of Existing Attacks into New Platforms

Creation of a Personalized Mirai Botnet with Increased Capabilities. The infamous Mirai botnet had gained publicity after it was used against several online web sites. After witnessing a large-scale DDoS (Distributed Denial of Service) attack on KrebsOnSecurity.com, Martin McKeay, Akamai's senior security advocate was quoted saying "Someone has a botnet with capabilities we haven't seen before. We looked at the traffic coming from the attacking systems, and they weren't just from one region of the world or from a small subset of networks—they were everywhere." [23]. Mirai malware infects IoT devices with an open Telnet port and default login credentials and add them to the attacker's botnet army. The source code for Mirai was leaked to the internet on and can be modified and used by anyone who desires [5]. By using the reverse engineering process we were able to extract new and previously unknown Telnet and SSH credentials belonging to several IoT devices that were never a part of the Mirai botnet. In order to create a customized version of the Mirai botnet, the source code was modified by adding the new passwords to the malware's source code. After building an isolated network and infecting it with the modified Mirai botnet. The bot activity over the network was monitored and the infection could be seen spreading to the IoT devices that were added to the network.

An interesting example case is that after extracting the login credentials of the ProVision PT-838 security camera, the modified botnet was able to successfully connect to the ProVision PT-737E security camera due to the shared credentials between the cameras of the same manufacturer. The aforementioned process allows the number of devices that are vulnerable to Mirai to be extended.

Remote Access to IoT Devices by Unauthorized Parties. Remote connection to an IoT device, via Telnet or SSH, can be performed not only by malware but also be used as an easy and quick way for an attacker to gain control over a device remotely. The Philips In.Sight Wireless HD Baby Monitor (B120N/10) was designed to allow parents to watch, listen and talk to their newborn [33]. During the reverse engineering process several critical engineering faults that allows an outsider to use this device were discovered. Credentials were revealed that allows anyone to connect through the open SSH port in all Philips In.Sight B120N monitors. Additionally, SSL private keys that allow an attacker to perform a man-in-the-middle attacks on device communication were discovered. Furthermore, as shown in the previous section, after gaining access to an IoT device the attacker can extract sensitive information about the device and its owner such as WiFi credentials.

Execution of Arbitrary Code on IoT Devices. During the reverse engineering process, software is often uploaded into the device in various ways. The ability to upload software and even have it maintain persistency after restarts has a great implication on device security. Since it was shown how to gain complete device control when physical access is available, physical access to a device can be used to modify the device's behavior even after the device is no longer in proximity.

4.2 Possible Theoretical Attacks

Discovery of New Vulnerabilities. By using the black-box reverse engineering process, an attacker with the possession of an unknown device (e.g. a security camera with no identification markings printed on it) that was obtained from a public area may extract crucial or sensitive information. While analyzing the results we found out that many IoT devices had old OS or firmware version that was outdated, when many issues were fixed in later versions. After learning about the firmware or OS version, the attacker can search the internet for known vulnerabilities or even find this information in the release notes of more updated versions. Furthermore, after obtaining the firmware the attacker can scan the software for security holes using static analysis methods [12,14].

Extraction of Secrets from Outdoor IoT Devices. Many IoT devices are marketed for an outdoor installation (e.g. security cameras, smart doorbells etc.). These products are mounted outside or in large halls and can be accessed by strangers. For example, the Ennio Doorbell (SYWIFI002) contains a camera, microphone and speaker in order to monitor and control an entrance and can also be wired to a door or a gate for remote unlocking. The doorbell is usually installed outside and may be accessed by a stranger. A direct result of the device's accessibility is the ability of an attacker to physically modify sabotage the device. However, it is not just the device that may be affected, secrets may be extracted from the device giving the attacker access to the whole network.

Supply Chain Attacks. Malicious activity can also be performed as a part of the supply chain. An untrustworthy seller or courier can reverse engineer a device without having any previous knowledge about the it and perform modifications to the device. The recipient of an IoT device may use it without knowing it was tampered with, perhaps even equipped with a backdoor, or some other malware.

4.3 Constructive Uses to the Reverse Engineering Process

There are uses of reverse engineering that can benefit the owner. Lower-end products are often received with insufficient information about the hardware or software inside. A concerned customer can use the described process and discover properties of the device she bought. If the device is rebranded she could search the internet for the similar devices by well other vendors. The consumer gains the ability to learn about the device's vulnerabilities and perhaps even upgrade the firmware and secure the device. This process can be performed on many types of IoT devices and may also assist with products that no longer have support.

Learning about the device's software and hardware can not only help the customer identify their product, but also allows her to customize it to her own needs. After gathering the desired information the owner can manipulate the firmware or configuration. She can also develop her own system that will operate the device and even add missing functionality. Modification of stock devices can also be used to hinder censorship and other information blocking instruments.

5 Discussion

The IoT market is evolving and so does the competition among the vendors for being the first to create better and cheaper devices. This pressure may affect the product's design and lead to devices with critical security issues being released. Time is not the only obstacle for creating a secure product; as competition drives the prices down, the production process must be become cheaper. Employing penetration testers and security analysts may be very expensive, while the hardware engineers that built the product might lack knowledge of cyber security. This trade-off between money and security is usually inclined towards cheaper but less safe products. The reverse engineering process empowers consumers and researchers with abilities to discover important details about devices available in the market and benchmark their security.

5.1 Recommendations for Implementers

The results and analysis shown in this paper support several recommendations for better securing IoT devices.

1. Disable UART ports or remove their terminals from the board design. If a UART port is required, it can be set up as read-only.
2. If a UART port is required and must be write-enabled, protect UART ports in a similar fashion to JTAG protection [34].
3. Use unique strong passwords for every single device hashed with a strong hashing algorithm. Passwords must be user replaceable in a convenient way.
4. If possible, encrypt all of the device's writable memory. Otherwise, encrypt all sensitive data stored on the device.

5.2 Conclusion

The increase in IoT technology popularity holds many benefits but on the other hand this surge of new, innovative and cheap devices reveals complex security and privacy challenges. Vulnerabilities and design flaws in innocent IoT devices are an opening for an adversary to exploit and misuse. As shown in Sect. 4, an attacker that gains remote or physical access to an IoT device may snoop on the owner's personal or sensitive information and even use the device's capabilities for her own desire. The evolution of cyber crime didn't pass over the IoT and in the last years we are witnessing new types of cyber attacks that involve IoT devices. Accessibility of the black-box reverse engineering process may accelerate the attacker's work and introduce new IoT cyber threats.

Appendix

Table 3. A list of hardware and software tools used

1	Screwdrivers and plastic spudgers including common and uncommon drive bits such as Philips, Torx, Security Torx and various star configurations
2	BK 2712 Multimeter
3	FTDI FT232R USB UART interface module
4	Saleae Logic Pro 8 logic analyzer with the Logic 1.2.12 software
5	CH341A USB EEPROM and Flash memory programmer module with software version 1.29
6	Intel i7-4790 desktop PC running Windows 10 operating system and Ubuntu 16.04.4 on a virtual machine
7	Intel i7-6900K server with four Titan X (Pascal) Nvidia GPUs running Ubuntu 16.04.2 LTS operating system with Nvidia driver version 375.66
8	John The Ripper 1.8.0 CPU password cracking software
9	Hashcat 3.6.0 multiple architecture password recovery software
10	Binwalk Firmware Analysis Tool - latest version pulled from github repository on 30/07/2017 and compiled locally, including all dependencies
11	Firmware-mod-kit - latest version pulled from github repository on 30/07/2017 and compiled locally

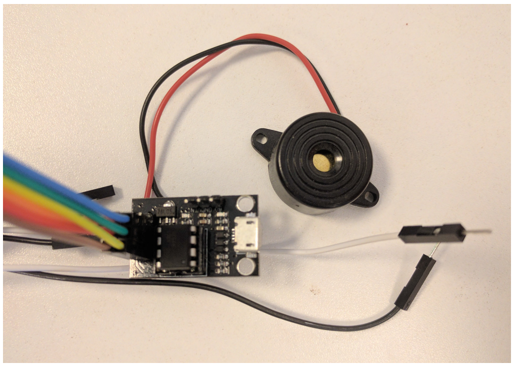


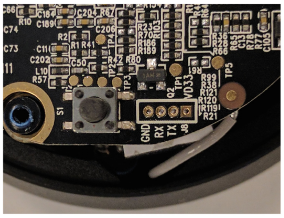
Fig. 3. UART discovery assistant module

Table 4. Inspected devices and the techniques effective on them

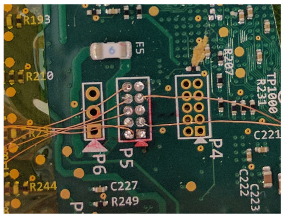
Device ID	UART location[a]	Bootloader password	Terminal password	Terminal password bypass technique	Data extraction technique
2	Marked pads	No	Yes	Shorted memory caused fallback	Used Wget to download NetCat
5	Unmarked pads[a]	No	No	-	Physically read the on-board flash
8	Unmarked pads[a]	No	No	-	Used "echo" to transfer NetCat over UART
10	Unmarked pads[a]	Yes[b]	Yes	Set bootcmd in bootloader	Used NetCat
11	Unmarked pads[a]	No	Yes	Trivial password	Used Wget to download NetCat
12	Marked pads	No	Yes	Set bootcmd in bootloader	Used NetCat
15	Unmarked pads[a]	No	Yes	Trivial password	Used TFTP to download NetCat
16	Unmarked pads[a]	No	No	-	Used NetCat

[a] Unmarked pads were discovered by inspection of the PCB assisted with the UART discovery assistant module Fig. 3.

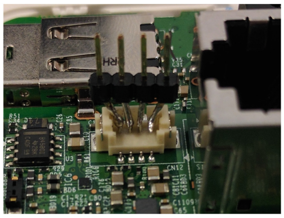
[b] Bootloader password was recovered using a logic analyzer that sniffs communication on the memory bus.



(a) UART terminals with marking inside Xtreamer Cloud Camera



(b) Wires soldered to a header pads that includes UART connections inside the Ecobee 3 Smart Thermostat



(c) Male pin header soldered on top of UART socket inside Samsung SNH-1011N Smart Camera

Fig. 4. Examples of UART terminals

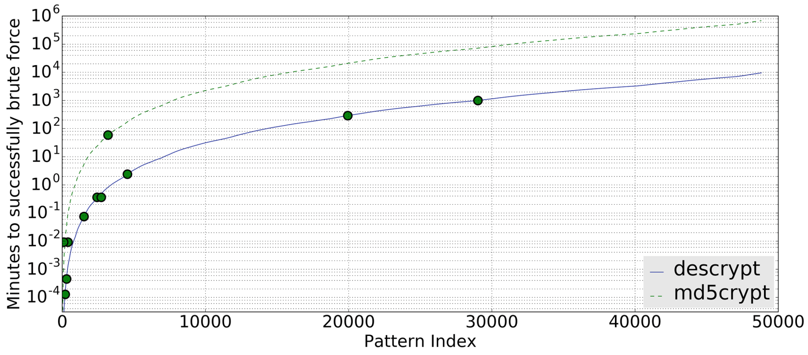


Fig. 5. Password recovery duration using the GPU server described in Table 3. Each marking on the graph is a successfully recovered password belonging to a device inspected.

References

1. crypt(3) Man Page: Linux Programmer's Manual. http://man7.org/linux/man-pages/man3/crypt.3.html
2. Firmware-mod-kit Github Repository. https://github.com/mirror/firmware-mod-kit
3. Hashcat Password Recovery Tool. https://hashcat.net/
4. John the Ripper Password Cracker. http://www.openwall.com/john/
5. Mirai Github Repository. https://github.com/jgamblin/Mirai-Source-Code
6. Alqassem, I., Svetinovic, D.: A taxonomy of security and privacy requirements for the internet of things (IoT). In: 2014 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2014, Selangor Darul Ehsan, Malaysia, 9–12 December 2014, pp. 1244–1248. IEEE (2014). https://doi.org/10.1109/IEEM.2014.7058837

7. Anderson, R., Kuhn, M.: Low cost attacks on tamper resistant devices. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 125–136. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0028165
8. Anonymous: The author's github repository. Details omitted for anonymous submission (2017)
9. Atmel Corporation: ATtiny13A Datasheet, May 2012. http://www.atmel.com/images/doc8126.pdf
10. Bodenheim, R., Butts, J., Dunlap, S., Mullins, B.E.: Evaluation of the ability of the Shodan search engine to identify internet-facing industrial control devices. IJCIP **7**(2), 114–123 (2014). https://doi.org/10.1016/j.ijcip.2014.03.001
11. Chen, D.D., Woo, M., Brumley, D., Egele, M.: Towards automated dynamic analysis for Linux-based embedded firmware. In: NDSS (2016)
12. Costin, A., Zaddach, J., Francillon, A., Balzarotti, D.: A large-scale analysis of the security of embedded firmwares. In: Fu, K., Jung, J. (eds.) Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014, pp. 95–110. USENIX Association (2014). https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin
13. Courbon, F., Skorobogatov, S., Woods, C.: Reverse engineering flash EEPROM memories using scanning electron microscopy. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 57–72. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_4
14. Cui, A., Costello, M., Stolfo, S.J.: When firmware modifications attack: a case study of embedded exploitation. In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, 24–27 February 2013. The Internet Society (2013). http://internetsociety.org/doc/when-firmware-modifications-attack-case-study-embedded-exploitation
15. DaRolt, J., Das, A., Natale, G.D., Flottes, M., Rouzeyre, B., Verbauwhede, I.: Test versus security: past and present. IEEE Trans. Emerging Topics Comput. **2**(1), 50–62 (2014). https://doi.org/10.1109/TETC.2014.2304492
16. Davis, R., Merriam, N., Tracey, N.: How embedded applications using an RTOS can stay within on-chip memory limits. In: 12th EuroMicro Conference on Real-Time Systems, pp. 71–77 (2000)
17. Gartner: Gartner says 4.9 Billion Connected "Things" will be in Use in 2015. Gartner.com (2014). http://www.gartner.com/newsroom/id/2905717
18. Gordon Lyon: Nmap Security Scanner. https://nmap.org/
19. Goubet, L., Heydemann, K., Encrenaz, E., De Keulenaer, R.: Efficient design and evaluation of countermeasures against fault attacks using formal verification. In: Homma, N., Medwed, M. (eds.) CARDIS 2015. LNCS, vol. 9514, pp. 177–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31271-2_11
20. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013). https://doi.org/10.1016/j.future.2013.01.010
21. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM **52**(5), 91–98 (2009). http://doi.acm.org/10.1145/1506409.1506429
22. Hollabaugh, C.: Embedded Linux: Hardware, Software, and Interfacing. Addison-Wesley, Boston (2002)
23. Krebs, B.: Krebsonsecurity Hit with Record DDoS. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

24. Lanet, J.-L., Bouffard, G., Lamrani, R., Chakra, R., Mestiri, A., Monsif, M., Fandi, A.: Memory forensics of a java card dump. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 3–17. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16763-3_1
25. Ling, Z., Luo, J., Xu, Y., Gao, C., Wu, K., Fu, X.: Security vulnerabilities of internet of things: a case study of the smart plug system. IEEE Internet Things J. **4**, 1899–1909 (2017)
26. Liu, M., Zhang, Y., Li, J., Shu, J., Gu, D.: Security analysis of vendor customized code in firmware of embedded device. In: Deng, R., Weng, J., Ren, K., Yegneswaran, V. (eds.) SecureComm 2016. LNICST, vol. 198, pp. 722–739. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59608-2_40
27. Lund, D., MacGillivray, C., Turner, V., Morales, M.: Worldwide and regional internet of things (IoT) 2014–2020 forecast: a virtuous circle of proven value and demand. International Data Corporation (IDC), Technical report (2014)
28. Mahmoud, R., Yousuf, T., Aloul, F.A., Zualkernan, I.A.: Internet of Things (IoT) security: current status, challenges and prospective measures. In: 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015, London, United Kingdom, 14–16 December 2015, pp. 336–341. IEEE (2015). https://doi.org/10.1109/ICITST.2015.7412116
29. Nest Labs: Nest Learning Smart Thermostat. https://nest.com/thermostat/meet-nest-thermostat/
30. Obermaier, J., Hutle, M.: Analyzing the security and privacy of cloud-based video surveillance systems. In: Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, pp. 22–28. ACM (2016)
31. Patton, M.W., Gross, E., Chinn, R., Forbis, S., Walker, L., Chen, H.: Uninvited connections: a study of vulnerable devices on the Internet of Things (IoT). In: IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014, The Hague, The Netherlands, 24–26 September 2014, pp. 232–235. IEEE (2014). https://doi.org/10.1109/JISIC.2014.43
32. San Pedro, M., Soos, M., Guilley, S.: FIRE: fault injection for reverse engineering. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 280–293. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21040-2_20
33. Philips: Philips In.Sight Wireless HD Baby Monitor. http://www.philips.co.uk/c-p/B120N_10/in.sight-wireless-hd-baby-monitor/overview
34. Rosenfeld, K., Karri, R.: Attacks and defenses for JTAG. IEEE Design Test Comput. **27**(1), 36–47 (2010). https://doi.org/10.1109/MDT.2010.9
35. Shodan: Shodan is the world's first search engine for internet-connected devices. https://www.shodan.io/
36. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in internet of things: the road ahead. Comput. Netw. **76**, 146–164 (2015). https://doi.org/10.1016/j.comnet.2014.11.008
37. Tellez, M., El-Tawab, S., Heydari, H.M.: Improving the security of wireless sensor networks in an IoT environmental monitoring system. In: Systems and Information Engineering Design Symposium (SIEDS), pp. 72–77. IEEE (2016)
38. Vlasenko, D.: BusyBox: The Swiss Army Knife of Embedded Linux. https://busybox.net/

39. Yu, T., Sekar, V., Seshan, S., Agarwal, Y., Xu, C.: Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things. In: de Oliveira, J., Smith, J., Argyraki, K.J., Levis, P. (eds.) Proceedings of the 14th ACM Workshop on Hot Topics in Networks, Philadelphia, PA, USA, 16–17 November 2015, pp. 5:1–5:7. ACM (2015). http://doi.acm.org/10.1145/2834050.2834095
40. Zhang, Z., Cho, M.C.Y., Wang, C., Hsu, C., Chen, C.K., Shieh, S.: IoT security: ongoing challenges and research opportunities. In: 7th IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2014, Matsue, Japan, 17–19 November 2014, pp. 230–234. IEEE Computer Society (2014). https://doi.org/10.1109/SOCA.2014.58

Optimal First-Order Boolean Masking for Embedded IoT Devices

Alex Biryukov, Daniel Dinu(✉), Yann Le Corre, and Aleksei Udovenko

SnT and CSC, University of Luxembourg, Luxembourg City, Luxembourg
{alex.biryukov,daniel.dinu,yann.lecorre,aleksei.udovenko}@uni.lu

Abstract. Boolean masking is an effective side-channel countermeasure that consists in splitting each sensitive variable into two or more shares which are carefully manipulated to avoid leakage of the sensitive variable. The best known expressions for Boolean masking of bitwise operations are relatively compact, but even a small improvement of these expressions can significantly reduce the performance penalty of more complex masked operations such as modular addition on Boolean shares or of masked ciphers. In this paper, we present and evaluate new secure expressions for performing bitwise operations on Boolean shares. To this end, we describe an algorithm for efficient search of expressions that have an optimal cost in number of elementary operations. We show that bitwise AND and OR on Boolean shares can be performed using less instructions than the best known expressions. More importantly, our expressions do no require additional random values as the best known expressions do. We apply our new expressions to the masked addition/subtraction on Boolean shares based on the Kogge-Stone adder and we report an improvement of the execution time between 14% and 19%. Then, we compare the efficiency of first-order masked implementations of three lightweight block ciphers on an ARM Cortex-M3 to determine which design strategies are most suitable for efficient masking. All our masked implementations passed the t-test evaluation and thus are deemed secure against first-order side-channel attacks.

Keywords: Boolean masking · Side-channel attack · IoT
Embedded device

1 Introduction

The Internet of Things (IoT) is one of the technical revolutions of our time, with many IoT devices being deployed every day to create a global network of smart objects. According to Gartner, 8.4 billion connected things will be in use worldwide by the end of 2017 [14]. From 2018 onwards, Gartner forecasts that devices such as those targeted at smart buildings (LED lighting, HVAC, and physical security systems) will have the biggest market share [14]. In light of the very recent security vulnerabilities [8,24] discovered in such devices, immediate action is required to prevent large-scale security incidents similar to the Mirai botnet [1].

The attack surface of IoT devices is considerably larger than the attack surface of classical Internet-connected systems due to the various use cases these gadgets, sensors, and actuators are built for. Most of the IoT systems are characterized by low physical security, with devices being deployed in easily accessible places. As a consequence, attack vectors that exploit these weaknesses came to light. Side-channel attacks, such as EM and power analysis attacks, fall in this category of attack vectors that require physical proximity to the target system. If the target system uses an unprotected implementation of a cryptographic algorithm, the adversary can determine the secret key used by the system from the leakage generated during the execution of the algorithm. Hence, countermeasures against side-channel attacks are mandatory for the security of IoT devices.

There are two main categories of countermeasures against side-channel attacks: masking and hiding [18]. One of the main advantages of masking over hiding is that the security of masking schemes can be proved under certain assumptions on the device leakage model and the attacker capabilities [17]. However, if the masking scheme is not correctly implemented, the implementation can leak and therefore it is not secure against side-channel attacks [4,21].

Boolean masking is one of the most widely used masking schemes. An $(n-1)$-th order Boolean masking scheme with $n \geq 2$ is based on the principle of secret sharing, splitting each variable x into at least n shares x_i such that $x = x_1 \oplus x_2 \oplus \ldots x_n$. Then, the protected algorithm processes the shares x_i in such a way that no information about the sensitive value x can be learned by an adversary which can probe up to $n-1$ wires. Yet, an $(n-1)$-th order masking scheme can be broken with an n-th order attack. The complexity of a such an attack grows exponentially with the number of shares since the attacker has to combine n points to reconstruct the leakage of the sensitive variable [7].

There are two main requirements an implementation of a cryptographic algorithm to be deployed in the IoT has to satisfy. On the one hand, the implementation must be *lightweight* (i.e. consume few resources) because of the limited computational resources of embedded devices for the IoT. On the other hand, the implementation must be secure against side-channel attacks given the attack surface specific to the IoT. Most implementations of the existing lightweight ciphers do not satisfy the second requirement, either because the cipher was not designed to facilitate masking, or because the best existing masking schemes add significant performance penalties to the unprotected implementation of the cipher. Therefore, there is a need for more efficient masking schemes. Any improvement of the existing masking schemes brings us closer to the goal of a secure IoT.

Conceptually, Boolean masking of a block cipher is done by replacing each unprotected operation by its masked counterpart. The most common operations used by lightweight block ciphers are logical operations (NOT, AND, OR, XOR), rotations, and modular addition/subtraction. Masked NOT is equivalent to the negation of a single share, while masked XOR and rotations can be realized by simply applying the operation to each pair of shares independently. To our knowledge, the best known expression for first-order Boolean masking of bitwise AND is based on the Trichina AND gate [28]. The same expression of the masked AND was latter used by Coron *et al.* in their algorithm for masked addition on Boolean shares [9]. Since there is almost no reference to a masked OR expression in the literature, one might try to derive such an expression by applying De Morgan's laws to the masked AND expression. Hence, we consider the derived expression using De Morgan's laws as the best known expression for masked OR, although Baek and Noh [3] proposed a masked OR gate that requires six elementary operations and no random value. The best known masked expressions of AND and OR require an additional random value.

Gross [16] showed how to design and implement a general purpose arithmetic logic unit using provably secure threshold implementations. He used an exhaustive search to find the best expression for efficient masking of AND and OR in hardware using three shares.

The best known algorithm for secure addition on Boolean shares is based on the Kogge-Stone adder [9]. Won and Han [29] presented a method to improve the execution time of this algorithm when the register size of the target microcontroller is smaller than the operand size. Schneider *et al.* described efficient hardware modules that perform addition on Boolean shares [26].

In this paper, we study the efficiency of Boolean masking for embedded IoT devices. Although our work is not limited to a specific microprocessor architecture, we evaluate our implementations on a 32-bit ARM Cortex-M3 since these microcontrollers are widely used for IoT applications [22].

Our Contributions. Firstly, we present an algorithm for efficient search of Boolean masking expressions (Sect. 2). Thanks to several algorithmic optimizations, the search is very fast. As a second contribution, we propose concrete expressions for Boolean masking of the AND and OR operations (Sect. 2.2). Our expressions use fewer elementary operations than the best known expressions in the literature. At the same time, unlike the best known expressions, our expressions for secure AND and OR on Boolean shares do no require any randomness. Thirdly, we improve the Kogge-Stone algorithm for addition/subtraction on Boolean shares [9] by using our masking expressions and by processing the shares in a clever way that does not require any randomness (Sect. 3.1). When implemented on an ARM Cortex-M3 processor (Sect. 4.1), the addition/subtraction of 32-bit values using the new algorithm is between 14% and 19% faster than similar implementations using the original algorithm [9]. Finally, we use our Boolean masking expressions to write first-order masked implementations of three lightweight block ciphers, namely Simon, Speck, and RECTANGLE (Sect. 4.2). By comparing the performance figures of the masked and unmasked implementations of the three ciphers, we learn which design strategies facilitate efficient masked implementations.

All software presented in this paper will is placed in the public domain[1] to support reproducibility of results and to maximize reusability.

[1] https://github.com/cryptolu/ofom.

2 Search Algorithm

In this section we describe our algorithm for searching optimal masking expressions. We start with a high-level description, then we dive into details. We give a pseudocode of the full search algorithm. Finally, we provide the optimal expressions we found using the algorithm.

2.1 The Algorithm

The algorithm takes as input a set of variables representing the input shares, a set of sensitive functions (i.e. functions that combine the shares of a sensitive value and thus leak the sensitive value) and a target function; it outputs the shortest sequences of operations required to compute the Boolean shares of the target function. A *sequence* is represented as a tuple that contains all intermediate terms of an expression in the order they are required to compute the expression. Moreover, any single intermediate value computed in a sequence does not leak any information about the sensitive functions. Multiple intermediate values may be considered for higher-order masking.

At its core, the algorithm performs a breadth-first search with several cut-off conditions. The functions are represented by their truth tables and are stored as integers for efficiency reasons. Initially, there is only an empty sequence available. The algorithm expands it into multiple sequences of length one. Afterwards, all sequences of length one are expanded into sequences of length two and so on, until the algorithm finds a sequence for which some of its intermediate values are the Boolean shares of the target function. The search is illustrated in Fig. 1.

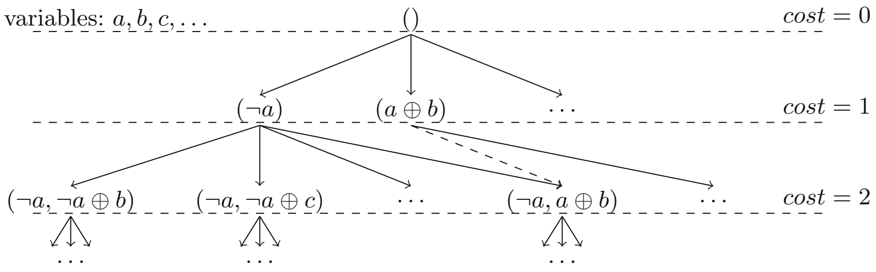


Fig. 1. High-level scheme of the search algorithm.

The core function of the algorithm is the extension step, where a given sequence is extended with one operation using all possible combinations to get the *extended sequences*. The new operation may take as its inputs either variables of the input shares or intermediate values computed in the current sequence. The cut-off conditions used to reduce the complexity of the algorithm are described next.

1. **Leakage test.** For each new function, the algorithm checks if the new function leaks information about the sensitive functions. If the function is leaking, the extended sequence is omitted and not considered anymore. In this way, the search space is effectively reduced only to non-leaking functions. This significantly improves the efficiency of the search algorithm and guarantees that the resulting sequences do not leak.
   The check is very efficient since it consists of performing a few bitwise operations on the truth tables and computing the Hamming weight. For example, a non-constant function f leaks information about function k if and only if
   $$\frac{\mathsf{HW}(k \wedge f)}{\mathsf{HW}(f)} \neq \frac{\mathsf{HW}(k \wedge \neg f)}{\mathsf{HW}(\neg f)},$$
   where $\mathsf{HW}(g)$ denotes the Hamming weight of the truth table of function g.
2. **Ignoring the order of operations.** For any sequence of operations, we exclude all other sequences that compute the same *set* of intermediate functions. Indeed, such sequences are *equivalent* in terms of extension, because an extension depends only on the *set* of intermediate functions but not on the way they are computed. From each such equivalence class we keep the representative that the algorithm reaches first. Note that this condition also excludes sequences that compute some function multiple times.
   Due to this cut-off, we may miss some optimal sequences. More precisely, from each such equivalence class we will preserve only one representative sequence. Since we do not allow to compute the same function twice in a sequence, the representative will have the shortest length. Hence, the algorithm will find at least one sequence of optimal length. If all optimal sequences are required, the full equivalence class can be recovered from its representative.
3. **Exploiting the symmetries of shares.** The Boolean shares are naturally symmetric: permuting the shares of a masked value does not change the masked value. Moreover, when we are masking a symmetric operation (e.g. AND, OR), swapping the input operands in the whole circuit will still give a correct circuit. We can exploit these symmetries and explore only one of the equivalent sequences, similarly to the cut-off condition 2. Again, the same reasoning shows that we do not miss an optimal sequence.

The pseudocode of the search algorithm is given in Algorithm 1. For simplicity and efficiency reasons, the algorithm keeps track only of computed functions but not of the applied operations. After the optimal function sequences are found, it is easy to recover the corresponding expressions. This approach also reduces the memory usage of the algorithm.

Optimality. We would like to stress that the algorithm is designed to find *optimal* expressions, not just to improve the existing ones. It is easy to show that the algorithm yields optimal expressions when it reaches the optimal cost level. Any optimal expression has (at least one) non-leaking sequence of operations to compute it. The algorithm explores all sequences except those omitted during

Algorithm 1. Searching for the Optimal Shares

```
Require:
    target function $t : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$;    ▷ e.g. $t(x_0, x_1, x_2, x_3) = (x_0 \oplus x_1) \wedge (x_2 \oplus x_3)$
    number of output shares $m$;
    set of sensitive functions $K = \{k_i\}, k_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$;    ▷ e.g. $(x_0 \oplus x_1), (x_2 \oplus x_3), t$
    set of allowed operations $O = \{op_i\}, op_i : \mathbb{F}_2^2 \rightarrow \mathbb{F}_2$
Ensure:
    set of $m$ functions $S = \{s_i\}, s_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ such that $\bigoplus_{s_i \in S} s_i = t$;
    optimal circuits computing all $s_i$ without leaking information about the value of
    any $k_i$;
 1: $seqs_0 \leftarrow \{()\}$    ▷ empty sequence
 2: $visited \leftarrow \{()\}$
 3: for $cost := 1$ to $\infty$ do
 4:     $seqs_{cost} \leftarrow \{\}$
 5:     for all $seq \in seqs_{cost-1}$ do
 6:         for all $seq' \in$ Extensions($seq$) do
 7:             if ShouldKeepSequence($seq'$) then
 8:                 $seqs_{cost} \leftarrow seqs_{cost} \cup \{seq'\}$
 9:                 if ContainsShares($seq'$, $t$) then    ▷ impl. omitted for brevity
10:                     yield $seq'$
11:                 end if
12:             end if
13:             $visited \leftarrow visited \cup \{set(seq')\}$
14:         end for
15:     end for
16: end for
17: function Extensions($seq$)
18:     for all $a, b \in seq \cup \{x_0, x_1, \ldots, x_{n-1}\}$ do
19:         for all $op \in O$ do
20:             yield $seq || op(a, b)$
21:         end for
22:     end for
23: end function
24: function ShouldKeepSequence($seq$)
25:     if Leaks($last(seq), K$) then    ▷ Cut-off [1]
26:         return False
27:     end if
28:     $seq \leftarrow$ SymmetryRepresentative($seq$)    ▷ impl. omitted for brevity; Cut-off [3]
29:     if $set(seq) \in visited$ then    ▷ Cut-off [2], [3]
30:         return False
31:     end if
32:     return True
33: end function
34: function Leaks($f, K$)
35:     for all $k \in K$ do
36:         if $\mathsf{HW}(k \wedge \neg f)\mathsf{HW}(f) \neq \mathsf{HW}(k \wedge f)\mathsf{HW}(\neg f)$ then    ▷ fraction equality check
37:             return True
38:         end if
39:     end for
40:     return False
41: end function
```

cut-offs. The first cut-off condition reduces the search to non-leaking sequences. It is easy to see that the effect of the other two cut-off conditions can be jointly seen as collapsing large equivalence classes into single representatives. Due to the breadth-first nature of the algorithm, the representative chosen by the algorithm has minimum cost.

Note that it is important to search for sequences of operations instead of expressions. Expressions may contain repeating terms and this reduces the computational cost. Moreover, this effect spreads over the output shares as well: they also may have common terms. Because of this effect, it is unclear how long are the expressions one has to consider to find a provably optimal expression. Searching for sequences of operations solves this problem at the cost of increasing the search space. The described cut-off conditions aim to narrow this gap and bring the algorithm to feasible complexities.

Instruction Set Architecture (ISA). We distinguish between two classes of IoT devices depending on the operations supported by the instruction set architecture (ISA): *basic* and *enhanced* devices. Most IoT devices have instructions only for the following bitwise logical operations: NOT, AND, OR, and XOR. We call these architectures *basic* ISAs. In addition to these operations, the *enhanced* ISAs have dedicated instructions for other bitwise logical operations, such as AND NOT or OR NOT. For example, the instruction set of ARM Cortex-M3 includes the `bic` (AND NOT) and `orn` (OR NOT) instructions that perform two basic bitwise logical operations in a single clock cycle instead of two clock cycles. Most microcontrollers execute all logical instructions in a single clock cycle.

Leakage Model. The power consumption of most microcontrollers is proportional to the number of bits that are set in the processed sensitive value [18]. Therefore, the Hamming weight power model is a reliable method for modeling the leakage of a sensitive variable. In addition to the bit-level leakage verification performed by the search algorithm, we performed a t-test leakage assessment [15] for each valid expression returned by the algorithm to confirm the absence of any leakage.

Extension to Higher-Order Masking. Our algorithm can naturally be extended to search expressions for higher-order masking. However, further optimizations might be required to ensure that the algorithm scales well for higher values of the number of shares.

2.2 Results

We have implemented the algorithm in Python language and ran it using the fast PyPy interpreter [11]. We searched for expressions for masked AND (SecAnd) and masked OR (SecOr). For example, to search for masked AND on a basic platform we used the following inputs to the algorithm:

Table 1. Expressions, number of randoms (Rand) and number of operations (Cost) for different secure operations. Basic cost gives the number of elementary operations, while the ARM cost gives the number of instructions. Expressions in parentheses have priority and operations are executed from left to right.

Source	Operation	Expression	Rand	Cost	
				Basic	ARM
Best known	SecAnd	$z_1 = r$ $z_2 = z_1 \oplus (x_1 \wedge y_1) \oplus (x_1 \wedge y_2) \oplus (x_2 \wedge y_1) \oplus (x_2 \wedge y_2)$	1	8	8
	SecOr	$z_1 = r$ $z_2 = \neg z_1 \oplus (x_1 \wedge y_1) \oplus (x_1 \wedge \neg y_2) \oplus (\neg x_2 \wedge y_1) \oplus (\neg x_2 \wedge \neg y_2)$	1	11	10
Our	SecAnd	$z_1 = (x_1 \wedge y_1) \oplus (x_1 \vee \neg y_2)$ $z_2 = (x_2 \wedge y_1) \oplus (x_2 \vee \neg y_2)$	0	7	6
	SecOr	$z_1 = (x_1 \wedge y_1) \oplus (x_1 \vee y_2)$ $z_2 = (x_2 \vee y_1) \oplus (x_2 \wedge y_2)$	0	6	6

1. target function $t(x_0, x_1, x_2, x_3) = (x_0 \oplus x_1) \wedge (x_2 \oplus x_3)$;
2. number of output shares $m = 2$;
3. set of sensitive functions $K = \{s_0, s_1, s_0 \wedge s_1, \neg s_0 \wedge s_1, s_0 \wedge \neg s_1, \neg s_0 \wedge \neg s_1, \}$, where $s_0 = x_0 \oplus x_1, s_1 = x_2 \oplus x_3$;
4. set of allowed operations $O = \{\wedge, \vee, \oplus, \neg\}$.

The hardest target was the search for SecAnd limited to 6 enhanced ISA operations which took 30 min and 10 GB RAM on a laptop. The optimal expressions for masked SecOr use 6 instructions on both platforms, while optimal expressions for SecAnd have a cost of 7 on a basic device and 6 on ARM.

The optimal expressions for SecOr and SecAnd using basic instructions are unique up to symmetries of the shares, whereas for ARM there are 48 different optimal expressions for SecAnd and 50 different optimal expressions for SecOr. The unique optimal expressions for a basic architecture are actually included in the optimal expressions for the ARM architecture, which makes them universal. A comparison of these two expressions with the best known expressions in the literature is given in Table 1. Besides using less operations than the best known expressions in the literature, our optimal expressions do not require a random value. Thanks to these two properties, our expressions have a significant performance advantage over the best known ones.

3 Applications

3.1 Modular Addition and Subtraction

Coron *et al.* [9] proposed a logarithmic-time algorithm for modular addition on Boolean shares based on the Kogge-Stone adder. Their algorithm for modular addition uses the following three secure operations: SecAnd, SecXor, and SecShift. The expression of SecAnd uses 8 elementary operations, the one of SecXor needs 2 elementary operations, while SecShift can be performed using 4 elementary operations. Algorithms for all these operations are presented in [9].

Although not described in the original paper [9], the algorithm for modular subtraction can be obtained from the algorithm for modular addition on Boolean shares by making several changes. Namely, the SecShift operations from lines 7 and 15 of [9, Algorithm 6] have to be replaced by SecShiftFill (secure operation for shift to the left by n followed by OR of $2^n - 1$). Similarly, SecXor operations from lines 9 and 17 of [9, Algorithm 6] must be replaced by SecOr. These changes affect the performance of the modular subtraction algorithm since operations with a lower cost are replaced by operations with a higher cost.

Algorithm 2. Improved Kogge-Stone Masked Addition

```
Require: $x_1, x_2, y_1, y_2 \in \{0,1\}^k$ such that $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$
Ensure: $z_1, z_2$ such that $z = z_1 \oplus z_2 = (x + y) \bmod 2^k$
 1: $p_1, p_2 \leftarrow \mathsf{SecXor}(x_1, x_2, y_1, y_2)$
 2: $g_1, g_2 \leftarrow \mathsf{SecAnd}(x_1, x_2, y_1, y_2)$
 3: $g_1, g_2 \leftarrow \big((g_1 \oplus x_2) \oplus g_2, x_2\big)$
 4: $n \leftarrow \max\big(\lceil \log_2(k-1) \rceil, 1\big)$
 5: for $i := 1$ to $n - 1$ do
 6:     $h_1, h_2 \leftarrow \mathsf{SecShift}(g_1, g_2, 2^{i-1})$
 7:     $u_1, u_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
 8:     $g_1, g_2 \leftarrow \mathsf{SecXor}(g_1, g_2, u_1, u_2)$
 9:     $h_1, h_2 \leftarrow \mathsf{SecShift}(p_1, p_2, 2^{i-1})$
10:     $h_1, h_2 \leftarrow \big((h_1 \oplus x_2) \oplus h_2, x_2\big)$
11:     $p_1, p_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
12:     $p_1, p_2 \leftarrow \big((p_1 \oplus y_2) \oplus p_2, y_2\big)$
13: end for
14: $h_1, h_2 \leftarrow \mathsf{SecShift}(g_1, g_2, 2^{n-1})$
15: $u_1, u_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
16: $g_1, g_2 \leftarrow \mathsf{SecXor}(g_1, g_2, u_1, u_2)$
17: $z_1, z_2 \leftarrow \mathsf{SecXor}(y_1, y_2, x_1, x_2)$
18: $z_1, z_2 \leftarrow \big((z_1 \oplus (g_1 \ll 1)) \oplus (x_2 \ll 1), y_2\big)$
```

One can improve the algorithms for modular addition/subtraction based on the Kogge-Stone adder by simply replacing the original expressions for SecAnd and SecOr with our optimal expressions. Yet, the algorithm can be improved further by replacing the expression of the SecShift operation, which requires a random variable, by a more efficient expression that does not require any randomness. Hence, the new versions of the algorithm do not require any randomness at all. The improved algorithm for addition on Boolean shares is described in Algorithm 2, while the analogous algorithm for subtraction is presented in Algorithm 3. It is important to note that lines 3, 10, and 12 from Algorithm 2 are required to prevent composition of operations that otherwise will leak. Similarly, lines 4, 10, 12, 14, and 19 of Algorithm 3 avoid composing operations that leak.

Algorithm 3. Improved Kogge-Stone Masked Subtraction

```
Require: $x_1, x_2, y_1, y_2 \in \{0,1\}^k$ such that $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$
Ensure: $z_1, z_2$ such that $z = z_1 \oplus z_2 = (x - y) \bmod 2^k$
 1: $y_1, y_2 \leftarrow \mathsf{SecNot}(y_1, y_2)$
 2: $p_1, p_2 \leftarrow \mathsf{SecXor}(y_1, y_2, x_1, x_2)$
 3: $g_1, g_2 \leftarrow \mathsf{SecAnd}(x_1, x_2, y_1, y_2)$
 4: $g_1, g_2 \leftarrow \big((g_1 \oplus x_2) \oplus g_2, x_2\big)$
 5: $n \leftarrow \max\big(\lceil \log_2(k-1) \rceil, 1\big)$
 6: for $i := 1$ to $n-1$ do
 7:     $h_1, h_2 \leftarrow \mathsf{SecShiftFill}(g_1, g_2, 2^{i-1})$
 8:     $u_1, u_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
 9:     $g_1, g_2 \leftarrow \mathsf{SecOr}(g_1, g_2, u_1, u_2)$
10:     $g_1, g_2 \leftarrow \big((g_1 \oplus x_2) \oplus g_2, x_2\big)$
11:     $h_1, h_2 \leftarrow \mathsf{SecShift}(p_1, p_2, 2^{i-1})$
12:     $h_1, h_2 \leftarrow \big((h_1 \oplus x_2) \oplus h_2, x_2\big)$
13:     $p_1, p_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
14:     $p_1, p_2 \leftarrow \big((p_1 \oplus y_2) \oplus p_2, y_2\big)$
15: end for
16: $h_1, h_2 \leftarrow \mathsf{SecShiftFill}(g_1, g_2, 2^{n-1})$
17: $u_1, u_2 \leftarrow \mathsf{SecAnd}(p_1, p_2, h_1, h_2)$
18: $g_1, g_2 \leftarrow \mathsf{SecOr}(g_1, g_2, u_1, u_2)$
19: $g_1, g_2 \leftarrow \big((g_1 \oplus x_2) \oplus g_2, x_2\big)$
20: $z_1, z_2 \leftarrow \mathsf{SecXor}(y_1, y_2, x_1, x_2)$
21: $z_1 \leftarrow \Big(z_1 \oplus \big((g_1 \ll 1) \vee 1\big)\Big) \oplus (x_2 \ll 1)$
22: $z_2 \leftarrow y_2$
```

Table 2. Comparison of the number of instructions required to perform different secure operations.

Platform	Source	Cost					
		SecNot	SecXor	SecAnd	SecOr	SecShift	SecShiftFill
Basic	Best known	1	2	8	11	4	6
	Our	1	2	7	6	2	4
	Gain	**0**	**0**	**1**	**5**	**2**	**2**
ARM	Best known	1	2	8	10	4	6
	Our	1	2	6	6	2	4
	Gain	**0**	**0**	**2**	**4**	**2**	**2**

Cost. A comparison between the cost of the secure expressions used by the original version of the algorithm and the new expressions used by the improved version of the algorithm is provided in Table 2. Based on these values, one can compute the total cost of these algorithms for different architectures and make an estimation of their performance for different values of the operand size k (see Table 3).

Table 3. Cost and random numbers (Rand) required for Kogge-Stone addition/subtraction on Boolean shares for different values of the operand size k. Basic cost gives the number of elementary operations, while the ARM cost gives the number of instructions.

Operation	Platform	Expressions	Rand	k	$k = 8$	$k = 16$	$k = 32$	$k = 64$
SecAdd	Basic	Best known	2	$28 \cdot \log_2 k + 4$	88	116	144	172
		Our	0	$22 \cdot \log_2 k + 6$	72	94	116	138
		Gain	**2**	$6 \cdot \log_2 k - 2$	**16**	**22**	**28**	**34**
	ARM	Best known	2	$28 \cdot \log_2 k + 4$	88	116	144	172
		Our	0	$22 \cdot \log_2 k + 4$	70	92	114	136
		Gain	**2**	$6 \cdot \log_2 k$	**18**	**24**	**30**	**36**
SecSub	Basic	Best known	2	$41 \cdot \log_2 k + 4$	127	168	209	250
		Our	0	$32 \cdot \log_2 k + 6$	102	134	166	198
		Gain	**2**	$9 \cdot \log_2 k - 2$	**25**	**34**	**43**	**52**
	ARM	Best known	2	$40 \cdot \log_2 k + 4$	124	164	204	244
		Our	0	$30 \cdot \log_2 k + 6$	96	126	156	186
		Gain	**2**	$10 \cdot \log_2 k - 2$	**28**	**38**	**48**	**58**

Security. We evaluated the secure operations presented in this section, including the two improved algorithms for addition and subtraction on Boolean shares, against first-order attacks using Welch's t-test [15]. Welch's t-test is a fast and robust way to verify the soundness of a masking scheme [12,25]. To determine if there is any leakage in our first-order implementations, we used a simple tool similar to the ones described in [20,21,23]. Firstly, we validated the correctness of our tool by performing evaluations against a set of masking schemes known to be either secure or broken. Then, we carefully applied the t-test to avoid false negatives [27]. All our secure implementations passed a set of *fixed vs. random* evaluations with up to 10^6 traces using both Hamming weight and Hamming distance models for the simulated leakage. See Appendix A for more details.

3.2 Other Applications

The optimal expressions for secure computation of AND and OR can be used to mask more complex structures such as S-boxes. They can also be used to efficiently mask ciphers that use only logical bitwise operations such as Simon [6], as well as bit-sliced designs such as Noekeon [10], RECTANGLE [30], or RoadRunneR [5]. In Sect. 4, we evaluate how these expressions can be applied to unprotected implementations of several lightweight block ciphers and we determine the performance penalty of the resulting first-order protected implementations.

4 Implementations

In this section we describe our efficient implementations of several first-order secure algorithms and block ciphers. All our implementations are written in assembly language for a Cortex-M3 processor for two reasons. Firstly, we wanted to avoid accidental leakages introduced by the transformations made by the `gcc` compiler which is not optimized for masked implementations, but only for efficiency [4]. On the other hand, when coding in assembly language, the implementer has full control of the register allocation and the sequence of instructions executed by the microcontroller. Hence, she can avoid combining instructions and registers in a way that leaks [4,21]. Secondly, we wanted to get a clear picture of the performance figures of our implementations in order to conduct a fair comparison of the first-order implementations. Hence, the effort spent by a programmer on a more demanding assembly implementation is paid off in the end by a better (i.e. more secure and efficient) implementation.

In line with previous work, we do not include the cost of random number generation for the implementations that need randomness since the cost of random number generation is different from one device to the other and we want a device-independent comparison. We report the execution time and the code size for protected implementations that do not leak in the Hamming weight model. The leakage of these implementations in the Hamming distance model can be fixed with minor changes. These changes have a similar effect on the performance of the implementations based on our expressions and the implementations based on the best known expression.

4.1 Masked Addition

We implemented the original algorithms for addition and subtraction on Boolean shares as well as the improved algorithms presented in this paper. For each algorithm we wrote a straightforward implementation and an implementation that unrolls the main loop of the Kogge-Stone adder. The execution time and code size of our implementations are given in Table 4.

The improved algorithms are between 14% and 19% faster than the original ones. At the same time, the code size of the improved algorithms is between 12% and 21% smaller than the code size of the original algorithms. Unlike the original algorithms, which require two random values, the improved algorithms do not require any random value. The generation of a 32-bit random number takes between 37 cycles for a XorShift RNG [19] and 85 cycles for the built-in TRNG [2]. Hence, the improved algorithms for addition and subtraction on Boolean shares outperform the original algorithms in all categories: execution time, code size, and required randomness.

4.2 Lightweight Block Ciphers

We selected the top-3 block ciphers that use a 64-bit block from the performance evaluation conducted using the FELICS benchmarking framework [13]

Table 4. Execution time and code size for secure addition and subtraction on Boolean shares using the Kogge-Stone adder.

Impl.	Expressions	Rand	Time (cycles)		Code size (bytes)	
			Addition	Subtraction	Addition	Subtraction
Rolled	Best known	2	275	388	292	416
	Our	0	228	333	232	332
	Gain	2	**47 (17%)**	**55 (14%)**	**60 (21%)**	**84 (20%)**
Unrolled	Best known	2	203	296	544	812
	Our	0	173	241	480	692
	Gain	2	**30 (15%)**	**55 (19%)**	**64 (12%)**	**120 (15%)**

and we protected them against first-order attacks using the best known algorithms for secure operations on Boolean shares as well as the ones introduced in this paper. Besides their very lightweight software implementations, these three ciphers (Speck, Simon, and RECTANGLE) have different design strategies. Hence, they facilitate an analysis of the relationship between their design strategies and the performance figures of their masked implementations.

Speck. Speck [6] is an ARX-based family of lightweight block ciphers designed for performance in software. Nevertheless, all ciphers of this family perform very well in hardware also. Speck-64/128 refers to the version of Speck characterized by a 64-bit block, a 128-bit key, and 27 rounds. The round function of Speck-64/128 uses only bitwise XOR, addition modulo 2^{32}, and rotations:

$$R_k(x, y) = \Big(\big((x \ggg 8) \boxplus y\big) \oplus k, (y \lll 3) \oplus \big((x \ggg 8) \boxplus y\big) \oplus k\Big),$$

where x and y are the two 32-bit branches of a Feistel network.

While the unprotected implementation of Speck requires only four registers in order to process the cipher's state, the protected implementations need all 13 general-purpose registers of the Cortex-M3 microcontroller. Moreover, the rolled implementations have to save the content of a register onto the stack at the beginning of the secure addition/subtraction. The initial value of this register is recovered at the end of the addition/subtraction operation. A pair of stack operations (i.e. `push` and `pop`) adds 4 cycles to the total execution time of the algorithm.

The implementations of Speck based on the improved algorithms for modular addition and subtraction on Boolean shares are faster and use less code space than the implementations of Speck based on the original versions of the same algorithms as can be seen in Table 5. When comparing the gain of the improved expressions over the original ones for rolled and unrolled implementations, we can see that the gain in the case of rolled implementations is higher than the gain in the case of unrolled ones. For example, the gain of rolled decryption is 27%, while the gain of unrolled decryption is only 17%.

Table 5. Execution time, code size and performance penalty factor for different secure implementations of Speck-64/128. For each set of expressions (best known, our) we wrote two implementations that correspond to the two implementation strategies of the Kogge-Stone adder (KSA): rolled/unrolled KSA.

Impl./Expr.	Rand	Time (cycles)		Code size (bytes)		Penalty factor	
		Enc	Dec	Enc	Dec	Enc	Dec
Unprotected	0	318	530	44	52	1	1
Rolled KSA/best known	2	7131	11368	340	488	22.42	21.44
Rolled KSA/our	0	5686	8258	272	400	17.88	15.58
Gain (%)	2	**1445 (21%)**	**3110 (27%)**	**68 (20%)**	**88 (18%)**	**4.54**	**5.86**
Unrolled KSA/best known	2	4945	7431	588	876	15.55	14.02
Unrolled KSA/our	0	4666	6188	536	712	14.67	11.67
Gain (%)	2	**279 (6%)**	**1243 (17%)**	**52 (9%)**	**164 (19%)**	**0.87**	**2.34**

Simon. Simon [6] is a family of lightweight block ciphers designed primarily for optimal performance in hardware, but its instances perform very good in software as well. The round function of Simon uses only bitwise XOR, bitwise AND, and rotations:

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

where $f(x) = (x \lll 1) \wedge (x \lll 8) \oplus (x \lll 2)$. Simon-64/128 is the instance of Simon that processes a 64-bit block using a 128-bit key in 44 rounds.

The two protected implementations of Simon are very efficient since the operations used by the cipher can be masked with a little impact on the execution time and code size. The most costly operation is secure bitwise AND which, depending on its expression, can be evaluated using 6 or 8 instructions. The other secure operations require only 2 instructions each. The unprotected implementation of Simon needs only four registers. The first-order protected implementation based on the best known expression of AND requires ten registers, while the one based on our optimal expression of AND takes nine registers. Consequently, the gain in execution time of the implementation based on the improved expression of AND over the implementation based on the best known expression of AND is modest (i.e. 5%). Nevertheless, the gain in code size is about 25%. The results of these implementations is presented in Table 6.

Table 6. Execution time, code size and performance penalty factor for different secure implementations of Simon-64/128.

Impl./Expr.	Rand	Time (cycles)		Code size (bytes)		Penalty factor	
		Enc	Dec	Enc	Dec	Enc	Dec
Best known	1	1736	1737	152	156	1.62	1.62
Our	0	1648	1649	136	140	1.54	1.54
Gain	1	**88 (5%)**	**88 (5%)**	**16 (27%)**	**16 (25%)**	**0.08**	**0.08**

RECTANGLE. RECTANGLE [30] is a block cipher designed to facilitate lightweight and fast implementations, both in hardware and software, using bit-slicing. RECTANGLE processes a 64-bit block in 25 rounds and supports keys of 80 and 128 bits. We refer to the 128-bit version of RECTANGLE as RECTANGLE-64/128. The cipher's state is represented as a matrix of 4×16 bits. Each round of RECTANGLE uses three transformations: `AddRoundKey` (bit-wise XOR), `SubColumn` (application of a 4-bit S-box to the state columns), and `ShiftRow` (rotations of the state rows by 1, 12 and 13 bits). The S-box of RECTANGLE can be described using a sequence of 12 basic logical instructions and hence the `SubColumn` transformation can be implemented in a bit-sliced fashion.

The unprotected implementation of RECTANGLE requires seven registers for encryption and eight for decryption. The protected implementations use all available registers of the microcontroller and several pairs of stack operations (i.e. `push` and `pop`). The protected implementation based on the best known expressions uses five pairs of stack operations, while the one based on our optimal expressions uses only three pairs for encryption and four pairs for decryption. The stack operations are necessary because the protected implementations have to keep track of more intermediate variables than they can fit into the registers of the ARM microcontroller.

In summary, the implementation based on our optimal expressions uses less instructions and less stack operations compared to the implementation based on the best known expression. The performance figures given in Table 7 show that the gain in execution time is 19% for encryption and 14% for decryption.

Table 7. Execution time, code size and performance penalty factor for different secure implementations of RECTANGLE-64/128.

Impl./Expr.	Rand	Time (cycles)		Code size (bytes)		Penalty factor	
		Enc	Dec	Enc	Dec	Enc	Dec
Unprotected	0	945	994	200	160	1	1
Best known	1	3661	3422	632	444	3.87	3.44
Our	0	2584	2954	564	372	2.73	2.97
Gain	1	**1077 (19%)**	**468 (14%)**	**68 (11%)**	**72 (16%)**	**1.13**	**0.47**

Comparison. When comparing the performance results of the unprotected implementations of the three ciphers (see Fig. 2), one can see that Speck is the fastest, followed by RECTANGLE and Simon; each of them takes about three times more cycles than Speck. On the other hand, when comparing first-order protected implementations, the implementations of Simon and RECTANGLE take the lead, while the implementation of Speck is the last one. The performance degradation of the first-order protected implementation of Speck stems from the high overhead associated with masking modular addition (see Table 4). The protected implementation of RECTANGLE is roughly three times slower than its unprotected implementation. Finally, the protected implementation of Simon is only 54% slower than its unprotected implementation.

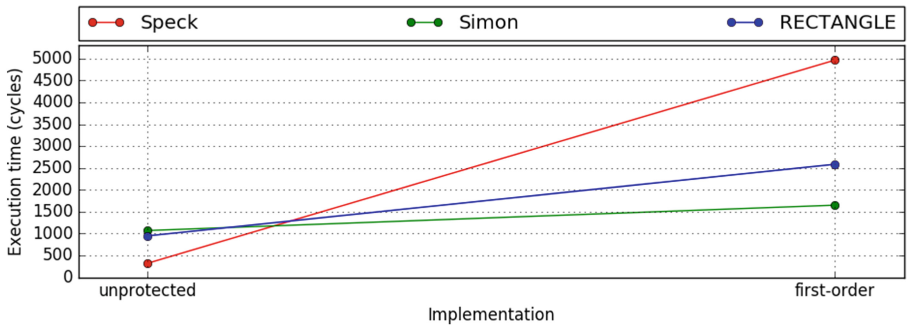


Fig. 2. Performance comparison of unprotected and first-order protected implementations of Speck, Simon, and RECTANGLE.

From this analysis, we learn that lightweight block ciphers that are very fast in unprotected software implementations (e.g. Speck), might not be the most suitable ones for first-order masking in software. A second key remark is that a cipher that uses only bitwise operations can have an efficient first-order masked implementation only if it has a small number of intermediate variables.

Discussion. Our implementations explored how far one can push the optimization level in Boolean masking of various algorithms and ciphers. Consequently, we lost the benefit of being able to provide strong security proofs for our implementations. In other words, one can insert a random value in our expressions for masked AND and OR and they will still be a little bit more efficient than the best known ones, but provably secure. On the other hand, if one removes the randomness from the best know expressions for masked AND and OR, they will leak. We kept the amount of randomness at a minimum level (i.e. one or two random values for algorithms using the best known expressions and no random for our expressions). In these settings, the composition problem (i.e. chaining basic secure operations in an unsecure way) is similar for algorithms and ciphers masked using the previously best known expressions and our expressions. Finally, we stress that we put a similar effort in all our implementations.

5 Conclusion

We described an efficient algorithm for searching of optimal Boolean masking expressions. Then, we proposed optimal expressions for the first-order masking of bitwise AND and OR. They require less elementary operations and no random values compared to the best known expressions in the literature. Based on these optimal expressions, we presented an improved version of the algorithm for modular addition on Boolean shares proposed by Coron *et al.* [9]. We implemented the original and improved algorithms for modular addition/subtraction of 32-bit values on an ARM Cortex-M3. Our results show that the improved algorithm is between 14% and 19% faster than the original algorithm of Coron *et al.* [9].

Finally, we used our optimal Boolean masking expressions to write first-order protected implementations of three lightweight block ciphers, namely Simon, Speck, and RECTANGLE. The evaluation of these implementations revealed that ciphers with a simple structure, based solely on bitwise logical operations and rotations, facilitate efficient software implementations of first-order masking.

Acknowledgements. The authors thank the anonymous reviewers for their valuable comments. The work of Aleksei Udovenko is supported by the *Fonds National de la Recherche*, Luxembourg (project reference 9037104).

A Leakage Assessment

The tool we used to assess the security of our implementations against first-order attacks is inspired from similar tools such as ELMO [20], ASCOLD [21], and the one described in [23]. The simulated leakages are computed as follows. For each register r_i we store its previous value r_i^{j-1} and its current value r_i^j. At each step j we dump the leakage as $\mathsf{HW}(r_i^j)$ or $\mathsf{HD}(r_i^{j-1}, r_i^j) = \mathsf{HW}(r_i^{j-1} \oplus r_i^j)$, where $\mathsf{HW}(\mathsf{r})$ is the Hamming weight of r.

The result of the t-test applied to 10^6 simulated traces (using the HW model) from our first-order protected implementation of Speck is exemplarily shown in Fig. 3. Similar results for Simon and RECTANGLE are given in Figs. 4 and 5, respectively. All results use our expressions to compute secure AND and OR. We can see that the value of the t-statistic is inside the ±4.5 interval for each point in time, which implies that the protected implementations are secure against first-order attacks.

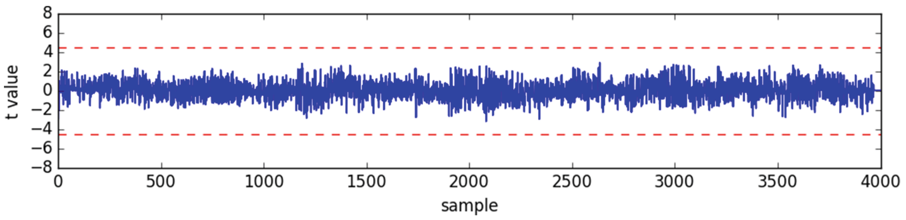


Fig. 3. The result of the t-test applied to our implementation of Speck.

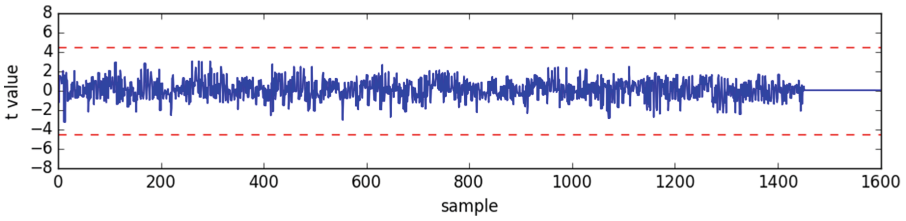


Fig. 4. The result of the t-test applied to our implementation of Simon

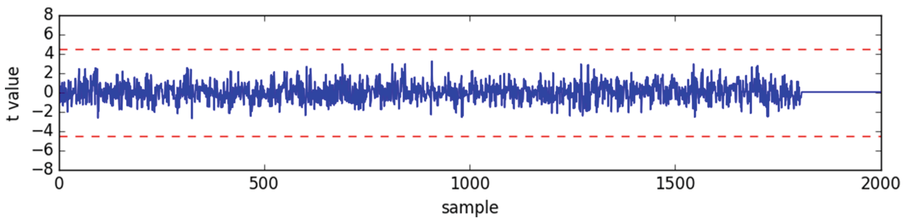


Fig. 5. The result of the t-test applied to our implementation of RECTANGLE.

References

1. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the Mirai Botnet. In: 26th USENIX Security Symposium (USENIX Security 2017), Vancouver, BC. USENIX Association (2017)
2. Random Number Generator (TRNG) API, October 2012. https://forum.arduino.cc/index.php?topic=129083.0. Accessed 03 July 2017
3. Baek, Y.-J., Noh, M.-J.: Differential power attack and masking method. Trends Math. **8**(1), 1–15 (2005)
4. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.-X.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 64–81. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16763-3_5
5. Baysal, A., Şahin, S.: RoadRunneR: a small and fast bitslice block cipher for low cost 8-bit processors. In: Güneysu, T., Leander, G., Moradi, A. (eds.) LightSec 2015. LNCS, vol. 9542, pp. 58–76. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29078-2_4
6. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 June 2015, pp. 175:1–175:6. ACM (2015)
7. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
8. Constantin, L.: Hackers Found 47 New Vulnerabilities in 23 IoT Devices at DEF CON, September 2016. http://www.csoonline.com/article/3119765/security/hackers-found-47-new-vulnerabilities-in-23-iot-devices-at-def-con.html. Accessed 03 July 2017
9. Coron, J.-S., Großschädl, J., Tibouchi, M., Vadnala, P.K.: Conversion from arithmetic to Boolean masking with logarithmic complexity. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 130–149. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_7
10. Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: Nessie proposal: Noekeon. In: First Open NESSIE Workshop, pp. 213–230 (2000)
11. T. P. Developers: PyPy Interpreter, version 5.1.2 (2016). https://pypy.org/

12. Ding, A.A., Chen, C., Eisenbarth, T.: Simpler, faster, and more Robust T-test based leakage detection. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 163–183. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43283-0_10
13. Dinu, D., Corre, Y.L., Khovratovich, D., Perrin, L., Großschädl, J., Biryukov, A.: Triathlon of Lightweight Block Ciphers for the Internet of Things. IACR Cryptology ePrint Archive, 2015:209 (2015)
14. Gartner: Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, February 2017. http://www.gartner.com/newsroom/id/3598917. Accessed 03 July 2017
15. Gilbert Goodwill, B.J., Jaffe, J., Rohatgi, P., et al.: A testing methodology for side-channel resistance validation. In: NIST Non-invasive Attack Testing Workshop (2011)
16. Gross, H.: Sharing is caring—on the protection of arithmetic logic units against passive physical attacks. In: Mangard, S., Schaumont, P. (eds.) RFIDSec 2015. LNCS, vol. 9440, pp. 68–84. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24837-0_5
17. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
18. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer, New York (2007)
19. Marsaglia, G., et al.: Xorshift RNGs. J. Stat. Softw. **8**(14), 1–6 (2003)
20. McCann, D., Oswald, E., Whitnall, C.: Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In: Kirda, E., Ristenpart, T. (eds.) 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, 16–18 August 2017, pp. 199–216. USENIX Association (2017)
21. Papagiannopoulos, K., Veshchikov, N.: Mind the gap: towards secure 1st-order masking in software. In: Guilley, S. (ed.) COSADE 2017. LNCS, vol. 10348, pp. 282–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64647-3_17
22. Public Comments Received on "Profiles for the Lightweight Cryptography Standardization Process", June 2017. https://www.nist.gov/sites/default/files/documents/2017/06/20/public-comments-profiles-i-ii-june2017.pdf. Accessed 03 July 2017
23. Reparaz, O.: Detecting flawed masking schemes with leakage detection tests. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 204–222. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_11
24. Ronen, E., Shamir, A., Weingarten, A., O'Flynn, C.: IoT goes nuclear: creating a zigbee chain reaction. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017, pp. 195–212. IEEE Computer Society (2017)
25. Schneider, T., Moradi, A.: Leakage assessment methodology. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 495–513. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_25
26. Schneider, T., Moradi, A., Güneysu, T.: Arithmetic addition over Boolean masking. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 559–578. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_27

27. Standaert, F.-X.: How (not) to use Welch's t-test in side-channel security evaluations. Cryptology ePrint Archive, Report 2017/138 (2017). http://eprint.iacr.org/2017/138
28. Trichina, E.: Combinational Logic Design for AES SubByte Transformation on Masked Data. IACR Cryptology ePrint Archive, 2003:236 (2003)
29. Won, Y., Han, D.: Efficient conversion method from arithmetic to Boolean masking in constrained devices. IACR Cryptology ePrint Archive, 2016:664 (2016)
30. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)

A First-Order Chosen-Plaintext DPA Attack on the Third Round of DES

Oscar Reparaz[1,2] and Benedikt Gierlichs[1](✉)

[1] imec-COSIC, KU Leuven, Leuven, Belgium
{oscar.reparaz,benedikt.gierlichs}@esat.kuleuven.be
[2] Square, Inc., San Francisco, USA

Abstract. DPA attacks usually exhibit a "divide-and-conquer" property: the adversary needs to enumerate only a small space of the key (a key sub-space) when performing the DPA attack. This is achieved trivially in the outer rounds of a cryptographic implementation since intermediates depend on only few key bits. In the inner rounds, however, intermediates depend on too many key bits to make DPA practical or even to pose an advantage over cryptanalysis. For this reason, DPA countermeasures may be deployed only to outer rounds if performance or efficiency are critical. This paper shows a DPA attack exploiting leakage from the third round of a Feistel cipher, such as DES. We require the ability of fixing inputs, but we do not place any special restriction on the leakage model. The complexity of the attack is that of two to three DPA attacks on the first round of DES plus some minimal differential cryptanalysis.

1 Introduction

Cryptographic implementations on embedded devices are susceptible to side-channel attacks [Koc96]. Differential Power Analysis (DPA) attacks are a powerful strand of side-channel attacks [KJJ99]. DPA is based on the fact that in an unprotected embedded device, the instantaneous power consumption depends somehow on the intermediate data handled by the implementation.

The basic working principle of DPA is to compare power consumption measurements from the device when executing the cryptographic implementation with a key-dependent model of its behavior. When modeling the device behavior, the practitioner places hypotheses on subkey values (obviously the key is secret and hence unknown). By comparing a model with the actual device behavior, DPA allows to verify or reject hypotheses on subkeys, and hence learn the actual key values. DPA and countermeasures are nowadays topics of intense research with dozens of scientific papers published per year on conferences devoted to the field.

Basic DPA attacks target the outer rounds: either the first one (if the input is known) or the last one (for known output). In outer rounds, every sensitive intermediate variable depends on only few key bits. Thus, a side-channel adversary can easily model the device behavior when handling such intermediates by placing hypothesis on only few key bits. A critical property of DPA attacks is

T. Eisenbarth and Y. Teglia (Eds.): CARDIS 2017, LNCS 10728, pp. 42–50, 2018.
https://doi.org/10.1007/978-3-319-75208-2_3

that they allow the adversary to "divide and conquer": the adversary just repeats the same methodology with different intermediates to learn different subkey bits until he learns enough key material to break the device.

DPA countermeasures aim to prevent DPA attacks, usually by lowering the SNR of the side-channel and by data randomization. However, countermeasures come with a considerable implementation overhead, e.g. increased execution time. Therefore, if performance is of importance, one may consider to protect only outer rounds until the cipher provides enough diffusion and intermediates depend on "many" key bits. This prevents basic DPA attacks on outer rounds and allows to use a more efficient, unprotected implementation for the inner rounds.

There are DPA attacks that target inner rounds. One way to circumvent the problem of an intermediate depending on too many key bits is to deactivate portions of input texts by fixing them to a constant value. As an example, suppose we target an intermediate V that is the xor of four S-box outputs $V = S(p_1 + k_1) + S(p_2 + k_2) + S(p_3 + k_3) + S(p_4 + k_4)$. If we set p_2, p_3 and p_4 to constant values, the intermediate V can be rewritten as $V = S(p_1 + k_1) + c$ for some constant c. Then, one can perform a DPA on V to jointly recover k_1 and the constant c. This is less effort (about 2^{2w} for w-bit variables) than jointly recovering (k_1, k_2, k_3, k_4) (about 2^{4w}). In many situations, when the practitioner carefully chooses the appropriate statistical distinguisher tools, it is even possible to first recover k_1 alone, and later, in a separate step, search for the constant c, further decreasing complexity to 2×2^w.

Previous work. Kunz-Jacques et al. describe a new DPA attack, called DMPA [KMV04], based on the Davies–Murphy attack on DES [DM95]. The basic idea is that the S-box output distribution of adjacent S-boxes is not independent, and the joint output distribution depends on (a linear function of) key bits. DMPA is a higher-order attack that does not need information on plaintexts but is rather expensive in terms of data and computational complexity. Handschuh and Preneel [HP06] present a differential attack on DES aided by collisions detected on power consumption traces. They hence require a device leakage behavior in the inner rounds that allows to reliably detect collisions on individual traces. Kim et al. showed that DES is vulnerable if not all rounds are masked [KLL10], relying also on collisions and subsequent cryptanalysis. Dodis and Pietrzak introduce highly theoretical attacks on generic Feistel networks in their CRYPTO 2010 publication [DP10]. Biryukov and Khovratovich present attacks that exploit leakage from inner rounds of AES in CHES 2007 [BK07].

Our contribution. We describe a simple DPA on the third round output of a Feistel cipher. The attack uses standard first-order DPA assumptions, and thus, it is very robust to noise and simple to mount. The attack is performed in two steps. In the first step, we perform a first-order DPA with chosen input texts to deactivate parts of the state and apply Jaffe's trick to push unknown constants into the key guess [Jaf07]. In the second step, we perform a minimal cryptanalytical differential attack. Contrary to other approaches, the number of required traces for our attack is not determined by any differential propagation probability, but only by the device SNR. We fully implemented and verified our attack on a software DES implementation.

2 A First-Order Chosen-Plaintext DPA Attack on the Third Round of DES

Notation. Figure 1 shows the relevant part of the first three rounds of a Feistel network and sets the notation for the remainder of this paper. In the case of DES, the initial permutation (IP) is applied to the 64-bit input, then the input is placed in two 32-bit words (L_0, R_0) and the iterated processing begins. The round function is applied to the right half R_i and the round key k_i and the result is xored to the left part L_i. Then, both parts are swapped. This is repeated for $r = 16$ rounds.

$$R_{i+1} = L_i \oplus F_{k_i}(R_i) \tag{1}$$
$$L_{i+1} = R_i \qquad 0 \leq i < r \tag{2}$$

In the last round, there is no swap and a final permutation is applied (IP^{-1}). The round function results from the composition of an Expansion stage E that maps 32 bits to 48 bits in a linear way, a key mixing stage that xors 48 subkey bits k_i, a non-linear substitution layer S and a linear permutation P as

$$F_{k_i}(R_i) = P(S(E(R_i) \oplus k_i)). \tag{3}$$

Decryption is identical to encryption up to a different key schedule. The observations of this paper can be applied either way. However, it is not possible to perform our attack to round 14 since we cannot choose the output.

Setting. In this paper, we assume the adversary acquires side-channel leakage corresponding to the third round, i.e., processing after (L_2, R_2). Normally, this would correspond to a device that deploys effective countermeasures only on the first two and last two rounds. We aim to recover the full DES key.

Our attack. Our attack consists of two steps. The first step is a DPA attack with chosen inputs. It recovers the second round key blinded by some unknown constant. The second step is a differential cryptanalysis that exploits differences in the unknown constant for different chosen inputs to reveal the first round key. Once the first round key is revealed, we can compute the blinding term of the second round key (this value was unknown after the first step) and thus derive the second round key. From two consecutive round keys, the full DES key is recovered.

2.1 Step 1

Step 1 consists of a DPA attack with chosen input targeting leakage of L_3. The input is chosen such that *after IP* we have varying L_0 and constant R_0.

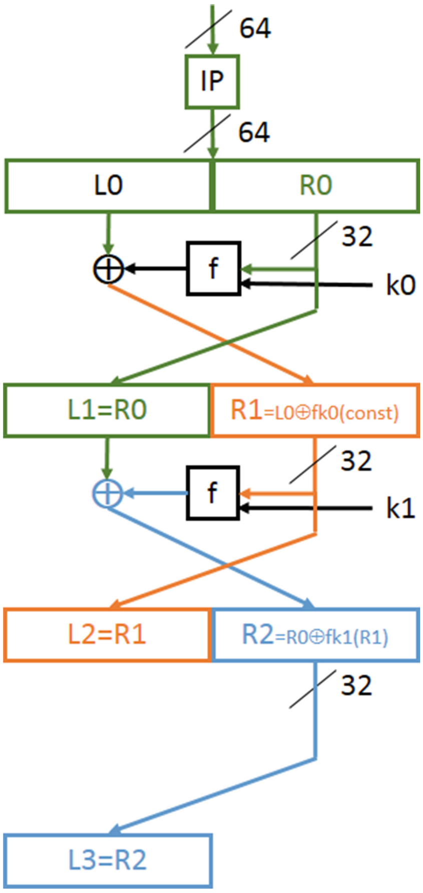


Fig. 1. First 2.5 rounds of a Feistel cipher.

This enables us to "skip" placing hypotheses on the first round subkey and instead we place hypotheses on second round keys. Note that $L_1 = R_0$ is known and that $R_1 = L_0 \oplus F_{k_0}(R_0)$ is only blinded by the unknown constant $C = F_{k_0}(R_0)$. We will for the moment assume that $R_1 = L_0$ and recover C later.

With a guess on k_1 we are able to compute the output of F in round 2, and since we know L_1 we can compute further until R_2. The Feistel construction gives us the next hop for free: $L_3 = R_2$. Therefore we can exploit the leakage of L_3 to recover k_1. More precisely, this attack recovers $k_1 \oplus E(C) = k_1 \oplus E[F_{k_0}(R_0)]$.

This approach works because one can view the S-box output in the second round $F_{k_1}(R_1)$ as

$$\begin{aligned} F_{k_1}(R_1) &= F_{k_1}(L_0 \oplus C) && (4)\\ &= F_{k_1}(L_0 \oplus F_{k_0}(R_0)) && (5)\\ &= F_{k_1 \oplus E[F_{k_0}(R_0)]}(L_0) && (6)\end{aligned}$$

where E is the expansion function inside the round function F. We are hence pushing the unknown constant $C = F_{k_0}(R_0)$ to the key hypothesis k_1. This can be thought of as a variant of Jaffe's trick [Jaf07].

Before we proceed with step 2, we need to iterate step 1 some small number of times with different constant values R_0' and R_0'', recovering $k_1 \oplus E(C') = k_1 \oplus E[F_{k_0}(R_0')]$ and $k_1 \oplus E(C'') = k_1 \oplus E[F_{k_0}(R_0'')]$. The second step will untangle the two terms k_1 and $E(C)$ from the recovered, "blinded", keys $k_1 \oplus E(C)$.

2.2 Step 2

Step 2 is a classic differential attack on 1-round Feistel to recover the first round subkey k_0 from the constants $E(C)$, $E(C')$ and $E(C)''$.

Consider the differences

$$\gamma = (k_1 \oplus E(C)) \oplus (k_1 \oplus E(C')) \tag{7}$$
$$\gamma' = (k_1 \oplus E(C')) \oplus (k_1 \oplus E(C'')) \tag{8}$$
$$\gamma'' = (k_1 \oplus E(C'')) \oplus (k_1 \oplus E(C))\,. \tag{9}$$

We have

$$\gamma = E(C) \oplus E(C') \tag{10}$$
$$= E(F_{k_0}(R_0)) \oplus E(F_{k_0}(R_0'))\,. \tag{11}$$

The values γ, γ' and γ'' are thus the first round output differences after the expansion E, which is invertible. Note that the adversary knows the first round input differences $R_0 \oplus R_0'$. Therefore, given the first round input and output differences, we can launch a key-recovery differential attack to recover k_0. Since we are targeting only one round, this differential attack can be performed in a divide and conquer, S-box by S-box, fashion.

In more detail: for each S-box in round 1 we place a 6-bit hypothesis on the corresponding part of k_0 and compute the output difference corresponding to input R_0 and R_0'. If the obtained output difference (after applying the expansion) is the same as the corresponding part of γ for that S-box, the subkey is kept as a candidate. Otherwise it is discarded. We repeat the procedure for different output differences γ' and γ''. The intersection of candidates is expected to yield a unique and correct subkey.

Once k_0 is recovered we can resolve $C = F_{k_0}(R_0)$, plug it in $k_1 \oplus E(C)$ to solve for k_1 and we are done. We recovered two round keys, thus, we can invert the key schedule and recover the DES key.

3 Implementation

We have fully implemented and verified our attack on an unprotected software implementation of DES in an 8-bit microcontroller. Figure 2, top, shows a power trace.

Step 1 is a classical DPA attack exploiting leakage from L_3. We made sure that this DPA attack does not exploit any leakage of rounds one and two. The target intermediate L_3 also appears as output of round 2, but we are assuming that the implementation starts leaking after round 2. In Fig. 2, bottom, we plot the result for the attack on one S-box after 200 traces. The correct value for a 6-bit chunk of $k_1 \oplus E(C)$ is distinguished with a comfortable margin, as Fig. 3, left, shows.

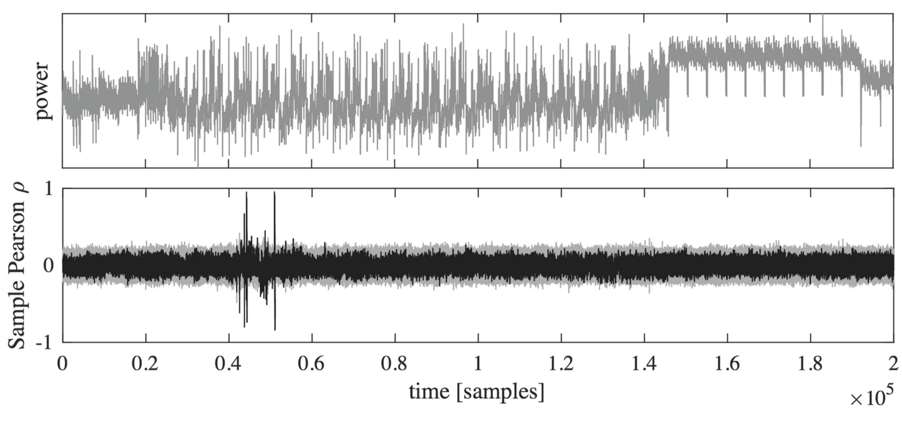


Fig. 2. Top: power consumption trace, heavily low-pass filtered to make SPA features more visible. Bottom: correlation traces. Incorrect key guesses in gray, correct key hypothesis in black. Peaks appear at the end of round 2 and at the end of round 3.

3.1 Step 1

We repeat this step three more times with different fix value R_0. The results of this step are:

- $R_0 =$ `88 00 17 FD`, recovered key $k_1 \oplus E[F_{k_0}(R_0)] =$ `25 0D 02 24 15 00 06 1F`.
- $R_0' =$ `A9 60 1B 9F`, recovered key $k_1 \oplus E[F_{k_0}(R_0')] =$ `2A 34 11 1A 31 08 05 23`.
- $R_0'' =$ `3E 57 8B 11`, recovered key $k_1 \oplus E[F_{k_0}(R_0'')] =$ `0B 2B 2D 11 0B 27 37 09`.
- $R_0''' =$ `3E 3E 3E 3E`, recovered key $k_1 \oplus E[F_{k_0}(R_0''')] =$ `0B 2E 39 18 1F 2F 32 19`.

(R is given as 4 8-bit values in hexadecimal; $k_1 \oplus E[F_{k_0}(R_0)]$ is given as 8 6-bit values, one per S-box).

3.2 Step 2

The differential attack from step 2 applied to the previous results yields the following four candidates for the k_0 round key. From each candidate for k_0 we can derive one candidate for the k_1 second round key by resolving C.

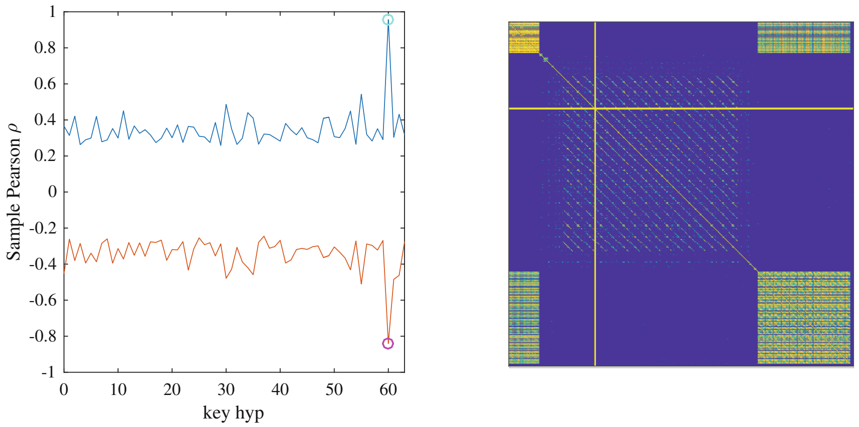


Fig. 3. Left: min/max correlation coefficient for each key, over all timesamples from round 3. The correct key hypothesis is marked with a circle. Right: cross-correlation matrix of a single trace, spanning the same time window as Fig. 2. The time sample for which the Pearson correlation is maximal is marked in the picture.

- $k_0 =$ `17 00 21 0C 15 18 3D 0F` $\Longrightarrow$ $k_1 =$ `14 12 37 30 19 09 1F 0C`
- $k_0 =$ `17 00 21 1F 15 18 3D 0F` $\Longrightarrow$ $k_1 =$ `14 12 37 30 19 09 1F 0C`
- $k_0 =$ `17 00 21 23 15 18 3D 0F` $\Longrightarrow$ $k_1 =$ `14 12 3F 30 1B 29 1F 0C`
- $k_0 =$ `17 00 21 30 15 18 3D 0F` $\Longrightarrow$ $k_1 =$ `14 12 3F 30 1B 29 1F 0C`

From every candidate for (k_0, k_1) we can invert the key schedule. (We could detect already incorrect candidates if the candidate (k_0, k_1) do not correspond to the DES key schedule, but since the number of candidates is so low in our case, we did not implement this option). The correct round keys are found to be $k_0 =$ `17 00 21 0C 15 18 3D 0F` and $k_1 =$ `14 12 37 30 19 09 1F 0C`; this corresponds to the DES key `3B 38 98 37 15 20 F7 5E`. We verified the correctness of the entire procedure with plaintext/ciphertext pairs.

4 Discussion

Distance leakage. In a typical hardware implementation, the attacker measures leakage roughly corresponding to $\mathrm{HW}(L_3 \oplus L_2)$. Exactly the same attack can be mounted in this case, *mutatis mutandis*, adjusting predictions in the DPA of step 1. The practitioner knows that $L_2 = R_1 \oplus C$. This C is unknown at this stage, but constant, so that he can revert to single-bit DPA (which would ignore the effect of C) or perform DPA recovering C as well.

If L_2 is masked, e.g. because the first two rounds are masked, the Hamming distance from L_2 to L_3 is also masked and the attack does not work immediately. However, a device that exhibits Hamming distance leakage typically exhibits also Hamming weight leakage (albeit possibly weaker).

Jaffe's trick. Jaffe [Jaf07] used a similar trick in a different context. He gave a surprisingly elegant attack on the CTR mode of operation, even when the starting counter value is unknown. (The amusing part here is that his is effectively a *blind* DPA attack with unknown inputs and outputs). The basic idea is to push the unknown counter value to the subkey hypothesis, so that the DPA attack recovers at the same time the subkey and the initial counter value.

Optimizations. It may be possible to choose clever values for input differences $R_0 \oplus R'_0$ to minimize the number of candidate keys output in the second step, and thus to accelerate the whole attack. However, the gain is very thin. One condition that the input difference should satisfy is that all first-round S-boxes should be active (otherwise, the differential attack of step 2 cannot eliminate any incorrect key guess for the inactive S-box). This can be achieved, for example, with the easy-to-memorize difference $R_0 \oplus R'_0 =$ `FF FF FF FF`.

How many different inputs do we need? It is possible to mount the attack with just one input difference, i.e., one known plaintext and one chosen plaintext. We have empirically determined that, if the input difference is `FF FF FF FF`, step 2 will (in the worst case) return 8, 14, 10, 16, 8, 8, 14 and 10 sub-key candidates for S-box number 1, ..., 8 respectively. This means that the step 2 yields $8 \times 14 \times \ldots \times 10 < 2^{28}$ keys, which can be easily bruteforced in a matter of seconds in a workstation. (This is a very rough upper bound, one can cut this number by first applying a consistency check if k_0 and k_1 fit the DES key schedule).

Influence of the key schedule. Note that in the process of deriving k_1 from k_0 by resolving C in Sect. 2.2, we did not exploit the fact that in DES the round keys k_0 and k_1 are heavily correlated (since the DES key schedule is so simple). This method can thus be used even for other Feistel ciphers with an arbitrary key schedule algorithm, even when k_1 is completely independent of k_0. Our method, as described, recovers the first two round keys k_0, k_1. If two round keys are not enough to invert the key schedule, once the adversary learns k_0 and k_1 he can iterate the attack peeling off the first two rounds to recover k_2 and k_3 until he gets the desired amount of round keys.

5 Conclusion

In this paper, we have described a first-order chosen-plaintext DPA attack on the DES exploiting leakage stemming from the third round. This stresses, once again, the necessity of protecting implementations of outer and inner rounds in Feistel ciphers. Our attack is very easy to carry out, is resilient to noise (we only make use of first-order statistics), can be carried out with negligible computational power and recovers the full DES key.

Acknowledgments. The authors wish to thank the anonymous reviewers for their insightful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the Flemish Government, FWO G.00130.13N, FWO G.0876.14N and Thresholds G0842.13; by the Hercules Foundation AKUL/11/19; and through the Cathedral ERC Advanced Grant 695305. Oscar Reparaz is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO). Benedikt Gierlichs is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

[BK07] Biryukov, A., Khovratovich, D.: Two new techniques of side-channel cryptanalysis. In: Paillier and Verbauwhede [PV07], pp. 195–208

[DM95] Davies, D.W., Murphy, S.: Pairs and triplets of DES S-boxes. J. Cryptol. **8**(1), 1–25 (1995)

[DP10] Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_2

[HP06] Handschuh, H., Preneel, B.: Blind differential cryptanalysis for enhanced power attacks. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 163–173. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74462-7_12

[Jaf07] Jaffe, J.: A first-order DPA attack against AES in counter mode with unknown initial counter. In: Paillier and Verbauwhede [PV07], pp. 1–13

[KJJ99] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25

[KLL10] Kim, J., Lee, Y., Lee, S.: DES with any reduced masked rounds is not secure against side-channel attacks. Comput. Math. Appl. **60**(2), 347–354 (2010)

[KMV04] Kunz-Jacques, S., Muller, F., Valette, F.: The Davies-Murphy power attack. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 451–467. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_32

[Koc96] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9

[PV07] Paillier, P., Verbauwhede, I. (eds.): CHES 2007. LNCS, vol. 4727. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2

A Strict Key Enumeration Algorithm for Dependent Score Lists of Side-Channel Attacks

Yang Li(✉), Shuang Wang, Zhibin Wang, and Jian Wang

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China
li.yang@nuaa.edu.cn

Abstract. Post-processing of side-channel attack trades computational efforts to recover the secret key even when some subkeys are not ranked the highest in their score lists. Recently, many key enumeration (KE) algorithms have been proposed, which attempt to effectively enumerate the key candidates in the sequence of the score of the combined key. However, the existing KE algorithm can only combine the score lists of independent subkeys. In this paper, we consider a more general key enumeration algorithm, which can combine the score lists that are internally restricted by each other. The proposed key enumeration algorithm can for example combine the score lists for k_0, k_1 and $k_0 \oplus k_1$, while the existing KE algorithms cannot be directly extended to solve this problem efficiently. We propose an efficient strict key enumeration algorithm that can run recursively for dependent score lists. With simulated side-channel leakage of AES-128, the proposed KE algorithm can enumerate the key according to 16 score lists of subkeys and 15 score lists of subkey difference. This KE algorithm can enumerate up to 2^{21} keys using 5 h and 128 MB of RAM with a normal PC. By taking advantage of the dependent score lists, the key recovery experiments using simulated power data show that the success rate is largely improved in general. The rank of correct key is statistically higher with the additionally used score lists.

Keywords: Key enumeration · Correlation-enhanced collision attacks
AES · Side-channel attacks

1 Introduction

Ever since Kocher's DPA proposal [1], the side-channel attack has been studied for almost two decades. Fundamentally, the SCA exploits the dependency between the side-channel leakage in the side-channel measurements and the key related intermediate values during cryptography calculations. Side-channel attack against block ciphers usually follows a divide-and-conquer approach, in which the entire secret key is divided into small subkeys and recovered one by one. The recovered subkeys are combined to obtain the entire key.

With enough leakage information and appropriate key recovery method, the entire key is the combination of all the most likely subkeys. When the leakage information is not enough or the information of the target device is limited, some of the correct subkeys are not ranked the highest in the score list, but still have a higher score than many of the other candidates. The adversary can test the key candidates in the sequence of its likelihood to be the correct key. The key enumeration (KE) algorithm is used to enumerate the entire key candidate in a non-increasing order of its score or probability. A reasonable key enumeration algorithm could largely accelerate the key recovery compared to a naive random search.

The KE algorithm is one of the post-processing techniques in side-channel attacks, as described in the unified framework of SCA [2]. It can obtain the required amount of computation for the key recovery, and can be used to improve the accuracy in the security evaluation of the cryptography implementations. In contrast to an unknown key assumption, the key rank estimation algorithm [3] attempts to locate the rank of a given secret key quickly and accurately. This work puts more focus on the KE algorithm.

Recently, the KE algorithm has been extensively studied for its great boost to the key-recovery attack performance. The key enumeration algorithm is interpreted by a geometry problem in [4], and transferred to a backpack problem in [5] and a convolution of histograms in [6]. The existing KE algorithms share one common fact that they only deal with the score lists corresponding to independent key information. For example, for AES-128, usually the input for key enumeration is 16 score lists corresponding to 16 key bytes.

In 2017, a key rank estimation algorithm that can estimate the key rank considering dependent score lists has been proposed [7]. In this paper, it proved that the rank of the correct key decreases when considering dependent score lists. We consider the KE algorithm should be expanded to be capable of enumerating the combined key using dependent score lists. Dependent score lists have related key information that restrict each other, e.g., k_0, k_1 and $k_0 \oplus k_1$. One possible application scenario is explained as follows. Two different approaches can be used to recover the key information in different forms. The attackers can use CPA-like attack [8] to recover individual key byte, and use correlation-enhanced collision attack [9] to recover the difference between key bytes. Then, one reasonable approach of the key enumeration is to consider all the recovered score lists. When more score lists are used and combined reasonably, the correct key can be recovered with fewer tests. There lacks an efficient KE algorithm to practically take advantage of the dependent score lists.

Contributions

This work proposes an efficient KE algorithm that combines the mutually restricted score lists. Also, this work investigates the advantage of such attack approach using simulated leakage data. The detailed contributions of this work are summarized as follows.

1. We start from the existing KE algorithm and try to extend it to fit the general KE problem. The result is a strict key enumeration algorithm that can enumerate 3 keys lists from the most probable to the least probable one. However, it cannot be effectively generalized to enumerate more score lists.
2. We propose a new KE algorithm that is efficient in time and memory, which can also run recursively. This new algorithm has the following features.
   (a) It is a deterministic algorithm that allows to strictly enumerate key candidates from any number of lists of any size.
   (b) It is very flexible to choose the number of combined score lists to trade for a balance of reasonable complexity.
3. A comprehensive evaluation is applied to verify the complexity of the proposed KE algorithm. In addition, this work demonstrates the key enumeration result by combining both the key-recovery attack and the key-difference recovery attack. The experimental result confirms the advantage of combining dependent score lists by statistically ranking the correct key higher in the enumeration.

The rest of this paper is organized as follows. Section 2 reviews the existing key enumeration algorithms. Section 3 explains the motivations for the key enumeration problem of dependent score lists. Section 4 explains the evolution towards the proposed KE algorithm and Sect. 5 performs a detailed evaluation of it. Finally, Sect. 6 concludes this paper.

2 Previous Key Enumeration Algorithms

The concept of key enumeration was first proposed in 1991. Meier et al. proposed a probabilistic key enumeration algorithm [10]. They aimed at stream cipher and enumerated the pseudo random sequences generated by cellular automata. However, it is a random selection process which causes useless repetitions.

After this article, key enumeration algorithm had not been extensively studied until Pan, Woudenberg et al. described a deterministic key enumeration algorithm [11] at SAC 2010. Due to its large memory requirement, the algorithm is limited to the enumeration of 2^{16} key candidates.

Optimal Key Enumeration. In 2012, Veyrat-Charvillon et al. proposed a strict key enumeration algorithm [4]. First, the authors proposed a Bayesian expansion method to obtain the probability information of the subkeys [12]. The proposed deterministic key enumeration algorithm outputs the full combined keys in a strict order - from more likely to less likely ones. This intrinsic quality could leads to a larger consumption about the memory. However, the strict order minimizes the expected number of key test trials. Any non-strict algorithm incurs an overhead in terms of trials during the key recovery phase.

Smart and Parallel in Key Enumeration. In 2015, Martin et al. proposed a key enumeration algorithm based on the knapsack problem [5]. Their solution can enumerate the keys in a parallel fashion. It needs to turn the floating scores into the integer type of weight, which could affect the accuracy of the results.

Integrated Approach in Key Enumeration. In 2016, Poussier, Standaert and incent Grosso proposed a key enumeration algorithm based on histogram [6]. In this article, the probability is divided into equal intervals. They assumed that all the keys in the same probability range are equivalent, and then the 16 key bytes are combined in the way of histogram merging.

Non-strict Approaches in Key Enumeration. In 2015, Bogdanov et al. proposed a score-based key enumeration algorithm [13], which is a variation of the depth first search. Though it is sub-optimal, its memory and running time requirements are more practical than that in [4]. Meanwhile it can be efficiently parallelizable.

In 2017, David and Wool proposed a new key enumeration algorithm which can use less memory space [14]. This algorithm is also a parallelizable algorithm that enumerates the keys in near-optimal order.

3 General Key Enumeration Problem with Dependent Score Lists

In this work, we discuss a more general key enumeration problem, in which the dependent score lists are used together to calculate the score of the combined key. Assume that the attacked key has n independent subkeys. An attacker can perform a key-recovery attack on the subkey, i.e. k_i, and the difference between subkeys, i.e. $k_{i,j} = k_i \oplus k_j$. Each key-recovery attack generates a score list that gives a score to each candidate value. As shown in Fig. 1, one can get $n(n+1)/2$ score lists in total, which are arranged in a triangular shape.

Existing key enumeration algorithms can only combine independent subkeys. In other words, selecting any candidate from a score list is independent from the candidate selection of the other score lists. A few examples of such sets of independent subkeys are shown in Fig. 1 inside the dotted line boxes.

The key enumeration problem proposed in this paper is to combine subkeys considering the dependent scores lists. The dependent score lists are mutually connected and restricted with each other. A few examples of such dependent score lists are shown inside the slide line boxes in Fig. 1.

In summary, the goal of this work is to propose an efficient strict key enumeration algorithm that can combine the arbitrarily selected score lists from Fig. 1.

Score Function. In this work, we don't distinguish the multiplication or the addition in the score calculation of the combined key. Instead, we define a score function to increase the generality.

Score function $S(\cdot)$ is a mapping from the scores from score lists to the score of the combined key. Normally, the score function is either a simple addition or multiplication. However, in our studied case, the score function could be complicated due to the inner connection between the combined score lists.

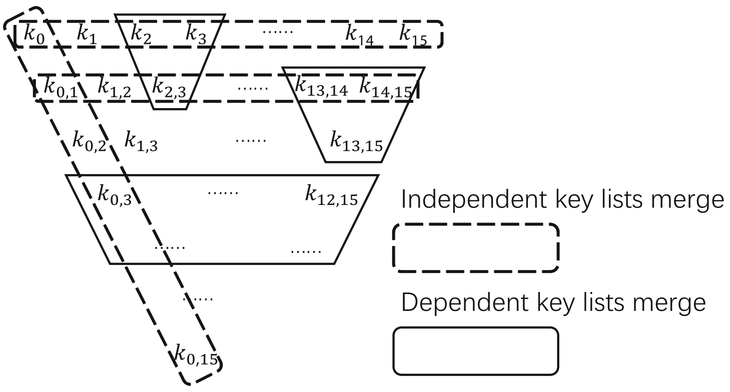

k_0 k_1 k_2 k_3 ...... k_{14} k_{15}
$k_{0,1}$ $k_{1,2}$ $k_{2,3}$ ...... $k_{13,14}$ $k_{14,15}$
$k_{0,2}$ $k_{1,3}$ ...... $k_{13,15}$
$k_{0,3}$ ...... $k_{12,15}$
$k_{0,15}$
Independent key lists merge
Dependent key lists merge

Fig. 1. Independent and dependent key list combine problem

In order to keep the generality of this paper, we only assume the score function is a monotonic function along each combined score. That is to say, for $S(a, b, c, \cdots)$, if $a1 > a2$, we know $S(a1, b, c, \cdots) > S(a2, b, c, \cdots)$. However, if $a1 > a2$ and $b1 < b2$, then the comparison between $S(a1, b1, c, \cdots)$ and $S(a2, b2, c, \cdots)$ can only be performed by calculating the scores. Both multiplication and addition fit this score function definition.

3.1 Significance of General Key Enumeration Problem

A good reason to consider dependent score list is the key recovery result of correlation-enhanced power analysis collision attack (CECA) [9]. Instead of recovering the key byte directly, the CECA attack focuses on recovering the difference between two key bytes, i.e. $k_{i,j} = k_i \oplus k_j$.

In 2003, Schramm et al. first proposed a side-channel attack that applied a collision attack to a block cipher [15]. Collision attack is used to detect whether the intermediate value is the same. Whenever the output of the first round S-box is the same, e.g. $S_0(i_0) = S_1(i_1)$, since the S-box is bijective, we have $p_0 \oplus k_0 = p_1 \oplus k_1$ and $p_0 \oplus p_1 = k_0 \oplus k_1$. An attacker who knows the plaintext and can detect the S-box output collision, can get the XOR relationship between 16 key bytes of AES-128. As a result, the key space is greatly reduced from 2^{128} to 2^8.

In 2010, Moradi et al. proposed the correlation-enhanced power analysis collision attack (CECA), which takes advantage of correlation calculation to put forward a new distinguisher for recovering the subkey difference. The basic assumption is that the S-box calculations with the same input lead to similar side-channel leakages. From all of the traces, the CECA attack collects and averages only those traces where the i-th plaintext byte equals a certain value $\alpha \in \mathrm{GF}(2^8)$. Hence, one can get 2^8 average traces M_i^α in position i, where M_i^α is the average of all traces where the i-th S-box input is $\alpha \oplus k_i$. One can get 2^8 average traces M_j^α in position j as well. Then, CECA exploits the leakage caused by the collision using all the measured traces for the collision between i-th and j-th S-boxes.

For positions i and j, whenever the plaintext bytes p_i and p_j satisfy $p_i \oplus p_j = k_i \oplus k_j = k_{i,j}$, the collision occurs for these two S-box calculations. Therefore, the M_i^{α} and $M_j^{\alpha \oplus k_{i,j}}$ should have similar side-channel leakage for $\alpha \in \mathrm{GF}(2^8)$. CECA's approach is to guess $k_{i,j}$ and verify the guess by verifying all resulting collisions for all α in $\mathrm{GF}(2^8)$ in a correlation calculation. The distinguisher of CECA is the correlation result of the averaged power consumption M_i^{α} and the averaged power consumption $M_j^{\alpha \oplus ki,j}$ for all $\alpha \in \mathrm{GF}(2^8)$. The correct key is expected to have the largest correlation.

CECA can exploit the first order leakage without knowing the precise leakage model, which fits most practical implementations with SCA countermeasures. In power analysis, it has been used to evaluate the side-channel leakage through static power in [16] and to evaluate masking countermeasure in [17]. The same distinguisher is also used in fault sensitivity analysis in [18,19] since the leakage model for fault sensitivity is different to model.

The target of CECA-like attack is the difference between key bytes. The key enumeration algorithm of dependent score lists is the tool to take advantage of the key-difference recovery attacks in the post-processing of SCA. In 2017, a key rank estimation algorithm for dependent score lists has been proposed [7], in which the authors consider combining the results of CPA-like attack and CECA-like attack in the key rank estimation. They use simulated power traces to prove that the rank of the correct key become generally higher when considering dependent key lists.

4 Strict Key Enumeration Algorithms for Dependent Score Lists

In this section, we first review the strict key enumeration algorithm for independent score lists from [4]. Then, we attempt to extend it to solve KE problem of dependent score lists. Lastly, we finalize our proposed KE algorithm.

4.1 Strict Key Enumeration Algorithm for Independent Score Lists

In [4], Veyrat-Charvillon et al. proposed an optimal (strict) key enumeration algorithm for independent score lists. The attacker obtains score lists for d independent subkeys, each score list taking n different values. The scores are viewed as the probability, and the score of the combined key is calculated by multiplication. The essence of their work can be explained as a 2-dimensional KE algorithm and a recursive enumeration approach.

A 2-dimensional KE problem is viewed as a geometric problem. Two subkeys are combined together so that the key space can be identified with a square of length 1 as shown in Fig. 2. The columns (resp. rows) correspond to the probability of the possible values of the first (resp. second) subkey, sorted by a non-increasing order of probability. Width and height correspond to the probability of the corresponding subkey.

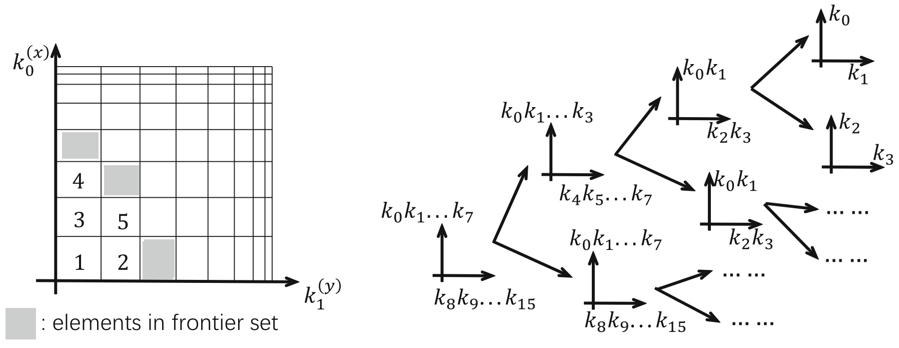


Fig. 2. 2-dimensional key enumeration and generalization to multiple lists [4].

Let us denote by $k_i^{(j)}$ the j^{th} likeliest value of the i^{th} subkey. In the map, the intersection rectangle corresponds to a combined key, and the area of the rectangle is the probability of it. The algorithm keeps a frontier set of rectangles as $\mathcal{F}$, which contains the candidates to be the next key with the largest probability. To keep the KE efficient in memory of storing $\mathcal{F}$, the KE algorithm takes advantage of the following Rule 1.

Rule 1. *$\mathcal{F}$ may contain at most one element in each column and row. For two elements in the same column or row, one of them definitely has a larger score than the other one.*

The key enumeration starts at the corner close to the origin. Each time when one element of $\mathcal{F}$ is outputted to the plaintext-ciphertext test, the rectangles next to it are added to $\mathcal{F}$ if no element inside $\mathcal{F}$ shares the same column or row. Using this geometric view, a strict key enumeration algorithm outputs compartments by a non-increasing order of area. This solution is illustrated in Algorithm 1. When implementing Algorithm 1, the frontier set $\mathcal{F}$ can be stored in an ordered structure and the test of rule 1 can be achieved by storing the arrays of Boolean values.

In [4], the generalization to multiple lists is achieved by the recursive use of the 2-dimensional KE algorithm. Taking an example of AES, enumerating 128-bit keys is done by merging two lists of size 2^{64}. Such lists can be generated at the time they are required in the key enumeration. Whenever a new subkey is inserted in the candidate set, the program obtain it from the enumeration algorithm applied to a lower level. This ensures that the storage and enumeration effort are minimized. In this way, merging n lists is done by merging two lists $n-1$ times. For more details, we refer the readers to [4].

Algorithm 1. A strict 2-dimensional KE algorithm [4]

```
1: $\mathcal{F} \leftarrow \{(k_0^{(1)}, k_1^{(1)})\}$;
2: while $\mathcal{F} \neq \emptyset$ do
3:     $(k_0^{(x)}, k_1^{(y)}) \leftarrow$ most likely candidate in $\mathcal{F}$;
4:     Output $(k_0^{(x)}, k_1^{(y)})$;
5:     $\mathcal{F} \leftarrow \mathcal{F} \backslash \{(k_0^{(x)}, k_1^{(y)})\}$;
6:     if $x + 1 \leq \#k_0$ and no candidate in row $x + 1$ then
7:         $\mathcal{F} \leftarrow \mathcal{F} \cup \{(k_0^{(x+1)}, k_1^{(y)})\}$;
8:     end if
9:     if $y + 1 \leq \#k_1$ and no candidate in column $y + 1$ then
10:        $\mathcal{F} \leftarrow \mathcal{F} \cup \{(k_0^{(x)}, k_1^{(y+1)})\}$;
11:    end if
12: end while
```

4.2 Attempt to Extend 2-Dimensional KE Algorithm to 3-Dimensional KE Algorithm

Similar to [4], our approach first considers a 3-dimensional key enumeration problem, and then tries to generalize it to more score lists in a recursive approach. The two subkeys in the 2-dimensional case are independent of each other. We add a third dimension to the key enumeration that corresponds to the difference between these two subkeys as shown in Fig. 3.

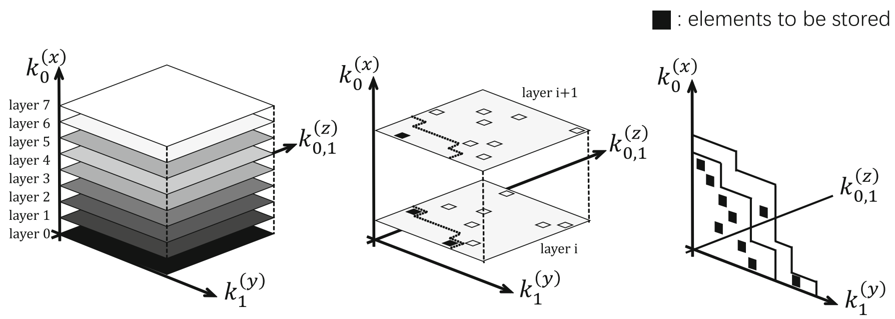


Fig. 3. Layers in 3-dimensional KE problem (left), concepts in Algorithm 2 (middle) and Algorithm 3 (right).

In case of 3-dimensional KE problem, a straightforward approach is to treat it as same as the independent score lists, and to deal with the dependency after the enumeration. Here, we assume that the score function is the multiplication so the combined key is a cube in the 3-dimensional space, and its volume is the score of the merged key. The algorithm locates the cube at the origin, which corresponds to the cube of the largest volume, and adds to the frontier set $\mathcal{F}$.

After this cube is outputted to test, all the neighboring cubes become candidates to $\mathcal{F}$. Similarly, we use Rule 2 to restrict the memory cost.

Rule 2 (Extended Rule 1). *$\mathcal{F}$ may contain at most one element in each line that is parallel to x-axis, y-axis, or z-axis.*

This process is repeated so that the cubes can be enumerated from this 3-dimensional space in a non-increase order of their volumes.

This method is theoretically feasible but inefficient. The main reason is that most of the cubes in this three-dimensional space are out of practical significance. For any given cube, the 3 axes represent two subkeys k_0, k_1 and their difference $k_{0,1}$. If $k_0^{(x)} \oplus k_1^{(y)} \neq k_{0,1}^{(z)}$, the corresponding cube is meaningless. Assume that the length of each axis is len, only $1/len$ of all the cubes are meaningful. The straightforward KE algorithm obviously wastes too much computation at the meaningless cubes. Thus, one can improve the algorithm by putting the verification before the key enumeration. Then, it is not difficult to notice the following fact.

Rule 3. *In the 3-dimensional space, there is only one valid element in each line that is parallel to x-axis, y-axis, or z-axis.*

According to Rule 3, whenever x and y are fixed, there is only 1 meaningful cube along z-axis. One can directly find the location of this only meaningful point with $(k_0^{(x)}, k_1^{(y)}, k_0^{(x)} \oplus k_1^{(y)})$. Consequently, an improved approach is to bypass all the meaningless cubes and still use $\mathcal{F}$ to achieve the memory control.

We use Fig. 3 to explain the improved KE algorithm. The 3-dimensional space is divided into different layers along the x-axis. The search starts from the layer close to the origin and gradually adds cubes to $\mathcal{F}$. For the current layer, the search goes through y, and directly finds the corresponding meaningful cube as $(k_0^{(x)}, k_1^{(y)}, k_0^{(x)} \oplus k_1^{(y)})$. In this search, we will maintain a boundary $\mathcal{B}$. Only the cubes inside $\mathcal{B}$ could be the candidates of the next cube with the largest volume. For each added cube to $\mathcal{F}$, the boundary B is updated as well. If a newly searched cube locates inside B, it will be added to $\mathcal{F}$, and B is updated accordingly.

For example, as shown in the middle of Fig. 3 the search in layer i ends when the boundary (black line) excludes all the rest part of layer 0. When the search restarts at layer 1, the boundary of layer 0 is kept for to the layer 1. Only the meaningful point inside $\mathcal{B}$ can be added to $\mathcal{F}$. Along the search, $\mathcal{F}$ become larger and larger, and the boundary $\mathcal{B}$ becomes smaller and smaller. After no more elements can be added to $\mathcal{F}$, the algorithm outputs the cube with the largest combined score and update $\mathcal{F}$ and $\mathcal{B}$. With each outputted cube, the boundary expands a little. The following search is applied according to the updated boundary area.

This algorithm is illustrated in Algorithm 2. Unfortunately, the evaluation of an implemented Algorithm 2 is not good enough for three reasons. First, the boundary $\mathcal{B}$ is complicated to maintain. Second, the search is slow since many cubes are checked multiple times. Third, the Algorithm 2 cannot be generalized

Algorithm 2. A strict 3-dimensional KE algorithm

```
1: $\mathcal{F} = \emptyset$;
2: $\mathcal{B} \leftarrow$ every point;
3: $x = 1, y = 1$;
4: repeat
5:     repeat
6:         if $(k_0^{(x)}, k_1^{(y)}, k_0^{(x)} \oplus k_1^{(y)}) \in \mathcal{B}$ then
7:             $\mathcal{F} = \mathcal{F} \cup (k_0^{(x)}, k_1^{(y)}, k_0^{(x)} \oplus k_1^{(y)})$
8:             Update($\mathcal{B}$); // remove all points that are smaller than $(k_0^{(x)}, k_1^{(y)}, k_0^{(x)} \oplus k_1^{(y)})$
9:         end if
10:        $y = y + 1$;
11:    until $k_0^{(x)} \oplus k_1^{(y-1)} \neq k_{0,1}^{(1)}$
12:    $x = x + 1$;
13: until $k_0^{(x-1)} \oplus k_1^{(1)} \neq k_{0,1}^{(1)}$
14: $(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)}) \leftarrow$ most likely candidate in $\mathcal{F}$;
15: Output $(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)})$;
16: $\mathcal{F} \leftarrow \mathcal{F} \backslash \{(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)})\}$;
17: Update($\mathcal{B}$); // add the removed points caused by $(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)})$ back to $\mathcal{B}$
18: Goto Step 4;
```

to more score lists in a recursive approach. The reason is that Algorithm 2 request at least one ordered axis to fast locate the meaningful cube in a line. In the given example, the z axis should be ordered before the search. When merging more dependent score lists recursively, the z axis will be the combination of other score lists that cannot be ordered before the search.

4.3 Strict 3-Dimensional Key Enumeration Algorithm

This section proposes a more efficient and an expandable KE algorithm to solve the strict 3-dimensional KE problem. First, in order to not check the same merged key multiple times, we want to store all the checked merged key cubes. Second, this new algorithm should not require an ordered axis so it can be performed in a recursive approach.

Still we divide the 3-dimensional space into layers and keep the same search sequence. The main difference is that we apply a new rule to stop each layer's search.

Rule 4. *In the search of a certain layer, x is fixed, $k_1^{(y)}$ is decreasing when y is increasing along the search. When $S(k_0^{(x)}, k_1^{(y)}, \text{MaxScore}_{k_{0,1}}) \leq \text{MaxScore}$, the search along y-axis stops, in which $\text{MaxScore}_{k_{0,1}}$ is the largest score of $k_{0,1}$ and* MaxScore *is current the largest score of all searched keys except the outputted ones.*

The reason is that when $S(k_0^{(x)}, k_1^{(y)}, \text{MaxScore}_{k_{0,1}}) \leq \text{MaxScore}$, the rest merged key along y axis is definitely smaller than MaxScore. For the same reason, $k_0^{(x)}$

and $k_1^{(y)}$ are both decreasing along the search, when $S(k_0^{(x)}, k_1^{(1)}, \mathrm{MaxScore}_{k_{0,1}}) \leq$ MaxScore, the rest merged key along x axis is definitely smaller than MaxScore.

For the search inside a layer, MaxScore is gradually increasing and $k_1^{(y)}$ is gradually decreasing so that the stop edge can be reached. When the search goes through different layers, the search stops when $S(k_0^{(x)}, k_1^{(1)}, \mathrm{MaxScore}_{k_{0,1}}) \leq$ MaxScore, which means the above layers cannot have candidates with a larger score. After the search over the 3-dimensional space, the combined key with MaxScore is outputted and MaxScore is updated to the previous second largest score. The search over 3-dimensional space can re-start from the stop edge of the previous search.

This search algorithm makes sure that each key candidate is only searched once so the time complexity is low. However, the memory cost is relative large since this algorithm keeps all the searched keys.

In order to achieve a balance between time and memory cost, we can apply a trade-off technique to this enumeration algorithm. Since the key search is applied in layers that are perpendicular to x-axis. Instead of storing all the searched keys, the algorithm can store only the keys with the largest score in each layer. The MaxScore can be easily find from the largest scores in all layers. Every time when a key candidate is outputted, the largest score of the corresponding layer becomes zero. Then we need to re-search the layer corresponding to the removed key and search other layers from the stop edge to find the updated largest score.

This trade-off can balance of space and time complexity and the parameter for this trade-off can be easily adjusted to fit the practical computational setup.

This efficient strict 3-dimensional KE algorithm after applying the time-memory trade-off is illustrated in Algorithm 3. Lines 9 to 15 correspond to the search inside a layer, and lines 7 to 18 correspond the search among different layers. For each layer, the algorithm stores the key with the largest score that has not been outputted and the search edge. This KE algorithm neither needs to store all the searched keys nor to re-search all the keys that are not stored.

Extension to Multiple Lists. Similar to 2-dimensional KE algorithm, the proposed 3-dimensional KE algorithm of Algorithm 3 can be extended to multiple lists in a recursive approach. For example, in order to combine $k_0||k_1$and $k_2||k_3$, the z-axis can be considered as $k_{0,2}, k_{0,3}, k_{1,2}, k_{1,3}$. The max score of the z axis can be a rough estimation by the score function using the max scores of $k_{0,2}, k_{0,3}, k_{1,2}$ and $k_{1,3}$ as the input. Though, this could lead to certain efficiency loss.

Actually, the 3-dimensional KE algorithm has a very nice feature that the searched lists can be easily adjusted without changing the KE algorithm. If the combined key want to ignore a certain score list, one can simply set all the scores to 0 in this score list. Also, different score lists can have different weights in order to increase the KE efficiency or to emphasize the significance difference of different score lists. In the example of combine $k_0||k_1$and $k_2||k_3$, the KE efficiency can be largely improved by either adding a low weight to $S(k_{0,2}, k_{0,3}, k_{1,2}, k_{1,3})$ or only using some of the four lists.

Algorithm 3. An efficient strict 3-dimensional KE algorithm

```
1: MaxScoreinLayer[1 : 256] = $\mathbf{0}$;
2: Edge[1 : 256] = $\mathbf{1}$; //Edge is the search edge of all layers
3: Find MaxScore$_{k_{0,1}}$; // MaxScore$_{k_{0,1}}$ is the largest score of $k_{0,1}$
4: MaxScore = 0; // MaxScore is the largest score in MaxScoreinLayer
5: repeat
6:    $x = 1$;
7:    while $S(k_0^{(x)}, k_1^{(1)}, \mathrm{MaxScore}_{k_{0,1}}) > \mathrm{MaxScore}$ do
8:       $y = \mathrm{Edge}[x]$;
9:       while $S(k_0^{(x)}, k_1^{(y)}, \mathrm{MaxScore}_{k_{0,1}}) > \mathrm{MaxScore}$ do
10:         if $(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)})$ is not outputted and $S(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)}) >$ MaxScoreinLayer[x] then
11:            MaxScoreinLayer[$x$] = $S(k_0^{(x)}, k_1^{(y)}, k_{0,1}^{(z)})$;
12:            if MaxScoreinLayer[$x$] > MaxScore then
13:               MaxScore = MaxScoreinLayer[$x$] ;
14:            end if
15:         end if
16:         $y = y + 1$
17:      end while
18:      Edge[$x$] = $y$;
19:      $x = x + 1$;
20:   end while
21:   Output MaxScore to test;
22:   MaxScoreinLayer[$x$ of MaxScore]=0;
23:   MaxScore = Max(MaxScoreinLayer);
24:   Edge[$x$ of MaxScore] = 1;
25: until Enough key candidates have been generated
```

If only the independent score lists are used in this algorithm, this algorithm degenerates into a 2-dimensional KE algorithm with some difference from Algorithm 1. The time complexity should be similar since each combined key is still only checked once. The memory complexity is larger than that of Algorithm 1 since Rule 1 is only applied in 1 axis rather than 2 axes. Still the generality of the proposed 3-dimensional KE algorithm is a nice feature as a more generalized KE tool.

4.4 General Strict Key Enumeration Algorithm for AES-128

This section discusses the practical and efficient usage of the proposed efficient strict KE algorithm of Algorithm 3. The intuition is that with the same measurements, the attackers can target different types of key information to improve the attack result. As mentioned, for n subkeys, one can get $n(n+1)/2$ score lists including the subkeys and the subkey differences. We consider that the key enumeration using all these score lists are not reasonable for the following reasons.

– The computational cost (time and memory) is too large for such KE, which could reduce the overall efficiency of the post-processing. For AES-128, there are in total 136 score lists and the internal relations are too many to be performed in practice.

- According to the key recovery methods, the information in the recovered score lists are overlapping with each other. Adding new score lists may not always lead to better performance of key enumeration. For example, since the correlation calculation is transitive[1], the new information of the newly added score list could be decreasing due to the overlap. A comprehensive evaluation of this issue is actually a very interesting topic to exploit as a future work.

In each attack, the attackers have to decide a strategy of the KE according to its purpose. In this work, we decided to pick up a reasonable approach to perform the KE strategy for AES-128 as shown in Fig. 4. The KE algorithm basically covers 2 rows of the score lists triangle, which including 16 key bytes score lists and 15 key byte difference score lists. To achieve this 31-score-list KE algorithm, we only need to set all the rest score lists to full zeros.

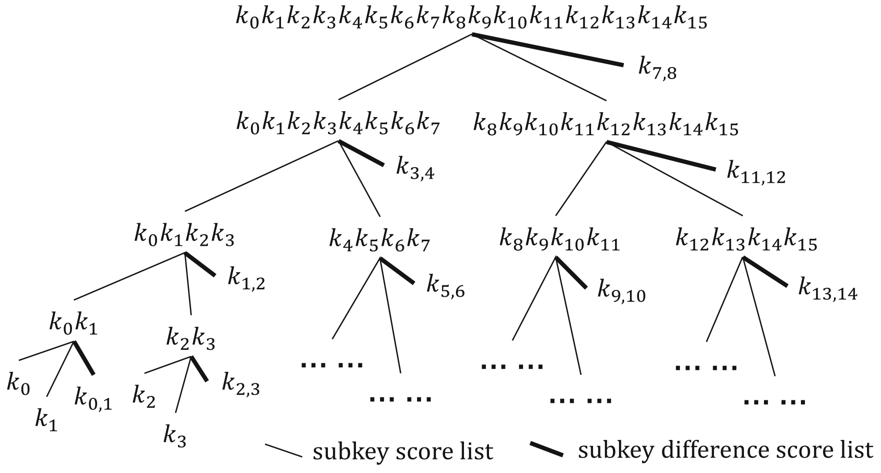

$k_0k_1k_2k_3k_4k_5k_6k_7k_8k_9k_{10}k_{11}k_{12}k_{13}k_{14}k_{15}$
$k_{7,8}$
$k_0k_1k_2k_3k_4k_5k_6k_7$
$k_8k_9k_{10}k_{11}k_{12}k_{13}k_{14}k_{15}$
$k_{3,4}$
$k_{11,12}$
$k_0k_1k_2k_3$
$k_4k_5k_6k_7$
$k_8k_9k_{10}k_{11}$
$k_{12}k_{13}k_{14}k_{15}$
$k_{1,2}$
$k_{5,6}$
$k_{9,10}$
$k_{13,14}$
k_0k_1
k_2k_3
k_0
k_1
$k_{0,1}$
k_2
k_3
$k_{2,3}$
subkey score list
subkey difference score list

Fig. 4. 31-score-list key enumeration of dependent score lists

5 Evaluations of Key Enumeration Algorithm Using Simulated Data

In this section, we investigate the impact of key enumeration to quantify the reduction in the data complexity of the key recovery. The side-channel leakage is

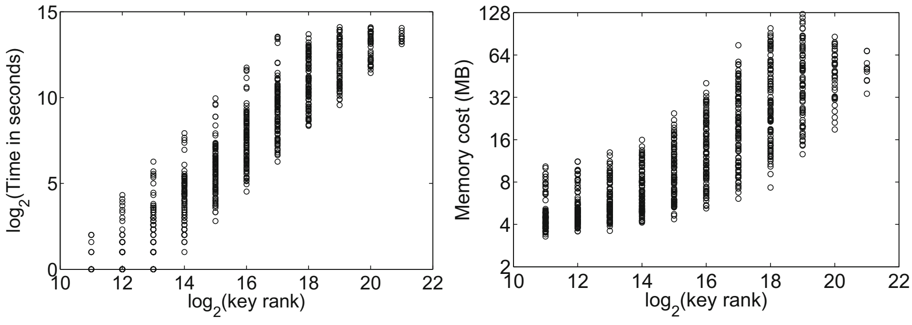
[1] For 3 random variables A, B and C, and denote their correlation coefficients are $\rho_{A,B}$, $\rho_{B,C}$ and $\rho_{A,C}$. It is known that $\rho_{A,B}$, $\rho_{B,C}$ gives an up-bound and a low-bound for $\rho_{A,C}$.

targeted the output of the S-boxes in the first AES round. All our experiments were performed using i7-6500 CPU running at 2.50 GHz with 8 GB of RAM on Windows 10. In our experiments, the score function is simply a summation calculation without any weight.

5.1 Complexity Evaluation

We first verify the performances of 31-score-list KE algorithm in terms of timing and memory. We repeated the KE algorithm for 100 times with a time limitation of 5 h. The time and memory cost for these 100 trails are shown in Fig. 5. As shown in Fig. 5, the KE algorithm can enumerate up to 2^{21} secret keys using about 5 h within 128 MB memory cost. In our evaluation, the memory cost of the program is measured using the API function of GetProcessMemoryInfo.

The memory-time trade-off can be adjusted for the proposed KE algorithm. For example, each layer can be divided into smaller blocks so that the algorithm stores the largest merged key of each block instead of each layer. Then more memory is used to reduce the time overhead. Also, for different levels of the recursive call, different polices of the memory-time balance can be applied to increase the overall performance.



(a) Time Overhead (b) Memory Cost

Fig. 5. Time overhead and memory cost of 31-score-list KE

5.2 Performance Evaluation for KE of Dependent Score Lists

The simulated leakages following a Hamming weight leakage model on the S-box output. Our simulation adds two types of noises in the leakage measurement, i.e. the leakage model noise and the measurement noise. We assume that the leakage model of the device is not perfectly the same with the leakage model (HW model) used by the attacker. Instead, the leakage model of device is constructed as

$$\text{LeakageModel}(\alpha) = \text{HW}(\alpha) + \mathcal{N}(0, 4^2), \tag{1}$$

for $\alpha \in \mathrm{GF}(2^8)$. To simulate the leakage measurement, we add an independent Gaussian noise to the measurement as well, i.e.

$$\mathrm{MesauredLeakage}(\alpha) = \mathrm{LeakageModel}(\alpha) + \mathcal{N}(0, 4^2). \quad (2)$$

The main reason to add a leakage model noise is to make sure that CPA and CECA have a similar attack efficiency.

For each set of measurement leakage, we applied 4 types of attack as (1) CPA using the correlation coefficient as the score to each key byte (2) CECA using the correlation coefficient as the score of each key byte difference (3) CPA+KE that uses the CPA score lists and the KE algorithm (4) CPA+CECA+KE that uses 31 score-lists from CPA and CECA, and the KE algorithm. For every KE algorithm, the search has a time limitation of 20 min. Note that for CPA+KE, we are using Algorithm 3 for the key enumeration in which the CECA lists are set to be all zeros.

Figure 6 illustrates the evolution of the success rate against the used leakage data for 4 types of key recovery attacks. We have the following observations.

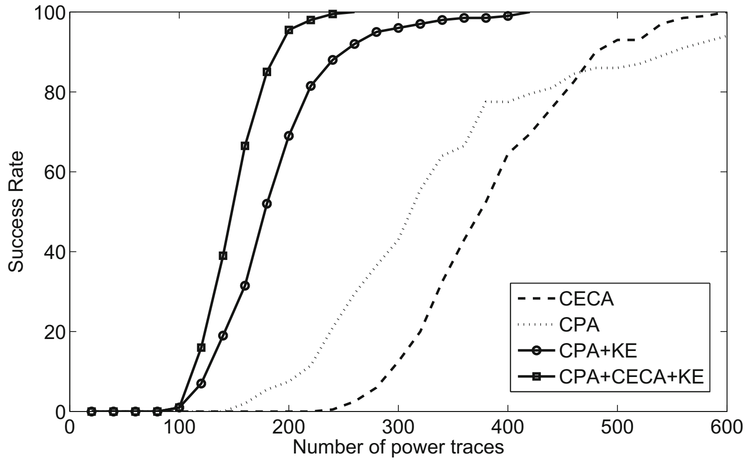


Fig. 6. Success rate curves for CPA, CECA, CPA+KE and CPA+CECA+KE.

- The simulation setting leads to a similar attack efficiency for CPA and CECA.
- Within the same search time, CPA+CECA+KE attack shows better attack results than CPA+KE. Both attack methods starts to recover the key around 100 traces. Using dependent score lists, the success rate reaches 1 with 250 traces, which is a large improvement compared to 400 traces without using 15 CECA score lists.

We also investigated the rank of the correct key for CPA+KE attack and CECA+CPA+KE attack. We first ignore the cases in which both attacks ranked the correct key the first. We also ignore the cases in which either one of the attack failed to enumerate the correct key within the time limitation. Then we compared the key rank of these two attacks. We calculated the rank improvement between the key rank in CECA+CPA and the key rank in CPA as $\log_2(\frac{\text{Rank}_{\text{CPA}}}{\text{Rank}_{\text{CECA+CPA}}})$. The distribution of the key rank improvement is shown in Table 1. In Table 1, the first line corresponds to the range of $\log_2(\frac{\text{Rank}_{\text{CPA}}}{\text{Rank}_{\text{CECA+CPA}}})$. The second line is the percentage of results within a certain range. The last line is the percentage of $\text{Rank}_{\text{CPA}} < \text{Rank}_{\text{CECA+CPA}}$ and $\text{Rank}_{\text{CPA}} > \text{Rank}_{\text{CECA+CPA}}$. It is clear that the key rank in CECA+CPA is statistically lower than that in CPA, which is a clear advantage of the 31-score-list KE.

Table 1. Distribution of Key Rank Improvement (bits), 16-list vs 31-list

$(\infty, -4)$	$[-4, -2)$	$[-2, 0)$	$[0, 2)$	$[2, 4)$	$[4, 6)$	$[6, 8)$	$[8, \infty)$
0.2%	0.4%	1.72%	40.11%	24.67%	13.32%	9.37%	10.22%
	2.32%				**97.68%**		

The efficiency of CPA-like attack is largely affected by the accuracy of the leakage model. In order to eliminate the difference in the nature of combined score lists, we further investigate two types of attacks as (1) CECA+KE attack using 15 independent subkey difference score lists and (2) CECA+KE attack using 15 + 14 dependent subkey difference score lists. From 120 score lists of subkey difference, the 15 independent subkey difference score lists correspond to $k_{0,1}, k_{1,2}, \ldots, k_{14,15}$. Correspondingly, the other 14 subkey difference score lists correspond to $k_{0,2}$,$k_{1,3}$,$\ldots$,$k_{13,15}$. The rest of the experiment is exactly the same with previous one. The evaluation of success rate is shown in Fig. 7, in which the attack using 29 lists shows clear advantage.

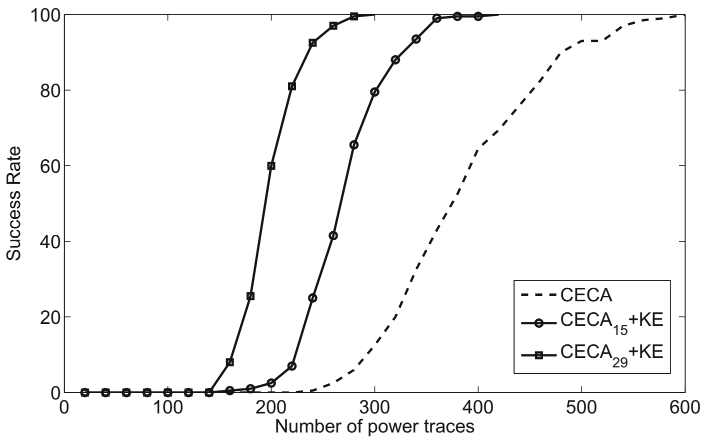


Fig. 7. Success rate curves for 15-list-CECA and 29-list-CECA.

As for the rank of the correct key, the distribution of the logarithm ratio between key rank using 15 lists and key rank using 29 lists is shown in Table 2, i.e. $\log_2(\frac{\text{Rank}_{15\text{CECALists}}}{\text{Rank}_{29\text{CECALists}}})$. Table 2 shows that the key rank in 29-list-CECA is statistically much lower than that in 15-list-CECA, which implies again that the more score lists are used in KE, the correct key can be ranked higher and the number of plaintext-ciphertext test can be reduced. For CECA, this reduction should be multiplied with a factor of 2^8 as the guess of k_0 in the plaintext-ciphertext test.

Table 2. Distribution of Key Rank Improvement (bits), 15-list vs 29-list

$(-\infty, 0)$	$[0, 1)$	$[1, 2)$	$[2, 3)$	$[3, 4)$	$[4, 5)$	$[5, 6)$	$[6, \infty)$
0.49%	20.42%	25.83%	13.16%	11.44%	9.59%	7.8%	11.07%
0.49%	**99.51%**						

6 Conclusions

This work proposed a strict but efficient key enumeration algorithm to combine the key-recovery score lists that are dependent with each other. First, we proposed a 3-dimensional strict key enumeration algorithm that shows much better performance than a straightforward extension of the previous 2-dimensional key enumeration algorithm. Second, this work generalized this 3-dimensional KE algorithm to form a 31-score-list key enumeration algorithm for AES-128. The KE algorithm can enumerate 2^{21} keys using 5 h and 128 MB of RAM using a normal PC. In the performance evaluation, we demonstrated that the correct key will be ranked much higher when dependent score lists are used in the key recovery. As a result, the success rate increases when more score lists are used. This improvement does not come from increased data complexity, which is significant to be noted. In the future work, we'd like to compare our proposal with other state-of-the-art key enumeration algorithms. This paper presented a new type of tool in the post-processing of SCA, whose usage scenario requires more comprehensive study in the future.

Acknowledgement. This research was supported by National Natural Science Foundation of China (No. 61602239) and Jiangsu Province Natural Science Foundation (No. BK20160808).

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
2. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
3. Glowacz, C., Grosso, V., Poussier, R., Schüth, J., Standaert, F.-X.: Simpler and more efficient rank estimation for side-channel security assessment. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 117–129. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_6
4. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 390–406. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_25
5. Martin, D.P., O'Connell, J.F., Oswald, E., Stam, M.: Counting keys in parallel after a side channel attack. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 313–337. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_13
6. Poussier, R., Standaert, F.-X., Grosso, V.: Simple key enumeration (and rank estimation) using histograms: an integrated approach. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 61–81. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_4
7. Wang, S., Li, Y., Wang, J.: A new key rank estimation method to investigate dependent key lists of side channel attacks. In: IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST). IEEE (2017)
8. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
9. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_9
10. Meier, W., Staffelbach, O.: Analysis of pseudo random sequences generated by cellular automata. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 186–199. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_17
11. Pan, J., van Woudenberg, J.G.J., den Hartog, J.I., Witteman, M.F.: Improving DPA by peak distribution analysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 241–261. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_17
12. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
13. Bogdanov, A., Kizhvatov, I., Manzoor, K., Tischhauser, E., Witteman, M.: Fast and memory-efficient key recovery in side-channel attacks. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 310–327. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31301-6_19

14. David, L., Wool, A.: A bounded-space near-optimal key enumeration algorithm for multi-subkey side-channel attacks. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 311–327. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_18
15. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39887-5_16
16. Moradi, A.: Side-channel leakage through static power. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 562–579. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_31
17. Roche, T., Lomné, V.: Collision-correlation attack against some 1st-order Boolean masking schemes in the context of secure devices. In: Prouff, E. (ed.) COSADE 2013. LNCS, vol. 7864, pp. 114–136. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40026-1_8
18. Moradi, A., Mischke, O., Paar, C., Li, Y., Ohta, K., Sakiyama, K.: On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 292–311. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_20
19. Schellenberg, F., Finkeldey, M., Gerhardt, N., Hofmann, M., Moradi, A., Paar, C.: Large laser spots and fault sensitivity analysis. In: IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 203–208. IEEE (2016)

A Novel Use of Kernel Discriminant Analysis as a Higher-Order Side-Channel Distinguisher

Xinping Zhou[1,2](✉), Carolyn Whitnall[3], Elisabeth Oswald[3], Degang Sun[1,2], and Zhu Wang[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, People's Republic of China
{zhouxinping,sundegang,wangzhu}@iie.ac.cn

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, People's Republic of China

[3] Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol BS8 1UB, UK
{Carolyn.Whitnall,Elisabeth.Oswald}@bristol.ac.uk

Abstract. Distinguishers play an important role in Side Channel Analysis (SCA), where real world leakage information is compared against hypothetical predictions in order to guess at the underlying secret key. However, the direct relationship between leakages and predictions can be disrupted by the mathematical combining of d random values with each sensitive intermediate value of the cryptographic algorithm (a so-called "d-th order masking scheme"). In the case of software implementations, as long as the masking has been correctly applied, the guessable intermediates will be independent of any one point in the trace, or indeed of any tuple of fewer than $d+1$ points. However, certain $d+1$-tuples of time points may jointly depend on the guessable intermediates. A typical approach to exploiting this data dependency is to pre-process the trace – computing carefully chosen univariate functions of all possible $d+1$-tuples – before applying the usual univariate distinguishers. This has a computational complexity which is exponential in the order d of the masking scheme. In this paper, we propose a new distinguisher based on Kernel Discriminant Analysis (KDA) which directly exploits properties of the mask implementation without the need to exhaustively pre-process the traces, thereby distinguishing the correct key with lower complexity. Experimental results for 2nd and 3rd order attacks (i.e. against 1st and 2nd order masking) verify that the KDA is an effective distinguisher in protected settings.

Keywords: Kernel Discriminant Analysis
Higher-order side channel analysis · Side channel distinguisher

X. Zhou—This work was done while the author was in the Department of Computer Science, University of Bristol.

1 Introduction

Protecting sensitive information from attacks exploiting the physically observable characteristics of cryptographic devices in operation has been a key aim for vendors and evaluation labs ever since the devastating effectiveness and simplicity of such attacks began to become apparent with the work of Kocher et al. in the late 1990s [11]. Software countermeasures such as masking [8] successfully disrupt the relationship between sensitive intermediate values and single points of observed leakage – precisely the trace feature that Differential Power Analysis (DPA) in particular targets[1]. However, *tuples* of points of size greater than the number of masks d can still *jointly* depend on the sensitive intermediates. This gives rise to so-called 'higher order' DPA [14], which typically proceeds by combining multiple points via some (non-linear) pre-processing function before applying a standard DPA distinguisher – essentially treating the pre-processed traces in a univariate manner, albeit with an exponential (in d) increase in the impact of noise relative to a 'first order' attack [22].

Aside from the greater data complexity implied by the inflated noise, higher-order attacks are also hampered by the increasing difficulty of locating the leaking tuples. The computational complexity of an 'exhaustive search' approach – in which all possible point combinations are computed and analysed – grows exponentially with d. Heuristics exist to reduce the search problem by placing informed restrictions on the regions of the trace to be iteratively explored [9] but, precisely because of their heuristic nature, these do not guarantee to find the best (or indeed any) exploitable combinations. A recent proposal (presented at Cardis 2016 [6]) aims to bypass the need for explicit enumeration of the $(d+1)$-tuples without recourse to heuristics, using Kernel Discriminant Analysis (KDA) [15].

KDA is a generalisation of Linear Discriminant Analysis (LDA), a statistical method to find linear combinations of features (i.e. variables in a dataset, or points in a trace) that characterise class separations. In particular, it outputs projection directions that maximise the ratio of between-group to within-group scatter, so that 'interesting' variation may be concentrated into a reduced-dimension space for further analysis. LDA has been promoted as one of a number of methods to extract sensitive data dependent features from side-channel traces for some years (beginning with [24], to the best of our knowledge). However, because it only finds linear combinations, it is unable to locate the types of *joint* data dependencies exhibited by traces which have been protected by software masking. By contrast, the 'kernel trick' employed by KDA allows to implicitly map the data into a higher dimensional feature space within which to perform the discriminant analysis, thereby extracting non-linear combinations of the sort that (in the case of DPA) *do* yield sensitive information on further analysis. Because the mapping of the tuple candidates need not be computed explicitly

[1] Hardware masking schemes also exist, which process shares in parallel but shift the exploitable leakage into higher moments of the (univariate) trace distributions [18].

(by contrast with the preprocessing required by established higher order DPA methodologies), its complexity is polynomial, rather than exponential, in d.

However, another recent development in the literature has been to demonstrate the direct applicability of LDA as a side-channel *distingisher*, not just a pre-processing method. In this capacity, it shares the advantages of other 'partition-based' [25] (aka 'nominal power model' based [30]) distinguishers – namely that it operates needing only a clustering of the intermediates into similarly leaking classes, rather than (e.g.) a proportional approximation of the leakage as would be necessary for a correlation DPA attack. It is therefore natural to suppose that KDA can similarly be extended for use as a distinguisher, with the same flexibility advantages over higher-order correlation DPA that LDA has over first-order correlation, as well as the reduction in complexity with respect to d. Indeed, in the following, we confirm that this is the case – KDA can be used, not just to locate the interesting leakage prior to an attack, but as a side-channel distinguisher in its own right. We show how to achieve this, provide experimental validation of the effectiveness of our methodology, and reason about its potential as well as its drawbacks.

1.1 Outline

The rest of the paper proceeds as follows. Section 2 covers the preliminaries of (higher-order) SCA, LDA and KDA. In Sect. 3 we describe the natural connection between KDA and the higher-order SCA problem, and present a methodology to extract sensitive information using KDA, before going on to experimentally verify its effectiveness. Section 4 discusses the efficiency and advantages (and drawbacks) of our proposed approach, and Sect. 5 concludes the paper.

2 Preliminaries

2.1 Differential Power Analysis

We consider a 'standard DPA attack' scenario as defined in [13], and briefly explain the underlying idea as well as introduce the necessary terminology here. We assume that the power consumption $\mathbf{P} = \{P_1, ..., P_T\}$ of a cryptographic device (as measured at time points $\{1, ..., T\}$) depends, for at least some $\boldsymbol{\tau} \subset \{1, ..., T\}$, on some internal value (or state) $F_{k^*}(X)$ which we call the *target*: a function $F_{k^*} : \mathcal{X} \to \mathcal{Z}$ of some part of the known plaintext—a random variable $X \overset{R}{\in} \mathcal{X}$—which is dependent on some part of the secret key $k^* \in \mathcal{K}$. Consequently, we have that $P_t = L_t \circ F_{k^*}(X) + \varepsilon_t, t \in \boldsymbol{\tau}$, where $L_t : \mathcal{Z} \to \mathbb{R}$ describes the data-dependent leakage function at time t and ε_t comprises the remaining power consumption which can be modeled as independent random noise (this simplifying assumption is common in the literature—see, again, [13]). The attacker has N power measurements corresponding to encryptions of N known plaintexts $x_i \in \mathcal{X}$, $i = 1, \ldots, N$ and wishes to recover the secret key k^*. The attacker can accurately compute the internal values as they would be under each key hypothesis $\{F_k(x_i)\}_{i=1}^N$, $k \in \mathcal{K}$ and uses whatever information he possesses about the true leakage functions L_t to construct a prediction model (or models) $M_t : \mathcal{Z} \to \mathcal{M}_t$.

A distinguisher D is some function which can be applied to the measurements and the hypothesis-dependent predictions in order to quantify the correspondence between them, the intuition being that the predictions under a correct key guess should give more information about the true trace measurements than an incorrect guess. For a given such comparison statistic, D, the *theoretic* attack vector is $\mathbf{D} = \{D(L \circ F_{k^*}(X) + \varepsilon, M \circ F_k(X))\}_{k \in \mathcal{K}}$, and the *estimated* vector from a practical instantiation of the attack is $\hat{\mathbf{D}}_N = \{\hat{D}_N(L \circ F_{k^*}(\mathbf{x}) + \mathbf{e}, M \circ F_k(\mathbf{x}))\}_{k \in \mathcal{K}}$ (where $\mathbf{x} = \{x_i\}_{i=1}^N$ are the known inputs and $\mathbf{e} = \{e_i\}_{i=1}^N$ is the observed noise). Then the attack is *o-th order theoretically successful* if $\#\{k \in \mathcal{K} : \mathbf{D}[k^*] \leq \mathbf{D}[k]\} \leq o$ and *o-th order successful* if $\#\{k \in \mathcal{K} : \hat{\mathbf{D}}_N[k^*] \leq \hat{\mathbf{D}}_N[k]\} \leq o$.

2.2 Masking

Since the scale of the threat of (first-order) DPA began to emerge [11], many countermeasure schemes have been proposed. The principle behind masking is to split the sensitive intermediate values $s = F_{k^*}(x)$ into $d+1$ shares $(r_0, ..., r_d \in \mathcal{Z})$ satisfying the relation[2]

$$s = r_0 \otimes r_1 \otimes ... \otimes r_d$$

where the $\otimes$ operation is the bitwise addition (or XOR) in the common case of Boolean masking. One of the shares, e.g. r_0, is sometimes referred to as the 'masked variable', with the other shares, $(r_1, ..., r_d)$ then viewed as the 'masks'. For a masking scheme to be sound, it is usually required that the masks are uniformly and independently generated from $\mathcal{Z}$. In the case of software implementations, which we focus on here, the shares are processed in sequence so that side-channel leakages are distributed across multiple points in the measured traces.

Classical Higher-Order DPA. In the case of a masked implementation, the leakage of the shares corresponding to the sensitive value s is

$$\mathbf{l} = (l_0, l_1, ..., l_d)$$

where

$$\begin{aligned} l_0 &= L_0 \circ (s \oplus r_1 \oplus \ldots \oplus r_d) + \varepsilon_0 \\ l_i &= L_i \circ (r_i) + \varepsilon_i, \qquad \text{for } 1 \leq i \leq d. \end{aligned}$$

It can be seen that no single component of the leakage $\mathbf{l}$ directly relies on s. The first order distinguisher will be unable to learn anything about the secret key k^* in this case.

[2] This relation exists implicitly even when it doesn't manifest directly in the cryptographic algorithm.

During a higher-order DPA, an attacker extracts information about k^* by monitoring the leakage of the unknown shares. Generally speaking, the dth order masking scheme can be attacked by a $(d+1)$th order attack. Since it is difficult for the attacker to precisely determine the location of l_i, we assume that ℓ time point candidates can be discovered for each share by some reverse engineering or *a priori* knowledge about the masked implementation. Thus, $(d+1)$-tuples of ℓ time points are available for analysis. To analyse the $(d+1)\ell$ time points by classical higher-order DPA, a 'combination function' is required – although it has been observed that such an approach inevitably incurs loss of information [17,27]. The most popular combination function is probably the normalised product, shown in [19] to be the optimal choice in the idealised setting of a correlation attack against Hamming weight leakage with Gaussian noise; other proposals include the absolute difference [14], and some more complex expressions involving sine functions [17].

Note that the combination function operates as a pre-processing procedure on all possible $(d+1)$-tuples of time points. This implies ℓ^{d+1} computations, resulting in ℓ^{d+1} points for analysis via a first order distinguisher D (typically correlation [3]) paired with a power model (recall Sect. 2.1).

2.3 Kernel Discriminant Analysis

Linear Discriminant Analysis. Linear Discriminant Analysis (LDA) is a widely-used (supervised) dimensionality reduction method. It seeks the directions along which the projection of a dataset displays large between-cluster distances and small within-cluster distances. Suppose $\mathbf{P}_i$ is row vector of a matrix $\mathbf{P} \in \mathbb{R}^{N\times U}$ with labels $\mathbf{m} \in \mathbb{R}^{N\times 1}$; then the LDA problem amounts to finding ω to maximize $J(\omega)$ in (1):

$$J(\omega) = \frac{\omega^T S_B \omega}{\omega^T S_W \omega} \tag{1}$$

This procedure is equivalent to solving (2)

$$S_B\omega = \lambda S_W \omega \tag{2}$$

where S_B and S_W represent the between-cluster and within-cluster scatter matrices given by (3) and (4) respectively

$$S_B = \sum_{m\in\mathcal{M}} n_m \left(\frac{1}{n_m}\sum_{m_i=m}\mathbf{P}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_i\right)^T \left(\frac{1}{n_m}\sum_{m_i=m}\mathbf{P}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_i\right) \tag{3}$$

$$S_W = \sum_{m\in\mathcal{M}}\sum_{m_i=m}\left(\mathbf{P}_i - \frac{1}{n_m}\sum_{m_i=m}\mathbf{P}_i\right)^T \left(\mathbf{P}_i - \frac{1}{n_m}\sum_{m_i=m}\mathbf{P}_i\right) \tag{4}$$

where $n_m = \#\{i|m_i = m\}$, i.e. the number of observations in the data set for which the label is m (for details see [10]). Performing LDA amounts to calculating the generalized eigenvalues $\lambda_1, \dots, \lambda_U$ (ordered from largest to smallest) and the corresponding generalized eigenvectors $\omega_1, \dots, \omega_U$.

The applications of LDA to SCA are two-fold. On the one hand, it can be used for dimensionality reduction, addressing the problem of 'interesting point selection' by (hopefully) projecting relevant leakage information along a small number of directions prior to further analysis [5,24]. It has been shown to be the optimal strategy for this purpose – at least in the case of unprotected implementations, where the leakage of sensitive intermediates resides in the marginal distributions of single trace points [4]. The procedure is as follows: sort the total power consumption $\{\mathbf{P}_i\}_{i=1}^N$ into different clusters $\{\{\mathbf{P}_i\} \,|M \circ F_k(x_i) = m\}$ under the correct key k^* and power model M; perform LDA on the labeled clusters; extract the eigenvectors $\omega_1, \dots, \omega_u \quad (u \leq U)$ corresponding with the first u largest eigenvalues, i.e. the u 'best' projected directions. Projecting the data along these u directions produces a dataset of lower dimension but with minimal information loss.

On the other hand, it has also recently been proposed for use directly as a DPA distinguisher [12]. To this end it operates as follows: sort the total power consumption $\{\mathbf{P}_i\}_{i=1}^N$ into different clusters $\{\{\mathbf{P}_i\} \,|M \circ F_k(x_i) = m\}$ under the key hypothesis k and power model M; perform LDA on the labeled clusters; extract the first (largest) generalized eigenvalue as the distinguisher score for the key hypothesis. This strategy takes advantage of the fact that, for a correct key guess, the arrangement produced by the power model should correspond with the true cluster structure of the leakage measurements, so that the indicator value stands out by comparison with the wrong key guesses.

Discriminant Analysis with Kernels. LDA can be used to find optimal linear mappings of high dimensional data but is not applicable when the relevant information is known to be contained in non-linear combinations of points, as is the case (e.g.) for side-channel leakages of masked implementations. To extend LDA to the non-linear case, we consider the problem in a feature space $\mathcal{F}$ induced by some mapping function (this mapping process is implicit as will be seen in the following subsection), $\Phi : R^n \to \mathcal{F}$. KDA [15] is used to find non-linear directions by first mapping the data non-linearly by Φ into some feature space $\mathcal{F}$ within which to compute linear discriminants, thus implicitly yielding a non-linear discriminant in the input space. To find such a discriminant, the goal (1) is replaced with:

$$J(\omega') = \frac{\omega'^T S_B^{\Phi} \omega'}{\omega'^T S_W^{\Phi} \omega'} \tag{5}$$

where $\omega' \in \mathcal{F}$ and S_B^{Φ} and S_W^{Φ} are the corresponding matrices in $\mathcal{F}$.

$$S_B^{\Phi} = \sum_{m \in \mathcal{M}} n_m \Big(\frac{1}{n_m} \sum_{m_i = m} \mathbf{P}_i^{\Phi} - \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}_i^{\Phi}\Big)^T \Big(\frac{1}{n_m} \sum_{m_i = m} \mathbf{P}_i^{\Phi} - \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}_i^{\Phi}\Big) \tag{6}$$

$$S_W^{\Phi} = \sum_{m \in \mathcal{M}} \sum_{m_i = m} \Big(\mathbf{P}_i^{\Phi} - \frac{1}{n_m} \sum_{m_i = m} \mathbf{P}_i^{\Phi}\Big)^T \Big(\mathbf{P}_i^{\Phi} - \frac{1}{n_m} \sum_{m_i = m} \mathbf{P}_i^{\Phi}\Big) \tag{7}$$

where $\mathbf{P}_i^{\Phi}$ is $\Phi(\mathbf{P}_i)$ projection of $\mathbf{P_i}$ on $\mathcal{F}$ by Φ. For a properly chosen Φ, an inner product $< \cdot, \cdot >$ can be defined on $\mathcal{F}$, which makes for a so-called 'reproducing kernel Hilbert space',

$$K(\mathbf{x}, \mathbf{y}) =< \Phi(\mathbf{x}), \Phi(\mathbf{y}) >$$

where K is known as the kernel function. Widely-used kernel functions include the Gaussian kernel $K(\mathbf{x}, \mathbf{y}) = exp(-||\mathbf{x} - \mathbf{y}||^2/c)$ ($|| \cdot ||$ is the 2-norm), and the polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^{d'}$, for positive constants c and d' satisfying Mercer's condition [20], as defined in [21].

Procedure of KDA. Generally, given a labelled data set $\{\mathbf{P}_i\}_{i \in [1,N]}$, the corresponding labels m_i (for simplicity, we use the notation $\mathbf{P}_i^{m_i}$ to denote the label of ith data sample is m_i), and the kernel function $K(\mathbf{x}, \mathbf{y})$, the KDA procedure can be briefly summarised as follows (for more details about the derivation, see [15]):

1. Calculate the between-class scatter matrix $\mathbf{M} \in R^{N \times N}$
$$\mathbf{M} = \sum_{m \in \mathcal{M}} n_m (\mathbf{M}_m - \mathbf{M}_*)(\mathbf{M}_m - \mathbf{M}_*)^T$$
where $\mathbf{M}_m$ and $\mathbf{M}_*$ are $N \times 1$ size column vectors given by
$$(\mathbf{M}_m)_j = \frac{1}{n_m} \sum_{m_i = m} K(\mathbf{P}_j, \mathbf{P}_i^{m_i})$$
$$(\mathbf{M}_*)_j = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{P}_j, \mathbf{P}_i).$$
2. Then calculate the within-class scatter matrix $\mathbf{N} \in R^{N \times N}$ given by
$$\mathbf{N} = \sum_{m \in \mathcal{M}} \mathbf{K}_m (\mathbf{I}_{n_m} - \mathbf{1}_{n_m}) \mathbf{K}_m^T$$
where $\mathbf{K}_m$ is an $N \times n_m$ matrix with $(\mathbf{K}_m)_{ij} = K(\mathbf{P}_i, \mathbf{P}_j^{m_j = m})$ (this is the kernel matrix for class m), $\mathbf{I}_{n_m}$ is the $n_m \times n_m$ size identity matrix, and $\mathbf{1}_{n_m}$ is the $n_m \times n_m$ matrix with all entries $1/n_m$.
3. The eigenvalues $\lambda_1', \ldots, \lambda_{N'}'$ ($N' \leq N$) and the eigenvectors $\alpha_1, \ldots, \alpha_{N'}$ can be extracted by solving
$$\mathbf{N}^{-1} \mathbf{M} \alpha_i = \lambda_i' \alpha_i. \tag{8}$$
4. Since the matrix $\mathbf{N}$ may be singular, it needs regularizing prior to Step 3, which is done by setting
$$\mathbf{N} = \mathbf{N} + \mu \mathbf{I}$$
for some positive μ.
5. Then the projection of $\mathbf{P}$ onto ω_i' is given by
$$< \omega_i', \Phi(\mathbf{P}) >= \sum_{j=1}^{N} \alpha_i(j) K(\mathbf{P}_j, \mathbf{P}). \tag{9}$$

Generally, KDA is used to transform U-dimensional data into C-dimensional data by taking the C eigenvectors $\alpha_1, \ldots, \alpha_C$ with the C largest eigenvalues and using Eq. (9) for the projection step.

KDA has been introduced to SCA as a tool for information extraction[3] in the presence of masking [6]. The authors sort the training data set into different clusters according to the sensitive intermediate value under a known power model, the plaintext and the known key. Then KDA is performed on these clusters to calculate the eigenvectors and corresponding eigenvalues. The two eigenvectors with the two largest eigenvalues are chosen as the projection directions (i.e. C is set to be 2) and used to transform profiling and attack acquisitions prior to performing a template attack.

3 Methodology

In this section, we introduce our proposed distinguisher to the setting of masked implementations, and analyse the method theoretically and empirically.

Due to the successful removal of sensitive intermediate values by masking, classical higher-order side-channel attacks typically proceed by first transferring the original trace points into a new space using a non-linear combination function (CF)[4]. Then, 'first order' distinguisher scores are computed in the new space. In fact, the KDA method combines these two processes (summarised in Figs. 1 and 2) without performing the non-linear mapping explicitly (the kernel trick embeds it implicitly).

$$\begin{tikzcd} R^{(d+1)\ell} \arrow[r, "CF"] & R^{\ell^{d+1}} \arrow[r, "D"] & k^* \end{tikzcd}$$

Fig. 1. Classical higher-order SCA.

$$\begin{tikzcd} R^U \arrow[r, "\Phi"] \arrow[rr, bend right, "KDA"'] & \mathcal{F} \arrow[r, "LDA"] & R^C \end{tikzcd}$$

Fig. 2. Process of KDA.

The calculations of between-class scatter matrix $\mathbf{M}$ and the within-class matrix $\mathbf{N}$ are based on the category (see procedure of KDA). And the maximum eigenvalue in Eq. (8) can be regarded as an indicator of dispersion degree of between-class and within-class (as can be seen in Eq. (1, 2 and 5)). Thus,

[3] Information extraction is typically understood to refer collectively to the similar but non-identical tasks of dimensionality reduction and interesting point selection.

[4] The combination functions mentioned in Subsect. 2.2 all are non-linear.

based on different (correct or wrong) categories, the dispersion degrees will differ. Therefore, the largest KDA eigenvalue functions as an effective distinguisher for attacks against masked intermediates.

3.1 General Approach

Let $\{x_i\}_{i=1}^{N}$ be the known plaintexts (or ciphertexts) associated with a set of trace measurements $\{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N\}$ each containing $d\ell$ time points as a dth-order masked implementation encrypts (or decrypts) the x_i. The power model mapping is $M : \mathcal{Z} \rightarrow \mathcal{M}$ and the kernel function is chosen as[5] $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^{d'}$ (in this paper we set the degree d' of the polynomial kernel function equal to the number of shares $d + 1$ into which each sensitive intermediate is divided). $\mathcal{D}$ denote the KDA distinguisher.

1. For each key hypothesis $k \in \mathcal{K}$, do the following:
   (a) Calculate the intermediate value $z_i = F_k(x_i)$ for each plaintext.
   (b) Map z_i to a power model prediction m_i, given by $M(z_i)$.
   (c) Compute the between-class scatter matrix $\mathbf{M}$ and the within-class scatter matrix $\mathbf{N}$, and regularize $\mathbf{N}$ by $\mathbf{N} = \mathbf{N} + \mu\mathbf{I}$.
   (d) Eigen-decompose the matrix $\mathbf{N}^{-1}\mathbf{M}$. Return the largest eigenvalue as the distinguisher score $\mathcal{D}_k$ for k.
2. Rank the pairs $(k, \mathcal{D}_k)$ according to $\mathcal{D}_k$.
3. Output the key hypothesis k with the largest $\mathcal{D}_k$ as the best guess on the true subkey.

3.2 Theoretical Rationale

In this subsection, we reason about the ‘soundness’ of the KDA distinguisher. We consider a distinguisher to be ‘sound’ if, given a sufficient sample of leakages and a ‘meaningful’ power model[6], it reduces the entropy on the unknown secret key. From an empirical perspective, soundness can be confirmed by observing a reduction in the mean key rank as the number of traces increases.

The essence of KDA is to transfer the raw data into a new, higher-dimensional space via an implicit non-linear projection, then compute the linear discriminant in the new space. This parallels the process of higher-order attacks in SCA. Thus, the soundness of KDA derives from (1) the effectiveness of the implicit projection, and (2) the effectiveness of LDA as a distinguisher in the first-order scenario.

The effectiveness of the projection has been recently demonstrated by the successful use of KDA to extract exploitable side-channel information [6]. Meanwhile, Mahmudlu *et al.* [12] have shown that the largest eigenvalue, which measures the (optimised) between- to within-scatter matrix ratio under a particular

[5] We only test this one example kernel function in our analysis; others, such as Gaussian kernel, are also available and may be effective.

[6] I.e. one that approximates some true aspect of the leakages; see, e.g. [30].

key guess, is typically higher for a correct guess (which produces a meaningful labelling on the traces) than an incorrect one (which produces a random labelling), thereby functioning as an effective distinguishing score.

It therefore seems reasonable to expect our proposed KDA distinguisher to be sound; the following experiments are designed to verify this.

3.3 Experimental Validation

We here present the outcomes of several experiments on simulated leakages and (in the case of second-order attacks only) on traces from real implementations to verify the soundness of KDA distinguisher.

We simulate multivariate leakages pertaining to shared intermediates in the presence of Gaussian noise. The basic principle is to add multivariate Gaussian noise ε to the hypothetical data-dependent consumption of the intermediate z,

$$l = M(z) + \varepsilon_G \tag{10}$$

where M is the leakage model (chosen to be the Hamming weight for the following), and z is the intermediate value. The Gaussian noise ε_G has zero mean and a covariance $\mathbf{\Sigma}$ given by,

$$\mathbf{\Sigma} = \mathbf{Q} * \rho * \mathbf{Q} \tag{11}$$

where $\mathbf{Q}$ is a diagonal matrix whose diagonal elements are the noise standard deviation σ and ρ is a co-correlation matrix estimated from real power traces.

For a dth order masked implementation, we simulate a trace of $d+1$-tuples of ℓ points with the secret key k^* as follows:

1. Generate $d+1$ random numbers $(x, r_1, r_2, ..., r_d)$ (the first random number, x, is the plaintext; the rest are masks).
2. For the first ℓ points in the trace, the intermediate values are the output of the XOR between the sensitive intermediate values $s = Sbox(x \oplus k^*)$ and the masks.
3. For the ith $(2 \leq i \leq d+1)$ sub-part of the trace, the intermediate value is r_{i-1}.

The Hamming weights of these intermediates are computed and additively combined with simulated noise samples of the specified Gaussian structure.

The real power traces are taken from the DPA Contest v4 [1]. The target is an 8-bit AVR microcontroller Atmega163 embedded in a smartcard. It contains 16 Kb of in-system programmable flash, 512 bytes of EEPROM, 1 Kb of internal SRAM and 32 general purpose working registers. The smartcard is read using a simple reader interface mounted on SASEBO-W board and controlled by Xilinx Spartan-VI FPGA. The traces are acquired using a LeCroy WaveRunner 6100 A oscilloscope using an EM probe. The acquisition bandwidth is 200 MHz and the sampling rate FS = 500 MS/s.

In the following experiments, the kernel function is $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^{d+1}$.

Second-Order Attacks. First, we perform second-order attacks on the simulated leakage. To keep the running time reasonable, we simulate 200,000 traces, and set ℓ to 5 so that the traces are 10 time points long. The noise deviation σ of the trace is set equal to 1, and μ for the KDA regularisation is set to be 100,000[7]. We use the second-order KDA distinguisher with the Hamming weight power model to attack the traces, the results of all key candidate distinguisher scores are shown in Fig. 3. The red line indicates the correct key. We can clearly see that, from 800 traces on, the distinguisher score associated with the correct key gradually separates from the scores for the alternative candidates, standing out first from 1500 traces onwards.

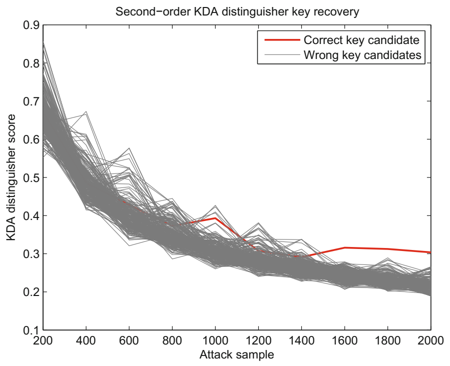


Fig. 3. Second-order attack with KDA on the simulated masked implementation leakage, with $\sigma = 1$. (Color figure online)

Figure 3 just shows the example result of a single trial; it cannot be interpreted as a stable indicator of the typical behaviour of the KDA distinguisher. To evaluate the performance of KDA we repeat the experiment multiple times (randomly selecting from the pool of 200,000 traces in each repetition). Our chosen evaluation metric is the Guessing Entropy [26], estimated from the average rank of the correct subkey. The result can be seen in Fig. 4. The red line indicates the second-order KDA distinguisher using a Hamming weight power model. The mean rank of the correct key decreases as the number of attack samples increases, converging to 1 after about 1900 traces.

Additionally, to further extend the experiments on KDA, we drop the Hamming weight assumption and investigate the performance of KDA when the attack samples are simply partitioned according to the least significant bit of the intermediate value[8]. Thus, the traces are separated into two clusters, labelled 0 and 1. The results are represented by the blue line in Fig. 4: the mean rank of

[7] $\mu = 100,000$ might not be the optimal one; we leave the optimisation μ as further work.

[8] Sometimes referred to as the 'LSB model'.

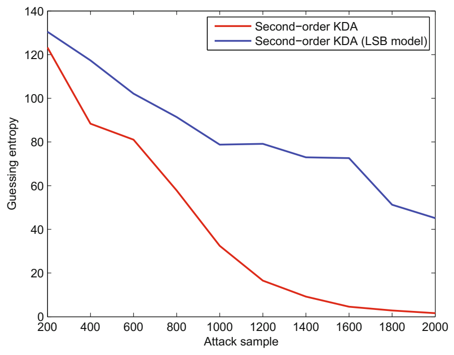


Fig. 4. Guessing entropy of second-order attack with KDA on the simulated masked implementation leakage, with $\sigma = 1$. (reps: 100) (Color figure online)

the correct key decreases as the number of traces increases, implying that the KDA distinguisher remains sound under this simpler power model.

Second, we test the performance of KDA against the real power traces from the DPA contest v4. The mask scheme implemented is the Rotating S-boxes Masking (RSM; for details, see [16]). The RSM scheme involves random masks and random offsets. There already exist several methods to attack these traces, as shown on the website [1]; we don't promote our KDA distinguisher as the optimal one, we simply make use of the data as a scenario in which to demonstrate its effective performance. We focus only on the second-order attack. It is a characteristic of the RSM scheme that the output of the masked S-box and the masked value of next sub-plaintext have the same mask, so that their XOR result can remove the mask. In detail, the first part is $MSbox(x_i \oplus k \oplus r_{i+offset})$, and the second part is $x_{i+1} \oplus r_{i+1+offset}$ where $MSbox$ is the masked sbox, i is the index of the sub-plaintext, $offset$ is a random number, and r is a mask table. According to the description of the RSM algorithm, the first part can be expressed as

$$MSbox(x_i \oplus k \oplus r_{i+offset}) = Sbox(x_i \oplus k) \oplus r_{i+1+offset}$$

Hence, the XOR result of the two parts is $Sbox(x_i \oplus k) \oplus x_{i+1}$ which, although slightly different to the intermediates targeted in the simulated leakage scenario, can be computed for each given key guess.

We choose 10 time points for each part as guided by a preliminary investigation of the traces. As for the previous experiment, we ran the second-order KDA distinguisher 100 times with randomly selected sub-samples of the traces. The guessing entropy results are presented in Fig. 5. We observe that, for a sufficient number of power traces, KDA (with a Hamming weight power model) can successfully recover the secret key.

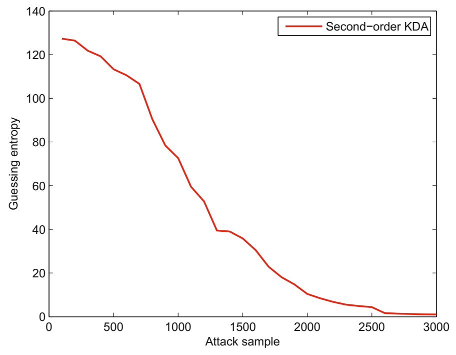


Fig. 5. Guessing entropy of second-order attack with KDA on DPA v4. (reps: 100)

Third-Order Attacks. In the previous subsection, we verified that KDA can indeed be used as an effective distinguisher for second-order attack. In this subsection, we attempt to extend the KDA into a higher-order attack.

We test the performance of third-order KDA in the simulated leakage scenario. We once more simulate 200,000 traces, but this time with a standard deviation of 0.01. We run third-order KDA to attack the traces 100 times; the guessing entropy of the correct key is indicated in Fig. 6. It decreases as the attack sample increases, as before, converging eventually to 1.

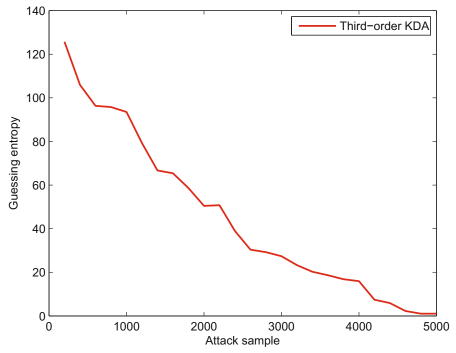


Fig. 6. Guessing entropy of third-order attack with KDA on the simulated masked implementation leakage, with $\sigma = 0.01$. (resp: 100)

4 Discussion

4.1 Complexity Analysis

Let N be the number of power traces, ℓ be the length of the sub-part relevant to the mask in the trace, and d the number of masks. Thus, for simplicity, the length of the trace is $(d+1)\ell$. The analysis which follows includes computational complexity and space complexity.

According to the general approach introduced in Sect. 3.1 and the KDA procedure in Sect. 2.3, for a key guess, the KDA distinguisher needs to compute two parts: the kernel between-class scatter matrix $\mathbf{M}$ and within-class scatter matrix $\mathbf{N}$, and the eigenvalue decomposition for $\mathbf{N}^{-1}\mathbf{M} \in R^{N \times N}$. The kernel scatter matrices need $N^2(d+1)\ell$ calculations, and the eigenvalue decomposition of $N \times N$ matrix needs $\mathcal{O}(N^3)$ computations ($\frac{9}{2}N^3$ to be more precise [7]). So the total computational complexity of the KDA distinguisher is $\mathcal{O}(N^2(N+(d+1)\ell))$. The memory usage in KDA is to store the two kernel scatter matrices, so the space complexity is $2N^2$.

The classical higher-order DPA attack first requires preprocessing the original data, incurring ℓ^{d+1} (e.g. 10^8 in the case of a 3rd-order masking with 100 time points for each tuple) calculations (subtraction or multiplication according to the combination function) for each trace. So the whole computation for the preprocessing is $N\ell^{d+1}$. Then the computation (via some first-order distinguisher) of 'similarity' between each column of the $N \times \ell^{d+1}$ matrix and the hypothetical power consumption vector requires $N \times \ell^{d+1}$ calculations in total. Therefore, the computational complexity of classical higher-order DPA is $\mathcal{O}(N\ell^{d+1})$. The main memory usage in classical higher-order DPA is to store the preprocessed traces, which implies a space complexity of $N\ell^{d+1}$.

We can see that the computation complexity of the KDA distinguisher is polynomial in N, d, and ℓ. It can still be optimized by the method of using regularized regression to avoid the eigenvalue decomposition in KDA, although the speed up ratio is 27 times [7]. However, the computational complexity of classical higher-order DPA is not only polynomial in N, but also exponential in d. The space complexity of KDA, which depends polynomially on N, represents another advantage over classical higher-order DPA, which requires additional exponential in ℓ space. If the mask order d is high and the ℓ is large, the computation of classical higher order DPA would be extremely high. When only time is considered, if $N(N+(d+1)\ell) < \ell^{d+1}$ given N, ℓ, d, then the KDA distinguisher becomes a better choice for the higher-order attack.

4.2 Flexible Power Model

Like other clustering-based distinguishers [2,12,23], the KDA distinguisher can be performed using different power models. In the dimensionality reduction setting, 256-class, 9-class (Hamming weight), and 3-class KDA have been investigated [6]. In Sect. 3, we investigated the binary power model that was first used in the seminal power analysis paper [11], as well as the 9-class Hamming weight model, for different masking orders; the attacks succeeded in all tested cases.

An especially appealing feature of clustering-based distinguishers is that, unlike classical higher-order DPA, they do not rely on the power model that they use to be *proportionally* approximate to the true leakage; any meaningful *partition* on the intermediate value will suffice. This property suggests KDA as an ideal candidate for use in conjunction with the robust 'semi-profiled' modelling proposed by Whitnall *et al.* at CHES 2015 [29]. The extension of their strategy to higher-order attacks via the KDA distinguisher would be an interesting avenue to explore in future work.

4.3 Limitations and Possibilities

As a baseline against which to compare the key recovery performance of KDA, we also tested higher-order correlation DPA using the 'normalised product' combining function (the best among tested alternatives in typical leakage scenarios [19]) with a Hamming weight power model. In fact, these correlation attacks substantially outperformed the KDA distinguisher in terms of the number of traces required to converge to a guessing entropy of 1. At 2nd and 3rd orders, they were also considerably faster to run.

While the relatively poor trace efficiency is disappointing, it is not surprising given the idealised (Hamming weight or close to Hamming weight) nature of the leakage scenarios tested so far. It is well known that correlation-based attacks perform very efficiently when provided with good proportional approximations of power data dependencies, while the advantages of 'partition'-based [25] (a.k.a./ 'nominal power model'-based [30]) DPA distinguishers only emerge as the true leakage increasingly diverges from standard model assumptions [28]. An interesting avenue for future work will therefore be to deploy the KDA distinguisher in scenarios where higher order correlation DPA is likely to struggle.

We should also stress that our experiments thus far have been *proof of concept*, with no attempt (yet) to optimise for KDA parameters, which may make a substantial difference to the performance of the distinguisher. In particular, it was shown in the dimensionality reduction setting that the quality of the projected traces is influenced by the value of the regularisation parameter μ (Sect. 4.2 in [6]). This is something we plan to explore in future work, along with alternatives to the polynomial kernel function (such as the Gaussian kernel).

The relatively slow computation time indicates that the overheads of the eigenvalue decomposition dominate at 2nd and 3rd orders, so that the complexity advantages of KDA may only begin to emerge as d increases beyond 3. Establishing the threshold at which KDA becomes computationally preferable to classical higher-order DPA is another interesting avenue for further investigation.

5 Conclusions and Future Perspectives

Following recent separate proposals to extend LDA to the task of directly recovering secret keys from unprotected implementations, and to use KDA for the extraction of points of (joint) interest from masked implementations, we have taken the logical next step of extending KDA likewise for application as a distinguisher. We have shown the natural common ground between higher-order DPA and the operation of KDA, and reasoned about the soundness of a KDA-based distinguisher from a theoretical perspective, before verifying its effectiveness empirically. Complexity analysis reveals a substantial advantage of KDA (polynomial in the number of traces and the order of the masking scheme) over higher-order DPA (exponential in the order of the masking scheme).

Although the theoretic complexity advantages of KDA do not translate into practical advantages in our proof-of-concept 2nd and 3rd order experiments, there remains considerable scope for enhancing the methodology and for deploying it in scenarios less vulnerable to classical higher-order DPA. The latter include yet higher masking orders and alternative masking forms, as well as data-dependencies which do not conform nicely to standard assumptions. These represent worthwhile avenues for further investigation. We also anticipate that fine-tuning the parameters (in particular, the regularization factor μ) and exploring alternative kernel functions will have a positive impact on performance.

Acknowledgements. The authors would like to thank Daniel P. Martin for the fruitful discussions on the complexity analysis. This work was supported by the National Natural Science Foundation of China (No.61372062) and by the EPSRC (EP/N011635/1).

References

1. DPA Contest v4. http://www.dpacontest.org/v4/
2. Batina, L., Gierlichs, B., Lemke-Rust, K.: Differential cluster analysis. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 112–127. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_9
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
4. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Less is more: dimensionality reduction from a theoretical perspective. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 22–41. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_2
5. Cagli, E., Dumas, C., Prouff, E.: Enhancing dimensionality reduction methods for side-channel attacks. In: Homma, N., Medwed, M. (eds.) CARDIS 2015. LNCS, vol. 9514, pp. 15–33. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31271-2_2
6. Cagli, E., Dumas, C., Prouff, E.: Kernel discriminant analysis for information extraction in the presence of masking. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 1–22. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_1
7. Cai, D., He, X., Han, J.: Efficient kernel discriminant analysis via spectral regression. In: Seventh IEEE International Conference on Data Mining, ICDM 2007, pp. 427–432. IEEE (2007)
8. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
9. Durvaux, F., Standaert, F.-X., Veyrat-Charvillon, N., Mairy, J.-B., Deville, Y.: Efficient selection of time samples for higher-order DPA with projection pursuits. In: Mangard, S., Poschmann, A.Y. (eds.) COSADE 2014. LNCS, vol. 9064, pp. 34–50. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21476-4_3

10. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(2), 179–188 (1936)
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
12. Mahmudlu, R., Banciu, V., Batina, L., Buhan, I.: LDA-based clustering as a side-channel distinguisher. In: Hancke, G.P., Markantonakis, K. (eds.) RFIDSec 2016. LNCS, vol. 10155, pp. 62–75. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62024-4_5
13. Mangard, S., Oswald, E., Standaert, F.X.: One for all-all for one: unifying standard differential power analysis attacks. IET Inf. Secur. **5**(2), 100–110 (2011)
14. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44499-8_19
15. Mika, S., Ratsch, G., Weston, J., Scholkopf, B., Mullers, K.R.: Fisher discriminant analysis with kernels. In: Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop, pp. 41–48. IEEE (1999)
16. Nassar, M., Souissi, Y., Guilley, S., Danger, J.L.: RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, pp. 1173–1178. IEEE (2012)
17. Oswald, E., Mangard, S.: Template attacks on masking—resistance is futile. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 243–256. Springer, Heidelberg (2006). https://doi.org/10.1007/11967668_16
18. Peeters, E., Standaert, F.-X., Donckers, N., Quisquater, J.-J.: Improved higher-order side-channel attacks with FPGA experiments. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 309–323. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_23
19. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
20. Saitoh, S., Sawano, Y.: Theory of Reproducing Kernels and Applications. DM, vol. 44. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-0530-5
21. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
22. Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11605805_14
23. Souissi, Y., Nassar, M., Guilley, S., Danger, J.-L., Flament, F.: First principal components analysis: a new side channel distinguisher. In: Rhee, K.-H., Nyang, D.H. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 407–419. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24209-0_27
24. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_26
25. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition *vs.* comparison side-channel distinguishers: an empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOS devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00730-9_16

26. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
27. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_7
28. Whitnall, C., Oswald, E.: A fair evaluation framework for comparing side-channel distinguishers. J. Cryptogr. Eng. **1**(2), 145–160 (2011)
29. Whitnall, C., Oswald, E.: Robust profiling for DPA-style attacks. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 3–21. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_1
30. Whitnall, C., Oswald, E., Standaert, F.-X.: The myth of generic DPA...and the magic of learning. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 183–205. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_10

Leakage Bounds for Gaussian Side Channels

Thomas Unterluggauer[1](✉), Thomas Korak[1], Stefan Mangard[1], Robert Schilling[1], Luca Benini[2], Frank K. Gürkaynak[2], and Michael Muehlberghuber[2]

[1] Graz University of Technology, Graz, Austria
{`thomas.unterluggauer,thomas.korak,stefan.mangard,robert.schilling`}`@iaik.tugraz.at`

[2] Integrated Systems Laboratory, ETH Zürich, Zürich, Switzerland

Abstract. In recent years, many leakage-resilient schemes have been published. These schemes guarantee security against side-channel attacks given bounded leakage of the underlying primitive. However, it is a challenging task to reliably determine these leakage bounds from physical properties.

In this work, we present a novel approach to find reliable leakage bounds for side channels of cryptographic implementations when the input data complexity is limited such as in leakage-resilient schemes. By mapping results from communication theory to the side-channel domain, we show that the channel capacity is the natural upper bound for the mutual information (MI) to be learned from multivariate side-channels with Gaussian noise. It shows that this upper bound is determined by the device-specific signal-to-noise ratio (SNR). We further investigate the case when attackers are capable of measuring the same side-channel leakage multiple times and perform signal averaging. Our results here indicate that the gain in the SNR obtained from averaging is exponential in the number of points of interest that are used from the leakage traces. Based on this, we illustrate how the side-channel capacity gives a tool to compute the minimum attack complexity to learn a certain amount of information from side-channel leakage. We then show that our MI bounds match with reality by evaluating the MI in multivariate Gaussian templates built from practical measurements on an ASIC. We finally use our model to show the security of the KECCAK-$f[400]$-based authenticated encryption scheme ISAP on this ASIC against power analysis attacks.

Keywords: Leakage-resilient cryptography · Leakage model mutual information · Channel capacity

1 Introduction

Side-channel attacks are a serious threat to cryptographic implementations as they allow attackers to learn secret information processed inside a device from

T. Eisenbarth and Y. Teglia (Eds.): CARDIS 2017, LNCS 10728, pp. 88–104, 2018.
https://doi.org/10.1007/978-3-319-75208-2_6

observing its physical behavior, e.g., the power consumption. In order to protect implementations from such attacks, one approach actively being researched for several years now is leakage-resilient cryptography. Leakage-resilient schemes are designed such as to resist a certain amount of side-channel leakage. This means that if every invocation of the underlying primitive leaks λ bits of information, leakage-resilient schemes guarantee that their overall leakage stays within predefined bounds. In addition, the side-channel leakage λ is commonly bounded by limiting the input data complexity to the internal primitives, as for example in leakage-resilient encryption using the 2PRG primitive [15]. However, it is an ongoing topic of research to specify concrete leakage bounds λ based on the implementation and its physical properties.

For example, Medwed et al. [9] evaluated a set of practical differential power analysis (DPA) attacks on simulated leakages from parallel implementations with unknown in- and outputs. Their resulting success probabilities indicate that even for identity leakage of the secret state, its exploitation is practically hard once enough processes happen in parallel. While their specific results also suggest security for limited data complexities, it is hard to derive a concrete leakage bound λ in bits. On the other hand, Standaert et al. [14] suggested using the mutual information (MI) from information theory as a general tool to concretely state the amount of information learned from side-channel leakage in bits. While the MI can only be exactly computed once the actual leakage distribution of an implementation is known, Duc et al. [4] mention an upper bound for the MI for univariate leakages that solely depends on the device- and measurement-specific signal-to-noise ratio (SNR). It, however, remains unclear how this bound scales for multivariate leakages that are exploited in practice.

For a single measurement of the side-channel leakage, physical constraints such as the SNR will typically bound the MI to suit leakage-resilient schemes. While most of these schemes indeed confine the attacker to a single measurement by requiring a fresh initial state on every invocation, there are also schemes allowing attackers to observe the same execution using the same data multiple times, e.g., as for multiple decryptions in Isap [3]. However, multiple measurements of the same decryption process allow an attacker to perform signal averaging to increase the SNR. This can allow unbounded side-channel attackers to distinguish tiny variances in the signal to learn the complete secret state. However, in practice, side-channel attackers are bounded by physical and computational resources. This gives the interesting question of the actual attack complexity when the side-channel attacker is capable of observing the same execution multiple times and performing signal averaging.

Our Contribution. In this work, we present a new approach to give reliable upper bounds for the leakage from side channels of cryptographic implementations under a single data input. For this purpose, we map results from communication theory to the side-channel domain. In particular, we show that the channel capacity of n-to-m communication channels is the natural upper bound for the MI in multivariate side-channel leakages with Gaussian noise. Without any further leakage assumptions, we show that this bound depends on a device- and measurement-specific SNR that is uniquely determined by the device's statistical leakage behavior in the m points of interest (POIs) in the leakage trace. In a second step, we investigate the effect of signal averaging on this SNR and show that averaging N leakage traces increases the SNR by a factor N^m. Our results provide both attackers and implementers with a tool for computing the expected minimum attack complexity, i.e., the number of leakage traces required to learn a certain amount of the processed state from side-channel information. We then show that our model and results fit the reality by evaluating the MI in multivariate Gaussian templates. For this purpose, we used power measurements from a real system on chip (SoC) that features a KECCAK-f[400] engine that computes three rounds per cycle. Last, we use our model to demonstrate the security of the scheme ISAP implemented on this SoC w.r.t. power analysis attacks.

Outline. This paper is organized as follows. Section 2 gives bounds for the information leakage of multivariate side channels with Gaussian noise. We analyze the case of signal averaging and provide a tool to compute the expected minimum attack complexity for side-channel attackers in Sect. 3. The soundness of our leakage model is shown in Sect. 4 based on power measurements of an ASIC, and we finally conclude in Sect. 5.

2 Modeling Side-Channel Leakage as a Communication Channel

In this section, we consider the case of leakage-resilient cryptography where an attacker can use the side-channel information in a single leakage trace to learn the secret state of a device. In particular, we adapt the results from communication theory to fit side-channel leakages and use the channel capacity of n-to-m wireless channels to give a leakage upper bound for multivariate side channels with Gaussian noise independent of the underlying leakage function.

2.1 Attack Model

We consider an attacker trying to recover the secret state x from a single leakage trace l_x generated by an implementation $\mathcal{I}$ with input complexity $q = 1$. This implies that the attacker is unable to perform multi-input attacks such as DPA. Moreover, attackers are allowed to observe the operation using the secret state x only a single time, i.e., they are not allowed to average traces to improve their SNR. However, we consider a profiled attack setting, i.e., the attacker has the opportunity to build templates before performing the actual attack.

2.2 Mutual Information

A common metric to assess the amount of information about a secret x contained in the leakage l_x is the mutual information (MI) [4,14]. We therefore introduce the random variables X and L_x to denote the distributions of x and l_x, respectively. The mutual information is then defined as

$$MI(X; L_x) = H[X] - H[X|L_x]. \tag{1}$$

Hereby, $H[X]$ and $H[X|L_x]$ denote the entropy of the random variable X and the conditional entropy of X given the leakage L_x, respectively. Note however that the (conditional) entropy (and thus the MI) is an average metric depending on the actual distribution of values $x_i \in X$ and $l_x \in L_x$. This means that the actual information learned from a side-channel leakage depends on the actually processed value and might thus for some events even be higher than the MI. Yet, the MI is a good metric to give bounds on the expected leakage behavior.

2.3 Linear Channel Model

For giving bounds on the MI of side channels, we consider an implementation that transmits the single bits of a secret state to the attacker via a side channel. Hereby, the physical leakage behavior and measurement effects define the mapping of the single bits to the output samples of the side channel. We model this multivariate side channel as an n-to-m linear communication channel with Gaussian noise, i.e., it transfers linear combinations of the bits of the secret state. While this linear channel model allows to adapt results from communication theory, the resulting bounds are yet independent from the concrete leakage behavior and Gaussian noise is the sole assumption. Namely, our final bounds will only depend on the side-channel signal observed by the attacker. Further note that non-linear mappings can easily be added to this model similar as for regression techniques [13].

In our linear channel model, the attacker observes an $m \times 1$ leakage trace $\mathbf{l}_x$ from the processing of the secret state x in the implementation $\mathcal{I}$. Let $\mathbf{x}$ denote the $n \times 1$ vector consisting of the n bits of the secret state x. We then model the leakage trace $\mathbf{l}_x$ as the multiplication of the secret state vector $\mathbf{x}$ with a $m \times n$ side-channel matrix $\mathbf{H}$ plus an $m \times 1$ noise vector $\boldsymbol{\nu}$:

$$\mathbf{l}_x = \mathbf{H}\mathbf{x} + \boldsymbol{\nu}. \tag{2}$$

The i-th row of $\mathbf{H}$ specifies how the n bits of the secret state x map to the i-th point of the measured leakage $\mathbf{l}_{\mathbf{x}}$. The maximum MI that an attacker can learn from the side-channel leakage according to Eq. 2 depends on the maximum number of states that are distinguishable at the receiver of this channel. This upper bound on the MI is typically called the channel capacity. In particular, Telatar [16] states the channel capacity C as the maximum average mutual information between in- and output over the choice of the input distribution, i.e.,

$$C = \max_{p(X)} MI(X, L_x). \tag{3}$$

We observe that the side-channel leakage given by Eq. 2 bears some familiarity with the notion of multi-input multi-output (MIMO) channels as used in wireless communication. For a constant, known channel $\mathbf{H}$, Goldsmith et al. [7] state the channel capacity for signals in the domain of complex numbers as follows:

$$C = \max_{\Sigma_{\mathbf{x}}:tr(\Sigma_{\mathbf{x}})=P} \log_2 |\mathbf{I}_m + \mathbf{H}\Sigma_{\mathbf{x}}\mathbf{H}^H| \tag{4}$$

Hereby, $\mathbf{I}_m$ and $\Sigma_{\mathbf{x}}$ denote the $m \times m$ identity matrix and $n \times n$ signal covariance matrix, respectively. P is the total power constraint of the transmitter, $\mathbf{H}^H$ the complex conjugate of $\mathbf{H}$, $|\cdot|$ the determinant, and $tr(\cdot)$ the trace of a matrix. For Eq. 2 to hold true, the noise vector $\boldsymbol{\nu}$ must consist of independent samples of Gaussian white noise with variance $\sigma_{\nu}^2 = 1$, i.e., the $m \times m$ noise covariance matrix $\Sigma_{\boldsymbol{\nu}}$ is the identity matrix $\mathbf{I}_m$.

We can use the channel capacity of MIMO channels as an upper bound for the MI in side-channel leakages according to Eq. 2. However, there are different constraints for side channels than in wireless communication, requiring some modifications of Eq. 4. For example, an attacker cannot influence the signal covariance $\Sigma_{\mathbf{x}}$ such as to optimize the capacity C. Moreover, side-channel attacks typically exploit real-valued information like the power consumption, whereas signals in communication channels are represented in the domain of complex numbers. This effectively halves the capacity for the side-channel case. In practice, we also observe that the samples in the noise vector $\boldsymbol{\nu}$ are not necessarily independent and have different variances. According to [7], dependent samples in the noise $\boldsymbol{\nu}$ can be modeled via a modified channel matrix $\tilde{\mathbf{H}} = \Sigma_{\boldsymbol{\nu}}^{-1/2}\mathbf{H}$ given the noise covariance matrix $\Sigma_{\boldsymbol{\nu}}$. By adapting Eq. 4 according to these considerations, we extract the special case of linear side channels as in Eq. 2 and state their leakage upper bound:

$$C = \max_{p(X)} MI(X, L_x) = \frac{1}{2} \log_2 |\mathbf{I}_m + \Sigma_{\boldsymbol{\nu}}^{-1}\mathbf{H}\Sigma_{\mathbf{x}}\mathbf{H}^H|. \tag{5}$$

2.4 Leakage Bound for Gaussian Side Channels

The side-channel matrix $\mathbf{H}$ is typically unknown but fixed. An interesting question thus is how to determine the channel capacity if $\mathbf{H}$ is unknown. A common approach to characterize a side channel are multivariate Gaussian templates. Hereby, for each secret state $\mathbf{x}$, the respective side-channel leakage $\mathbf{l}_{\mathbf{x}}$ is described as a multivariate Gaussian distribution. This characterization gives a set of templates $(\mu_i, \Sigma_{\boldsymbol{\nu},i})$ with mean μ_i and noise covariance $\Sigma_{\boldsymbol{\nu},i}$ for all states $\mathbf{x}_i$. The means μ_i give an estimation of the $n \times n$ covariance matrix $\Sigma_{\mathbf{y}}$ of the side-channel signal $\mathbf{y} = \mathbf{H}\mathbf{x}$. This covariance matrix $\Sigma_{\mathbf{y}}$ equals $\mathbf{H}\Sigma_{\mathbf{x}}\mathbf{H}^H$ from Eq. 5. Similarly, assuming that the noise is independent from the signal and thus has constant covariance (as in [11]), the single noise covariances $\Sigma_{\boldsymbol{\nu},i}$ give an estimation of $\Sigma_{\boldsymbol{\nu}}$.[1] Putting this together, we adapt Eq. 5 to derive our main result. Namely,

[1] The constant covariance assumption is invalid in case the covariance carries information as, e.g., in masked implementations. However, leakage-resilient cryptography aims to bound the leakage without the use of countermeasures like masking, and thus noise will typically be independent from the signal.

we use the signal and noise covariance matrices $\Sigma_{\mathbf{y}}, \Sigma_{\boldsymbol{\nu}}$ to state the capacity of a side channel characterized via multivariate Gaussian templates, or more generally, of multivariate leakages with Gaussian noise.

Main Result (Leakage Bound of a Gaussian Side Channel). *The mutual information of a multivariate side channel with signal covariance $\Sigma_{\mathbf{y}}$ and Gaussian noise $\Sigma_{\boldsymbol{\nu}}$ is bounded by*

$$C = \frac{1}{2} \log_2 |\mathbf{I}_m + \Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}}|. \tag{6}$$

Interestingly, the term $\Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}}$ is an SNR taking noise and signal covariances between the POIs into account. The capacity of the side channel is thus determined by the actual power of signal and noise, and correlations in the samples of $\boldsymbol{\nu}$ and $\mathbf{y}$. Such correlations typically mark redundancies that effectively reduce the side-channel capacity. Moreover, note that the side-channel capacity given here depends on the side-channel signal $\mathbf{y}$ only. This means that our result applies to any leakage function/model having the properties given by $\Sigma_{\mathbf{y}}$.

For univariate leakages or when the same leakage is observed in multiple POIs within the leakage trace, the leakage bound in Eq. 6 can further be simplified.

Univariate Leakage. An attacker exploiting univariate leakage is confined to the leakage in a single point of the execution of an implementation $\mathcal{I}$. This means that the side channel degenerates to

$$l_{\mathbf{x}} = \mathbf{hx} + \nu, \tag{7}$$

where $l_{\mathbf{x}}$ and ν are scalars and the $1 \times n$ channel vector $\mathbf{h}$ specifies the leakage of the single bits of the state $\mathbf{x}$. Let us now assume the channel vector $\mathbf{h}$ maps the n bits in $\mathbf{x}$ to y according to the identity of the respective state variable x. Intuitively, the MI between the secret state $\mathbf{x}$ and its leakage $l_{\mathbf{x}}$ is here bounded by the number of different states that an attacker can distinguish in the single leakage point $l_{\mathbf{x}}$. This number depends both on the distance between the different states along the measured signal range and the noise. When adapting Eq. 6 for univariate leakage, we can observe exactly this dependence:

$$C = \frac{1}{2} \log_2 \left(1 + \frac{\sigma_y^2}{\sigma_\nu^2}\right) = \frac{1}{2} \log_2 \left(1 + SNR\right), \tag{8}$$

where σ_y^2 is the variance of the signal $y = \mathbf{hx}$ and σ_ν^2 is the variance of the noise ν. As also noted in [4, 10], this upper bound for the MI in univariate leakages solely depends on the SNR and is better known as the Shannon-Hartley theorem [2].

Identical Leakage in Multiple Points. In many cases, an attacker will try to exploit the leakage in multiple points of the execution to increase their success rate. If these points are chosen to be in close vicinity within the leakage trace, these POIs will often carry highly redundant information. An example where this case occurs are attackers sampling the side channel at a very high rate and using several consecutive sampling points in their attack. In such situation, one can assume the leakage to be the same for all points of the leakage trace. This case is equivalent to single-input multiple-output (SIMO) channels in wireless communication. The side-channel matrix is then expressed as the vector multiplication $\mathbf{H} = \mathbf{h}_{gain} \cdot \mathbf{h}_l$, where $\mathbf{h}_l$ states the $1 \times n$ side-channel vector mapping the n bits of x to a scalar value and $\mathbf{h}_{gain}$ is the $m \times 1$ gain vector over the m POIs used by the attacker. The capacity formula in Eq. 5 degenerates for such leakage behavior, but can simply be expressed using the vector $\mathbf{h}_{gain}$ only [6]:

$$C = \frac{1}{2} \log_2 \left(1 + \sigma_z^2 \mathbf{h}_{gain}^H \Sigma_{\boldsymbol{\nu}}^{-1} \mathbf{h}_{gain}\right), \tag{9}$$

where σ_z^2 is the variance of the signal $z = \mathbf{h}_l \mathbf{x}$ such that $\mathbf{l}_\mathbf{x} = \mathbf{h}_{gain} z + \boldsymbol{\nu}$.

2.5 Description of Common Leakage Models

Our leakage model in Eq. 2 allows to easily describe linear side-channel leakages. We now give several examples on how to map existing power models to Eq. 2. Note that we give these examples without consideration of the effective signal range in the leakage $\mathbf{l}_\mathbf{x}$.

Identity Leakage. In identity leakage, the n-bit secret state $\mathbf{x}$ leaks linear to the value x it represents. If $\mathbf{x}$ leaks the identity in the i-th sample of $\mathbf{l}_\mathbf{x}$, the i-th row in the side-channel matrix $\mathbf{H}$ takes the form $\mathbf{h} = \left(2^0\ 2^1\ 2^2\ \ldots\ 2^{n-2}\ 2^{n-1}\right)$.

Hamming Weight Leakage. In Hamming Weight (HW) leakage, the secret state $\mathbf{x}$ leaks the number of bits set to one. HW leakage in the i-th sample of $\mathbf{l}_\mathbf{x}$ results in the i-th row of $\mathbf{H}$ to take the form $\mathbf{h} = \left(1\ 1\ 1\ \ldots\ 1\ 1\right)$. Hamming Distance (HD) leakage is modeled in the same way by setting the secret x to be the xor of the leaking state before and after it toggles.

Time-Serialized Leakage. In time-serialized implementations, an attacker collecting the side-channel leakage at different points in time will be able to learn different information in the different POIs. One prominent example are byte-oriented cryptographic implementations, where in each clock cycle a different byte of the n-bit state $\mathbf{x}$ is processed and leaks. For simplicity, let us assume an 8-bit state and HW leakage of a 2-bit chunk processed in the respective clock cycle. This will give a side-channel matrix of the form

$$\mathbf{H} = \begin{pmatrix} 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,1 \end{pmatrix}.$$

Localized Leakage. Localized electromagnetic emanation (EM) attacks are a powerful way to extract information from parts of the secret state. Such localized

EM leakage can in principal be modeled the same way as time-varying leakage. For example, consider the leakages $\mathbf{l}_{\mathbf{x},1}$ and $\mathbf{l}_{\mathbf{x},2}$ observed in two different EM positions. Moreover, assume that $\mathbf{l}_{\mathbf{x},1}, \mathbf{l}_{\mathbf{x},2}$ consist each of two samples leaking the identity of the first or second half of a 4-bit state, respectively. Concatenating the two leakages $\mathbf{l}_{\mathbf{x}}^T = (\mathbf{l}_{\mathbf{x},1}^T \mathbf{l}_{\mathbf{x},2}^T)$ means concatenating the respective channel matrices $\mathbf{H}_1, \mathbf{H}_2$ to a combined side-channel matrix of the form

$$\mathbf{H} = \begin{pmatrix} 2^0 & 2^1 & 0 & 0 \\ 2^0 & 2^1 & 0 & 0 \\ 0 & 0 & 2^0 & 2^1 \\ 0 & 0 & 2^0 & 2^1 \end{pmatrix}.$$

This model underlines the intuition that gathering additional leakage from observing a parallel implementation in different locations and measuring a serial implementation at different times is equivalent. In particular, it shows that side-channel leakage becomes optimal if the leakages in the side-channel signal $\mathbf{y} = \mathbf{Hx}$ are independent. In the best case, the signal covariance matrix becomes a diagonal matrix, i.e., $\Sigma_{\mathbf{y}} = diag(\sigma_{y_1}^2, \sigma_{y_2}^2, ..., \sigma_{y_m}^2)$. In the same way, noise effects are canceled out the best if the noise samples in $\boldsymbol{\nu}$ are independent, i.e., $\Sigma_{\boldsymbol{\nu}}$ is a diagonal matrix as well.

3 Complexity of State Recovery

The side-channel capacity is an upper bound on the MI to be learned via a side channel. This bound essentially depends on the implementation's SNR. While in most leakage-resilient schemes an attacker is restricted to a single leakage trace for a specific state, there are schemes, e.g., Isap [3], that allow attackers to observe the execution of an implementation processing the same data multiple times. This gives attackers the option to perform signal averaging, which improves the side-channel SNR and thus side-channel capacity.

In this section, we therefore consider an attacker capable of averaging multiple leakage traces. We show how averaging improves the side-channel capacity in multivariate attacks and provide attackers and implementers with a tool to compute the expected minimum complexity to learn the secret state of a device.

3.1 Attack Model

As in Sect. 2, we assume an attacker trying to recover a secret state x from side-channel leakages l_x generated by an implementation $\mathcal{I}$ with input complexity $q = 1$ and thus preclude multi-input attacks. However, the attacker is capable of observing the same execution of $\mathcal{I}$ multiple times. This attack setting is observed when a ciphertext, e.g., a firmware image, must be decrypted multiple times using a leakage-resilient scheme like Isap.

3.2 Averaging Attacker

An attacker that observes the same processing of the secret state x multiple times is capable of averaging the side-channel leakage l_x to yield a better SNR and thus higher side-channel capacity. In general, averaging N observations gives the averaged noise covariance matrix

$$\overline{\Sigma}_{\boldsymbol{\nu}} = \frac{1}{N} \Sigma_{\boldsymbol{\nu}}, \tag{10}$$

where $\Sigma_{\boldsymbol{\nu}}$ is the noise covariance matrix valid for a single leakage trace. This means that the noise (co-)variances reduce linearly with the number of averaged traces. Note here that for the univariate case Eq. 10 simplifies to the well-known relation $\overline{\sigma}_{\nu}^2 = \frac{\sigma_{\nu}^2}{N}$. Given the noise covariance matrix after averaging $\overline{\Sigma}_{\boldsymbol{\nu}}$, we can now investigate the effect of averaging on the side-channel capacity. Inserting Eq. 10 into the generic side-channel capacity given in Eq. 4 yields

$$C = \frac{1}{2} \log_2 \left| \mathbf{I}_m + N \cdot \Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}} \right| . \tag{11}$$

Note that the SNR term $N \cdot \Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}}$ is an $m \times m$ matrix and its determinant behaves proportionally to N^m. This means that the side-channel capacity increases stronger with the number of averaged traces the more POIs are used in an attack. This is because each POI can potentially transfer completely independent data as, e.g., for time-serialized and localized EM leakages.

Identical Leakage in Multiple Points. For identical leakage in all POIs, the side-channel capacity behaves differently. Inserting Eq. 10 into the SIMO channel capacity given in Eq. 9 yields

$$C = \frac{1}{2} \log_2 \left(1 + N \cdot \sigma_z^2 \cdot \mathbf{h}_{gain}^H \Sigma_{\boldsymbol{\nu}}^{-1} \mathbf{h}_{gain} \right) . \tag{12}$$

It shows that the number of traces N used for averaging has a linear influence on the SNR and is independent of the number of POIs m.

3.3 Expected Minimum Attack Complexity

In the worst case, physical attackers have unbounded complexity. This means they can measure and average an unlimited number of leakage traces $N \rightarrow \infty$, leading to zero noise and virtually unlimited channel capacity and MI. This can be thought of state differences causing vanishingly small differences in the side-channel signal being distinguishable if the noise is eliminated completely. It thus seems reasonable to set the side-channel capacity in relation with the actual attack complexity, i.e., the number of leakage traces N, to learn a certain amount of bits. This is also the common approach when assessing the security of masked implementations.

It is yet difficult to determine such attack complexity since it is strongly influenced by the implementation's leakage behavior, which is commonly unknown.

For example, it is unknown to what extent information and noise in the single points of a leakage trace correlate, and as shown in Sect. 2, these effects strongly influence the side-channel capacity. However, the device- and measurement-specific multivariate $SNR = \Sigma_{\mathbf{y}} \cdot \Sigma_{\boldsymbol{\nu}}^{-1}$ takes exactly these effects into account and can thus be used to generically express the expected minimum complexity of a side-channel attacker without any concrete leakage assumptions. In particular, we can rewrite the multivariate channel capacity for averaging attackers (Eq. 11) as follows:

$$C = \frac{1}{2} \log_2 N^m \left| \frac{1}{N} \mathbf{I}_m + \Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}} \right| . \tag{13}$$

For a large number of averaged traces N, Eq. 13 can be further approximated to give the side-channel capacity in dependence of a scalar device SNR.

$$C \approx \frac{1}{2} \log_2 \left(1 + N^m \left|\Sigma_{\boldsymbol{\nu}}^{-1} \Sigma_{\mathbf{y}}\right|\right) = \frac{1}{2} \log_2 (1 + N^m \cdot SNR_m) \tag{14}$$

An implementation will in practice give some side-channel $SNR_m = |\Sigma_{\mathbf{y}} \cdot \Sigma_{\boldsymbol{\nu}}^{-1}|$ that is observed in m POIs in the leakage traces. This SNR takes into account all kinds of correlations in both noise and side-channel leakage. For an implementation that is expected to give a certain SNR_m, designers and implementers can thus compute the expected minimum attack complexity in terms of traces to measure and average.

Figure 1 gives an overview on the expected side-channel capacity for $m = 1, 5, 10$ POIs given the number of averaged traces. It shows that the side-channel capacity rises quickly with the number of averaged traces for multivariate leakages. In particular, it shows that if SNR_m is not sufficiently low, a state of

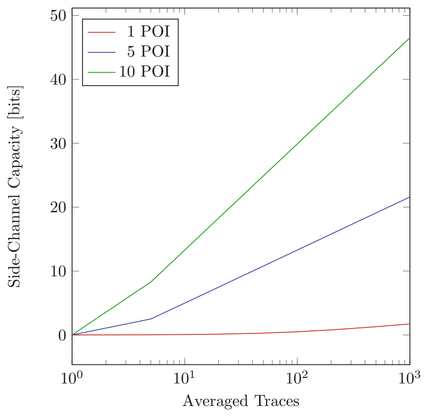


Fig. 1. Expected side-channel capacity given the number of averaged leakage traces for different numbers of POIs and $SNR_m = 0.01$.

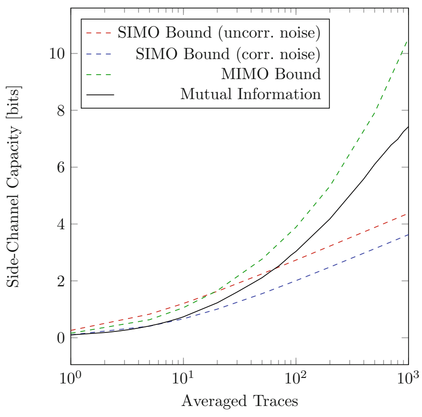


Fig. 2. Mutual information of KECCAK-$f[400]$ on FULMINE and side-channel capacity of different channel models (256 classes, 10 POIs).

virtually any size can theoretically be recovered with practical complexity. However, this effect is also limited by the available POIs with sufficiently low signal correlations.

4 Experimental Verification and Security Analysis

The previous sections introduced theoretical leakage upper bounds for multivariate side channels with Gaussian noise. In this section, we show that these bounds match the real leakage behavior by evaluating the MI on a hardware implementation of the KECCAK-f[400]-based scheme ISAP [3] on the real system on chip FULMINE. Our experiments further show the security of this implementation of ISAP in terms of power analysis attacks.

4.1 Evaluation Hardware: Fulmine

At FSE 2017, Dobraunig et al. [3] presented the sponge-based authenticated encryption scheme ISAP to inherently prevent DPA during both en-/decryption. This is achieved by limiting the number of inputs processed under a single key by one. To further express their scheme's capability to cope with side-channel leakage from a single data input, the authors proposed using the sponge parameters themselves. However, in the view of ISAP allowing for the multiple decryption of the same ciphertext and tag, it is an open question how much information an attacker can learn when averaging multiple leakage traces.

To verify the soundness of our leakage bounds and to evaluate the side-channel resistance of ISAP, we developed and fabricated the multi-core SoC FULMINE, a prototype ASIC in the UMC 65 nm LL 1P8M technology. FULMINE, as shown in Fig. 3, is based on the PULP platform [12] including four general purpose processing elements (enhanced OpenRISC cores with DSP extensions [5,8]) and two dedicated hardware accelerators: the Hardware Cryptography Engine (HWCRYPT) and the Hardware Convolution Engine (HWCE). All processing elements share the same 64 kB level-1 Tightly-Coupled Data Memory (TCDM) to support a fast and efficient communication and to avoid single point-to-point channels.

HWCRYPT is a flexible, software-programmable hardware accelerator supporting various cryptographic primitive functions such as the KECCAK-f[400] permutation [1]. Moreover, the accelerator supports high-level encryption schemes such as ISAP. The accelerator is designed to achieve maximum throughput. To achieve that goal, the KECCAK-f[400] permutation utilizes three fully parallel round instances to maximize the throughput but to also match the length of the critical path of other parts of the accelerator. When using ISAP, HWCRYPT supports a flexible configuration of the rate (from 1 bit to 128 bits in powers of two) and the number of permutation rounds in multiples of three including 20 to flexibly trade-off between throughput and security. HWCRYPT is configured and monitored via a set of status registers. A flexible event and interrupt system indicates other processing elements when an operation has finished.

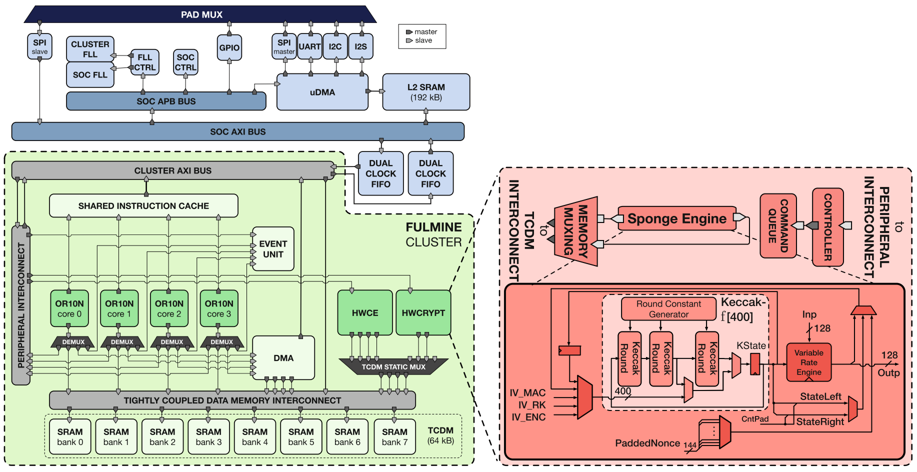
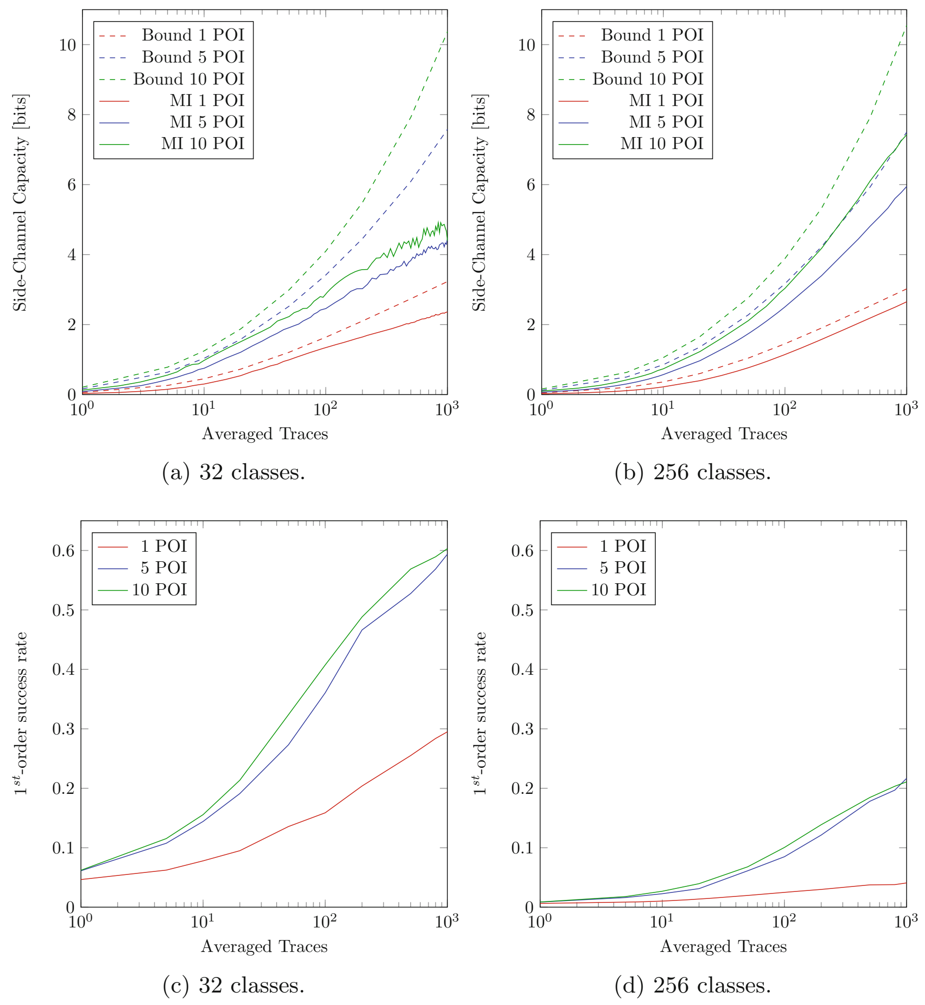


Fig. 3. FULMINE SoC and HWCRYPT architecture.

4.2 Soundness of Model and Bounds

To verify the soundness of our model and the bounds in Sect. 2, we analyzed the leakage behavior of the KECCAK-f[400] permutation on FULMINE. For this purpose, we constructed multivariate Gaussian templates for the power consumption of FULMINE for 5- and 8-bit parts of the 400-bit state of KECCAK-f[400]. More concretely, we target the intermediate state *KState* of KECCAK-f[400], depicted in Fig. 3, such that FULMINE computes three rounds of the permutation before and after the target state to preclude load-time leakages. The remaining state not covered by our templates, i.e., 395 and 392 bits respectively, was held constant. For each class, we used 1400 power measurements in the training phase. The POIs were chosen as the points of highest variance fulfilling a certain minimum distance within the leakage trace and include both register and combinatorial activity. Based on these templates, we computed the side-channel capacity and evaluated both the MI and the 1st-order success rate of classification. The evaluations were done in dependence of the number of leakage traces used for signal averaging.

Our evaluation results in Fig. 4 suggest that the channel model used to compute the side-channel capacity of multivariate leakages is sound. In particular, for both 5-bit and 8-bit templates the MI between leakage and secret state stays within the bounds given by the side-channel capacity. While there is a gap between the MI and the channel capacity, the MI follows the shape of the side-channel bound well. Moreover, the first-order classification rate rises accordingly. However, Fig. 4a also shows that for higher numbers of averaged traces the MI goes into saturation, and thus the gap between capacity and the learned information gets bigger. In particular, it shows that once the MI converges to the maximum number of bits that could be recovered using the trained template set, i.e., 5 or 8 bits respectively, the increase in learned information for additional



(a) 32 classes.

(b) 256 classes.

(c) 32 classes.

(d) 256 classes.

Fig. 4. Side-channel capacity, mutual information and success rate for the KECCAK-$f[400]$ permutation given the number of averaged traces and different numbers of POIs and number of classes. The remaining state was held constant.

numbers of averaged traces gets successively smaller. This indicates that the side-channel information is not distributed to perfectly use the channel.

We further investigated how different channel models suit the actual leakage behavior. We therefore compared the MIMO channel model used in the previous evaluation with the SIMO channel model, which assumes identical leakages in the POIs of a leakage trace, e.g., within a clock cycle. For the SIMO channel model, we analyzed two cases: one taking noise correlations into account, and one assuming independent noise. The channel capacities of the different channel models were computed based on the 8-bit templates constructed in the previous evaluation. In particular, for the SIMO model we used the signal variance in each POI, but neglected signal covariances. The results of our evaluations are shown in Fig. 2. These suggest that the leakages in the single POIs are not identical and thus the MIMO channel model suits the leakage behavior clearly better than the SIMO channel model. Moreover, from the plots using the SIMO model one can observe that there is some noise correlation that lowers the channel capacity.

4.3 Security of ISAP

In most situations, designers and implementers want to assess the security of a complete cryptographic implementation. However, the state sizes involved in a cryptographic scheme like Isap are typically large and the channel capacity computed from a low number of templates cannot be directly used since more hardware will be active. On the other hand, it is impossible to build templates for a 400-bit state that would allow to compute the channel capacity exactly. Yet, we can use the experiments on the keccak-f[400] permutation to estimate leakage bounds for the full state of Isap.

As we can see from Fig. 4, the channel capacity is practically the same for both 5- and 8-bit templates. The reason for this is that the SNR we observed on Fulmine using our measurement setup is the same. This gives the question whether and how the SNR would change for 400-bit templates. Now if the same measurement setup was used for constructing 400-bit templates, we can safely say that the range of the measured noise will not decrease by orders of magnitude. In the same way, the range of the side-channel signal will definitely not rise by orders of magnitude using the same setup, especially since the diffusion of three rounds of keccak-f[400] already causes large parts of the logic to become active within the profiled clock cycle.

On the other hand, the side-channel capacity from a single power measurement of Fulmine is very low, and thus, even if the channel SNR was 100 times higher, the channel capacity would hardly rise. We thus scale the SNR with a factor γ to get a security margin that allows to estimate how many traces an attacker will at least require to recover the complete state or to exceed the leakage bounds. Using the SNR_m of the m-variate leakage from the 8-bit templates, we compute the minimum number of traces needed to learn the state of size S:

$$N = \left(\frac{2^{2S} - 1}{\gamma \cdot SNR_m} \right)^{1/m} . \qquad (15)$$

The authors of Isap state concrete leakage bounds for their re-keying function and encryption scheme to still provide 128-bit security. We thus evaluated Eq. 15 on Fulmine for three different state sizes: the full state of keccak-f[400], the leakage bound for the Isap re-keying function (272 bits), and the leakage bound for the Isap encryption itself (128 bits). The results in Fig. 5 indicate that the minimum attack complexity in terms of measurement traces is impracticable for less than 20 POIs and all mentioned state sizes. However, for higher numbers of POIs the minimum attack complexities tend towards practically feasible.

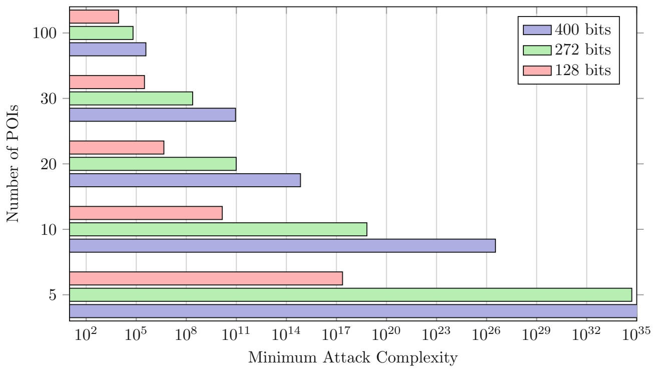


Fig. 5. Minimum attack complexity as the number of measurements needed to average to recover (parts) of the ISAP state from FULMINE. As a security margin we set $\gamma = 100$.

Namely, when using 100 POIs, 10 000 measurements can be enough to learn 128 bits of the state, and 500 000 measurements are the minimum to recover the full state.

However, using that many POIs often hampers template building or leads to overfitting effects reducing the classification rate. Besides, side-channel leakage is not distributed such as to perfectly use the channel. This becomes visible in the gap between channel capacity and MI in Fig. 4. While this might allow an attacker to recover a few states more easily, in consideration of all possible states the attack complexity yet stays above the bounds in Fig. 5. Namely, for non-ideal distributions of the leakage, an attacker will, in general, require even more measurements to learn the specified amount of information.

From a practical perspective, conducting such powerful attack would require an attacker to successfully build templates on the respective state. In many cases, this is however not possible, e.g., when the attacker does not have control over the state on a suitable device. Even further, the complexity to build, measure, and evaluate such large set of templates is clearly impractical. In this respect, the implementation of ISAP on FULMINE can for the used measurement setup be considered secure against power analysis attacks also above the bounds in Fig. 5.

5 Conclusion

In this work, we presented a novel approach to determine leakage upper bounds for side channels of cryptographic implementations under a single data input. Without any further leakage assumptions we showed that the channel capacity of transmission channels with multiple in- and outputs gives the natural upper bound for information leakage in multivariate side channels with Gaussian noise.

We then considered the case where attackers are capable of performing multiple measurements of the same execution in order to improve their SNR. We showed that the gain in the SNR of multivariate leakages resulting from signal averaging is exponential in the number of POIs. This observation gives a tool for attackers to learn about the feasibility of an attack and for implementors to assess the minimum attack complexity of state recovery in leakage-resilient schemes allowing for multiple decryptions like Isap. We verified the soundness of our model and our bounds using the ASIC Fulmine implementing Isap and the keccak-f[400] permutation. Finally, we gave lower bounds on the complexity for recovering the Isap state using power analysis. The results indicate that recovery of the Isap state on Fulmine is practically infeasible with power analysis and the used measurement setup.

Acknowledgements. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402) and from the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS).

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document. Submission to NIST (Round 2) 3, 30 (2009)
2. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, New York (2012)
3. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: Isap - towards side-channel secure authenticated encryption. IACR Trans. Symmetric Cryptol. **2017**(1), 80–105 (2017). http://tosc.iacr.org/index.php/ToSC/article/view/585
4. Duc, A., Faust, S., Standaert, F.-X.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 401–429. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_16
5. Gautschi, M., Schiavone, P.D., Traber, A., Loi, I., Pullini, A., Rossi, D., Flamand, E., Gürkaynak, F.K., Benini, L.: Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(10), 1–14 (2017)
6. Goldsmith, A.: Wireless Communications. Cambridge University Press, Cambridge (2005)
7. Goldsmith, A., Jafar, S.A., Jindal, N., Vishwanath, S.: Capacity limits of MIMO channels. IEEE J. Sel. Areas Commun. **21**(5), 684–702 (2003). https://doi.org/10.1109/JSAC.2003.810294
8. Lampret, D., Chen, C.M., Mlinar, M., Rydberg, J., Ziv-Av, M., Ziomkowski, C., McGary, G., Gardner, B., Mathur, R., Bolado, M.: Openrisc 1000 architecture manual. Description of assembler mnemonics and other for OR1200 (2003)
9. Medwed, M., Standaert, F.-X., Nikov, V., Feldhofer, M.: Unknown-input attacks in the parallel setting: improving the security of the CHES 2012 leakage-resilient PRF. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 602–623. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_22

10. Mizuno, H., Iwai, K., Tanaka, H., Kurokawa, T.: Information theoretical analysis of side-channel attack. In: Bagchi, A., Ray, I. (eds.) ICISS 2013. LNCS, vol. 8303, pp. 255–269. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45204-8_20
11. Rivain, M.: On the exact success rate of side channel analysis in the Gaussian Model. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 165–183. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_11
12. Rossi, D., Conti, F., Marongiu, A., Pullini, A., Loi, I., Gautschi, M., Tagliavini, G., Capotondi, A., Flatresse, P., Benini, L.: Pulp: a parallel ultra low power platform for next generation iot applications. In: Hot Chips 27 Symposium (HCS), 2015 IEEE, pp. 1–39. IEEE (2015)
13. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
14. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
15. Standaert, F., Pereira, O., Yu, Y., Quisquater, J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Towards Hardware-Intrinsic Security - Foundations and Practice, pp. 99–134 (2010)
16. Telatar, E.: Capacity of multi-antenna Gaussian Channels. Eur. Trans. Telecommun. **10**, 585–595 (1999). https://doi.org/10.1002/ett.4460100604

Towards Sound and Optimal Leakage Detection Procedure

A. Adam Ding[1](✉), Liwei Zhang[1], Francois Durvaux[2], Francois-Xavier Standaert[2], and Yunsi Fei[3]

[1] Department of Mathematics, Northeastern University, Boston, MA, USA
a.ding@northeastern.edu
[2] ICTEAM/ELEN/Crypto Group, Universite catholique de Louvain, Louvain-la-Neuve, Belgium
[3] Department of ECE, Northeastern University, Boston, MA, USA

Abstract. Evaluation of side-channel leakage for cryptographic systems requires sound leakage detection procedures. The commonly used standard approach is the test vector leakage assessment (TVLA) procedure. We first relate TVLA to the statistical minimum p-value (mini-p) procedure, and propose a sound method of deciding leakage existence in the statistical hypothesis setting. An advanced statistical procedure, Higher Criticism (HC), is adopted to improve leakage detection when there are multiple leakage points. The HC-based procedure is optimal in side-channel leakage detection, because for a given number of traces with a given length, it detects the existence of leakage at the signal level as low as possibly detectable by any statistical procedure. Numerical studies show that our HC-based procedure perform as well as the mini-p based procedure when leakage signals are very sparse, and can improve the leakage detection significantly when there are multiple leakages.

Keywords: Side-channel analysis · Leakage detection
Higher criticism

1 Introduction

Side-channel analysis (SCA) has been shown to be a serious threat to modern cryptographic implementations. For more than a decade now, researchers actively invented various side-channel attacks and proposed countermeasures to protect devices against such attacks. As countermeasures are integrated into commercial customer devices, evaluating the resistance of devices against SCA becomes an important issue. A *leakage detection* test procedure, Cryptography Research (CRI)'s test vector leakage assessment (TVLA) [1,2], is often used for blackbox evaluation of SCA resistance. The TVLA procedure scans the leakage traces (e.g., physical measurements of the power consumption) with a univariate test, and declares no leakage if the test statistics at all points along the leakage trace falls below a critical value.

It is preferred to use a generic univariate test in the TVLA procedure to avoid dependence on a specific leakage model. The CRI's TVLA proposal runs the Welch's t-test [1,2] on data sets sampled according to a *nonspecific* partition, usually the fixed-vs-fixed sampling or the fixed-vs-random sampling, where the fixed class of measurements come from encryptions of fixed plaintexts while the random class of measurements come from encryptions of random plaintexts. Recently several extensions of the t-test (e.g., higher order and multivariate leakage detection) have been proposed by researchers [3–6].

Durvaux and Standaert [5] at EuroCrypt2016 proposed a correlation-based test (ρ-test) to detect exploitable leakage aimed at a particular intermediate computation. Such a *specific* test yields sparser leakage relating to this targeted intermediate value, and is better suited for identifying Point-Of-Interest (POI) for exploitable leakage. While this identification is necessary for practical SCA, it is not required for the purpose of leakage detection. Both the specific and non-specific leakage detection tests can be used in the TVLA framework.

In this work we first study the TVLA procedure itself from a theoretical perspective. The TVLA procedure declares a device as leaky, if the maximum test statistic (over all points on the trace) exceeds a critical value. For the Welch's t-test, current TVLA procedure generally uses the critical value of 4.5 [2,7–9], which corresponds to a statistical significance level of $\alpha < 0.00001$ for the univariate test. However, this significance level does not consider the total number of univariate tests, i.e., the total number of points on the trace. The overall significance level increases as the number of leakage points on the trace increases. For long traces, the overall significance level can be quite large, so is the test statistic value, and therefore a non-leaky device can not pass the TVLA t-testing with the critical value of 4.5. Hence, Balasch et al. [10] suggested raising the critical value to 5 for longer traces based on numerical experiments. However, for even longer traces, the non-leaky devices still can not pass at this higher critical value of 5 (see Sect. 3.1). The issue is caused by the multiple univariate tests at all time points which led [3] to suggest using false discovery rate to decide the detection limit. However, an explicit rigorous way of setting the threshold value would help for sound application of the current TVLA procedure.

In view of this state-of-the-art, we make two contributions in this paper. First, we propose a sound method to set the threshold value according to an overall statistical significance level. The current TVLA procedure makes the decision (leaky versus non-leaky) based on the largest test statistic, hence it is a statistical minimum p-value (mini-p) procedure that decides only with the minimal p-value of all those univariate tests. The threshold can be set through the mini-p procedure at any given statistical significance level, taking account of the trace length. For the t-test based TVLA, we provide explicit expression of this threshold, which also varies with the number of traces (used as the degree of freedoms in the test).

Second, we propose to improve the (univariate) leakage detection procedure with a statistically optimal HC metric. For the leakage detection purpose, the evaluator searches for evidence of key-dependent leakages along the trace, without necessarily identifying the POIs exactly. Hence it is very similar as the statistical independence scanning procedure [11–17] widely used in other high-dimensional statistical applications. Depending on the signal strength and signal sparsity, there is an *undetectable region* [18] where no statistical test can discern the existence of leakage. An optimal leakage procedure should be able to detect any leakage outside this minimal theoretical undetectable region. The current TVLA (mini-p) procedure is not optimal, as its undetectable region is larger. We incorporate the "Higher Criticism" (HC), a state-of-art statistical method for detecting sparse and weak signals, into the TVLA procedure.

Our work improves the TVLA procedure to optimally utilize multiple POIs for leakage detection. This is independent of whether the univariate test itself is optimal. Both specific and nonspecific leakage detection tests above can be used, with their relative advantages and limitations [5] still applying. Our work is also orthogonal to the work of combining multiple leakages for a single attack, e.g., [19–24]. Our proposed procedure optimally combines *detections* of univariate leakage existence at all points along the leakage trace. It works as well as the mini-p for very sparse leakage signals, and significantly improves the detection in scenarios where there are multiple leakage signals.

2 Background and Model Notations

2.1 TVLA Procedure as a mini-p Testing Method

In the TVLA leakage detection setup, an evaluator collects many traces of physical measurements, and tries to find if some points on the traces leak key information through a key-sensitive intermediate variable V. Let n_{tr} and n_L denote, respectively, the total number of traces and the total number of points on each trace. That is, the evaluator has n_{tr} realizations of the random vector $\boldsymbol{L} = [L_1, \cdots, L_{n_L}]$. The scanning procedure such as TVLA do a univariate statistical test at each time point, and makes decision by combining the results. That is, we test the null hypothesis (there is no leakage signal):

$$L_i = r_i \tag{1}$$

versus the alternative hypothesis (there is leakage):

$$L_i = V + r_i \tag{2}$$

at the i-th time point, where r_i is random noise.

The test is usually done with a test statistic $\widehat{\boldsymbol{s}}_i$. Statistically the p-value is the probability that test statistic value can be observed under null hypothesis, i.e., $\mathrm{P}(|S| \geq |\widehat{\boldsymbol{s}}_i|)$ where S denotes a random variable that follows the distribution of the test statistic under the null hypothesis (1). For a single hypothesis test, the null hypothesis is rejected when $|\widehat{\boldsymbol{s}}_i|$ is too big or equivalently when the p-value is too small.

The TVLA procedure decides that leakage exists as long as any one of the tests rejects the null hypothesis. That is, the device is considered leaky when

$\max_{1 \le i \le n_L} |\widehat{\boldsymbol{s}}_i| \ge \text{TH}$ for a threshold value TH (or equivalently when the minimum p-value is smaller than a threshold value α_{TH}). Therefore, the current TVLA procedure is in fact a mini-p test method for considering multiple (n_L) testing but utilizing only the test with the minimal p-value. We will propose changing this mini-p multiple testing method later.

While the usage of a particular univariate test is not essential to the framework, we first describe two common univariate tests to use as concrete examples for a better understanding of the overall leakage detection framework.

2.2 Univariate Tests: $\boldsymbol{\rho}$-test, t-Test, Specific versus Nonspecific Tests

Given the leakage model (2) with the known intermediate value V, the most natural attack is the correlation power analysis (CPA) distinguisher. The CPA uses the Pearson correlation, ρ, which can also be used for leakage detection in *ρ-test*. The correlation is:

$$\hat{\rho}_i = \text{Corr}(L_i, V). \tag{3}$$

The test statistic is taken as the Fisher's transformation on $\hat{\rho}_i$ scaled by $\sqrt{n_{tr}}$:

$$\widehat{\boldsymbol{s}}_i = \frac{1}{2} \ln \left(\frac{1 + \hat{\rho}_i}{1 - \hat{\rho}_i} \right) \sqrt{n_{tr}}. \tag{4}$$

Under the null hypothesis (no leakage at the i-th time point), $\widehat{\boldsymbol{s}}_i$ approximately follows the standard normal distribution $N(0, 1)$. So the corresponding p-value is calculated by:

$$p_i = 2 \times (1 - \text{CDF}_{N(0,1)}(|\widehat{\boldsymbol{s}}_i|)), \tag{5}$$

where $\text{CDF}_{N(0,1)}(\cdot)$ is the cumulative distribution function of the standard normal distribution.

The ρ-test can be considered as an ideal test for perfectly modeled power leakage, often Hamming Weight or Hamming Distance of a nonlinear (SBox) output. A more generic version of ρ-test is proposed by Durvaux and Standaert [5] where they profiled the leakage on the targeted V, thus allowing implementation in a blackbox manner.

Another common generic test is the Welch's t-test [1,2], where the $\boldsymbol{L}_i$ measurements are partitioned into two sets $L_{i,A}$ and $L_{i,B}$, and compared by the test statistic

$$\widehat{\boldsymbol{s}}_i = \frac{\overline{L}_{i,A} - \overline{L}_{i,B}}{\sqrt{\frac{\hat{\nu}_{i,A}^2}{n_A} + \frac{\hat{\nu}_{i,B}^2}{n_B}}}, \tag{6}$$

where $\overline{L}_{i,A}$ and $\overline{L}_{i,B}$ denote the sample means (average values) in each set, $\hat{\nu}_{i,A}$ and $\hat{\nu}_{i,B}$ denote the sample standard deviations, n_A and n_B denote the numbers of measurements for the set A and B, respectively. The corresponding p-value is calculated as the probability, under a t-distribution with ν_t degree of freedom, that the random variable exceeds the observed statistic value $\widehat{\boldsymbol{s}}_i$:

$$p_i = 2 \times (1 - \text{CDF}_t(\widehat{\boldsymbol{s}}_i, \hat{\nu}_i)), \qquad i = 1, \cdots, n_L, \tag{7}$$

where $\mathrm{CDF}_t(\cdot, \hat{\nu}_i)$ is the cumulative distribution function of t-distribution with the degree of freedom

$$\hat{\nu}_i = (\hat{\nu}_{i,A}^2/n_A + \hat{\nu}_{i,B}^2/n_B)^2/[(\hat{\nu}_{i,A}^2/n_A)^2/(n_A - 1) + (\hat{\nu}_{i,B}^2/n_B)^2/(n_B - 1)].$$

In practice, the degree of freedom $\hat{\nu}_i$ may be big so that the $\mathrm{CDF}_t(\cdot, \hat{\nu}_i)$ can be approximated by $\mathrm{CDF}_{N(0,1)}(\cdot)$. In that case, the p-value for t-test can also be calculated from (5).

Recall that [5] used the ρ-test as a specific test on data partitioned according to the specific intermediate value. The t-test is naturally used on data with two classes with nonspecific partition (fixed-vs-fixed and fixed-vs-random). The data collection methods, specific versus nonspecific, affect how sparse and how strong the leakage signals are in the data. Those are the two critical factors in the theoretical analysis in Sect. 4.

3 Methodology

In this section, we first discuss how to set the threshold for the mini-p procedure correctly. We then describe the higher criticism (HC) procedure.

3.1 Threshold Setting in the mini-p Procedure

The current TVLA procedure declares a device as leaky when $\max_{1 \le i \le n_L} |\widehat{\boldsymbol{s}}_i| \ge$ TH. However, the threshold value TH was not set at a given significance level (Type I error rate) as in usual statistical methods. The t-test threshold of TH $=$ 4.5 is suggested originally as it corresponds to a significance level of < 0.00001 for each *univariate test* [1,2]. However, the overall significance level varies with the number of time points n_L on the trace, so that the procedure is not doing a fair testing for traces with different lengths. Particularly, for a long trace, a leakage free device is often declared as leaky. For this reason, Balasch et al. [10] suggested raising the threshold to TH $= 5$ for long traces. In Table 1(a), we give the type I error rates under both TH $= 4.5$ and TH $= 5$ for the current TVLA procedure. As the number n_L increases, the type I error rate increases. Particularly when $n_L = 1,000,000$, a leakage free device will almost always be declared as leaky (99.9% Type I error rate) under the threshold TH $= 4.5$, and still about 44% chance of being declared as leaky with the higher threshold TH $= 5$. Either way, we observe that for any such fixed threshold for the test statistic, type I error rate varies greatly for different n_L. Thus a more formal way of setting the threshold value according to the trace length is needed, to allow fair evaluation across different trace lengths.

Realizing that the current TVLA procedure is in fact a mini-p procedure, the threshold for the minimum p-value should be set as $\alpha_{\mathrm{TH}} = 1 - (1 - \alpha)^{1/n_L}$ for an overall significance level α. Then for the t-test, the threshold is TH $=$ $\mathrm{CDF}_t^{-1}(1 - \alpha_{\mathrm{TH}}/2, \nu_s)$ where CDF_t^{-1} is the inverse of CDF of t-distribution. This threshold value depends on the number of traces n_{tr} which affects the

Table 1. T-test threshold and Type I error rates for varying trace lengths n_L.

(a) Type I error rates α under fixed threshold values.

n_L	10^2	10^3	10^4	10^5	10^6
TH = 4.5	0.00068	0.0068	0.0661	0.4957	0.9987
TH = 5	0.000057	0.00057	0.0057	0.0557	0.4363

(b) Threshold values TH under fixed type I error rates.

n_L	10^2	10^3	10^4	10^5	10^6	10^7	10^8
$\alpha = 0.001$	4.417	4.892	5.327	5.731	6.110	6.467	6.806
$\alpha = 0.01$	3.889	4.416	4.891	5.326	5.730	6.109	6.466

degrees of freedom ν_s in the t-distribution. When ν_s is big, this can also be calculated as $\text{CDF}^{-1}_{N(0,1)}(1-\alpha_{\text{TH}}/2)$. In Table 1(b), we list the cutoff values, for the type I error rate of 0.001 and 0.01 under various trace lengths (assuming ν_s is big).

Next, we propose an improved leakage detection method based on the higher criticism (HC) [11,12] which utilize the information contained in all n_L test statistics more efficiently.

3.2 Higher Criticism

Statistically, the leakage detection can be formulated as testing

$$H_0 : \text{Model (1) holds at all time points } (i = 1, ..n_L), \quad (8)$$

$$\textit{versus} \quad H_1 : \text{Model (2) holds at some time points..} \quad (9)$$

The current mini-p procedure ignores the information on all other p-values except for the minimal p-value $\min_{1 \le i \le n_L} p_i$. The HC method utilizes the information stored in the distribution of p-values. Under the null hypothesis (8), all observed p-values should follow a uniform distribution on the interval $[0, 1]$. For the time points where leakage exists as Eq. (2), the expected p-values will be smaller than those generated from the uniform distribution. Hence under the alternative hypothesis (9) of some POIs with leakage (a mixture distribution), the obtained p-values trend to be smaller than those generated under the uniform distribution. Figure 1 draws two curves of the ordered p-values under these two hypothesises. The figure clearly shows the difference of the distributions of the ordered p-values under H_0 and H_1.

The leakage detection problem can now be restated as comparing the distribution of the obtained p-values p_1, ..., p_{n_L} with the uniform distribution, or equivalently as detecting the difference between the two curves in Fig. 1. Naturally, to detect the difference between the two curves, we can compare the ordered p-values $p_{(1)} \le p_{(2)} \le ... \le p_{(n_L)}$ with their expected values $1/n_L, 2/n_L, ..., n_L/n_L$ under the uniform distribution. The HC procedure is

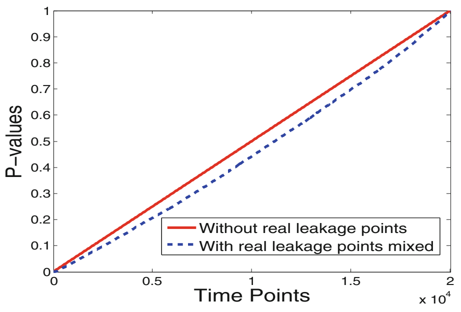


Fig. 1. Comparison of the distributions of ordered p-values under the null hypothesis and under the alternative hypothesis.

based on the normalized distances for these comparisons,

$$\widehat{\mathrm{HC}}_{n_L,i} = \frac{\sqrt{n_L}(i/n_L - p_{(i)})}{\sqrt{p_{(i)}(1-p_{(i)})}}, \quad i = 1, ...n_L. \tag{10}$$

The HC procedure makes the detection if the maximum of these normalized distance $\widehat{\mathrm{HC}}_{n_L,i}$ exceeds a threshold. In contrast, the mini-p procedure only use the first distance $\widehat{\mathrm{HC}}_{n_L,1}$ corresponding to the smallest p-value $p_{(1)}$ only. That is, the mini-p procedure focused on the difference between the two curves in Fig. 1 at the lower-left corner only. When n_L is big, the maximum normalized distance often does not occur at $\widehat{\mathrm{HC}}_{n_L,1}$. Thus the HC procedure can be more effective in detecting the difference by comparing the whole curves instead of using only the pair of extreme points at the lower-left corner.

Formally, the HC procedure is as follows:

(1) Sort the p-values in ascending order $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(n_L)}$.
(2) Calculate $\widehat{\mathbf{HC}}_{n_L,i}$, $i = 1, ...n_L$, from Eq. (10).
(3) The HC statistic $\widehat{\mathrm{HC}}_{n_L,max}$ is defined as,

$$\widehat{\mathrm{HC}}_{n_L,max} = \max_{1 \leq i \leq n_L/2} \widehat{\mathrm{HC}}_{n_L,i}. \tag{11}$$

(4) Compare the obtained HC statistic $\widehat{\mathrm{HC}}_{n_L,max}$ with the HC threshold $b^{\mathrm{HC}}_{n_L,\alpha}$ at a given significance level α. When $\widehat{\mathrm{HC}}_{n_L,max} \leq b^{\mathrm{HC}}_{n_L,\alpha}$, we accept the null hypothesis of no leakage. When $\widehat{\mathrm{HC}}_{n_L,max} > b^{\mathrm{HC}}_{n_L,\alpha}$, we reject the null hypothesis and declare that leakage exists.

The HC threshold $b^{\mathrm{HC}}_{n_L,\alpha}$ is set to the $1-\alpha$ quantile of the HC statistic $\widehat{\mathrm{HC}}_{n_L,max}$'s distribution under the null hypothesis. Since each $\widehat{\mathrm{HC}}_{n_L,j}$ asymptotically follows a standard normal distribution $N(0,1)$ under the null hypothesis, this quantile $b^{\mathrm{HC}}_{n_L,\alpha}$ can obtained by simulation from the n_L standard normal random variables. For large n_L, the threshold $b^{\mathrm{HC}}_{n_L,\alpha}$ can be approximated through the connection to Brownian bridge, for example the calculation formula provided in Li and Siegmund [15].

When $n_L \geq 100$, $b^{\mathrm{HC}}_{n_L,\alpha} \approx 10.10$ and 31.65 for $\alpha = 0.01$ and 0.001 respectively. To compare the mini-p procedure and HC procedure, let us assume that the HC threshold is achieved at the max T-statistic (same as mini-p procedure), and translate the HC threshold in terms of the max T-statistic. The thresholds of maximum T-statistics for mini-p and HC procedures are then listed in the following Table 2.

Table 2. Thresholds of maximum t-test statistics for mini-p and HC procedures.

α	n_L	10^2	10^3	10^4	10^5	10^6	10^7	10^8
0.001	$Tmax_{mini-p}$	4.417	4.892	5.327	5.731	6.110	6.467	6.806
	$Tmax_{HC}$	4.418	4.892	5.327	5.731	6.110	6.468	6.807
0.01	$Tmax_{mini-p}$	3.889	4.416	4.891	5.326	5.730	6.109	6.466
	$Tmax_{HC}$	3.900	4.426	4.899	5.334	5.737	6.116	6.473

In terms of the maximum t-test statistic, we notice that the thresholds for the two procedures are almost the same, with the HC threshold being barely higher. The HC procedure gains more detection power than the mini-p procedure when $\widehat{\mathrm{HC}}_{n_L,max}$ does not occur at the largest t-test statistic. Particularly for devices with some countermeasures, the remaining hard-to-detect leakage points may not have strong leakage signals. Then the test statistics corresponding to those real leakage points may not become the largest, compared to the test statistics at other noisy points on a long n_L trace. However, they do move the curve downward in Fig. 1 without becoming the largest one, and these differences can be picked up by the HC procedure but not by the mini-p procedure.

3.3 HC Framework

Next we present our HC detection framework with salient steps. Figure 2 gives the flow chart, where the steps within the dash-circled box are common in the current TVLA procedure as well.

(I) The evaluator collects a set of physical measurements, then calculate a selected univariate test (e.g., tests in [3–6]) at each time point along the measurement traces. Therefore, n_L statistic values are obtained.

(II) The evaluator finds the cumulative distribution function (CDF) of the above statistic under the null hypothesis H_0 (pure noise model). Using the CDF, the n_L statistic values are translated into n_L p-values (which may also be used by mini-p procedure), e.g., as in Eqs. (5) and (7).

(III) Based on the n_L p-values, the HC procedure in Sect. 3.2 is used to decide if any leakage exists at a given type I error rate α.

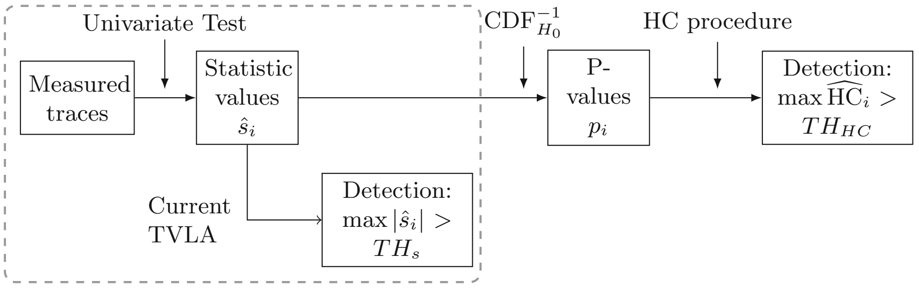

Univariate Test
Measured traces
Statistic values $\hat{s}_i$
Current TVLA
Detection: $\max|\hat{s}_i| > TH_s$
$\text{CDF}_{H_0}^{-1}$
P-values p_i
HC procedure
Detection: $\max \widehat{\text{HC}}_i > TH_{HC}$

Fig. 2. HC leakage detection flow chart.

The current TVLA does not do step (II) and the threshold is not chosen according to a statistical type I error rate. We have shown that it is equivalent to doing step (II) and then conducting a mini-p procedure, which can be made sound by choosing the threshold as in Sect. 3.1. The proposed approach conduct the HC procedure in step (III) instead. A computation module to efficiently calculate the thresholds of HC in step (III) is provided in an extended version of this paper [25].

4 Theory on Optimal Leakage Procedure Using HC

This section introduces the theory on optimality of the HC procedure in high-dimensional statistical testing, and apply it to the leakage detection setting.

4.1 Optimality of the HC Procedure in Mixture Gaussian Testing

We first describe the statistical theory on the optimality of the HC procedure for the common mixture Gaussian setting in literature. That is, we test

$$H_0 : x_i \sim N(0,1), \text{ versus } H_1 : x_i \sim (1-q)N(0,1) + qN(\Delta,1), \tag{12}$$

for observations x_i, $i = 1, ..n_L$. We then show how such theory can be used in the leakage detection testing of (8) versus (9) in the next subsection.

This mixture Gaussian distribution setting can be considered as testing the q proportion of signals with strength Δ in a sample of n_L dimension. The high-dimensional statistical theory indicates how strong (Δ) a signal can be detected for any given sparsity level as $n_L \to \infty$. The common notations in literature re-express the sparsity factor and the signal strength as two parameters $\beta = -\log(q)$ and $\gamma = \Delta^2/[2\log(n_L)]$. On the Euclidean space constructed by these two factors, statistical theory indicates that there is an undetectable region where no statistical tests can distinguish H_0 and H_1 well. More precisely, we first introduce the following definition.

Definition 1. *A statistical test procedure is asymptotically powerless (or asymptotically powerful) if the sum of its type I and type II error rates converges to 1 (or 0) as n_L goes to infinity.*

Theoretical Boundary. For the mixture Gaussian distribution testing problem (12), all statistical procedures are asymptotically powerless in the region below the detection boundary given by equation (1.6) in [11] (proofs in [18]),

$$g(\beta) = \begin{cases} \beta - 1/2 & 1/2 < \beta < 3/4, \\ (1 - \sqrt{1-\beta})^2 & 3/4 \leq \beta < 1. \end{cases} \tag{13}$$

Detection Boundaries of HC and mini-p Procedures. The HC procedure is optimal [11] for testing (12) because the HC procedure is asymptotically powerful when $\gamma > g(\beta)$, i.e., for all parameters (β, γ) above the theoretical minimum detection boundary (13). In contrast, a mini-p procedure is not optimal since it is asymptotically powerless for all parameters (β, γ) below the following boundary, according to the Theorem 1.4 of [11],

$$g_{max}(\beta) = (1 - \sqrt{1-\beta})^2, 1/2 \leq \beta < 1. \tag{14}$$

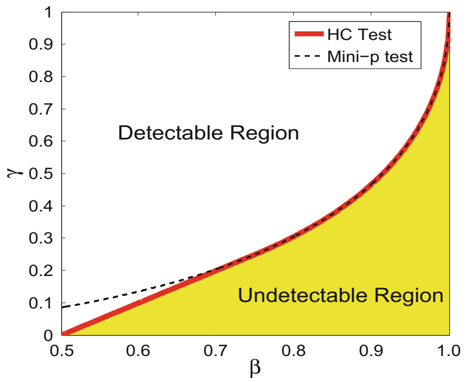


Fig. 3. The undetectable/detectable regions for mini-p test and HC test. (Color figure online)

Figure 3 draws these two detection boundaries (13) and (14). The solid red line is the detection boundary for HC procedure which coincides with the theoretical minimum detection boundary. Below this line (the yellow shade area) is the undetectable region, and above this line is the detectable region. The mini-p procedure's detection boundary curve is plotted as the black dash line, higher than the red line. In the next subsection, we show that these optimality theory do apply to the leakage detection setting.

4.2 Leakage Detection Boundaries and Optimal Procedures

For any test statistic $\hat{s}_i$ based on L_i and for any linear transformation $f(L_i)$, there is always an equivalent test statistics based on $f(L_i)$ that gives exactly the same p-value. Therefore, without loss of generality, we consider the noise and the intermediate variables in (1) and (2) are normalized so that

$$L_i = \tilde{V}\delta_i + r_i, \quad i = 1, \cdots, n_L \tag{15}$$

where $r_i \sim N(0,1)$ is standard Gaussian distributed noise, $\tilde{V}$ is the normalized intermediate variable so that $E(\tilde{V}) = 0$ and $Var(\tilde{V}) = 1$. Hence the model $SNR = Var(\tilde{V}\delta_i)/Var(r_i) = \delta_i^2$ at the i-th time point.

Theoretical Boundary. For simplicity, we consider the simple model where there are $n_0 = qn_L$ POIs with same SNR Δ^2. That is, q proportion of δ_i taking a common non-zero value Δ (and the rest of $\delta_i = 0$). There are n_{tr} observations for each L_i: $L_{i,1}, ..., L_{i,n_{tr}}$. The most powerful test for the i-th univariate hypothesis test must be based on the sufficient statistic [26] for (15): $U_i = (1/n_{tr})\sum_{j=1}^{n_{tr}} \tilde{V}_j L_{i,j}$. Clearly U_i follows the $N(\delta_i, 1/n_{tr})$ distribution, and hence $\sim N(0, 1/n_{tr})$ at time points with no leakage($\delta_i = 0$). Let $x_i = \sqrt{n_{tr}}U_i$. Then the leakage detection problem becomes a mixture Gaussian distribution testing problem, using the sample x_1, ..., x_{n_L}, for

$$H_0 : x_i \sim N(0,1), \text{ versus } H_1 : x_i \sim (1-q)N(0,1) + qN(\sqrt{n_{tr}}\Delta, 1). \tag{16}$$

This is same as the problem (12) except the factor $\sqrt{n_{tr}}$. Therefore, the theoretical minimum detection boundary is given by (13), but with

$$\gamma = n_{tr}\Delta^2/[2\log(n_L)]. \tag{17}$$

Detection Boundaries of HC and mini-p Procedures. For the earlier concrete examples in Sect. 2.2, the test statistic $\widehat{\boldsymbol{s}}_i$ for both the t-test in (6) and the ρ-test in (4), we can show that $\widehat{\boldsymbol{s}}_i$ converges to a $N(\sqrt{n_{tr}\delta_{s_i}^2}, 1)$ distribution as $n_{tr} \to \infty$. The detailed proofs are provided in an extended version [25]. Therefore, given a test statistic $\widehat{\boldsymbol{s}}_i$, we can consider its test SNR as $n_{tr}\delta_{s_i}^2$. At non-leaky time points ($\delta_i = 0$), the test SNR also equals zero, and $\widehat{\boldsymbol{s}}_i \sim N(0,1)$ as described in Sect. 2.2. At POIs with $\delta_i \neq 0$, $\delta_{s_i} = \delta_i$ for the ρ-test statistic (4) in all cases and for the t-test statistic (6) in the fixed-vs-fixed test setup. In the fixed-vs-random setting, $\delta_{s_i} = \delta_i \tilde{V}_{cons}$ for a constant $\tilde{V}_{cons} < 1$ (The proofs are in [25].).

Therefore, for the ρ-test statistic (4) in all cases and for the t-test statistic (6) in the fixed-vs-fixed test setup, $\{\widehat{\boldsymbol{s}}_i : i = 1, ..., n_L\}$ consists a data sample of size n_L for (16). Hence the HC-based leakage procedure achieves the theoretical minimum detection boundary above. But the current TVLA (mini-p) procedure is not optimal with boundary (14).

Remark 1. For the fixed-vs-random t-test, $\{\widehat{\boldsymbol{s}}_i : i = 1, ..., n_L\}$ consists a data sample for a problem similar to (16) but with SNR reduced by a factor $\tilde{V}_{cons}^2$. Thus the HC-procedure's detection boundary $g(\beta)/\tilde{V}_{cons}^2$ is the theoretical minimum detection boundary given $\{\widehat{\boldsymbol{s}}_i : i = 1, ..., n_L\}$. Our proposed HC-based leakage procedure is optimal in combining the n_L given univariate tests in this case too. We do not claim that the univariate test itself (such as the fixed-vs-random t-test, a generic test) is optimal, but rather claim that the procedure framework is optimal in combining the given univariate tests.

Remark 2. When there is only a single POI ($n_0 = 1$, corresponding to sparsity $\beta = 1$), the detection efficiencies are the same for the HC procedure and for the mini-p procedure. As more POIs exist on the trace (i.e., as β decreases), the detection of leakage existence also becomes much easier using HC procedure than using mini-p procedure, which is reflected by the smaller n_{tr} needed to raise γ in (17) above the detection boundary $g(\beta)$ than $g_{max}(\beta)$.

Remark 3. To find exploitable leakage about a particular V, [5] sampled random plaintexts with varying V values for the ρ-test, to detect and locate sparse signals for this targeted V. The fixed-versus-fixed and fixed-versus-random t-tests, being nonspecific, would find more leakage signals along the trace. The choice of sampling scheme and tests affects both the SNR and the sparsity of the leakage signals. The HC procedure can lead to better detection than the mini-p procedure when there are multiple leakages, as likely in the fixed-versus-fixed and fixed-versus-random settings. Note that the identification of the exploitable leakage is a harder question than simply detecting leakage existence. For certification of non-leaky device, the optimal leakage detection procedure proposed here should be applied. To identify exploitable leakage, after leakage detection, specific test such as the ρ-test should be conducted and possibly more traces need to be collected for identification.

Remark 4. The HC procedure above assumed that the noise are independent among different time points along the trace. However, the performance of HC procedure is not affected under the likely short-range dependence [27] (i.e., the dependence among noises is concentrated to nearby time points) in practice. Extending generalized HC procedure [28] for strongly dependent noise for leakage detection can be considered in the future work.

5 Numerical Results

In this section, we investigate the performance of HC procedure and mini-p procedure on synthetic data and real implementations. The results on synthetic data validate the theoretical analysis on the impact of the signal strength and the signal sparsity on the leakage detection performances. Then, the experiments on real traces show the relevance of using the HC metrics in typical case-studies: *(i)* an unprotected and *(ii)* a masked implementation of AES.

5.1 Validation on Synthetic Data

Setup Description. For these experiments, we simulate traces of a complete execution of a 8-bit AES-128 implementation (10 rounds) with a Hamming weight leakage function and Gaussian noise. The 16 Hamming weights corresponding to the 16 intermediate bytes are computed for the plaintext and the result of every `AddRoundKey`, `SubBytes`, and `MixColumns` operation. Each of the 496 calculated values is uniquely reflected in one time sample (hence dictating the traces length) on which random noise following a Gaussian distribution is added. We consider two cases with levels of noise corresponding to SNRs of 0.1 and 0.01. For both cases, the three detection tests discussed in Sect. 4 are applied: *(i)* non-specific t-test with fixed-vs-random plaintexts, *(ii)* non-specific t-test with fixed-vs-fixed plaintexts, and *(iii)* specific ρ-test with random plaintexts.

In order to test the performance of the HC and mini-p procedures, we observe their evolution when adding more and more traces. If a statistic is greater than its respective threshold calculated at level $\alpha = 0.01$ from the method in above sections, we consider that a leakage is detected (returning 1), otherwise there is no detected leakage (returning 0). This experiment is repeated 100 times on independent trace sets. The 100 obtained vectors are then averaged to build success curves. Figure 4 shows the success rates of the HC (red solid curve) and mini-p (blue dash curve) procedures that are applied on the p-values outputted by these three detection tests.

Note: the purpose of these experiments is not to directly compare non-specific and specific leakage detection tests. They are rather chosen because of the different signals they exploit. In the first case, a non-specific detection test aims at finding leakages in a non-sparse signal with a larger amplitude: every sample can lead to detection regardless of its actual usability (i.e. to retrieve the key).

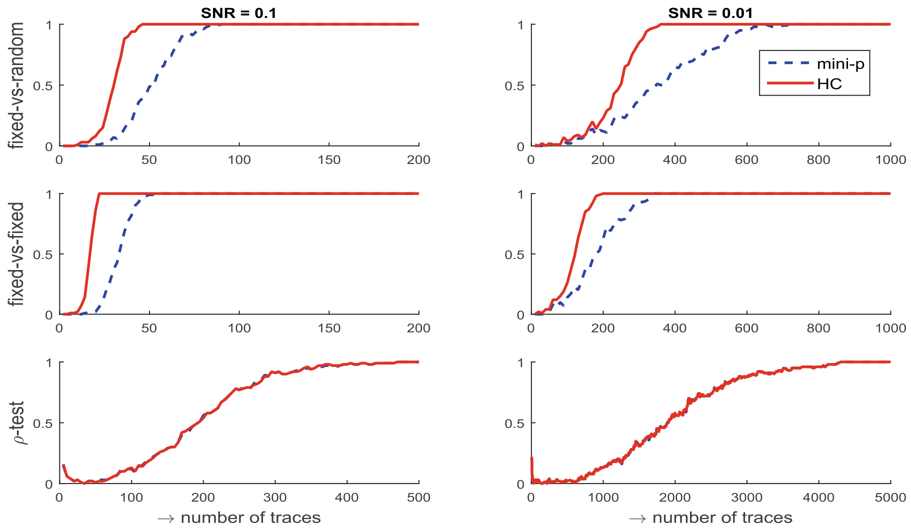


Fig. 4. Success rates curves for the HC (red solid) and mini-p (blue dash) procedures applied on the fixed-vs-random, fixed-vs-fixed, and ρ leakage detection tests. (Color figure online)

In the second case, a specific detection test aims at finding leakages in a sparse signal with lower amplitude: it only spots the useful points-of-interest.

Results Interpretation. The results depicted in Fig. 4 allow us to make the following observations:
(I) On the signal sparsity: the detection based on the HC procedure performs better than the one based on the mini-p only with the non-specific tests, i.e. when the signal is not sparse (all data-dependent samples can be spotted by the test, independent of their exploitability). Conversely, the specific test targets a very specific value. Therefore, the signal is very sparse (there is a single point-of-interest) and the HC and mini-p success rate curves completely match. This first observation supports the detectable region boundaries depicted in Sect. 4. The single point-of-interest in the specific test here is a simulated extreme case. In practice, a single leaky instruction can also lead to multiple points-of-interest on the measured traces (e.g., due to high sampling rate). Then, even for the specific tests, the HC procedure will detect the leakage faster than the mini-p procedure in practice.
(II) On the impact of the noise: as previously observed in the literature [29], increasing the noise leads to decreasing the detection speed by the same factor for a given procedure. Therefore, the ratio between the detection speed of the HC and mini-p procedures remains constant. However, although it does not change much for devices with low levels of noise, it can have an impact on the certification outcome for devices with large levels of noise.
(III) Non-specific detection tests: as previously stated by Durvaux et al. [5] one can notice that appropriately choosing the input of a non-specific test can lead to a better detection: the fixed-vs-fixed test performs approximately twice better than the fixed-vs-random test. Due to our Hamming weight leakage function, the maximum distances are twice larger with the fixed-vs-fixed than with the fixed-vs-random test. Similarly to the impact of the noise, the larger the noise, the bigger the potential impact on a certification outcome.

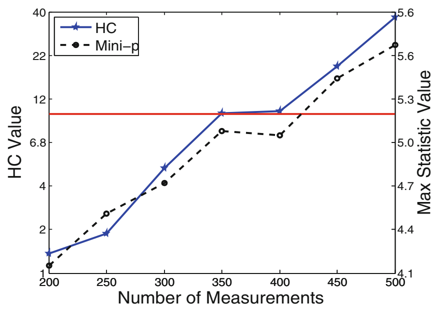
To summarize, these preliminary results mostly show that there is a clear practical improvement of the HC procedure over the mini-p in cases where *(i)* the signal is not sparse, and *(ii)* the SNR is low. In the next experiments, we focus on the ρ leakage detection test.

5.2 Leakage Detection on Real Traces: Unprotected AES

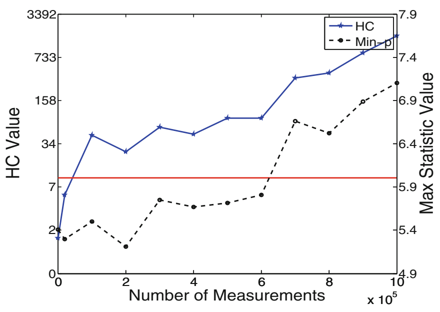
Setup Description. In this section, we investigate the performances of the HC and mini-p procedures on real power traces for non-sparse signal and high SNR. For this purpose, we consider an unprotected AES implementation running on an AVR 8-bit micro-controller embedded on a SASEBO-W board. Power traces are sampled with a LeCroy WaveRunner 640zi oscilloscope that produces $50,000$-sample leakage traces. The results based on a ρ test are given in Fig. 5(a). Instead of previous success rate curves, we show statistical values of HC and mini-p procedures as what an evaluator would use during leakage examination.

They are displayed with respectively the blue solid and the black dash curves (scales are respectively labeled on the left and right sides).

Results Interpretation. Under the significance level of 0.01, with $n_L = 50,000$, the thresholds of maximum ρ test statistic (Fisher's transformation) and HC statistic are 5.2 and 10.1, respectively. In Fig. 5, the red line denotes these cutoffs. Once the obtained statistic value exceeds the red line, evaluators declare that the leakage is detected. The HC procedure detects the existence of leakage with about $n_{tr} = 350$ while the mini-p procedure requires $n_{tr} = 450$. HC procedure is a little more efficient than mini-p procedure, and it coincides with the strong leakage signal strength (estimated SNR around 0.2).



(a) Unprotected AES data



(b) Masked AES data

Fig. 5. Statistic Values of Mini-p and HC procedures on two AES implementations. (Color figure online)

5.3 Leakage Detection on Real Traces: Masked AES

Setup Description. We then illustrate the application of HC procedure on detecting second order leakage on a masked AES implementation, i.e. low SNR (sparsity in this case is hard to estimate, but the results indicate that there are multiple leakage points for masked values). For this purpose, we make use of traces available on the website of the TeSCASE project [30]. The masked implementation of the AES follows the scheme described in [31] and runs on the Virtex-5 FPGA embedded on a SASEBO-GII board. This set of traces contains $N = 1,400,000$ power traces of $n_w = 3125$ samples. It was previously verified that the traces embed no first-order leakage. Then, HC and mini-p procedures are compared for detecting the second-order leakage existence for this protected implementation. Since the centered-product is the natural candidate when attacking second-order leakages [32–35], we use it to combine all pairs of leakages. The result is then used as observations for leakage detection [4]. That is, for a n_w long trace, we examine correlations of the $n_L = (n_w^2 + n_w)/2$ centered-product leakages with the Hamming distance of a targeted SBox (1st SBox byte in last round). The detection results based on ρ test are given in the Fig. 5(b).

Results Interpretation. Under the significance level of 0.01, with $n_L = (n_w^2 + n_w)/2$, the thresholds of maximum ρ test statistic and HC statistic are 6.1 and 10.1, respectively. Compared to unmasked AES, its leakage signal strength is lower, both mini-p and HC procedure require much more measurements to detects the existence of second-order leakages. The HC procedure requires about $n_{tr} = 40,000$ measurements while the mini-p procedure requires $n_{tr} = 620,000$. In other words, in this case-study, the HC procedure allows detecting the leakages 15 times faster than the mini-p procedure.

Acknowledgment. This work has been funded in parts by National Science Foundation grants CNS-1314655, CNS-1337854 and CNS-1563697, and by the European Commission through the H2020 project 731591 (acronym REASSURE) and the ERC project 724725 (acronym SWORD). François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

1. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side-channel resistance validation. In: NIST Non-Invasive Attack Testing Workshop, September 2011. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf
2. Cooper, J., DeMulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., Rohatgi, P.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference (2013). http://icmc-2013.org/wp/wp-content/uploads/2013/09/goodwillkenworthtestvector.pdf
3. Mather, L., Oswald, E., Bandenburg, J., Wójcik, M.: Does my device leak information? an *a priori* statistical power analysis of leakage detection tests. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 486–505. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_25
4. Schneider, T., Moradi, A.: Leakage assessment methodology. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 495–513. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_25
5. Durvaux, F., Standaert, F.-X.: From improved leakage detection to the detection of points of interests in leakage traces. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 240–262. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_10
6. Ding, A.A., Chen, C., Eisenbarth, T.: Simpler, faster, and more robust T-test based leakage detection. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 163–183. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43283-0_10
7. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 326–343. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45608-8_18
8. Nascimento, E., López, J., Dahab, R.: Efficient and secure elliptic curve cryptography for 8-bit AVR microcontrollers. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) SPACE 2015. LNCS, vol. 9354, pp. 289–309. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24126-5_17

9. De Cnudde, T., Bilgin, B., Reparaz, O., Nikova, S.: Higher-order glitch resistant implementation of the PRESENT S-box. In: Ors, B., Preneel, B. (eds.) BalkanCryptSec 2014. LNCS, vol. 9024, pp. 75–93. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21356-9_6
10. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.-X.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 64–81. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16763-3_5
11. Donoho, D., Jin, J.: Higher criticism for detecting sparse heterogeneous mixtures. Ann. Stat. **32**, 962–994 (2004)
12. Donoho, D., Jin, J.: Higher criticism thresholding: optimal feature selection when useful features are rare and weak. Proc. Nat. Acad. Sci. **105**, 14790–14795 (2008)
13. Fan, J., Lv, J.: Sure independence screening for ultra-high dimensional feature space. J. Royal Stat. Soc. Ser. B **70**, 1–35 (2008)
14. Fan, J., Feng, Y., Song, R.: Nonparametric independence screening in sparse ultra-high-dimensional additive models. J. Am. Stat. Assoc. **106**(494), 544–557 (2011)
15. Li, J., Siegmund, D., et al.: Higher criticism: p-values and criticism. Ann. Stat. **43**(3), 1323–1350 (2015)
16. Donoho, D., Jin, J., et al.: Higher criticism for large-scale inference, especially for rare and weak effects. Stat. Sci. **30**(1), 1–25 (2015)
17. Wu, Z., Sun, Y., He, S., Cho, J., Zhao, H., Jin, J.: Detection boundary and higher criticism approach for rare and weak genetic effects. Ann. Appl. Stat. **8**(2), 824–851 (2014). https://doi.org/10.1214/14-AOAS724
18. Ingster, Y.I.: Minimax detection of a signal for i (n)-balls. Math. Methods Stat. **7**(4), 401–428 (1998)
19. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_1
20. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_26
21. Bär, M., Drexler, H., Pulkus, J.: Improved template attacks. In: International Workshop on Constructive Side-Channel Analysis and Secure Design (2010)
22. Elaabid, M.A., Meynard, O., Guilley, S., Danger, J.-L.: Combined side-channel attacks. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 175–190. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-17955-6_13
23. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 253–270. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_17
24. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Less is more. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 22–41. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_2
25. Zhang, L., Ding, A.A., Durvaux, F., Standaert, F.-X., Fei, Y.: Towards sound and optimal leakage detection procedure, Cryptology ePrint Archive, Report 2017/287 (2017). http://eprint.iacr.org/2017/287
26. Lehmann, E.L., Romano, J.P.: Testing Statistical Hypotheses. Springer, New York (2006). https://doi.org/10.1007/0-387-27605-X
27. Hall, P., Jin, J.: Properties of higher criticism under strong dependence. Ann. Stat. **36**, 381–402 (2008)

28. Barnett, I., Mukherjee, R., Lin, X.: The generalized higher criticism for testing SNP-set effects in genetic association studies. J. Am. Stat. Assoc. **112**(517), 64–76 (2017)
29. Mangard, S., Oswald, E., Standaert, F.X.: One for all - all for one: unifying standard differential power analysis attacks. IET Inf. Secur. **5**(2), 100–110 (2011)
30. Testbed for side channel analysis and security evaluation (2014). http://tescase.coe.neu.edu
31. Akkar, M.-L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_26
32. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
33. Schramm, K., Paar, C.: Higher order masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006). https://doi.org/10.1007/11605805_14
34. Prouff, E., Rivain, M., Bevan, R.: Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
35. Ding, A.A., Zhang, L., Fei, Y., Luo, P.: A statistical model for higher order DPA on masked devices. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 147–169. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_9

Connecting and Improving Direct Sum Masking and Inner Product Masking

Romain Poussier[1(✉)], Qian Guo[1], François-Xavier Standaert[1], Claude Carlet[2], and Sylvain Guilley[3]

[1] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Louvain-la-Neuve, Belgium
romain.poussier@uclouvain.be

[2] LAGA, CNRS, Univ. Paris VIII and Paris XIII, Paris, France

[3] Secure-IC S.A.S, TELECOM-ParisTech, Crypto Group, Paris-Saclay University, CNRS, Paris, France

Abstract. Direct Sum Masking (DSM) and Inner Product (IP) masking are two types of countermeasures that have been introduced as alternatives to simpler (e.g., additive) masking schemes to protect cryptographic implementations against side-channel analysis. In this paper, we first show that IP masking can be written as a particular case of DSM. We then analyze the improved security properties that these (more complex) encodings can provide over Boolean masking. For this purpose, we introduce a slight variation of the probing model, which allows us to provide a simple explanation to the "security order amplification" for such masking schemes that was put forward at CARDIS 2016. We then use our model to search for new instances of masking schemes that optimize this security order amplification. We finally discuss the relevance of this security order amplification (and its underlying assumption of linear leakages) based on an experimental case study.

1 Introduction

Masking is among the most investigated countermeasures against side-channel analysis. It aims at performing cryptographic computations on encoded (aka secret shared) data in order to amplify the impact of the noise in the adversary's observations [10,13,14,31]. For example, in the context of block ciphers, a lot of attention has been paid to the efficient exploitation of simple encodings such as additive (e.g., Boolean ones in [12,25,32]) or multiplicative ones (e.g., [19,20]). Very summarized, the main advantage of these simple encodings is that they enable efficient implementations [22].

In parallel, an alternative trend has investigated the potential advantages of slightly more complex encodings. Typical examples include polynomial masking (e.g., [18,21,33]), Inner Product (IP) masking [1,2,17] and code-based masking (e.g., [5–9]). While computing over these encodings is generally more expensive [25], the recent literature has shown that their elaborate algebraic structure also leads to improved security properties. For example, it can decrease the information leakages observed in "low noise conditions" [1,2,18,21,33]. Also, it can improve the "statistical security order" (or security order in the bounded moment leakage model [3]) in case of linear leakage functions [8,26,38]. So while it is an open problem to find out which masking scheme offers the best security vs. efficiency tradeoff for complete implementations in actual devices, the better understanding and connection of simple and complex encoding functions appears as a necessary first step in this direction.

For this purpose, and as a starting point, we note that it has already been shown in [2] that IP masking can be viewed as a generalization of simpler encodings (Boolean, multiplicative, affine and polynomial). So our focus in this paper will be on the connection between IP masking and the Direct Sum Masking (DSM) [5,9], which is a quite general instance of code-based masking. Our contributions in this respect are as follows:

First, we connect IP masking and DSM by showing that the first one can be seen as a particular case of the latter one. Second, we analyze the security properties of these masking schemes. In particular, we show that the "security order amplification" put forward in previous works can be easily explained thanks to (a variation of) the probing model [27], by considering bit-level security, rather than larger (field-element-level) security. Thanks to this connection, we then express how to best optimize the security order amplification (i.e., the bit-level security) based on the dual distance of a binary code. We further perform an informed search on code instances which allows us to improve the state-of-the-art parameters for IP encodings. We finally propose experiments discussing the interest and limitations of security order amplification in practice (i.e., the relevance of the linear leakage assumption).

Cautionary note. Our focus in this paper is on encodings. Admittedly, an even more important issue is to compute (in particular, multiply) efficiently over encodings. In this respect, while the literature on IP masking provides solutions to this problem [1,2], it remains an open challenge to describe efficient multiplication algorithms for DSM.

2 Connecting DSM and IP Masking

In this section we first introduce the two masking schemes that we will analyze, namely IP masking and DSM. We then show how these two methods are connected, focusing only on their functional description (security will be investigated in Sect. 3).

2.1 Notations

We use capital letters for random variables and small caps for their realizations. We denote the conditional probability of a random variable A given B with $\mathrm{P}[A|B]$. We use sans serif font for functions (e.g., F) and calligraphic fonts for sets (e.g., $\mathcal{A}$). Given a field $\mathbb{K}$, we denote by $a \cdot b$ the field multiplication between two elements a and b. We denote by $[a]_2$ the binary vector representation of some element $a \in \mathbb{F}_{2^k}$ for some k. We use capital bold letters for matrices (e.g., $\mathbf{M}$) and small bold caps for raw vectors (e.g., $\mathbf{v}$). We denote by $\mathbf{v}(i)$ the i-th element of a vector $\mathbf{v}$. We denote by $\mathbf{M}^T$ (resp. $\mathbf{v}^T$) the transpose of a matrix M (resp. a vector $\mathbf{v}$). The inner product between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is noted $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$. In the rest of the paper, x denotes a k-bit secret value that we wish to mask and $[x]_2$ its binary vector representation.

2.2 Inner Product Masking

IP masking was introduced in [1,2,17] as a generalization of Boolean masking. Instead of simply splitting a secret value as the sum of random shares, it decomposes the secret as the inner product between random values and a public vector. More formally, the first step of the IP encoding is to select a public vector $\mathbf{l} = (l_0, ..., l_{n-1}) \in \mathbb{F}_{2^k}^n \backslash \{0\}$ (with l_0 generally set as $l_0 = 1$ for performance reasons), where n is the number of shares. A sensitive variable $x \in \mathbb{F}_{2^k}$ is then encoded as the vector $\mathbf{s}_{IP} = (s_0, ..., s_{n-1}) \in \mathbb{F}_{2^k}^n$ such that $x = \langle \mathbf{l}, \mathbf{s}_{IP} \rangle$. Algorithm 1 describes the masking initialization procedure, where the function $\mathsf{rand}(\mathbb{F}_{2^k})$ returns a random element uniformly from $\mathbb{F}_{2^k}$. Boolean masking is the particular case of IP masking where $l_i = 1$ for $i \in [0, n-1]$.

Algorithm 1. IPMask.

```
Require: $x, \mathbf{l}, n$
Ensure: $\mathbf{s}_{IP}$ such that $x = \langle \mathbf{l}, \mathbf{s}_{IP} \rangle$
  for $i = 1$ to $n - 1$ do
    $s_i \leftarrow \mathsf{rand}(\mathbb{F}_{2^k})$
  end for
  $s_0 = x + \sum_{i=1}^{n-1} l_i \cdot s_i$
  return $\mathbf{s}_{IP}$
```

2.3 Direct Sum Masking

DSM [5,9] describes masking from an error correcting code viewpoint. As opposed to IP masking, this scheme works on the bit level. That is, a sensitive variable x is viewed as belonging to $\mathbb{F}_2^k$ instead of $\mathbb{F}_{2^k}$ and is thus represented as $[x]_2$. It allows adding an arbitrary amount m of bits of randomness to the encoding of $[x]_2$ (i.e., not necessarily a multiple of k as in IP masking). As a result, the final encoding $\mathbf{s}_{DSM}$ of $[x]_2$ lays in $\mathbb{F}_2^{k+m}$. As a first step, the vector space $\mathbb{F}_2^{k+m}$ is decomposed in two subspaces C and D of dimensions k and m:

$$\mathbb{F}_2^{k+m} = C \oplus D, \tag{1}$$

where C and D respectively represent the spaces where the sensitive variable and the mask lay. That is, the sensitive variables (resp., the mask) are the code words of C (resp., D) of length of $k + m$. We denote by $\mathbf{G}$ and $\mathbf{H}$ the generator matrices of C and D. The encoding of $[x]_2$ is the vector $\mathbf{s}_{DSM} = [x]_2\mathbf{G} + \mathbf{y}\mathbf{H}$, where $\mathbf{y} \in \mathbb{F}_2^m$ is a random binary vector. Recovering $[x]_2$ (i.e., decoding) or $\mathbf{y}$ from $\mathbf{s}_{DSM}$ can then be achieved thanks to a projection on their respective space. We stress the fact that while this scheme has been designed to thwart both side-channel and fault attacks (if C and D are orthogonal), we only focus on the side-channel part.

2.4 Unifying DSM and IP Masking

From the previous description of IP masking and DSM, we now show how these two schemes are connected. We recall that the IP encoding of a sensitive variable x using n shares is the vector $\mathbf{s}_{IP} = (s_0 = x + l_1 \cdot s_1 + ... + l_{n-1} \cdot s_{n-1}, s_1, ..., s_{n-1}) \in \mathbb{F}_{2^k}^n$. In order to make the connection with DSM, we first have to move its base field from $\mathbb{F}_2$ to $\mathbb{F}_{2^k}$. That is, we want the final DSM encoding to belong to $\mathbb{F}_{2^k}^n$. We next decompose $\mathbb{F}_{2^k}^n$ using two supplementary subspaces C and D such that $\mathbb{F}_{2^k}^n = C \oplus D$, where the dimension of C is 1 and the dimension of D is $n-1$. As in the previous subsection, we denote by $\mathbf{G}$ and $\mathbf{H}$ the generator matrices of C and D that we define as follow (where each element belong to $\mathbb{F}_{2^k}$):

$$\mathbf{G} = \begin{pmatrix} 1 \; 0 \ldots 0 \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} l_1 & 1 & 0 & \ldots & 0 \\ l_2 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n-1} & 0 & \ldots & 0 & 1 \end{pmatrix}. \tag{2}$$

Equation 3 then shows the encoding vector $\mathbf{s}_{MIX}$ of a secret $x \in \mathbb{F}_{2^k}$ using a randomness vector $\mathbf{y} = (y_1, ..., y_{n-1}) \in \mathbb{F}_{2^k}^{n-1}$:

$$\begin{aligned} \mathbf{s}_{MIX} &= x\mathbf{G} + \mathbf{y}\mathbf{H}, \\ &= (x, 0, ..., 0) + (l_1 \cdot y_1 + ... + l_{n-1} \cdot y_{n-1}, y_1, ..., y_{n-1}), \\ &= (x + l_1 \cdot y_1 + ... + l_{n-1} \cdot y_{n-1}, y_1, ..., y_{n-1}), \\ &= \mathbf{s}_{IP}. \end{aligned} \tag{3}$$

The encoding of an IP masking can thus be written by adapting the DSM scheme base field and choosing the generating matrices accordingly. However, this modification discards one property of the original DSM scheme. Namely, the number of bits of randomness added to the encoding cannot be arbitrarily chosen as it has to be a multiple of k. Besides, we note that the discussions in [5] additionally required the codes C and D to be orthogonal. Yet, this additional property is not required in our discussions that focus only on side-channel security, and where the secret x can be recovered using a projection: $x = \langle \mathbf{l}, \mathbf{s}_{MIX} \rangle$.

3 Probing Security and Bit Probing Security

In this section, we discuss the side-channel resistance of the IP masking and DSM in the probing model [27]. For each method, we look at the security of the encoding. In the case of IP masking, we assume that the size k of the base field corresponds to the word size of the implementation and the probes allow the adversary to observe such field elements. As the DSM works on the bit level, we additionally introduce the bit-probing model, where each probe can only look at one bit of the encoding. We finally make the link between the (general, ie.g., field-level) probing security and the bit-probing security of the inner product masking. This connection will be used in the next section in order to explain the security order amplification of the IP masking.

3.1 Probing Security

The probing model introduced in [27] formalizes the security improvement obtained with the masking countermeasure. Informally, dth-order probing security ensures that the distribution of any d or less intermediate variables manipulated during the algorithm execution is independent of any secret value. From an attacker point-of-view, it implies that only the combination of at least $d+1$ intermediate variables can give information on the secret. As a result, the practical security increases exponentially in the number of shares, which is intuitively explained by the fact that the adversary has to estimate higher-order statistical moments, a task of which the sampling complexity grows exponentially in the order [10,13,14,31]. In the case of IP masking with n shares, previous works showed that the encoding has a probing security of order $d = n - 1$ [1,2].

3.2 Bit-Probing Security

Thanks to the link exhibited in the previous section, we naturally have that DSM is secure in the probing model as well, which also follows the analysis in [5,9]. However, since DSM works at the bit level, we additionally define the bit-probing security as the security in a tweaked probing model, where each probe can evaluate only one bit of the encoding (even if this encoding is defined for larger fields). In this model, the security order is thus the maximum number d' such that any combination of d' bits of $\mathbf{s}_{DSM}$ is independent of the secret $[x]_2$. More formally, the bit-probing security of the DSM scheme is given by:

Proposition 1. *Let C and D two codes of generator matrices $\mathbf{G}$ and $\mathbf{H}$ define a DSM encoding. Let k and m respectively be the dimensions of C and D. The bit-probing security of the DSM encoding defined by C and D is equal to the distance of the dual code (called the dual distance) of D minus 1.*

Proof. Let $\mathbf{s}$ be the encoding of some value $[x]_2$. We have $\mathbf{s}_{DSM} = [x]_2\mathbf{G} + \mathbf{y}\mathbf{H}$, a vector of $k + m$ bits. The bit-probing security is the number d' such that at least $d' + 1$ elements of $\mathbf{s}_{DSM}$ are required to recover at least one bit of $[x]_2$. If we consider the system given by Eq. 4:

$$([x]_2\mathbf{G} + \mathbf{y}\mathbf{H})^T = \mathbf{s}_{DSM}^T, \tag{4}$$

where only the right part is known, $d' + 1$ is equal to the smallest number of sub-equations of this system that allows recovering at least one bit of $[x]_2$. We assume that the dual distance of D is equal to $d + 1$, which is the minimum number of columns of $\mathbf{H}$ that can be linearly dependent. This means that at least $d + 1$ sub-equations of the system in Eq. 4 are required to suppress the influence of $(\mathbf{yH})^T$, and thus get information on $[x]_2$. As a result, the bit-probing security of the DSM encoding is equal to d. $\square$

Note that as will be clear next, the bit-probing security order (which can be higher than the probing security order) does not always guarantee a higher practical security order (i.e., in the bounded moment or noisy leakage models [3,31]) than predicted by the (field-level) probing security order. Yet, it will be instrumental in explaining the security order amplification for certain types of leakage functions put forward in [38].

3.3 Inner Product and Bit-Probing Security

We now consider the bit-probing security of the IP masking encoding. In Sect. 2.4, we showed how IP masking and DSM are linked by changing the base field of the DSM scheme from $\mathbb{F}_2$ to $\mathbb{F}_{2^k}$. In order to assess the bit-probing security of the IP masking by using Proposition 1, we have changed the base field back from $\mathbb{F}_{2^k}$ to $\mathbb{F}_2$. As a result, we want a new encoding $\mathbf{s}_{MIX2}$ that belongs to $\mathbb{F}_2^{kn}$ such that each bit of $\mathbf{s}_{MIX}$ and $\mathbf{s}_{MIX2}$ are the same: $[\mathbf{s}_{MIX}(i)]_2 = (\mathbf{s}_{MIX2}(ki), ..., \mathbf{s}_{MIX2}(k(i+1)-1))$.

As a preliminary, we first define by $\mathbf{L}_i$ (with $i \in [1, n-1]$) the $k \times k$ binary matrices that represent the multiplication by l_i in $\mathbb{F}_{2^k}$. That is, given some value $x \in \mathbb{F}_{2^k}$, we define $\mathbf{L_i}$ such that $[l_i \cdot x]_2 = (\mathbf{L}_i \times {[x]_2}^T)^T$. The matrix $\mathbf{L}_i$ can be constructed such that its j-th column is equal to $[\alpha^j \cdot l_i]_2$, where α is a root of the polynomial used to create $\mathbb{F}_{2^k}$.

We now define two codes C and D such that $\mathbb{F}_2^{kn} = C \oplus D$, the dimension of C is k, and the dimension of D is $k(n-1)$, with their generator matrix $\mathbf{G}$ and $\mathbf{H}$ specified as follow:

$$\mathbf{G} = \begin{pmatrix} 1 \dots 1 \; 0 \dots 0 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \mathbf{L}_1 & \mathbf{1}_k & \mathbf{0}_k & \dots & 0_k \\ \mathbf{L}_2 & 0_k & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0_k \\ \mathbf{L}_{n-1} & 0_k & \dots & 0_k & \mathbf{1}_k \end{pmatrix}, \tag{5}$$

such that the first k columns of $\mathbf{G}$ are 1 and the next $k(n-1)$ are 0. Here, $\mathbf{1}_k$ denotes the $k \times k$ identity matrix and $\mathbf{0}_k$ denotes the $k \times k$ zero matrix. Equation 6 then shows the encoding vector $\mathbf{s}_{MIX2}$ of a secret $[x]_2 \in \mathbb{F}_2^k$ using a randomness vector $\mathbf{y} = (y_1, ..., y_{k(n-1)}) \in \mathbb{F}_2^{k(n-1)}$:

$$\begin{aligned} \mathbf{s}_{MIX2} &= [x]_2\mathbf{G} + \mathbf{yH}, \\ &= ([x]_2, 0, ..., 0) + ([l_1 \cdot y_1]_2 + ... + [l_{n-1} \cdot y_{n-1}]_2, [y_1]_2, ..., [y_{n-1}]_2), \\ &= ([\mathbf{s}_{MIX}(0)]_2, ..., [\mathbf{s}_{MIX}(n-1)]_2)\cdot \end{aligned} \tag{6}$$

From Proposition 1, we know that the bit-probing security of $\mathbf{s}_{MIX2}$ corresponds to the dual distance of $\mathbf{H}$ minus 1 (which depends on the selection of the $\mathbf{l} = (l_1, ..., l_{n-1})$ vector of the IP masking, as already hinted in [38]). As a result, the best bit-probing security using n shares can be achieved by selecting $\mathbf{l}$ such that the dual distance of $\mathbf{H}$ is maximized.

4 Security Order Amplification

Under some physical assumption that will be discussed later, it has been observed that the concrete security order of the IP encoding (in the bounded moment or noisy leakage models [3,31]) can be higher than the one given by its probing security [38]. In this section, we provide a formal explanation of this phenomenon that we so-far denoted as security order amplification. We first introduce the bounded moment model that we will use for this purpose [3]. We then apply this model to the IP masking, and explain its link with security order amplification.

4.1 Bounded Moment Model

The bounded moment leakage model has been introduced in [3], mainly in order to formalize the security of parallel implementations and to connect probing security with current (moment-based) evaluation practices such as [35].

For our following discussions, we will consider a n-share masked implementation with encoding $\mathbf{s} = (s_0, ..., s_{n-1})$ of a secret x that manipulates all the shares within N cycles. As in [3], we denote by $\mathcal{Y}_c$ the set of shares that are manipulated during the cycle c ($0 \leq c \leq N-1$) and by n_c the cardinal of $\mathcal{Y}_c$. We assume that the random variable L_c that represents the leakage associated to the computation during the cycle c follows a linear model:

$$L_c = \alpha_c^0 \mathsf{L}_c^0(\mathcal{Y}_c(0)) + ... + \alpha_c^{n_c-1} \mathsf{L}_c^{n_c-1}(\mathcal{Y}_c(n_c-1)) + R_c, \tag{7}$$

where L_c^i denotes the deterministic leakage part associated to the manipulation of the share $\mathcal{Y}_c(i)$ and R_c a random noise variable. Note that by linear model, we mean that the different L_c's are summed in Eq. 7, which is needed to ensure that the leakages corresponding to different shares are independent (otherwise even the probing security order will not be reflected in the bounded moment or noisy leakage models). By contrast, so far we do not assume that the L_c's are linear (this will be only needed for security order amplification).

A fully serial implementation corresponds to the case $N = n$ and $n_c = 1, c \in [0, N-1]$. On the opposite, a fully parallel implementation would be $N = 1$ and $n_0 = n$. In the later case, higher-order probing security can never be achieved as a single variable contains the information on all shares. Hence, the bounded moment model has been introduced to characterize the security of such fully parallel implementations. Basically, having a bounded moment security of order d means that any statistical moment up to the order d of the leakage distribution $\{\mathsf{L}_c\}_{c=0}^{N-1}$ is independent of the secret. More formally, Definition 1 describes the bounded moment security.

Definition 1. *Let $\{L_c\}_{c=0}^{N-1}$ be the leakages of a N cycles parallel masked implementation that manipulates a secret x. We denote by E the expectation operation. This implementation is security at order d if, for all N-tuples $d_i \in \mathbb{N}^N$ such that $\sum_{i=0}^{N-1} d_i \leq d$, we have that $\mathsf{E}(\mathsf{L}_0^{d_0} \times ... \times \mathsf{L}_{N-1}^{d_{N-1}})$ is independent of x.*

Interestingly, it has been shown in [3] that proving security at order o in the probing model (for a serial n-cycle implementation) implies security at order o in the bounded moment model for the corresponding parallel (1-cycle) implementation. We will use this theorem in the next subsection to prove the security order amplification of the inner product masking.

4.2 Security Order Amplification for IP Masking

We now assume an implementation of the IP masking with n shares on $\mathbb{F}_2^k$, where one share corresponds to one variable. That is, we consider the encoding $\mathbf{s}_{MIX} = (s_0, ..., s_{n-1})$ such that each s_i is manipulated independently. We denote by L_i the random variable that represents the leakage on s_i. We assume that the different L_i's are independent and are the sum of a deterministic and random part: $L_i = \mathsf{L}_i(s_i) + R_i$, where L_i denotes the deterministic part of the leakage and R_i denotes a random noise variable. As stated in Sect. 3.1, this encoding has a probing security of order $d = n - 1$. However, it has been noticed in [38] that the actual security of the encoding can be higher than d if the leakage function is linear in the bits of the variable. That is, in this case, information on the secret can be only obtained by estimating a statistical moment d' of the leakage distribution, with $d' \geq d$. This can be intuitively explained by the public vector $\mathbf{l}$ that mixes the bits of the different shares, as opposed to Boolean masking where knowing the first bit of each share directly reveals the first bit of the secret. More formally, the security amplification property of the inner product masking is given by Proposition 2.

Proposition 2. *Let $\mathbf{s}_{MIX} = (s_0, ..., s_{n-1})$ be the n shares of the IP encoding vector defined by the public vector $\mathbf{l} = (1, l_1, ..., l_{n-1})$. If the functions L_i manipulating these shares are linear in the bits of s_i, the bounded moment security order d' of the IP encoding given by $\mathbf{s}_{MIX}$ is equal to the bit-probing security of its equivalent encoding $\mathbf{s}_{MIX2}$.*

Proof. As we assume that the L_i's are linear in the bits of s_i, the leakage L_i can be represented as follows:

$$\begin{aligned}
L_i &= \mathsf{L}_i(s_i) + R_i,\\
&= \alpha_i + \alpha_i^0 \times [s_i]_2(0) + ... + \alpha_i^{k-1} \times [s_i]_2(k-1) + R_i,\\
&= \alpha_i^0 \times ([s_i]_2(0) + \frac{\alpha_i}{\alpha_i^0}) + ... + \alpha_i^{k-1} \times [s_i]_2(k-1) + R_i,\\
&= \alpha_i^j \times \mathsf{F}_i^0([s_i]_2(0)) + ... + \alpha_i^{k-1} \times \mathsf{F}_i^k([s_k]_2(0)) + R_i, \qquad (8)
\end{aligned}$$

with $\mathsf{F}_i^j, j \in [0, k-1]$ a deterministic function in the bit j of s_i and $(\alpha_i, \alpha_i^0, ..., \alpha_i^{k-1}) \in \mathbb{R}^{k+1}$. We can see that the last line of Eq. 8 has the same form as Eq. 7. As we have n different leakages L_i, each one linearly manipulating k bits, the full leakages $\{L_i\}_{i=0}^{n-1}$ of $\mathbf{s}_{MIX}$ can be viewed as the leakages of a parallel implementation of $\mathbf{s}_{MIX2}$ with $N = n$ cycles, each one manipulating $n_c = k$ single-bit variables. As a result, the bounded security of a serial implementation (called A) of $\mathbf{s}_{MIX}$ is the same as a parallel implementation (called B) of $\mathbf{s}_{MIX2}$ with $N = n$ and $N_c = k$. From [3], we know that proving the bounded security of B is equivalent to proving the probing security of its serial implementation. As the probing security of the serial implementation of B corresponds to the case where one probe can only evaluate one bit, it corresponds to the bit-probing security of $\mathbf{s}_{MIX2}$, which conclude the proof. □

Intuitively, this result simply corresponds to the observation that while probing security at order d implies bounded moment security at order d in case the leakages of the shares are independent (with each share a field element), bit-security at order d' implies bounded moment security at order d' in the case where not only the leakages of the shares are independent (with each share being a field element), but also the different bits of each field element is manipulated independently (which is ensured by the linear leakage assumption). This result implies that, under the linear leakage assumption, maximizing the bounded moment security of the IP encoding $\mathbf{s}_{MIX}$ is the same as maximizing the bit-probing security of $\mathbf{s}_{MIX2}$. The next step is thus to find the best public vectors $\mathbf{l}$ so that the bit-probing security of $\mathbf{s}_{MIX2}$ is maximized.

5 Searching for Good Codes

As shown in Proposition 1, the key parameter characterizing the bit-probing security is the dual distance of the linear code $\mathcal{D}$ with generator matrix:

$$\mathbf{H} = \begin{pmatrix} \mathbf{L}_1 & \mathbf{I}_k & \mathbf{0}_k & \dots & 0_k \\ \mathbf{L}_2 & 0_k & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0_k \\ \mathbf{L}_{n-1} & 0_k & \dots & 0_k & \mathbf{I}_k \end{pmatrix}, \tag{9}$$

where $\mathbf{I}_k$ is an identity matrix with dimension k and $\mathbf{L}_i$ is the matrix representation of a finite field element l_i. Therefore, we have the following proposition.

Proposition 3. *The problem of searching for instantiations of an IP masking scheme with good bit-probing security is equivalent to that of searching for an $[nk, k]$ linear code $\mathcal{C}_g$ over $\mathbb{F}_2$ with large minimal distance and generator matrix:*

$$\mathbf{G}_g = \left(\mathbf{I}_k \; \mathbf{L}_1^T \; \mathbf{L}_2^T \; \dots \; \mathbf{L}_{n-1}^T\right). \tag{10}$$

The best possible linear codes with a small dimension k (e.g., $k \leq 8$) are well-studied in literature, see [4,23,37]. Therefore, the minimal distance of $\mathcal{C}_g$ can be upper-bounded. Moreover, the sub-matrix $\mathbf{L}_i^T$ in $\mathbf{G}_g$ is connected to the underlying irreducible polynomial $g(x) \in \mathbb{F}_2[X]$, which defines the field extension from $\mathbb{F}_2$ to $\mathbb{F}_{2^k}$.

We now consider the practically-relevant case study of implementing the AES securely, i.e., when $k = 8$ and we use the AES polynomial $x^8 + x^4 + x^3 + x + 1$. In the following subsection, we show that one can choose the l_i's to form an IP masking scheme with bit-probing security that is close to the upper bound defined by the best possible eight dimensional binary linear codes.

5.1 Application: 8-bit Implementation of the AES

The problem of determining the largest possible minimum distance of an eight dimensional binary linear code is settled in [4], i.e., it is equal to or slightly smaller than the distance defined by the Griesmer Bound [24].
The companion matrix (see [28]) of $g(x) = x^8 + x^4 + x^3 + x + 1$ is defined as:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \\ 1 & 1 & \dots & 0 & 0 \end{pmatrix}, \tag{11}$$

whose last row is of the form (1 1 0 1 1 0 0 0). Thus, all the possible field elements of $\mathbb{F}_{2^8}$ can be enumerated as:

$$\sum_{j=0}^{7} a_j \mathbf{A}^j,$$

for all $\mathbf{a} = (a_0, a_1, \dots, a_7) \in \mathbb{F}_2^8$. We next use three approaches for finding good linear codes with generator matrix satisfying the constraint in Eq. 10.

Exhaustive search: When n is small, i.e., less than 4, we can do a brute-force search. That is, we enumerate all possible generator matrices $\mathbf{G}_g$ with the same form as that in Eq. (10), and then test its minimum distance.

Random search: We choose $\mathbf{L}_i$ at random to construct $\mathbf{G}_g$ and then test its minimum distance.

Inductive search: We construct good $[8n, 8]$ linear codes satisfying Eq. (10) from good $[8(n - n_0), 8]$ linear codes satisfying Eq. (10), where n_0 is a small positive integer (e.g., $1, 2, 3$, or 4). That is, we fix the first $8(n - n_0)$ columns of $\mathbf{G}_g$ as the found generator matrix of a good code with length $8(n - n_0)$, exhaust all possible $\mathbf{L}_i$'s, for $i = n - n_0, \dots, n - 1$, and then test its minimum distance.

The numerical results by running Magma are shown in Table 1, where the column n is the number of shares, the column $n \cdot k$ is the code length, the column d_{best}^{IP} is the best minimum distance found from linear codes with a generator matrix satisfying Eq. (10), the column d_{best}^{U} is the upper bound derived from the best achievable minimum distance for any $[8n, 8]$ linear codes, and the last column Δ is the difference between the prior two columns (i.e., the gap between IP masking and DSM). It is clear from this table that IP masking can achieve near-optimal bit-probing security if the number of shares is relatively small. Actually, most of the interesting choices of n in practice are covered in this table (since, due to performance reasons, state-of-the-art implementations of IP masking so far did not go beyond 2 or 3 shares). The constructed good codes also show an approach to instantiate IP masking with good bit-probing security. That is, one can determine the finite field elements l_i's from the found generator matrix $\mathbf{G}_g$ corresponding to a good linear code. We did exhaustive search for $n = 2, 3, 4$, random search for $n = 5, 6$, and inductive search for $n = 7$. Therefore, we cannot rule out the possibility of finding a linear code to reach the upper bound when $n \geq 6$ with more computational efforts.[1]

Concretely and as an example, this table shows that when considering IP masking with three shares, the standard probing model guarantees a security order 2. In case the shares only give rise to linear leakages, the bit-level probing model guarantees a security of order 7.

6 Experimental Validation

The previous positive results admittedly (highly) depend on a physical assumption (i.e., linear leakages) that may not be perfectly respected. So in order to

Table 1. The best linear codes corresponding to an IP masking scheme found by Magma. The field extension from $\mathbb{F}_2$ to $\mathbb{F}_{2^k}$ is defined by the AES polynomial.

n	$n \cdot k$	d_{best}^{IP}	d_{best}^{U}	Δ
2	16	4	5	-1
3	24	8	8	0
4	32	12	13	-1
5	40	16	16	0
6	48	21	22	-1
7	56	23	24	-1

[1] Note that the inductive search allows us to find good linear codes with relatively large minimum distance rather quickly. For instance, we can easily obtain a desired $[72, 8, 30]$ linear code by the inductive search with the code length $8n$ increasing by 16 gradually from 24 to 72. The gap Δ here is only -2. This task takes less than 2 min when using the online Magma calculator, while it is almost intractable by the other two approaches even running on a powerful local Magma server.

validate the theoretical results, we now consider a practical security evaluation of an IP encoding implementation. In this respect, this case study comes with the (usual) cautionary note that the only thing we show next is that there exist implementations for which (essentially) linear leakages are observed for certain samples. This does not imply that the security order amplification can be observed for full implementations (which, as mentioned in the introduction, is left as an important scope for further research). Yet, it shows that this security order amplification can at least be used to reduce the amount of leaky samples in an implementation and/or their informativeness.

6.1 Target Implementation

Our experiments are performed on a 32 bits ARM Cortex-M4 microcontroller using the Atmel SAM4C-EK evaluation kit running at 100 MHz.[2] We implemented the IP encoding using two shares. We performed the trace acquisition using a Lecroy WaveRunner HRO 66 ZI oscilloscope running at 500 megasamples per second. We monitored the voltage variation using a 4.7 Ω resistor set in the supply circuit of the chip. For each execution and a given value of l_1, we select a random secret x, a random value s_1 and compute the encoding $\mathbf{s} = (s_0 = x + l_1 \cdot s_1, s_1)$. We acquire the leakages by triggering the measurement prior to the successive load of these two shares s_0 and s_1 into the memory.

6.2 Analysing the Leakages

Leakage Detection. Our first experiments aim at analyzing how the device leaks. As a preliminary, we start by identifying the points of interest that corresponds to the manipulation of s_0 and s_1 (in an evaluator-friendly setting where we know these values). Figure 1 shows the result of the correlation between the different time samples with the Hamming weight of s_0 (left) and s_1 (right) using 40,000 traces. Our following analyses only focus on the two samples giving the maximum correlation for both shares. We refer to the time sample corresponding to the manipulation of s_0 (resp. s_1) as t_0 (resp. t_1).

Linear Regression. The theoretical results on the security order amplification of Sect. 4 rely on the assumption that the leakage function is linear. As this assumption is hardware-dependent, we first evaluated the linearity of the leakages produced by our target. For a given time sample, linear regression is perfectly suited for this purpose [34]: it allows estimating how the manipulated data is leaked at the bit level. Denoting by $\mathsf{L} : \mathbb{F}_2^8 \rightarrow \mathbb{R}$ the deterministic part of the actual leakage function, a linear regression of degree q gives the function $\hat{\mathsf{L}}_q$ that approximate L by using bit combinations of degree up to q. As a results, it is a suitable tool to estimate the linearity of the leakages, by just comparing

[2] http://www.atmel.com/tools/SAM4C-EK.aspx.

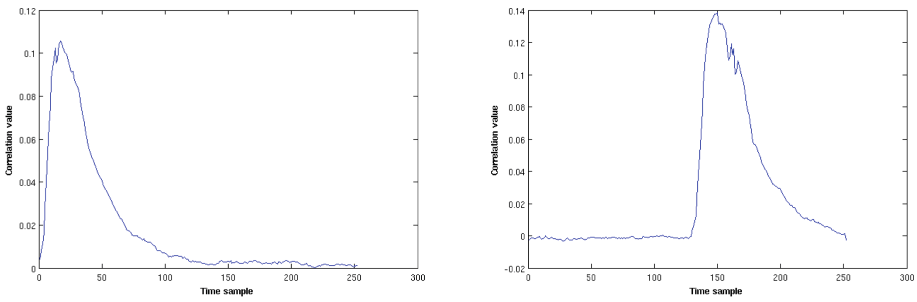


Fig. 1. Detection of points of interest.

regressions of degree 1 and 2. The description of the resulting $\hat{\mathsf{L}}_1$ and $\hat{\mathsf{L}}_2$ approximations are given by Eq. 12. The coefficients α, α_i and $\alpha_{i,j}$ belong to $\mathbb{R}$ and are the results of the linear regression:

$$\hat{\mathsf{L}}_1(x) = \alpha + \sum_{i=0}^{7} \alpha_i \times [x]_2(i)$$
$$\hat{\mathsf{L}}_2(x) = \alpha + \sum_{i=0}^{7} \alpha_i \times [x]_2(i) + \sum_{i=0}^{6} \sum_{j=i+1}^{7} \alpha_{i,j} \times [x]_2(i) \times [x]_2(j) \qquad (12)$$

Using the same traces as for the points of interest detection, we computed the linear regression at t_1 using both a linear ($\hat{\mathsf{L}}_1$) and a quadratic ($\hat{\mathsf{L}}_2$) basis (t_0 gave same results and is thus omitted). The left (resp., right) part of Fig. 2 shows the resulting coefficients for the linear (resp., quadratic) basis. The first value indexed by 0 corresponds to the offset α. The next 8 values indexed from 1 to 8 are the linear coefficient α_i. Finally, the quadratic coefficients $\alpha_{i,j}$ are indexed from 9 to 36 (only in the right figure). We can see that the linear terms are significantly more dominant than the quadratic ones. As a result, it provides some confidence that our target (samples) are good candidates for the linear leakages assumption.

Mutual Information. Evaluating the linearity of the leakage function by only looking at the coefficients of $\hat{\mathsf{L}}_1$ and $\hat{\mathsf{L}}_2$ has two drawbacks. First, it is hard to judge if the models have converged. Secondly, we cannot know if the small values given by the quadratic coefficients are significant or come from estimation errors. In order to get rid of these two problems and push the analysis one step further, we compute the perceived information introduced in [16] arising from $\hat{\mathsf{L}}_1$ and $\hat{\mathsf{L}}_2$. The latter metric essentially captures the amount of information that can be extracted from a model, possibly biased by estimation and assumption errors. (Because of place constraints, we refer to this previous work for the details).

Figure 3 shows this perceived information for the linear model $\hat{\mathsf{L}}_1$ and the quadratic model $\hat{\mathsf{L}}_2$ in function of the number of traces used for the estimation of

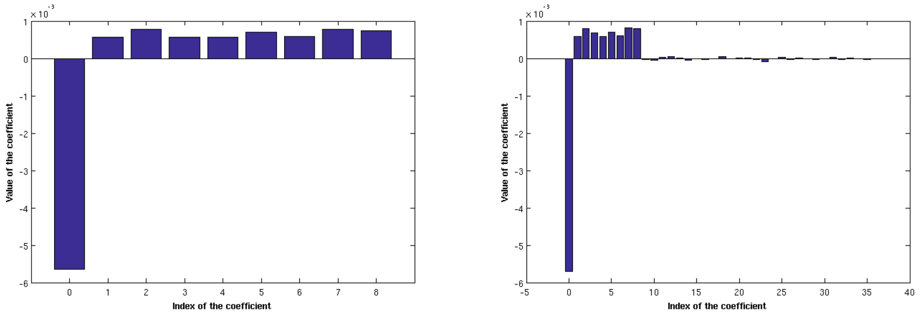


Fig. 2. Linear regression results.

the model. As expected, the quadratic model needs more samples to converge as it is more complex. Interestingly, we can see that both models converge towards approximately the same value. This now formally confirms that the quadratic model does not bring significantly more information than the linear one in our setting. As a consequence, we deduce that the true leakages of our target are close to linear (and therefore that it is a good candidate to benefit from security order amplification).

6.3 Concrete Security Assessment

We now present additional results of concrete security analyses performed on our implementation. For this purpose, and in order to directly evaluate whether the security order of our IP encoding was amplified, we aim at detecting the lowest statistical moment in the leakages that reveals information on the secret. To do so, we apply the ρ-test with K-fold cross-validation as described in [15]. Note that in order to limit the (high) data requirements for this last experiment, we used the trick proposed in [36], Sect. 3.2 and performed a preliminary averaging of our traces (assuming mask knowledge) before trying to detect higher-order statistical dependencies. Namely, we used a 60× averaging for the second-order detections and 100× averaging for the third-order ones.

Correlation-Test. Given a leakage L, the ρ-test allows detecting a mean dependency between L and the secret x. The first step is to estimate a model from a profiling set $\mathcal{L}_{prof}$ of N_{prof} leakage samples on L. This model corresponds to the average leakage on L for each value of the secret x. The next step is to use this model on a test set $\mathcal{L}_{att}$ of N_{test} samples. We compute the correlation r between $\mathcal{L}_{test}$ and our model applied on the secret values used to generate $\mathcal{L}_{test}$. We then compute the normalized Fisher's z-transformation on r:

$$r_z = \frac{\sqrt{N_{test}-3}}{2} \times \ln\left(\frac{1+r}{1-r}\right), \qquad (13)$$

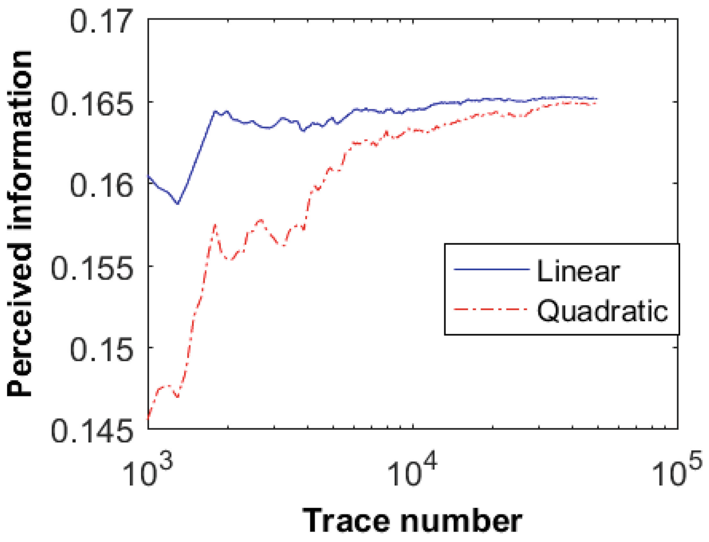


Fig. 3. Perceived information from linear and quadratic leakage models.

where ln denotes the natural logarithm. The obtained value r_z can be approximately interpreted as following a normal distribution with mean 0 and variance 1. As in [15], we set the confidence threshold of r_z that detects the presence of a dependency to 5.

K-fold cross validation. We use a 4-fold cross validation in order to reduce the variability of the ρ-test. That is, we acquire set $\mathcal{L}$ of N leakages that we partition in 4 independent subsets $\mathcal{L}_i, i \in [1, 4]$ of equal size. We then apply the ρ-test 4 times by using a different test set each time (more precisely, iteration i uses $\mathcal{L}_i$ as a test set and $\cup_{j \neq i} \mathcal{L}_j$ as profiling set).

Evaluation results. In our first (reference) experiment, we used l_1 equal to 1, which is equivalent to a Boolean masking encoding. The corresponding DSM representation is such that the dual distance of D is equal to two. As we expect a second-order dependency, we apply the ρ-test on the center product $L = (L_{t_1} - \bar{L}_{t_1}) \cdot (L_{t_2} - \bar{L}_{t_2})$, where $\bar{L}_{t_1}$ (resp. $\bar{L}_{t_2}$) denotes the sample mean of L_{t_1} (resp. L_{t_2}). Figure 4 shows the result of the ρ-test with 4-fold cross validation. The x-coordinate shows the number of average traces being used, and the y coordinate shows the confidence value. The black curves is the line $y = 5$ that shows the confidence threshold. Each of the remaining 4 curves represents one of the cross validation tests. As expected, we quickly detect a second order leakage after roughly 5,000 average traces.

As a second experiment, we set $l_1 = 3$, which is the hexadecimal representation of the polynomial $x + 1$. The corresponding ODSM representation is such that the dual distance of D is equal to three. That is, we expect the lowest statistical moment that gives information on the secret to be equal to three, thus

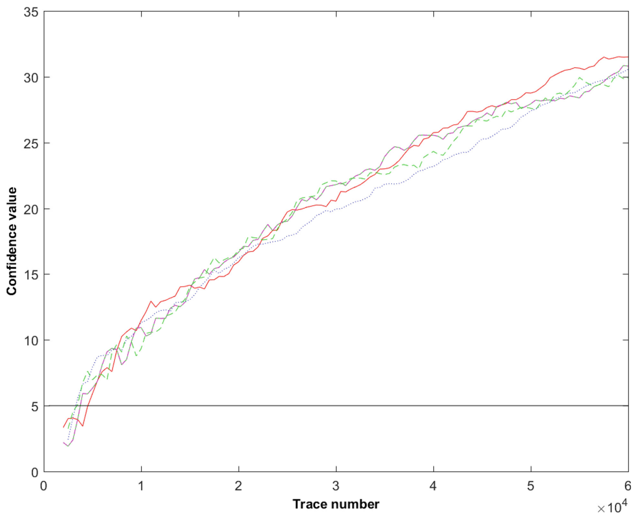


Fig. 4. Results of the ρ-test for IP masking with $l_1 = 1$.

having a security order amplification. We verify this in two steps. First, we apply the ρ-test on the center product as in the previous experiment to detect if any second-order dependency can be seen. Secondly, we apply the ρ-test on a new center product $L = (L_{t_1} - \bar{L}_{t_1}) \cdot (L_{t_2} - \bar{L}_{t_2}) \cdot (L_{t_2} - \bar{L}_{t_2})$ to detect the presence of a third-order dependency. The left (resp., right) part of Fig. 5 shows the results of the ρ-test with 4-fold cross validation for the second-order (resp., third-order) test. Again, the x-coordinate represents the number of average traces, the y coordinates the confidence value and the black curve the confidence threshold. As we can see on the left part of the figure, no second-order leakages are detected with up to 100,000 average traces. However, the right part of the figure shows a third-order dependency around 60,000 average traces. This confirms both the high linearity of the leakages of this chip and the relevance of the theoretical investigations in Sect. 4.

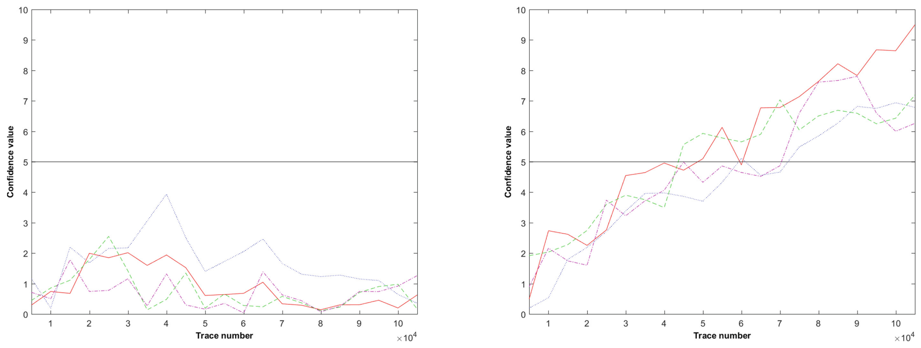


Fig. 5. Results of the ρ-test for IP masking with $l_1 = 3$

Discussion. To conclude, we emphasize that the results of the ρ-test experiment for $l_1 = 3$ were based on $100\times$ averaged traces, leading to a Signal-to-Noise Ratio close to 1 (which is out of the noise levels where masking security proofs apply [14]). So this experiment does not formally prove that no second-order dependency could appear for this noise level (without averaging). In this respect, we recall that this choice was motivated by time constraints (without averaging, we could not detect third-order dependencies either). Besides, and in view of the leakage analysis in Sect. 6.2, we are confident that the security order amplification put forward in this last section does actually correspond to our theoretical expectations with (close enough to) linear leakages.

Acknowledgements. François-Xavier Standaert is a research associate of the Belgian Fund for Scientific Research. This work has been funded in parts by the European Commission through the H2020 project 731591 (acronym REASSURE), the CHIST-ERA project SECODE and the ERC project 724725 (acronym SWORD).

References

1. Balasch, J., Faust, S., Gierlichs, B.: Inner product masking revisited. In: Oswald, E., Fischlin, M. (eds.) [30], pp. 486–510. Springer, Heidelberg (2015)
2. Balasch, J., Faust, S., Gierlichs, B., Verbauwhede, I.: Theory and practice of a leakage resilient masking scheme. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 758–775. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_45
3. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.-X., Strub, P.-Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J.-S., Nielsen, J.B. (eds.) [11], pp. 535–566. Springer, Cham (2017)
4. Bouyukliev, I., Jaffe, D.B., Vavrek, V.: The smallest length of eight-dimensional binary linear codes with prescribed minimum distance. IEEE Trans. Inf. Theor. **46**(4), 1539–1544 (2000)
5. Bringer, J., Carlet, C., Chabanne, H., Guilley, S., Maghrebi, H.: Orthogonal direct sum masking - a smartcard friendly computation paradigm in a code, with builtin protection against side-channel and fault attacks. In: Naccache, D., Sauveron, D. (eds.) WISTP 2014. LNCS, vol. 8501, pp. 40–56. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43826-8_4
6. Carlet, C., Danger, J.-L., Guilley, S., Maghrebi, H.: Leakage squeezing of order two. In: Galbraith, S.D., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 120–139. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34931-7_8
7. Carlet, C., Danger, J.-L., Guilley, S., Maghrebi, H.: Leakage squeezing: optimal implementation and security evaluation. J. Math. Cryptol. **8**(3), 249–295 (2014)
8. Carlet, C., Danger, J.-L., Guilley, S., Maghrebi, H., Prouff, E.: Achieving side-channel high-order correlation immunity with leakage squeezing. J. Cryptogr. Eng. **4**(2), 107–121 (2014)
9. Carlet, C., Guilley, S.: Complementary dual codes for counter-measures to side-channel attacks. Adv. Math. Commun. **10**(1), 131–150 (2016)

10. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
11. Coron, J.-S., Nielsen, J.B. (eds.): EUROCRYPT 2017. LNCS, vol. 10210. Springer, Cham (2017)
12. Coron, J.-S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 410–424. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_21
13. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) [29], pp. 423–440. Springer, Heidelberg (2014)
14. Duc, A., Faust, S., Standaert, F.-X.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In: Oswald, E., Fischlin, M. (eds.) [30], pp. 401–429. Springer, Heidelberg (2015)
15. Durvaux, F., Standaert, F.-X.: From improved leakage detection to the detection of points of interests in leakage traces. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 240–262. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_10
16. Durvaux, F., Standaert, F.-X., Veyrat-Charvillon, N.: How to certify the leakage of a chip? In: Nguyen, P.Q., Oswald, E. (eds.) [29], pp. 459–476. Springer, Heidelberg (2014)
17. Dziembowski, S., Faust, S.: Leakage-resilient cryptography from the inner-product extractor. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 702–721. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_38
18. Fumaroli, G., Martinelli, A., Prouff, E., Rivain, M.: Affine masking against higher-order side channel analysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 262–280. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_18
19. Genelle, L., Prouff, E., Quisquater, M.: Thwarting higher-order side channel analysis with additive and multiplicative maskings. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 240–255. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_16
20. Golić, J.D., Tymen, C.: Multiplicative masking and power analysis of AES. In: Kaliski Jr., B.S., Koç, C.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 198–212. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_16
21. Goubin, L., Martinelli, A.: Protecting AES with Shamir's secret sharing scheme. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 79–94. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_6
22. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J.S., Nielsen, J. (eds.) [11], pp. 567–597. Springer, Cham (2017)
23. Grassl, M.: Tables of linear codes and quantum codes (2015). http://www.codetables.de/. Accessed 25 Apr 2017
24. Griesmer, J.H.: A bound for error-correcting codes. IBM J. Res. Dev. **4**(5), 532–542 (1960)
25. Grosso, V., Prouff, E., Standaert, F.-X.: Efficient masked S-boxes processing – a step forward –. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 2014. LNCS, vol. 8469, pp. 251–266. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06734-6_16

26. Grosso, V., Standaert, F.-X., Prouff, E.: Low entropy masking schemes, revisited. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 33–43. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_3
27. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
28. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and its Applications. Advanced Book Program/World Science Division. Addison-Wesley Publishing Company, Boston (1983)
29. Nguyen, P.Q., Oswald, E. (eds.): EUROCRYPT 2014. LNCS, vol. 8441. Springer, Heidelberg (2014)
30. Oswald, E., Fischlin, M. (eds.): EUROCRYPT 2015. LNCS, vol. 9056. Springer, Heidelberg (2015)
31. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_9
32. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_28
33. Roche, T., Prouff, E.: Higher-order glitch free implementation of the AES using secure multi-party computation protocols - extended version. J. Cryptogr. Eng. **2**(2), 111–127 (2012)
34. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
35. Schneider, T., Moradi, A.: Leakage assessment methodology - extended version. J. Cryptogr. Eng. **6**(2), 85–99 (2016)
36. Standaert, F.-X.: How (not) to use welch's t-test in side-channel security evaluations. IACR Cryptology ePrint Archive, 2017, p. 138 (2017)
37. van Tilborg Henk, C.A.: The smallest length of binary 7-dimensional linear codes with prescribed minimum distance. Discrete Math. **33**(2), 197–207 (1981)
38. Wang, W., Standaert, F.-X., Yu, Y., Pu, S., Liu, J., Guo, Z., Gu, D.: Inner product masking for bitslice ciphers and security order amplification for linear leakages. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 174–191. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_11

May the Force Be with You: Force-Based Relay Attack Detection

Iakovos Gurulian[1(✉)], Gerhard P. Hancke[2], Konstantinos Markantonakis[1], and Raja Naeem Akram[1]

[1] Information Security Group Smart Card Centre, Royal Holloway, University of London, Egham, UK
{Iakovos.Gurulian.2014,k.markantonakis,r.n.akram}@rhul.ac.uk

[2] Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
gp.hancke@cityu.edu.hk

Abstract. Relay attacks pose a significant threat against communicating devices that are required to operate within a short-distance from each other and a restricted time frame. In the field of smart cards, distance bounding protocols have been proposed as an effective countermeasure, whereas, in the field of smartphones, many proposals suggest the use of (natural) ambient sensing as an effective alternative. However, empirical evaluation of the proposals carried out in existing literature has reported negative results in using natural ambient sensing in distance- and time-restricted scenarios, like EMV contactless payments that require the proximity to be less than 3 cm and the transaction duration to be under 500 ms. In this paper, we propose a novel approach for Proximity and Relay Attack Detection (PRAD), using bidirectional sensing and comparing button presses and releases behaviour (duration of press and gap between presses and releases), performed by a genuine user during the transaction. We implemented a test-bed environment to collect training and analysis data from a set of users, for both the genuine and attacker-involved transactions. Analysis of the collection-data indicates a high effectiveness of the proposed solution, as it was successful in distinguishing between proximity and relay-attack transactions, using thresholds set after analysis of genuine training transaction data. Furthermore, perfect classification of genuine and relay-attack transactions was achieved by using well-known machine learning classifiers.

Keywords: Mobile payments · Relay attacks · Contactless Experimental analysis

1 Introduction

Relay attacks [6,7,33] are passive man-in-the-middle attacks, aiming to extend the physical distance of devices involved in a transaction beyond their operating environment. Contactless smart cards [7,12,13,15,17], as well as smartphones

[6,19,22,23] are susceptible to relay attacks. Using such attacks, an attacker can gain unauthorised access to services and facilities that a genuine user is eligible for, like payments and access to buildings.

In the field of smart cards, distance-bounding protocols have been proposed as an effective countermeasure [14,27]. However, distance bounding protocols may not be applicable in the field of smartphones due to unpredictable behaviour related to their multi-tasking architecture and the multitude of hardware components [30].

In recent years, a number of proposals suggest the use of ambient sensing as an alternative Proximity and Relay Attack Detection (PRAD) mechanism against the off-the-shelf attacker [11,18,25,29,31,32]. Such proposals rely on the collection of data over a period of time from both transaction devices, using ambient sensors, and subsequently comparing the collected data for similarity.

Specific scenarios susceptible to relay attacks, like EMV contactless payments and transport ticketing, have industry-imposed time limits regarding the completion of a contactless transaction. In the case of EMV, the limit is 500 ms [2–4], and in the case of transport ticketing, typically between 300 and 500 ms [1]. Recent evaluations of previously proposed PRAD techniques have yielded limited evidence that ambient sensing can effectively counter relay attacks in contactless transactions under 500ms [10,24]. The generation of an artificial ambient environment (AAE) has been proposed instead, and evaluated using infrared light, with promising results [9]. However, attacks against ambient sensing (in sound, WiFi, Bluetooth, temperature, humidity, gas, and altitude) have been demonstrated in the presence of an attacker with context manipulating capabilities [26]. Even though such attacks might not be able to cause a false positive in the case of using infrared light as an AAE actuator, denial of service attacks might be achieved.

In this paper we propose a novel approach towards PRAD based on sensing button presses on the user's smartphone by both transaction devices (transaction terminal and transaction 'user' smartphone) simultaneously. During the time of the transaction, the user is requested to press four buttons randomly picked by a smartphone application. The input button sequence, as well as timings of button presses and between consequent button presses (referred to as *'releases'*) are captured by both transaction devices and used as features for similarity comparison. Empirical evaluation demonstrated high success rate of the method as a PRAD mechanism. Using threshold-based evaluation, all the attack attempts were detected. Perfect classification was achieved by using the Support Vector Machine classifier. Near-perfect classification (up to 99.8%) was achieved by other well-known machine learning classifiers.

The main contributions of this paper are:

- Force Sensing Relay Attack Detection: A novel approach for PRAD based on force sensing by the transaction devices (Sect. 4).
- Evaluation Frameworks: The design of two evaluation frameworks, for evaluating the proposed solution as a proximity detection mechanism (Sect. 6),

and as a relay-attack detection mechanism, by subjecting it against a set of volunteers attempting to attack the scheme (Sect. 7).
- Two-Fold Evaluation: Threshold- and machine learning-based evaluation of the proposed system, in the presence of a relay-attacker.

2 Relay Attacks

A variety of applications are affected by relay attacks, like Near Field Communication (NFC) based contactless transactions. During a relay attack, the goal of the attacker is to relay communication messages between two devices that are located beyond their designated operational environment, without being detected.

The relay of the communication messages is performed using some relay equipment that the attacker possesses. For example, for the case of an NFC contactless payment scenario (Fig. 1), an attacker can present to a genuine user a masquerading (malicious) payment terminal. At a distant location, the attacker should present a masqueraded (malicious) payment instrument to a genuine payment terminal. When the user attempts to perform a transaction, the communication messages of the payment transaction will be relayed between the attacker's relay equipment. If the attack is successful, an unauthorised transaction between two genuine parties will be performed.

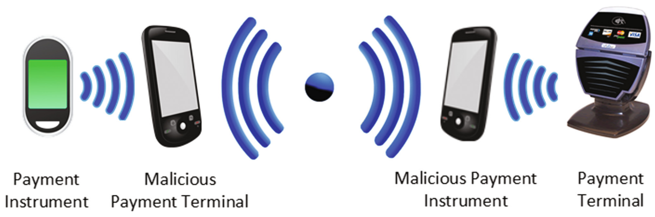


Fig. 1. Overview of a relay attack

Relay attacks against mobile devices have been demonstrated [6,22,23]. In order to detect the existence of a relay attack, evidence regarding the co-presence of the genuine transaction devices should be established. As already mentioned, in the field of smartphones, establishment of proximity evidence by the assistance of ambient sensing has demonstrated positive results. This technique requires the two transaction devices to capture environmental data, using some ambient sensor, for some predefined period of time.

An alternative approach, by generating an artificial ambient environment (AAE) using the peripherals of the communicating devices has also demonstrated positive results. This approach has demonstrated positive results when using infrared light as an AAE actuator in transactions with industry imposed time limits of up to 500 ms, like EMV contactless payments and transport ticketing.

For both techniques, the data from the two devices is then compared for similarity, based on which a decision is made regarding their co-presence. The comparison process can be performed either by one of the communicating devices, or by a trusted third party (TTP).

3 Related Work

In this section, we identify and summarise key pieces of related work that have suggested using natural ambient sensing as a PRAD mechanism.

Ma et al. [18] proposed the use of GPS (Global Positioning System) as a means of co-location detection. A time frame of 10 seconds was used for data collection, and values were recorded every second. High success rate was reported by the authors for proximity detection.

Halevi et al. [11] proposed the use of ambient light and sound. Values were captured for 30 and two seconds, respectively. The authors used various comparison algorithms, and high success rate was reported.

Varshavsky et al. [32] compared the WiFi networks, along with the signal strengths, that the devices were able to detect. The main objective of this work was device pairing, and positive results were reported.

Urien et al. [31] combined ambient temperature and an elliptic-curve based RFID and/or NFC authentication protocol. No performance results were presented by the authors, as there was no practical implementation.

Mehrnezhad et al. [20] recorded values using the accelerometer of the devices involved in a payment transaction in order to detect device co-location. A double tap was required in their proposal. According to the authors, the transaction time lasted between 0.6 and 1.5 s, and a high success rate was observed.

Truong et al. [29] assessed a variety of sensors for proximity detection. The recording time frame was between 10 and 120 s, and positive results were reported.

Shrestha et al. [25] used a Sensordrone and recorded multiple sensors. The precise sample duration is not provided in this work, however the authors state that recordings lasted for a few seconds.

Jin et al. [16] used the magnetometer in order to pair devices that are in proximity. An average of 4.5 s is required for the pairing to succeed. The authors only focus on proximity detection, and do not claim that this method can be used as an effective relay attack detection mechanism.

In [10,24], the effectiveness of recording the natural ambient environment in short transactions (up to 500 ms) was empirically evaluated, with different results from the existing literature. Comparison algorithms used in previous works, as well as machine learning techniques, produced very high false negative results.

Gurulian [9] proposed a relay attack detection framework by using artificial ambient environments (AAE). Infrared light was evaluated as an AAE actuator. Relay attacks were successfully detected, while the false rejection rate was approximately 2%.

4 Force-Sensing PRAD

In this section we present the theoretical foundation of the proposed framework, and the threat model.

4.1 PRAD Framework

During the course of a transaction, the user is called to position the Transaction Instrument (TI) on an extension force-sensitive panel of the Transaction Terminal (TT) in order to complete the PRAD procedure. A smartphone application (running on TI) presents to the user an interface with buttons that the user is called to tap. Alternative interface design approaches can be followed. For example, the application might present buttons with numbers and ask the user to input a provided 4 digit sequence. Alternatively, the application can present a button at a time that the user has to press in order for the next button to appear, until all the buttons of the sequence have been pressed. It should be stressed that even though the aforementioned method was used during the evaluation of the proposed framework (Sect. 5), it is not restrictive, as alternative approaches can be used instead. Figure 2 presents the basic architecture of the framework, when using the later application design method. Also, the use of a Personal Identification Number (PIN) and other forms of user identification codes may pose a threat, as an attacker could potentially capture them if the user attempts to perform a transaction with a malicious terminal.

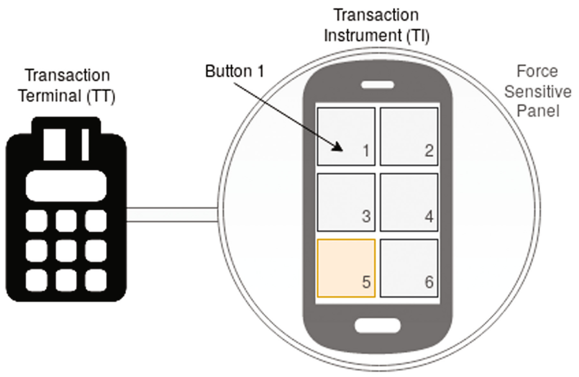


Fig. 2. The basic framework architecture

Each button is assigned an ID based on the location of the button (e.g. the top left button is assigned the ID *'1'*). When the user taps on a button, both transaction devices record the ID of the button that was pressed, and the duration of the button press. The duration between subsequent button presses is also recorded by the two devices. Further features can also potentially be used, like the amount of pressure applied, but were not considered in this work due to limitations inflicted by the architecture of the majority of modern smartphones (further discussion in Sect. 8).

In order for device TI to recognise the buttons and timings, simple API calls are required. Device TT can recognise the buttons and timings based on the coordinates and duration of the detected pressure on the force-sensitive panel.

The assumption is that the two devices are going to record approximately the same values, and that an attacker has a low probability of accurately replicating the movement of a genuine user. The accuracy of the system should be at the millisecond (ms) level. The captured data of the two devices should provide sufficient proximity evidence when compared against each other, while data captured when a relay attack is taking place should be detectable at a high rate, in accordance to the requirements of the deployment scenario.

The proximity verification process can take place during the course of the transaction by one of the communicating devices, or afterwards, by a Trusted Third Party (TTP). The captured data should be communicated between the devices or to the TTP in an encrypted and authenticated form, but discussion regarding the trusted comparison party's (i.e. the TTP or one of the transaction devices) architecture is out of the scope of this work. As the main focus of this work is to examine the effectiveness of the proposed solution as a PRAD mechanism, we limit the discussion and do not investigate the integration with existing applications.

4.2 Threat Model

In this paper, the attacker is of opportunistic nature and requires no prior interaction or knowledge of neither TT nor TI. However, the potential implications if TT or TI are compromised are out of the scope of this paper, since a relay attack may not be necessary to achieve the same goals under these circumstances. We focus primarily on the issue of genuine devices requiring proximity assurance in order to conduct a legitimate transaction.

The attacker only has access to off-the-shelf relay equipment. Usually transaction limits apply on mobile transactions, for example a £30 limit on digital payment transactions in the UK [5]. Moreover, an attacker that might be using more advanced techniques, like a robotic arm that replicates the movement of a genuine user with accuracy, is likely going to be detected by the genuine device operators. Therefore, a very powerful attacker might not be a major concern.

5 Test-Bed Architecture

We designed and built a test-bed in order to evaluate the proposed solution. Android devices were used to represent TI, and an Arduino-based prototype was developed and used as TT.

An Android application was built and installed on three devices; a light and small Android smartphone, a heavier and larger Android smartphone, and an

Android tablet device. As the first, a Samsung Galaxy S5 Mini (SM-G800F) was used. It features a 4.5-inch display, and weighs 120 g. A Samsung Galaxy S4 (GT-I9500) device was used as the second device. It features a 5-inch display and weighs 130 g. Finally, a Nexus 9 was used as the tablet. Its display is 8.9-inches and weighs 425 g[1].

The Android application, running on TI, displayed six equal size zones on the touch screen (as in Fig. 2). In this paper, we refer to these zones as 'buttons'. Initially, a random button on the screen was highlighted using a colour. This indicated to the user to press the randomly highlighted button on the screen. Releasing a button would trigger it to disappear, and the application would randomly pick and highlight a second button. A total of four buttons were randomly highlighted on each transaction. A random shared ID assigned to each transaction, along with the button sequence IDs, and the timings of button presses and interval duration between subsequent button presses were appended to a local CSV file that was later extracted from the device for data comparison.

An Arduino Due was used for the TT prototyping. Four Force Sensitive Resistors (FSRs) were connected to the board's analog inputs (Fig. 3a). FSRs can be used to detect pressure or weight. The basic working principle is that the resistive value of the sensor alternates depending on the amount of force that is being applied on the sensor. Silicone buffer pads[2] were manually attached to the centre the sensors, as the force resistive area of the FSRs was not reachable by the surface of the smart devices otherwise. Even though FSRs were used in this prototype, other sensors might potentially also be used alternatively, for example capacitive touch sensors.

Initially, when TI was placed on TT and the process started, a calibration phase had to be conducted, in order to cancel the weight of the device. For one

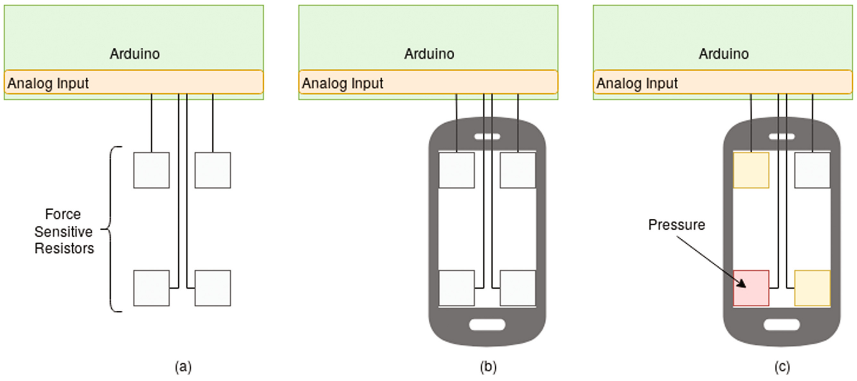


Fig. 3. Detection of presses by TT

[1] All device characteristics found at http://www.gsmarena.com/.
[2] Example of buffer pads: https://www.amazon.co.uk/gp/product/B00P11D4VK/ref=s9u_simh_gw_i2.

second after the process initiation, TT would capture values from all four sensors and the maximum recorded value of each sensor would be used as a reference point. A LED would indicate the completion of the calibration phase.

After the calibration phase, the user was called to input the sequence on TI. While a button was being pressed, the calibrated TT sensor(s) closest to that button were recording the highest values (Fig. 3). The indication used for detecting that a button was being pressed was that some of the sensors were recording values above their calibration point. The time during which higher values than the calibration point were being recorded by some of the FSRs was considered to be the pressing time. Similarly, the time between subsequent button presses was the time during which none of the sensors were recording higher values than the calibration threshold.

After a button was released, TT would estimate the button that was being pressed. To achieve this, each sensor was assigned a value, which was the average of all the values captured by that particular sensor during the button press period. Six virtual buttons had to be detected, corresponding to the buttons presented by TI. For the detection of the pressed button, the values that were assigned to each of the four sensors were used.

Algorithm 1 lists pseudocode for detecting the pressed buttons when the button ID numbers of device TI are per Fig. 2, and the sensor ID numbers of device TT as per Fig. 3. The input *sensor1 – sensor4* is the value assigned on each of the sensors. Initially, the side (left or right) of the screen on which the press was on was determined by comparing the sum of the sensor values returned of each side. Once the side was determined, the proportion of the force applied on the top sensor was calculated. The proportion was divided in three equal

Algorithm 1. Detection of pressed button

```
Input  : int sensor1, int sensor2, int sensor3, int sensor4
Output: int pressedButtonID
1  leftSide ← sensor1 + sensor2;
2  rightSide ← sensor3 + sensor4;
3  if leftSide > rightSide then
4      force ← sensor1 / leftSide;
5      if force ≥ 0 and force < 0.33 then
6          return 5
7      else if force ≥ 0.33 and force < 0.66 then
8          return 3
9      return 1
10 end
11 force ← sensor3 / rightSide;
12 if force ≥ 0 and force < 0.33 then
13     return 6
14 else if force ≥ 0.33 and force < 0.66 then
15     return 4
16 return 2
```

parts. If the proportion was between 0.66 and 1, the top virtual button ID was returned. Similarly, values between 0.33 and 0.66 corresponded to the middle virtual button, and between 0 and 0.33 to the bottom one.

When four button presses were detected by the prototype, it would return the pressed button sequence and the timings to a computer through Arduino's serial port. An application written in Python would request from the experiment operator to manually input the ID that was assigned to the particular transaction by TI. The returned values and the transaction ID would be appended to a CSV file with the same format as the one on TI's side.

Two experimental frameworks were developed, for proximity and relay attack evaluation. At the end of each phase, the stored data would be extracted form each of the devices and moved to a computer for the evaluation of the results.

6 Proximity Detection Framework

In order to evaluate the performance of the proposed solution in a proximity detection scenario, 100 transactions were performed with each of the three devices listed in Sect. 5. A random sequence was generated and presented by TI on each transaction. The FSRs were aligned in a set-up for the smallest device (SM-G800F), and this set-up was maintained throughout the experimental phase, regardless of the device being used. After the completion of the first phase, using each of the three TI devices, the collected data in CSV format was extracted from both transaction devices (the smartphone and the laptop on which the Arduino was connected).

The aim of the framework was to assess whether sufficient information for establishing proximity evidence is collected during the process. The results were used to set acceptable upper and lower bounds for presses and releases.

6.1 Evaluation Methodology

A Python application was developed for the data analysis process, which would compare the input sequence recorded by both devices, and calculate the difference of individual corresponding features (button press and release timings). The minimum and maximum press and release differences were detected for each of the devices, and all 300 measurements combined. The minimum and maximum press and release differences were used as limits for distinguishing genuine from relay attack transactions in the relay attack detection framework.

6.2 Results and Discussion

The results of the proximity detection framework are listed in Table 1. They refer to the time difference in milliseconds between the timings recorded by TT and TI. For example, the difference of the first press for a single transaction is calculated as:

$$Diff_{Press1} = TT_{Press1} - TI_{Press1}$$

Table 1. Proximity detection results – in ms (negative results indicate that TI's measurement durations were larger than TT's)

	Minimum		Maximum		Span		Average	
	Press	Release	Press	Release	Press	Release	Press	Release
SGS5 mini	−46	11	−11	48	35	37	−29.09	28.71
SGS 4	−28	−8	9	30	37	38	−5.32	5.26
Nexus 9	−18	−8	12	23	30	31	−2.16	2.83
Total	−46	−8	12	48	58	56	−12.19	12.27

The maximum, minimum, and the average calculated values are listed, as well as the span between the maximum and minimum observed values. The analysis has been performed for each of the three devices, as well as for the combination of all the recordings of the three (referred to as 'Total'). Negative timings denote that the measurement of a press or release by TI was longer than that of TT.

Differences among the devices were observed, likely related to their weight. However, the measured time span was approximately the same for all three devices. Therefore, a real-world deployment is recommended to take into account the device model, in order to minimise the attack window approximately by half.

High accuracy was also ascertained in the detection of the input pattern by device TT. Prior to the initiation of the experimental phase, the pattern detection of the smaller devices was very high. Failures occurred in a few occasions where the buttons on device TI were pressed very close to neighbouring buttons. Slightly reducing the size of the buttons effectively restricted this issue.

The detection of button presses on the tablet required longer presses than on the smartphones, which had no special input requirements. Even though perfect detection was recorded when longer button presses were applied, the use of tablets is not recommended without readjustment of the underlying FSRs.

7 Relay Attack Detection Framework

We subsequently evaluated the effectiveness of the solution as an anti-relay mechanism. We gathered 10 volunteers who attempted to attack the system in two phases, explained below. A separate Android application was developed for the purposes of the experiments, based on the application used in the proximity evaluation framework. It was presenting predefined sequences instead of random ones. Device TT's architecture did not alter between the two frameworks.

During the first attack phase, a set of videos of a person inputting a sequence (genuine user) were presented to the volunteers. The sequence timings entered by the genuine user on TI were stored in a CSV file as per Sect. 6 for later comparison against relay attack data. The videos were played on a large screen, and the movement of the genuine user was evident. The camera used to record the videos was placed on the left of the device, and at a 45 degree angle, in order to make the presentation of the three dimensions more clear. Prior to the initiation of the experiments, the volunteers were asked to choose the rotation and flip of the video that they preferred. A set of 10 different patterns were presented to each volunteer, who was asked to attempt to replicate/mimic the movement of the genuine user with accuracy while the video was playing, or afterwards. The same pattern was presented on both the video and the device provided to the volunteers. Also, the same device (SM-G800F) was used on the video and by the volunteers. Data from both devices was stored in separate CSV files, separately for each volunteer.

All volunteers were university students and staff who had a good understanding of security and relay attacks. Prior to the experiments, their goal and the principle on which the anti-relay mechanism was based were thoroughly explained to them. Moreover, four of the volunteers had background in playing some musical instrument (guitar, piano, or both), ranging from medium to advanced level.

During the second phase, the volunteers were asked to attempt and attack the exact same video 10 times. Meaning, a genuine user entered a sequence that was recorded, then the attacker was given 10 tries to train to replicate the sequence as close to the genuine user as possible. This tested the possibility of whether an attacker who watches the same pattern being entered a number of times, his/her potential of replication would increase. The same set-up as in the previous phase was used. In both phases the attacker was very powerful, as the input sequence was known, and there was a clear view of the genuine user's movements.

7.1 Evaluation Methodology

Measurements captured from the TI that the genuine user was using on the video were compared against measurements from device TT used by the volunteers. In a relay attack, these two devices would be the genuine devices, the other two, the devices operated by the attacker.

In order to evaluate the performance of the volunteers, two methods were used; threshold- and machine learning-based analyses. Initially, we set acceptable minimum and maximum thresholds for presses and releases, based on the results of the proximity evaluation framework (Sect. 6.2). The recorded presses and releases from the two devices were sequentially compared against each other. If the difference between a press or a release captured by TT and TI was within the bounds, it was considered to be acceptable. Otherwise a relay attack was detected. The point at which the inconsistency appeared was the detection point. For example, if an attacker performed the first press and the first release correctly, but the second press was out of bounds, the second press would be considered as the detection point. This part of the evaluation was conducted in two phases. We first subjected the relay attack data against the thresholds set when by all the three devices, and then against device specific thresholds (SM-G800F).

Weka [8] was used to apply a suite of well-known classifiers. The classifiers were trained on a set of feature vectors with corresponding binary labels (genuine or relayed transaction), which were collected beforehand. The trained model was used to classify subsequent transaction data streams as genuine or relayed. We tested the Random Forest, Naïve Bayes, Logistic Regression, Decision Tree (grown using the C4.5 algorithm), and Support Vector Machine with the RBF[3] kernel (the SVM w/RBF hyper-parameters, C and γ, were established using standard exhaustive grid search) classifiers. The threshold was based on the probability estimate output by the learned classification model, i.e. the estimated probability that a transaction is genuine. As genuine transactions were considered transactions collected during the proximity evaluation phase (Sect. 6).

7.2 Results and Discussion

A total of 200 relay transactions were evaluated. The results of both threshold- and machine learning-based analyses are presented in this section.

Evaluation 1: Threshold-Based. The results of the threshold-based analysis are listed in Table 2. *'General Threshold'* refers to the threshold set by combining the proximity results of all three tested devices, while *'Device Specific Threshold'* to the threshold set by SM-G800F, as it was the device used for the relay attack detection evaluation. *'Detected'* refers to the percentage of relay attempts detected at a particular press or release, because the attackers failed to accurately replicate the movement of the genuine user at that point. *'Correct'* refers to the percentage of times that a single press or release was successfully replicated by the attacker. Finally, the first phase refers to the user trying to attack a different video on each try, and the second phase, the attacker trying to attack the same video 10 times.

Table 2. Threshold-based relay attack detection results

	Press 1	**Release 1**	**Press 2**	**Release 2**	**Press 3**	**Release 3**	**Press 4**	**False Accept**
				General Threshold – Phase 1				
Detected	73	24	0	2	1	0	0	0
Correct	27	10	37	10	24	5	31	—
				Device Specific Threshold – Phase 1				
Detected	84	16	0	0	0	0	0	0
Correct	16	7	25	9	9	4	23	—
				General Threshold – Phase 2				
Detected	57	37	5	1	0	0	0	0
Correct	43	11	29	24	26	9	20	—
				Device Specific Threshold – Phase 2				
Detected	65	33	2	0	0	0	0	0
Correct	35	6	21	22	20	6	9	—

None of the volunteers attempted to successfully attack the system, using threshold-based analysis. Moreover, the volunteers with background in playing

[3] RBF: Radial Basis Function.

a musical instrument did not present improvements over the rest of the user. However the sample was limited, so further investigation is required.

The best attempt is presented in Fig. 4. Device TI′ in the figure represents the measurement recorder by the transaction instrument that the volunteer was using. It illustrates the corresponding proximity transaction, for which all presses and releases were within bounds. The relay attack was detected on the third press, using the threshold set by the combination of the three devices. Using device specific threshold, the attack was successfully detected on the first release.

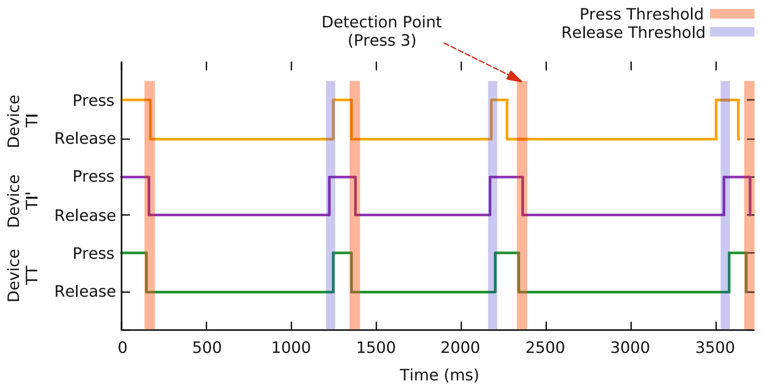


Fig. 4. Graphical representation of the best attack attempt (TT – TI) versus the corresponding genuine transaction (TT – TI′)

The majority of the transactions were detected on the first press and release, using both general, and device specific thresholds, even when the volunteers attempted to attack the same video multiple times (second phase). During the second phase, a small incline was observed in the user performance as the same video was being attacked multiple times, most evident in the performance of releases. Figure 5 depicts that incline, along with the average, maximum, and minimum performance of each round's presses and releases.

Evaluation 2: Machine Learning-Based. The results of the machine learning-based analysis, obtained by repeating stratified 10-fold cross-validation, are presented in Table 3. A training set of 300 genuine and 200 relay attack transactions was used. The default settings of each algorithm were used on Weka. The metrics listed are the classification accuracy *(Accuracy)*, the Area Under the Receiver Operating Characteristic (ROC) Curve *(AUC)*, the *F1-score*, and the Equal Error Rate *(EER)*.

Perfect classification was achieved by using the Support Vector Machine classifier. Near perfect classification ($> 98\%$) was achieved by the Random Forest, Naïve Bayes, and Decision Tree classifiers, with best performance observed by the first three ($> 99.5\%$ accuracy). The Random Forest algorithm failed to accurately classify two relay transactions, the Naïve Bayes one, and the Decision Tree six.

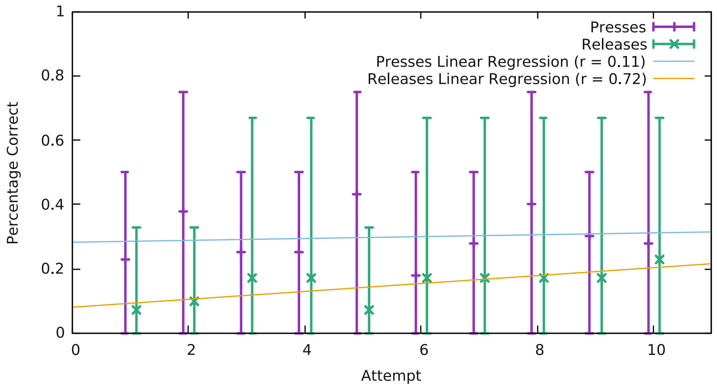


Fig. 5. Performance variation of each of the 10 attempts in the second phase

Table 3. Machine learning classification results obtained by repeating 10-fold cross-validation 10 times

	Random forest	Naïve bayes	Logistic regression	Decision tree	Support vector machine
Accuracy (%)	99.62	99.80	86.58	98.78	100.00
AUC	0.9999	0.9996	0.8163	0.9873	1.0
F1-score	0.9969	0.9984	0.90	0.9901	1.0
EER	0.0022	0.0047	0.1993	0.104	0.0

8 Discussion and Outcome

The analysis of the experimental data indicated that the effectiveness of the proposed solution as a PRAD mechanism was high. However, some usability concerns might arise, especially for visually impaired users, and users with motor difficulties. Compared to the majority of previous works, additional steps are required by the user, including the correct placement of device TI on TT.

Since the positioning of device TI on TT is important for the later to accurately detect the ID of the pressed button, device-specific positioning lines were presented on the TI's display prior to the process initiation. This assisted in more equally dividing the mass across the four FSRs. The model of the device could be determined through API calls, which was used to display the correct positioning. It should be stressed that the positioning precision had to be performed with only some degree of accuracy. The detection was found to be very accurate even without perfect placement of the devices. Figure 6 depicts the guidelines and the pattern input interface on a SM-G800F device.

Failure to click on the correct button (mistapping) will also lead to inconsistencies between the captured data of the two devices, so the process will have to be restarted. Moreover, a vibration of the phone during the process will cause

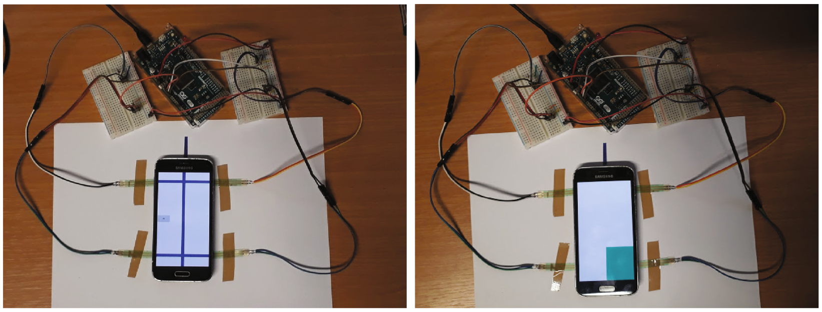

(a) Device Calibration (b) Pattern Entry

Fig. 6. The evaluation test-bed

inconsistencies, so they should temporarily be disabled, until the completion of the process. Finally, the performance of the solution might also degrade if phone cases are used on TI, or the device's camera is located above the point where an FSR should touch. However, significant advantages over previous works exist, so depending on the deployment scenario this technique might be preferable.

The relay attack detection rate was empirically found to be higher than previously proposed solutions [11,21,25,28]. The detection rate can potentially improve even further by considering more features, like the amount of pressure detected by the two devices. Even though this is possible on TT, at the moment the majority of smartphones are not capable of accurately measuring the amount of pressure applied on their screen. Some Android devices estimate the amount of pressure through the number of pixels being covered on the screen by the user's finger. This technique is not very accurate, and it is also not available on all devices. The feature is not supported by the GT-I9500. On the other two devices, the accuracy was highly dependant on the finger orientation rather than the amount of pressure being applied on the screen. Moreover, inconsistencies were observed among the values recorded by the two devices. We concluded that the technology was not mature enough to provide quality data.

Moreover, no additional or non-standard hardware on the TI side is required, like in many of the previously proposed solutions, as in [9,25]. Many of the previously proposed solutions might also be vulnerable in the presence of an attacker with context manipulating capabilities [26]. Since this solution is not dependant upon the surrounding environment, such attacks do not apply, unless the attacker physically tampers with the devices.

9 Conclusion and Future Directions

Communicating devices that require to operate within proximity are vulnerable to relay attacks. Traditional distance bounding protocols that aim to counter such attacks might not be applicable in the field of smartphones. Alternative approaches against the off-the-shelf attacker have been proposed, mostly based on sensing of the ambient environment. However, these might not be suitable or secure under certain scenarios, like in the presence of an attacker with context manipulating capabilities. In this work we presented a novel approach for Proximity and Relay Attack Detection (PRAD) by using bidirectional sensing and comparing of subsequent button presses and releases by the transaction devices.

A test-bed was designed and built for the evaluation of the proposed solution as a PRAD mechanism. Initially, the effectiveness in proximity detection was examined. Afterwards, the test-bed was subjected against a set of volunteers who attempted to attack the system. All the attack attempts were successfully detected through threshold-based analysis. Moreover, perfect classification was achieved by using the Support Vector Machine classifier. Classification accuracy of up to 99.8% was achieved by other well-known machine learning classifiers.

As part of our ongoing investigation, we are planning to conduct a more extensive user study. We are also planning to examine the use of additional features that would further minimise the potential of an attacker to perform a relay attack, like pressure intensity.

References

1. Transit and Contactless Open Payments: An Emerging Approach for Fare Collection. White paper, Smart Card Alliance Transportation Council, November 2011
2. How to Optimize the Consumer Contactless Experience? The Perfect Tap. Technical report. MasterCard (2014)
3. EMV Contactless Specifications for Payment Systems: Book A - Architecture and General Requirements. Spec V2.6. EMVCo, LLC, April 2016
4. Transactions Acceptance Device Guide (TADG). Specification Version 3.1. VISA, November 2016
5. Digital Payments Solutions Industry Considerations. Online report. The UK Cards Association, June 2017. http://www.theukcardsassociation.org.uk/wm_documents/Digital%20Wallets%20-%20Industry%20Considerations%20Outline.pdf
6. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical NFC peer-to-peer relay attack using mobile phones. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 35–49. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16822-2_4
7. Francis, L., Hancke, G.P., Mayes, K., Markantonakis, K.: Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. IACR Cryptology Archive 2011, p. 618 (2011)
8. Frank, E., Hall, M.A., Witten, I.H.: The WEKA workbench. In: Data Mining: Practical Machine Learning Tools and Techniques. 4 edn. Morgan Kaufmann, Burlington (2016)
9. Gurulian, I., Akram, R.N., Markantonakis, K., Mayes, K.: Preventing relay attacks in mobile transactions using infrared light. In: Proceedings of the Symposium on Applied Computing SAC 2017, pp. 1724–1731. ACM, New York (2017)
10. Gurulian, I., Shepherd, C., Frank, E., Markantonakis, K., Akram, R., Mayes, K.: On the effectiveness of ambient sensing for nfc-based proximity detection by applying relay attack data. In: The 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2017. IEEE, August 2017

11. Halevi, T., Ma, D., Saxena, N., Xiang, T.: Secure proximity detection for NFC devices based on ambient sensor data. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 379–396. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_22
12. Hancke, G.P.: Distance-bounding for RFID: Effectiveness of 'terrorist fraud' in the presence of bit errors. In: 2012 IEEE International Conference on RFID-Technologies and Applications (RFID-TA), pp. 91–96, November 2012
13. Hancke, G.P.: Practical attacks on proximity identification systems (short paper). In: IEEE Symposium on Security and Privacy, pp. 328–333. IEEE Computer Society (2006). http://dblp.uni-trier.de/db/conf/sp/sp2006.html#Hancke06
14. Hancke, G.P., Kuhn, M.G.: An RFID distance bounding protocol. In: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SECURECOMM 2005, pp. 67–73. IEEE Computer Society, Washington DC (2005)
15. Hancke, G., Mayes, K., Markantonakis, K.: Confidence in smart token proximity: relay attacks revisited. Comput. Secur. **28**(7), 615–627 (2009). http://www.sciencedirect.com/science/article/pii/S0167404809000595
16. Jin, R., Shi, L., Zeng, K., Pande, A., Mohapatra, P.: MagPairing: pairing smartphones in close proximity using magnetometers. IEEE Trans. Inf. Forensics Secur. **11**(6), 1306–1320 (2016)
17. Kfir, Z., Wool, A.: Picking virtual pockets using relay attacks on contactless smartcard systems. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, pp. 47–58. IEEE (2005)
18. Ma, D., Saxena, N., Xiang, T., Zhu, Y.: Location-aware and safer cards: enhancing RFID security and privacy via location sensing. IEEE TDSC **10**(2), 57–69 (2013)
19. Madlmayr, G., Langer, J., Kantner, C., Scharinger, J.: NFC devices: security and privacy. In: Third International Conference on Availability, Reliability and Security, ARES 2008. pp. 642–647. IEEE (2008)
20. Mehrnezhad, M., Hao, F., Shahandashti, S.F.: Tap-Tap and Pay (TTP): Preventing Man-in-the-Middle Attacks in NFC Payment Using Mobile Sensors. Technical report CS-TR-1428. Newcastle University, July 2014
21. Mehrnezhad, M., Hao, F., Shahandashti, S.F.: Tap-tap and pay (TTP): preventing man-in-the-middle attacks in NFC payment using mobile sensors. In: 2nd International Conference on Research in Security Standardisation (SSR 2015), October 2014
22. Roland, M., Langer, J., Scharinger, J.: Relay attacks on secure element-enabled mobile devices. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) SEC 2012. IAICT, vol. 376, pp. 1–12. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30436-1_1
23. Roland, M., Langer, J., Scharinger, J.: Applying relay attacks to Google Wallet. In: 2013 5th International Workshop on Near Field Communication (NFC), pp. 1–6, February 2013
24. Shepherd, C., Gurulian, I., Frank, E., Markantonakis, K., Akram, R., Mayes, K., Panaousis, E.: The applicability of ambient sensors as proximity evidence for NFC transactions. In: IEEE Security and Privacy Workshops on Mobile Security Technologies, MoST 2017. IEEE, May 2017
25. Shrestha, B., Saxena, N., Truong, H.T.T., Asokan, N.: Drone to the rescue: relay-resilient authentication using ambient multi-sensing. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 349–364. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_23

26. Shrestha, B., Saxena, N., Truong, H.T.T., Asokan, N.: Contextual proximity detection in the face of context-manipulating adversaries. CoRR abs/1511.00905 (2015). http://arxiv.org/abs/1511.00905
27. Trujillo-Rasua, R., Martin, B., Avoine, G.: The Poulidor distance-bounding protocol. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 239–257. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16822-2_19
28. Truong, H.T.T., Gao, X., Shrestha, B., Saxena, N., Asokan, N., Nurmi, P.: Using contextual co-presence to strengthen zero-interaction authentication: design, integration and usability. Pervasive Mob. Comput. **16**(Part B), 187–204 (2015). http://www.sciencedirect.com/science/article/pii/S1574119214001771. Selected Papers from the Twelfth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2014)
29. Truong, H.T.T., Gao, X., Shrestha, B., Saxena, N., Asokan, N., Nurmi, P.: Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentication. In: 2014 IEEE International Conference on Pervasive Computing and Communications, pp. 163–171. IEEE (2014)
30. Umar, A., Mayes, K., Markantonakis, K.: Performance variation in host-based card emulation compared to a hardware security element. In: 2015 First Conference on Mobile and Secure Services, pp. 1–6. IEEE (2015)
31. Urien, P., Piramuthu, S.: Elliptic curve-based RFID/NFC authentication with temperature sensor input for relay attacks. Decis. Support Syst. **59**, 28–36 (2014)
32. Varshavsky, A., Scannell, A., LaMarca, A., de Lara, E.: Amigo: proximity-based authentication of mobile devices. In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 253–270. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74853-3_15
33. Verdult, R., Kooman, F.: Practical atacks on NFC enabled cell phones. In: 2011 3rd International Workshop on Near Field Communication (NFC), pp. 77–82, February 2011

Instruction Duplication: Leaky and Not Too Fault-Tolerant!

Lucian Cojocar[1](✉), Kostas Papagiannopoulos[2], and Niek Timmers[3]

[1] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
l.cojocar@vu.nl
[2] Radboud University Nijmegen, Nijmegen, The Netherlands
k.papagiannopoulos@cs.ru.nl
[3] Security Lab, Riscure, Delft, The Netherlands
timmers@riscure.com

Abstract. Fault injection attacks alter the intended behavior of microcontrollers, compromising their security. These attacks can be mitigated using software countermeasures. A widely-used software-based solution to deflect fault attacks is *instruction duplication* and *n-plication*. We explore two main limitations with these approaches: first, we examine the effect of instruction duplication under fault attacks, demonstrating that as fault tolerance mechanism, code duplication does not provide a strong protection in practice. Second, we show that instruction duplication increases side-channel leakage of sensitive code regions using a multivariate exploitation technique both in theory and in practice.

1 Introduction

Fault Injection (FI) and Side-Channel Analysis (SCA) attacks are a risk for microcontrollers operating in a hostile environment where attackers have physical access to the target. These attacks can break cryptographic algorithms and recover secrets either by e.g. changing the control flow of the program (FI) or by monitoring the device's power consumption (SCA) with little or no evidence.

Multiple countermeasures such as random delays [12], masking [42], infection [25], data redundancy checks [33,35] and instruction redundancy [6] have been proposed to tackle these threats, yet their impact, effectiveness and potential interactions remain open for investigation. Such countermeasures can be implemented at hardware or at software level, often translating to overheads in silicon area and execution runtime. This exacerbates the need for a detailed analysis of the benefits introduced by these countermeasures before their actual deployment.

1.1 Motivation

In this work, we focus on the Instruction Duplication (ID) countermeasure, applied as a fault tolerance mechanism in software. The assembly-level redundancy introduced by ID can prevent attacks aiming to skip instructions and alter the control flow. Recent defenses (e.g., *infection* [19]) build further on code redundancy in order to provide a stronger protection.

Manually applying these defenses, however, does not scale well for a large code base that needs to be protected: it is an error-prone process and it costs many highly skilled man-hours, therefore, in practice, it is often automated using compiler techniques [8,32,37]. On top of protecting against fault attacks, compilers can also provide support to reduce the information leakage through side channels [4,9,10,34,39]. While there is previous work exploring the effect of one defense mechanism on another [5,30,41], to the best of our knowledge, the effect of ID on side-channel leakage has not been explored before. We perform an in-depth investigation of ID, focusing on its applicability against FI as well on its interaction with side-channel attacks.

Specifically, regarding fault attacks, the defender needs to exercise caution when applying ID, since the device may not adhere to the "single instruction skip" model. In such cases, the countermeasure is ineffective and we demonstrate that it can even benefit certain fault injection strategies. In addition, we highlight how even an effective application of ID can enhance our capability to perform side-channel attacks on the underlying implementation. Thus, we establish that care needs to be taken with respect to the equilibrium between fault injection defenses and side-channel resistance.

In the process of investigating these software defenses, we built the first open-source compiler capable of generating duplicated code for any C/C++ program. In this way, we hope to stimulate further research in this area.

1.2 Contribution

We summarize our contributions as follows:

- We experimentally determine that instruction skipping is not a realistic fault model for modern ARM Cortex-M4 MCUs.
- We develop and open source[1] an instruction duplication compiler for ARM Thumb2 architectures. To our knowledge, this is the first time that such a compiler is publicly available.
- We examine the interaction between n-plication and side-channel resistance and demonstrate the trade-off using an information-theoretic approach. In addition, we show how horizontal exploitation techniques can leverage the side-channel introduced by ID-based defenses.
- We examine how the redundancy of infective countermeasures can interact with side-channel resistance and demonstrate how a Hidden Markov Model can render infection [19] equivalent to ID from a side-channel point-of-view.

This paper starts with the background (in Sect. 2) and with an overview of the related work in Sect. 3. Sections 4 and 5 investigate the limitations of the assumed FI model as well as the limits of compiler-based ID. In Sects. 6 and 7 we determine the impact of hardening code with ID on SCA attacks. We summarize our findings in Sect. 8.

[1] The code is available at: https://github.com/cojocar/llvm-iskip.

2 Background

Software-based instruction redundancy methods for *fault detection* were proposed by Barenghi et al. [6]. In this technique, the original stream of instructions to be executed is duplicated (or even *triplicated*), one instruction after another, either manually or automatically [8,32,38].

For example a load from memory (`ldm r0, [r2, #0]`) is transformed by duplication in two loads originating from the same memory. To provide *fault detection* the destination registers must be different and then checked for differences (Listing 2). Under single instruction skip model, the *fault tolerance* arises when using the same register as destination. Indeed, skipping one single instruction from Listing 1 has the same effect as executing the original instruction.

```
ldr r0, [r2, #0]
ldr r0, [r2, #0]
```

Listing 1. Fault tolerance

```
ldr r0, [r2, #0]
ldr r1, [r2, #0]
cmp r0, r1
bne fault_detected
```

Listing 2. Fault detection

In practice, Moro et al. [37] showed that every ARM Thumb-1/2 instruction can be duplicated. We differentiate three classes of instructions: idempotent instructions, separable instructions and specific instructions. While the idempotent instructions are duplicable with no extra transformation, the other two classes often require an extra register to perform the duplication.

Therefore, on ARM Thumb-1/2, ID is generic and can be applied automatically regardless of the algorithm that the instruction stream implements.

Automatic Deployment. Maebe et al. [32] apply ID for fault detection at link-time for the ARM architecture. Barry et al. [8] described a compiler able to produce duplicated instructions, however their tool is not publicly available.

Our LLVM based compiler emits duplicated instructions for the ARM Thumb2 instruction set. Through code annotations, the hardening can be enabled or disabled at function level, as instructed by the developer. The modified LLVM based compiler has a similar architecture as the implementation described by Bary et al. [8] and it can compile code in any language supported by Clang (e.g. C, C++) with different optimization levels, including the AES-128 implementation used in this paper. It is designed to be a drop-in replacement for any LLVM based toolchain. Due to space constraints we omit the implementation details. The compiler is available as an open-source project.

3 Related Work

ID and the FI Model. Moro et al. [38] practically evaluates instruction duplication as a defense for FI on a Cortex-M3 Microcontroller (MCU). They use electromagnetic (EMI) pulses to insert glitches and show the importance of the fault model. Riviere et al. [45] show that the single instruction model is invalid when caches are enabled. The observed skip behavior, in the presence of an EMI glitch, is: the last 4 instructions are re-executed and 4 instructions are skipped—this partially invalidates the instruction duplication defense. Dureuil et al. [27] model the fault injection attack by including the EMI probe position. When an attack succeeds, the most probable outcome is to skip 1–4 instructions on a common smart card. They show that a probable outcome is the corruption to 0 of the destination operand of a `ld` instruction. Yuce et al. [48] show the effect of a single clock glitch on the ID scheme at clock granularity. They observe that the first instance of the instruction is corrupted and that its duplicated counterpart is transformed to a NOP instruction, thus defeating the ID. They use a 7-stage FPGA based implementation and clock glitches for experiments. Instead, we use a 3-stage pipeline off-the-shelf device and voltage glitches to investigate ID.

ID and SCA Interaction. Regazzoni et al. [43] first looked at the interaction between fault injection defenses and Power Analysis (PA) attacks. Specifically, they studied an AES implementation with parity based error detection circuitry. They conclude that the presence of a parity error detection circuit will leak important information to an attacker through PA. One year later, Regazzoni et al. [44] experimentally show the exploitability of an known-by-the-attacker error detection circuit. Pahlevanzadeh et al. [40] look at three fault detection methods designed specifically for AES: double module redundancy, parity checks, inverse execution; all implemented on an FPGA. They find that parity checks are actually improving the resistance against standard Correlation Power Analysis (CPA). Similarly, Luo et al. [31] use CPA to attack an FPGA implementation of AES which is hardened for fault detection. They conclude that duplication does not improve the success rate of the attack in respect to the unhardened AES implementation. However, we stress that the approaches of [31,40] use naive CPA attacks and do not rely on multivariate, horizontal exploitation of the leakage. Such attack-dependent techniques do not reveal the full picture and may lure the side-channel evaluator in a false sense of security.

4 FI Preliminaries

Because ID and n-plication are defenses for faults, we experimentally evaluate them in a realistic fault injection scenario.

4.1 Fault Injection Background

Fault injection attacks change the intended behavior of a target by manipulating its environmental conditions. This can be accomplished using different fault injection techniques such as: *voltage FI*, *electromagnetic FI* and *optical FI*. In this paper we focus only on *voltage FI* where glitches are introduced in the voltage signal that powers the subsystem responsible for executing software. Voltage FI is easy to mount as it does not require sophisticated equipment and it is invasive.

FI Model. Faults can target different physical layers of the device: single transistors, logic gates or computation units [47]. In this paper, we are interested in the observable effect of faults, namely, in faults that can cause a change in the program flow and that manifest at the instruction level. We note several types of faults in respect to instructions: single instruction skip [7], multiple instruction skip [45,46], instruction re-execution [29,45] and instruction corruption [46]. These types of faults are from now on referred to as the *fault model*.

Fault Injection Parameters. The following glitch parameters are important when performing voltage FI:

– the *Normal Voltage* is the voltage supplied to the target.
– the *Glitch Voltage* is the voltage *subtracted* from the *Normal Voltage* when the glitch is injected.
– the *Glitch Offset* is the time between when the trigger is observed and when the glitch is injected.
– the *Glitch Length* is the time for which the *Glitch Voltage* is set.

Finding the right parameters for a target is defined as *characterization*.

4.2 Experimental FI Setup

Fault Injection Target. All fault injection experiments described in this section are performed targeting an off-the-shelf development platform built around an STM32F407 MCU. This MCU is implemented using 90 nm technology and includes an ARM Cortex-M4 core running at 168 MHz. This Cortex-M4 based MCU has an instruction cache, a data cache and a prefetch buffer.

Related research used a similar experimentation target. Moro et al. [36,38] used a development board designed around an 130 nm technology MCU featuring an ARM Cortex-M3 core running at 56 MHz. The Cortex-M3 and Cortex-M4 are very similar and we expect the differences to have minimal impact. The latter includes additional specialized instructions which are not targeted in this paper. The pipeline size (3 stages) and the rest of the instruction set are the same.

To avoid instruction re-execution, which was shown to be possible by Rivier et al. [45], all experiments are performed with the prefetch buffer disabled and with caches enabled, unless otherwise stated.

Fault Injection Tooling. The voltage FI test bed is created using Riscure's VC Glitcher product[2] that generates an arbitrary voltage signal with a pulse resolution of 2 ns. Similarly to previous work, in a synthetic setup, we use a General Purpose Input Output (GPIO) signal to time the attack which allows us to inject a glitch at the moment the target is executing the targeted code. The target's reset signal is used to reset the target prior to each experiment to avoid data cross-contamination.

[2] https://www.riscure.com/security-tools/hardware/vc-glitcher.

4.3 Fault Injection Characterization

We use the code snippet from Listing 3 for two purposes: (a) to find the glitch parameters (characterization) and (b) to invalidate the single instruction skip model for the target described in Sect. 4.2. The code is a copy-loop construction that is known to be a common target for fault injection because it has significant duration [46]. The targeted code is executed in a loop to minimize the impact of the *Glitch Offset* parameter as it does not matter what iteration of the loop and which part of the loop is hit.

```
0: ldm r0, {r4-r10}
   stm r0, {r4-r10}
   subs r1, #1
   bne 0b    //loop back
```

Listing 3. Characterization code

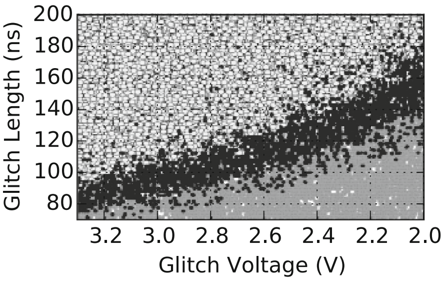


Fig. 1. Behavior under faults

The target's susceptibility to voltage FI attack is determined using the following glitch parameters: voltage $\in [-3.3\,\text{V}, -2.0\,\text{V}]$, offset $\in [2\,\mu s, 5\,\mu s]$ and length $\in [70\,\text{ns}, 200\,\text{ns}]$. The normal voltage is set to 3.3 V. The results of FI experiments can be classified in three groups: *Expected*, *Successful* and *Mute*. The experiments are plotted in Fig. 1 and show a clear relationship between the voltage and the length of the glitch. For the *Successful* experiments (black, the diagonal boundary) we observed a change in the target's behavior without affecting its continuation. For all *Mute* experiments (light gray, above the diagonal) the target halted or performed a reset. The *Expected* experiments (dark gray, below diagonal) are the ones for which we did not observe a change in the execution.

Instruction Corruption Model. Executing under faults the code from Listing 3 yields the following result: the memory pointed by `r0` after the loop is different that its contents before the loop (*Successful*). If only the instruction skip fault model applies to the target, then the memory pointed by `r0` should be the same as before the loop executes (*Expected*). We ran the experiment 20 K times and, in 15.91% (SE $= 25 \times 10^{-4}$) of cases were *Successful*. In 65.59% (SE $= 33 \times 10^{-4}$) of cases the device crashed or failed to answer and, the rest of the cases were *Expected*. The standard error (SE) is computed as $\sqrt{P * (1 - P)/N}$, where N is the number of experiments (20 K in this case) and P is the success rate.

The non-negligible number of *Successful* cases indicates that the target adheres to a more complex fault model than single instruction skip model – i.e.

the *instruction corruption* model. We say the *instruction corruption* fault model holds *iff* the observed behavior of the target under faults cannot always be explained by removing one (or multiple) instructions from the execution stream. This loose definition captures as well the data corruption fault model.

5 Fault Injection Effectiveness

In this section, we practically evaluate ID under instruction corruption FI model.

5.1 Inaccuracies in the FI Model

We resort to two experiments, that show how ID can negatively affect the fault tolerance of ID if a different model than single instruction skip holds. Furthermore, we show that when applying ID the runtime configuration of the target must be considered.

ID and the "real" FI Model. We determine the impact of ID by *duplicating* and n-plicating code from Listing 3. For each code instance, we perform 10 K experiments, using the glitch parameters outlined in Sect. 4.3.

Table 1. Success rate of FI and n-plication levels

	Original	$n = 2$ (ID)	$n = 3$	$n = 4$	$n = 5$
SR (%)	15.91	15.61	11.59	13.5	11.96
SE ($\times 10^{-4}$)	25	25	22	24	22

Table 1 shows that ID does not provide fault tolerance for software for our target. Even if the instruction is n-plicated three times or more, the fault tolerance is not substantially improved. Because we use a real target with no access to low level hardware features (i.e. flip-flop states), we do not aim to detail the root cause of this behavior. Instead, we note that the instruction corruption model captures this result.

Limitations of a Static FI Model. When ID is deployed automatically at compile time, the compiler is not aware of the runtime configuration (e.g. cache configuration). In this experiment, we show how ID and n-plication affects the success of FI when several runtime configurations are used.

In Fig. 2 we enable and disable the prefetch buffer (`p`), the instruction cache (`i`) and the data cache (`d`) and plot the fault injection success rate on the code similar to Listing 5. A capital letter in the title of the subplot means that the specific feature is enabled. We use the color scheme defined in Sect. 4.3.

Because the data on which our test operates is stored in registers, toggling the data cache has no impact on the fault tolerance. However, we observe four

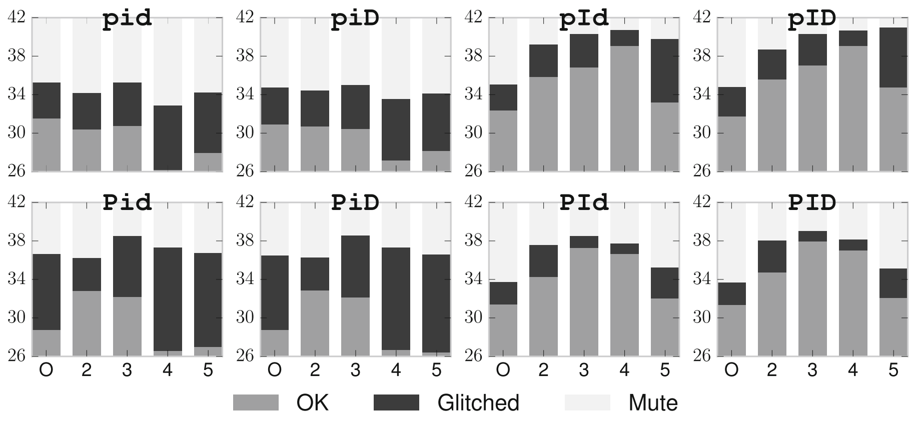


Fig. 2. SR of faults vs. multiple n-plication levels and runtime configurations

interesting results. First, ID increases the probability of a successful fault when the device is used with all its functionality enabled (`PID`). In this case, n-plication with $n = 3$ and $n = 4$ has the highest fault tolerance. Second, when all features are disabled (`pid`), none of the n-plication level improve the fault tolerance. Thirdly, when the instruction cache is disabled, enabling the prefetch buffer makes ID the most effective amongst the n-plication levels (`pid`, `piD` vs. `Pid`, `PiD`). Finally, comparing the right-most four subplots with the left-most subplots, the instruction cache offers an improved resilience against voltage glitches.

As a consequence, the compiler must be aware of the runtime configuration of the device when it emits redundant instructions.

5.2 Impact of Compiler Techniques

We now explore two compiler techniques that affect the effectiveness of ID.

Register Allocation Pressure. Register Allocation (RA) is the process in which the compiler maps the *virtual* (unlimited) registers to physical (limited) registers. This process is highly optimized to yield the best space and runtime performance. In this section we show that the modified register allocation scheme that ID requires has a negative impact on the fault tolerance.

```
add r5, r5, #1
add r7, r7, #1
```

Listing 4. Registers are incremented

```
add r4, r5, #1
mov r5, r4
add r6, r7, #1
mov r7, r6
```

Listing 5. Code ready for duplication

Listing 5 is the transformation of the code from Listing 4 with the `add` being replaced by an idempotent sequence that uses an extra temporary register (see Sect. 2). We define a successful glitch with respect to the contents of the registers `r5` and `r7`. If the contents of the registers is different than what is expected (i.e. the number of iterations added to the initial value of the registers) then we count this trial as a success. Otherwise, the glitch was not inserted or the parameters caused a *mute*.

The ID aware RA yields a higher success rate for FI (SR = 18.64%, SE = 20×10^{-5}) than the unmodified one (SR = 10.63%, SE = 16×10^{-5}). Apart from runtime performance degradation, the increased register pressure induced by the custom RA has a two fold negative impact on the fault tolerance. First, it increases the probability of a register to be *spilled* on the stack. As a consequence, the compiler will likely chose complex multi-memory access operations over simple load or stores. The multi-memory operations (e.g. `ldm`, `stm`) are more prone to faults than single memory operations [46] or than register to register operations. Second, an extra instruction to write back the result is needed (`mov`). This extra instruction is duplicated, therefore it increases the window in which a fault can be injected and it adds another leakage point.

In short, not only does the ID register allocation works against the established RA optimizations, but it also has a negative effect on the fault tolerance guarantees. This is a fundamental limitation of ID.

Instruction Ordering. The compiler has the freedom to emit instructions in any order. This is done either for optimization purposes (e.g. benefit from a multi-stage pipeline) or to avoid a certain illegal order of instructions. Barry et al. [8] showed that the correct scheduling of duplicated instructions can reduce the runtime overhead of the duplicated code, from 2.14X down to 1.70X–2.09X on a software AES implementation. Yuce et al. [48] hint at the interaction between ID and the processor pipeline.

To analyze what is the impact of the instruction order on the success rate of injected faults we compare the success rate of the code Listing 6 and its possible scheduled version Listing 7. We define a successful trial whenever the memory pointed by `r6` is different than its initial value. Our results show that instruction scheduling *decreases* the success rate of injecting a fault, from 8.51% to 4.00%.

Intuitively, the pipeline for Listing 6 contains the protected instruction and its copy right after another. Therefore, the chances that a fault affects the protected instruction and its copy at a given clock cycle is higher than in the case when the protected instruction and its copy are one (or more) instruction apart (Listing 7). These results are in line with the work of Yuce et al. [48], which shows that ID can be bypassed with a single glitch because multiple instructions are in the pipeline at a given clock cycle.

When emitting duplicated code the order is important, yet to date a FI model that captures the order interaction does not exist, let alone a compiler that uses this model. We leave the design of such a model and compiler as future work. We conclude that compiler optimization techniques (e.g. instruction scheduling, register allocation optimality) interact with the fault tolerance guarantees of ID.

```
add r0 , r4 , r1
add r0 , r4 , r1
ldr r5 , [r6 , #0]
ldr r5 , [r6 , #0]
```

Listing 6. Natural order

```
add r0 , r4 , r1
ldr r5 , [r6 , #0]
add r0 , r4 , r1
ldr r5 , [r6 , #0]
```

Listing 7. Possible re-ordering

6 SCA of ID and Infection Countermeasures

This section demonstrates the interactions between the redundancy-based FI countermeasures and the side-channel resistance of an implementation that is employing them. In Sect. 6.1 we analyze the theoretical effect of ID and n-plication on SCA using an information-theoretic approach. Section 6.2 demonstrates how to perform SCA on infective countermeasures using a Hidden Markov Model that simplifies the exploitation phase of infection to that of ID. Throughout this section, capital letters denote random variables and small case letters denote instances of random variables or constants. Bold letters denote vectors.

6.1 Information-Theoretic Evaluation of ID for SCA

From a side-channel perspective, the ID countermeasure increases the available leakage in a horizontal manner, either as a fault detection or as a fault tolerance mechanism. Analytically, in the case of an unprotected implementation (without ID) a univariate adversary can acquire the leakage of a key-dependent value v, i.e. observe $L_v \sim \mathcal{N}(v, \sigma)$, assuming identity leakage model. On the contrary, when instruction n-plication is implemented ($n > 1$), the adversary can observe over time an n-dimensional leakage vector $\mathbf{L}_v = [L_v^{t=1}, \dots, L_v^{t=n}]$. The vector contains n independent observations of value v under the same noise level, i.e. we assume that $L_v^t \sim \mathcal{N}(v, \sigma)$, $t = 1, \dots, n$.

Given that the side-channel adversary has located the sample positions of the repeated leakages, he can perform a pre-processing step where he averages all available samples that leak v, i.e. he computes $\bar{L}_v = (1/n) * \sum_{t=1}^{n} L_v^t$. The averaging step results in noise reduction of factor $\sqrt{n}$, obtaining $\bar{L}_v \sim \mathcal{N}(v, \sigma/\sqrt{n})$ and as a result side-channel attacks can be enhanced. Note that noise reduction can be particularly hazardous even when additional side-channel protection is implemented. For instance, both masking and shuffling countermeasures [13,14] amplify the existing noise of a device and will perform poorly if the noise level has been reduced by a large factor $\sqrt{n}$. In order to demonstrate the effect of noise reduction, we employ the information-theoretic framework of Standaert et al. [13] which evaluates the resistance against the worst possible attack scenario. The MI between the key-dependent value V and leakage $\mathbf{L}_v$ can be computed using the following formula: $MI(V; \mathbf{L}_v) = H[V] + \sum_{v \in \mathcal{V}} Pr[v] \cdot \int\limits_{\mathbf{l}_v \in \mathcal{L}^n} Pr[\mathbf{l}_v|v] \cdot \log_2 Pr[v|\mathbf{l}_c] \, \mathrm{d}\mathbf{l}_v,$ where $Pr[v|\mathbf{l}_v] = \frac{Pr[\mathbf{l}_c|v]}{\sum_{v^* \in \mathcal{V}} Pr[\mathbf{l}_v|v^*]}$.

From Fig. 4 we derive the following three conclusions. First, we observe that n-plication (for $n > 1$) shifts the MI-curve to the right, i.e. the FI countermeasure produces repeated leakages which have a direct impact on the side-channel security of the implementation. Second, we note that if ID translates to more than two assembly instructions that manipulate the same value, we will likely observe even more hazardous repetitions. Third, it follows that a countermeasure designer needs to balance the need for side-channel resistance and FI resistance by fine-tuning the parameter n.

6.2 Converting Infection to ID for SCA

It is important to point out that, apart from straightforward instruction duplication, a wide variety of FI countermeasures rely on some form of spatio-temporal redundancy. For instance, detection methods such as full/partial/encrypt-decrypt duplication & comparison of a cipher [21] produce repetitions of intermediate values that are exploitable by the side-channel adversary. Thus, an MI-based evaluation of duplication & comparison is identical to Fig. 4. Similarly, countermeasures that rely on particular error detection/correction codes [22] also introduce redundancy that has been evaluated in the side-channel context by Regazzoni et al. [26].

In this section, we expand in the same direction and examine the interaction between side-channel analysis and the more recent infective countermeasure [19][3]. Specifically, we demonstrate how the application of a Hidden Markov Model (HMM) [2,16] in a low-noise setting can render infective countermeasures equivalent to ID from a side-channel point-of-view.

Infective countermeasures were developed as a solution to the vulnerabilities of the duplicate & compare methods [25]. Instead of vulnerable comparisons, infection diffuses the effect of faults in order to make the ciphertext unexploitable. In particular, we focus on the infective countermeasure of Tupsamudre et al. [19], which has been proven secure against DFA [24], given that the adversary cannot subvert the control flow and that certain fault models are not applicable [20]. The countermeasure is shown in Algorithm 1.

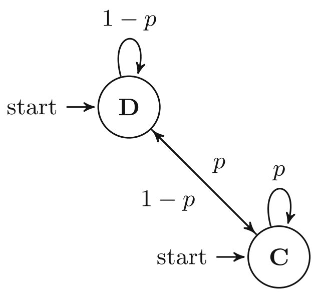
The infective countermeasure alternates between real, redundant and dummy cipher rounds (step 8). It requires an r bit random number $rstr$ (step 3), consisting of $2n$ 1's that trigger computation rounds (redundant or real) and $(r - 2n)$ 0's that trigger dummy rounds (steps 5–7). In the event of FI, the difference is detected via function $BLFN : size(R) \rightarrow 1$, where $BLFN(0) = 0$ and $BLFN(x) = 1, \forall x \neq 0$. The error is propagated via step 11.

From a side-channel perspective, the infective countermeasure can be viewed as a random sequence of r round functions, where only the $2n$ computation rounds are useful for exploitation. Thus, the objective of the side-channel adversary is to uncover the hidden sequence of rounds and to isolate the useful ones. Subsequently, one can exploit e.g. the first redundant and first real round

[3] Infective countermeasures in this [19] work do not pertain to the modular arithmetic infective techniques used by Rauzy and Guilley [3].

Algorithm 1: Infection Tupsamudre et al. [19]

```
Input: Plaintext $P$, key $K$, round $j$ key $k^j$, $\forall j = 1, \ldots, n$, $n$ number of rounds, dummy plaintext $\beta$, dummy round key $k^0$
Output: Ciphertext C=Cipher($P$, $K$)
1  Real $R_0 \leftarrow P$, Redundant $R_1 \leftarrow P$, Dummy $R_2 \leftarrow \beta$
2  $i \leftarrow 1$
3  $rstr \in_R \{0,1\}^r$ //r random bits
4  for $q = 1$ until $r$ do
5      $\lambda \leftarrow rstr[q]$
6      $\kappa \leftarrow (i \wedge \lambda) \oplus 2(\neg\lambda)$
7      $\zeta \leftarrow \lambda\lceil i/2 \rceil$
8      $R_\kappa \leftarrow RoundFunction(R_\kappa, k^\zeta)$
9      $\gamma = \lambda(\neg(i \wedge 1)) \cdot BLFN(R_0 \oplus R_1)$
10     $\delta \leftarrow (\neg\lambda) \cdot BLFN(R_2 \oplus \beta)$
11     $R_0 \leftarrow (\neg(\gamma \vee \delta) \cdot R_0) \oplus ((\gamma \vee \delta) \cdot R_2)$
12     $i \leftarrow i + \lambda$
13     $q \leftarrow q + 1$
14 end
15 return $R_0$
```



$$T = \begin{array}{c|cc} & \mathbf{C} & \mathbf{D} \\ \mathbf{C} & p & 1-p \\ \mathbf{D} & p & 1-p \end{array} \quad T_{pr} = \begin{array}{cc} \mathbf{C} & \mathbf{D} \\ p & 1-p \end{array}$$

Fig. 3. The Markov model describing the states, transition probabilities T and prior probabilities T_{pr}.

together via averaging, which is identical to the aforementioned exploitation of ID. Distinguishing effectively dummy rounds from computational ones is non-trivial, especially when extra randomization steps are involved [23]. However, the presence of control logic in the infective countermeasure such as variables λ, ζ and κ can emit noisy side-channel information about the sequence of rounds. We model such leakage as $\mathbf{L}_c = [\Lambda, Z, K] + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, where the deterministic part $[\Lambda, Z, K]$ is defined over $\{0,1\}^3$ and $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ denotes 3-dimensional noise vector with zero mean and diagonal covariance matrix $\mathbf{\Sigma}$.

The suggested HMM is constructed the following way. We encode the main loop of Algorithm 1 using two states, i.e. at a given time t, the state $s_t = i \in \{C, D\}$, where C corresponds to a computational round and D to a dummy round. The transitions in the sequence of states is described by matrix T, where $T_{i,j} = Pr(s_{t+1} = j | s_t = i)$. Figure 3 shows the state diagram and the probabilities for matrix T, namely $p = 2n/r$. We note that it is possible to unroll the loop and use additional states to describe the transitions, such that we can fine-tune the probabilities. However, we opt for such simple representation to minimize the model's data complexity.

In the HMM, the round sequence $\mathbf{s} = [s_1, \ldots, s_r]$ is unknown, but the adversary is assisted by leakage observations $[\mathbf{l}_c^{t=1}, \ldots, \mathbf{l}_c^{t=r}]$. To exploit the observations, the HMM associates every state $i \in \{C, D\}$ with an estimated emission probability function, i.e. emission $e_i(\mathbf{l}_c^t) = Pr(\mathbf{l}_c^t | s_t = i)$.

Having established the HMM for our scenario, we perform a simulated experiment where we try to identify the round sequence for a gradually increasing noise level. The simulated sequence contains 22 computational rounds and 78 dummy rounds, i.e. it corresponds to a computation of AES-128 using infection with $r = 100$. For every noise level we apply the Viterbi algorithm [1], which can

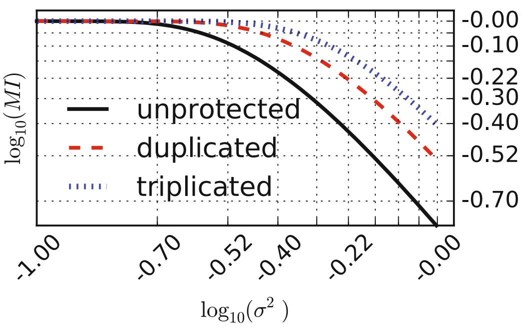


Fig. 4. MI of instruction n-plication

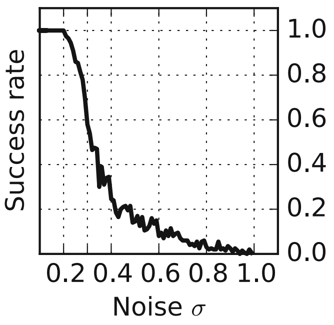


Fig. 5. Success rate of HMM-based sequence detection vs. noise level σ

recover the most probable sequence $\mathbf{s}$ of length r, while factoring in the leakage observations $\mathbf{l}_c^{t=1\ldots r}$ and the transition probabilities of T. The simulation (Fig. 5) shows that for fairly small noise levels (e.g. $\sigma < 0.3$) we are able to uncover the hidden sequence with high probability, making the side-channel exploitation of infection equivalent to the exploitation of instruction duplication.

7 Practical SCA Results

In this section, we apply the exploitation techniques of Sect. 6.1 in our experimental setup that protects an AES-128 implementation using ID. We verify the technique's applicability to real-world scenarios by showing their increased efficiency compared to standard SCA methods. We use an AVR MCU (XMEGA128D4) as the main target for our SCA experiments and we collect power traces using the open-source ChipWhisperer product[4]. The clock frequency of the target is 7.3728 MHz and we sample the power consumption of the target 4 times per clock cycle.

We use three different code patterns to evaluate the interaction between SCA and ID in different scenarios. Patterns Ⓐ and Ⓑ demonstrate how ID affects different instructions, namely instructions `eor` and `ld` respectively. Pattern Ⓒ showcases the duplicated key addition and Sbox parts of a lookup-table-based AES implementation (Fig. 6).

7.1 Horizontal Exploitation Using CPA

For the aforementioned patterns, we perform an experimental evaluation where we put forward a variant of the traditional Correlation Power Analysis (CPA) [11]. In the case of n-plication, we involve a horizontal averaging pre-processing strategy as follows.

[4] https://newae.com/tools/chipwhisperer/.

Ⓐ	`eor` **`r10`** `,` **`r17`**
Ⓑ	`ld` **`r10`** `,` `Y`

Ⓒ	`eor` **`r9`** `,` **`r17`**
	`add` `r28` `,` **`r9`**
	`ld` **`r10`** `,` `Y`

Fig. 6. Code snippets for the SCA experiments. `Y` is the output buffer and `r17` contains the hardcoded secret key.

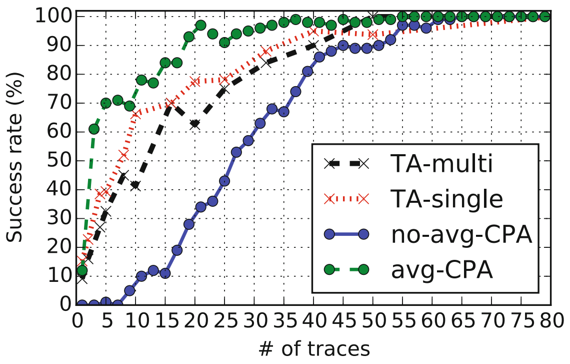


Fig. 7. CPA vs. Template on Ⓒ

1. Locate the intervals pertaining to the n different repeated leakages. In every interval, heuristically select the point in time with the highest correlation to the targeted key-dependent value, obtaining vector $\mathbf{l} = [l^{t=1}, \ldots, l^{t=n}]$.
2. For every vector $\mathbf{l}$ compute the average value $\bar{l} = (1/n) * \sum_{t=1}^{n} l^t$, thus reducing the noise level.
3. Perform CPA using the averaged values $(\bar{l})$.

In Fig. 8, we observe how the averaged CPA using a Hamming weight model outperforms naive CPA that ignores horizontal leakage, since it requires less traces to converge. Thus, the theoretical results of Sect. 6.1 are confirmed in practice and we conclude that horizontal averaging rejects noise. In addition, the difference between the naive CPA on the original code and averaged CPA on the duplicated code is larger on the duplicated `eor` pattern rather than on the duplicated `ld`. This behavior is attributed to the SNR of `ld/st` instructions, which is significantly higher compared to the SNR of ALU operations (such as `eor`)[5], since the later do not manipulate the memory bus. As a result, there is less need to reject noise on memory instructions. Last, we observe that a naive CPA attack when ID is in place may be slower to converge due to interference between duplicated consecutive instructions.

This work focuses on n-plication used as a fault tolerance mechanism, the same averaging technique can be applied when n-plication is used as a fault detection mechanism. In the latter case the instruction stream is the same as in the former case when no faults are injected, therefore, the side channel is similarly amplified.

7.2 Horizontal Exploitation Using Templates

In order to fully exploit the available horizontal leakage, we also use a template-based approach [15,18], which comprises two phases for attacking an AES-128

[5] SNR(Ⓐ) = 2.23 and SNR(Ⓑ) = 18.20.

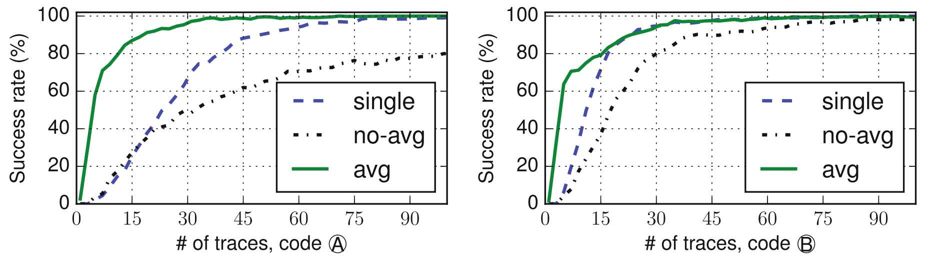


Fig. 8. Success rate of the CPA attack. *single* is CPA on the original code. On duplicated code, *no-avg* is the naive CPA and *avg* is the CPA with averaging

implementation: a profiling phase, in which templates are built for 256 key candidates of an AES-128 key byte and an extraction phase, where a number of traces are used to recover the unknown key. In our experiments, for the profiling phase, we use 3.2 k traces of the device per key candidate and perform dimensionality reduction, selecting Points of Interest (POIs) via Principal Component Analysis [17]. We deployed the following two template attacks. To ensure that the side-channel effect of ID is exploited during the heuristic step of POI selection, the first attack breaks the trace in multiple intervals, each containing a single assembly instruction and performs POI selection in every interval separately. The second template attack considers the full trace as a single interval and performs POI selection in the whole region.

In Fig. 7, we focus on code pattern Ⓒ. We perform the CPA attack (naive and averaged) that exploits the duplication of the `ld` instruction computing the Sbox output. Moreover, we perform the multi-interval and single-interval template attacks. We observe that both template attacks achieve similar performance and surpass the averaged CPA. Thus, we verify the applicability of templates in a horizontal context and conclude that they constitute an optimized way to exploit repeated leakages. We note that template attacks are inherently multivariate and may often require an extensive profiling phase to effectively characterize the model. On the other hand, averaged CPA compresses multiple samples, i.e. it is a univariate technique with a less informative model compared to templates, yet it has the upside of being faster to train and compute.

8 Conclusion

In this paper we analyzed the limitations of Instruction Duplication (ID) as a fault tolerance mechanism. First, we proved that the model under which ID operates has fundamental limitations, rendering the ID ineffective or even harmful. ID is designed under the assumption of a single fault model. However, in practice a more complex model can hold for a specific target, thus relying only on ID as a fault tolerance mechanism is not effective against FI attacks.

Second, the information leakage through side channel is amplified. We showed that the side channel introduced by instruction by ID, can be successfully exploited to extract secret information. Moreover, other instruction redundancy based defenses suffer from the same weaknesses in respect to side channels.

Finally, while automatically applying redundancy based defenses is promising, the FI model has to be fine tuned and extended for each targeted device according to its runtime configuration. The compiler must use this model to carefully balance fault tolerance guarantees and performance. Whether or not this is possible is still an open question.

Acknowledgements. This research was supported by the NWO CYBSEC "OpenSesame" project (628.001.005) and the NWO project ProFIL (628.001.007). We thank our anonymous reviewers and our shepherds, Fischer Jean-Bernard and Romailler Yolan for their invaluable feedback. We also thank Marius Schilder and Dominic Rizzo from Google Inc. for their support in developing the compiler.

Appendix

Differential Fault Analysis (DFA) Attack on Software AES-128

In Sect. 5 we determined the impact of ID as a fault tolerance mechanism on synthetic code. Now we show the interaction between ID and the number of trials needed to conduct a fault based attack. To this extent, we automatically apply ID on a large and complex code construction, the *AES-128* cryptographic algorithm, and perform the DFA attack described by Dusart et al. [28]. The goal of the attack is to extract the fixed key by observing the faulty output.

We use the tiny-AES128-C[6] implementation of the AES-128 cipher, in ECB mode for our target to encrypt a fixed input with a fixed key. A trigger is implemented between the 9th and the 10th round to guarantee we always hit the right location within the algorithm. Two versions of the AES-128 implementation are compiled: a *hardened* version (with ID in place) and an *non-hardened* version.

A 2 K trace set containing traces with faulty outputs is acquired for each implementation. We randomly select n_t from these trace sets and use them in the DFA attack. We repeat this process 100 times for each implementation and we plot how often the attack is successful in Fig. 9.

The non-hardened implementation outperforms the hardened implementation in terms of FI tolerance. A clear indication that ID is not effective for protecting the AES-128 algorithm when the instruction corruption fault model holds. Depending on the time penalty required for a single experiment, the small difference can have a noticeable effect. If the target needs to be reset before each experiment then tens of seconds are added for each experiment. Moreover, the target might remove or change the keys after a limited amount of encryptions.

We analyzed the outputs in more detail and counted how often multi byte changes are observed in both implementations (Table 2). From the number of all faults observed (i.e. at least 1 byte difference), 4 bytes faults[7] are more probable to be observed in the hardened implementation.

To conclude, fewer successful faults are needed to attack the hardened AES.

[6] https://github.com/kokke/tiny-AES128-C.
[7] These are the faults useful for DFA on AES.

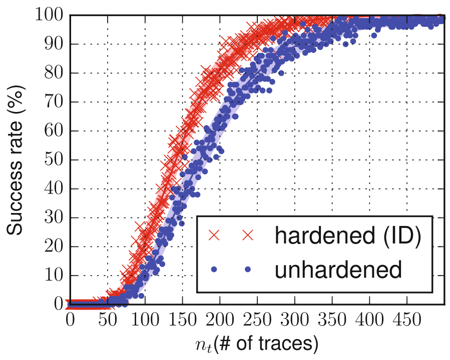


Fig. 9. DFA on AES-128

Table 2. Bytes changed in the output

	3 or less	4	5 or more
Hardened (ID)	0.2%	64.0%	35.7%
Unhardened	1.1%	41.5%	57.4%

References

1. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans. Inf. Theor. **13**(2), 260–269 (1967). https://doi.org/10.1109/TIT.1967.1054010. ISSN 0018-9448
2. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989). https://doi.org/10.1109/5.18626. ISSN 0018-9219
3. Rauzy, P., Guilley, S.: Countermeasures against high-order fault-injection attacks on CRT-RSA. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 68–82, September 2014. https://doi.org/10.1109/FDTC.2014.17
4. Agosta, G., Barenghi, A., Pelosi, G.: Automated instantiation of side-channel attacks countermeasures for software cipher implementations. In: Proceedings of the ACM International Conference on Computing Frontiers, CF 2016, Como, pp. 455–460. ACM (2016). https://doi.org/10.1145/2903150.2911707. ISBN: 978-1-4503-4128-8
5. Amiel, F., et al.: Passive and active combined attacks: combining fault attacks and side channel analysis. In: Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2007, pp. 92–102. IEEE (2007)
6. Barenghi, A., et al.: countermeasures against fault attacks on software implemented AES: effectiveness and cost. In: Proceedings of the 5th Workshop on Embedded Systems Security, p. 7. ACM (2010). http://dl.acm.org/citation.cfm?id=1873555. Accessed 14 Oct 2016
7. Barenghi, A., et al.: Fault injection attacks on cryptographic devices: theory, practice, and countermeasures. Proc. IEEE **100**(11), 3056–3076 (2012)
8. Barry, T., Couroussé, D., Robisson, B.: Compilation of a countermeasure against instruction-skip fault attacks. In: Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, pp. 1–6. ACM (2016). http://dl.acm.org/citation.cfm?id=2858931. Accessed 14 Oct 2016
9. Bayrak, A.G., et al.: A first step towards automatic application of power analysis countermeasures. In: Proceedings of the 48th Design Automation Conference, DAC 2011, San Diego, pp. 230–235. ACM (2011). https://doi.org/10.1145/2024724.2024778. ISBN: 978-1-4503-0636-2

10. Bayrak, A.G., et al.: Automatic application of power analysis countermeasures. IEEE Trans. Comput. **64**(2), 329–341 (2015)
11. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
12. Clavier, C., Coron, J.-S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44499-8_20
13. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_7
14. Veyrat-Charvillon, N., Medwed, M., Kerckhof, S., Standaert, F.-X.: Shuffling against side-channel attacks: a comprehensive study with cautionary note. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 740–757. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_44
15. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 253–270. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_17
16. Durvaux, F., Renauld, M., Standaert, F.-X., van Oldeneel tot Oldenzeel, L., Veyrat-Charvillon, N.: Efficient removal of random delays from embedded software implementations using Hidden Markov Models. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 123–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37288-9_9
17. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_1
18. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
19. Tupsamudre, H., Bisht, S., Mukhopadhyay, D.: Destroying fault invariant with randomization - a countermeasure for AES against differential fault attacks. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 93–111. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_6
20. Battistello, A., Giraud, C.: A note on the security of CHES 2014 symmetric infective countermeasure. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 144–159. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43283-0_9
21. Lomné, V., Roche, T., Thillard, A.: On the need of randomness in fault attack countermeasures - application to AES. In: FDTC 2012 (2012)
22. Malkin, T.G., Standaert, F.-X., Yung, M.: A comparative cost/security analysis of fault attack countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 159–172. Springer, Heidelberg (2006). https://doi.org/10.1007/11889700_15
23. Gierlichs, B., Schmidt, J.-M., Tunstall, M.: Infective computation and dummy rounds: fault protection for block ciphers without check-before-output. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 305–321. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33481-8_17

24. Patranabis, S., Chakraborty, A., Mukhopadhyay, D.: Fault tolerant infective countermeasure for AES. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) SPACE 2015. LNCS, vol. 9354, pp. 190–209. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24126-5_12
25. Joye, M., Manet, P., Rigaud, J.-B.: Strengthening hardware AES implementations against fault attacks. In: IET Information Security (2007)
26. Regazzoni, F., Breveglieri, L., Ienne, P., Koren, I.: Interaction between fault attack countermeasures and the resistance against power analysis attacks. In: Joye, M., Tunstall, M. (eds.) Fault Analysis in Cryptography. Information Security and Cryptography, pp. 257–272. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29656-7_15
27. Dureuil, L., Potet, M.-L., de Choudens, P., Dumas, C., Clédière, J.: From code review to fault injection attacks: filling the gap using fault model inference. In: Homma, N., Medwed, M. (eds.) CARDIS 2015. LNCS, vol. 9514, pp. 107–124. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31271-2_7
28. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45203-4_23
29. Korak, T., et al.: Clock glitch attacks in the presence of heating. In: FDTC 2014 (2014)
30. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 320–334. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_22
31. Luo, P., et al.: Side-channel power analysis of different protection schemes against fault attacks on AES. In: ReConfig 2014 (2014)
32. Maebe, J., De Keulenaer, R., De Sutter, B., De Bosschere, K.: Mitigating smart card fault injection with link-time code rewriting: a feasibility study. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 221–229. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_19
33. Maistri, P., Leveugle, R.: Double-data-rate computation as a countermeasure against fault analysis. IEEE Trans. Comput. **57**(11), 1528–1539 (2008)
34. Malagón, P., et al.: Compiler optimizations as a countermeasure against side-channel analysis in MSP430-based devices. Sensors **12**(6), 7994–8012 (2012)
35. Medwed, M., Schmidt, J.-M.: A generic fault countermeasure providing data and program flow integrity. In: 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2008, pp. 68–73. IEEE (2008)
36. Moro, N., et al.: Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. In: FDTC 2013 (2013)
37. Moro, N., et al.: Formal verification of a software countermeasure against instruction skip attacks. J. Cryptogr. Eng. **4**(3), 145–156 (2014)
38. Moro, N., et al.: Experimental evaluation of two software countermeasures against fault attacks. In: HOST 2014 (2014)
39. Moss, A., Oswald, E., Page, D., Tunstall, M.: Compiler assisted masking. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 58–75. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_4
40. Pahlevanzadeh, H., Dofe, J., Yu, Q.: Assessing CPA resistance of AES with different fault tolerance mechanisms. In: ASP-DAC 2016 (2016)
41. Patranabis, S., et al.: One plus one is more than two: a practical combination of power and fault analysis attacks on PRESENT and PRESENT-like block ciphers. In: FDTC 2017. IEEE (2017)

42. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_9
43. Regazzoni, F., et al.: Power attacks resistance of cryptographic s-boxes with added error detection circuits. In: DFT 2007 (2007)
44. Regazzoni, F., et al.: Can knowledge regarding the presence of countermeasures against fault attacks simplify power attacks on cryptographic devices? In: DFT 2008 (2008)
45. Riviere, L., et al.: High precision fault injections on the instruction cache of ARMv7-M architectures. In: HOST 2015 (2015)
46. Timmers, N., Spruyt, A., Witteman, M.: Controlling PC on ARM Using Fault Injection. In: FDTC 2016 (2016)
47. Verbauwhede, I., Karaklajic, D., Schmidt, J.-M.: The fault attack jungle-a classification model to guide you. In: FDTC 2011 (2011)
48. Yuce, B., et al.: Software fault resistance is futile: effective single-glitch attacks. In: FDTC 2016, pp. 47–58 (2016)

An EM Fault Injection Susceptibility Criterion and Its Application to the Localization of Hotspots

Maxime Madau[1,2](✉), Michel Agoyan[1,2], and Philippe Maurine[1,2]

[1] Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier (LIRMM), Montpellier, France
philippe.maurine@lirmm.fr

[2] STMicroelectronics, Rousset, France
{maxime.madau,michel.agoyan}@st.com

Abstract. Electromagnetic (EM) fault injection has been proven efficient in attacking targets such as system-on-chip (SoC) or smartcards. Nonetheless, security characterisations, performed either by certification laboratories or by firms, are time consuming and this impacts on the final result. Indeed complete tests of integrated circuits (ICs) require trying numerous parameters, from pulse polarity to probes geometry and coupling, hence many maps are required to test all surface of Devices Under Test (DUT) and are unfortunately rarely performed.

In this paper we propose a criterion to find zones with a high susceptibility to EM Fault Injection (EMFI). By using preprocessing tools based on both the effects of EMFI on circuits and the analysis of EM emission traces, we are able to speed up the search of zones where faults are more likely to occur hence reducing the time required for security characterisations.

Keywords: EM fault injection · EM susceptibility
Security characterisation

1 Introduction

Nowadays we are more and more surrounded by embedded devices either in automotive or Internet of Things (IoT). The security of these devices was recently brought back to the foreground with the Mirai botnet. Thus for security reasons or for added-market value, firms add cryptographic components in their devices either in hardware or software form. To ensure these measures are effective, firms test them in order to avoid being hijacked just after releasing a product on the market, which would result in very bad advertising for associated systems-on-chips or microcontrollers. There are also some markets that require security certifications such as Common Criterion norm or EMVCo. Evaluators must perform several tests to certify product's robustness against threats, including Fault Attacks (FA).

Fault Attacks are one of the major threats to electronic devices since they corrupt data which are normally inaccessible as shown by the recent Rowhammer attack [7,11]. Additionally faults can also be used to target cryptographic algorithms using correct and faulty ciphertext to obtain secret keys. In this case, we talk about Differential Fault Analysis (DFA) [1,12].

The most powerful way to induce faults in Integrated Circuits (IC) is the use of a laser beam because of its high spatial and time resolutions. However, the use of a laser beam requires access to the die of IC, access which is increasingly difficult for complex systems.

Another way for inducing faults into ICs has been introduced in 2002 in [14]. In this paper, authors demonstrated that they were able to corrupt memories using an external EM field. Later an EM Pulse (EMP) was used to inject faults into a CRT RSA [15].

The main advantages of EM Fault Injection (EMFI) are that it is inexpensive, able to target a chip without having to remove the packaging or perform additional pretreatments, and above all, it can induce faults on either the front or the back side.

Nowadays there are two types of EMFI platforms. On one hand there are harmonic platforms (described later) which have been proven efficient to influence the behavior of true random generators [2]. On the other hand there are EMP injection platforms. That deliver a short EMP inducing a parasitic current into ICs which in return generates faults.

Despite their efficiency, fault attacks are problematic when it comes to their evaluation. Indeed, both laser and EMP attacks suffer of their combinatorial complexity, which implies a trade-off between parameters to test and the time allocated to their characterisation. This complexity comes from the large number of parameters involved in the setting of an EMFI. To list a few of them related to EM pulse injection, we can distinguish:

- the pulse amplitude, width and polarity,
- the position, the orientation and the geometry of the injection probe,
- the injection time, etc.

In this paper we propose a technique to ease EMFI characterisations and more precisely to decrease the spatial complexity by introducing a technique to select the most susceptible ICs' coordinates to EMFI.

The proposed method relies on the state of the art related to EMFI and more precisely on the so called Sampling Fault Model [8]. It also relies on the exploitation of several EM traces (coarse grain EM maps) collected during the execution of the algorithm under test. These EM traces are processed according to an EMFI Susceptibility Criterion (EMFISC) introduced in this paper used to rank the different coordinates of the IC surface with regards to their EMFI susceptibility and thus identifying EM hotspots. One key point here is that EMFI susceptibility is not considered as a constant IC characteristic but rather as a figure of merit depending on the ICs' activity. EM susceptibility thus varies in time with the applications it executes.

The method we propose lets evaluators choose a percentage P of the target IC surface they want to keep or test with regards to the available time for characterisation. It returns the appropriate list of positions over the IC characterized by the highest EMFI susceptibilities. This method thus identifies the reduced zone of the device under test (DUT) surface with the highest EMFI susceptibility. Hence it enables performing more complete tests and to take full advantage of the time dedicated to security characterisation.

Relying on EM emission maps is not a drawback in terms of characterisation time because EM maps are fast to acquire. To give an order of magnitude, using our experimental setup (shown Fig. 1), one can acquire an EM emission map for three different algorithms within a day, while three days are necessary to acquire a fault injection map of one executable with only one EM pulse configuration (a single pulse width, a single pulse amplitude and a single pulse polarity).

This paper is organized as follows. After the short preamble of Sect. 2, Sect. 3 gives the state of the art about EMFI with a focus on EMFI fault models. Section 4 then details the sampling fault model and introduces the EMFI Susceptibility Criterion on which relies the hotspot localization method. Section 5 gives experimental results on two different targets (designed with two different CMOS technologies) running the same kind of algorithms. Finally a conclusion is drawn in Sect. 6.

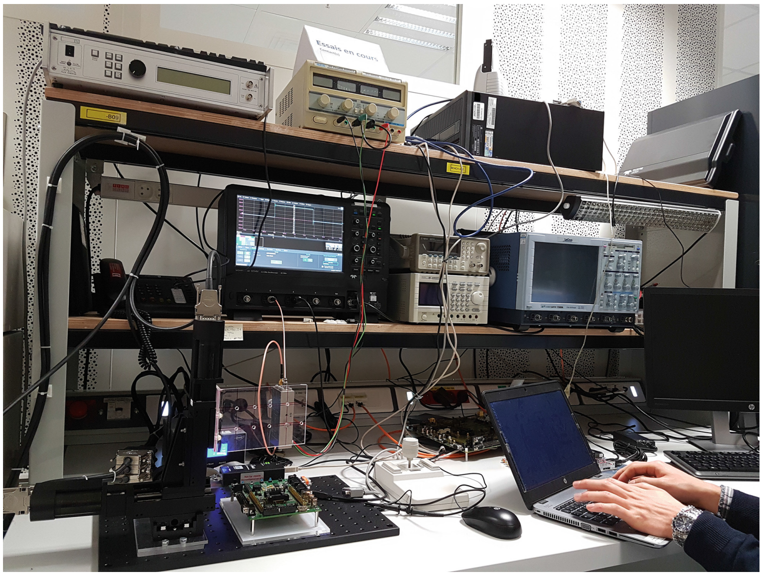


Fig. 1. EM analysis & EMFI platform

2 Preamble

For the sake of simplicity, in this paper we extend the term *data* to all binary words used by an algorithm. In other words, we do not make a distinction between *data* and *instructions* and they are both denoted by the term *data*.

Similarly, we use the term *fault* to refer to any perturbation of the behavior of the algorithm under test. So faults can either be no-responses, bit sets, bit resets, instruction corruptions, memory corruptions, etc.

3 Related Works

Two means to induce fault injections recently appeared. The first one is Body Bias Injection which consists in injecting a voltage spike directly into the substrate of the DUT to produce ground bounces or voltage drops [16]. The second one is EM injection which, despite the early warning of Quisquater et al. in 2002 [14], only found recently a larger echo in the community thanks to its inherent advantages highlighted in [15].

Two types of EM injection platforms can be mounted to induce faults into ICs. Harmonic EMFI platform refers to the first type. It produces continuous sinusoidal EM waves to induce faults in analogue blocks like internal clock generators [13] or true random number generators (TRNG) [2].

Pulsed EMFI platform refers to the second type of platform. The assumption done in this paper is that it produces a single but powerful EMP that creates a sudden current flow in the power/ground networks of ICs and therefore voltage drops and/or ground bounces. The efficiency of this type of platform was first reported in [3] to inject faults into an older microcontroller (designed with 350 nm technology).

A first analysis of the fault obtained with pulsed EMFI platforms was conducted in [4]. The authors concluded that pulsed EMFI produces timing faults and more precisely setup time constraint violations. As a result, a delay-based glitch detector was evaluated against EMFI in [17] and demonstrated partially efficient. However this result was questioned in [10] that demonstrated that EMFI can induce faults in a DUT at rest and more precisely when its clock is stopped. Following this work, authors of [8,9] introduced a specific model for pulsed EMFI called the Sampling Fault Model. This model was then used in [6] to build a fully digital EMP detector.

4 Hot-Spots Localization and EMFI Susceptibility Criterion

Since the Sampling Fault Model is the state of the art to describe how EMFI interact with ICs it is at the heart of the criterion. Hence, it will be detailed in the next paragraphs. Then our hotspots localization method is gradually introduced.

4.1 The Sampling Fault Model

The sampling fault model, which is illustrated Fig. 2, states that the EM susceptibility of synchronous ICs is:

- periodic with a periodicity equal to the DUT clock frequency, f_{CK}.
- maximal during short time windows (during which it is therefore much more easy to inject a fault) centered around rising edges of the clock signal.
- minimal the rest of the time.

This periodic behavior is explained in [8,9] by the fact that DFF are the most susceptible gates in ICs and especially during their switching. DFF higher susceptibility over any other standard CMOS logic gates (inverter, nand, ...) is due to the constraints that must satisfy their input signals and supply voltage for a correct operation. All DFF's operation constraints can be summed up by stating that their input data (D on Fig. 2), their clock input (CK) and their supply voltage (Vdd and Gnd) must not be disrupted during their switching. Stability constraints of D signal are well known from IC designers and are called the setup time and the hold time constraints of D. They state that D must be stable $t_{setup}ps$ before the rising clock edge and hold $t_{hold}ps$ after clock rising edge.

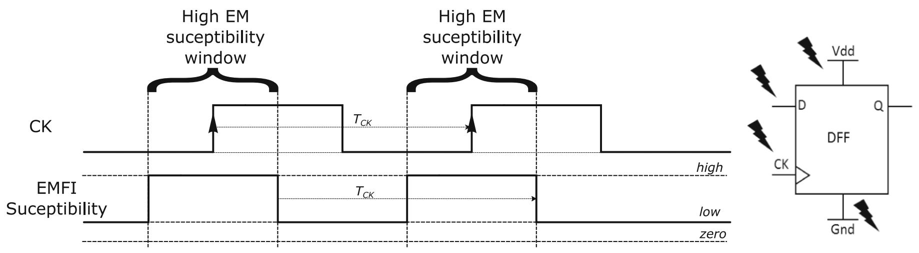


Fig. 2. Sampling Fault Model: (left) evolution of the EMFI susceptibility with regards to the clock signal CK, (right) lightnings showing the EMFI injection paths on a DFF

The Sampling Fault Model is therefore a model at functional level which does not take into account physical details about ICs' design except one. This exception is not a drawback and is related to the higher susceptibility of DFF with regards to other CMOS gates. The sampling fault model thus seems sufficiently general to be at the heart of an EMFI hotspots localization method.

4.2 Guidelines for Detecting EMFI Hotspots

From the above considerations, one may conclude that the Sampling Fault Model simply expresses that EMFI mainly induces faults by either disrupting the input signals D of DFF, their supply voltage (Vdd and Gnd nets) or the clock signal (CK). It thus appears that identifying EMFI hotspots from EM emission maps consists in finding on-chip antennas/EM probe positions:

- (guideline 1) emitting the signals which is tightly bind to the execution of the target algorithm and to the clock frequency (f_{CK}) i.e. the switching of the DFF involved in the computation of the algorithm,
- (guideline 2) emitting the strongest signal (in terms of power) associated to the clock signal or clock tree.

Indeed these antennas (made up by interconnects below the EM analysis probe) emitting the strongest signal (related either to the CK signal or the switching of DFF) constitute the best entry points for EMFI for two reasons. On one hand guideline 1 can be viewed as a side channel analysis aiming at looking forward to areas of IC surface where EM emissions enable to differentiate two runs of the same algorithm with different data. In other words the goal is to find EMFI susceptible areas of the running algorithm. On the other hand guideline 2 aims at finding the best entry point for EMFI according to the reciprocity of antennas. This property can be stated as follows: the efficiency of a receiving antenna is as important as its transmitting efficiency. In the present context it means that if an area of a circuit characterized by powerful EM emissions it is very likely to receive well the energy of an EM pulse provided the EM emission traces are collected with the injection probe or with a probe with the same characteristics.

4.3 EMFI Susceptibility Criterion

There are surely different ways of translating these guidelines into a criterion allowing to process EM analysis maps to disclose EMFI hotspots. The following paragraphs reports the most efficient criterion we have tested.

Power Spectral Density. The two above guidelines indicate that one has to rank antennas (EM probe positions above the IC surface) according to the power related to the clock signal they emit or to the power related to DFF switching they radiate. The most natural and simplest way is to consider the Power Spectral Density, $psd(f)$, of EM traces at $f = f_{CK}$. This of course requires the knowledge of the clock frequency value, or if this not the case, to deduce it from EM measurements. This is not a big issue in practice.

Magnitude Squared Incoherence. If the $psd(f)$ at $f = f_{CK}$ allows quantifying the power emitted by DFF on the clock tree, it does not allow deciding if these DFF are involved or not in the execution of the target algorithm i.e. are depending or not on the data processed by the algorithm. To discriminate DFF involved in the target algorithm execution among all DFF one can use a Student's t-test on the values of $psd(f_{CK})$ obtained for different data. One can also employ the Magnitude Squared Incoherence as suggested in [5] to estimate how much the power emitted at a position is linked to the data processed by the target algorithm. However, contrarily to [5] only the Magnitude Squared Incoherence at the frequency $f = f_{CK}$ must be considered accordingly to our guidelines.

We have chosen to employ the incoherence because with only $n = 50$ traces we get $\frac{1}{2} \cdot n \cdot (n-1) = 1225$ estimates in total with a stable mean value. Indeed, we recall that the Magnitude Squared Incoherence (inc) at a frequency f is a measure of dissimilarity between two time domain signals, $s_1(t)$ and $s_2(t)$. It is computed as follows:

$$inc_{s_1,s_2}(f) = 1 - \frac{psd_{s_1,s_2}(f)^2}{psd_{s_1,s_1}(f) \cdot psd_{s_2,s_2}(f)} = 1 - \frac{C_{s_1 s_2}(f)^2}{C_{s_1 s_1}(f) C_{s_2 s_2}(f)} \tag{1}$$

where $psd_{s_1,s_1}(f)$ and $psd_{s_2,s_2}(f)$ are the power spectral densities of $s_1(t)$ and $s_2(t)$ respectively, and $psd_{s_1,s_2}(f)$ is the cross power spectral density of those two signals. It can also be seen in terms of correlation in the spectral domain where C_{s_i,s_j} is the Fourrier transform of the cross-correlation of signals s_i and s_j (or autocorrelation if $i = j$). At a given frequency f, a $inc_{s_1,s_2}(f)$ value of 0 indicates that the two signal spectra have exactly the same amplitude i.e. are coherent while a value of 1 means that the signal spectra are fully different i.e. incoherent.

EMFI Susceptibility Criterion. $psd(f_{CK})$ and $inc_{s_1,s_2}(f_{CK})$ are two distinct criteria allowing to cover the two guidelines of Sect. 4.2, defined to identify EMFI hot spots thanks to near field EM scans of ICs. However these two guidelines have to be gathered into a single criterion. Unfortunately, the Sampling Fault Model does not provide any insight about which type of faults (among faults on the CK signal and faults on DFF) have to be privileged. To overcome this lack, we gathered the two distinct criteria in a tunable criterion. It takes the form of an euclidean distance in a well chosen plan and provide an EMFI Susceptibility score to each (x, y) coordinate of the IC surface.

To introduce this criterion let us recall that at this stage of our reasoning we have at disposal two matrices acquired by near field scan of the DUT. The first one is a matrix associated to Power Spectral Density distribution, $PSD_{x,y}$ over the IC surface and the second one is the matrix $INC_{x,y}$ related to the distribution of the Incoherence.

However those two quantities have different order of magnitude which is expected according to the different nature of the metrics (PSD and Incoherence). This difference is illustrated on Fig. 3(a) which represents values of matrices $PSD_{x,y}$ and $INC_{x,y}$ without taking into account their position on the IC. Since the re-scaling must be done at the IC surface level and must be robust to the occurrence of outlier measurements it is done by standardizing these two distributions (Fig. 3(b)) into two new matrices $PSDn_{x,y} = \{psdn_{x,y}\}$ and $INCn_{x,y} = \{incn_{x,y}\}$ with

$$psdn_{x,y} = \frac{psd_{x,y} - \overline{psd}}{\sigma(psd)} \tag{2}$$

$$incn_{x,y} = \frac{inc_{x,y} - \overline{inc}}{\sigma(inc)} \tag{3}$$

with $\overline{psd}$ (resp. $\overline{inc}$) the mean of all $psd_{x,y}$ (resp. $inc_{x,y}$) values and $\sigma(psd)$ and $\sigma(inc)$ the standard deviations of the associated distributions.
This re-scaling done, these two quantities can be combined to get an EMFI Susceptibility matrix (and therefore a new mono variate distribution), $EMFISC_{x,y} = \{emfisc_{x,y}\}$, with $emfisc_{x,y}$:

$$emfisc_{x,y} = \sqrt{\begin{array}{l}(1-a)\cdot(psdn_{x,y} - \min\limits_{x,y}(psdn_{x,y}))^2 + \\ a\cdot(incn_{x,y} - \min\limits_{x,y}(incn_{x,y}))^2\end{array}} \tag{4}$$

Where a is an empirical coefficient allowing to give more importance to one guideline than to the other.

As illustrated on Fig. 3 by the arrows, this criterion defines as highly EMFI susceptible (x, y) positions characterized by powerful emissions at f_{CK}, or by emissions highly dependent of the processed data or both. This is in accordance with our guidelines of Sect. 4.2 and therefore with the Sampling Fault Model.

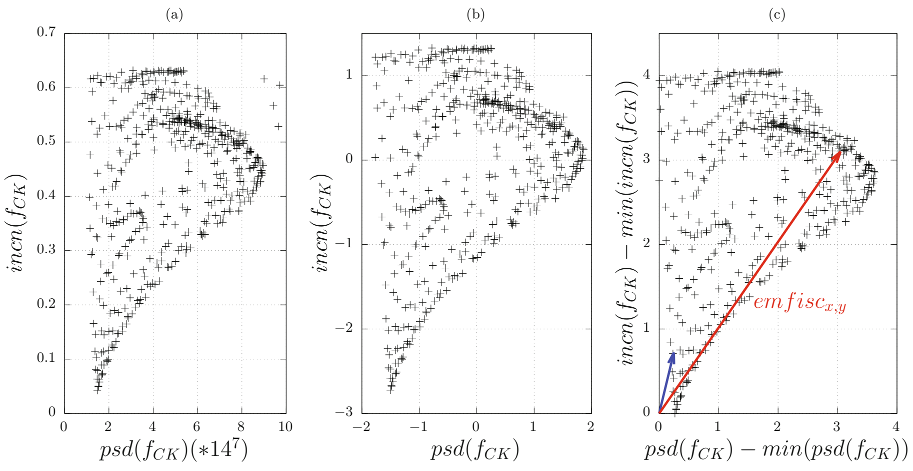


Fig. 3. (a) Joint distributions of raw $psdn_{x,y}$ and $incn_{x,y}$ values, (b) standardized joint distributions of $psdn_{x,y}$ and $incn_{x,y}$ values and (c) standardized joint distributions of $psdn_{x,y}$ and $incn_{x,y}$ values after change of the origin. In this example a = 0.5 thus the norm of red (blue) arrow shows a high (low) EMFI Susceptibility value. (Color f igure online)

EMFI Hot Spots Selection Method. At that stage of the paper we have a matrix representing the EMFI susceptibility distribution on the IC surface. The last point is to define a simple and practical method to select a set of points to be tested considering that an evaluator just want to test a given percentage β of the surface for time constraint reasons.

Being given the $EMFISC_{x,y}$ matrix, the solution is straightforward. This evaluator has just to reject the $\alpha = (1 - \beta)$ % positions of the IC surface with the lower $emfisc_{x,y}$ values, i.e. all points falling in the quantile q_α of the $EMFISC_{x,y}$ distribution.

EMFISC Protocol. An experimental protocol can thus be drawn from those metrics and has been used in the experimental part of this paper. To begin with the application of the criterion introduced in the preceding paragraphs requires to meet some constraints.

Firstly the running frequency f_{CK} of the target has to be known or deduced from experiments. Let's take the example of an evaluator who have to characterise an AES. If it is a hardware AES then the evaluator requires to know its running frequency which could be different from the rest of the IC while for its software counterpart the device clock frequency is required.

Secondly the computation of the criterion $emfisc_{x,y}$, done at several coordinates (x, y) above the IC, is done from raw EM traces of the signal s_1, where s_1 refers to the EM emissions of the IC during the execution of the algorithm (or the cryptographic hardware block) under test. However, these EM traces should satisfy the some properties in order to properly compute $psdn_{x,y}$ and $incn_{x,y}$. These properties lead to some experimental rules:

1. $\underline{rules\ for\ psdn_{x,y}}$: n EM traces of the same signal s_1 with fixed input are required,
2. $\underline{rules\ for\ incn_{x,y}}$: n EM traces of signal s_1 with fixed input (s_{11}) and n EM traces of s_1 with a different set of inputs (s_{12}) are also required.

Thus in the case of our AES example, it means that n EM traces with same key and plaintext have to be acquired at each coordinate to compute $psdn_{x,y}$ values and n EM traces with different plaintext and/or keys are required to compute all $incn_{x,y}$ values. This leads to the experimental protocol Algorithm 1 an evaluator can follow to get the susceptible parts of an IC under test. It should be observed

Algorithm 1. EMFISC

```
    Input: $f_{CK}$, matrix of $s_{11}$ and $s_{12}$,
    $\alpha$ (% chip to keep),
    a (weight psd compared to incoherence)
    Output: $emfisc_{x,y}$
1: for X,Y positions do
2:     compute $psd_{s_1}(f)$
3:     compute $inc_{s_{11},s_{12}}(f)$
4: end for
5: $psdn_{x,y}$ and $incn_{x,y}$ = center reduce $psd_{x,y}$ and $inc_{x,y}$ population
6: remap $psdn_{x,y}$ and $incn_{x,y}$ population
7: compute $emdisc_{x,y} = \sqrt{(1-a) * psdn^2_{x,y} + a * incn^2_{x,y}}$
8: quantile($emfisc_{x,y}$,$\alpha$)
```

that for the sake of simplicity, the protocol is reduced to two signals s_{11}, s_{12} but can be easily extended to more signals. Then given the obtained EMFI susceptible zones, the evaluator can carry on by performing its EMFI campaigns on a reduced surface. The resulting time reduction can then be used to be more exhaustive on the other parameters involved in the settings of an EMFI (such as pulse amplitude or polarity, injection time, etc.) making the characterization more accurate.

5 Validation Protocol and Experimental Results

To demonstrate the efficiency of our methodology and the correctness of our EMFI susceptibility criterion, we applied it to two different devices. This section gives information about these testchips and the algorithm they executed during our tests. It also introduces metrics used to quantify the efficiency of our method and the experimental protocol we have followed.

5.1 Devices Under Test

To be as independent as possible of the nature of the testchips during our experimental validation campaign, two testchips were chosen. These testchips, which are two microcontrollers, were designed in two different CMOS technology nodes by two different founders. Both testchips have their clock signal generated by an internal clock signal generator based on a PLL and internal RC oscillator.

The first one (testchip1) features an ARM Cortex M4 core operating at 80 MHz, a Memory Protection Unit (MPU), a Floating Point Unit (FPU) and a Digital Signal Processor instructions. It also embeds 96 kb of SRAM, 32 kb of bootable RAM and 1 Mb of flash memory. This testchip, designed in a 90 nm process technology, has an area around 12 mm^2.

The second testchip (testchip2) is organized around a ARM cortex M3 core operating at 64 MHz. It also features a MPU but no FPU. However, it only embeds 64 kb of SRAM, 512 kb of Flash and does not have an FPU. This testchip, designed in the same process technology, has an area around 16 mm^2.

5.2 Algorithm Under Test

During all our experimental validations, EMFI have targeted Algorithm 2 that was executed by the two devices under test. As shown, Algorithm 2 mainly consists in reading a chosen word at a chosen address in a first SRAM (AddrSRAM32 in Algorithm 2), and then in writing the result of the reading in another SRAM (AddrSRAM96 in Algorithm 2) and finally in re-reading it to check if all operations have been performed correctly. In Algorithm 2, we can observe ADD instructions that are repeated 11 times before and after performing the reads and the write in the SRAM. This repetition was done to isolate our target from the rest of the code and be able to interpret what happens during an EMFI. However, the interpretation of the faults obtained during our experimentations

Algorithm 2. Pattern (AddrSRAM32, AddrSRAM96)

```
1: PUSH { lr }
2: ADD R0,R0,#0; 11 times
3: VLDR.F32 S0,[R0]; read SRAM32
4: VSTR.F32 S0,[R1]; write SRAM96
5: VLDR.F32 S1,[R1]; read back
6: ADD R0,R0,#0; 11 times
7: POP { pc }
```

falls out of the scope of this paper. It also acts as a hard-coded delay after the trigger signal delivered to the EMP generator. Finally, it can also be seen as syntactic sugar to ease the find of the algorithm in the EM traces. One can also notice that S0 and S1 registers are FPU registers and thus were changed by CPU registers for testchip2.

5.3 Figures of Merit

To quantify the efficiency of the proposed hotspot localization method, we defined two figures of merit. This solution was preferred to a visual approach consisting in comparing EMFISC maps with fault maps.

The first figure of merit we defined is the Coverage Rate (CR). It is the percentage of all considered coordinates (EM probe positions) above the IC surface leading to a fault that are discovered by our methodology, i.e. that falls in the quantile q_α. Of course, because CR depends on α, the evolution of CR with regards to this parameter has to be analyzed rather than particular values. If the hotspot localization method is efficient CR must remain high for high values of α.

The second figure of merit is the False Positive Rate (FPR) defined as the percentage of positions ranked in the category highly EMFI susceptible, i.e. falling in he quantile q_α that do not lead to a fault. Once again, the evolution of this figure of merit with regards to α is considered in the rest of the paper rather than a single value. If our localization methodology is correct FPR must be low for high values of α.

If these figures of merit give scores to measure the efficiency of the hotspot localization method, they can also give a score to any other localization method. In order to set a reference and be able to evaluate the performance of the EMFISC approach, we have choosen to compare it to a random approach consisting in selecting randomly the set of points as highly EM susceptible (in red on Figs. 4 and 5). We could have considered a smarter approach. However to the best of our knowledge there is none in the literature aiming at finding EMFI hotspots.

5.4 Experimental Validation Protocol

The experimental validation protocol has consisted in performing EM near field scan of testchips with an EMFI probe. The X and Y map steps were of 100 µm for both testchips. At each position, 1000 EM traces corresponding to the execution of target algorithm with specific input were acquired with a sampling rate equal to 10 GS/s for testchip1 and 1 GS/s testchip2. This relatively high number of measurements at each position was imposed by the presence of noise generated by the analog part of these circuits. Collected traces were then gathered in small set of traces to generate a reduced set of median traces. In the absence of such noise a significantly lower number of traces would have be sufficient by position. So without noise due to analog parts those acquirement have been reduced to 1350 curves divided in 450 acquirement for different sets of value and different addresses to read/write from. After the EM near field scans, EMFI maps were performed. They were drawn with different pulse amplitudes using the same EMFI probe than the one used to perform the near field scans. The X and Y displacement steps for EMFI maps were fixed to the same values as EM near field scans. For testchip1 (testchip2) pulse amplitudes of ±50 V and ±130 V (±198 V) were considered.

5.5 Experimental Results

Figures 4 and 5 give the evolution of the defined figures of merit with regards to α in % for our two testchips submitted to EMFI. The first panel of these figures reports an averaged EM analysis traces showing the time windows on which the $EMFISC_{x,y}$ values are computed. The second panel gives the evolution of CR with regards to α for $a = 0.25, 0.5, 0.75$.

As shown, if we do not privilege any guideline and thus consider $a = \frac{1}{2}$, the CR remains higher than 80% for $\alpha < 50\%$ (and $\alpha < 60\%$) for testchip1 (testchip2) which is a value significantly better than those obtained with a random selection of points. This means that more than 80% of positions leading to faults are captured by our method while keeping only 50% of the IC surface for testing. For $\alpha = 75\%$, CR remains higher than 60% values which is two to three times (depending on the considered testchip) the values obtained with a random selection of points (for $a = \frac{1}{2}$). This demonstrates the soundness of the proposed approach.

Regarding the FPR, one can observe that from $\alpha = 50\%$ and $a = 0.5$, its value is close to 80%. This value is significantly better than that (90%) obtained with a random selection of points. One can also be surprised by such high FPR values. However such values are normal because the number of positions leading to a fault represent 20% for testchip1 (11% for testchip2) of the IC surface. One can finally observe that the FPR decreases while α increases. This decrease demonstrates the soundness of EMFISC based method because we reject more points that do not lead to a fault than points leading to faults while increasing α.

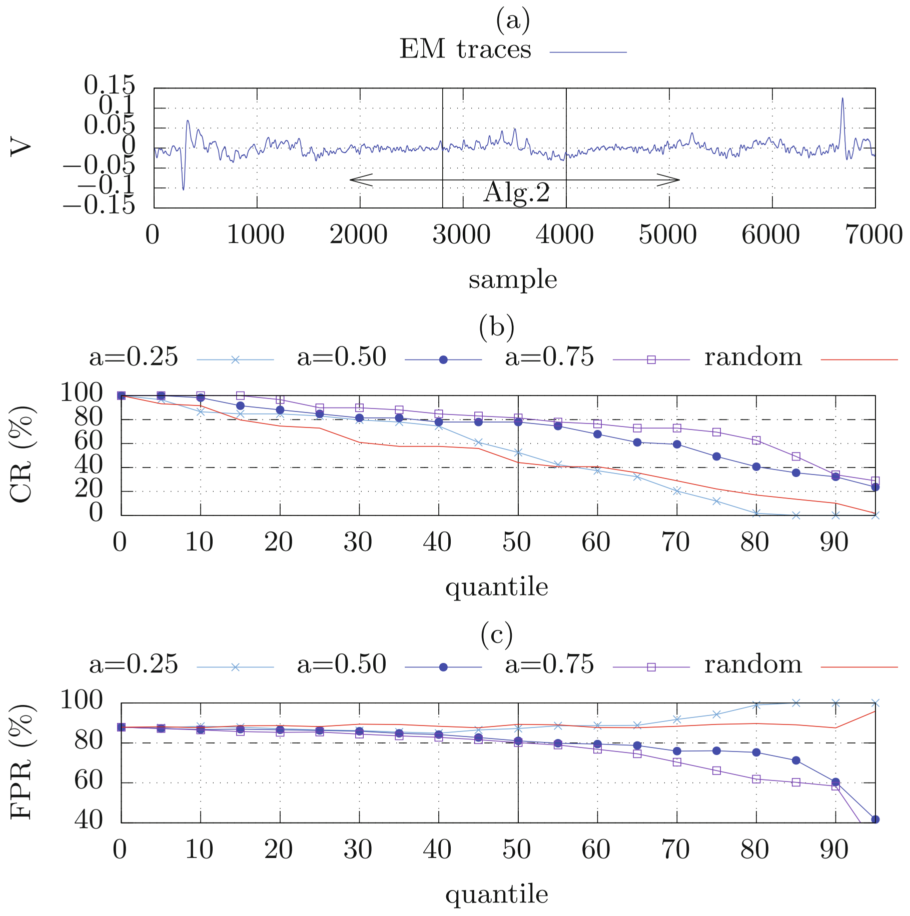


Fig. 4. Testchip 1 (pulse amplitude ±130 V): (a) averaged EM traces at a given position, (b) evolution of CR with regards to α for $a = 0.25, 0.5, 0.75$ (c) evolution of FPR with regards to α for $a = 0.25, 0.5, 0.75$, red curve = randomly selected point over the IC. (Color figure online)

Finally one can observe that CR and FPR values are much better for $a = 0.25$ for both testchips. This indicates that antennas with incoherent EM emissions must be privileged over antennas with strong but coherent EM emissions related to the clock signal. This suggests that EMFI induces more easily faults on the datapath of ICs than on the clock tree of processors (glue logic). However this points must be further investigated to definitively conclude.

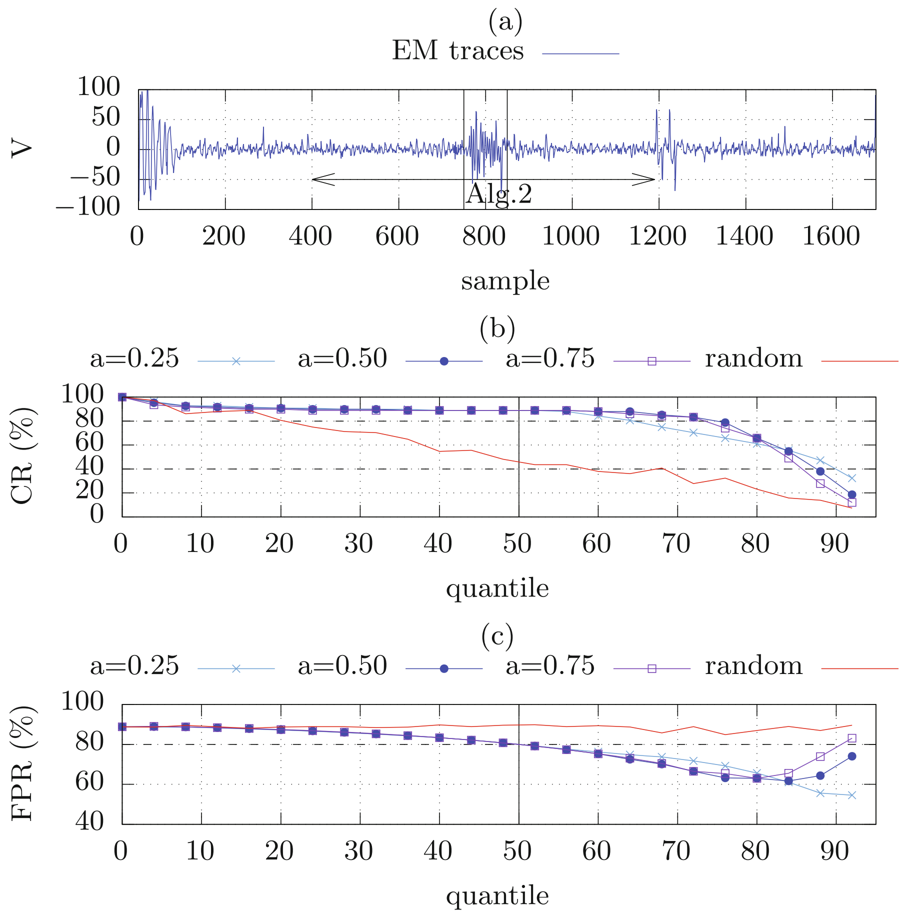


Fig. 5. Testchip 2 (pulse amplitude ±198 V): (a) averaged EM traces at a given position, (b) evolution of CR with regards to α for $a = 0.25, 0.5, 0.75$ (c) evolution of FPR with regards to α for $a = 0.25, 0.5, 0.75$, red curve = randomly selected point over the IC (Color figure online)

6 Conclusion

In this paper a criterion to rank position on a IC according to their electromagnetic fault injection susceptibility relying on both antennas reversibility and EM near field scans has been introduced. Moreover metrics to measure the efficiency of the method as well as to be able to compare future criterion on the same basis has also been introduced.

Dedicated to pulsed EMFI, it has been deduced from the sampling fault model introduced in former works. This criterion has been used to define a method allowing to rationally choose points of ICs surface to be tested against EMFI. Such method was defined in order to ease the security characterisation

of ICs which is time consuming as soon as it involves the evaluation of the robustness to fault attacks and especially EMFI. It allows rejecting more than 50% of the ICs surface while missing only few EMFI hotspots, i.e. while detecting more than 80% of positions at which faults where obtained.
Hence it affords to perform many tests with different EMFI parameters at these positions and therefore a more complete security characterisation in the smart-card context.

Furthermore this method could also be extended to complex systems on chip characterized by large surfaces or systems in package as shown in the experimental part of this paper. The criterion could also be extended by fine tuning it to the DUT and EMFI effect on circuits (probe coupling, ...). For instance, Sampling Fault Model is by nature a quite general model that relies on a more fundamental assumption about the effect of EMFI on the circuit. Thus the idea to add a restriction along with *psd* and *incoherence* that is bind to the physical effect of EMFI and more DUT specific should enhance the accuracy of EMFISC.

References

1. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. IACR Cryptology ePrint Archive, 2004:100 (2004)
2. Bayon, P., Bossuet, L., Aubert, A., Fischer, V., Poucheret, F., Robisson, B., Maurine, P.: Contactless electromagnetic active attack on ring oscillator based true random number generator. In: COSADE, pp. 151–166 (2012)
3. Dehbaoui, A., Dutertre, J.-M., Robisson, B., Orsatelli, P., Maurine, P., Tria, A.: Injection of transient faults using electromagnetic pulses - practical results on a cryptographic system. IACR Cryptology ePrint Archive, 2012:123 (2012)
4. Dehbaoui, A., Dutertre, J.-M., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementations of AES. In: FDTC, pp. 7–15 (2012)
5. Dehbaoui, A., Lomné, V., Ordas, T., Torres, L., Robert, M., Maurine, P.: Enhancing electromagnetic analysis using magnitude squared incoherence. IEEE Trans. VLSI Syst. **20**(3), 573–577 (2012)
6. El-Baze, D., Rigaud, J.-B., Maurine, P.: An embedded digital sensor against EM and BB fault injection. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, 16 August 2016, pp. 78–86 (2016)
7. Gruss, D., Maurice, C., Mangard, S.: Rowhammer.js: a remote software-induced fault attack in javascript. In: Detection of Intrusions and Malware, and Vulnerability Assessment - 13th International Conference, DIMVA 2016, Proceedings, San Sebastián, Spain, 7–8 July 2016, pp. 300–321 (2016)
8. Ordas, S., Guillaume-Sage, L., Maurine, P.: Electromagnetic fault injection: the curse of flip-flops. J. Cryptographic Eng., 1–15 (2016)
9. Ordas, S., Guillaume-Sage, L., Maurine, P.: EM injection: fault model and locality. In: 2015 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2015, Saint Malo, France, 13 September 2015, pp. 3–13 (2015)

10. Ordas, S., Guillaume-Sage, L., Tobich, K., Dutertre, J.-M., Maurine, P.: Evidence of a larger em-induced fault model. In: Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, 5–7 November 2014, pp. 245–259 (2014). Revised Selected Papers
11. Park, K., Lim, C.S., Yun, D., Baeg, S.: Experiments and root cause analysis for active-precharge hammering fault in DDR3 SDRAM under 3× nm technology. Microelectron. Reliab. **57**, 39–46 (2016)
12. Piret, G., Quisquater, J.-J.: A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Proceedings, Cologne, Germany, 8–10 September 2003, pp. 77–88 (2003)
13. Poucheret, F., Tobich, K., Lisart, M., Chusseau, L., Robisson, B., Maurine, P.: Local and direct EM injection of power into CMOS integrated circuits. In: FDTC, pp. 100–104 (2011)
14. Quisquater, J.J., Samyde, D.: Eddy current for magnetic analysis with active sensor. In: Proceedings of ESmart 2002, Eurosmart, pp. 185–194 (2002)
15. Schmidt, J.-M., Hutter, M: Optical and EM fault-attacks on CRT-based RSA: concrete results. In: Wolkerstorfer, J., Posch, K.C., (eds.) 15th Austrian Workhop on Microelectronics, Austrochip 2007, Proceedings, Graz, Austria, 11 October 2007, pp. 61–67. Verlag der Technischen Universität Graz (2007)
16. Tobich, K., Maurine, P., Liardet, P.-Y., Lisart, M., Ordas, T.: Voltage spikes on the substrate to obtain timing faults. In: DSD, pp. 483–486 (2013)
17. Zussa, L., Dehbaoui, A., Tobich, K., Dutertre, J.-M., Maurine, P., Guillaume-Sage, L., Clédière, J., Tria, A.: Efficiency of a glitch detector against electromagnetic fault injection. In: DATE, pp. 1–6 (2014)

Fault Analysis of the ChaCha and Salsa Families of Stream Ciphers

Arthur Beckers(✉), Benedikt Gierlichs, and Ingrid Verbauwhede

imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
{arthur.beckers,benedikt.gierlichs,ingrid.verbauwhede}@esat.kuleuven.be

Abstract. We present a fault analysis study of the ChaCha and Salsa families of stream ciphers. We first show that attacks like differential fault analysis that are common in the block cipher setting are not applicable against these families of stream ciphers. Then we propose two novel fault attacks that can be used against any variant of the ciphers. We base our attacks on two different fault models: the stuck-at fault model and the biased fault model. Each of them is exploited differently by the attacker. If the attacker knows the plaintexts and the ciphertexts both fault models can be successfully exploited. If the ciphers operate on fixed yet unknown plaintexts only the biased fault model can be successfully exploited. We evaluate exemplary attacks using both models in simulation. Their low complexity confirms that they are practical. To the best of our knowledge these are the first fault attacks against ChaCha and Salsa that do not require faults in the control flow (e.g. instruction skip).

Keywords: ChaCha · Salsa · Stream cipher · Fault analysis

1 Introduction

We analyse fault attacks that target implementations of the Salsa [1] and ChaCha [2] families of stream ciphers in all their variants. Both stream cipher families are designed by Dan J. Bernstein. The Salsa family of stream ciphers was designed in 2005 for the eSTREAM project. In 2009 Salsa20/12 was selected as one of the stream ciphers for the eSTREAM portfolio [3]. In 2008 the ChaCha family was proposed as a variant on the Salsa family. The goal of ChaCha was to increase the diffusion within a single round while maintaining the same performance as the Salsa family of ciphers. ChaCha is becoming more widely deployed since Google implemented it in its openSSL cipher suite [4]. Salsa20 and ChaCha support both 128- and 256-bit keys. We focus on the ChaCha and Salsa ciphers but the proposed techniques can be applied to every cryptographic function that has the same feed-forward and modular addition structure.

Fault attacks were introduced by Boneh et al. [5] in 1996 on RSA signature generation with CRT. Since then numerous attacks on many cryptographic primitives were published. A fault attack can be split up in three components. The injection method, the resulting fault model and the exploitation of the fault.

T. Eisenbarth and Y. Teglia (Eds.): CARDIS 2017, LNCS 10728, pp. 196–212, 2018.
https://doi.org/10.1007/978-3-319-75208-2_12

The most common injection methods include introducing glitches in the clock [6,7] or power supply [8], laser fault injection [9,10] and EM fault injection [11,12]. Each of these methods will result in a specific fault model that will strongly depend on the targeted device, making it hard to predict the outcome of a certain fault injection method. For our attacks we assume that a fault injection method is capable of delivering the fault model that we need and build up our attack from there.

The proposed attacks do not require fault injection in the control flow of the cipher [25], but they aim to exploit faults injected in intermediate values of the ciphers operation.

1.1 Background

Although ChaCha and Salsa are software oriented stream ciphers their operation resembles that of a block cipher in counter mode. Therefore we look at common fault analysis techniques for both block and stream cipher implementations.

Most stream ciphers have an initial state that holds the key and an initialization vector (IV). In an initialisation phase the initial state is randomized such that the key stream resulting from the internal state can not be used to recover the initial state. After the initialization phase a key stream is produced starting from the randomized state by updating the state continuously. For the randomization stream ciphers often use feedback shift registers (FSRs).

Fault attacks on stream cipher implementations generally try to recover the initial state by either targeting the initialization phase [13,14] or by forcing the cipher in an unintended state [15,16]. Often cryptanalytic techniques are applied to the gathered faulted outputs in order to recover the initial state.

The ChaCha and Salsa families of stream ciphers do not rely on FSRs for their operation. They instead rely mainly on a hash function to generate a pseudorandom key stream. Therefore common fault attack techniques for stream ciphers are not applicable to ChaCha and Salsa.

Common fault attacks on block cipher implementations are differential fault analysis (DFA) [17,18], collision fault analysis (CFA) [19], ineffective fault analysis (IFA) [20] and safe-error analysis (SEA) [21]. All these methods require multiple encryptions of chosen messages under a fixed key. In the stream cipher setting this would be possible only if the adversary was able to re-initialize the stream cipher with the same secret key, nonce and counter values. But of course this would be against any reasonable stream cipher specification, since in such a scenario also much more powerful cryptanalytic attacks would be possible.

Recently, statistical attacks against block cipher implementations have been proposed that do not require an adversary to have any control over the cipher's inputs [22–24]. They can be used to target block ciphers that operate for instance in counter mode. These attacks rely on fault injections in one of the last rounds of the cipher combined with statistical techniques to retrieve the key. The key retrieval is possible due to the addition of a round key in the last rounds. This key addition (XOR) results in an exploitable relationship between the faulted output and the key. ChaCha and Salsa, however, have a different structure and use a hash function and not a block cipher as their source of randomness making the attacks proposed in [22–24] not applicable.

2 The ChaCha and Salsa Ciphers

In this section we describe mainly the ChaCha family of stream ciphers since the different attacks are simulated using ChaCha. The design of Salsa differs only slightly, the general structure remains the same. We detail the differences at the end of this section, but they do not influence the proposed fault attacks.

2.1 ChaCha

The ChaCha stream cipher operates on a state matrix M that contains 16 words. Each word is 32 bits wide. M is initialized as shown in Eq. (1).

$$M = \begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix} \Leftarrow \begin{pmatrix} c & c & c & c \\ k_1 & k_2 & k_3 & k_4 \\ k_5 & k_6 & k_7 & k_8 \\ cnt & cnt & n & n \end{pmatrix} \tag{1}$$

The first four words denoted by c are constants given by the ChaCha specification. The next eight words are the key words denoted by $k_i (i = 1, \ldots, 8)$. The last row is made up out of two counter words cnt and two nonce words n. ChaCha works with both 128- and 256-bit key lengths. In the 256-bit case the key words are copied in order in $(k_1, \ldots, k_8)$ of M. In the 128-bit case the key words are copied in order in both the second and third row of M, $(k_1, \ldots, k_4) = (k_5, \ldots, k_8)$.

M is put through a hash function which has as its basic building block the quarter-round function (see Algorithm 1). The quarter-round function is made up out of ARX operations (addition modulo 2^{32}, rotation, exclusive-or) and is the source of non-linearity for the ChaCha stream cipher.

Algorithm 1. Quarter-round function

```
$(a, b, c, d) = quarterround(a, b, c, d)$
begin
    $a \leftarrow a \boxplus b$
    $d \leftarrow (a \oplus d) \lll 16$
    $c \leftarrow c \boxplus d$
    $b \leftarrow (c \oplus b) \lll 12$
    $a \leftarrow a \boxplus b$
    $d \leftarrow (a \oplus d) \lll 8$
    $c \leftarrow c \boxplus d$
    $b \leftarrow (c \oplus b) \lll 7$
    return $a, b, c, d$
end
```

One round of the hash function is built from four quarter-rounds. The number of rounds used in the hash function of ChaCha$r$ is a trade-off between security and efficiency. r is the number of rounds used in the hash function. The rounds of ChaCha$r$ are invertible therefore we need to modularly add M to the round output to form the hash. We focus on ChaCha20 but the proposed attacks are independent of the number of rounds used.

Figure 1 shows the general structure of ChaCha$r$. M is permuted using r rounds. After permuting M, the output of the round function (R) is modulo added to M which results in the key stream (K). K is xored with the plain text (PT) or cipher text (CT) for en- or decryption. After every hashing of M the counter cnt in M must be incremented (Eq. 1, [2]).

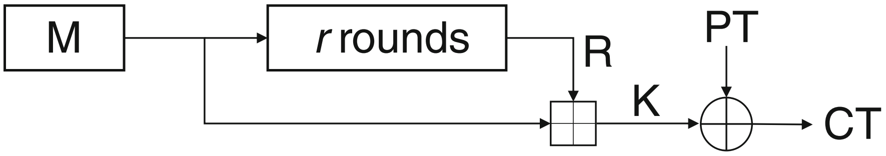


Fig. 1. Structure of ChaCha family of stream ciphers.

2.2 Salsa

Salsa differs in the structure of M and has a slightly different round function design. The Salsa state matrix differs from the ChaCha state matrix in the ordering of the key, constant, nonce and counter words. The rounds in the Salsa hash function use a different quarter-round function and order the inputs of their quarter-rounds differently. We refer to [1,2] for a detailed overview of the differences.

3 Fault Attacks on ChaCha and Salsa

We describe the attacks on ChaCha, application to Salsa is straightforward. The attacks are explained from an encryption point of view. Due to the symmetric nature of ChaCha an attack demonstrated on encryption is equally valid for decryption.

Our attacks aim to recover the secret key $k_1, \ldots, k_8$ by exploiting the modulo addition of the corresponding elements of M and R (m_i and $r_i (i = 4, \ldots, 11)$). In the rest of the paper we detail the attacks on one element of M (one key word k_i) but for readability we will write M, R and K without index. Application to the other key words is straightforward. For our attacks we assume the adversary has no control over M. The adversary is thus not capable of resetting the cipher to a previous state by manipulating the counter or nonce words. Note that M is constant for the relevant m_i, $i = 4, \ldots, 11$.

In normal operation R is uniformly distributed and therefore K does not reveal any information about M. The general idea for both attacks is to change R's uniform distribution into a distribution that can be exploited for key recovery.

Our attacks make use of two different distributions. Firstly the distribution of the values of R. Under normal operation R has a discrete uniform distribution, every value is equally likely to occur. Secondly the distribution of the Hamming weight of the values of R. The Hamming weight distribution of a discrete uniform distribution is a binomial distribution.

The distribution of R will change due to the fault injection. The resulting distribution of R and subsequently K and CT will depend on the type of fault. Throughout the remainder of the paper we will note faulted values with an asterisk (R^*, K^*, CT^*). To perform the proposed attacks an attacker has to make guesses for M, R^* and K^*. These guesses are denoted with an apostrophe (M', R', K').

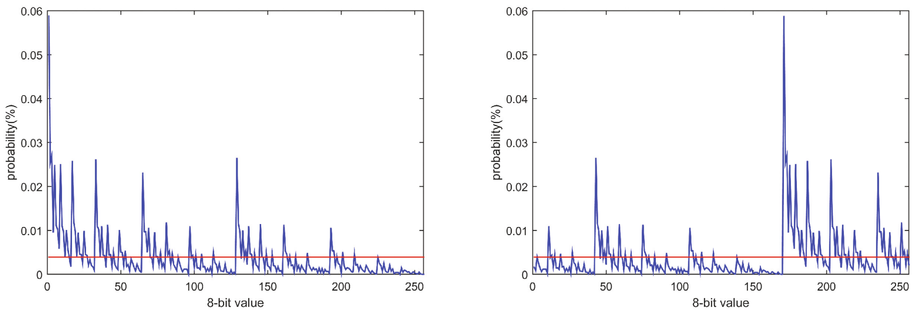
The location of the fault injection might be the registers storing the output of the round function or any arithmetic operation in one of the last quarter-round functions. We assume a fault is injected somewhere in the last round resulting in the desired fault model, but we do not make an assumption on which exact operation is faulted. It is *important* to note that we do not use the resulting value of any single fault but the changed *distribution* of R to retrieve the key.

3.1 Attack for the Biased Fault Model

R is a 32-bit word. The probability that any bit X in R is one or zero is equal: $P(X = 1) = P(X = 0) = 0.5$. With the biased fault model we assume that a fault injection alters these probabilities. The probability distribution could for instance become $P(X = 1) = 0.75, P(X = 0) = 0.25$ for every bit of R^*. This fault model has been used previously in [22,26,27]. The influence of such a bias on the distribution of an 8-bit word is illustrated in Fig. 2a. The biased distribution is plotted in blue, here $P(X = 1) = 0.35, P(X = 0) = 0.65$. The uniform distribution is plotted in red. The bias introduced on the bit level alters the distribution on the word level. We get a distribution that is skewed towards the lowest (in the example) or highest value of the distribution depending on the bias at bit level. The bias changes the distribution of Hamming weights on the word level too. It is *important* to note that the biased distribution on word level is the consequence of a bias introduced at bit level.

As explained in Sect. 2, R^* gets modulo added to M resulting in K^* (Fig. 1). Recall that we will recover the m_i one word at a time but do not write the index. The addition of R^* and a constant modulo a power of two merely rotates the biased distribution of R^*. Figure 2b illustrates the effect of modulo adding 170 to the biased distribution from Fig. 2a.

In order to retrieve the key we make use of the distribution of the Hamming weight of R^*. Note that this approach does *not* require any knowledge about the effect of any single fault injection or the amount of bias. It requires only that the probability distribution of the bits of R^* is biased and that the bias is equal



(a) Expected biased distribution (b) Rotated biased distribution

Fig. 2. Biased output distribution. (Color figure online)

for all the bits of R^*. We sample the biased distribution of K^* and subtract all possible key guesses from the key stream values in order to compute hypothetical R'. For each key guess, we look at the resulting distribution of Hamming weights of R'. The correct key guess will be the one for which the mean Hamming weight μ_{HW} of R' is maximal if $P(X = 1) > 0.5$ or minimal if $P(X = 1) < 0.5$.

The mean Hamming weight of a distribution can be calculated using Eq. (2). W denotes the values of a word of size n, $W = [0, \ldots, 2^n - 1]$. $P(w), w \in W$ is the probability that a value w of W occurs. Under a biased bit model this probability is given by Eq. (3). $Pr(q, n, p)$ (Eq. 4) is the binomial probability mass function and is used to calculate the probability that a certain HW occurs. p is the probability that a bit is 1 and thus p is equal to the amount of bias.

$$\mu_{HW} = \sum_{w=0}^{2^n-1} HW(w) \times P(w) \qquad w \in W \tag{2}$$

$$P(w) = \frac{Pr(HW(w), n, p)}{\binom{n}{HW(w)}} \qquad w \in W \tag{3}$$

$$Pr(HW, n, p) = \binom{n}{HW} p^{HW} (1-p)^{n-HW} \tag{4}$$

Equation (3) shows that the probability that a value w of W occurs depends solely on its HW and the bias p. If $p \neq 0.5$ the probabilities of $P(w)$ have an ascending or descending order depending on the Hamming weight and p as shown by Eqs. (5) and (6).

$$\frac{Pr(0, n, p)}{\binom{n}{0}} > \frac{Pr(1, n, p)}{\binom{n}{1}} > \ldots > \frac{Pr(n, n, p)}{\binom{n}{n}} \qquad \text{if } p < 0.5 \tag{5}$$

$$\frac{Pr(0, n, p)}{\binom{n}{0}} < \frac{Pr(1, n, p)}{\binom{n}{1}} < \ldots < \frac{Pr(n, n, p)}{\binom{n}{n}} \qquad \text{if } p > 0.5 \tag{6}$$

When we make a guess for the key we look at μ_{HW} of $R' = R^* \boxplus M \boxminus M'$ with M' our key guess and R' our resulting guess for R^*. We can distinguish the correct key guess since μ_{HW} is maximal for $p > 0.5$ or minimal for $p < 0.5$ if the distribution of the HW is binomial and thus if $R' = R^*$. The μ_{HW} is maximal for $p > 0.5$ since the probability that a value occurs rises with its Hamming weight (Eq. 6). If we permute the labels of a binomial distribution as it happens for a wrong key guess, for instance by swapping two elements with different Hamming weights, we inevitably assign a lower probability to the value with a higher Hamming weight and a higher probability to the value with a lower Hamming weight, see Eq. (7). i and j in Eq. (7) are the Hamming weights of two $w \in W$ with $i \neq j$.

$$\frac{Pr(i,n,p)}{\binom{n}{i}} \times i + \frac{Pr(j,n,p)}{\binom{n}{j}} \times j > \frac{Pr(i,n,p)}{\binom{n}{i}} \times j + \frac{Pr(j,n,p)}{\binom{n}{j}} \times i \tag{7}$$

The consequence of the swap will be a lowering of μ_{HW}. We provide an example in Table 1. A similar reasoning can be made for the case of $p < 0.5$.

Table 1. Example with $P(X = 1) = 0.7$ and $P(X = 0) = 0.3$.

value	00	01	10	11
HW	0	1	1	2
prob	0.09	0.21	0.21	0.49

$\mu_{HW} = 0.3725$

$\rightarrow$ swap 10 and 11 $\rightarrow$

value	00	01	11	10
HW	0	1	2	1
prob	0.09	0.21	0.21	0.49

$\mu_{HW} = 0.28$

We can thus distinguish a skewed binomial distribution ($p \neq 0.5$) from a skewed binomial distribution that is permuted. If the permutation however maps onto a skewed binomial distribution it will be indistinguishable from the original one.

An example is plotted in Fig. 3. The red curve shows the Hamming weight distribution if the 8-bit values are uniformly distributed, i.e. when there is no bias. The blue curve shows the Hamming weight distribution for the correct key guess and the grey curves show the Hamming weight distributions for all wrong key guesses. The mean of the Hamming weight of the distribution for the correct key guess lies farthest from the mean of the Hamming weight of the uniform distribution. Hence the correct key guess can be easily identified.

Different distinguishers can be used to identify the correct key guess. In [24] the authors propose to use Maximum likelihood, the mean Hamming weight or the Squared Euclidean Imbalance (SEI) as distinguishers. The Maximum likelihood approach requires an exact knowledge of the bias and is therefore discarded as an identifier. We will use the SEI and a standard two sample t-test (Eq. 8) as distinguishers. The use of the t-test is similar to the mean Hamming weight distinguisher.

For the standard two sample t-test we can compute the mean and standard deviation of the uniform distribution using Eq. (9). n is the word size in bits.

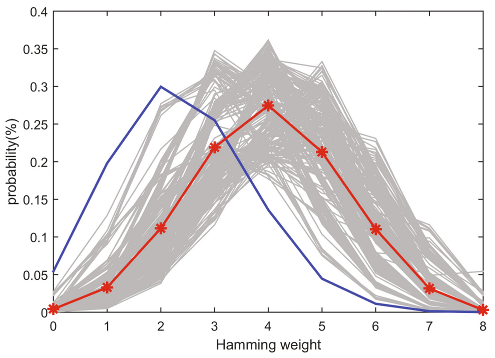


Fig. 3. Hamming weight distribution of K. (Color figure online)

z is the number of samples, s is the standard deviation of the Hamming weight of the sampled values and $\overline{x}$ is the mean Hamming weight of the sampled values.

$$t = \left| \frac{\mu - \overline{x}}{\sqrt{\frac{\sigma^2 + s^2}{z}}} \right| \tag{8}$$

$$\mu = \sum_{i=0}^{n} \left[\frac{\binom{n}{i}}{2^n} i \right], \sigma = \sqrt{\sum_{i=0}^{n} \left[\frac{\binom{n}{i}}{2^n} (i - \mu)^2 \right]} \tag{9}$$

In order to compute the SEI we use Eq. (10). $Pr(i, n, 0.5)$ is the binomial probability mass function from Eq. (4).

$$SEI = \sum_{i=0}^{n} \left[\frac{\#((HW)^z = i)}{z} - Pr(i, n, 0.5) \right]^2 \tag{10}$$

When performing an actual attack the sampling of the distribution becomes injecting a fault and collecting the faulted output. We can exploit the biased distribution at two positions either at K or at CT. When attacking at position K we need to know both PT and CT^* for the attack to succeed. When attacking at position CT we do not need to know PT but we require PT to be constant, since having a random unknown PT would be the equivalent of a one time pad making it impossible to retrieve the key.

Attack at Position *K*. Assume that all bits of K^* are equally biased. We split the 32-bit word K^* in four 8-bit chunks that all have a biased distribution. We do this to reduce the computational complexity. By splitting up the 32-bit word in h chunks we need to compute only $h2^{32/h}$ t-tests instead of 2^{32}. We are of course not restricted to a chunk size of 8-bits. The presence of a carry when splitting up a word will influence the bias but does not need to be accounted for by the attacker.

For every chunk we make a guess M' for the corresponding part of the key in M and compute $R' = PT \oplus CT^* \boxminus M' = K^* \boxminus M' = R^* \boxplus M \boxminus M'$. The only permutations performed on R^* are rotations due to the modulo additions. These rotations are not able to map R^* onto an R' such that R' has a probability distribution as described by Eq. (3) except for the one corresponding to the correct key guess. Therefore the known PT CT^* case will yield a single solution.

Attack at Position *CT*. We can perform a similar attack if we have a fixed unknown PT. The main difference is that we have to make a guess PT' on the value of PT too. We compute $R' = CT^* \oplus PT' \boxminus M' = R^* \boxplus M \oplus PT \oplus PT' \boxminus M'$, with PT' and M' our guesses for plaintext and key respectively.

The combination of xor operations and modulo additions maps R^* onto an R' that results in a binomial distribution of the Hamming weight in four cases. The attack will thus recover the correct guess and three false positives. If $PT \oplus PT' = 0$ we have the correct key guess, if $PT \oplus PT' = 2^n/2$ we get a false positive that has a distribution that is skewed according to the same bias as R^*. If $PT \oplus PT' = 2^n - 1$ or $PT \oplus PT' = 2^n/2 - 1$ we get a distribution for R' that is skewed in the opposite direction ($P(X = 1) = 1 - \text{bias}$). Recall that n is the size of the chunks in which we split the 32-bit word.

We can group the four key candidates in two sets by looking at the means of the probability distribution of the Hamming weight. A set with the bits biased towards zero and a set with the bits biased towards one. If we split up all the 16 words in 8-bit chunks we end up with 2×2^{32} key candidates for a 256-bit key. This reduction of the key space is sufficient to find the 256-bit key by exhaustive search.

3.2 Attack for the Stuck-At Fault Model

The stuck-at fault model assumes that some bits of R^* (see Fig. 1) are fixed to either one or zero. Due to the fixed bits the set of values R^* can attain will be limited. We look at a single 8-bit chunk of a 32-bit word R^*. Due to the fixed value of some of the bits certain values of R will no longer appear. Figure 4 shows a possible distribution for an 8-bit word with two bits stuck at zero. If we have enough faulty outputs we can compute the histogram of this distribution and deduce the amount of bits that are stuck at zero by counting the empty bins. The number of faults we need is determined by the number of bits that are stuck. For instance if we fault an 8-bit chunk and two bits get stuck at zero then there are 2^6 possible values left. In order to make a correct guess for the number of bits that are stuck we need to draw sufficiently many samples in order to fill the remaining bins.

The location of the empty bins in the histogram allows us to reduce the key space. We can again exploit the effect at two positions, either at K or at CT.

As with the biased fault model $K = M \boxplus R$. This modular addition modulo a power of two again only rotates the distribution. In order to reduce the key space we subtract our key guess modularly from K^* and obtain a hypothetical R'. Based on our reduction criteria we keep or discard the key guess. By iterating over all possible key guesses we are able reduce the key space.

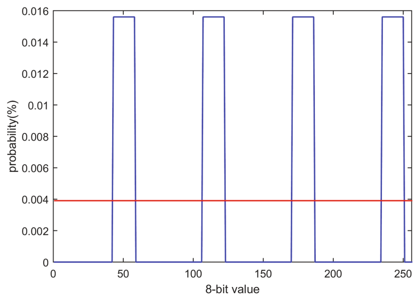


Fig. 4. Stuck-at fault model output distribution.

When reducing the key space we use two reduction criteria. The first one is the distribution of the Hamming weight of the words of R' for the different key guesses and the second is the location of the stuck bits. For the first reduction criterion we look at the Hamming weight distribution for a certain key guess. There should not be any Hamming weight values higher than the number of bits that are not stuck (in the case of a stuck-at 0 model) or a Hamming weight value lower than the number of stuck bits (in the case of a stuck-at 1 model).

From the histogram of the faulty values (K^* or CT^*) we can determine the number of bits that are stuck by counting the empty bins. Besides the number of empty bins we also need to know whether the bits get stuck at either 0 or 1 since we cannot deduce this from CT^* or K^*.

For the second criterion we look at the consistency of the location of the stuck bits in R'. If the distribution that results from our key guess does not have a fixed position for its stuck bits we can discard the key guess. The number of key candidates that remain after applying the reduction criteria depends on the number of bits that are not stuck and on their location.

As an example of the application of the reduction criteria we consider the following case. We have the distribution of a 4-bit chunk of R^* that has 2 bits stuck at 0, thus only 4 bins of the distribution of R^* are filled. We look at K^* and make three guesses for the key. As a result of these guesses we get three distributions R' which have the following bins filled, $(11, 6, 1, 4)$, $(5, 6, 4, 12)$ and $(8, 12, 4, 0)$. If we convert these to binary we get, $(1011, 0110, 0001, 0100)$, $(0101, 0110, 0100, 1100)$ and $(1000, 1100, 0100, 0000)$. The first key guess gets rejected since it fails the first reduction criteria, 11 has a Hamming weight of three while the maximum Hamming weight is two. The second key guess passes the first reduction criteria but fails the second. For the second key guess the location of the stuck bits is not consistent since there are three bit positions that have both a 1 and a 0, bit one, three and four. The last key guess results in a distribution R' that fits both criteria.

The exploitation of the stuck-at model is deterministic in nature. Using our reduction criteria we are certain that we do not discard valid key guesses. When we use a fixed but unknown PT with the stuck-at model we have to guess the values of PT and M.

Guessing full 32-bit words would require too many faulty outputs since we have to fill the entire histogram. Therefore we split the words of K^* or CT^* into smaller chunks. The chunks are not completely independent since there might be a carry propagating from one chunk to the other. The presence of a carry might reduce the efficiency of the attack. The carry bit will potentially increase the Hamming weight observed by the attacker by one. The attacker can not and does not have to account for this since the attack remains deterministic in the presence of a carry.

The number of faults needed depends on the number of bits that are stuck and the number of chunks we divide the 32-bit words in. The probability that any bin that will not stay empty due to stuck bits has at least one value is given by Eq. (11). s is the number of possible output values (2 to the power of the number of non stuck bits), f is the number of faults and b is the probability that every bin in the histogram that will not stay empty due to stuck bits has at least one value. We can simplify this equation to Eq. (11) in order to make an estimate of the number of faults needed to fill the histogram. Table 2 shows an example. If we have an 8-bit chunk which has 4 non-stuck bits, then we need 149 faults to have a probability of 99.9% that there is at least one value in every bin of the histogram.

$$b = 1 - \left[\sum_{i=1}^{s-1} \binom{s}{i} \left(\frac{s-i}{s}\right)^{f} (-1)^{i+1}\right] \implies f = \frac{\log \frac{1-b}{s}}{\log \frac{s-1}{s}} \tag{11}$$

Table 2. Number of faults to fill the non-stuck bins (b = 99.9%)

# non-stuck bits	1	2	3	4	5	6	7	8
# faults	11	29	67	149	326	702	1500	3181

When dealing with larger words such as the 32-bit words used in ChaCha we need to split up the words in smaller chunks and apply the reduction criteria described above to each of the individual chunks. The simplest approach is dividing the 32-bit word in four 8-bit chunks and applying the reduction criteria to each chunk individually. An alternative approach is to divide the 32-bit word in overlapping chunks, applying the reduction criteria to every chunk. Each chunk will result in a subset of key candidates. By looking at the intersection of the different key sets we end up with a single set of key candidates for the 32-bit word. The sizes of the different chunks do not have to be constant and can be chosen by the attacker.

4 Simulation Results

For our simulations we assume that with each fault injection an entire 32-bit word is faulted according to the assumed fault model. We operate the cipher as described in the specification. For our simulations we use a random key and nonce for M. The counter is also incremented for every encryption thus we have a continuously varying and random R^*.

4.1 Biased Fault Model

For our simulation for both the known PT and fixed unknown PT case we set the bias of the bits at $P(X = 1) = 0.75, P(X = 0) = 0.25$. We split up the 32-bit word in 8-bit chunks and apply the technique described in Sect. 3.1 to each chunk.

In the following we report results for a single 8-bit chunk. In the simulated scenario where all bits of the word are equally affected by the faults the average number of faults needed to recover 8-bits or an entire word of the key are the same. Thus the total amount of faults needed to recover the key equals the number of faults needed to recover the key from an 8-bit chunk times the number of key words.

Figure 5 shows the result of the two sample t-test for the different key guesses in case we have a known PT. The correct key guess is depicted in black. The t-scores for wrong key guesses do not converge towards zero since the distribution of hypothetical R' based on a wrong key guess will not be uniform but just the biased distribution rotated over a number of positions equal to the difference between the wrong and the correct key guess. The distribution for a wrong key guess will therefore also be different from the Hamming weight distribution of a uniform distribution.

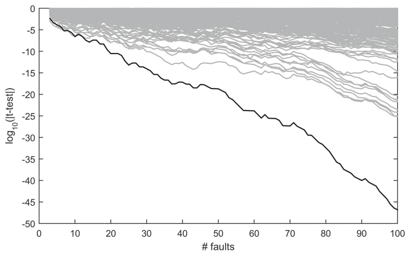


Fig. 5. Result of biased fault attack for known PT and CT^*; t-test values on y-axis in log scale, number of faults on x-axis.

The number of faulty outputs needed in order to retrieve the key is determined by the amount of bias there is on R^*. Table 3 shows the average number of faulty outputs needed for different biases and distinguishers. Once the distribution is biased the attack always succeeds given sufficiently many faulty outputs. We can see that the t-test consistently requires less faults than the SEI method to distinguish the correct key guess.

For the case where we have a fixed but unknown PT we have to make a guess for both the PT and M. We thus need more faults in order to distinguish the

Table 3. Bias vs. faults for known PT and CT^*

Bias	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%
# faults t-test	/	400	115	50	27	11	7	5	5	5	/
# faults SEI	/	570	173	80	46	35	31	26	36	78	/

correct guess. Table 4 shows the average number of faults needed. The needed number of faults goes back up if we have a bias > 85%. This is due to the large probability attributed to singular values with a high or low Hamming weight. If for a wrong key guess one of those values also maps to a value with a high or low Hamming weight it will take more faults to distinguish these wrong guesses with certainty from the correct one. When we use SEI as a distinguisher we need significantly more faults. When the bias becomes larger than 85% the number of faults becomes too large to be considered practical.

The result of the two sample t-test between the Hamming weight of one of the 8-bit chunks and the uniform distribution can be seen in Fig. 6. The four possible key, PT pairs are situated in the two lowest curves. Due to the false positive and mirrored results we have to do a brute-force search over the remaining key candidates. As for the known PT case the number faulty outputs needed to retrieve the key will depend on the amount of bias.

Table 4. Bias vs. faults for fixed but unknown PT

Bias	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%
# faults t-test	/	1356	420	158	100	77	65	63	78	134	/
# faults SEI	/	2235	548	367	329	517	877	11720	>20000	>20000	/

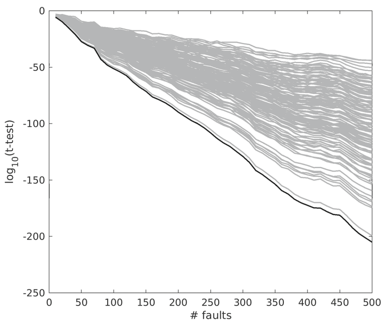


Fig. 6. Result of biased fault attack for fixed unknown PT; t-test values on y-axis in log scale, number of faults on x-axis.

4.2 Stuck at Fault Model

For the stuck at fault model we use the techniques described in Sect. 3.2 to reduce the key space. The average number of key bits we can retrieve depends on the number of bits that are stuck. Figure 7 shows the number of key bits we can recover in a 32-bit word versus the percentage of bits that are stuck. The number of faults needed for this attack is dependant on the number of stuck bits and the chosen chunk size and can be estimated using Eq. (11). Section 3.2 suggests two ways to apply the reduction criteria. We either split up the word in chunks and apply the reduction criteria on the individual chunks or we run the reduction criteria iteratively on all possible subsets of chunks. First we consider the known PT CT^* case. The green curve shows the average number of bits that can be recovered if we split the 32-bit word in individual 8-bit chunks. A better result, shown by the blue line, can be achieved if we apply the iterative algorithm. The brown curve shows the result for a fixed but unknown PT for individual 8-bit chunks. The curve for the iterative algorithm applied to a fixed but unknown PT is plotted in pink.

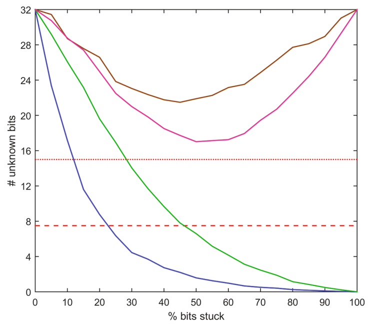


Fig. 7. Key space reduction by the stuck at attack. (Color figure online)

In the known PT CT^* case the number of unknown bits converges towards zero as the number of stuck bits increases. In the fixed but unknown PT case the number of unknown bits has a minimum when half of the bits are stuck. Intuitively this can be understood as follows. In the known PT CT^* case $K^* = M \boxplus R^*$. The distribution of R^* will get sparser, the histogram will have a lot of empty bins, if more bits are stuck. Thus the key words of M will be easier to recover the more bits are stuck at one or zero. In the fixed but unknown PT case $CT^* = (M \boxplus R^*) \oplus PT$. If the distribution of R^* is sparse then the distribution of CT will also be sparse. In order to retrieve the key words of M we have to iterate over all possible PT values. Iterating over all possible PT values equals making a guess for the distribution of K^* since $K^* = CT^* \oplus PT$. Since the distribution of CT^* is sparse there is a higher probability (increasing with the number of bits stuck once greater 50%) that a PT guess will result in a valid stuck at distribution that fits our reduction criteria. The key space will thus become larger as the number of stuck bits increases.

We assume an attacker can brute force a key space of 2^{60} and the key length is 256 bits. Under these assumptions we end up with an upper limit on the average number of remaining bits per word of 7.5 in order to make the key retrievable. This upper limit corresponds to the dashed red line in Fig. 7. In the 128-bit case we can have up to 15 unknown bits per word. This upper limit is depicted by the dotted red line. Due to the replication of the key in the 128-bit case we can also fault both copies of the key and compare the two sets of remaining key candidates. This allows us to reduce the remaining key space even more since the correct key guess has to be present in both candidate sets. In the known PT CT^* case we can always recover the key if we have enough stuck bits. In the fixed but unknown PT case we are able to reduce the number of unknown bits per word to 17 in the best case. For an 128-bit key this would require an attacker capable of brute-forcing a key space of 2^{68}.

Having all bits stuck at zero or one is a special case of the stuck at model. When we know both PT and CT^* retrieving the key is trivial since then $M = K^*$ or $M = K^* \boxplus 1$. When we have a fixed unknown PT however we are xoring two unknown constants making it impossible to retrieve the key.

5 Conclusion

We presented a fault analysis study of the ChaCha and Salsa families of stream ciphers. These software oriented stream ciphers gained popularity since ChaCha was included in google's openSSL cipher suite. We explained why classical fault analysis techniques are ineffective against these ciphers. Then we proposed two novel fault attacks using ChaCha as example. Targeting K proved to be effective and efficient for both the biased fault model and the stuck-at fault model. Attacks based on these models however assume an adversary that knows both PT and CT^*. In the fixed but unknown PT case only the attack based on the biased fault model will succeed when we attribute realistic computing powers to the adversary. The attack based on the stuck-at fault model is able to reduce the key space but not to identify the key uniquely. The proposed attacks all exploit a non-uniform distribution of R. An on-line randomness test that detects when the distribution of R^* differs significantly from uniform may be a possible countermeasure.

Acknowledgements. We thank the anonymous reviewers for their insightful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the Flemish Government, FWO G.00130.13N, FWO G.0876.14N and Thresholds G0842.13; by the Hercules Foundation AKUL/11/19, and through the Horizon 2020 research and innovation programme under Cathedral ERC Advanced Grant 695305. Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

1. Bernstein, D.J.: Salsa20. ECRYPT Stream Cipher Project, 025 edn. (2005). http://cr.yp.to/snuffle.html
2. Bernstein, D.J.: ChaCha, a variant of Salsa20. In: The State of the Art of Stream Ciphers, SASC 2008. ECRYPT (2008)
3. http://www.ecrypt.eu.org/stream/
4. Bursztein, E.: Speeding up and strengthening HTTPS connections for chrome on android. https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html
5. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. J. Cryptol. **14**(2), 101–119 (2001)
6. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Proc. IEEE **94**(2), 370–382 (2006)
7. Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 105–114, September 2011
8. Schmidt, J.M., Herbst, C.: A practical fault attack on square and multiply. In: 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 53–58, August 2008
9. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_2
10. Agoyan, M., Dutertre, J.M., Mirbaha, A.P., Naccache, D., Ribotta, A.L., Tria, A.: How to flip a bit? In: 2010 IEEE 16th International On-Line Testing Symposium, pp. 235–239, July 2010
11. Dehbaoui, A., Dutertre, J.M., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementations of AES. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 7–15, September 2012
12. Quisquater, J.J., Samyde, D.: Eddy current for magnetic analysis with active sensor. In: Esmart 2002, Nice, France, September 2002
13. Hojsík, M., Rudolf, B.: Floating fault analysis of trivium. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 239–250. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89754-5_19
14. Debraize, B., Corbella, I.M.: Fault analysis of the stream cipher snow 3G. In: 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 103–110, September 2009
15. Biham, E., Granboulan, L., Nguyên, P.Q.: Impossible fault analysis of RC4 and differential fault analysis of RC4. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 359–367. Springer, Heidelberg (2005). https://doi.org/10.1007/11502760_24
16. Hoch, J.J., Shamir, A.: Fault analysis of stream ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_18
17. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052259
18. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2004. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005). https://doi.org/10.1007/11506447_4

19. Blömer, J., Krummel, V.: Fault based collision attacks on AES. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 106–120. Springer, Heidelberg (2006). https://doi.org/10.1007/11889700_11
20. Blömer, J., Seifert, J.-P.: Fault based cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45126-6_12
21. Yen, S.M., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Trans. Comput. **49**(9), 967–970 (2000)
22. Lashermes, R., Reymond, G., Dutertre, J.M., Fournier, J., Robisson, B., Tria, A.: A DFA on AES based on the entropy of error distributions. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 34–43, September 2012
23. Dobraunig, C., Eichlseder, M., Korak, T., Lomné, V., Mendel, F.: Statistical fault attacks on nonce-based authenticated encryption schemes. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 369–395. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_14
24. Fuhr, T., Jaulmes, E., Lomné, V., Thillard, A.: Fault attacks on AES with faulty ciphertexts only. In: Proceedings of the 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2013, pp. 108–118. IEEE Computer Society (2013)
25. Kumar, S.V.D., Patranabis, S., Breier, J., Mukhopadhyay, D., Bhasin, S., Chattopadhyay, A., Baksi, A.: A practical fault attack on ARX-like ciphers with a case study on ChaCha20. Cryptology ePrint Archive, Report 2017/1074 (2017). https://eprint.iacr.org/2017/1074
26. Ghalaty, N.F., Yuce, B., Schaumont, P.: Analyzing the efficiency of biased-fault based attacks. IEEE Embed. Syst. Lett. **8**(2), 33–36 (2016)
27. Järvinen, K., Blondeau, C., Page, D., Tunstall, M.: Harnessing biased faults in attacks on ECC-based signature schemes. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 72–82, September 2012

Applying Horizontal Clustering Side-Channel Attacks on Embedded ECC Implementations

Erick Nascimento[1(✉)] and Łukasz Chmielewski[2]

[1] Institute of Computing, University of Campinas, Campinas, Brazil
enascimento.pub@gmail.com
[2] Riscure BV, Delft, The Netherlands
chmielewski@riscure.com

Abstract. Side-channel attacks are a threat to cryptographic algorithms running on embedded devices. Public-key cryptosystems, including elliptic curve cryptography (ECC), are particularly vulnerable because their private keys are usually long-term. Well known countermeasures like regularity, projective coordinates and scalar randomization, among others, are used to harden implementations against common side-channel attacks like DPA.

Horizontal clustering attacks can theoretically overcome these countermeasures by attacking individual side-channel traces. In practice horizontal attacks have been applied to overcome protected ECC implementations on FPGAs. However, it has not been known yet whether such attacks can be applied to protected implementations working on embedded devices, especially in a non-profiled setting.

In this paper we mount non-profiled horizontal clustering attacks on two protected implementations of the Montgomery Ladder on Curve25519 available in the μNaCl library targeting electromagnetic (EM) emanations. The first implementation performs the conditional swap (cswap) operation through arithmetic of field elements (cswap-arith), while the second does so by swapping the pointers (cswap-pointer). They run on a 32-bit ARM Cortex-M4F core.

Our best attack has success rates of 97.64% and 99.60% for cswap-arith and cswap-pointer, respectively. This means that at most 6 and 2 bits are incorrectly recovered, and therefore, a subsequent brute-force can fix them in reasonable time. Furthermore, our horizontal clustering framework used for the aforementioned attacks can be applied against other protected implementations.

Keywords: ECC · EM analysis · ARM · Horizontal clustering

1 Introduction

Public-key cryptosystems based on ECC [23,28] are frequently used in a wide range of applications, such as: credit card, e-commerce and cryptocurrency.

E. Nascimento—This work was partially done by the author in a research internship at Riscure BV.

Running on embedded systems, they are a common target of side-channel attacks. The main goal of these attacks is to recover the private key, which is typically the scalar in a scalar multiplication – the main ECC operation in most protocols.

Horizontal attack (HA) is a methodology for side-channel attacks against basic cryptographic operations in protocols based on RSA or ECC, the modular exponentiation and the scalar multiplication (ECSM), respectively. In theory, a horizontal attack against ECC allows the recovery of secret scalar bits through the analysis of individual traces, i.e., a single trace from the actual target is sufficient; thus, they are effective against implementations protected by classic and popular countermeasures such as scalar randomization (SR), coordinate randomization (CR), point blinding and scalar splitting. A fundamental requirement for an attacker to apply HA is the knowledge of the scalar multiplication algorithm; implementation details, however, are not required. In addition, HA requires to have a good comparison tool, thereafter referred to as a *distinguisher*, to efficiently extract parts of the keys. The following methods can be applied, among others: *correlation, collision-correlation, cross-correlation* and *cluster analysis*.

The correlation analysis method [5] follows the same principle as correlation power analysis (CPA) applied to a set of traces arranged vertically. The difference in the horizontal context is that a single trace is divided in several segments and a hypothetical intermediate value is assigned to each segment, based on a guess about the key value. The correlation between the segment samples and hypothetical values is computed in the same way as in CPA. This method works against implementations protected only with scalar randomization, or when coordinate randomization is applied with a short random parameter. The method of collision-correlation analysis [1,2,4,39,41] computes the correlation or Euclidean distance between segments of a trace. The goal is to identify the occurrence of the same intermediate data in different parts of the trace, and by doing so derive the secret bits. In theory, this method is feasible against the classic countermeasures.

Many side-channel attacks do not work when a stronger version of coordinate randomization is used, the so-called coordinate re-randomization (CRR) [30]. This countermeasure randomizes the working points coordinates at every ECSM iteration. Therefore, the correlation or collision-correlation attacks are prevented, because they rely on the fact that output points of an iteration are equal to the input points of the next iteration. On the other hand, attacks that target iterations independently, like the attacks presented in this paper, are not influenced by this countermeasure. The implementations attacked in this paper are protected with all the aforementioned countermeasures.

Most horizontal attacks require advanced preprocessing of traces, characterization and leakage assessment before applying distinguishers. The main challenge of the horizontal approach revolves around extracting meaningful leakage from a single trace, which usually has strong noise. In this paper we consider the non-profiled scenario (also called unsupervised) in which the adversary does not know and cannot change the private key in any test device. Moreover, she is not allowed to turn off countermeasures[1]. Therefore, the second major challenge is caused by the unavailability of labeled samples. Note that leakage assessment methods, like TVLA [12], require labeled samples and this is not possible when scalar randomization is enforced.

Related Work. Unsupervised learning methods, especially those based on clustering, have been applied to solve the aforementioned limitations and they have been shown to be able to work in practice.

Heyszl et al. [14] apply multi-dimensional K-Means [11,25] clustering to successfully attack an FPGA-based ECC implementation by correctly classifying the scalar bits. Sprecht et al. [37] later improved this attack by using Expectation-Maximization clustering [7], Principal Component Analysis (PCA) [21] and multiple EM probes. Both methods target ECC implementations for FPGAs and work well for low noise measurements. In this paper we do not employ dimensionality reduction techniques such as PCA (unsupervised) or LDA [10] (supervised), but instead we apply a points of interest selection method.

Perin et al. [35], consider a heuristic approach based on unidimensional difference of means for points of interest selection. This method uses a single trace for the leakage assessment, which is likely affected by noise. Perin and Chmielewski [34] propose a methodology for clustering attacks to amend the aforementioned deficiency by using multiple unlabeled traces for leakage assessment and improving attack robustness in high noise scenarios. Similarly to the above works we use unidimensional clustering for points of interest selection; however, for the attack, we evaluate various clustering methods including the multi-dimensional one.

Jarvinen et al. [20] present an unsupervised clustering attack on m-ary ECSM with precomputations[2]. The proposed attack is evaluated using a low noise 8-bit AVR device. While our attacks target binary ECSM, they can be straightforwardly extended to the m-ary case: instead of using binary clustering, the attack would need to employ 2^m clusters. Most clustering algorithms support an arbitrary number of clusters, e.g. K-Means.

Another related work concerns error correction. In [14], to derive the error locations the authors use a probability for cluster belonging derived from the K-Means results. Essentially, scalar indexes with the lowest probability are brute-forced. Similarly, the papers [34,35] use probability density function for various clustering algorithms to perform error correction.

We have applied the approach from [34] to detect the errors in the recovered scalar. Unfortunately, this approach does not work for our experiments because of a presence of strong noise pulses in the EM traces. Essentially, some bit errors occur even if their probabilities of being correct are high. Therefore, we have abandoned this approach and applied a brute-force method sped up by the time-memory trade-off algorithm from [30].

[1] Turning off the countermeasures is not always possible in the ICC EMVCo smart card evaluations [9], for example.

[2] In an m-ary method, m bits of the scalar are processed in one iteration of ECSM while in a standard ECSM a single bit is processed per iteration.

Contributions. The main contributions of this paper are summarized below. First of all, using EM we perform a horizontal clustering attack (HCA) against the arithmetic-based cswap.[3] Curve25519 μNaCl Montgomery Ladder running on a 32-bit ARM Cortex-M4F. The implementation is additionally protected with projective coordinate re-randomization and scalar randomization. We compare a wide range of leakage assessment methods and statistical classifiers to find out the best settings and we achieve the best success rate of 97.64%[4]. This means there are at most 6 erroneous bits, which can be brute-forced in a reasonable time even without knowing error locations.

Secondly, we attack the pointer-based cswap μNaCl Montgomery Ladder implementation that is protected the same as for the first one. Our best attack on this implementation has a success rate of 99.60%, i.e., only 2 errors.

Note that in our non-profiled approach, choosing the best settings implies running the attack for all the considered parameters. This would significantly increase the attack time. To partially mitigate this issue, we use the same settings for both implementations. Moreover, the attack can be easily parallelized, for example, by using cloud computing; significantly improved results justify this.

Thirdly, we improve the unsupervised RSA HCA framework from [34]. This framework implements the leakage assessment by combining multiple RSA traces protected with exponent blinding. We extend that framework to ECC by attacking a randomized scalar instead of a blinded exponent. In addition, we propose the usage of: (i) multiple dimensions clustering; (ii) methods for outlier detection; and (iii) intrinsic quality evaluation of clusters.

Finally, we generalize the method from [30] to tolerate a certain number of incorrectly recovered scalar bits without relying on confidence probabilities.

Our attacks demonstrate the feasibility of scalar recovery from the μNaCl-based ECSM. Breaking ECSM implies that an attacker can compromise the key exchange protocols: Elliptic Curve Diffie-Hellman (ECDH) and its ephemeral version (ECDHE). Examples of current publicly known applications using μNaCl on ARM Cortex-M devices, and thus potentially vulnerable, include: [8,36,40].

Paper Organization. The remainder of this paper is structured as follows. In Sect. 2, we describe the setup of the attacks. Subsequently, Sect. 3 covers preliminaries and Sect. 4 presents our horizontal cluster framework. The experimental results are shown in Sect. 5. We describe how to efficiently correct errors in Sect. 6. Finally, Sect. 7 discusses countermeasures and future work.

2 Attack Setup

2.1 Target Software Implementations

We target μNaCl[5], a cryptographic library for ARM Cortex-M that provides implementations of Curve25519, an elliptic curve at the 128-bit security level

[3] Cswap means conditional swap. In a Montgomery ladder ECSM, the cswap condition value tells whether or not to swap and it depends on the secret scalar bit. Thus, it should ideally be constant time and not leak through other side channels.

[4] We use the term *success rate* to refer to the percentage of correctly recovered bits.

[5] http://munacl.cryptojedi.org/curve25519-cortexm0.shtml.

Algorithm 1. Montgomery ladder with cswap and coordinate re-randomization.

```
// ... initialization omitted ..
$bprev \leftarrow 0$
for $i = 254 \ldots 0$ do
    RE_RANDOMIZE_COORDS($work\_state$)
    $b \leftarrow$ bit $i$ of scalar
    $s \leftarrow b \oplus bprev$
    $bprev \leftarrow b$
    CSWAP($work\_state$, $s$)
    LADDERSTEP($work\_state$)
end for
// ... return ommited ..
```

and its associated X25519 key exchange protocol based on Diffie-Hellman. This library provides two ECSM implementations, both based on the Montgomery ladder algorithm. They differ on how the conditional swap (cswap) operation, fundamental to implement it in constant time, is performed: either by arithmetic means (cswap-arith) or pointers swapping (cswap-pointer).

At each algorithm iteration, the cswap condition depends on the secret scalar bit processed at that iteration and thus its value should not leak. We argue that in both implementations the cswap condition value leaks. We investigate and confirm that the leakage is strong enough to be exploited by our proposed attacks.

In the cswap-arith implementation, the if/else branch is replaced by conditional swaps of the respective coordinate values of the working points, $P_1 = (X_1, Z_1)$ and $P_2 = (X_2, Z_2)$, to achieve constant time. A high level description of such strategy is described in Algorithm 1. Another cswap implementation performs a conditional swap of pointers to the field elements instead (cswap-pointer)[6]. In the latter implementation, during each ECSM iteration, the mask is touched far fewer times by the `AND` (`&`) instruction (3 times) than in the cswap-arith (16 times); thus, in theory, a weaker side-channel leakage is expected.

The ECSM implementations in μNaCl do not provide countermeasures against power/EM analysis, besides a regular and constant-time implementation. To evaluate our proposed attacks against properly protected targets, we added coordinate re-randomization to both implementations[7]. The re-randomization countermeasure multiplies a randomly generated $\lambda \in \mathbb{F}_p$ with the coordinates of P_1 and P_2 at the beginning of every ECSM iteration (Algorithm 1).

2.2 Target Device and Measurement Setup

The target software runs on the STM32F4 microcontroller chip on the board, with a 32-bit ARM-M4 CPU core, clocked at 168 MHz. We acquired electromagnetic (EM) traces from the ECSM execution by the target device, using a single

[6] Selected by preprocessor definition `DH_SWAP_BY_POINTERS`.

[7] Our attack also works against implementations protected with scalar randomization. We have not implemented this countermeasure, but instead we set a random scalar for each ECSM execution.

EM probe. The setup consisted of a Lecroy Waverunner 8254M oscilloscope, a Langer RF-U 2.5-2 H-field probe, an amplifier and analog low pass filter (250 MHz).

For the acquisition of each trace, the host PC sends to the target device a pair of scalar (k) and input point (P), both randomly generated. The device receives the pair and executes the scalar multiplication, returning the output point ($R = [k]P$) to the host PC. We have acquired the traces with the following settings: 2.5 GS/s sample rate, 16 mV amplitude and 70 million samples. We have also used a low pass BNC analog filter: BLP-250+ from Mini-Circuits. We acquired and analyzed traces using Riscure's Inspector software package.[8]

3 Preliminaries

3.1 Traces Characterization

The n-th measured side-channel trace, which represents the electromagnetic emanation (EM) of a device over the time domain, is denoted by the uni-dimensional ($1 \times aL$) vector $\mathbf{t}^n = \{O_1^n, O_2^n, ..., O_{aL}^n\}$. Here, we consider a trace $\mathbf{t}^n$ as being the side-channel information of an ECSM composed by a fixed number aL of iterations. The factor a depends on the ECSM algorithm and L is the bit-length of the scalar; for Montgomery Ladder on Curve25519, $a = 1$ and $L = 255$. The trace $\mathbf{t}^n$ can be described by a set of ℓ-sized sub-vectors:

$$\mathbf{t}^n = \{O_1^n, O_2^n, ..., O_{a.L}^n\} = \big\{(t_{1,1}^n, ..., t_{1,\ell}^n), (t_{2,1}^n, ..., t_{2,\ell}^n), ..., (t_{a.L,1}^n, ..., t_{a.L,\ell}^n)\big\}$$

where $t_{i,j}^n$ is the j-th element of each sub-vector O_i^n and ℓ is the number of samples. The element $t_{i,j}^n$ can be viewed as a sample in time from the side-channel trace $\mathbf{t}^n$. The set $\{t_{i,j}^n\}$, $i = 1..aL$, refers to a set of samples where each element $t_{i,j}^n$ is extracted from one ECSM iteration O_i^n for a fixed j. For example, $\{t_{i,10}^n\}$ contains aL samples, each element $t_{i,10}^n$ is selected from the 10^{th} sample of each sub-trace O_i).

A *traceset* is defined as the set of trace segments of one or more ECSM runs, and each trace segment consists of the samples from a single ECSM iteration. A *full traceset* is a set of traces of multiple ECSM runs, where each trace in the set consists of the samples from a full ECSM run, i.e., it is a contiguous trace containing all the samples from all iterations of that ECSM run. The tracesets are assumed to be unlabeled, except in those traces used for known-key analysis.

Trace Processing. Each full ECSM trace is cut into segments, one per each algorithm iteration. Since the considered ECSM implementation is based on Montgomery Ladder, each iteration i corresponds to the processing of a *swap* bit (s in Algorithm 1), which depends on the i-th scalar bit. The trace cutting was done by first applying a low pass filter and then detecting patterns that appeared repeatedly for 254 times using correlation with a specified threshold; the patterns were detected visually with ease. A simple time-based trace cutting did not work in our case, despite the constant time of the implementation, due

[8] http://www.riscure.com/.

to visible time drifts in the measured samples, probably due to clock drifting or measurement imprecisions.

After cutting, we align the traces using Pearson correlation on the pattern corresponding to cswap. We select the pattern based on the time of a single Montgomery ladder iteration and source code analysis.

3.2 Clustering

Clustering Algorithms. The clustering algorithms successfully employed so far in the context of horizontal attacks are K-Means (KM), Fuzzy K-Means (FKM) and Expectation-Maximization (EM) [14,20,34,35]. K-Means is a *rigid* clustering algorithm, meaning that each instance (a sample in the context of HCA) is assigned (labeled) to a single cluster. On the other hand, Fuzzy K-Means and EM are *soft* clustering algorithms, because their output includes an association probability matrix, where each instance is associated with its degree of linkage to each cluster.

Intrinsic Cluster Quality Measure. Given a set of clustering outputs from multiple clustering algorithm runs, an intrinsic cluster quality measure can be applied to evaluate the best among them. Such measures are called intrinsic or internal because they consider just the structure of the clusters, and do not take into account any labeled information that might be available and could be used for testing. A clustering result with the best intrinsic cluster quality measure does not guarantee the best results for the application (in this work, for use by cluster leakage assessment and horizontal cluster analysis). But, it is nevertheless useful when clustering results cannot be otherwise tested.

Several intrinsic quality measures have been proposed for unsupervised clustering, among them: Silhouette coefficient [22], Calinski-Harabaz index [3] and Davies-Bouldin (DB) index [6]. We chose to use the DB index, which is based on the ratio of within-cluster and between-cluster distances, and can be defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_{j \neq i}\{D_{i,j}\} \qquad (1) \qquad\qquad D_{i,j} = \frac{\bar{d_i} + \bar{d_j}}{d_{i,j}} \qquad (2)$$

where $D_{i,j}$ is the within-to-between cluster distance ratio for clusters i and j; $\bar{d_c}$ is the mean distance between the centroid of cluster c and each point in that cluster; and $d_{i,j}$ is the Euclidean distance between the centroids of clusters i and j. The smaller the DB index, the better is the clustering.

The DB index favors clusters that are compact and distant from each other. These are exactly the properties we expect to get at points in time where the clusters provide a good separation of the classes (two classes, one for each possible value of the swap bit b in Algorithm 1). We applied the Davies-Bouldin index measure to the clustering outputs of multiple runs of the same randomized clustering algorithm, each run with a different RNG seed, and selected the clustering output with the best (i.e., smallest) index value.

3.3 Outlier Detection and Handling

According to Hawkings [13], "*An outlier is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*" In the HCA context, outliers can appear in the measured samples due to, e.g., measurement errors or unknown device behavior, and have a significant impact on clustering. Most clustering algorithms are not intrinsically robust to outliers, so depending on how large an outlier value deviates from "normal" values, the resulting labels might be negatively influenced by the outlier. Then the resultant clusters might be completely different, and thus wrong, from what would be expected, leading to potentially misleading results. Hence, outlier detection is desirable as a preprocessing step before clustering in HCA.

We implemented and tested the following outlier detection methods for HCA: distance from mean and Tukey's test. A simple outlier detection method, hereafter called *distance from mean*, is given by considering the values that are far from the mean as outliers, i.e., a value x is an outlier if $|x - \mu| \geq \beta\sigma$, for a non-negative parameter β; μ and σ are the mean and standard deviation, respectively. We chose $\beta = 2.0$.[9] Tukey's range test [33] is a method based on order statistics. If Q_1 and Q_3 are the lower and upper quartile, respectively, and $IQ = Q_3 - Q_1$ is the interquartile, any observation outside the closed interval $[Q_1 - k \cdot IQ, Q_3 + k \cdot IQ]$ is considered an outlier, for a non-negative parameter k. We chose $k = 1.5$, the value proposed in [33].

If an outlier detector flags some samples as outliers, they must be dealt with in some way, i.e., the outliers have to be handled. Outlier handling methods are usually heuristic and dependent of the context where they are applied [32]. A simple outlier handling method that could be applied in the context of this work is to simply exclude the data point from consideration. Albeit simple, the implementation of this method is potentially inefficient in the HCA context, due to the need of more complex data structures (e.g., dynamic lists rather than static arrays). To keep the implementation simple and efficient, we replace outliers values by the median of non outliers.

4 Horizontal Cluster Analysis Framework

The horizontal attack described in this paper roughly follows the HCA framework from [34], with contributed analysis methods. Figure 1 shows the steps in the HCA framework. The first step is to run clustering leakage assessment (CLA). CLA takes as input iteration traces from multiple ECSM runs and finds points in the traces where the leakage most likely is, known as points of interest (POIs). Next, key recovery (KR) is run, yielding an approximate scalar. Then, given the approximate scalar, points-of-interest optimization (POI-OPT) produces a refined list of POIs. Finally, the final KR step outputs the recovered scalar.

[9] Assuming that the sample values at a given index come from a normal distribution, choosing $\beta = 2.0$ implies that 95% of the values are within the interval $[x - \mu, x + \mu]$.

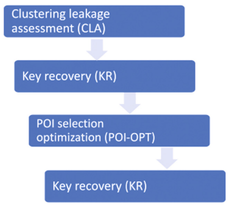

Clustering leakage assessment (CLA)
Key recovery (KR)
POI selection optimization (POI-OPT)
Key recovery (KR)

Fig. 1. HCA framework.

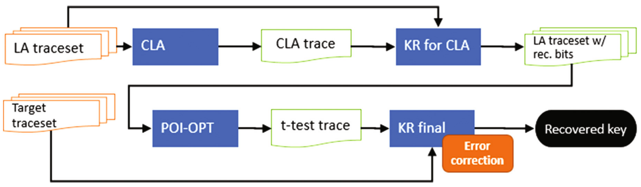


Fig. 2. The full key recovery process.

Figure 2 shows the full key recovery process in more detail, including the inputs and outputs at each step. The inputs are a traceset to be used for leakage assessment and the actual target traceset. The output is the correct recovered key/scalar, if it could be found.

The "KR final" step takes as input a traceset with traces from multiple ECSM runs and attacks the sets of segment traces of each ECSM run independently from one another. This step is a probabilistic algorithm that consists of sequentially running KR and error correction (Sect. 6) on each trace in the set, and recovering the correct scalar for at least one of these traces, with a given probability of success. We call this probability the *success rate* of the attack.

In our HCA framework, POIs are chosen from a leakage assessment trace, be it a CLA or t-test trace. They are selected as the time indices of the top m highest peaks in such traces, where m is a parameter. Suitable values for m are derived experimentally (cf. Sect. 5 and Fig. 4).

4.1 Cluster Leakage Assessment (CLA)

Leakage assessment (LA) methods are used to determine whether a cryptographic device leaks information through a side-channel and how strong such leakage is. They are typically employed to find out the points in time (sample indices) where the leakage is strongest, i.e., the points of interest. The sample values at those points are used in later steps, e.g. in the key recovery phase, and they serve two major purposes: (i) for dimensionality reduction, i.e., to use only the samples that provide useful information and thus reduce computation time; (ii) to avoid bringing noisy samples to the attack phase (i.e., key recovery), where they will negatively impact the success rate and potentially turn unfeasible an otherwise successful attack.

In the HCA and non-profiled attack contexts, leakage assessment methods should not require knowledge of the secret key or ephemeral secret data (e.g., numbers randomly generated by the device). Essentially, these methods assume that the adversary does not have control over device's secret information. In particular, the countermeasures like SR and CRR cannot be disabled. Additionally, it is desirable that LA methods be non-parametric and do not require leakage models. That is because in the single-trace HCA attack context the target device is not known a priori. Combined with the fact that real-world modern microcontrollers are complex devices, it means that the estimation of leakage distributions and thus building an accurate leakage model is not trivial.

Distinguishers. Welch's t-test is a parametric statistical test that can be employed to this end; e.g., in methodologies like the TVLA [12]. Certain conditions have to be met for Welch's t-test to be used: normality of the distributions, equal variances and independence. The Mutual Information Analysis (MIA) is a distinguisher that does not require a leakage model. Standaert et al. [38] were the first to propose the use of MI as an statistical leakage assessment tool in the SCA context. Meynard et al. [27] applied MIA as a method to locate strong leakage in the frequency domain and, consequently, to find the frequency bands in EM traces where the differences between modular squares and multiplications are highest, from a device running RSA modular exponentiation. Mather et al. [26] compared the statistical power of t-test and the discrete and continuous versions of MIA for detecting leakage in multiple leakage models. Following [34], we provide results for four different methods for leakage detection: sum-of-squared differences (SOSD), sum-of-squared t-values (SOST) and MIA.

CLA. The LA method proposed in [34] shows how multiple traces can be combined utilizing clustering, an unsupervised learning method, and demonstrate it through an attack on a RSA software implementation. That method, hereafter called clustering-based leakage assessment (CLA), in principle works even if the device applies any combination of the classic countermeasures for modular exponentiation, i.e., exponent blinding, message or modulus randomization.

4.2 Key Recovery (HCA-KR)

The key recovery methods implemented in this work can be classified into two classes, based on the way clustering is applied: single or multi-dimensional. In either case, the number of clusters output by a clustering algorithm is two, one cluster for each possible key bit value.

Single-Dimensional Clustering Method. In the first group, we run clustering on the set of samples at a given single time index (POI), across multiple trace segments. This is the approach used by [34]. After running the clustering for every time index, a set of recovered key candidates is obtained, which are then combined to decide the final key candidate. The following combination methods are used for this purpose: majority rule (MJ) and log-likelihood (LL). We refer the reader to [34] for more details.

Multi-dimensional Clustering Method. In the second group, clustering is run on multiple attributes or dimensions, i.e., the clustering algorithm is run on all samples, at all points of a set of time indices (POIs), at the same time.

On one hand, the multi-dimensional method has two main advantages over the single-dimensional. First, the combination step is not required. And second, it is capable of exploiting higher order leakage, while the first method exploits only leakage of first order. On the other hand, we verified experimentally that key recovery based on multi-dimensional clustering is more sensitive to noisy samples, because a very noisy sample at a given POI can directly negatively influence the value of the output key candidate. Note that for the first group of attacks the noisy sample effect is contained, i.e., only the key candidate at that POI is affected. Therefore, outlier detection and handling are mandatory in the multi-dimensional method to achieve satisfactory results.

First key-recovery step ("KR for CLA" in Fig. 1). After points of interest have been found by CLA, key recovery is run on the "LA traceset" using the POIs from the CLA trace. The outcome is a list of recovered candidate keys for those traces ("LA traceset w/ recv.bits").

POI optimization step (POI-OPT in Fig. 1). The "LA traceset w/ recv.bits" is used as input for the points of interest selection optimization step. This step refines the POIs found from CLA by applying a t-test with two groups, the first group containing the traces whose corresponding candidate key bit is zero and the second group corresponds to the traces where the candidate key bit is one. The points with the largest t-statistics are considered the refined POIs.

Final key recovery step ("KR final" in Fig. 1). Finally, given the refined list of POIs (i.e., the peaks on "t-test trace"), key recovery is applied sequentially to each trace in the target traceset. For each trace, the key recovery outputs a (possibly incorrect) key/scalar, over which the probabilistic key error correction algorithm in Sect. 6 is applied. If the correct scalar is found, it is returned and the full key recovery process stops. Otherwise, the key recovery is applied to the next trace in the target traceset and the process is repeated.[10]

5 Attack Results

We acquired 300 full Curve25519 ECSM traces for the cswap-arith, and the same number of traces for the cswap-pointer. The traces were preprocessed, resulting in two tracesets of 76,500 ECSM iteration traces each, that are used in all experiments described in this section. Each iteration trace used for the analysis contains 8,000 8-bit samples for cswap-arith. For cswap-pointer, as the time interval where leakage happens is narrower (cf. Sect. 2.1), we trimmed the traces to 1,000 samples for efficiency.[11]

In the evaluation experiments, the recovered scalars are the output of the last KR step, but before error correction. The reported success rates are, unless otherwise noted, the maximum success rates, i.e., if the success rate or percentage of correctly recovered bits for the target traces is $SR_1, \ldots, SR_{n_t}$ (where n_t is the number of traces in the target traceset), then the reported success rate is $\max\{SR_1, \ldots, SR_{n_t}\}$. We use the maximum success rates because, as explained

[10] We note that the steps POI-OPT and key recovery can be repeated. Due to the high increase in computational time required to run them more than once, as well as the fact that the results using a single iteration were already feasible for a successful attack, we chose not to further investigate whether that could improve the results.

[11] We knew where and by how much to trim because we knew from the source code and binary the approximate location of the cswap in the iteration traces.

Table 1. Key recovery max success rate (%) for cswap-arith and cswap-pointer, for all combinations of CLA distinguishers (SOSD, MIA and SOST) and HCA statistical combination methods. The best results for each implementation are highlighted.

		cswap-arith			cswap-pointer		
		MJ	LL	MD	MJ	LL	MD
KM	SOSD	92.15	**94.11**	58.03	97.25	60.78	96.47
	MIA	60.78	57.64	58.82	96.47	95.68	57.25
	SOST	**94.11**	92.15	57.64	99.60	96.07	**100.00**
FKM	SOSD	87.84	57.25	58.43	57.64	58.82	98.82
	MIA	60.78	84.31	59.60	99.60	99.21	56.86
	SOST	67.45	59.21	58.03	59.60	98.82	**100.00**
EM	SOSD	60.00	61.56	60.39	97.64	57.64	57.64
	MIA	60.39	68.23	60.39	99.21	95.29	99.60
	SOST	64.31	61.96	57.64	97.64	95.29	57.64

in Sect. 4, our full HCA key recovery attack framework is probabilistic and recovers the correct scalar for at least one of the target traces with a given probability of success, the success rate of the attack. By taking the max success rate (SR^*) in an attack evaluation experiment as the success rate, we guarantee (except with a negligible chance) that if the full key recovery attack with error correction is applied to a set of target traces, the recovered key for at least one of them will have a ratio of at least SR^* of correctly recovered bits. Therefore, the errors can be successfully corrected by the error correction algorithm in Sect. 6.

5.1 Initial Attack Evaluation Experiment

To evaluate the effect of different distinguishers for CLA and statistical combinators for HCA-KR, we fixed the clustering algorithm (KM, FKM or EM) and ran a full key recovery attack varying the value of such parameters. The evaluation results are shown in Table 1.[12]

In this experiment we used: 100 traces for CLA; 100 traces and 20 POIs for "KR for CLA" and "KR final". We experimented with different numbers of traces for these operations, but from those we tried, 100 was the minimum number of traces that resulted in good enough attack success rates; we did not see any improvement when more traces were used. POI-OPT is enabled. We used Tukey test and replace by median as outlier detection and handling methods, respectively. Intrinsic clustering quality evaluation is disabled.

According to Table 1, the cswap-pointer implementation has a very strong leakage dependent on the cswap bit, in two cases reaching a success rate of 100%, i.e., all scalar bits were correctly recovered. Despite having obtained 100% success rate for two combinations of algorithms KM/SOST/MD and FKM/SOST/MD

[12] MJ, LL and MD stand for majority rule, log-likelihood and multi-dimensional, resp.

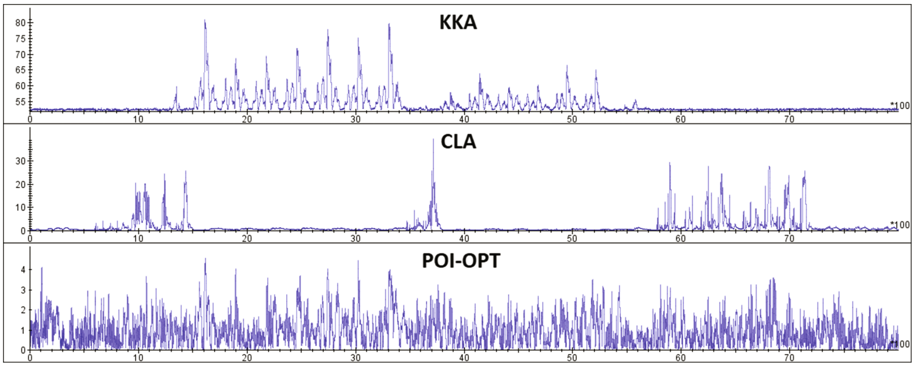


Fig. 3. Leakage assessment for cswap-arith (top-bottom): KKA, CLA, and POI-OPT.

for the cswap-pointer implementation, similar success rate results for such combinations of algorithms on the cswap-arith implementation do not hold. In fact they were very low for that implementation, with success rates below 60%.

The results obtained in Table 1 do not indicate a single combination of parameters where the success rates are high enough ($\geq$97%) for both implementations simultaneously, so as to enable a successful recovery of the correct scalar in feasible time even when error correction (Sect. 6) is applied.

Besides, in a practical non-profiled or single-trace attack scenario, where the attacker do not know details about the implementation targeted, she should fix/choose beforehand the values of all parameters for the full key recovery[13] and run "KR final" for every trace in the target traceset. The motive is the long computation time required to run a full key recovery, where the most expensive step, error correction, can take hours to complete on a common desktop machine.

For the aforementioned reasons, we test our attack with more combinations of parameters values. The results of these experiments are in the full version of the paper [29]. The values of those parameters that gave the best results are presented in the next subsection.

5.2 Final Results

Figure 3 illustrates, for cswap-arith, the approximated side-channel leakage assessed right after the CLA and POI-OPT steps when compared with a known-key analysis (KKA) trace. The leakage assessment trace after POI-OPT shows peaks that match or are very close to those in the KKA trace. A known-key analysis in essence consists of running the clustering algorithm at each sample index to recover the scalar bit and comparing whether the guessed bit value is equals the known key bit. The output is a trace with the success rate of these guesses, which is indicates the strength of the leakage. Such an analysis is used only for illustrative purposes, we remark that our attacks are completely unsupervised.

[13] Among them: number of traces for CLA, "KR for CLA" and "KR final" steps, clustering algorithms, distinguishers and statistical combination methods.

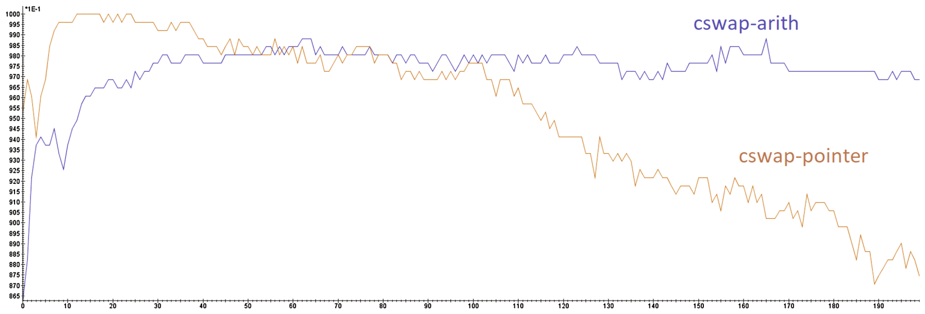


Fig. 4. Success rate versus number of POIs for cswap-arith and cswap-pointer.

Figure 4 shows the success rate evolution as the number of POIs used by the final HCA-KR step increases, for both cswap implementations. For both implementations, the number of traces used are 100 for CLA is 100, "KR for CLA" and "KR final", the same as in the other experiments. For cswap-arith, the success rate is 97.64% for 100 POIs. The success rate for cswap-pointer is above 90% if the number POIs used is in $[7, 38]$. In particular, it is 99.60% for 38 POIs. Thus, for cswap-pointer a small number of POIs is sufficient to achieve a very high success rate. The curves in Fig. 4 have quite different shapes. The cswap-pointer curve shape means that leaks in narrow time intervals, so it is sensitive to the number of POIs used. In this case, using a lot of POIs means adding noise to the analysis, which decreases the success rate. On the contrary, the cswap-arith curve means that it leaks on a wider time interval, so more POIs can be added without dramatically affecting success rate.

Considering these success rate values and taking into account the fact that only 251 bits out of the 255 bits of the scalar are unknown (the first bit is always 1 and the last three are fixed to 100_2), there are at most 6 and 2 errors in the recovered scalar for cswap-arith and cswap-pointer, respectively.

6 Error Correction

Due to noise and other aspects interfering with the side-channel analysis (misalignment for example), the scalar derived by the attack contains errors. A naive brute-force would check all possibilities of 6 and 2 errors in the 251 bits, for each of the 100 recovered scalars. This totals to $100 \cdot \binom{251}{6} \approx 2^{44.9}$ operations for cswap-arith and $100 \cdot \binom{251}{2} \approx 2^{21.6}$ for cswap-pointer. As we can see, the required computation effort is quite feasible, especially for the cswap-pointer case.

Note that confidence probabilities coming from clustering can be used to detect errors, as shown in [14,34,35]. We applied the approach from [34], but unfortunately this method occurred unreliable: some errors occured with high confidence probabilities. We suspect that it was caused by strong noise pulses present in our traces. Therefore, we concentrate on improving the naive brute-force.

Efficient Error Correction Based on Precomputations. The above naive brute-force can be further by using a modified algorithm from [30]. In [30] the authors use template attack confidence scores to detect errors. Unfortunately, as mentioned above, we cannot use confidence probabilities and therefore, we need to modify the approach in [30], as described below.

First let us assume that the number of errors is at maximum 6 (like for cswap-arith). Now let us divide the scalar in half and assume the errors locations are uniformly distributed across it. Let us denote $R = [k]P$, where R is the resulting point, k the scalar to be recovered, and P is the input point. Then, clearly $R = [k]P = [a \cdot 2^{|k|/2} + b]P = [a]([2^{|k|/2}]P) + [b]P$, where a is the most significant half of k and b is the least significant one[14]. If we denote $[2^{|k|/2}]P$ by H, then the above equation reduces to $R - [b]P = [a]H$.

Consider all different possible guesses for a assuming that there are at most 4 errors in a: that is $\binom{|k|/2}{4}$ guesses. Following [30], for each guess, we compute $[a]H$ and store all pairs $(a, [a]H)$. We then sort all pairs based on the value of $[a]H$ and store them in an ordered table. We make a guess for b assuming it contains at most 4 errors (again $\binom{|k|/2}{4}$ guesses) and compute $z = R - [b]P$. If our guess for b is correct, then z is present in the second column of some row in the table – the first column is the corresponding a. If z is present then we have determined the scalar. Otherwise, we make a new, different guess for b and continue. The complexity of this attack totals to $\binom{126}{4} \cdot 2 \cdot 100 \approx 2^{31}$ operations, because there are 251 unknown bits. The required memory is $\binom{126}{4} \cdot 100 \approx 2^{30}$ points.

The above assumption on uniform distribution of errors can be dropped (cf. Appendix A). We need to estimate the probability that the attack works. The probability that out of 6 errors, 2, 3, or 4 of them are not in a equals: 14/64; for details about computing this probability we refer the reader to Appendix A. Thus, to minimize the error to approximately e.g., 0.0005, it is enough to repeat the algorithm 5 times. Then the overall complexity of the attack would be 2^{36}.

7 Countermeasures and Future Work

In this paper we described horizontal clustering attacks against two Curve25519 Montgomery Ladder ECSM implementations from the μNaCl library. We also showed how to extend the RSA horizontal clustering framework from [34]. Furthermore, we generalized the method from [30] to tolerate a certain number of incorrectly recovered scalar bits without relying on normal exhaustive search.

Now we briefly discuss possible countermeasures against our attack. First let us recall that the following countermeasures do not work against our attack: point re-randomization, scalar blinding and splitting.

The countermeasure of [31] splits scalar into two parts and to randomly interleave two scalar multiplications. We believe that our attack might still be mounted if four clusters are used to recognize which bit is processed and during which ECSM. The idea behind the memory-address countermeasure [16] is

[14] $|k|$ denotes the length of the base 2 representation of the scalar k.

to store sensitive variables at addresses that share the same Hamming weight. Although this would decrease the effectiveness of the attack, the addresses leakage may still be identified by clustering. This countermeasure can be improved by randomizing not only the addresses but also the memory accesses [17–19].

The countermeasure of [15] protects against localized EM template attacks on Montgomery ladder ECSM by randomly swapping the ladder registers at the end of a ladder iteration. This countermeasure is uniform in its operation sequence what makes our attack infeasible in principle. In addition, several randomization-based protection techniques for the Montgomery ladder are presented in [24]. Similar to [15], these techniques generate operation sequences independent from the scalar and thus, our attack might be ineffective against them.

We consider evaluating and improving our attacks with respect to the two latter countermeasures as future work. We also regard attacking other ECC implementations improving our attacks with PCA as future developments.

Acknowledgements. This work was supported by the European Union's H2020 Programme under grant agreement number ICT-731591 (REASSURE).

A Probability of Successful Efficient Error Correction

We assumed in Sect. 6 that the errors are uniformly distributed. Now we show how to drop this assumption. We create a in the following way: we randomly choose a set A of indices in k such that $|A| = |k|/2$ and we set the corresponding bits to zero. Then we create b by setting the remaining indices of the original k to zero (the set of indices is denoted as B). Now $R = [a]P + [b]P$ holds and if we set $H = P$ then $R - [b]P = [a]H$. The attack can be performed as before assuming that when we guess a and b, we limit the indices to A and B, respectively.

We now compute the probability that the attack from Sect. 6 works correctly, namely, that the 6 errors are corrected. Without loss of generality let us first assume that positions of the 6 errors position are fixed, because the partition to a and b is random. Therefore, the following situations are possible:

– all errors are in a or in b: 2 possibilities;
– one error is in a or b: 12 possibilities;
– two errors are in a or b: 30 possibilities;
– three errors are in both a and b: 20 possibilities.

In the first two cases the numbers of errors in a is 0, 1, 5, or 6. Therefore, the probability that out of 6 errors, 2, 3, or 4 of them are not in a equals:

$$\frac{1+6+6+1}{2+12+30+20} = \frac{14}{64}.$$

References

1. Bauer, A., Jaulmes, É.: Correlation analysis against protected SFM implementations of RSA. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 98–115. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_7

2. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal and vertical side-channel attacks against secure RSA implementations. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_1
3. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. Theory Methods **3**(1), 1–27 (1974)
4. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 140–155. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34931-7_9
5. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17650-0_5
6. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 224–227 (1979)
7. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B (Methodol.) **39**, 1–38 (1977)
8. Dürr, F.: Key 2.0 is a Bluetooth IoT Door Lock (2017). https://github.com/duerrfk/key20
9. EMV: EMVCo Security Evaluation Process, version 5.1, Security Guidelines (2016)
10. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugenics **7**(7), 179–188 (1936)
11. Forgy, E.W.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**, 768–769 (1965)
12. Goodwill, G., Jun, B., Jaffe, J., Rohatgi, P.: A testing methodology for side channel resistance validation. In: NIST Workshop 2011 (2011)
13. Hawkings, D.: Identification of Outliers. Chapman and Hall, London (1980)
14. Heyszl, J., Ibing, A., Mangard, S., De Santis, F., Sigl, G.: Clustering algorithms for non-profiled single-execution attacks on exponentiations. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 79–93. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_6
15. Heyszl, J., Mangard, S., Heinz, B., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of cryptographic implementations. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 231–244. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_15
16. Itoh, K., Izu, T., Takenaka, M.: Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 129–143. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_11
17. Itoh, K., Izu, T., Takenaka, M.: A practical countermeasure against address-bit differential power analysis. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 382–396. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45238-6_30
18. Izumi, M., Ikegami, J., Sakiyama, K., Ohta, K.: Improved countermeasure against address-bit DPA for ECC scalar multiplication. In: 2010 Design, Automation and Test in Europe Conference and Exhibition (DATE 2010), pp. 981–984. IEEE (2010)
19. Izumi, M., Sakiyama, K., Ohta, K.: A new approach for implementing the MPL method toward higher SPA resistance. In: International Conference on Availability, Reliability and Security, ARES 2009, pp. 181–186. IEEE (2009)

20. Järvinen, K., Balasch, J.: Single-trace side-channel attacks on scalar multiplications with precomputations. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 137–155. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_9
21. Jolliffe, I.: Principal Component Analysis. Springer Series in Statistics. Springer, Heidelberg (2002). https://doi.org/10.1007/b98835
22. Kauffman, L., Rousseeuw, L.: Finding Groups in Data. An Introduction to Cluster Analysis. Wiley, New York (1990)
23. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
24. Le, D.-P., Tan, C.H., Tunstall, M.: Randomizing the montgomery powering ladder. In: Akram, R.N., Jajodia, S. (eds.) WISTP 2015. LNCS, vol. 9311, pp. 169–184. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24018-3_11
25. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
26. Mather, L., Oswald, E., Bandenburg, J., Wójcik, M.: Does my device leak information? An *a priori* statistical power analysis of leakage detection tests. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 486–505. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_25
27. Meynard, O., Réal, D., Flament, F., Guilley, S., Homma, N., Danger, J.L.: Enhancement of simple electro-magnetic attacks by pre-characterization in frequency domain and demodulation techniques. In: 2011 Design, Automation and Test in Europe, pp. 1–6 (2011)
28. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-39799-X_31
29. Nascimento, E., Chmielewski, L.: Applying horizontal clustering side-channel attacks on embedded ECC implementations (extended version). Cryptology ePrint Archive, Report 2017/1204 (2017). https://eprint.iacr.org/2017/1204
30. Nascimento, E., Chmielewski, Ł., Oswald, D., Schwabe, P.: Attacking embedded ECC implementations through cmov side channels. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 99–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_6
31. Negre, C., Perin, G.: Trade-off approaches for leak resistant modular arithmetic in RNS. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144, pp. 107–124. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19962-7_7
32. NIST: NIST/SEMATECH e-Handbook of Statistical Methods. Section 7.1.6. What are outliers in the data? (2013)
33. NIST: NIST/SEMATECH e-Handbook of Statistical Methods. Section 7.4.7.1. Tukey's method (2013)
34. Perin, G., Chmielewski, Ł.: A semi-parametric approach for side-channel attacks on protected RSA implementations. In: Homma, N., Medwed, M. (eds.) CARDIS 2015. LNCS, vol. 9514, pp. 34–53. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31271-2_3
35. Perin, G., Imbert, L., Torres, L., Maurine, P.: Attacking randomized exponentiations using unsupervised learning. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 144–160. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10175-0_11
36. T. H. Project: Picotls - TLS 1.3 implementation in C (2017). https://github.com/h2o/picotls

37. Specht, R., Heyszl, J., Kleinsteuber, M., Sigl, G.: Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In: Mangard, S., Poschmann, A.Y. (eds.) COSADE 2014. LNCS, vol. 9064, pp. 3–19. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21476-4_1
38. Standaert, F.-X., Malkin, T.G., Yung, M.: A formal practice-oriented model for the analysis of side-channel attacks. IACR e-print archive 2006/134 (2006)
39. Walter, C.D.: Sliding windows succumbs to Big Mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_24
40. Wilkinson, T.: HomeKit for Bluetooth Low Energy (BLE) for Nordic nRF51 (2015). https://github.com/aanon4/HomeKit
41. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_6

Trace Augmentation: What Can Be Done Even Before Preprocessing in a Profiled SCA?

Sihang Pu[1(✉)], Yu Yu[1], Weijia Wang[1], Zheng Guo[1], Junrong Liu[1], Dawu Gu[1], Lingyun Wang[2], and Jie Gan[3]

[1] Shanghai Jiao Tong University, Shanghai, China
{push.beni,yyuu,aawwjaa,guozheng,liujr,dwgu}@sjtu.edu.cn

[2] Shanghai Viewsource Information Science and Technology Company, Shanghai, China
lingyun.wang@viewsources.com

[3] Beijing Smart-Chip Microelectronics Technology Co., Ltd., Beijing, China
ganjie@sgitg.sgcc.com.cn

Abstract. Preprocessing is an important first step in side-channel attacks, especially for template attacks. Typical processing techniques, such as Principal Component Analysis (PCA) and Singular Spectrum Analysis (SSA), mainly aim to reduce noise and/or extract useful information from raw data, and they are barely robust to tolerate differences between profiling and target traces. In this paper, we propose an efficient and easy-to-implement approach to preprocessing by applying the data augmentation method from deep learning, whose appropriate parameters can be efficiently determined using a simple validation. Our trace augmentation method, when added prior to existing profiling methods, significantly enhances robustness and improves performance of the attacks. Simulation-based experiments show that our approach not only results in a more robust profiling (even show an enhancement to the known robust profilings), but also works well in the ideal scenario (no distortions between profiling and target traces). The results of FPGA-based and software experiments are consistent to the ones of simulation-based counterparts. Thus, we conclude that the proposed augmentation method is an efficient performance-boosting add-on to profiled side-channel attacks in real world.

1 Introduction

1.1 Motivation

The crypto community has witnessed the fast development of Side-Channel Attacks (SCAs) since Kocher's original works [11–13], and various more efficient SCA methods are proposed for different scenarios. Profiled SCA, first proposed by Chari et al. [5], adds a profiling phase (prior to the online exploitation phase) to the original SCA and can be considered as the powerful class of power analysis. Since then, various profiled SCA methods have been introduced and studied (see [6,15,19,24–26] for an incomplete list).

T. Eisenbarth and Y. Teglia (Eds.): CARDIS 2017, LNCS 10728, pp. 232–247, 2018.
https://doi.org/10.1007/978-3-319-75208-2_14

Despite its excellent effectiveness, profiled SCA presumes high similarity between profiling device and target device, which might otherwise result in less desirable performance and thus limit their applicability in practice. This issue was noticed and studied by Standaert et al. [22], Elaabid and Guilley [10] and Choudhary and Kuhn [7]. Furthermore, Whitnall and Oswald [26] proposed a robust profiling method by applying clustering technique, and Wang et al. [24] proposed another robust profiling technique using the ridge regression method. To the best of our knowledge, all the existing solutions mainly focus on the profiling phase rather than preprocessing stage.

1.2 Preprocessing Techniques

Preprocessing techniques are widely used to increase the success rate of side-channel analysis. The most widely used technique is Principal Components Analysis (PCA), which was first introduced by Archambeau et al. [1] and extended by Batina et al. [3]. Standaert and Archambeau [21] compared PCA with a more contrived method named Fisher Linear Discriminant Analysis (LDA). Bruneau et al. [4] carried out a mathematical analysis of dimensionality reduction methods (i.e., PCA and LDA), and they concluded that LDA is asymptotically the optimal dimensionality reduction strategy. Choudary and Kuhn [6] compared several Points of Interest (POI) techniques and discussed the rules of selecting components. Recently, Merino Del Pozo and Standaert [17] used Singular Spectrum Analysis (SSA), a technique originally used in time series analysis, to ameliorate the Signal Noise Ratio (SNR) of raw traces. We stress that the aforementioned preprocessing techniques may not work well with deviated target devices, and they need to rely on subsequent robust profiling techniques. To resolve the issue, we propose a new preprocessing method based on data augmentation.

1.3 Data Augmentation

The term data augmentation refers to methods for constructing iterative optimization or sampling algorithms via the introduction of unobserved data or latent variables. This method was popularized in the deep learning community and it can produce a better profiling set to mitigate overfitting and build a more robust model. Simard et al. [20] first created a general set of elastic distortions that vastly expanded the size of the training set. Moreover, in deep learning, it is the easiest and most common method to artificially enlarge the dataset using label-preserving transformations (e.g., [8,9,18]) in order to reduce overfitting. Krizhevsky et al. [14] significantly reduced the error rate using data augmentation techniques.

1.4 Our Contributions

In this paper, we tackle the following problem:

What can be done during (or even before)
the preprocessing stage to make the subsequent attacks more robust?

We answer this question affirmatively. Borrowing ideas from deep learning, we introduce the "trace augmentation" technique (i.e., applying data augmentation in the SCA context), which shifts each trace to some random extent and then combine them all (both original and perturbed traces) to form an augmented trace set. Our trace augmentation method can be applied prior to existing preprocessing procedures (e.g., PCA, LDA) and significantly improve the robustness and performance of the attacks.

Further, we propose a very efficient method (named lazy validation) to find out an appropriate range of the shift, only based on the profiling traces. Informally speaking, this method splits the profiling traces into two partitions, profiles on one distorted partition, validates (by conducting attacks) on the other undistorted one, and chooses the largest range of distortion that doesn't essentially impact the result of the attack. Intuitively, the resulting suggestion can be seen as a conservative one that at least doesn't impact the effectiveness in the ideal setting (where there are no distortions between profiling and target trace).

At last, we conduct both simulation-based and practical experiments to verify our approach. They both show that trace augmentation not only improves the performance of the subsequent attack in scenarios where discrepancy exists between the profiling device and target device, but also works well in the ideal case (i.e., without distortions). In addition, simulated-based results suggest that the improvement of our method is related to the correlation among points of each trace. In the practical setting, we carry out the experiments on both software and FPGA implementations, whose results are consistent to the simulation.

2 Trace Augmentation

In this section, we present our trace augmentation method, and show how to determine the suitable parameters efficiently. We stress that trace augmentation can either work independently, or can be added prior to any other preprocessing techniques such as PCA, in order to produce a more robust trace set.

2.1 Trace Augmentation Through Random Shift

Generally speaking, this augmentation approach manages to increase the number of traces exploitable in profiling phase, in order to improve the robustness and performance in profiled SCA. This is achieved through random shift of each trace to form an expanded trace set (i.e. in this process we misalign the traces by shifting the trace entirely). We visualized this random-shift-based augmentation approach in Fig. 1 and we sketch its procedure as follows.

1. Shift each trace horizontally at random up to some extent (the shift ratio to be determined later).
2. Repeat step 1 several times (corresponding to the augmentation ratio) to yield many perturbed traces.
3. Append these new perturbed traces to original set.

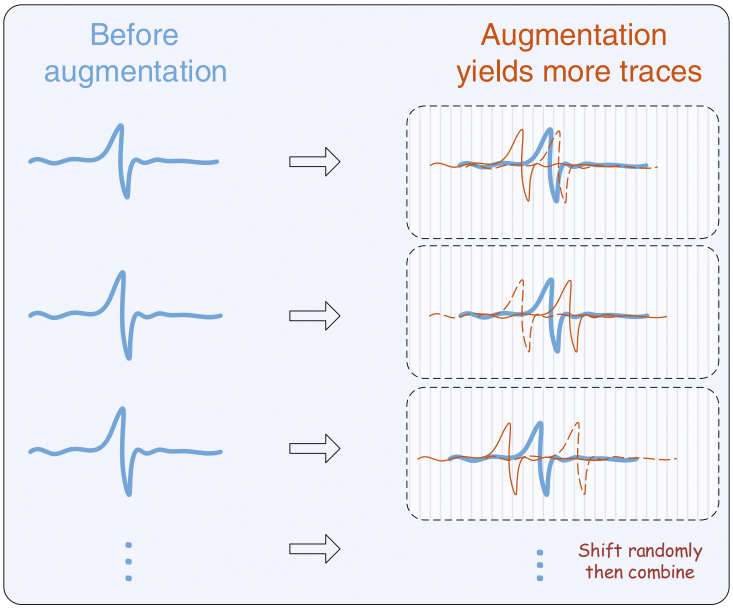


Fig. 1. A visual illustration of trace augmentation.

The above is parameterized by the shift ratio and augmentation ratio, denoted as γ and C respectively. The shift ratio γ, quantifies the extent of random shift: perturb each trace with a horizontal random shift drawn uniformly from $[-\gamma \cdot d, \gamma \cdot d]$, where d denotes the number of leakage points. The number of expanded traces is determined by the augmentation ratio C, shift ratio γ and the original trace number, that is, we have that $N_{new} = \gamma C \cdot N_{original}$, where N_{new} and $N_{original}$ are the numbers of original and new added traces respectively. Algorithm 1 presents this approach in formal details.

Intuitively, the more traces to exploit in the profiling phase the more effective the attack will be (against the target device). This motivates our approach which, given a fixed number of traces, augments the trace set (and exploit its information) as much as possible for a better performance in profiled SCA. The idea to enlarge the existing trace set is to apply perturbations since enlarging the dataset using such label-preserving method is the most straightforward yet efficient way to reduce overfitting. This approach works well especially when trace set is small. Moreover, we suggest to combine the trace augmentation approach with some data selecting (or points of interests) method such as PCA and LDA.

2.2 Search for Appropriate Parameter Though Lazy-Validation

As illustrated above, there are two undetermined parameters (i.e., γ and C), and it is somewhat tricky to define a general rule for to select them. We noted that the

Algorithm 1. Trace augmentation

```
Input: original trace set $\boldsymbol{L}$ ($N_{original}$ traces), number of leakage points $d$, shift
       ratio $\gamma$, augmentation ratio $C$
Output: augmented trace set $\boldsymbol{L}^{agmt}$ composed of $N_{original} + \gamma \cdot CN_{original}$
        traces
1 Append $\boldsymbol{L}$ to $\boldsymbol{L}^{agmt}$;
2 for $i = 1$ to $N_{original}$ do
3     Generate $\lfloor \gamma C \rfloor$ traces by randomly shift $\boldsymbol{L}_i$;
4     /* Note that each point of same trace shares a common horizontal
         shift                                                         */
5     Append the new traces to $\boldsymbol{L}^{agmt}$;
6 end for
7 return $\boldsymbol{L}^{agmt}$;
```

choice of γ affect the improvement of trace augmentation significantly, whereas (as verified in Appendix A) the choice of the other one (i.e., C) does not change the result much. We simply choose $C = 10$, and (as shown in Algorithm 2) design a lazy-validation method to yield a conservatively appropriate value for γ. We sketch the procedure as follows.

1. Select $C = 10$.
2. Split profiling traces into two (disjoint) partitions at random.
3. Perform trace augmentation on one partition with a certain shift ratio.
4. Build the template from this augmented partition.
5. Perform profiled SCA on the other partition with the template, and calculate a guessing entropy.
6. Repeat steps 3–5 with varied shift ratios, and obtain guessing entropy for each.
7. Select γ as the largest shift ratio that doesn't impact the result of the attack.

The underlying intuition of this procedure is that it is safe to perturb traces to some certain extent as long as it does not affect the performance in the ideal setting (no misalignment between profiling and target trace). Thus, the largest possible γ in respect of this condition can be seen as a conservative one. The optimal value of γ is highly specific to actual target trace set, whereas our conservative choice is in general an appropriate one universal for target traces with different levels of misalignment.

On the other hand, such lazy-validation might cost a little more time to yield the final parameter (γ). However, these procedures (including both augmentation and validation) are supposed to be finished at profiling stage, and attackers just end up getting a robust template. Therefore, attacking time will not be lengthened in practical.

Algorithm 2. Lazy-Validation

```
Input: original trace set $\boldsymbol{L}$ ($N_{original}$ traces), number of leakage points $d$, a
       vector of candidates of shift ratio $\Gamma$ in ascending order
Output: an appropriate shift ratio $\hat{\gamma}$
1  Split $\boldsymbol{L}$ into two random partitions $\boldsymbol{L}^{prof}$ and $\boldsymbol{L}^{valid}$;
2  Build template $T$ from $\boldsymbol{L}^{prof}$;
3  $ge_0 \leftarrow$ CalculateGuessingEntropy($T, \boldsymbol{L}^{valid}$);
4  $\hat{\gamma}$ is initialized to the first element of $\Gamma$ ;
5  for each $\gamma \in \Gamma$ do
6      $\boldsymbol{L}^{agmt} \leftarrow$ TraceAugmentation($\boldsymbol{L}^{prof}, d, \gamma, C = 10$);
7      Build template $T'$ from $\boldsymbol{L}^{agmt}$;
8      $ge \leftarrow$ CalculateGuessingEntropy($T', \boldsymbol{L}^{valid}$);
9      if $ge > ge_0$ then
10         break;
11     end if
12     $\hat{\gamma} \leftarrow \gamma$;
13 end for
14 return $\hat{\gamma}$;
```

3 Experimental Results

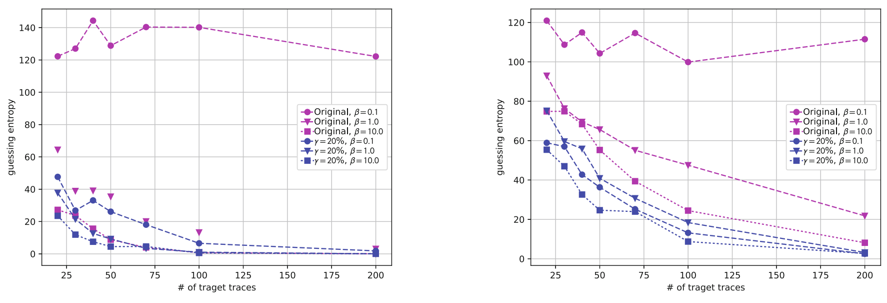
We test our approach through simulation-based and practical experiments. In simulation scenes, we disturb the power model of profiling and attacking stages. Whereas in practical scenes, since changing power model is knotty to control, we follow Whitnall and Oswald [26] and conduct the experiments by adding misalignment between profiling and target traces. We target on the first 8 bits of the AES's subkey and the output of the corresponding S-box in the first round. To evaluate the effectiveness of our method, we compute the guessing entropy [23] for comparison; in particular, we mount attacks for 100 times on different inputs and then compute the average rank of the correct key.

3.1 Simulation-Based Experiments

In simulation-based experiments, we assume that the leakage is subject to the multi-variant Gaussian distribution. Thus, we choose the mean of the distribution by randomly picking numbers and rely on a refined method named 'vine' [16] to simulate the covariance matrix. The 'vine' method is an efficient way to generate random correlation matrices, and correlations between leakage points are controlled by a single parameter β: higher β value corresponds to the more dependencies among points of each trace (please refer to Appendix B for more details). In order to simulate the imperfect case where profiling and attacking traces exist discrepancies, we perturb the (standardized) leakage function of the exploitation trace by imposing a 'noise' of uniform distribution $\mathcal{U}(-5, 5)$. And then we can generate the deviated exploitation trace using the perturbed leakage function.

Impacts of Correlation Among Points. Our augmentation method is followed by LDA (reduced to only one point) to extract the points of interest and the Gaussian template [5] building is used as the profiling phase. Figure 2 compares the guessing entropies of profiled attacks by varying the power model of profiling and target traces, using different correlation matrices (reflected by the value of β), with and without using trace augmentation. It shows that with augmentation the guessing entropies are declining much faster than those without augmentation ($\gamma = 0$), not only in imperfect cases with noise (as expected), but also in the ideal cases (no noise). A probable reason of this surprising result might be that more traces (although artificial one) can be accessed in profiling phase to overcome 'overfitting' issue.

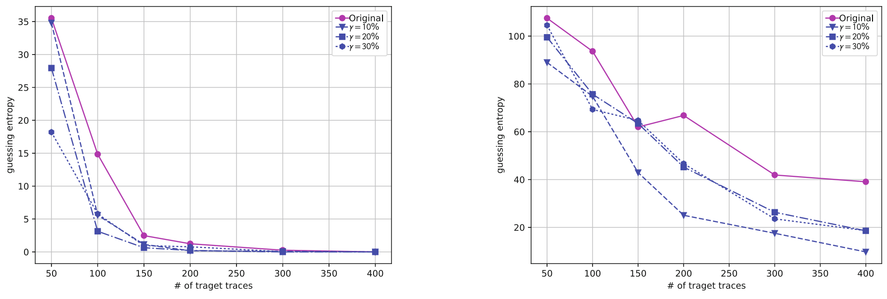
Further, we can see that attacks using augmentation are more 'insensitive' to β and in contrast, without augmentation, attacks are much more affected and become less effective while β decreases. We thus conclude that the effectiveness of this approach is enhanced while correlations among points are increasing (as β decreases). This means the power of this preprocessing method is correlated with the characteristics of the trace itself.



(a) perfect case (without discrepancy) (b) introducing noise as discrepancy

Fig. 2. The guessing entropies by varying the distribution between profiling and target traces (in terms of power model used), where simulation-implementation consists of 50 leaking points; 100 repetitions (to compute the guessing entropies) and 2000 profiling traces; using Gaussian templates

Enhancing the Robust Profiling Algorithm. We combine the our new method with the robust profiling algorithm proposed by Whitnall and Oswald [26]. We use K-means and Differential Cluster Analysis (DCA) introducted in [2] to perform attacks on target traces, in which the 'optimal' cluster number is according to the silhouette of the attack result. And other simulation facts are similar to the former simulation instance but β is fixed to 1.0. Figure 3 compares the guessing entropies of profiled attacks by different power models of profiling and target traces, with and without trace augmentation. It shows that with augmentation the guessing entropies are declining much faster than those without augmentation ($\gamma = 0\%$), even in the ideal cases (no discrepancy). We may conclude that even combining with robust algorithm (such as K-means), our trace augmentation technique can improve the performance in profiled attacks. Besides, we also provide comparison with another robust profiling algorithm (named ridge-based profiling [24]) in real FPGA-based contexts later (see Fig. 6).



(a) perfect case (without discrepancy) (b) introducing noise as discrepancy

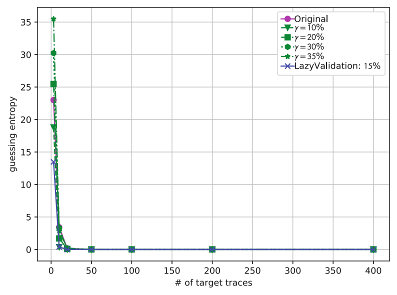
Fig. 3. The guessing entropies by varying the distribution between profiling and target traces (with different γ), where simulation-implementation consists of 50 leaking points; 100 repetitions (to compute the guessing entropies) and 2000 profiling traces; using cluster-based templates (K-means with DCA)

3.2 Software-Based Experiments

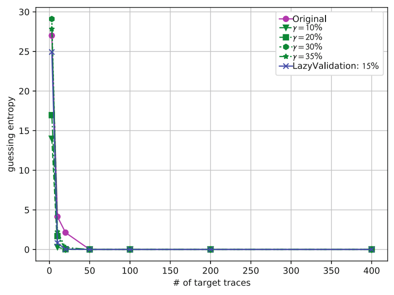
In software scenario, we target the AES implementation on Atmel ATMega-163 whose traces consist of 54 leakage points. Our augmentation method is followed by PCA (to reduce to 70% principal components [6]) to extract the points of interest and carry out the Gaussian templates building [5] as the profiling phase. Particularly, we use Hamming weight of target values as mean values in templates building, considering of its software implementation.

To provide a more comprehensive evaluation, we vary the parameter (of trace augmentation) γ from 10% to 35%, and the value $\gamma = 15\%$ is picked by lazy-validation of Sect. 2.2. For comparison, we also give the guessing entropies without trace augmentation. In such scenes, we follow Whitnall and Oswald [26] and conduct the experiments by adding misalignment between profiling and target traces. Note that this 'misalignment' (which misalign target traces with same points) should not be confused with 'shift' mentioned before.

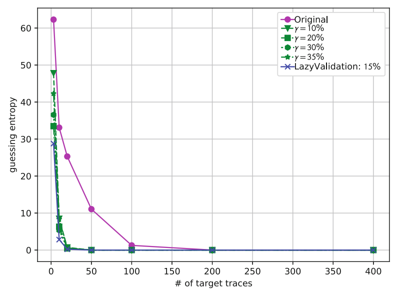
As shown in Fig. 4, with trace augmentation the performance of attacks have been improved in all settings even for those without misalignment. Note that misaligns of each trace are common for the same trace set. Further, we can see from following 6 sub-figures that the improvement of trace augmentation becomes more significant as the misalignment increases. Another observation is that, despite not always being the optimal, parameter γ chosen by lazy-validation is good enough for the attacks.



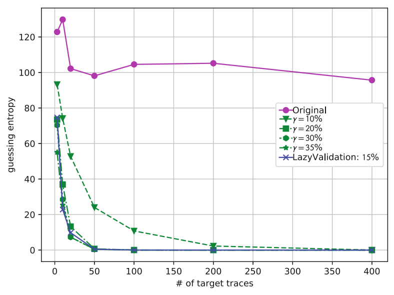
(a) no misalignment



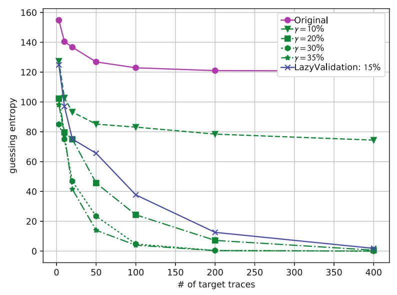
(b) misalignment = 5%



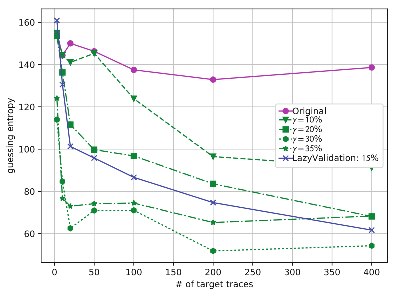
(c) misalignment = 10%



(d) misalignment = 15%

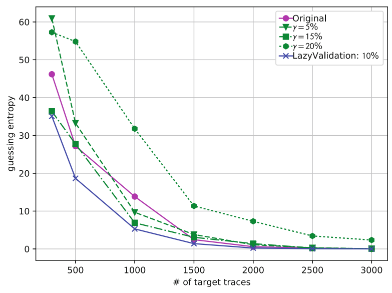


(e) misalignment = 20%

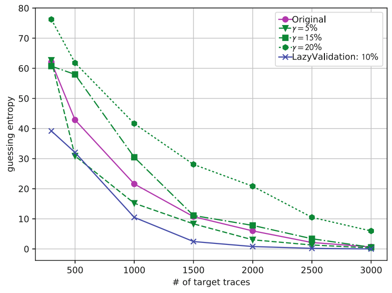


(f) misalignment = 25%

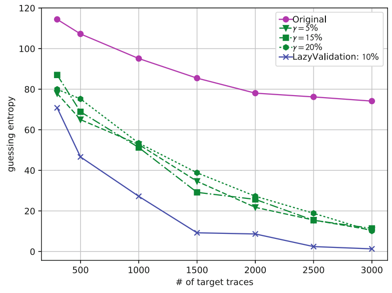
Fig. 4. The guessing entropies by varying the amount of deviation between profiling and target traces (in terms of misalignment), where software-implementation consists of 54 leaking points; 100 repetitions (to compute the guessing entropies) and 4000 profiling traces; using Gaussian templates



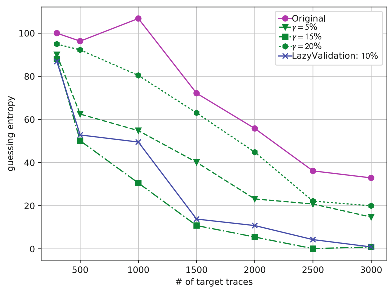
(a) no misalignment



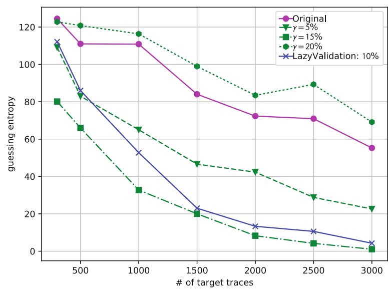
(b) misalignment = 5%



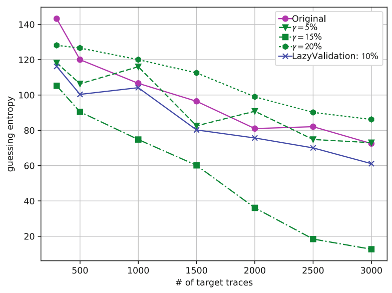
(c) misalignment = 10%



(d) misalignment = 15%



(e) misalignment = 20%



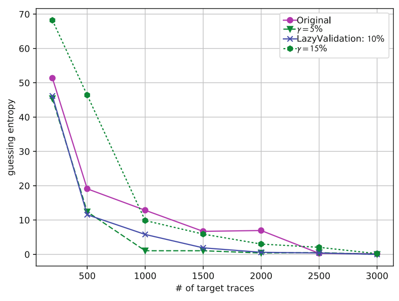
(f) misalignment = 25%

Fig. 5. The guessing entropies by varying the amount of deviation between profiling and target traces (in terms of misalignment), where FPGA-implementation consists of 20 leaking points; 100 repetitions (to compute the guessing entropies) and 5000 profiling traces; using linear-regression-based profiling

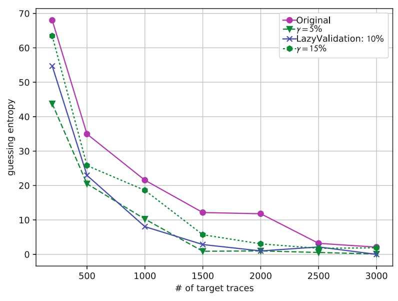
3.3 FPGA-Based Experiments

In hardware scenario, we target the AES implementation running on SAKURA-X board, whose traces contain 20 leakage points. Our attack strategy is similar to the one in software-based experiments, except for using linear-regression based profiling (in which the degree of power model is 1) from [19,25]. Compare to Gaussian template building, linear-regression based profiling can build up a model more efficiently with less number of measurements thus is more suitable to the FPGA scenario (with larger noise). In such scenes, we also conduct the experiments by adding misalignment between profiling and target traces as same as what we do in software experiments.

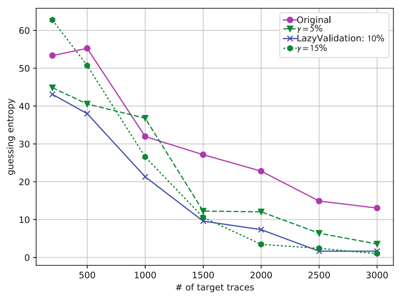
As shown in Fig. 5, the results of FPGA-based experiments is very similar to the software-based and simulation-based ones. Specifically, in the scenarios of high misalignment, the attacks without trace augmentation hardly distinguishes



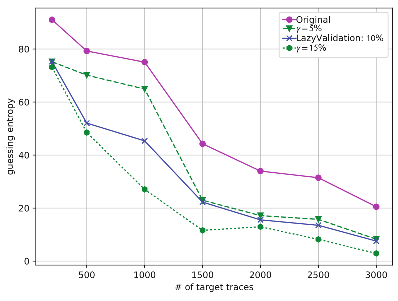
(a) no misalignment



(b) misalignment = 5%



(c) misalignment = 10%



(d) misalignment = 15%

Fig. 6. The guessing entropies by varying the amount of deviation between profiling and target traces (in terms of misalignment), where FPGA-implementation consists of 20 leaking points; 100 repetitions (to compute the guessing entropies) and 5000 profiling traces; using ridge-based profiling

between correct and incorrect keys, whereas the guessing entropies with trace augmentation still tend to zero.

The combination of trace augmentation with another known robust profiling method is also very interesting. Thus in Fig. 6 we present the guessing entropies of the attacks using ridge-based profiling (in which the degree of power model is 4) from [24] when combined with trace augmentation (and other experiments settings are same as the former one using linear regression based profiling). The result shows that our trace augmentation can also improve known robust profiling's performance.

4 Conclusion

In this paper, we show that trace augmentation based preprocessing can effectively improve the performance of profiled SCA, in both scenarios where the profiling device either deviates significantly from (or behaves close to) the target device. Further, we use a fast method called lazy-validation to obtain conservative but appropriate parameters. Finally, we provide simulation-based and practical experiments to confirm the effectiveness of our approach, and the former also indicates that the improvement of our approach (over other preprocessing techniques) depends on the correlation among points of each trace. We leave it as future work to explore other possible ways to augment the trace without using distortion, and whether such preprocessing strategies (if exist) can outperform trace augmentation.

Acknowledgment. This work is supported by the National Natural Science Foundation of China (Nos. 61472249, 61572149, 61572192, U1536103, 61402286, 61472250), the Major State Basic Research Development Program (973 Plan, 2013CB338004), the National Cryptography Development Fund MMJJ20170209, Industry & Education & Research Cooperation Program of Minhang District (2016MH310) and International Science & Technology Cooperation & Exchange Projects of Shaanxi Province (2016KW-038).

A The Impact of Augmentation Ratio C

Figure 7 shows the impact of augmentation ratio C in trace augmentation, and we can see that it is insignificant to the improvement.

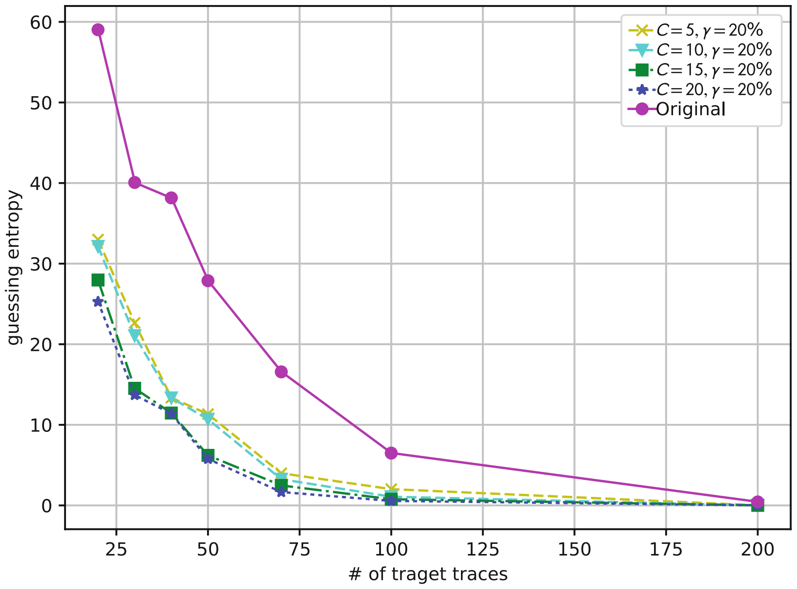
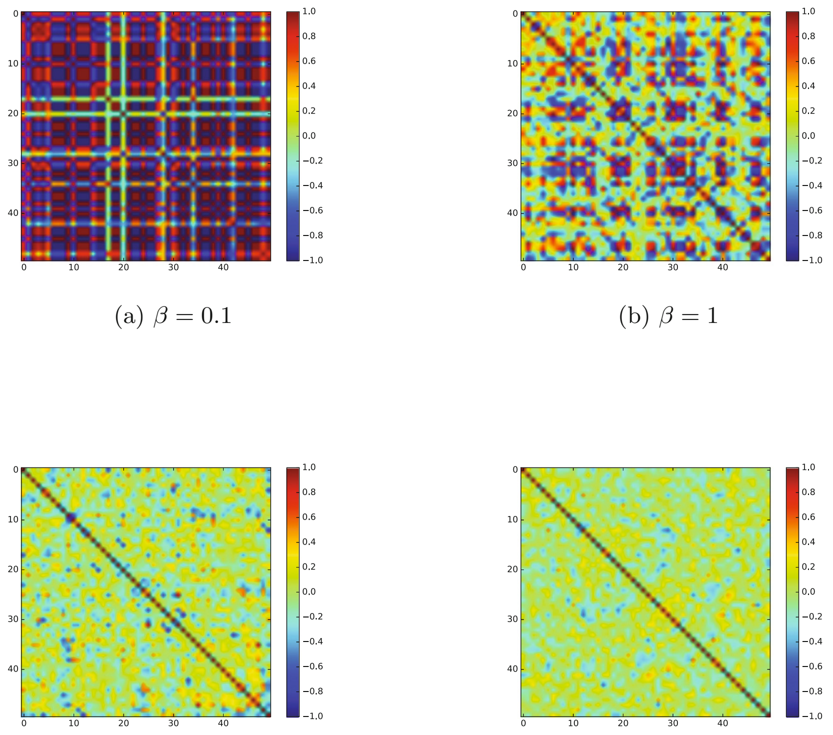


Fig. 7. The guessing entropies by varying augmentation ratio C in ideal scenario (no misalignment); simulation-based experiment containing 50 leakage points; 100 repetitions (to compute the guessing entropies) and 2000 profiling traces

B Correlation Matrices

'Vine' works in this way: off-diagonal values are derived from a beta distribution whose parameters satisfying $\alpha = \beta$, then perform a linear transform of these values to the interval $[-1.0, +1.0]$ (since beta distribution is defined on the interval $[0, 1]$). Correspondingly, values of correlation matrix are controlled by the single parameter β—higher β value corresponds to the less dependencies among points of each trace.

The correlation matrices of varied β value are provided as Fig. 8, colored according to correlations, from $[-1.0, +1.0]$. It is observed that correlations among points are enhanced as β decreasing.



(a) $\beta = 0.1$ (b) $\beta = 1$

(c) $\beta = 5$ (d) $\beta = 10$

Fig. 8. Correlation matrix (50 $\times$ 50) of each β parameter: 0.1, 1, 5, 10

References

1. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_1
2. Batina, L., Gierlichs, B., Lemke-Rust, K.: Differential cluster analysis. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 112–127. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_9
3. Batina, L., Hogenboom, J., van Woudenberg, J.G.J.: Getting more from PCA: first results of using principal component analysis for extensive power analysis. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 383–397. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_24
4. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Less is more - dimensionality reduction from a theoretical perspective. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 22–41. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_2
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3

6. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 253–270. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_17
7. Choudary, O., Kuhn, M.G.: Template attacks on different devices. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 179–198. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10175-0_13
8. Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: High-performance neural networks for visual object classification. CoRR abs/1102.0183 (2011)
9. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012, pp. 3642–3649 (2012)
10. Elaabid, M.A., Guilley, S.: Portability of templates. J. Crypt. Eng. **2**(1), 63–74 (2012)
11. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
13. Kocher, P.C., Jaffe, J., Jun, B., Rohatgi, P.: Introduction to differential power analysis. J. Crypt. Eng. **1**(1), 5–27 (2011)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3–6, 2012, Lake Tahoe, NV, USA, pp. 1106–1114 (2012)
15. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. J. Crypt. Eng. **5**(2), 123–139 (2015)
16. Lewandowski, D., Kurowicka, D., Joe, H.: Generating random correlation matrices based on vines and extended onion method. J. Multivar. Anal. **100**(9), 1989–2001 (2009)
17. Merino Del Pozo, S., Standaert, F.-X.: Blind source separation from single measurements using singular spectrum analysis. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 42–59. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_3
18. Sánchez, J., Perronnin, F.: High-dimensional signature compression for large-scale image classification. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011, pp. 1665–1672 (2011)
19. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
20. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: 7th International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, Scotland, UK, 3–6 August 2003, vol. 2, pp. 958–962 (2003)

21. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_26
22. Standaert, F.-X., Koeune, F., Schindler, W.: How to compare profiled side-channel attacks? In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 485–498. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01957-9_30
23. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
24. Wang, W., Yu, Y., Standaert, F.-X., Gu, D., Sen, X., Zhang, C.: Ridge-based profiled differential power analysis. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 347–362. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52153-4_20
25. Whitnall, C., Oswald, E.: Profiling DPA: efficacy and efficiency trade-offs. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 37–54. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_3
26. Whitnall, C., Oswald, E.: Robust profiling for DPA-style attacks. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 3–21. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_1

Author Index