Countermeasures to side-channel attacks and secure multi-party computation

Adrian Thillard

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'École normale supérieure

Countermeasures to side-channel attacks and secure multi-party computation.

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité Informatique

COMPOSITION DU JURY

M. PROUFF Emmanuel
Safran Identity & Security
Directeur de thèse

M. VERGNAUD Damien
École normale supérieure
Directeur de thèse

M. CORON Jean-Sébastien
Université du Luxembourg
Rapporteur

Mme. OSWALD Elisabeth
University of Bristol
Rapporteur

M. GILBERT Henri
Agence Nationale de la Sécurité
des Systèmes d'Information
Membre du jury

M. ISHAI Yuval
Technion
Membre du jury

M. ZEMOR Gilles
Université de Bordeaux
Membre du jury

Soutenue par **Adrian THILLARD**
le 12 décembre 2016

Dirigée par
Emmanuel PROUFF
et **Damien VERGNAUD**

Contents

Remerciements

La section que vous avez sous vos yeux sera, pour l'écrasante majorité d'entre vous, je le crains, la seule partie que vous lirez de tout ce manuscrit. En fait, même, il y a des chances pour que certains d'entre vous (pas forcément les plus égoïstes d'ailleurs), n'y fassiez qu'un rapide parcours visuel pour détecter la présence rassurante de votre nom, ainsi que d'un éventuel petit mot l'accompagnant. A cause de ma pudeur, ce petit mot, sachez-le, ne saura pas retranscrire avec perfection l'affection que j'ai pour vous. Si jamais, par mégarde, maladresse, malice, malveillance, méchanceté, ou malignité, vous pensez que je vous ai oublié, sachez qu'il n'en est rien et que vous êtes spécial. C'est pourquoi je'ai simplement gardé une place spéciale pour vous dans le cryptogramme apparaissant à la fin de ces remerciements. Afin de vous le prouver, vous pourrez venir me voir et je me ferai une joie de vous générer une clef *personnalisée*, que vous n'aurez qu'à *xorer* afin que vous puissiez découvrir par vous-même toute l'estime et l'amour que je vous porte.

Je pense que la façon la plus naturelle de remercier toutes les personnes qui ont compté pendant ces trois années est de procéder par ordre chronologique. C'est donc logiquement que je tiens tout d'abord à remercier mes parents, pour leur soutien indéfectible et pour tout ce qu'ils m'ont appris. La liberté qu'ils m'ont laissé est un véritable trésor, et les leçons qu'ils m'ont apporté (et m'apportent encore) m'ont considérablement enrichi. J'étends ces remerciements à l'ensemble de ma famille, et particulièrement à mon cousin Jérôme, avec qui j'ai partagé beaucoup d'expériences, et qui a su être à mes côtés dans de nombreux moments importants.

J'aimerais remercier mes professeurs de mathématiques de collège et de lycée, monsieur Forcet et monsieur Wybrecht, qui m'ont donné goût à leur matière et m'ont permis de m'orienter vers une discipline aussi amusante et fascinante.

Je remercie également mes amis rencontrés tout au long de mes années d'études (mais pas forcément sur les bancs), et avec qui j'ai gardé le contact pendant ces années: Mathieu (actuellement en camping à Azeroth), David (enfin monté dans la grande ville, prêt à réconcilier les ondes et les particules), Baptiste (sans le mode hyène, mais avec le costume de bucheron), Adrien (champion régional du lancer de table), Rodolphe (pour ses conseils diététiques et ses burgers), Fabien (qui m'aura appris à apprécier les travaux de Louis Pasteur et la fragilité des chaises), Cyril (qui vient régulièrement se perdre dans le froid au milieu de formations), Eloi (je pense que je n'ai rien le droit de leak ici sous peine de représailles), Alberto (qui a réussi l'année parfaite), Jean-Christophe (pour *tout*), Lionel (qui va finir par faire une indigestion de pommes !), Natalya (qui vit une belle aventure outre-atlantique), Irene (su temperamento es mejor que un sandwich), Laia (que es mucho más que un gran guía turística), Maxime (et papa écureuil... un truc bien cancer).

J'adresse également de sincères remerciements à une bande du passé dont j'ai eu l'occasion de recroiser quelques membres au bord de l'eau et au fil du temps: Victor, Déborah, Lucie, Agathe, Clara, Valentin, Hugo, Benjamin, Jilian, Thomas, Laurie et Cloé.

Je suis énormément redevable à mes encadrants de stages successifs, Nicolas et Laurent en seconde année de Master, ainsi évidemment que Christophe en première année, avec qui j'ai coécrit mon premier article, et auprès duquel j'ai passé trois mois très riches en enseignements, qui sont devenus le socle d'une partie de mes recherches.

Je remercie tous les gens que j'ai pu rencontrer à l'ANSSI, et tout particulièrement les membres passés et présents du laboratoire auquel j'appartiens: Thomas (avec un remerciement particulier pour ses conseils de lecture qui m'ont fait découvrir Céline et Kundera), Victor (ou les plusieurs personnes qui prennent son identité et se relaient pour faire des manips, de la recherche, et assurer une présence continue dans tous les festivals, boites de nuit, bars d'Europe, de Californie, du Nevada, d'Asie, sans jamais éprouver le besoin de dormir plus de 4 heures), Jean-Claude (qui a réussi à me faire comprendre les subtilités d'instuments tels que le biniou, ce qui n'est pas rien), Karim (avec qui j'ai partagé mon bureau de nombreuses soirées, bercées par nos écoutes -et de façon moins avouable, nos chants- de *the Great Gig in the Sky*), Guillaume (avec qui je partage désormais mon bureau, et qui aime bien une expression qui parle de table), Patrick (ou plutôt Docteur Patrick), Marc (qui m'a fait ouvrir les yeux sur le type de restaurants fréquentés par nos élus), Ryad (qui sait à peu près tout faire sauf se tromper ou proposer un burrito après 11h45), et Manuel (qui sait bien faire chauffer ses neurones). A l'ajout de ces membres, je veux adresser un énorme merci à Emmanuel, qui a passé quelques années parmi nous, et a accepté d'encadrer cette thèse. Sa disponibilité extrême, ses idées, son savoir, sa rigueur, et son amour du vin versé (parfois à côté du verre) ont rendu les nombreuses réunions agréables, les rédactions d'articles tout à fait supportables, et mon vécu de la thèse inoubliable. Je suis très heureux d'avoir pu faire ma thèse dans de telles conditions, et je remercie particulièrement les gens de l'ANSSI que j'ai eu le plaisir et la chance de cotoyer et avec qui j'ai pu travailler. Merci à Jean-Pierre, Jérôme, Yannick, Brice, Thomas, Aurélie, Jean-René, Jérémy, Joana, Henri, Valentin, Philippe, Pierre-Michel, Chaouki, José, Christophe, Benoît, Damien, Emmanuel, Eric, Patrick, Emmanuel, Arnaud, Arnaud, Arnauld, Mathieu, Anaël, Pierre, Alain, Mickaël, fkz, kutio, Lucas, Soline, Jonathan, Jérôme, Alexandre, Moïse, Eric, Coralie et Daniel et Yann. Je remercie également toute ma hiérarchie (là encore, passée et actuelle), pour avoir permis, voire même encouragé, mon lancement dans cette aventure. Merci donc à Eliane, Loic, Benjamin, José et Vincent.

J'aimerais adresser de sincères remerciements à tout le laboratoire de cryptologie de l'ENS, qui m'a accueilli pendant cette thèse. Je remercie particulièrement Damien pour avoir accepté de m'encadrer. Son soutien et ses compétences m'ont particulièrement impressionné pendant nos nombreuses discussions, et je pense avoir beaucoup appris à ses côtés. Je remercie les membres permanents (passés et présents, vous commencez à être habitués) du laboratoire: David, Michel, Vadim, Hoeteck, Céline, Hieu et Georg. Evidemment, je remercie chaleureusement tous les étudiants qui ont partagé l'open-space avec moi, et supporté mes bonjours qui duraient, parait-il, un peu longtemps. Je remercie donc Geoffroy (celui qui en fait joue vraiment du piano), Fabrice (l'oracle du laboratoire), Pierre-Alain (qui m'avait fichu une bonne trouille en regardant le site web que j'avais mis en place pour le challenge CHES...), Thierry (pour son sourire permanent), Houda (désormais partie dans des aventures nippones), Michele (che voglio la aqua), Rafael (qui se lève quand on lui dit bonjour, lui), Florian (qui trouve toujours plein de problèmes mais se plaint quand on lui demande de trouver un anagramme qui n'en est pas un), Thomas (le maître-ponceur, duc de la pétanque et prince des petits-fours), Alain (qui a partagé pas mal de déboires siultanément à moi, mais dont le sommet du talent est, il faut bien l'avouer, la réservation airbnb), Razvan (pour le voyage en train), Romain (qui n'aime pas mes suggestions de cadeau), Mario (whose look on *a lot* of things is particularly unique, always funny, and often appreciable), Sylvain (pour ses éclats de rire) Jérémy (désormais parti dans des aventures limougeaudes), Aurélien (vraiment, merci d'avoir apporté un jeu où je suis totalement imbattable, ça fait plaisir), Louiza (qui a une bonne place dans l'open space mais risque la regretter avec le bruit de l'ordinateur en face d'elle), Anca (cel mai bune dardeuse din Paris), Antonia (für alle seine Unterstützung, seine Freude und seine Musik), Sonia (qui m'a donné l'idée de faire des remerciements aussi longs, tellement j'ai pu lire le début de sa thèse de nombreuses fois), Dahmun (la plus belle rencontre que j'ai faite en conférence, et un guide exceptionnel si vous voulez visiter la Bretagne -ou compter un troupeau), et Pierrick (qui n'usurpe pas son titre d'âme du laboratoire, et sans lequel ce dernier perdrait une belle part de cohésion et d'amitié). Je remercie également Simon et Rémi, avec qui j'ai toujours eu des conversations très enrichissantes. J'adresse aussi un merci (et aussi un bon courage) aux personnes que je n'ai que trop peu vues (soit par ma présence au laboratoire trop faible, soit par leur arrivée trop tardive): Quentin, Balthazar, Pooya, Michele, et Julia.

Je voudrais remercier également le personnel administratif du département informatique, pour m'avoir accompagné lors des quelques démarches que j'ai eu à faire. Merci à Joëlle, Lydie, Nathalie, Michelle, Lise-Marie et Valérie.

Je remercie tous les gens que j'ai pu rencontrer grâce à la cryptologie depuis que j'ai commencé ma thèse. Ils sont très nombreux, mais j'aimerais remercier tout particulièrement pour tous les bons moments passés en conférence, en bar, en soirée, en montagne, en boite, ou au bord de la mer, Romain, Vincent, Elise, Tom, Colin, Tancrède, Nicolas, Julien, Eleonora, Jérôme, Jean-Christophe, Ange, Gilles, Luk, Yannick, Annelie, Sylvain, Hélène, Ronan, Renaud, Rina, Aurore, Frédéric, Pascal, Julia, Matthieu (qui est également un de mes chers co-auteurs, et qui fait sans doute partie, comme Victor, du gang des gens qui ne dorment pas), Pierre-Yvan, Philippe, Olivier, Guénaël, Julien, Céline, Cyril, Karim, Ninon, Pierre, Léo, Antoine, Damien.

Je tiens à remercier les gens que j'ai pu rencontrer au cours de ces années à Paris (ou pendant quelques voyages). Auprès de chacune d'entre elles, j'ai pu évoluer et apprécier une différente facette du monde dans lequel nous vivons. Merci à Eyandé (qui a mis la barre haut dès le départ), Jérémy (qui sait toujours trouver les mots justes), Divya (et sa pétillance), Simon (l'homme le plus cred du monde), Wuz (le meilleur modérateur du pire site), 0ver (et sa passion pour l'adansonia digitata), ivanlef0u (nunuche !), Pseeko (pour ses douces mélodies à la basse), Elmi (pour les folles parties de cartes), Djo, Jérémy, Diane, Paul (pour l'invention du livre de plomb et la découverte de bars parisiens étranges), Priscila (que amplió mi vida), Thania, Romain (qui est quand même très fort en quizz), Myriam (pour toutes les aventures musicales que nous avons vécues), Raphael, Raphael, Alexis, Charlene, Raphael et Miyu.

Je veux aussi remercier Jean-Sébastien Coron et Elisabeth Oswald pour avoir accepté de rapporter pour ma thèse, ainsi que Henri Gilbert, Yuval Ishai et Gilles Zémor pour avoir accepté de prendre part au jury. Je remercie également tous les gens venus assister à ma soutenance, et vous aussi, qui lisez ces remerciements.

Pour terminer, j'aimerais remercier Anne, pour tout son support pendant la rédaction de ce manuscrit, et pour avoir accepté les quelques sacrifices qui en ont découlé, tout en continuant de m'apporter un immense bonheur jour après jour.

J'aimerais ajouter ici quelques remerciements supplémentaires, pour une foule de gens qui m'ont accompagné pendant ces trois années, et dont la contribution ne peut pas être négligée. Malheureusement, je ne pourrais pas tous les citer, cependant, j'éprouve une gratitude démesurée pour, dans l'ordre:

Bob Dylan, Jean-Jacques Goldman, Akira Kurosawa, Leonard Cohen, Francis Ford Coppola, Harrison Ford, Stephen King, Jimmy Hendrix, Richard Wright, Alan Moore, Michel Gondry, Mick Jagger, Mylène Farmer, James Deen, Louis-Ferdinand Céline, Trey Parker, Johann Sebastian Bach, Daniel Day Lewis, Orson Welles, Peter Dinklage, Sid Meier, J. R. R. Tolkien, Akira Toriyama, Lindsey Stirling, Tom Morello, Matt Stone, Matthew Bellamy, Alain Chabat, Salvador Dalí, David Lynch, Diplo, Antoine Daniel, Gérard Depardieu, Sébastien Rassiat, Madonna, Quentin Tarantino, Voltaire, Louise Fletcher, Bruce Willis, Joe Pesci, Bernard Blier, Naomi Watts, Sam Mendes, Tim Commerford, Ray Manzarek, Robert De Niro, Johann Wolfgang von Goethe, Frederic Molas, Syd Barrett, Frank Miller, Roger Waters, Paul McCartney, Steven Spielberg, Lana del Rey, Patti Smith, Heath Ledger, Milan Kundera, Laurent Baffie, Zack de la Rocha, Rihanna, Jodie Foster, Ringo Starr, Leonard de Vinci, Marlon Brando, David Gilmour, Brad Pitt, Lady Gaga, Luc Besson, Uma Thurman, Arthur Rimbaud, Fauve $\neq$, Christopher Nolan, Jacques Brel, Chuck Palahniuk, John Lennon, Karim Debbache, Jim Morrison, Christopher Wolstenholme, Ludwig van Beethoven, Justin Theroux, Robby Krieger, Xavier Dolan, Matt Groening, Alfred Hitchcock, Orelsan, Christopher Lloyd, David Fincher, Anthony Hopkins, Shane Carruth, Stanley Kubrick, Thomas Bangalter, Joan Miro, Serge Gainsbourg, Charlie Watts, Nick Mason, Douglas Rain, Arthur C. Clarke, Keith Richards, Bob Marley, Tori Black, Jean-Luc Godard, Lexi Belle, Leon Tolstoï, Alan Walker, Christopher Walken, Michael Madsen, Ke$ha, Katy Perry, Ridley Scott, Eminem, Hector Berlioz, John Irving, Skrillex, Dominic Howard, Peter Jackson, George Harrison, Luke Rhinehart, Brian De Palma, Edgar Allan Poe, John Densmore, James Caan, Wolfgang Amadeus Mozart, Alexandre Astier, Anne-Louis Girodet, Milos Forman, Albert Camus, Friedrich Nietzsche, Pablo Picasso, Tim Burton, Terry Pratchett, Bruce Springsteen, Abella Anderson, Robert Zemeckis, Alejandro Iñárritu, Jack Nicholson, Martin Scorcese, Steeve Bourdieu, James Joyce, David Carradine, Jérémy Amzallag, David Bowie, Michel Hazanavicius, Brad Wilk, Emma Stone, Guy-Manuel de Homem-Christo , Faye Reagan, Felix Jaehn, John Travolta, Odieux Connard, Bill Wyman, Gringe, Antonín Dvořák.

Etant friand d'énigmes et de challenges, je vous propose finalement, si jamais vous vous ennuyez pendant de longues soirées d'hiver, ou bien que vous éprouvez le besoin de vous occuper lors d'un passage un peu long d'une présentation, le cryptogramme suivant à résoudre. Le gagnant pourra prétendre à un lot exceptionnel comprenant un exemplaire dédicacé de ce manuscrit et un regard impressionné.

79+1, 94+3, 33-5, 29-2, 104-4, 18-7, 150+3, 115-3, 112-1, 51-8, 27+6, 54+3, 87+2, 116+4, 22+5, 49-1, 96-1, 55-2, 108+5, 20+6, 99+11, 26+2, 10+3, 3-2, 66-3, 16+5, 146-2, 53+6, 125+2, 133+5, 7+4, 70-8, 80+2, 113+5, 98+2, 73-2, 138+5, 102+3, 89-1, 101-1, 56-3, 35-2, 59+3, 44+3, 129-1, 13-1, 83+2, 135-7, 147-2, 63+2, 57+5, 60+4, 130-1, 152-1, 82-1, 48-3, 139+7, 142-2, 118+3, 15-5, 93+3, 109-7, 69-4, 25+7, 21-2, 122+1, 128+3, 58-1, 1-2, 90+2, 6+3, 78+4, 105-6, 88+4, 119-4, 100-2, 111+6, 38+3, 4+2, 120+4, 123+3, 85+2, 46-5, 71-1, 23+2, 68-3, 19-4, 140+2, 43+7, 50+4

Chapter 1

Evaluation of Embedded Cryptography

Schätzen ist Schaffen: hört es, ihr Schaffenden!
Schätzen selber ist aller geschätzten Dinge Schatz und Kleinod.

Friedrich Nietzsche - *Also sprach Zarathustra*

1.1 History

Cryptosystems are present in a lot of devices used in everyday life, such as smart cards, smartphones, set-top-boxes, or passports. All those products embed cryptography for various purposes, ranging from the privacy of user's data in his phone, to the security of banking transactions. Nonetheless, implementing cryptographic algorithms in such constraint environments is a challenging task, and the apparition of *side-channel analysis* in the 90's [Koc96] showed that specific precautions should be taken. Indeed, specific emanations about the manipulation of variables can occur when such algorithms are performed. On embedded devices, these emanations are quite easy to observe, and may hence hinder the strength of the underlying cryptography. Recent works [GST14; GPPT15; GPT15; GPPT16; GPP+16] have illustrated that these phenomena can also be observed on larger devices, such as laptops or desktops.

To ensure reliability on the designer's and reseller's claims of security, guidelines and standards have been published by governments and economic interests groups. One of the earliest examples of such standardisation effort is the *Trusted Computer System Evaluation Criteria* (TCSEC) [Def85], often referred to as the *orange book*, released by the United States Government Department of Defence in 1983, and updated in 1985. This book was the first part of a whole collection of standards on computer security, named the *rainbow series* after their colourful covers. This standard defined four divisions A, B, C, D of decreasing level of security. The level achieved by the evaluated system was determined by a list of requirements on hardware and software protections and resilience against a vast class of attacks.

Inspired by this work, France, Germany, the Netherlands and the United Kingdom published in 1990 the *Information Technology Security Evaluation Criteria* (ITSEC), which was standardised by the European Union one year later [EC90]. This document introduced the term of *target of evaluation* (TOE) to design the part of the device subjected to a detailed examination. In particular, its *functionality*, *effectiveness* and *correctness* are studied. This time, the product can obtain one of six levels of security (E1 to E6), reflecting the requirements in terms of development and operational processes and environment.

In 1993, the *Canadian Trusted Computer Product Evaluation Criteria* (CTCPEC) was introduced by the Communications Security Establishment Canada. The goal of this standard is to build upon ITSEC and TCSEC to fix their respective shortcomings. In particular, the observation that the orange book put a strong emphasis on *confidentiality* of data and not much on *integrity* led the CTCPEC to include a part on the evaluation of mechanisms preventing unauthorised modifications.

1.2 Common Criteria

The TCSEC, ITSEC and CTCPEC standards were unified and superseded in 1999 by the *Common Criteria for Information Technology Security Evaluation* (Common Criteria, or CC), through the creation of the international standard ISO/IEC 15408 [ISO]. Even though this norm has been revised several times since its creation (at the time of writing, Common Criteria are in version 3.1 revision 4), their philosophy has stayed the same. Common Criteria define three entities around the life of a security product: the *designer*, which imagines and produces it, the *evaluator* (or Information Technology Security Evaluation Facility (ITSEF)), which tests the resilience of the product against attacks and the *certification body* (oftentimes a governmental organism - like ANSSI), which ensures the quality and pertinence of the results of the evaluation, and the *end user*, to which the tested device is sold.

The process in the common criteria setting starts by the designer's will to *certify* a product, in order to provide a certain assurance on its technical level, and, consequently, to be able to sell it to end users seeking for strong security properties. To this end, a request is registered by a certification body of its choice. The designer must specifically define the target of evaluation, and list its claimed security features. The document describing these features is called the *Security Target* (ST), and is formally constituted of a list of *Security Functional requirements* (SFR), which specify individual security functions. To ease the redaction of the ST, the CC propose a catalogue of standard SFRs, and several *Protection Profiles* (PP), which serve as guidelines. Precisely, a PP is a collection of security features, descriptions of use environments, threats, and requirements, stating the security problem for a given family of products. Examples of PPs include smart cards, firewalls, anti-virus, or trusted environment systems. The designer also claims, through the security target, the assurance level of the product which is the target of the evaluation.

In CC, seven *Evaluation Assurance Levels* (EAL) are defined, of increasing insurance of strength:

- *Functionally Tested* (EAL1): provides an evaluation of the TOE through basic analysis of the SFRs, supported by a functional testing and a simple penetration testing. EAL1 can be successfully conducted without assistance from the developer of the TOE.

- *Structurally Tested* (EAL2): provides an evaluation of the TOE through a vulnerability analysis demonstrating resistance against basic penetration attacks, evidence of developer testing, and confirmation of those tests. EAL2 requires the provision of a basic description of the architecture of the TOE.

- *Methodically Tested and Checked* (EAL3): provides an evaluation of the TOE on the same basis as EAL2, enhanced with a more complete documentation. In particular, EAL3 requires controls of development environments, and the furniture of a more complete architectural design of the TOE.

- *Methodically Designed, Tested and Reviewed* (EAL4): provides an evaluation of the TOE through a vulnerability analysis demonstrating resistance against enhanced penetration attacks. EAL4 requires the furniture of an implementation representation (for example, the source code) of all security functions.

- *Semi-formally Designed and Tested* (EAL5): provides an evaluation of the TOE through a vulnerability analysis demonstrating resistance against advanced penetration attacks. EAL5 requires semi-formal design descriptions of the TOE and a structured and analysable description of its architecture.

- *Semi-formally Verified Design and Tested* (EAL6): provides an evaluation of the TOE through a vulnerability analysis demonstrating resistance against high-level penetration attacks. EAL6 requires formal design descriptions of the TOE and its architecture, and controls the structured development process.

- *Formally Verified Design and Tested* (EAL7): provides an evaluation of the TOE on the same basis as EAL6, enhanced with an assurance that the TOE was formally designed. In particular, EAL7 requires a comprehensive analysis using formal representations and formal correspondence, as well as a comprehensive analysis of the TOE.

The process of evaluation is thoroughly described in the *Common Methodology for Information Technology Security Evaluation* ([CEM]). The evaluator is charged to verify the SFRs of the security target, according to the claimed Evaluation Assurance Level. To this end, the ITSEF verifies the claims through five *conformity* classes and one *attack* class. Each class is subdivided in one or several *families*, and the product is evaluated against each requirement corresponding to these families:

- ADV *Development*: provides information about the TOE. The knowledge obtained by this information is used as the basis for conducting vulnerability analysis and testing upon the TOE. The class encompasses requirements for structuring and representing the security functions at various levels and forms of abstraction. It is subdivided in 6 families: *Security Architecture* (ADV_ARC), *Functional specification* (ADV_FSP), *Implementation representation* (ADV_IMP), *TOE Security Functions internals* (ADV_INT), *Security policy modelling* (ADV_SPM), *TOE design* (ADV_TDS).

- AGD *Guidance documents*: provides the requirements for guidance documentation for all user roles. The description of all relevant aspects for the secure handling of the TOE is mandatory. The class also addresses the possibility of incorrect configuration of the TOE. The class is subdivided in 2 families: *Operational user guidance* (AGD_OPE), *Preparative procedures* (AGD_PRE).

- ALC *Life-cycle support*: establishes discipline and control in the processes of refinement of the TOE during its development and maintenance. This class allows for an accurate definition of whether the TOE is under the responsibility of the developer or user depending on the phase of its life. It is subdivided in 7 families: *Configuration Management capabilities* (ALC_CMC), *Configuration Management scope* (ALC_CMS), *Delivery* (ALC_DEL), *Development security* (ALC_DVS), *Flaw remediation* (ALC_FLR), *Life-cycle definition* (ALC_LCD), *Tools and techniques* (ALC_TAT).

- ASE *Security Target evaluation*: evaluated the soundness and internal consistency, and, if the ST is an instantiation of one or several protection profiles, that this instantiation is correct.

This class is subdivided in 7 families: *ST introduction* (ASE_INT), *Conformance claims* (ASE_CCL), *Security problem definition* (ASE_SPD), *Security objectives* (ASE_OBJ), *Extended components definition* (ASE_ECD), *Security requirements* (ASE_REQ), *TOE summary specification* (ASE_TSS).

- ATE *Tests*: provides assurance that the TOE Security Functions behave as described (in the specification, TOE design, and implementation representation). Two families of this class address the completeness of developer testing. The two others address the documentation and performance of those tests. The class is hence subdivided in 4 families: *Coverage* (ATE_COV), *Depth* (ATE_DPT), *Functional tests* (ATE_FUN), *Independent testing* (ATE_IND).

- AVA *Vulnerability assessment*: addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE. This is the only *attack* class of the Common Criteria framework. This class is subdivided in a single family: *Vulnerability analysis* (AVA_VAN).

For each family, the product is attributed a grade (from 1 to 6)[1] depending on the met requirements. The norm defines precisely the requirements needed to achieve each grade in each family. The obtention of a given Evaluation Assurance Level depends on all the grades obtained during the evaluation. Table 1.1 summarises the required grades for the obtention of the EAL levels.

Sometimes, an Evaluation Assurance Level can be *augmented*, that is, the requirements for a certain family can be upped to a greater grade than the one required by the level. For example, a product can be certified EAL4, with a grade in the AVA_VAN family of 5, instead of the required 3. The product is hence certified with an Evaluation Assurance Level 4, augmented with AVA_VAN 5. This is sometimes written by simply adding a + sign: the product is certified EAL4+. Note however that the + sign only signifies that the Evaluation Assurance Level is augmented, but do not tell which class or family has actually met higher requirements.

Based on its analyses, the evaluator redacts an Evaluation Technical Report (ETR), where its methodology is described, and which supports or denies the claimed security features of the product. This report is transmitted to the certification body, which validates or invalidates it, allowing to judge for the security of the TOE. If the features claimed by the TOE are not met, the designer can modify its design and recommendation guides to better resist the attacks or problems discovered by the evaluation. This process can be repeated as many times as wanted, until the evaluator claims the security of the product, and the certification body validates the report. The certification body hence issues a *certificate*, mentioning the Evaluation Assurance Level of the chip, which can then be presented by the designer to the end users. The ETR is however kept confidential.

This certificate may be valid in several countries, thanks to the existence of the Common Criteria Recognition Arrangement (CCRA). This arrangement is signed by 17 countries, that all have a certification body and can hence actually produce certificates: Australia, Canada, France, Germany, India, Italy, Japan, Malaysia, Netherlands, New Zealand, Norway, Republic of Korea, Spain, Sweden, Turkey, United Kingdom, United States. Moreover, 10 other countries recognise these evaluations: Austria, Czech Republic, Denmark, Finland, Greece, Hungary, Israel, Pakistan, Qatar, Singapore. The members of CCRA agreed on recognition of certificates claiming Evaluation Assurance Levels 1 and 2 as well as the family ALC_FLR. The members of the *Senior Officials Group Information Systems Security* (SOG-IS), a committee working together to define and update Protection Profiles and coordinate the standardisation of CC, composed by Austria, Finland,

[1] For some of the families, the maximum grade is less than 6.

Class	Family	Assurance Components by EAL						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Development	ADV_ARC		1	1	1	1	1	1
	ADV_FSP	1	2	3	4	5	5	6
	ADV_IMP				1	1	2	2
	ADV_INT					2	3	3
	ADV_SPM						1	1
	ADV_TDS		1	2	3	4	5	6
Guidance Documents	AGD_OPE	1	1	1	1	1	1	1
	AGD_PRE	1	1	1	1	1	1	1
Life-cycle support	ALC_CMC	1	2	3	4	4	5	5
	ALC_CMS	1	2	3	4	5	5	5
	ALC_DEL		1	1	1	1	1	1
	ALC_DVS			1	1	1	2	2
	ALC_FLR							
	ALC_LCD			1	1	1	1	2
	ALC_TAT				1	2	3	3
ST evaluation	ASE_CCL	1	1	1	1	1	1	1
	ASE_ECD	1	1	1	1	1	1	1
	ASE_INT	1	1	1	1	1	1	1
	ASE_OBJ	1	2	2	2	2	2	2
	ASE_REQ	1	2	2	2	2	2	2
	ASE_SPD		1	1	1	1	1	1
	ASE_TSS	1	1	1	1	1	1	1
Tests	ATE_COV		1	2	2	2	3	3
	ATE_DPT			1	1	3	3	4
	ATE_FUN		1	1	1	1	2	2
	ATE_IND	1	2	2	2	2	2	3
Vulnerability assessment	AVA_VAN	1	2	2	3	4	5	5

Table 1.1: Required grades for the obtention of EAL levels.

France, Germany, Italy, Netherlands, Norway, Spain, Sweden and United Kingdom recognise certificate until EAL4. The ex-members of the ITSEC consortium (France, Germany, the Netherlands and the United Kingdom) also recognise with each others certificates of any Evaluation Assurance Level.

Nonetheless, despite its will of genericity, Common Criteria are not the only evaluation and certification scheme. Indeed, a wide range of criticism has been directed towards this model. The main drawbacks are the length and cost of evaluation. Indeed, the whole process for the obtention of a CC certificate induces an overhead of at least 6 months, sometimes more depending on the aimed EAL, while the overhead costs range from several dozens of thousands euros to several hundreds of thousands euros. Several specific alternatives to CC have then be proposed. For example, one can cite the US's *Federal Information Processing Standard* (FIPS) 140-2, or France's *Certification de Sécurité de Premier Niveau* (CSPN), both of which aiming at a lower assurance of security, but faster and cheaper evaluations, and many private schemes, such as EMVCo certifications for banking smart cards and payment terminal, aiming at more specific tests and evaluations.

1.3 Penetration testing

Every evaluation process aiming at the issuance of a high level certificate should contain a phase of *penetration testing*, *ie.*, a phase where the evaluator tries to circumvent all the protections of the evaluated product to retrieve sensitive information (secret keys, personal data, etc.). This phase is critical, as its goal is to reflect the resilience of the product in the field, in the hands of malicious attackers. Consequently, it is often the longest and costliest part of the evaluation. In the Common Criteria, this phase is reflected by the family AVA_VAN.

In this thesis, we will mainly focus on the context of side-channel attacks on embedded cryptography. The general consensus on these attacks is that, by collecting a large enough number of observations, and considering a huge number of points of interest and a perfect knowledge of the target, it is *always* possible to retrieve information on the targeted sensitive data (see Chapter 2 for more concrete bounds). However, such ideal attacks would sometimes require unrealistic amounts of time and/or money to be actually performed. The goal of the penetration testing phase can then rather be seen as an estimation of how realistic these attacks are against the evaluated product.

In the Common Criteria, as well as in several other schemes, the realism issue is captured by the notion of *attack potential*. The grade attributed in the AVA_VAN family corresponds to the resilience of the product against an adversary with *basic*, *enhanced-basic*, *moderate*, or *high* attack potential. Roughly, an adversary with a basic (resp. enhanced-basic, moderate, high) attack potential is an adversary that can only perform at most basic (resp. enhanced-basic, moderate, high) attacks. This approach hence induces a hierarchisation of the attacks, and necessitates a metric to compare them. This metric is defined by the norm thanks to a *cotation table*, associating to each attack a score, reflecting its difficulty to perform. The goal of the table is to rate the difficulty of performing the attack against several criteria, and to give a global score by summing these ratings. The sum is directly translated to an attack potential: less than 9 points makes a basic attack, less than 14 points makes an enhanced-basic attack, etc. The rating criteria are:

- *elapsed time*: the time required to perform the attack, in days, months, or years
- *expertise*: the expertise of the attacker, from the layman to the necessity of having multiple experts in various domains
- *knowledge of the TOE*: the amount of information needed about the evaluated product, from its source code, to its datasheets, or precise layout
- *access to the TOE/window of opportunity*: the access opportunity to the TOE, usually meaning the number and variety of products that are needed to mount the attack
- *equipement*: the equipement needed to perform the attack, from simple pen and paper to multi-million specifically crafted devices.

In previous revisions of the Common Criteria, the cotation table was also divided in two parts: the *identification* part, aiming at reflecting the difficulty for an adversary to find the attack and the *exploitation* part, which reflected the difficulty to actually perform it once all details were known and published. This distinction is no longer applied in the latest revision.

It is interesting to note how the Common Criteria assessment of the dangerosity of an attack differs from the classical cryptanalytical approach. In the classical approach, cryptanalyses are considered according to their complexity classes, for example logarithmic, linear, polynomial, or (sub)exponential, in one or several parameters. Such complexity can be computed for time and memory, sometimes even very explicitly, allowing for an accurate evaluation of the requirements for an attacker to perform this attack. However, these evaluations alone do not tell if the attack is actually *feasible*, especially for very borderline bounds. Consequently the classical notion of complexity sometimes fail to give a practical answer to the security of an implementation.

Take for example the classic case of a bruteforce attack on the *Data Encryption Standard* (DES). The feasibility of the exhaustion of the 2^{56} possible keys depends on numerous factors that have to be taken into account. An obvious factor is the cost of devices built for this specific issue in a reasonable, practical time. As early as 1976, Diffie and Hellman estimated that such a machine would cost about $20 million [Des]. In 1998, the Electronic Frontier Foundation built such a machine for less than $250 thousand, a cost that was reduced to less than $10 thousands with the construction of COPACOBANA in 2006 [KPP+06]. Nowadays, such an exhaustion can be performed as a service, for only a few hundreds dollars[2]. This simple example shows that the time complexity of 2^{56} was, during all these times, reachable for different attackers. However, the complexity alone does not allow to judge for this feasibility. Simply stating the complexity of an attack also overlooks the specifics of the algorithm itself. Especially, the execution times of asymmetric algorithms are several orders of magnitude higher than the one of symmetric algorithms. How would one qualify an attack which necessitates the exhaustion of 2^{56} RSA keys, given that one execution of RSA algorithm takes around 2^{10} times longer than one execution of DES (see for example [GPW+04; STC07])?

The cotation table precisely aims at answering all those shortcomings, by translating each aspect of the attack into one or several practically measurable criteria. The notion of time complexity is replaced by the notion of time itself, and the specific devices which can be built correspond to points in the equipement criterion, that can be rated with more or less points depending on their cost. This results in an accurate grade reflecting the practical relevance of an attack, whereas the classical complexity approach would sometimes fail to do so.

1.4 Evaluation of smart cards and related devices

Smart cards and related devices are of particular relevance in the scope of this thesis. Indeed, the numerous constraints of size, consumption, and usability make them a prime target to the attacks and constructions described in this manuscript. We hence chose to further describe the specificities of their evaluation.

In fact, numerous documents have been produced by several working groups of the SOG-IS to define attack potentials depending on the product type. Consequently, specifications have been written for the evaluation of particular products, such as hardware devices with security boxes, or smartcards. We describe hereafter the specificities of the so-called *Application of Attack Potential to Smartcards* [Lib], that is, the specificities of the vulnerability analysis of such devices.

This document, along with the other documents produced by the SOG-IS, is presented as an interpretation of the Common Methodology for Information Technology Security Evaluation (CEM), based on the experience of smartcard CC evaluation and several working groups of industrials, academics, and certification bodies. Nonetheless, some aspects of the CEM are in fact modified by this document, such as the precise way of rating attacks, or new criteria. As such, this document supersedes the CEM on several points.

The main difference with the CEM is that, in the context of smartcards and similar devices, the distinction between identification and exploitation part is kept. The identification is defined as the effort required to create the attack and to demonstrate that it can be successfully applied to the

[2] A specific DES challenge was also broken in 1997 through distributed computing. It is however difficult to accurately assess the cost of this attack.

TOE. The exploitation is defined as the effort required to reproduce the attack on another TOE, while following a tutorial describing it in details. The final score of the attack is then computed by adding the scores obtained in both parts.

The rating criteria vary slightly from the CEM. Obviously, both the identification and the exploitation part are separately rated against all the criteria, each with a specific notation. We describe hereafter the criteria and their changes from the CEM:

- *elapsed time*: further granularity is introduced, distinguishing from *less than one hour*, *less than one day*, *less than one week*, *less than one month*, *more than one month*, or *unpractical.*

- *expertise*: several types of experts are defined, depending of their domain of knowledge. Examples are given, including expertise in chemistry, focused ion beam manipulation, chemistry, cryptography, side-channel analysis, reverse engineering, *etc.* To reflect the diversity of such a vast array of fields, a new level *multiple expert* is introduced, rating higher than the conserved levels *expert*, *proficient* and *layman.* This level is chosen when the attack requires expertise from different domains. It should be noted that for this level to apply, the expertise fields must be strictly different.

- *knowledge of the TOE*: any information required for the exploitation phase is not considered for the identification (this information should be in the description of the attack already). The criterion distinguishes between *public* information, *restricted* information such as guidance documents or administrative documents which can leak during the various phases of smartcard developments, *sensitive* information such as high or low level design of the architecture, *critical* information such as the source code or design, and *very critical hardware design.*

- *access to the TOE*: defined as the number of samples required to perform the attack. In some cases, the attack can succeed only with a small probability depending on the device, or even destroy some devices. The number of devices is then taken into account for the rating, depending on whether the attack requires *less than 10 samples*, *less than 30 samples*, *less than 100 samples*, *more than 100 samples*, or if it is *unpractical.*

- *equipement*: a list of tools is provided, ranging from UV-light emitter to atomic force microscope. Each of this tool is assigned a level, being *standard*, *specialised* or *bespoke.* These levels are completed by the possibility of using *no* equipement, or using *multiple bespoke* tools.

An important notion introduced by this document is the definition of *open samples*, and of *samples with known secrets.* Within the context of CC, it is sometimes possible for an ITSEF to have access of an exact copy of the TOE, where the evaluator can load specific software, or set certain variables at chosen values. Such copies are called open samples. The evaluator can for example use open samples to fix secret keys or random numbers used by cryptographic algorithms, hence allowing an easier characterisation of the behaviour of the device. This characterisation can then be used to perform a realistic attack on the TOE. A simple example of the use of open samples is the so-called *template attack*, which is described in Chapter 2. We also explain in Chapter 3 how the characterisation of a chip can be used to efficiently deduce the probability of success of various classical attack. The samples with known secrets cover the same notion, but consider devices where secret parameters are known, instead of chosen by the evaluator. The use of open samples and samples with known secrets in an evaluation is hence added as a (single) rating criterion. The rating for open samples models the difficulty for an attacker to have access to such a device, which could occur as a leak during the development process. The rating for samples with known secrets reflects the difficulty for an attacker to find a copy of the TOE where secrets can be deduced, either from leaks of specific documentations, or the use of copies in lower security schemes.

Chapter 2

Side-Channel Analysis

We don't need the key. We'll break in.
Zack de la Rocha - *Know your enemy*

2.1 Introduction

The first known example of side-channel analysis dates back to World War I. For cost reasons, telephone wires were built such that only one signal wire was deployed, while the ground was used for the return circuit. Some information was hence propagating directly in the ground. The German army used valve amplifiers and stuck electrodes in the trenches in order to intercept such compromising signals [Bau04].

In 1943, a Bell's researcher observed peaks on an oscilloscope while he was typing on a distant device. By studying these peaks, he was able to retrieve the pressed keys on the computer [NSA]. The discovery triggered the launch of the *TEMPEST* program by the National Security Agency, in an effort to study these compromising emanations. The first academic result on this topic appeared in a 1985 paper by Van Eck [Eck85], describing how to use these techniques to reconstruct a video signal.

In 1996, Kocher published the first public side-channel analysis of a cryptographic implementation [Koc96]. In this paper, he showed how the execution time of an implementation can leak information on the secret value being manipulated. In 1998, Kocher *et al.* showed how such information can be obtained by observing the power consumption of a device during a computation [KJJ99]. Following these seminal works, more and more side-channels were used to retrieve information on manipulated data: electromagnetic emanations [QS01], temperature [HS14], acoustics [GST14],...

2.2 Principle

Most cryptosystems are now built in accordance to Kerckhoffs's principle: the construction of the whole system is public, and its security only resides in a small secret parameter known as the *key*.

In classical cryptanalysis hence, following this principle, an attacker tries to recover the secret key knowing the algorithm description and some inputs and/or outputs of this algorithm.

This approach however fails to take into account the *implementation* of the targeted cryptographic algorithm. The algorithm is indeed seen as a black box, in the sense that no internal variable manipulated during the execution can be observed. Nonetheless, the implementation of the algorithm can have a tremendous and devastating effect on the security of such algorithms, as illustrated by the examples in Section 2.1.

Side-channel analysis captures the possibility for an attacker to get information about the internal variables manipulated by the algorithm. A side-channel *attack* is an attack where physical observations about the device are used to recover a secret parameter manipulated by it. Oftentimes this parameter is the key itself.

A side-channel attack can be described as the succession of four distinct but related phases:

1. *Sensitive data identification.* The attacker studies the cryptographic algorithm and its implementation to find a *sensitive* intermediate value v_k. A sensitive value is a value that can be expressed as a deterministic part of the plaintext and a guessable part $k^\star$ of the secret key (for example, an Sbox output). Note that the identification of a sensitive value can either be the result of a careful analysis of the algorithm or the consequence of observations about the device behaviour. In this last case, several statistical techniques can be used, such as *signal-to-noise* ratio or *t-test* approaches.

2. *Leakage collection phase.* The attacker observes the device behaviour during the manipulation of the identified sensitive variable. Using a particular setting depending on the physical observable of his choice, he collects several *leakages* $(\boldsymbol{\ell}_{k^\star,i})_i$ characterizing the manipulation of the data, while being provided a constant key $k^\star$ and various plaintexts $(p_i)_i$.

3. *Hypotheses construction.* The attacker exhausts all possible values for $k^\star$. For each hypothesis k and for each plaintext p_i, he computes the corresponding sensitive variable $(v_{k,i})$ hypothetically manipulated by the device. Then, the attacker chooses a leakage model function $\mathtt{m}$, to map the value of the sensitive data towards the estimated leakage. He hence obtains $(h_{k,i})_i$, where for every i, we have $h_{k,i} = \mathtt{m}(v_{k,i})$.

4. *Distinguishing.* The attacker compares the leakages $(\boldsymbol{\ell}_{k^\star,i})_i$ obtained in the second phase with all hypotheses $(\mathtt{m}(v_{k,i}))_i$ he constructed in third phase. This comparison is done using some *statistical distinguisher* and lays the most likely value for the used key.

According to this description, side-channel attacks can hence differ on the *targeted variable*, the *measurement setup*, the *leakage model*, and the *statistical distinguisher*.

Targeted variable

The targeted variable heavily depends on the targeted algorithm. It must be small enough to allow an exhaustion of all its possible values in the third phase. Usually, it corresponds to an information depending on both the plaintext and a small part of the key. Its size is typically less than 32 bits. For example, the typical targeted value in an AES implementation is the 8-bits output of the first substitution table. The set of all guessable key hypotheses is denoted $\mathcal{K}$.

Measurement setup

The measurement setup heavily depends on the chosen physical observable. However, the general setting is often similar, and is illustrated in Figure 2.1. In this figure, the targeted device is a smart card.

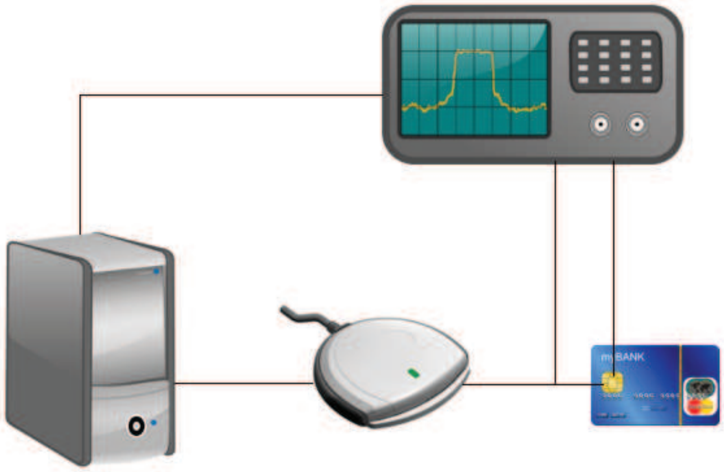


Figure 2.1: Example of measurement setup. A smart card is plugged in a reader, driven by a computer. An oscilloscope monitors both the communications and physical emanations of the card. The computer records the observations captured by the oscilloscope, and post-processes them.

This setup allows an attacker to obtain measurements of the processing of the cryptographic algorithm. Nonetheless, physical phenomena and/or countermeasures can induce desynchronizations between different executions. The measurement setup hence also include a *post-processing* of the leakages. The objective of this part is twofold: first, to ensure a *synchronization* of all measurements, and secondly, to identify *points of interest*, *ie.* points that actually correspond to the manipulation of the targeted variable. It is indeed interesting to note that, due to physical phenomena and/or countermeasures, relevant information about the sensitive value can occur at different points of the trace.

The synchronization can be done using several methods, which mainly consist in comparing measurements with one another. This comparison are oftentimes customly made in practice, depending on the shape of the curves themselves. Nonetheless, statistical methods are also used, such as auto-correlation or difference of means (see for example [HNI+06; RCC+08]).

The identification of points of interest can be done using several statistical tools. The computation of a *signal-to-noise* ratio for each point is computed, and dimension reduction techniques such as *principal component analysis* [Jol86].

Leakage model

The observed physical phenomena can be explained by the impact of different physical causes. An often exploited source is the switching gates of an electronic circuit, which leak information about the processed internal variables through power consuption and electromagnetic radiation. The nature of this phenomenon can give to an attacker an accurate relation between the manipulated data and the physical observation. Such a relation is called a leakage model. Most commonly used leakage models are the *Hamming weight* model and the *Hamming distance* model. In the Hamming weight model, it is assumed that the physical observation depends on the number of set bits of the data being processed. In the Hamming distance model, it is assumed that the physical observation depends on the Hamming weight of the XOR between two data being successively processed. The idea capture that both of these models is that each bit leaks information independently, through the charge or decharge of its corresponding gate. More complexes models are sometimes used, which can consider cross-impacts on those charges, considering that physical effects occuring on a gate can also impact its neighbours. Actually, the modelisation of these behaviors is a complex issue, and has been the subject of several researches. We describe in Section 2.4 a wider state-of-the-art on this subject.

Statistical Distinguisher

As highlighted by our description of an attack in Phase 4, the notion of statistical distinguisher is essential in side-channel analysis. We must hence introduce some notations from the field of statistics and probabilites in order to allow for a formal explanation. In the following, calligraphic letters, like $\mathcal{X}$, are used to denote definition sets of random variables denoted in capital letters, like X. Any realization of a random variable on this set denoted in lowercase, like x. The probability of an event ev is denoted $P[ev]$. The size of the collection of inputs $(p_i)_i$, and equivalently of leakages $(\boldsymbol{\ell}_i)_i$ is denoted by n. Each leakage $\boldsymbol{\ell}_i$ is moreover defined as a vector of a finite integer t points.

The statistical distinguisher takes as input the leakages $(\boldsymbol{\ell}_i)_i$ obtained in Phase 2 and the hypotheses $(h_{k,i})_i)$ obtained in Phase 3. It outputs a *score* for each key hypothesis k, which measures how much the associated collection matches the leakages. In order to allow for the distinguishing, an order needs to be defined on the scores. Most of the time, the scores take values in $\mathbb{R}$, with its standard order. These distinguishers come from the world of statistics or machine learning. For example, we can cite the linear correlation coefficient [Pea95], the Kullback–Leibler divergence [KL51], the difference of means [Fis20] or the maximum likelihood [Fis22].

2.3 Examples of side-channel attacks

Several types of side-channel attacks have been introduced in the last two decades. Most of them are agnostic with regard to the targeted variable and measurement setup. We propose hereafter to shortly describe some of the most commonly used side-channel attacks:

Template analysis

Template analysis was introduced in 2002 by Chari, Rao and Rohatgi [CRR03]. The attacker is assumed to be able to perform encryptions with the secret key of his choice on the targeted device, or a similar one [CK14]. Using this ability, he collects several leakage traces and uses them to extract as many meaningful statistical moments as possible for each possible key. Specifically, a *Gaussian model approach* [Tre68; CRR03] is usually performed, considering a Gaussian noise and the need to estimate statistical moments up to the second order. For each sensitive value v_k, the adversary is hence able to compute the associated leakage distribution (also called *template* in this context) $\mathcal{D}_k$ composed of a mean vector $\boldsymbol{\mu}_k$ and a covariance matrix Σ_k.

The model function $\mathtt{m}$ of a template analysis simply maps back every value $v_{k,i}$ to its estimated leakage distribution $\mathcal{D}_k$. A *maximum-likelihood* approach is hence performed.

In the Gaussian model, for a single observation (*ie.* when $n = 1$), the score of the hypothesis k is given by the *likelihood*:

$$\mathrm{ML}(\boldsymbol{\ell}_i, h_{k,i}) = \mathrm{ML}(\boldsymbol{\ell}_i, \mathcal{D}_k) = \frac{1}{\sqrt{(2\pi)^t det(\Sigma_k)}}\, e^{-\frac{(\boldsymbol{\ell}_i-\boldsymbol{\mu}_{\boldsymbol{k}})^\top(\Sigma_k)^{-1}(\boldsymbol{\ell}_i-\boldsymbol{\mu}_{\boldsymbol{k}})}{2}}. \tag{2.1}$$

Furthermore, in the case where several observations are made, the score of the hypothesis k is given by:

$$\mathrm{ML}((\boldsymbol{\ell}_i)_i, (h_{k,i})_i) = \prod_i \mathrm{ML}(\boldsymbol{\ell}_i, h_{k,i}). \tag{2.2}$$

Remark 2.3.1. The computation of $\mathrm{ML}((\ell_i)_i, (h_{k,i})_i)$ involves the product of several small values. When the number of observations is too high, this score can hence be so small that it lies below most systems precision. To avoid such a problem, which would cause the impossibility of computing a maximum-likelihood, this score is often replaced in practice by the equivalent *log-likelihood*, *ie.* its logarithmic value. Consequently, the score is given by:

$$\text{log-ML}((\boldsymbol{\ell}_i)_i, (h_{k,i})_i) = \sum_i \text{log-ML}(\boldsymbol{\ell}_i, h_{k,i}), \tag{2.3}$$

where $\text{log-ML}(\boldsymbol{\ell}_i, h_{k,i}) = \log(\mathrm{ML}(\boldsymbol{\ell}_i, h_{k,i}))$.

Linear regression analysis

Linear regression analysis was introduced in 2005 by Schindler, Lemke and Paar [SLP05].

The linear regression analysis (similarly to all the following examples of side-channel attacks) performs on *univariate* leakages, that is, leakages considering only one point. Consequently, the attack is either performed sequentially on each point of the acquired leakage, or the leakages are reduced to only one point, which is done during the post-processing part of the measurement setup.

The dimension of every vector $\boldsymbol{\ell}_i$ being $t = 1$, the collection of leakages can be seen as a n-dimensional vector $\boldsymbol{\ell}$, of coordinates $(\ell_i)_i$.

The statistical distinguisher makes extensive use of linear algebra: it performs linear regressions of the leakage against functions of the hypotheses' bits, and compares the goodness of fit of these regressions. Specifically, an attacker chooses a *regression basis* of functions $(g_j(\cdot))_j$. The leakage of an hypothesis $v_{k,i}$ is then represented as the polynomial $p(v_{k,i}) = \sum_j \alpha_j g_j(v_{k,i})$, where $(\alpha_j)_j$ are coefficients in $\mathbb{R}$. The goal of the linear regression is to find the collection of coefficients $(\alpha_j)_j$ such that the values $(p(v_{k,i}))_i$ are as close as the observed leakages $(\ell_i)_i$ as possible. To perform this regression, for each k, the functions $(g_j(v_{k,i}))_j$ are evaluated, and form an n rows matrix M_k.

The score for k is then given by the goodness of fit:

$$\mathrm{LRA}((\boldsymbol{\ell}_i)_i, (h_{k,i})_i)) = \mathrm{LRA}(\boldsymbol{\ell}, M_k)) = \frac{||\boldsymbol{\ell} - M_k((M_k)^\top M_k)^{-1}(M_k)^\top \boldsymbol{\ell}||}{\frac{1}{n}\sum_{i=1}^n (\ell_i - \bar{\ell})^2}, \tag{2.4}$$

where $||\cdot||$ denotes the Euclidean distance, n denotes the number of measurements, and $(\bar{\ell})$ denotes the mean $\frac{1}{n}\sum_i \ell_i$ of $\boldsymbol{\ell}$. It is interesting to note that the vector $((M^k)^\top M^k)^{-1}(M^k)^\top \boldsymbol{\ell}$ gives the coefficients $(\alpha_j)_j$ of the regression of the leakages on the basis chosen by the attacker, with respect to hypothesis k.

Correlation power analysis

Correlation power analysis was introduced in 2004 by Brier, Clavier and Olivier [BCO04]. The model function maps the hypothetic sensitive variable towards an element of $\mathbb{R}$. Usually, the chosen function is the Hamming weight of this value, but several other model functions can be used. The statistical distinguisher is the Pearson linear correlation coefficient. Denoting by $\bar{\ell}$ (resp. $\bar{h}_k$) the mean $\frac{1}{n}\sum_i \ell_i$ (resp. $\frac{1}{n}\sum_i h_{k,i}$), the score is given by:

$$\mathrm{CPA}((\ell_i)_i, (h_{k,i})_i)) = \frac{\frac{1}{n}\sum_{i=1}^n (\ell_i - \bar{\ell}) \cdot (h_{k,i} - \bar{h}_k)}{\sqrt{\frac{1}{n}\sum_{i=1}^n (\ell_i - \bar{\ell})^2}\sqrt{\frac{1}{n}\sum_{i=1}^n (h_{k,i} - \bar{h}_k)^2}}. \tag{2.5}$$

Differential power analysis

Differential power analysis was introduced in 1999 by Kocher, Jaffe and Jun [KJJ99]. The statistical distinguisher is the difference of means. In this context, the values taken by the hypotheses are reduced to a set of cardinal 2. Without loss of generality, we denote these two possible values 0 and 1. Usually, the model function maps the sensitive variable towards the value of one specific bit (for example, its least significant bit). Using these notations, and the same as introduced for the correlation power analysis, the score is hence:

$$\mathrm{DPA}((\ell_i)_i, (h_{k,i})_i)) = \frac{\sum_{i|h_{k,i}=0} \ell_i}{\#(i|h_{k,i}=0)} - \frac{\sum_{i|h_{k,i}=1} \ell_i}{\#(i|h_{k,i}=1)}. \tag{2.6}$$

Remark 2.3.2. Is is argued in [DPRS11] that the differential power analysis, correlation power analysis and linear regression analysis are not only asymptotically equivalent when the number of observations increase, but can also be rewritten one in function of the other, only by refining the used model leakage. Specifially, the differential and correlation power analysis can be seen as specific cases of linear regression analysis.

Simple power analysis

Simple power analysis was introduced in 1996 by Kocher [Koc96]. It can be seen as an implicit template analysis, where only a very limited number of templates -typically two- is used. Instead of collecting these patterns by performing encryptions with a known secret, the attacker is supposed to know them beforehand, or at least distinguish between them, because of their small number and simplicity.

2.4 Problem Modeling

State of the art

The study of the effectiveness of side-channel attacks and their countermeasures requires a formal framework. In 2003, Ishai, Sahai and Wagner introduced in [ISW03] the *d-probing model*, where an attacker can learn the exact value of a bounded number d of intermediate values during a computation. Furthermore, they proposed a method to transform any black-box circuit implementing a function to an equivalent circuit secure in this model. This paper proposed a method to build *private circuits* resistant in such a model, at a cost of a size blow-up quadratic in d. Nonetheless, even though this paradigm is convenient to write security proofs, it does not accurately represent a side-channel attacker. Indeed, this setting supposes that the attacker knows the exact value of only the d targeted intermediate variables, and fails to capture the whole information available. In particular, to achieve a good understanding of side-channel analysis, it is necessary to define a model that captures all the information that is leaked to the adversary.

The first general approach towards this understanding is the *physically observable cryptography* framework introduced by Micali and Reyzin in [MR04], which allows to represent any physical implementation. A very important axiom in this model is that *only computation leaks information.* This founding work paved the way for subsequent works on the resilience of cryptographic implementations against side-channel attacks, in particular the seminal paper written by Dziembowski and Pietrzak [DP08], which introduced the notion of *continuous leakage model* to the world of side-channel, that is, the assumption that only a bounded amount of information about the manipulated variable can leak during a given period. In this model, Dziembowski and Pietrzak were moreover able to construct a particular cipher proven to be resilient against any side-channel attack, using a pseudo-random number generator. However, the overhead (in terms of computation time, memory, and randomness) induced by this construction is too high to be considered as practical. Moreover, as argued by Standaert in [Sta11], the continuous leakage model is too theoretical, as it encompasses a very strong and unrealistic attacker. Specifically, the model allows the attacker to arbitrarly define the leakage function, even if it could revert the cryptographic properties of the algorithm itself. The paper [FRR+10] proposed an alternative approach, by restraining the choice of the leakage model to functions in the complexity class AC^0, *ie.* which circuit is of constant depth and polynomialy sized in the number of inputs. Sadly, this class is too small to actually describe all practical attacks. For example, the parity function is not in AC^0 [FSS84]. Another approach proposed in the same paper is a slight variation on the probing model, considering a possible error on each read bit with a fixed probability. It is nonetheless unclear how realistic this leakage model fits the physical reality of power and/or electromagnetic leakages.

A more realistic approach was proposed in [CJRR99] by Chari *et al.*, introducing the *noisy leakage model*. This model considers that the physical observable is a noisy observation of a deterministic function of the sensitive variable. This is particularly relevant in the practical case, as it seems to accurately capture physical phenomena induced by both the device architecture and the attacker's measurement setup [MOP07]. This work moreover focuses on the case where the noise is Gaussian. This noise model is generally considered to be a good approximation of the reality. Indeed, the central limit theorem ensures that the sum of noises caused by enough different sources converges towards a Gaussian distribution. The paper [CJRR99] however restrains its study to the case where the sensitive value is only one bit, shared as a sum of d bits (cf. 2.5.1). Nonetheless, this model allows to formally evaluate the effectiveness of a side-channel attack in this context. Precisely, for any side-channel attack to succeed in distinguishing the secret bit with probability α, the number of measurements n is lower bounded, by a value depending on the standard deviation σ of the Gaussian noise and the number d of shares:

$$n > \sigma^{t+4\frac{\log\alpha}{\log\sigma}}. \tag{2.7}$$

This work was generalized by Prouff and Rivain in 2013 [PR13] by getting rid of the single-bit limitation, and allowing for arbitrary noise. This study extends to the analysis of any affine function or multiplication. Even more, these results can be composed to the protection of a *whole* cryptographic algorithm, by assuming the presence of *leak-free gates ie.* components allowing to refresh the sharing of a variable. These gates allow them to compose several secure operations while maintaining a security all along the execution of the algorithm. Furthermore, this construction allows them to prove information-theoretical results about the security of the obtained circuit. For any manipulated variable X, the authors model the leakage obtained by an attacker by a non-deterministic function $L(X)$. This non-deterministic part can be represented in several ways. This modeling can be seen as a generalization of an often chosen model: the *additive noisy leakage*, where $L(X)$ is written as the sum of a noise B and a deterministic function φ of the manipulated variable X:

$$L(X) = \varphi(X) + B. \tag{2.8}$$

In [PR13], the authors choose to model the noise of a leakage by the *bias* introduced on the distribution of the manipulated random variable. For a manipulated variable X, and a measured leakage $L(X)$, the bias $\beta(X|L(X))$ of X given $L(X)$ is defined as the expected statistical distance between X and $X|L(X)$:

$$\beta(X|L(X)) = \sum_y P[L(X) = y] \cdot \Delta(X, (X|L(X) = y)), \tag{2.9}$$

where $\Delta(X, X|L(X) = y) = ||P_X - P_{X|L(X)=y}||$, with $||\cdot||$ denoting the Euclidean distance, and P_X (resp. $P_{X|L(X)=y}$) denoting the vector $(P[X = x])_x$ (resp. $(P[X = x|Y = y])_x$). The bias is an information metric between X and $L(X)$. Note that if X and $L(X)$ are independent, the bias is null. It is deeply related to the mutual information MI between the random variables X and $L(X)$, that is the difference between the Shannon entropies [Sha48] of X and of $X|L(X)$.

Indeed, by assuming a uniform distribution of X in a set of cardinality N, the mutual information between the sensitive variable X and its leakage $L(X)$ can be bounded:

$$\mathrm{MI}(X, L(X)) \leq \frac{N}{\ln 2}\beta(X|L(X)). \tag{2.10}$$

This property is the cornerstone of the proofs in [PR13]. Assuming that the designer of a circuit can control a noise parameter ω, and that, at any point of the execution, the bias between the manipulated variable and the leakage never exceeds $\frac{1}{\omega}$ (this assumption defines the so-called $\frac{1}{\omega}$-*noisy leakage model*), the authors show that the information an attacker can get by looking at a whole execution is upper bounded by $\omega^{-(d+1)}$.

The approach in [PR13] suffers from several shortcomings: it requires the existence of leak-free gates, the attacker can only access to randomly generated inputs, and the proof method is hard to generalize to new protection schemes. The next year, Duc *et al.* solved these shortcomings in [DDF14], and proved a reduction from the noisy leakage model used in [PR13] to the probing model used in [ISW03], in which proofs are more convenient. Specifically, they proved that for any attacker in the $\frac{1}{\omega}$-noisy leakage model introduced in [PR13], there exists an attacker in the d-probing model, where d is a value linear in $\frac{1}{\omega}$. This is of particular interest, since it implies that any circuit proven secure in the d-probing model is also secure in the $\frac{1}{\omega}$-noisy leakage model for some ω. However, the expression of d as a linear function of $\frac{1}{\omega}$ implies large constants, which render the obtained security impractical.

The link between these models have further been studied in [DFS15b; DFS15a]. In particular, it is shown in these papers that, if a circuit is secure in the d-probing model, then the probability of distinguishing the correct key among $|\mathcal{K}|$ hypotheses is upper bounded by $|\mathcal{K}| \cdot 2^{-d/9}$, if the leakage $L(X)$ satisfies, at each time:

$$\mathrm{MI}(X, L(X)) \leq 2 \cdot (|\mathcal{K}| \cdot (28d + 16))^{-2}. \tag{2.11}$$

These results imply that any proof of security obtained in the probing model can be transcribed as a proof of security in the noisy leakage model. Namely, if a construction is proven secure in the probing model (where the amount of observations an adversary can have is bounded), then there exists a bound on the amount of information one can have by observing the whole computation in the noisy leakage model. This is particularly interesting, since proofs in the probing model are easier.

We now formally describe a side-channel attack based on the noisy leakage model, which closely corresponds to a practical setting. This description is based on the one made in [PR13].

Algorithm

An algorithm is modeled by a sequence of *elementary calculations* $(C_i)_i$ implemented by Turing machines possessing a set of adresses called the *state*. An elementary calculation reads its input and writes its output on the state. When an elementary calculation C_i is invoked, its input is written from the state to its input tape, then C_i is executed, and its output is written back to the state.

Physical implementation

Similarly to an algorithm, its physical implementation is modeled by a sequence of *physical elementary calculations*. A physical elementary calculation is simply an elementary calculation C_i associated with a *leakage function* $L_i(\cdot)$. A leakage function is defined as a function taking two parameters: the first one corresponds to the value of the part of the state accessed by its corresponding elementary calculation, and the second one is a random chain modeling the leakage noise. Denoting $\mathcal{I} = (C_i, L_i)_i$ the physical implementation of an algorithm, each execution of $\mathcal{I}$ leaks the values $L_i(Z_i, r_i)_i$, where Z_i designs the part of the state accessed by C_i before its execution and r_i the corresponding random chain. By definition, the different r_i are supposed mutually independent. For the sake of simplicity, we will often omit the random chain and write $L_i(Z_i)$ the leakage corresponding to the value Z_i. In this sense, $L_i(Z_i)$ can be seen as the result of a *probabilistic* Turing machine. When only one variable is considered, we will omit the index i and simply write $L(Z)$.

Noisy leakage model

In the noisy leakage model, the leakage $L(Z)$ occuring during the manipulation of an intermediate value Z is modeled as a non-deterministic function of Z. In the remaining of this thesis, we will often consider a particular case, corresponding to the additive noisy leakage model, as described in Equation 2.8. Even more specifically, we will model $L(Z)$ as:

$$L(Z) = \varphi(Z) + B, \tag{2.12}$$

where φ is an unknown deterministic function depending on the architecture and where B is a Gaussian noise of null mean and of variance σ^2: $B \sim \mathcal{N}(0, \sigma^2)$. We will often instanciate, φ as the Hamming weight or Hamming distance. Note that, according to this definition, the leakage on variable Z only depends on Z, though φ could somehow capture information about previous states. This is a slight imperfection of the model. This simple model will be used as a basic brick of our study of side-channel attacks effectiveness in Chapter 3.

Moreover, the distribution $f_{L(Z)}$ of the leakage $L(Z)$ is a *Gaussian mixture* sastisfying:

$$f_{L(Z)} = \sum_{\epsilon \in \mathrm{Im}(\varphi)} P[\varphi(Z) = \epsilon]\ \mathcal{N}(\epsilon, \sigma^2). \tag{2.13}$$

We will use this observation in the study of the particular case of maximum-likelihood in Section 3.6.

2.5 Countermeasures

One can distinguish two main families of countermeasures against side-channel analysis. The first one, called *masking*, modifies the algorithm in order to make the sensitive variables manipulated independent from the secret key. The second one, called *hiding*, covers the processing of the algorithm by noise in order to make the exploitation of the observed signal too difficult.

2.5.1 Masking

A brief history

The first countermeasure against side-channel analysis actually appeared alongside the first attack in the seminal paper of Kocher [Koc96]. The idea is to introduce randomness in the computation of the algorithm, in order to *mask* the sensitive data manipulated by the algorithm. The countermeasure modifies the input m fed into the RSA algorithm thanks to a uniformly drawn random value r. Instead of cumputing a ciphertext c defined as $c = m^d \bmod N$, the algorithm now computes a ciphertext c' defined as $c' = (mr^e)^d \bmod N$. Then, the ciphertext c is retrieved by computing $c = (c'r^{-1}) \bmod N$. Due to the multiplication by r^e, the intermediate values manipulated during the computation of c' are not trivially deduced from m^d. The information that can be retrieved by the attacker on the private exponent is hence smaller.

This approach was generalized and applied to symmetric algorithms independently by Goubin and Patarin [GP99] and Chari *et al.* [CJRR99]. Usually, for historical reasons the countermeasure applied to asymmetric algorithms is called *blinding*, while it is called *masking* when applied to symmetrical algorithms. Nonetheless, the *modus operandi* of the countermeasures stays the same. This family of countermeasures affects the distinguishing phase of the attack, by shrinking the statistical dependencies between the manipulated variables and the secret data. The impact of masking is nowadays well understood thanks to the works of modeling described in 2.4.

Formally, each sensitive variable x is divided into $d+1$ variables $\{x_i\}_{0\leq i\leq d}$ referred to as *shares*. Among those values, d are generated uniformly at random and the last one is computed such that the combination of the $d+1$ shares, according to a chosen composition law $\star$, is equal to the initial variable x, *ie.*, $x = x_0 \star \cdots \star x_d$. One of the shares say x_0 is sometimes referred to as *masked variable* and the other shares $x_1, \cdots x_d$ as the *masks*. For the masking approach to be sound, it is required that the masks are uniformly and independently generated. In that case, the $(d+1)$-tuple of shares can be modeled as a random vector $(X_0, \cdots, X_d)$ where for $i \geq 1$, the X_i are mutually independent random variables with uniform distribution on $\mathcal{X}$, and X_0 is computed such that it holds:

$$x = \bigstar_{i=0}^{d} X_i. \tag{2.14}$$

The choice of the law $\star$ depends on the algorithm and its implementation. The most trivial instantiations are with the bitwise addition (aka. XOR or addition in the field of two elements) law $\oplus$ or with the field product law $\times$, respectively used by the *Boolean sharing* [GP99; CJRR99] and *multiplicative sharing* [AG01]. More complex instantiations can be deduced from descriptions of other secret sharing schemes, such as *Shamir's secret sharing scheme* [Sha79] or *inner masking product* [BFGV12; BFG15].

Remark 2.5.1. For security reasons, $\star$ should form a group law on the definition set of the sensitive variables x. This property has been sometimes overlooked and provoked flaws in some schemes. For example, the multiplicative masking proposed in [AG01] could not mask the value 0, which however was in the definition set of x. This allows a statistical attack breaking the scheme's security [GT03]. Multiplicative masking is however a possible construction when the definition set does not include the value 0, as illustrated in [GPQ10]. Its use in practical settings is nonetheless quite costly due to the workarounds required to solve the 0 problem.

Remark 2.5.2. In an implementation protected by making, the sensitive variable x is not manipulated anymore, but all shares are manipulated instead. Often, these manipulations are sequential. An immediate consequence is that the attacker cannot obtain a leakage depending on x anymore, but he can instead obtain a leakage tuple $\boldsymbol{L}$ corresponding to the different leakages of each share:

$$\boldsymbol{L} = (L_0, \cdots, L_d), \tag{2.15}$$

wherer L_i denotes the leakage associated to the random value X_i, for any $i < d+1$. The obtention of such a tuple can lead to so-called *high order side-channel attacks*, which aim at recovering the sensitive value x by studying the distribution of $\boldsymbol{L}$. We will detail and study the effectiveness of such attacks in Chapter 3.

The inverse operation of masking, constisting in the retrieval of x from the shares $\{x_i\}_{0\leq i\leq d}$ is referred as a *reconstruction*, and can be performed thanks to Equation 2.14.

Remark 2.5.3. The so-called *threshold implementations* [BGG+14] are a family of countermeasures, extending the idea of masking, in a wider model. In this model, additional physical leaks can occur, such as leakages due to *glitches*, where more than one variable is involved in each observation. Glitches are unwanted transitions at the output of gates resulting from differences in signals arrival time. The number of such transitions can be detected by side-channel analysis and reveal information about the sensitive value. Threshold implementations hence increase the number of shares used in the classical masking countermeasures, to ensure that these leaks do not induce a security flaw (see for example [RBN+15]).

Masking as linear codes

All these countermeasures except multiplicative masking can be expressed in the domain of linear error correcting codes theory. This link was first made by Massey in [Mas93], and allows for a better understanding of the mathematical properties of these constructions. The author proved an equivalence between linear coding theory and linear secret-sharing techniques. The sharing can hence be reformulated in those terms. To proceed to the sharing of a sensitive value x into $d+1$ shares $x_0, \cdots, x_d$, a linear code $\mathcal{C}$ of length $d+2$, dimension n and minimal distance δ is chosen. Denoting the minimal distance of its dual code $\mathcal{C}^\perp$ by $\delta^\perp$, the results of [Mas93] imply that, as long as $\delta, \delta^\perp \geq 2$, one can build a sharing scheme from $\mathcal{C}$. Then, $n-1$ random variables $r_1, \cdots, r_{n-1}$ are uniformly drawn, and the vector $(x, r_1, \cdots, r_{n-1})$ is encoded by $\mathcal{C}$ hence obtaining the codeword $(x, x_0, x_1, \cdots, x_d)$. Denoting by G the generator matrix of $\mathcal{C}$, we have:

$$(x, r_1, \cdots, r_{n-1}) \cdot G = (x, x_0, x_1, \cdots, x_d). \tag{2.16}$$

The reconstruction of the shared variable x is hence seen as a *decoding* problem: several shares $(x_i)_i$ are known, and the reconstruction of x amounts to the decoding of the codeword $(x, x_0, x_1, \cdots, x_d)$. Note that this procedure must also be performed securely. Formally, a vector $\boldsymbol{\lambda} = (\lambda_i)_i$ exists, such that $x = \sum_i \lambda_i x_i$. In secret-sharing theory, a set of indices $\mathcal{I}$ such that the shares $\{x_i | i \in \mathcal{I}\}$ allow for the reconstruction of x is called a *qualified set*. The set of qualified sets is further called an *access structure*. It is argued in [Mas93] that the access structure of a secret-sharing scheme is entirely defined by the dual code $\mathcal{C}^\perp$.

In [CCG+07], Chen *et al.* furthermore studied the security properties of such a construction. Considering x as the realization of a random variable X and each x_i as the realization of a random variable X_i, they show that, for any set of indices $\mathcal{I}$ of cardinality less than $\delta^\perp - 2$, the mutual information between X and $(X_i)_{i\in\mathcal{I}}$ is null. Moreover, the authors prove that there exists a set $\mathcal{I}$ of cardinality $d - \delta + 3$ such that x can be reconstructed from the shares $(x_i)_{i\in\mathcal{I}}$.

These results are of particular interest in the case of *maximum distance separable* (MDS) codes, *ie.* codes whose parameters achieve the so-called Singleton bound, *ie.* verify the equation $\delta = d+2-n+1 = d-n+3$. Indeed, if $\mathcal{C}$ is MDS, then, $\mathcal{C}^\perp$ is also MDS. Therefore we have that $\delta^\perp = d+2-(d+2-n)+1 = n+1$. This results applied to the bounds of [CCG+07] show that no information can be retrieved on the shared variable when the number of known shares is strictly lower than n, but that n shares suffice to reconstruct x. Such a scheme is called a *threshold* secret-sharing scheme.

Algorithm 1 Transcoding algorithm

```
Require: linear codes $\mathcal{C}, \mathcal{C}'$ of respective length $\ell+2, \ell'+2$, dimension $k, k'$, minimal distance $\delta, \delta' \geq 2$ and minimal distance of their duals $\delta, \delta'^{\perp} \geq 2$, shares $(x'_0, \cdots, x'_{\ell'})$ such that $(x, x'_0, \cdots, x'_{\ell'}) \in \mathcal{C}'$, a reconstruction vector $\boldsymbol{\lambda}'$ of $\mathcal{C}'$
Ensure: shares $(x_0, \cdots, x_\ell)$ such that $(x, x_0, \cdots, x_\ell) \in \mathcal{C}$
  for $i = 0$ to $\ell'$ do
      $(z_{0,i}, \cdots, z_{\ell,i}) \leftarrow$ encoding of $\lambda'_i x'_i$ in $\mathcal{C}$ (cf Eq. 2.16)
  for $j = 0$ to $\ell$ do
      $(x_0, \cdots, x_\ell) = (\sum_{i=0}^{\ell'} z_{0,i}, \cdots, \sum_{i=0}^{\ell'} z_{\ell,i})$
```

Computing on masked data

Obviously, being able to share and reconstruct a sensitive variable is not sufficient to ensure the full security of an implementation. Indeed, operations must be performed on these variables, and consequently the masking needs to be resilient to these operations. Namely, if a sensitive variable x is shared, and an operation f is to be performed, it is required that the execution of f does not leak information on x, and that the obtained value is still a sharing of $f(x)$. This problem is a major issue in the field of countermeasures against side-channel attacks, and also appears in another form in the multi-party computation domain.

We first tackle the issue of *stable transformations* (*ie.* any linear operation such as the addition with another codeword or a scalar multiplication). In those cases, the security is trivially maintained thanks to the linear coding theory. To see it, let $(\beta, \beta_0, \beta_1, \cdots, \beta_d)$ denote a codeword of $\mathcal{C}$, and let α denote a scalar, then the shares of the value $f(x) = \alpha x + \beta$ are $f(x_0), \cdots, f(x_d)$. Indeed, the vector $(\alpha x + \beta, \alpha x_0 + \beta_0, \alpha x_1 + \beta_1, \cdots, \alpha x_d + \beta_d)$ is still a codeword of $\mathcal{C}$.

The problem of non-linear operations is way more complex. Indeed, if f is non-linear, then the corresponding word $(f(x), f(x_0), \cdots, f(x_d))$ is not ensured to be in $\mathcal{C}$. The issue of the secure evaluation of f is hence tackled in two parts: first, f is written as a composition of linear operations and multiplications (note that this is always possible, since every function f defined over a finite field can be written as a polynomial over this field), then, each of this operation is secured independently, and composed in such a way to maintain security. The multiplication of two shared variables is still a non-linear operation and requires some tricks to be performed. This construction was described by [BOGW88] and [CCD88] for a particular secret-sharing scheme before being generalized in terms of coding theory in [CC06].

Let us consider two codewords $c_x = (x, x_0, \cdots, x_d)$ and $c_y = (y, y_0, \cdots, y_d)$ in $\mathcal{C}$, corresponding respectively to the sharings of two variables x and y. The goal of the multiplication step is to retrieve a sharing of the product xy in the code $\mathcal{C}$. The multiplication of all coordinates of c_x with those of c_y lays the word $c_{xy} = (xy, x_0y_0, \cdots, x_dy_d)$. This word does not belong to the code $\mathcal{C}$. Nonetheless, it belongs to another linear code $\mathcal{C}'$, defined by the span of vectors in $\{c \cdot c' | c, c' \in \mathcal{C}\}$. When $\mathcal{C}'$ and its dual code both have minimal distance higher than 2, we know from Section 2.5.1 that they can define a secret-sharing scheme. Consequently, under those conditions on the minimal distances of $\mathcal{C}'$ and its dual, there exists a reconstruction vector λ such that $xy = \sum_i \lambda x_i y_i$. This observation allows for a *transcoding* operation, that is, an algorithm allowing to map a sharing from a linear code $\mathcal{C}'$ towards a corresponding sharing from another linear code $\mathcal{C}$.

To complete the multiplication, it hence suffices to perform Algorithm 1 to transcode the shares $(x_0y_0, \cdots, x_dy_d)$ from $\mathcal{C}'$ to the corresponding shares in $\mathcal{C}$.

Remark 2.5.4. The protocol described in Algorithm 1 is furthermore a *secure transcoding* operation, in the sense that, as long as the respective minimal distances $\delta, \delta', \delta^{\perp}, \delta'^{\perp}$, of C, C' and their duals are greater than 2, any set of at most $\max(d'^{\perp}, d) - 2$ intermediate values is independent from the shared variable.

Being the cornerstone of the evaluation of the multiplication, the transcoding operation has been the subject of several works, proposing new algorithms to achieve more efficient or more secure results. Some other constructions can be found for example in [GM11; CPR12].

Sometimes, the code $\mathcal{C}'$ resulting from the coordinate-wise product of the codewords of $\mathcal{C}$ does not verify the requirements on the minimal distance. For example, for the code associated to the *Boolean sharing*, which we describe lower, the code $\mathcal{C}'$ is exactly the whole space on which $\mathcal{C}$ is defined. Consequently, the corresponding minimal distance is 1. In this case, the described trick can be adapted, by replacing the code $\mathcal{C}'$ by the code $\mathcal{C}^{\star}$, where the coordinate-wise product is replaced by the outer product, viewed as a vector. The transcoding operation described in Algorithm 1 can then be used. We study particularly this setting in Chapter 4 of this thesis, and we propose new efficient solutions, that can be written as new transcoding procedures.

Even with clever tricks, dealing with linear operations is still cheaper than dealing with non-linear ones. Consequently, to improve performance, the decomposition of the function f must minimize the number of non-linear operations. These last years, the protection of the symmetrical algorithms, especially AES, has been studied. In AES, the only non linear part is the substitution layer, itself composed of small non-linear functions (called Sboxes). The decomposition of the AES'-Sbox function has consequently been the subject of several studies. The element manipulated by these functions are in a field of characteristic 2, hence the squares operations are linear. Rivain and Prouff showed in [RP10] that this function can be evaluated using as few as 4 multiplications, and proposed a secure scheme for it, by extending the seminal construction of [ISW03] from $\mathbb{F}_2$ to $\mathbb{F}_{2^8}$. The problem of the minimization of non-linear operations in the expression of Sboxes in fields of characteristic 2 has recently been tackled by the works of [CGP+12; RV13; CRV14; GPS14; CPRR15]. These papers allow to prove lower bounds on the number of such operations that one can obtain by decomposing any polynomial function on a field of characteristic 2, and they provide several constructions to efficiently find good decompositions.

In [GHS12], Gentry *et al.* proposed a new decomposition of the AES Sbox where 2 out of 4 multiplications are computed in parallel. Moreover, it can be observed that these two multiplications have a common input. Recently, Coron *et al.* [CGPZ16] took advantage of this parallelization to further reduce the complexity of the evaluation. Though not really reducing the number of multiplications in the expression of the Sbox, the authors design a trick to artifically reduce the computation complexity, by introducing the notion of *common shares*. Denoting by $c \cdot a$ and $c \cdot b$ the two parallel multiplications occuring in the construction of [GHS12], they set the first half of the sharing of b as equal to the first half of the sharing of a. Consequently, part of the operations performed to compute $c \cdot a$ and $c \cdot b$ are similar, and can hence be computed only once. This trick allows to reduce the number of multiplications in the AES Sbox evaluation to the equivalent of 2.8 instead of 4.

The composition of several secure constructions, or *gadgets*, is a prime issue in the design of secure evaluations. Indeed, the succession of two secure gadgets is not necessarly secure: one needs to prove that combining observations between both gadgets does not give information about the manipulated sensitive values. To allow proofs of compositions, the notion of *strong non-interference* (SNI) was introduced by Barthe *et al.* in [BBD+15a], extending the previous notion of *non-interference* (NI) defined in [BBD+15b]. Essentially, a gadget is said t-SNI if it can be simulated using a number of each of its input shares that is only bounded by the number of observations made by the adversary on inner variables manipulation, and is independent from the number of observations made on output values, as long as the total number of observations does not exceed t. This definition directly implies that a t-SNI algorithm is t-secure. Indeed, for any t variables observed by an attacker, it is possible to perfectly simulate the distribution of these variables from some random bits and a set of less than t inputs. The composition of two t-SNI algorithms is itself t-SNI. The composition of t-NI and t-SNI gadgets is not necessarly t-SNI, and such constructions must hence be proven otherwise. We extensively use these notions in Chapter 4 to prove the security of our constructions.

Examples of secret sharing schemes

For illustration purposes, we describe hereafter the matrices G_{bool}, G_{IP} and G_{SSS}, respectively corresponding to the Boolean sharing, inner product masking, and Shamir's scheme, and the generator matrices H_{bool}, H_{IP} and H_{SSS} of their dual codes. For any integer n, we denote by I_n the n-dimensional identity matrix.

In the Boolean sharing, the sensitive value x is divided in $d+1$ shares through the law $\oplus$. Hence, G_{bool} is simply the *parity* matrix:

$$G_{bool} = \left(\begin{array}{c|c} & 1 \\ I_{d+1} & \vdots \\ & 1 \end{array} \right).$$

The generator matrix H_{bool} of the dual code is therefore:

$$H_{bool} = (-1, \cdots, -1, 1).$$

The masking of a sensitive value through inner product sharing is described in [BFG15]. The sharing is in fact very similar to the Boolean sharing's one. In fact, the only difference is that the last share is not computed using a XOR law, but with an inner product with a publicly known vector $(1, l_1, l_2, \cdots, l_d)$. This allow us for the description of the generator matrix G_{IP}:

$$G_{IP} = \left(\begin{array}{c|c} & 1 \\ & l_1 \\ I_{d+1} & \\ & \vdots \\ & l_d \end{array} \right).$$

The generator matrix H_{IP} of the dual code is therefore:

$$H_{IP} = (-1, -l_2, \cdots, -l_d, 1).$$

In Shamir's secret sharing, the shares correspond to evaluations of a certain polynomial in $d+1$ different public points α_i, for $i \in [0, d]$. This is equivalent to a Reed-Solomon code, where the coefficients of the polynomial correspond to the input vector. In particular, the sensitive value is the constant term of this polynomial. Consequently, G_{SSS} is the Vandermonde matrix:

$$G_{SSS} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & \alpha_0 & \cdots & \alpha_d \\ 0 & \alpha_0^2 & \cdots & \alpha_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \alpha_0^{n-1} & \cdots & \alpha_d^{n-1} \end{pmatrix}.$$

The dual code is therefore also a Reed-Solomon code, of generator matrix H_{SSS}:

$$H_{SSS} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & \alpha_0 & \cdots & \alpha_d \\ 0 & \alpha_0^2 & \cdots & \alpha_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \alpha_0^{d-n-1} & \cdots & \alpha_d^{d-n-1} \end{pmatrix}.$$

Remark 2.5.5. All these codes are MDS. Note that, contrary to the Boolean sharing or the inner product masking, where the dimension of the code has to be $d + 1$, Shamir's secret sharing scheme allows for arbitrary dimension n. The corresponding dual codes for Boolean sharing and inner product masking are therefore of dimension 1, and consequently, reconstructions with these schemes require that *all* shares are known. Shamir's secret sharing allow for more a complex access structure, *ie.* every set of shares of cardinality higher than a chosen dimension n.

2.5.2 Hiding

Another approach is taken by so-called *hiding* countermeasures, introduced in [KJJ99]. The goal of such countermeasures is to hide the manipulation of data among a lot of noise. Using this approach, the sensitive variable is indeed manipulated by the device, but the exploitation of the leakage obtained by the attacker is very difficult. This family of countermeasures target the leakage collection phase of the attack, by increasing the difficulty of the signal resynchronisation and the identification of the correct instant when the sensitive value is manipulated.

The first approach designed to achieve such a feit is the so-called *balancing* method. This countermeasure aims at the insurance of a balance of the power consumed by an electronic device in such a way that, at each time, the consumption is as constant as possible. The obtained signal is consequently unrelated to the data being processed. Several means have been considered to reach this goal: balanced gate constructions [KJJ99], dual-rail precharge logic [PM05], modification of current mode [MSH+04] and asynchronous logic [MMC+02]. Through theoretically efficient, these countermeasures are very hard to implement in practice. Indeed, the power consumption of electronic devices depends on a large number of parameters on which the modern designer has little to no control, such as wire routing, capacitive effects or even cross-talk between wires. These countermeasures are therefore seldom used in the real world.

The second approach is built on the idea of artificially inducing noise in the measurement themselves. We can further separate this approach in two sub-families, the first one hiding the relevant signal with *amplitude* noise, and the second one hiding it with *temporal* noise. With the *amplitude* approach, the device will perform unrelated operations at the exact same time as the relevant computation is being performed. The amplitude of the resulting signal obtained by an attacker will hence be polluted by the unrelated operations. Nonetheless, this countermeasure is also difficult to efficiently implement. Indeed, while a power measurement of the whole device would be affected, a very local measurement of the electromagnetic emanations could be unaffected by operations occuring in another area. The *temporal* approach uses random interruptions, random accesses, and temporal duplications of algorithms to hide the relevant operation among *dummy* ones. The relevant signal is hence temporaly lost in the middle of a lot of noise. An usual circumvention to this countermeasure applies integrating techniques on the signal in order to fall back into an *amplitude* noise problem [CCD00].

2.6 Randomness complexity

Most countermeasures described in Section 2.5 make use of randomness. Hiding countermeasures might require randomness to shuffle executions, and insert unpredictable delays. Masking countermeasures require randomness to share variables such that the manipulations are uncorrelated to the secret. Moreover, operations manipulating masked values induce an overhead in term of needed randoms during the transcoding part. Quite surprisingly, the problem of the amount of randomness required by those countermeasures is often overlooked.

The production of random bits is however a complex issue in the design of secure devices. Indeed, in practice, randomness is often generated using the succession of sophisticated components: a *true random number generator* (TRNG) exploits an unpredictable process, such as analog phenomena, to accumulate entropy and output random bits, which are then retreated by a *deterministic random number generator* (DRBG), such as a cryptographic algorithm. The unpredictability of the TRNG is ensured physically, and that of the DRBG is ensured computationally and/or statistically. In addition of being a theoreitcal *chicken-and-egg* problem, this whole process often suffers from a low throughput and a long execution time, and countermeasures usually require a lot of randomness to efficiently secure algorithms. Indeed, in general, for a DRBG based on a 128-bit block cipher, one call to this block cipher enables to generate 128 pseudorandom bits (see [BK12]). However, one invocation of the standard AES-128 block cipher with the ISW multiplication requires as much as 30,720 random bits (6 random bytes per multiplication, 4 multiplications per S-box [RP10]) to protect the multiplications when masked at the low order $d = 3$, which corresponds to 240 preliminary calls to the DRBG.

We hence argue that the amount of randomness required by countermeasures against side-channel attacks should be estimated along the classical evaluations of time, area and number of operations. To the best of our knowledge however, this problem was only tackled in the seminal paper of [ISW03]. In Chapters 4 and 5, we put a primary focus on the notion of *randomness complexity*, that is, the amount of randomness needed to achieve secure constructions. Respectively, we evaluate this complexity in several new proposed computations of non-linear operations on masked variables, and on new protocols allowing to securely compute a veto.

2.7 Effectiveness of side-channel attacks

The wide variety of side-channel attacks available in the state of the art calls for the need to compare their respective effectiveness. Indeed, in the constraints of limited time and resources, designers, evaluators and attackers are not able to perform all attacks, and a choice must hence be made. This choice is mostly driven by the chance of success of an analysis, that is, the probability for an attacker to retrieve the secret variable manipulated by the device in the least possible number of observations.

This need for a classification of attacks based on their chance of success led the community to apply the notion of *success rate* to side-channel analysis. The success rate measures the probability of success of an attack.

In [WO11b], Whitnall and Oswald proposed a framework to further compare side-channel attacks and their distinguishers, using several criteria:

- *Correct key ranking*: the position of the correct key when ranked by score value.

- *Relative distinguishing margin*: the difference between the scores of the correct key and the best alternative, normalized by the variance of scores.

- *Absolute distinguishing margin*: the difference between the scores of the correct key and the best alternative, divided by this expected difference in a noiseless setting.
- *Standard score*: the number of standard deviations above or below the mean for the correct key score.

The authors also added two other criteria regarding the cardinality of the set of plaintexts required for an attack to be successful: the *average critical support* and the *critical support for p% success rate*, respectively corresponding to the average required cardinality of the set for the attack to be successful, and to the cardinality for which the rate of success is at least $p\%$.

All these criteria are of interest to compare different side-channel attacks. However, the main issue for a practical attacker is that they all rely on *a posteriori* results, that is, they measure the effectiveness of an attack *after* it has been performed. Oftentimes, for example, the success rate is estimated empirically: the attack is repeated a large number of times, and the number of successes is counted, allowing for the computation of the empirical success rate SR_{emp}:

$$SR_{emp} = \frac{\text{number of successes}}{\text{number of attacks}}. \tag{2.17}$$

Such an estimation is hence very dependent of the number of attacks, and consequently of the time and resources deployed by the attacker. It is hence oftentimes impractical to compute. Moreover, the clever choice of an attack should rely on an *a priori* approach, allowing the estimation of effectiveness *before* it is even performed.

We tackle this problem in Chapter 3, by reviewing state of the art approaches to theoretically estimate the success rate, and we extend those works to a whole family of side-channel attacks, based on so-called *additive distinguishers*. Moreover, we give rationales for the practical interest of this approach in the choice of an attack, by illustrating how the characterization of a device and measurement setup is much cheaper to perform than the attack itself.

Chapter 3

Estimating the Success Rate of Side-Channel Attacks

My success rate is 100%. Do the math.

Charlie Sheen

In this chapter, we propose to tackle out the problematic of success rate's evaluation. Part of this work has been the subject of two papers published in the workshop on *Cryptographic Hardware and Embedded Systems* in 2013 [TPR13] and 2014 [LPR+14].

3.1 Introduction

Side-channel attacks against block cipher implementations usually consider the secret as a tuple of so-called *subkeys* and apply a *divide-and-conquer* strategy to recover them separately. During the conquering phase, a *partial attack*, limited in time and space, is run against each subkey. Heuristics are then applied to decide on the success or unsuccess of each of these attacks. Subkeys corresponding to attack failures are deduced by exhaustive search. In practice, this last step is often executed either for efficiency reasons or because it is assumed that there is no chance to get the missing subkeys directly by side channel analysis. This description makes apparent that the attack effectiveness greatly depends on the heuristic applied by the adversary. Indeed, incorrect heuristics leave the subsequent exhaustive search little chance to succeed.

As described in Section 2.2, a partial attack is performed on a finite set of measurements and aims at the recovery of a correct subkey k^* among a small set $\mathcal{K}$ of hypotheses (usually, $|\mathcal{K}| = 2^8$ or 2^{16}). For such a purpose, a *score* is computed for every subkey hypothesis $k \in \mathcal{K}$, leading to an ordered *scores vector*. The position r_k of an hypothesis k in this vector is called its *rank*. Note that the notion of rank of an hypothesis trivially extends the correct key ranking criteria described in Section 2.7. The attack is said to be *successful* if r_{k^*} equals 1. Extending this notion, an attack is said *o-th order successful* if r_{k^*} is lower than or equal to o.

Depending on the context, the secret k^* can either be known or unknown. Before putting a product in the field, resilience of its cryptographic implementations against side-channel attacks is often evaluated both by their designers and by independent evaluation laboratories. In this context, k^* is often known and/or chosen, allowing for a very accurate assessment of the implementation's security. Once in the field, the profile of attacker changes for someone more malicious. The approach is now totally black-box, and the key is unknown. In this chapter, we study how accurately security can be assessed in both contexts.

Under the assumption that the secret k^* is known, the success of a partial attack can be unambiguously stated. This even allows for the estimation of its *success rate*, by simply dividing the number of attack successes (for which $r_{k^*} \leq o$) by the total number of attacks (see Equation 2.17). Estimating the success rate of a side-channel attack –that uses a given number of leakage observations– is a central issue regarding the physical security evaluation of a cryptographic implementation. The empirical way is to perform the attack a certain number of times and to record the average number of successes. However, this approach is prohibitive against implementations protected by effective countermeasures since the attacks may become too costly to be performed several times (or even once). This does not mean that the implementation is *secure* though; this only means that the implementation is secure beyond the means of the evaluator (which may not compete with the means of a motivated attacker). This situation is not satisfactory in practice where one desires that the computational cost of performing a security evaluation be fairly low and uncorrelated to the actual security of the target implementation.

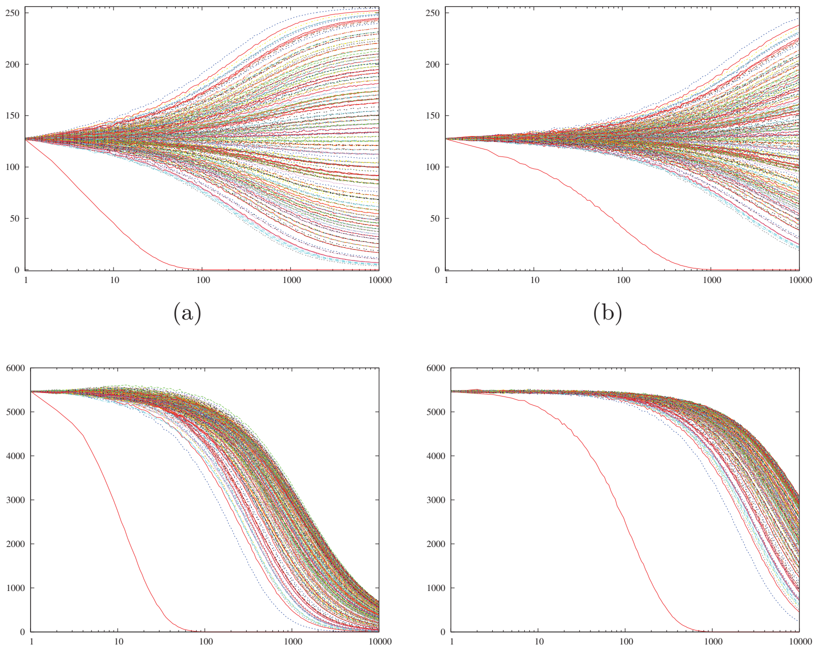
The problem of evaluating the success of an attack has already been tackled in several papers [FLD12; Man04; SPRQ06; Riv09]. In [Man04] and [SPRQ06], the CPA success rate is evaluated by using Fisher's transformation (see for instance [Fis22]): simple formulae are exhibited to estimate the success rate in terms of both the noise standard deviation and the correlation corresponding to the correct key. These works were a first important step towards answering our problem. However, they were conducted under the assumption that wrong hypotheses are uncorrelated to the leakage, which turned out to be incorrect.

To illustrate this, we study the sampling distribution of the rank, by running an experiment where several CPA targeting the output of the AES sbox are performed, assuming a Hamming weight leakage model with a Gaussian noise of standard deviation 3. A random subkey k^* is drawn, and q leakage observations are generated. Then, the rank $r_{k,q}$ of each hypothesis k is computed with respect to the CPA scores. This experiment is repeated several times with new leakage observations, and the mean and variance of the associated random variables $R_{k,q}$ are computed. We then perform the same experiment on a leakage of standard deviation 10.

The results, plotted in Figure 3.1 show that this assumption, sometimes called *wrong key randomization hypothesis* [Har96], does not fit with the reality: each hypothesis score indeed actually depends on the bit-wise difference between the hypothesis and the correct key. The error induced by the assumption is not damaging when one only needs to have an idea about the general attack trends. It is however not acceptable when the purpose is to have a precise understanding of the attack success behavior and of the effect of the sbox properties on it. This observation has been the starting point of the analyses conducted in [FLD12] and [Riv09], where the wrong key randomization hypothesis is relaxed. In Rivain's paper, a new and more accurate success rate evaluation formula is proposed for the CPA. In [FLD12], Fei *et al.* introduce the notion of *confusion coefficient*, and use it to precisely express the success rate of the monobit DPA. This work can be viewed as a specification of Rivain's, as monobit DPA is a particular case of a CPA [DPRS11].

When the secret k^* is unknown, the adversary chooses a candidate which is the most likely according to some *selection rules*. In this case, the success can only be decided *a posteriori* and a *confidence level* must hence be associated *a priori* to the choice before the decision is made. Clearly the soundness of the latter process depends on both the selection and the confidence, which must hence be carefully defined. In particular, to be effective in a practical setting, the confidence associated to a decision must be accurately evaluated even for a small number of observations.

This need is illustrated in Figure 3.2. A usual selection rule is to simply choose the best ranked key. Using 280 observations, this rule would lead to the choice of the right subkey, whereas a wrong



(a)

(b)

(c)

(d)

Figure 3.1: Results of CPA experiments on the AES sbox. The averages of the ranks are plotted, in function of the number of measurements used for each attack (logscaled), in (a) and (b) for Gaussian noises of standard deviation respectively equal to 3 and 10. Their respective variances are plotted in (c) and (d).

subkey would have been chosen using 420 observations. An optimal heuristic would then deem the first attack a success, and the second one a failure.

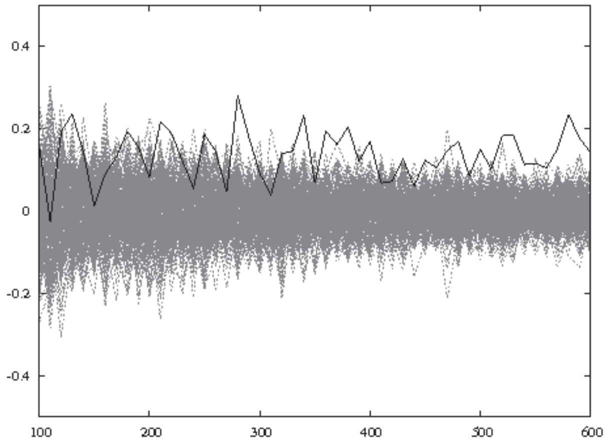


Figure 3.2: Correlation coefficients obtained from a CPA on AES. The correct hypothesis is plotted in black.

Interestingly, the invalidation of the wrong key randomization hypothesis, as illustrated by Figure 3.1, implies that the evolution of the sampling distribution of every R_k is eventually related to the value of the correct key and hence brings information about it. In other terms, the full vector of ranks gives information on the correct key (and not only the hypothesis ranked first). Based on this observation, it seems natural to use this information to increase the attack efficiency and/or the confidence in the attack results. To be able to precisely assess both kinds of increase, the distributions of all the variables R_k therefore need to be understood. Bearing this in mind, we now formalize some information that an adversary can obtain while performing a side-channel attack on a set of q independent observations. Scores are computed using a *progressive approach*, *i.e.* taking an increasing number of traces into account. Namely, the scores are computed after $q_1 < q$ observations, then again after $q_2 > q_1$ observations, and so on until the q observations have been considered. This approach enables the computation of the matrix[1]:

$$\mathcal{M}_s = \begin{pmatrix} s(1,q_1) & s(1,q_2) & \cdots & s(1,q) \\ \vdots & \vdots & \ddots & \vdots \\ s(|\mathcal{K}|,q_1) & s(|\mathcal{K}|,q_2) & \cdots & s(|\mathcal{K}|,q) \end{pmatrix},$$

where $s(k, q_i)$ denotes the score of the hypothesis k computed using q_i observations.

According to the Neyman-Pearson lemma [NP33], an optimal selection rule would then require the knowledge of the statistical distribution of this matrix when the correct subkey is known. In a real attack setup however, the latter subkey is unknown and one then has to proceed with a likelihood-ratio approach in order to retrieve it. Even optimal from an effectiveness point of view, this approach is not realistic as it reposes on two major issues: the knowledge of the distribution of the matrix (which requires a theoretical study over highly dimensional data) and the computation and storage of every score (which may require a lot of time and memory). Moreover, one could wonder if all the information contained in the matrix is relevant, or if there is some redundancy. On the opposite side, the actual attacks only use small parts of the available information. For example, the classical selection of the best ranked key simply amounts to choose the maximum of the last column of scores in $\mathcal{M}_s$. Between those two extrem approaches, one could wonder if other tractable parts of the matrix can be used to give better selection rules or better confidence estimators.

Several criteria indicating the effectiveness of side-channels have also been studied to compare side-channel attacks (e.g. [WO11a]). Among those, the particular behavior of the right subkey ranking have been exploited in [NSGD11] to propose an improvement of the attack efficiency when the correct key is unknown. This approach illustrates the importance of such criteria in practical attacks, but it is purely empirical. To optimize the exhaustive search phase, the authors of [VGRS11] propose an algorithm taking as inputs the confidence of the results of each partial attack, thus illustrating the need for a precise study of this problem.

In this chapter, we propose a methodology to estimate the success rate of side-channel attacks targeting implementations protected by masking. Our methodology is based on the approach proposed by Rivain in [Riv09] in the context of first-order side-channel attacks. It trivially extends to first order side-channel attacks against unprotected implementations. The principle of this approach is to study the multivariate distribution of the score vector resulting from an attack. Specifically, Rivain suggests to approximate this distribution by a multivariate Gaussian distribution, which is sound in the context of *additive distinguishers* such as the correlation and the likelihood. We generalize this approach to high order side-channel analysis (HO-SCA) and we show how to derive the distribution parameters with respect to the leakage parameters. We show that using this methodology makes it possible to accurately estimate the success rate of a high order side-channel attack based on a simple profiling of the leakage parameters. Moreover, we demonstrate the soundness of our methodology by comparing its results to various attack experiments against masked

[1] Note that this matrix does not capture *all* the information available to an attacker. Indeed, for example all sets of size q_1 could be used to gain information, not only the first set of measurements.

AES implementations running on two different microcontrollers. We also introduce the notion of *confidence level*, aiming at evaluating the probability of correctness of a side-channel attack result.

3.2 Preliminaries

The expectation and the variance of a random variable X are respectively denoted by $\mathrm{E}[X]$ and $\mathrm{Var}[X]$. The covariance between two random variables X and Y is denoted by $\mathrm{Cov}[X, Y]$.

The Gaussian distribution of dimension T with expectation vector $\boldsymbol{m}$ in $\mathbb{R}^T$ and $T \times T$ covariance matrix $\boldsymbol{\Sigma}$ is denoted by $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$, and the corresponding probability density function (pdf) is denoted by $\phi_{\boldsymbol{m},\boldsymbol{\Sigma}}$. We recall that this pdf is defined for every $\boldsymbol{x} \in \mathbb{R}^T$ as

$$\phi_{\boldsymbol{m},\boldsymbol{\Sigma}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^T |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^\intercal \cdot \boldsymbol{\Sigma}^{-1} \cdot (\boldsymbol{x}-\boldsymbol{m})\right) , \tag{3.1}$$

where $(\boldsymbol{x}-\boldsymbol{m})^\intercal$ denotes the transpose of the vector $(\boldsymbol{x}-\boldsymbol{m})$ and $|\boldsymbol{\Sigma}|$ denotes the determinant of the matrix $\boldsymbol{\Sigma}$. The corresponding cumulative distribution function (cdf) is denoted $\Phi_{\boldsymbol{m},\boldsymbol{\Sigma}}$ and is defined for a pair of vectors $\boldsymbol{a} = (a_1, a_2, \ldots, a_T)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_T)$ over $(\mathbb{R} \cup \{-\infty, +\infty\})^T$ by

$$\Phi_{\boldsymbol{m},\boldsymbol{\Sigma}}(\boldsymbol{a}, \boldsymbol{b}) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_T}^{b_T} \phi_{\boldsymbol{m},\boldsymbol{\Sigma}}(\boldsymbol{x}) \, d\boldsymbol{x} . \tag{3.2}$$

If the dimension T equals 1, then the Gaussian distribution is said to be *univariate* and its covariance matrix is reduced to the variance of the single coordinate denoted σ^2. If T is greater than 1, the Gaussian distribution is said to be *multivariate*.

3.3 Side-Channel Model

We consider a cryptographic algorithm protected by *masking* and running on a leaking device. A (high order) side-channel attack exploits the leakage resulting from intermediate computations in order to recover (part of) the secret involved in the cryptographic algorithm. Let s denote such an intermediate variable satisfying:

$$s = \varphi(x, k^*) , \tag{3.3}$$

where x is (part of) the public input of the algorithm, k^* is (part of) the secret input of the algorithm, and φ is some function from $\mathcal{X} \times \mathcal{K}$ to $\mathcal{S}$.

As described in Section 2.5.1, for an implementation protected with masking, such a variable s is never stored nor handled in clear but in the form of several, say $d + 1$, *shares* s_0, s_1, …, s_d satisfying the relation

$$s_0 \star s_1 \star \cdots \star s_d = s \tag{3.4}$$

for some operation $\star$. In the common case of Boolean masking this operation is the bitwise addition modulo 2 (or XOR, denoted $\oplus$), but it might be some other group addition law. In that case, it can be noted that the $(d+1)$-tuple of shares can be modeled as a random vector $(S_0, S_1, \ldots, S_d)$ where $S_0 = s \oplus \bigoplus_{j=1}^{d} S_j$ and, for $j \geqslant 1$, the S_j are mutually independent random variables with uniform distribution over $\mathcal{S}$.

3.3.1 Leakage Model

As stated in Remark 2.5.2, during the execution of the algorithm, the processing of each share S_j produces some leakage L_j revealing some information about the share value, hence forming the leakage tuple

$$\boldsymbol{L} = (L_0, L_1, \ldots, L_d) \ . \tag{3.5}$$

We shall sometimes use the alternative notation $\boldsymbol{L}_s$ or $\boldsymbol{L}_{x,k^*}$ to indicate that the leakage arises for the shared value $s = \varphi(x, k^*)$.

In this chapter, we use the leakage model described in Section 2.4, in particular the additive Gaussian noisy leakage. Therefore, Equation 2.12 implies that, for every $j \leq d+1$, we have: shares, the leakage has a Gaussian distribution. This assumption is referred

$$L_j = f_j(S_j) + N_j \ , \tag{3.6}$$

where f_j is a deterministic function.

As a final assumption, we consider that for any fixed values of the shares, the leakage components are independent. That is, for every $(s_0, s_1, \ldots, s_d) \in \mathcal{S}^{d+1}$, the random variables $(L_j \mid S_j = s_j)$ are mutually independent. Under the Gaussian noise assumption, this simply means that the noises N_j are mutually independent, and that is why we shall refer this assumption as the *independent noises assumption*.

Remark 3.3.1. For the sake of simplicity, we consider that all the leakages L_j have the same dimension T. Note that our analysis could be easily extended to the general case where each leakage L_j has its own dimension T_j.

3.3.2 Side-Channel Attacks

In a side-channel attack (SCA), the adversary aims to extract information about k^* by monitoring the leakage of the shares. Specifically, the adversary observes several samples $\boldsymbol{\ell}_i \in \boldsymbol{\mathcal{L}}$ of the leakage $\boldsymbol{L}_{x_i,k^*}$, corresponding to some public input x_i that he may either choose or just know. According to the above leakage model, the leakage space $\boldsymbol{\mathcal{L}}$ is defined as $\boldsymbol{\mathcal{L}} = \mathbb{R}^{T\times(d+1)}$ and each leakage sample can be written as

$$\boldsymbol{\ell}_i = (\ell_{i,0}, \ell_{i,1}, \cdots, \ell_{i,d}) \ , \tag{3.7}$$

with $\ell_{i,j} \in \mathbb{R}^T$ for every j. Moreover, the Gaussian noise assumption implies that each leakage sample coordinate can be further written as

$$\ell_{i,j} = f_j(s_{i,j}) + n_{i,j} \ , \tag{3.8}$$

where $s_{i,1}$, $s_{i,2}$, …, $s_{i,d}$ are d random mask values, where $s_{i,0} = \varphi(x_i, k^*) \oplus \bigoplus_{j=1}^{d} s_{i,j}$, and where $n_{i,0}$, $n_{i,1}$, …, $n_{i,d}$ are samples of the Gaussian noises N_0, N_1, …, N_d.

Once several, say q, leakage samples have been collected, the adversary makes use of a *distinguisher*, that is a function mapping the input-leakage samples $(x_1, \boldsymbol{\ell}_1)$, $(x_2, \boldsymbol{\ell}_2)$, …, $(x_q, \boldsymbol{\ell}_q)$ to some *score vector* $\mathbf{d} = (d_k)_{k\in\mathcal{K}} \in \mathbb{R}^{|\mathcal{K}|}$. If the distinguisher is sound and if the leakage brings enough information on the shares, then the equality

$$k^* = \operatorname*{argmax}_{k\in\mathcal{K}} d_k$$

should hold with a probability substantially greater than $\frac{1}{|\mathcal{K}|}$.

In what follows, we shall consider a natural equivalence relation between distinguishers. We say that two score vectors are *rank-equivalent* if for every $n \in \{1, 2, \ldots, |\mathcal{K}|\}$, the n coordinates with highest scores are the same for the two vectors. Two distinguishers d and d' are then said *equivalent*, denoted $\mathsf{d} \equiv \mathsf{d}'$ if for every $(x_i, \boldsymbol{\ell}_i)_i \in (\mathcal{X} \times \boldsymbol{\mathcal{L}})^q$, the score vectors $\mathsf{d}\big((x_i, \boldsymbol{\ell}_i)_i\big)$ and $\mathsf{d}'\big((x_i, \boldsymbol{\ell}_i)_i\big)$ are rank-equivalent.

In this chapter, we focus on *additive distinguishers* which we formally define hereafter.

Definition 3.3.2. *A distinguisher* d *is* additive *if for every* $(x_1, x_2, \ldots, x_q) \in \mathcal{X}^q$*, there exists a family of functions* $\{g_{x,k} : \boldsymbol{\mathcal{L}} \to \mathbb{R}\ ;\ (x, k) \in \mathcal{X} \times \mathcal{K}\}$ *such that for every* $(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_q) \in \boldsymbol{\mathcal{L}}^q$ *we have*

$$\mathsf{d}\big((x_i, \boldsymbol{\ell}_i)_i\big) = (d_k)_{k\in\mathcal{K}} \quad \textit{with} \quad d_k = \frac{1}{q}\sum_{i=1}^{q} g_{x_i,k}(\boldsymbol{\ell}_i) \quad \textit{for every } k \in \mathcal{K}.$$

A distinguisher equivalent to an additive distinguisher as defined above is also said to be additive.

It was shown in [Riv09] that the widely used first-order correlation and likelihood distinguishers are both additive distinguishers in the sense of the above definition. We will show in Sections 3.5 and 3.6 that their high order counterparts are also additive.

3.4 Estimating the Success Rate

In this section, we generalize the methodology introduced in [Riv09] to high order side-channel attacks as modeled in the previous section. Namely, we show how to get a sound estimation of the attack success rate by studying the multivariate probability distribution of the score vector for the case of additive distinguishers.

The success rate of a high order side-channel attack, denoted $\mathsf{Succ}^{\mathsf{d}}_{\boldsymbol{x},k^*}$, is defined with respect to some input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_q)$, some secret k^*, and some distinguisher d, as the probability:

$$\mathrm{P}\Big[k^* = \underset{k\in\mathcal{K}}{\operatorname{argmax}}\ d_k \ \Big|\ \boldsymbol{\ell}_1 \overset{\$}{\leftarrow} \boldsymbol{L}_{x_1,k^*};\ \ldots;\ \boldsymbol{\ell}_q \overset{\$}{\leftarrow} \boldsymbol{L}_{x_q,k^*};\ (d_k)_{k\in\mathcal{K}} = \mathsf{d}\big((x_i, \boldsymbol{\ell}_i)_i\big)\Big] ,$$

where $\boldsymbol{\ell}_i \overset{\$}{\leftarrow} \boldsymbol{L}_{x_i,k^*}$ means randomly sampling $\boldsymbol{\ell}_i$ according to the distribution of $\boldsymbol{L}_{x_i,k^*}$.

Remark 3.4.1. For the sake of generality, we chose to fix the input vector $\boldsymbol{x}$ as a parameter of the attack so that we do not need to assume any specific strategy for the choice of the public inputs. However, we will investigate the particular setting where the x_i are uniformly distributed.

According to Definition 3.3.2, the score vector $(d_k)_{k\in\mathcal{K}}$ resulting from an additive distinguisher satisfies

$$d_k = \frac{1}{q}\sum_{i=1}^{q} g_{x_i,k}(\boldsymbol{\ell}_i) \ , \tag{3.9}$$

for some $g_{x,k} : \boldsymbol{\mathcal{L}} \to \mathbb{R}$. Then a simple application of the central limit theorem yields the following result, where we define the *occurrence ratio* τ_x of an element $x \in \mathcal{X}$ in the input vector $(x_1, x_2, \ldots, x_q)$ as

$$\tau_x = \frac{|\{i;\ x_i = x\}|}{q} \ . \tag{3.10}$$

Proposition 3.4.2. *The distribution of the score vector* $(d_k)_{k \in \mathcal{K}}$ *tends toward a multivariate Gaussian distribution as* q *grows, with expectation vector* $(\mathrm{E}\,[d_k])_{k \in \mathcal{K}}$ *satisfying*

$$\mathrm{E}\,[d_k] = \sum_{x \in \mathcal{X}} \tau_x \, \mathrm{E}\,[g_{x,k}(\boldsymbol{L}_{x,k^*})] \tag{3.11}$$

for every $k \in \mathcal{K}$*, and with covariance matrix* $(\mathrm{Cov}\,[d_{k_1}, d_{k_2}])_{(k_1,k_2) \in \mathcal{K}^2}$ *satisfying*

$$\mathrm{Cov}\,[d_{k_1}, d_{k_2}] = \frac{1}{q} \sum_{x \in \mathcal{X}} \tau_x \, \mathrm{Cov}\,[g_{x,k_1}(\boldsymbol{L}_{x,k^*}), g_{x,k_2}(\boldsymbol{L}_{x,k^*})] \tag{3.12}$$

for every $(k_1, k_2) \in \mathcal{K}^2$*.*

Proof. The first statement results by definition of additive distinguishers and the central limit theorem. Equation (3.11) and Equation (3.12) directly hold by mutual independence between the leakage samples. □

The above proposition shows that for a sufficient number of leakage observations, the distribution of the score vector $\boldsymbol{d} = (d_k)_{k \in \mathcal{K}}$ can be soundly estimated by a multivariate Gaussian. As in [Riv09], we now define the *comparison vector* as the $(|\mathcal{K}| - 1)$-size vector $\boldsymbol{c} = (c_k)_{k \in \mathcal{K}/\{k^*\}}$ whose coordinates satisfy

$$c_k = d_{k^*} - d_k \ , \tag{3.13}$$

for every $k \in \mathcal{K}/\{k^*\}$. The comparison vector is a linear transformation of the score vector by a $((|\mathcal{K}| - 1) \times |\mathcal{K}|)$-matrix P whose expression straightforwardly follows from Equation (3.13). This implies that the distribution of the comparison vector can also be soundly estimated by a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{m_c}, \boldsymbol{\Sigma_c})$ where $\boldsymbol{m_c} = P \cdot \boldsymbol{m_d}$ and $\boldsymbol{\Sigma_c} = P \cdot \boldsymbol{\Sigma_d} \cdot P'$. Moreover, by definition of the comparison vector, an attack is successful (*ie.* the correct secret k^* is ranked first in the score vector) if and only if all the coordinates of the comparison vector are positive. We deduce that the success rate $\mathsf{Succ}^{\mathsf{d}}_{\boldsymbol{x},k^*}$ of a distinguisher d satisfies

$$\mathsf{Succ}^{\mathsf{d}}_{\boldsymbol{x},k^*} \;=\; \mathrm{P}[\boldsymbol{c} > \boldsymbol{0}] \;\approx\; \Phi_{\boldsymbol{m_c},\boldsymbol{\Sigma_c}}(\boldsymbol{0}, \boldsymbol{\infty}) \tag{3.14}$$

where $\Phi_{\boldsymbol{m},\boldsymbol{\Sigma}}$ denotes the Gaussian cdf as defined in Equation (3.2), $\boldsymbol{0}$ denotes the null vector, and $\boldsymbol{\infty}$ denotes the vector $(\infty, \infty, \ldots, \infty)$.

Remark 3.4.3. In [SMY06], Standaert *et al.* propose to extend the notion of success rate to different orders. The o-th order success rate of a side-channel attack is defined as the probability that the target secret k^* is ranked among the o first key guesses by the score vector. The authors of [SMY06] also suggest to consider the so-called *guessing entropy*, which is defined as the expected rank of the good key guess in the score vector [Mas94; Cac97]. As shown in [Riv09], both the success rate of any order and the guessing entropy can be estimated using a similar approach as above.

Methodology.

According to the above analysis, we propose the following methodology for an evaluator of some cryptographic algorithm to estimate the success rate of a (HO-)SCA against his (masked) implementation. We consider that the evaluator has access to the random masks generated during the computation, and is therefore able to predict the value of each share involved in the successive execution of the protected algorithm. The methodology is composed of three main steps:

1. Profile the leakage of every share using standard estimation techniques. Under the Gaussian leakage assumption, this estimation amounts to compute the sample means and the sample covariance matrices of the leakage $(L_i \mid S_i = s)$ for every share S_i and every possible value $s \in \mathcal{S}$ based on a set of collected leakage samples. using for instance linear regression techniques as described in [SLP05].

2. Use Proposition 3.4.2 to compute the expectation vector and covariance matrix of the score vector with respect to the leakage parameters.

3. Deduce the parameters of the comparison vector distribution and evaluate the success rate according to Equation (3.14).

The precision of the obtained estimation is impacted by two main factors:

- the accuracy of the leakage parameter estimations, and

- the tightness of the Gaussian approximation arising in Proposition 3.4.2.

The accurate estimation of leakage parameters has been a widely investigated issue and efficient techniques are known to deal with it (see for instance [CRR03; SLP05; APSQ06; GLRP06]). Basically, the more noisy the leakage, the more samples must be used to get an accurate estimation. Note that in our approach, the evaluator only has to estimate first-order leakage parameters with respect to the share values. Practical aspects of leakage parameter estimation are further discussed in Section 3.8.

On the other hand, the Gaussian approximation is the main issue in our approach. One can fairly expect that if the considered implementation is not too weak, the convergence toward the Gaussian distribution should be rather fast compared to the number of leakage observations required to succeed the HO-SCA. In order to validate this intuition, we provide in Section 3.7 an empirical validation of the Gaussian approximation.

3.5 Application to the Correlation Distinguisher

In this section, we apply the general methodology described in Section 3.4 when the linear correlation coefficient is used as a distinguisher [BCO04]. To simplify the explanations and developments, we assume that the dimension T of the components L_j of the leakage vector $\boldsymbol{L}$ equals 1. This implies in particular that the L_j follow a univariate Gaussian distribution with zero mean and variance σ_j^2 (instead of a Gaussian multivariate distribution $\mathcal{N}(\mathbf{0}, \Sigma_j)$). The reasoning straightforwardly extends to the case $T > 1$. For two samples $\boldsymbol{x} = (x_1, x_2, \ldots, x_q) \in \mathbb{R}^q$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_q) \in \mathbb{R}^q$, the linear coefficient is defined by

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{\frac{1}{q}\sum_{i=1}^{q}(x_i - \overline{\boldsymbol{x}}) \cdot (y_i - \overline{\boldsymbol{y}})}{\sqrt{\frac{1}{q}\sum_i (x_i - \overline{\boldsymbol{x}})^2} \cdot \sqrt{\frac{1}{q}\sum_i (y_i - \overline{\boldsymbol{y}})^2}} \quad , \tag{3.15}$$

where $\overline{\boldsymbol{x}}$ (resp. $\overline{\boldsymbol{y}}$) denotes the sample mean $q^{-1}\sum_i x_i$ (resp. $q^{-1}\sum_i y_i$).

In the context of HO-SCA, the correlation coefficient is used together with a *model function* $\mathtt{m} : \mathcal{X} \times \mathcal{K} \mapsto \mathbb{R}$ and a *combining function* $\mathtt{C} : \boldsymbol{\mathcal{L}} \mapsto \mathbb{R}$ (see for instance [PRB10]). The combining function is involved to map a leakage sample into a univariate sample combining the leakages of the different shares. On the other hand, the model function computes some expected value for the combined leakage with respect to some input x and some guess k on the target secret. The correlation distinguisher $\mathsf{d}_{\mathsf{cor}}$ is then defined as

$$\mathsf{d}_{\mathsf{cor}}((x_i, \boldsymbol{\ell}_i)_i) = \rho((\mathtt{m}(x_i, k))_i, (\mathtt{C}(\boldsymbol{\ell}_i))_i) \ . \tag{3.16}$$

The following proposition extends the analysis conducted in [Riv09] and states that the (high order) correlation distinguisher $\mathsf{d}_{\mathsf{cor}}$ is additive. This particularly implies that the methodology described in Section 3.4 can be applied to this distinguisher.

Proposition 3.5.1. *For any model function* $\mathtt{m} : \mathcal{X} \times \mathcal{K} \mapsto \mathbb{R}$ *and any combining function* $\mathtt{C} : \boldsymbol{\mathcal{L}} \mapsto \mathbb{R}$, *the correlation distinguisher* $\mathsf{d}_{\mathsf{cor}}$ *is additive. Moreover,* $\mathsf{d}_{\mathsf{cor}}$ *is equivalent to the distinguisher* $\mathsf{d}'_{\mathsf{cor}}$ *defined for every* $(x_i, \boldsymbol{\ell}_i)_i \in (\mathcal{X} \times \boldsymbol{\mathcal{L}})^q$ *by*

$$\mathsf{d}'_{\mathsf{cor}}((x_i, \boldsymbol{\ell}_i)_i) = \Big(\frac{1}{q}\sum_{i=1}^{q} g_{x_i,k}(\boldsymbol{\ell}_i)\Big)_{k\in\mathcal{K}} \ ,$$

where the function $g_{x,k} : \boldsymbol{\mathcal{L}} \to \mathbb{R}$ *satisfies*

$$g_{x,k}(\boldsymbol{\ell}) = \frac{1}{s_k}\,(\mathtt{m}(x,k) - \overline{\mathtt{m}}_k) \cdot \mathtt{C}(\boldsymbol{\ell}) \ , \tag{3.17}$$

for every $(x, k) \in \mathcal{X} \times \mathcal{K}$, *with* $\overline{\mathtt{m}}_k = \frac{1}{q}\sum_i \mathtt{m}(x_i, k)$ *and* $s_k = \sqrt{\frac{1}{q}\sum_i (\mathtt{m}(x_i, k) - \overline{\mathtt{m}}_k)^2}$.

Proof. Let $(d_k)_{k\in\mathcal{K}} = \mathsf{d}_{\mathsf{cor}}((x_i, \boldsymbol{\ell}_i)_i)$ and $(d'_k)_{k\in\mathcal{K}} = \mathsf{d}'_{\mathsf{cor}}((x_i, \boldsymbol{\ell}_i)_i)$ for some input-leakage samples $(x_i, \boldsymbol{\ell_i})_{i\leqslant q} \in (\mathcal{X} \times \mathcal{L})^q$. We have:

$$d_k \ = \ \frac{1}{s_{\mathtt{C}}} \frac{\sum_{i=1}^{q}(\mathtt{m}(x_i,k) - \overline{\mathtt{m}}_k)\mathtt{C}(\boldsymbol{\ell_i})}{s_k} \ = \ \frac{1}{s_{\mathtt{C}}}\, d'_k \ ,$$

where $s_{\mathtt{C}} = \sqrt{\frac{1}{q}\sum_i (\mathtt{C}(\boldsymbol{\ell_i}) - \overline{\mathtt{C}})^2}$ with $\overline{\mathtt{C}} = \frac{1}{q}\sum_i \mathtt{C}(\boldsymbol{\ell_i})$.

Since $s_{\mathtt{C}}$ is strictly positive and constant with respect to the guess k, the score vectors $(d_k)_{k\in\mathcal{K}}$ and $(d'_k)_{k\in\mathcal{K}}$ are clearly rank-equivalent, implying that the distinguishers $\mathsf{d}_{\mathsf{cor}}$ and $\mathsf{d}'_{\mathsf{cor}}$ are equivalent. Moreover, after denoting by $g_{x,k}$ the function $\boldsymbol{\ell_i} \mapsto s_k^{-1}(\mathtt{m}(x,k) - \overline{\mathtt{m}}_k)\mathtt{C}(\boldsymbol{\ell_i})$, we get $d'_k = \frac{1}{q}\sum_{i=1}^{q} g_{x_i,k}(\boldsymbol{\ell_i})$, which implies that $\mathsf{d}'_{\mathsf{cor}}$ is additive. □

Remark 3.5.2. If we focus on the uniform setting where the input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_q)$ is balanced (meaning that each value $x \in \mathcal{X}$ have an occurrence ratio of $\tau_x = \frac{1}{|\mathcal{X}|}$), then $\overline{\mathtt{m}}_k$ and s_k are constant with respect to k and $\mathsf{d}_{\mathsf{cor}}$ is equivalent to another simpler distinguisher:

$$\mathsf{d}''_{\mathsf{cor}} : ((x_i, \boldsymbol{\ell}_i)_i) \mapsto \Big(\frac{1}{q}\sum_i \mathtt{m}(x_i, k) \cdot \mathtt{C}(\boldsymbol{\ell_i})\Big)_{k\in\mathcal{K}} \ . \tag{3.18}$$

Confusion coefficient

We propose hereafter to show how our approach can be seen as an extension to the work of Fei *et al.* [FLD12], which is itself an extension of the notion of *transparency order* introduced in [Pro05] (see also the recent publication [CSM+14], which redefines this notion). We apply our approach to the first-order attack, and illustrate how the notion of confusion coefficient can be extended in this setting. In [FLD12], the confusion coefficient associated to a key k was defined for the mono-bit DPA as the probability on x that the prediction $\mathtt{m}(x, k)$ differs from the *correct* prediction $\mathtt{m}(x, k)$. In a multi-bit setting, we modify this definition in order to accurately capture the confusing effect of the cryptographic operation between predicted values.

Definition 3.5.3. *Let k^* denote the correct key of a cryptographic implementation, and k any key in the set of possible keys $\mathcal{K}$. Let x denote an element of the set of possible inputs $\mathcal{X}$. Let $\mathtt{m}$ denote a modeling function. The CPA confusion coefficient κ_k is then defined as:*

$$\kappa_k = \frac{1}{\#\mathcal{X}} \sum_{x \in \mathcal{X}} \mathtt{m}(x,k)\mathtt{m}(x,k^*).$$

Remark 3.5.4. In the case of a first-order CPA, the confusion coefficient allows for the expression of the parameters of the multivariate normal law followed by $\mathsf{d}''_{\mathsf{cor}}$, hence recovering the results of [Riv09]. Indeed, the expectation of the score d''_k associated to a key hypothesis k is equal to the confusion coefficient κ_k. Furthermore, in the particular case where all occurrences ratios are equal, the covariance matrix associated to the distribution of $\mathsf{d}''_{\mathsf{cor}}$ can also be expressed with confusion coefficients: namely, each coefficient of coordinate (i,j) of this matrix is equal to $\frac{\sigma^2}{N}\kappa_{i\oplus j}$.

In Figure 3.3 we illustrate the CPA confusion coefficient in the case where $\mathtt{m}$ is the composition of the Hamming weight with some classical sboxes. This figure, along with Remark 3.5.4 gives a straightforward intuition on why CPA performs differently depending on the targeted sbox. Indeed, one can observe a stronger dispersion in the DES and PRESENT sboxes than in the AES sbox.

Application to the Normalized Product Combining.

Let us now study the particular case of the high order correlation distinguisher based on the *centered product* combining function [PRB10]. This combining function is defined for univariate share leakages (*ie.* for $T = 1$ in the model of Section 3.3), namely its domain is $\boldsymbol{\mathcal{L}} = \mathbb{R}^{d+1}$. For every $(\ell_0, \ell_1, \ldots, \ell_d) \in \boldsymbol{\mathcal{L}}$, it is defined as

$$\mathtt{C}(\ell_0, \ell_1, \ldots, \ell_d) = \prod_{j=0}^{d} (\ell_j - \mu_j) \ , \tag{3.19}$$

where μ_j denotes the leakage expectation $\mathrm{E}[L_j]$.

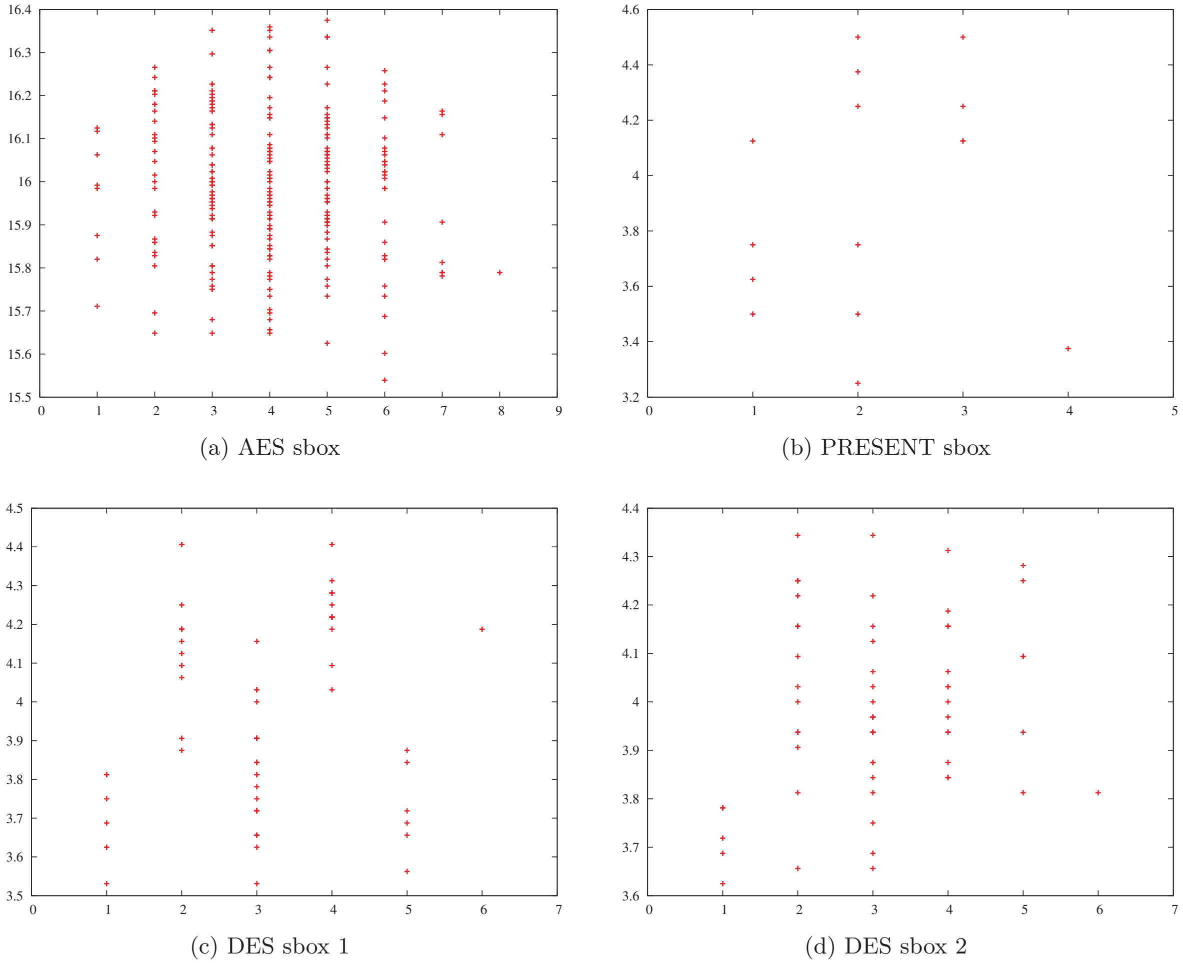
Note that in practice, the adversary does not know the exact expectation μ_j but he can estimate it based on leakage samples. As argued in [PRB10], the number of leakage samples required to succeed a HO-SCA is substantially greater than the number of leakage samples required to get precise estimations of the expectations μ_j. Therefore, we can soundly assume that the μ_j in Equation (3.19) are the exact expectations $\mathrm{E}[L_j]$. Note that replacing them by inexact estimations would not change our analysis.

We recall that, according to the leakage model presented in Section 3.3.1, the j-th leakage component L_j satisfies $L_j = f_j(S_j) + N_j$ where $f_j : s \mapsto m_{j,s}$ and $N_j \sim \mathcal{N}(0, \sigma_j^2)$. Since the noise N_j is centered in 0, we have $\mathrm{E}[f_j(S_j)] = \mathrm{E}[L_j] = \mu_j$. Moreover, we shall denote $\nu_j = \mathrm{Var}[f_j(S_j)]$. By uniformity of S_j over $\mathcal{S}$, we have:

$$\mu_j = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} m_{j,s} \quad \text{and} \quad \nu_j = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} (m_{j,s} - \mu_j)^2 \ . \tag{3.20}$$

In the following we shall further denote, for every $s \in \mathcal{S}$,

$$\alpha_s := \frac{1}{|\mathcal{S}|^d} \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \cdots \sum_{s_d \in \mathcal{S}} \prod_{j=0}^{d} (m_{j,s_j} - \mu_j) \tag{3.21}$$



(a) AES sbox

(b) PRESENT sbox

(c) DES sbox 1

(d) DES sbox 2

Figure 3.3: Values of κ_δ under the assumption that φ is the Hamming weight function, for different sboxes S, in function of the Hamming weight of δ.

and

$$\beta_s := \frac{1}{|\mathcal{S}|^d} \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \cdots \sum_{s_d \in \mathcal{S}} \prod_{j=0}^{d} (m_{j,s_j} - \mu_j)^2 \tag{3.22}$$

where $s_0 = s \oplus \bigoplus_{j=1}^{d} s_j$.

Note that both Equation (3.21) and Equation (3.22) can be expressed as a higher-order convolution product of the form

$$H(s) = \sum_{s_1} \sum_{s_2} \cdots \sum_{s_d} h_0(s \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_d) \cdot h_1(s_1) \cdot h_2(s_2) \cdots h_d(s_d)\ . \tag{3.23}$$

We show in proposition 3.5.5 how such a convolution can be efficiently computed for all values over $\mathcal{S}$ in $O(d \cdot |\mathcal{S}| \cdot \log |\mathcal{S}|)$ operations.

Proposition 3.5.5. *Let d be a positive integer, and let $(\mathcal{S}, \oplus)$ be a group of size $|\mathcal{S}| = 2^m$. Let $(h_j)_{0 \leq j \leq d}$ be a family of functions from $\mathcal{S}$ into $\mathbb{R}$, such that $h_j(s)$ can be efficiently evaluated for* *every $s \in \mathcal{S}$ in $o(1)$ operations (one typically has a look-up table for every h_j). Consider the function $H : \mathcal{S} \to \mathbb{R}$ defined as*

$$H : s \mapsto \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \cdots \sum_{s_d \in \mathcal{S}} h_0(s \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_d) \cdot h_1(s_1) \cdot h_2(s_2) \cdots h_d(s_d) \ .$$

Then, the whole set of outputs $\{H(s) \ ; \ s \in \mathcal{S}\}$ can be computed in $O(d \cdot 2^m \cdot m)$ operations.

Proof. For every $s \in \mathcal{S}$, the function H satisfies

$$H(s) = \sum_{s_d \in \mathcal{S}} h_d(s_d) \cdots \sum_{s_2 \in \mathcal{S}} h_2(s_2) \sum_{s_1 \in \mathcal{S}} h_1(s_1) \cdot h_0(s \oplus s_1 \oplus s_2 \oplus \cdots s_d)$$

Consider the convolution product of the form

$$h_1 \otimes h_0 : s \mapsto \sum_{t \in \mathcal{S}} h_1(t) \cdot h_0(s \oplus t) \ .$$

We have

$$\mathcal{WH}(h_1 \otimes h_0) = 2\,\mathcal{WH}(h_1) \star \mathcal{WH}(h_2) \ ,$$

where $\mathcal{WH}$ is the Walsh-Hadamard transform (WHT), and $\star$ denotes the coordinate-wise multiplication (or Schur product). This convolution product can hence be efficiently computed from three evaluations of fast WHT[2] that each takes $O(2^m \cdot m)$ operations.

One can check that the sequence of functions $(H_i)_{0 \le i \le d}$ defined as

$$\begin{cases} H_0 = h_0 \\ H_i = h_i \otimes H_{i-1} \quad \text{for every } i \geqslant 1 \end{cases}$$

is such that $H_d = H$. One can then sequentially compute the set of outputs of H_1, H_2, …, $H_d = H$ by evaluating d convolution products, which gives a total cost of $O(d \cdot 2^m \cdot m)$ operations. □

We then have the following corollary of Proposition 3.4.2 for the distinguisher $\mathsf{d}'_{\mathsf{cor}}$ with centered product combining function.

Corollary 3.5.6. *Let $k^* \in \mathcal{K}$, let $(x_1, x_2, \ldots, x_q) \in \mathcal{X}^q$ and let $\boldsymbol{\ell}_i \xleftarrow{\$} \boldsymbol{L}_{x_i,k^*}$ for every $i \in \{1, 2, \ldots, q\}$. Then the distribution of the score vector $(d'_k)_{k \in \mathcal{K}} = \mathsf{d}'_{\mathsf{cor}}((x_i, \boldsymbol{\ell}_i)_i)$ with centered product combining function tends toward a multivariate Gaussian distribution with expectation vector $(\mathrm{E}[d'_k])_{k \in \mathcal{K}}$ satisfying*

$$\mathrm{E}\left[d'_k\right] = \sum_{x \in \mathcal{X}} \tau_x \, \mathtt{M}(x,k) \, \alpha_{\varphi(x,k^*)} \ , \tag{3.24}$$

for every $k \in \mathcal{K}$, and with covariance matrix $(\mathrm{Cov}\left[d'_{k_1}, d'_{k_2}\right])_{(k_1,k_2) \in \mathcal{K}^2}$ satisfying

$$\mathrm{Cov}\left[d'_{k_1}, d'_{k_2}\right] = \frac{1}{q} \sum_{x \in \mathcal{X}} \tau_x \, \mathtt{M}(x,k_1) \, \mathtt{M}(x,k_2) \times \left(\beta_{\varphi(x,k^*)} - \alpha^2_{\varphi(x,k^*)} + \prod_{j=0}^{d} (\nu_j + \sigma_j^2) - \prod_{j=0}^{d} \nu_j \right) \ , \tag{3.25}$$

for every $(k_1, k_2) \in \mathcal{K}^2$, where

$$\mathtt{M} : (x,k) \mapsto \frac{\mathtt{m}(x,k) - \overline{\mathtt{m}}_k}{s_k} \ . \tag{3.26}$$

[2] The WHT is involutive, hence we have $h_1 \otimes h_0 = 2\,\mathcal{WH}\big(\mathcal{WH}(h_1) \star \mathcal{WH}(h_2)\big)$.

Proof. To prove the corollary, we first introduce the following lemma.

Lemma 3.5.7. *The expectation and variance of the random variable* $\mathtt{C}(\boldsymbol{L}_{x,k^*})$ *respectively satisfy*

$$\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] = \alpha_{\varphi(x,k^*)} \tag{3.27}$$

and

$$\mathrm{Var}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] = \beta_{\varphi(x,k^*)} - \alpha^2_{\varphi(x,k^*)} + \prod_{j=0}^{d}(\nu_j + \sigma_j^2) - \prod_{j=0}^{d}\nu_j \ . \tag{3.28}$$

Proof. Since the N_j are independent and centered in 0, we have

$$\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] = \mathrm{E}\left[\mathtt{C}(f_0(S_0), f_1(S_1), \ldots, f_d(S_d))^2\right] = \alpha_{\varphi(x,k^*)} \ ,$$

On the other hand, by definition of the variance, we have

$$\mathrm{Var}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] = \mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})^2\right] - \mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right]^2 = \mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})^2\right] - \alpha^2_{\varphi(x,k^*)} \ .$$

Then, we have

$$\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})^2\right] = \mathrm{E}\left[\prod_{j=0}^{d}(f_j(S_j) + N_j - \mu_j)^2\right] = \mathrm{E}\left[\prod_{j=0}^{d}((f_j(S_j) - \mu_j)^2 + N_j^2)\right]$$

where the second holds since the N_j have zero means and are mutually independent and independent of the S_j. By developing the product, we get a sum of monomials, such that each monomial involves random variables that are mutually independent, except for one single monomial which is $\prod_{j=0}^{d}(f_j(S_j) - \mu_j)^2$. We can then develop the above equation as

$$\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})^2\right] = \prod_{j=0}^{d}(\mathrm{E}\left[(f_j(S_j) - \mu_j)^2\right] + \mathrm{E}\left[N_j^2\right]) \\ - \prod_{j=0}^{d}\mathrm{E}\left[(f_j(S_j) - \mu_j)^2\right] + \mathrm{E}\left[\prod_{j=0}^{d}(f_j(S_j) - \mu_j)^2\right] \ ,$$

which gives

$$\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})^2\right] = \prod_{j=0}^{d}(\nu_j + \sigma_j^2) - \prod_{j=0}^{d}\nu_j + \beta_{\varphi(x,k^*)}.$$

□

Proof of Corollary 3.5.6. Applying Equation (3.11) and Equation (3.12) to the functions $g_{x,k} : \boldsymbol{\ell} \mapsto \frac{1}{s_k}(\mathtt{m}(x,k) - \overline{\mathtt{m}}_k) \cdot \mathtt{C}(\boldsymbol{\ell})$ as defined in Equation (3.17), we get

$$\mathrm{E}\left[d'_k\right] = \frac{1}{s_k}\sum_{x\in\mathcal{X}}\tau_x\,(\mathtt{m}(x,k) - \overline{\mathtt{m}}_k)\,\mathrm{E}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] \ ,$$

and

$$\mathrm{Cov}\left[d'_{k_1}, d'_{k_2}\right] = \frac{1}{q}\frac{1}{s_{k_1}s_{k_2}}\sum_{x\in\mathcal{X}}\tau_x\,(\mathtt{m}(x,k_1) - \overline{\mathtt{m}}_{k_1})\,(\mathtt{m}(x,k_2) - \overline{\mathtt{m}}_{k_2})\mathrm{Var}\left[\mathtt{C}(\boldsymbol{L}_{x,k^*})\right] \ ,$$

Then Lemma 3.5.7 directly yields the corollary statement. □

□

Remark 3.5.8. For the distinguisher $\mathsf{d}''_{\mathsf{cor}}$ defined in Equation (3.18) and which is equivalent to the correlation distinguisher in the uniform setting (see Remark 3.5.2), we have the same result as in Corollary 3.5.6 but the function $\mathtt{M}$ is simply defined as the model function $\mathtt{m}$.

According to Corollary 3.5.6, the methodology presented in Section 3.4 can be applied to estimate the success rate of a HO-SCA based on the correlation distinguisher with centered product combining. The first step of the methodology shall provide estimations of the leakage functions $f_j : s \mapsto m_{j,s}$ (and hence of the corresponding μ_j and ν_j), while the second step shall simply consist in the evaluations of Equation (3.24) and Equation (3.25).

3.6 Application to the Likelihood Distinguisher

In this section, we apply the general methodology described in Section 3.4 when the likelihood is used as distinguisher [CRR03]. The likelihood distinguisher, denoted $\mathsf{d}_{\mathsf{lik}}$, is usually applied after a *profiling step* whose goal is to provide an estimation $\hat{\mathsf{p}}_s$ of the pdf of the random variable $\boldsymbol{L}_s$ for every $s \in \mathcal{S}$. Then, for every sample $(x_i, \boldsymbol{\ell_i})_i \in (\mathcal{X} \times \boldsymbol{\mathcal{L}})^q$, the likelihood distinguisher is defined as

$$\mathsf{d}_{\mathsf{lik}}((x_i, \boldsymbol{\ell_i})_i) = \prod_{i=1}^{q} \hat{\mathsf{p}}_{\varphi(x_i,k)}(\boldsymbol{\ell_i}) \quad . \tag{3.29}$$

In practice, one often makes use of the equivalent (averaged) log-likelihood distinguisher $\mathsf{d}'_{\mathsf{lik}}$ defined as

$$\mathsf{d}'_{\mathsf{lik}}((x_i, \boldsymbol{\ell_i})_i) = \frac{1}{q} \log \mathsf{d}_{\mathsf{lik}}((x_i, \boldsymbol{\ell_i})_i) = \frac{1}{q} \sum_{i=1}^{q} \log(\hat{\mathsf{p}}_{\varphi(x_i,k)}(\boldsymbol{\ell_i})) \quad . \tag{3.30}$$

The log-likelihood distinguisher is usually preferred as it less susceptible to approximation errors than the likelihood. We straightforwardly get the following proposition.

Proposition 3.6.1. *The likelihood distinguisher* $\mathsf{d}_{\mathsf{lik}}$ *is additive and equivalent to the log-likelihood distinguisher* $\mathsf{d}'_{\mathsf{lik}}$*. Moreover, for every* $(x_i, \boldsymbol{\ell}_i)_i \in (\mathcal{X} \times \boldsymbol{\mathcal{L}})^q$*,* $\mathsf{d}'_{\mathsf{lik}}$ *satisfies*

$$\mathsf{d}'_{\mathsf{lik}}((x_i, \boldsymbol{\ell_i})_i) = \Big(\frac{1}{q} \sum_{i=1}^{q} g_{x_i,k}(\boldsymbol{\ell_i})\Big)_{k \in \mathcal{K}} \quad , \tag{3.31}$$

where the function $g_{x,k} : \boldsymbol{\mathcal{L}} \to \mathbb{R}$ *satisfies*

$$g_{x,k}(\boldsymbol{\ell}) = \log(\hat{\mathsf{p}}_{\varphi(x,k)}(\boldsymbol{\ell})) \quad , \tag{3.32}$$

for every $(x, k) \in \mathcal{X} \times \mathcal{K}$*.*

Under the Gaussian leakage assumption, it can be checked that the variable $\boldsymbol{L}_s$ has a Gaussian mixture distribution, with pdf p_s satisfying

$$\mathsf{p}_s : (\ell_0, \ell_1, \ldots, \ell_d) \mapsto \frac{1}{|\mathcal{S}|^d} \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S}} \cdots \sum_{s_d \in \mathcal{S}} \prod_{j=0}^{d} \phi_{\boldsymbol{m}_{j,s_j}, \boldsymbol{\Sigma}_j}(\ell_j) \quad , \tag{3.33}$$

where $s_0 = s \oplus \bigoplus_{j=1}^{d} s_j$. Note that for every $s \in \mathcal{S}$, the estimated pdf $\hat{\mathsf{p}}_s$ obtained from the profiling phase has a similar expression as p_s but with estimations $\hat{\boldsymbol{m}}_{j,s_j}$ and $\hat{\boldsymbol{\Sigma}}_j$ for the leakage means and covariance matrices.

Here again, it can be seen from Equation (3.33) that for a given $\boldsymbol{\ell} \in \boldsymbol{\mathcal{L}}$ the probability $\mathsf{p}_s(\boldsymbol{\ell})$ is a higher-order convolution product as in Equation (3.23). The set of probability values $\{\mathsf{p}_s(\boldsymbol{\ell})\ ;\ s \in \mathcal{S}\}$ can then be computed in $O(d \cdot |\mathcal{S}| \cdot \log |\mathcal{S}|)$ operations thanks to Proposition 3.5.5.

Let us now consider the two functions:

$$\lambda(s_1, s_2) := \int_{\boldsymbol{\ell} \in \boldsymbol{\mathcal{L}}} \log(\hat{\mathsf{p}}_{s_1}(\boldsymbol{\ell}))\, \mathsf{p}_{s_2}(\boldsymbol{\ell})\, d\boldsymbol{\ell} \ , \tag{3.34}$$

and

$$\psi(s_1, s_2, s_3) := \int_{\boldsymbol{\ell} \in \boldsymbol{\mathcal{L}}} \log(\hat{\mathsf{p}}_{s_1}(\boldsymbol{\ell}))\ \log(\hat{\mathsf{p}}_{s_2}(\boldsymbol{\ell}))\, \mathsf{p}_{s_3}(\boldsymbol{\ell})\, d\boldsymbol{\ell} \ . \tag{3.35}$$

Then, by definition, we have

$$\Lambda(x, k, k^*) := \lambda(\varphi(x, k), \varphi(x, k^*)) = \mathrm{E}\left[g_{x,k}(\boldsymbol{L}_{x,k^*})\right]$$

and

$$\begin{aligned}\Psi(x, k_1, k_2, k^*) \ &:= \ \psi(\varphi(x, k_1), \varphi(x, k_2), \varphi(x, k^*)) \\ &= \mathrm{E}\left[g_{x,k_1}(\boldsymbol{L}_{x,k^*}) \cdot g_{x,k_2}(\boldsymbol{L}_{x,k^*})\right] \ .\end{aligned}$$

A direct application of Proposition 3.4.2 then yields the following corollary for the log-likelihood distinguisher.

Corollary 3.6.2. *Let $k^* \in \mathcal{K}$, let $(x_1, x_2, \ldots, x_q) \in \mathcal{X}^q$ and let $\boldsymbol{\ell}_i \overset{\$}{\leftarrow} \boldsymbol{L}_{x_i,k^*}$ for every $i \in \{1, 2, \ldots, q\}$. Then the distribution of the score vector $(d'_k)_{k\in\mathcal{K}} = \mathsf{d}'_{\mathsf{lik}}((x_i, \boldsymbol{\ell}_i)_i)$ tends toward a multivariate Gaussian distribution with expectation vector $(\mathrm{E}\left[d'_k\right])_{k\in\mathcal{K}}$ satisfying*

$$\mathrm{E}\left[d'_k\right] = \sum_{x\in\mathcal{X}} \tau_x\, \Lambda(x, k, k^*) \ , \tag{3.36}$$

for every $k \in \mathcal{K}$, and with covariance matrix $(\mathrm{Cov}\left[d'_{k_1}, d'_{k_2}\right])_{(k_1,k_2)\in\mathcal{K}^2}$ satisfying

$$\mathrm{Cov}\left[d'_{k_1}, d'_{k_2}\right] = \frac{1}{q} \sum_{x\in\mathcal{X}} \tau_x(\Psi(x, k_1, k_2, k^*) - \Lambda(x, k_1, k^*) \cdot \Lambda(x, k_2, k^*)) \ . \tag{3.37}$$

According to Corollary 3.6.2, the methodology presented in Section 3.4 can be applied to estimate the success rate of a HO-SCA based on the likelihood distinguisher.

3.7 Empirical Validation of the Gaussian Approximation

In Section 3.4, we have presented a methodology to estimate the success rate of side-channel attacks based on so-called additive distinguishers. The principle of this methodology is to approximate the distribution of the score vector by a multivariate Gaussian distribution whose parameters are derived from the leakage parameters. This Gaussian approximation is asymptotically sound by the central limit theorem. However, in the non-asymptotic context of a SCA with a given number of leakage samples, it is fair to question whether this approximation is sound or not. In this section, we conduct an empirical study of the Gaussian approximation. For this purpose, we compare the success rates obtained from attack simulations, to the success rates obtained by applying the methodology of Section 3.4.

Since our purpose here is the sole validation of the Gaussian approximation, we do not focus on the leakage estimation issue, but we assume the exact leakage parameters $\{(m_{j,s},\sigma_j^2)\ ;\ 0 \leqslant j \leqslant d,\ s \in \mathcal{S}\}$ are known (in a univariate setting). From these leakage parameters, and for a given HO-SCA based on some distinguisher $\mathsf{d} \in \{\mathsf{d}_{\mathsf{cor}}, \mathsf{d}_{\mathsf{lik}}\}$, we evaluate the success rate with the two following approaches:

- Simulation success rate. We perform several attack simulations and count the number of successes in order to get an estimation of the success rate. For each attack simulation, we randomly generate input-leakage samples $(x_1,\boldsymbol{\ell}_1)$, $(x_2,\boldsymbol{\ell}_2)$, $\ldots$, $(x_q,\boldsymbol{\ell}_q)$. Specifically, for every i, x_i is uniformly picked up and $\boldsymbol{\ell}_i$ is randomly sampled from the variable $\boldsymbol{L}_{x_i,k^*}$ according to the leakage parameters. Then we apply the distinguisher d to these samples, and we count a success whenever the good secret is ranked first.

- Gaussian cdf evaluation. We apply Corollaries 3.5.6 and 3.6.2 to compute the expectation vector and covariance matrix of the score vector with respect to the leakage parameters and taking $\tau_x = 1/|\mathcal{X}|$ as occurrence ratio for every $x \in \mathcal{X}$ (in accordance to the uniform distribution of the x_i). Then we compute the Gaussian cdf of the comparison vector to evaluate the success rate according to Equation (3.14).

We plot hereafter the results obtained with these two approaches for different HO-SCA targeting an AES Sbox output:

$$\varphi(x,k^*) = \mathsf{SB}(x \oplus k^*)\ ,$$

where SB denote the AES Sbox function. For the leakage parameters, we used sample means and sample variances obtained by monitoring the leakage of two different devices running masked AES implementations (Device A and Device B, see Section 3.8 for details).

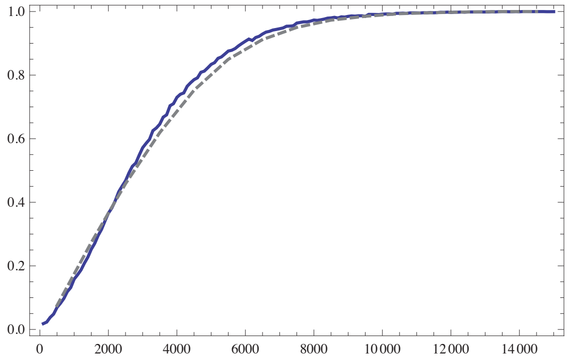


Figure 3.4: Simulation SR (plain curve) *vs.* theoretical SR (dashed curve) for 2nd-order correlation attack.

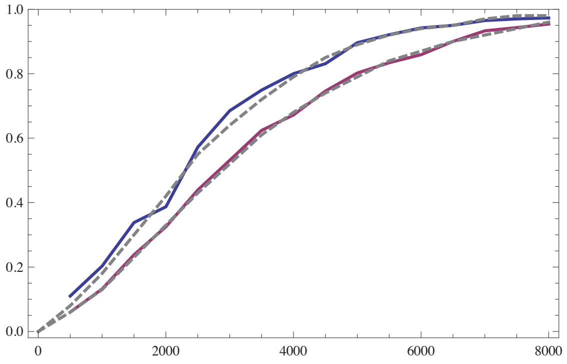


Figure 3.5: Simulation SR (plain curves) *vs.* theoretical SR (dashed curves) for 2nd-order likelihood attacks.

Figure 3.4 shows the results obtained for a second-order correlation attack with centered product combining function and Hamming weight model function (*ie.* $\mathtt{m} = \mathtt{HW}$), for leakage parameters from Device A. Figure 3.5 plots the results of a second-order likelihood attack with the same leakage parameters, assuming a perfect profiling (*ie.* $\hat{\mathsf{p}}_s = \mathsf{p}_s$ for every s) on the one hand and a slightly erroneous profiling on the other hand.[3] We observe that for both distinguishers, the experimental

[3] Specifically, we introduce random errors in the $(m_{j,s})_{j,s}$ used in the estimated pdfs $\hat{\mathsf{p}}_s$.

success rate curves and theoretical success rate curves clearly match. This validates the Gaussian approximation in these HO-SCA contexts.

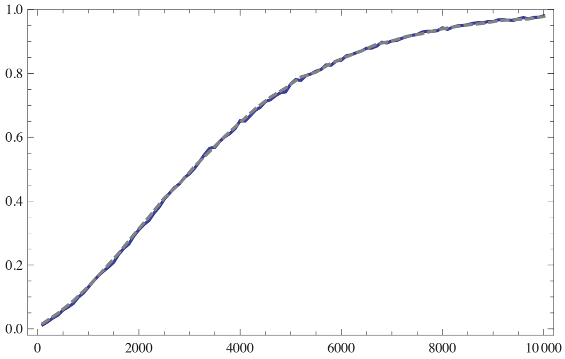


Figure 3.6: Simulation SR (plain curve) *vs.* theoretical SR (dashed curve) for 3rd-order correlation attack.

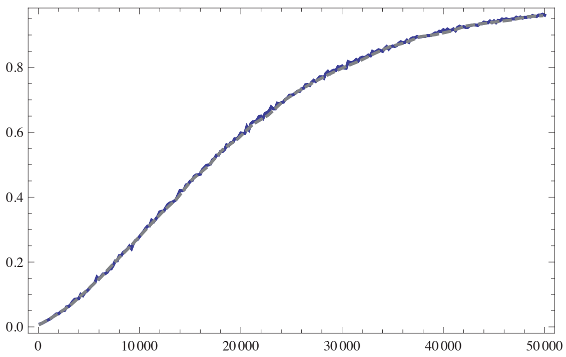


Figure 3.7: Simulation SR (plain curve) *vs.* theoretical SR (dashed curve) for 4th-order correlation attack.

In order to test the Gaussian approximation to high orders, we also performed third-order and fourth-order attacks, with leakage parameters from Device B. The results of the correlation attacks (centered product combining function and Hamming weight model function) are presented in Figure 3.6 and Figure 3.7 respectively. We see that the curves perfectly match, which further validates the Gaussian approximation in these high order contexts.

3.8 Practical Experiments

In this section, we confront our methodology to practical attack experiments. We report the results of several high order correlation attacks against two different devices running masked AES implementations. We also apply our methodology to estimate the expected success rate of these attacks with respect to the inferred leakage parameters.

Experimental setup. Practical experiments were performed on two microcontrollers made in different CMOS technologies (130 and 350 nanometer processes, respectively called devices A and device B in the sequel). The side-channel traces were obtained by measuring the electromagnetic (EM) radiations emitted by the device during a masked AES-128 encryption handling one byte at a time. To this aim, an EM sensor was used (made of several coils of copper with diameter of 500μm), and was plugged into a low-noise amplifier. To sample the leakage measurements, a digital oscilloscope was used with a sampling rate of 10G samples per second for the device A and 2G samples per second for the device B, whereas microcontrollers were running at few dozen of MHz. As the microcontrollers clocks were not stable, we had to resynchronize the EM traces. This process is out of the scope of this work, but we would like to emphasize that resynchronization is always required in a practical context and it has a non negligible impact on the measurements noise.

In our attack context, the random values involved in the masking/sharing could be known by the evaluator and we used this ability to identify the time samples corresponding to the different manipulation of the different shares. This step allowed us to associate each share to a unique time sample (the one with maximal SNR) and to profile the leakage parameters.[4]

Estimation of the leakage parameters. To estimate the leakage functions $f_j : s \mapsto m_{j,s}$, we applied linear regression techniques on 200000 leakage samples. When applied on leakage samples $\ell_{1,j}$, $\ell_{2,j}$, ..., $\ell_{q,j}$, corresponding to successive share values $s_{1,j}$, $s_{2,j}$, ..., $s_{q,j}$, a linear regression *of degree t* returns an approximation of $f_j(s)$ as a degree-t polynomial in the bits of s (see [SLP05; DPRS11] for more detail on linear regression in the context of side-channel attacks). We applied linear regression of degree 1 and 2 on Device A and B respectively. Once the f_j function estimated, we could easily get an estimation for the variance σ_j^2 of the noise N_j by computing the sample variance of $(\ell_{i,j} - f_j(s_{i,j}))_i$ for every j.

Methodology *versus* practice.

In order to validate our methodology in practice, we performed high order correlation attacks with centered product combining function (see Section 3.5) and Hamming weight model function (*ie.* $\mathtt{m} = \mathtt{HW}$). On the other hand, the success rate was estimated using the methodology described in Sections 3.4 and 3.5 by computing the parameters of the multivariate Gaussian distribution arising for the correlation distinguisher with respect to the inferred leakage parameters.

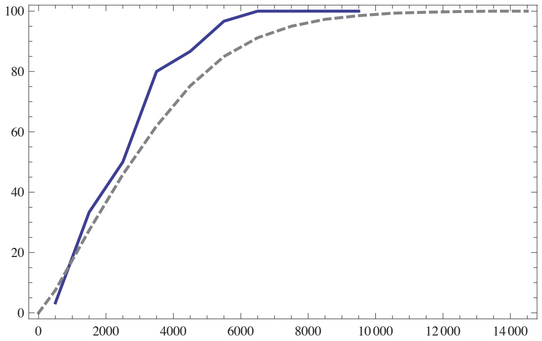


Figure 3.8: Experimental SR (plain curve) *vs.* theoretical SR (dashed curve) for 2nd-order correlation attack on Device A.

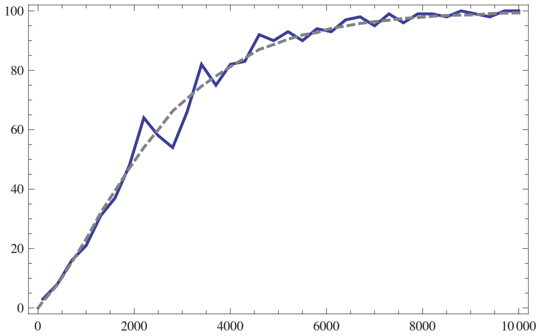


Figure 3.9: Experimental SR (plain curve) *vs.* theoretical SR (dashed curve) for 2nd-order correlation attack on Device B.

Figures 3.8 and 3.9 plot the experimental success rates *versus* the theoretical success rates for the second-order correlation attacks against Device A and Device B. In order to validate our approach with respect to high order attacks in practice, we also compare the results obtained with our methodology to third-order and fourth-order attack results on Device B (see Figures 3.10 and 3.11). We observe a clear match between the experimental and theoretical success rate curves. These results demonstrate the soundness of the methodology in practice.

Impact of the leakage profiling.

In order to observe the impact of the leakage profiling phase on our methodology, we applied it using a lower profiling rate. In order to determine the minimal number of samples required for a sound estimation, we first observed the convergence of the linear regression. Figure 3.12 plots the

[4] The knowledge of the masks was however not used in the attack phase itself.

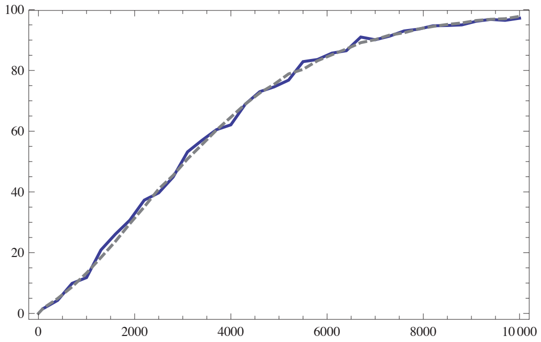


Figure 3.10: Experimental SR (plain curve) *vs.* theoretical SR (dashed curve) for 3rd-order correlation attack on Device B.

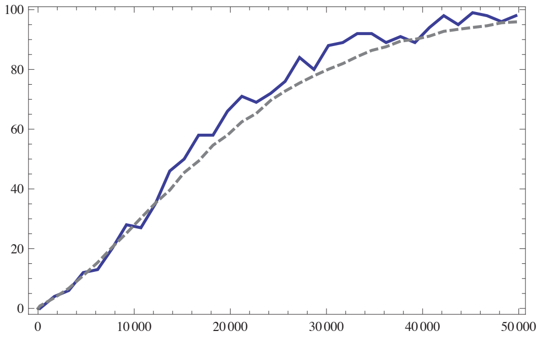


Figure 3.11: Experimental SR (plain curve) *vs.* theoretical SR (dashed curve) for 4th-order correlation attack on Device B.

8 coefficients of degree one (the dominant coefficients) of the function f_0 obtained from the linear regression with respect to the number of used samples. We observe that the coefficients converge after around 1200 samples. Then we applied our methodology to estimate the success rate of the second-order correlation attack on Device B based on a profiling using either 400 or 1500 samples (instead of 200000 samples). The results are presented in Figure 3.13. The plain curves represents the experimental success rate and theoretical success rate with full profiling (200000 samples), while the dotted and dashed curves represent the theoretical success rates with profiling based on 400 and 1500 samples respectively. We see that our methodology still matches quite well for a profiling based on 1500 samples but clearly fails for a profiling based on 400 samples. This shows that it is sound to study the convergence of the linear regression to determine the number of samples required for a sound estimation of the success rate.

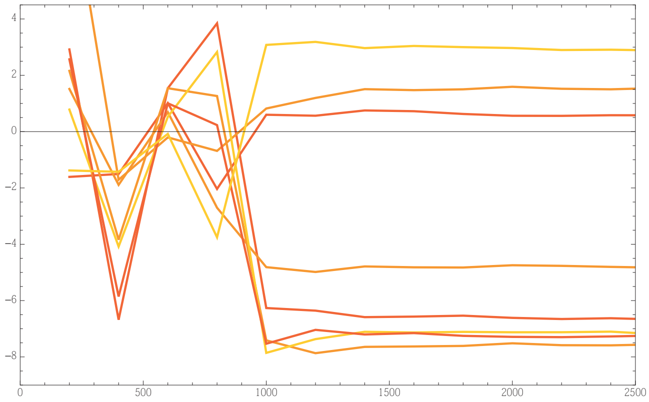


Figure 3.12: Convergence of linear regression coefficients.

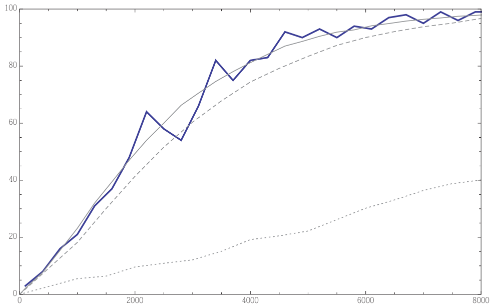
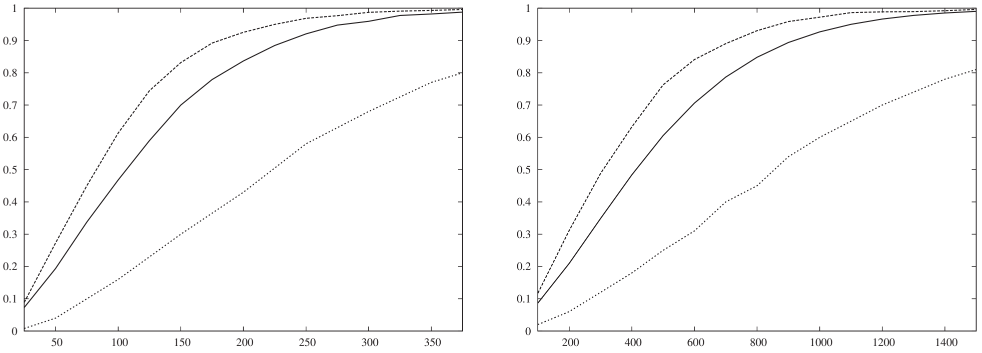


Figure 3.13: 2nd-order correlation attack on Device B: experimental SR *vs.* theoretical SR for different amounts of profiling samples.

3.9 Confidence in a result

When performing an attack without the knowledge of the correct subkey k^*, the adversary needs to determine *how* to select the most likely hypothesis, and *when* (*ie.* after which number of observations). Usually, the *how* problem is answered by using a *selection rule*, such as "choosing the best ranked subkey". This rule however does not deal with the *when* problem: it simply processes the result of the attack and *always* returns a candidate key. This type of selection rule is called an *unconditioned selection rule*. To answer the *when* problem, one should condition this rule by the observation of some pattern, like the stabilization of the rank of the best hypothesis. Figure 3.14 aims at experimentally validating the latter approach. In the first case, we perform several CPA against an AES Sbox output using an increasing number q of observations and we compute the attack success rate as a function of q. In the second case, we perform the same CPA but we output a candidate subkey only if it has been ranked first both with q and $\frac{q}{2}$ observations. For the latter experiment, we plot the attack success rate considering either the total number of experiments in dotted light grey and considering only the experiments where a key candidate was output (*ie.* appeared ranked first with q and $\frac{q}{2}$ observations) in dashed light grey.



(a) Noise standard deviation $\sigma = 5$

(b) Noise standard deviation $\sigma = 10$

Figure 3.14: Probabilities of the correctness of the output of attacks in function of the number of observations q in different contexts: 1) the best ranked subkey is always returned (plain dark grey, 2)) the best ranked subkey is returned only when it was also ranked first with $\frac{q}{2}$ observations and the success is computed against the number of times both attacks returned the same result (dashed light grey) 3) the best ranked subkey is returned only when it was also ranked first with $\frac{q}{2}$ observations and the success is computed against the number of times the attack has been launched (dotted light grey).

As it can be seen on Figure 3.14, when it outputs a result, the attack based on the stabilization criterion has a better chance (up to 15%) to be correct. However, the success rate takes into account all executions of the attack. Hence, the success rate of this attack is significantly lower than the classical CPA success rate. To summarize, the candidate selection rule increases the *confidence* in the selected subkey but decreases the success rate. In fact, we argue here that the two notions are important when studying an attack effectiveness. When attacking several subkeys separately, the assessment of a wrong candidate as a subpart of the whole secret key will lead to an indubitable failure, whereas a subkey that is not found (because the corresponding partial attack does not give a satisfying confidence level) will be recovered by bruteforce.

In the following, we give a theoretical justification to this empirical and natural attack effectiveness improvement. To this end, we introduce the notion of *confidence*, which aims at helping the adversary to assess the success or failure of an attack with a known error margin.

3.9.1 Confidence in an hypothesis

Applying the notations introduced previously, we assume that a partial attack is performed on a set of q independent observations and aims at the recovery of a correct subkey k^* among a set of hypotheses. For our analysis, the score of each candidate is computed at different steps of the attack (*ie.* for an increasing number of traces). Namely, the scores are computed after $q_1 < q$ observations, then again after $q_2 > q_1$ observations, and so on until the q observations are considered. In the sequel, the attack on q_i observations is called the i-th attack. All those attacks result in a matrix $\mathcal{M}_s$ containing the scores $s(k, q_i)$ for every hypothesis k and every number q_i of observations. With this construction, the last column vector $(s(k, q))_k$ corresponds to the final attack scores, whereas $(s(k, q_i))_k$ corresponds to intermediate scores (for the i-th attack). In other terms, the right-column of $\mathcal{M}_s$ is the attack result, and the rest of the matrix corresponds to the attack history. With this formalism in hand, the key candidate selection may be viewed as the application of some selection rule $\mathcal{R}$ to $\mathcal{M}_s$, returning a subkey candidate $K^{\mathcal{R}}$. The question raised in the preamble of this section may then be rephrased as: "For some rule $\mathcal{R}$, what is the confidence one can have in $K^{\mathcal{R}}$?". To answer this question, we introduce hereafter the notion of *confidence* in $K^{\mathcal{R}}$.

Definition 3.9.1 (Confidence)**.** *For an attack aiming at the recovery of a key k^* and applying a selection rule $\mathcal{R}$ to output a candidate subkey $K^{\mathcal{R}}$, the* confidence *is defined by:*

$$c(K^{\mathcal{R}}) = \frac{\mathbf{P}(K^{\mathcal{R}} = k^*)}{\sum_{k \in \mathcal{K}} \mathbf{P}(K^{\mathcal{R}} = k)}.$$

Remark 3.9.2. The *confidence level* associated to a rule $\mathcal{R}$ merges with the notion of success rate only when the selection rule always outputs a subkey candidate, eg. the rule $\mathcal{R}_0$ defined in the following.

Let us illustrate the application of the confidence level with the comparison of the two following rules, corresponding to the criterion described in the preamble of this section:

- Rule $\mathcal{R}_0$: output the candidate ranked first at the end of the $q - th$ attack.
- Rule $\mathcal{R}_t$: output the candidate ranked first at the end of the $q - th$ attack, only if it was also ranked first for all attacks performed using q_t to N observations.

By definition of $\mathcal{R}_0$, the confidence associated to $\mathcal{R}_0$ satisfies:

$$c(K^{\mathcal{R}_0}) = \frac{\mathbf{P}(R_0(q) = 1)}{\sum_\delta \mathbf{P}(R_\delta(q) = 1)} = \mathbf{P}(R_0(q) = 1),$$

which can be computed thanks to the method presented in the previous sections.

With a similar reasoning, we have:

$$c(K^{\mathcal{R}_t}) = \frac{\mathbf{P}(R_0(q_t) = 1, R_0(q_{t+1}) = 1, \cdots, R_0(q) = 1)}{\sum_\delta \mathbf{P}(R_\delta(q_t) = 1, , \cdots, R_\delta(q) = 1)},$$

whose evaluation requires more development than that of $c(K^{\mathcal{R}_0})$. For such a purpose, the distribution of the ranks vector $(R_\delta(q_t), R_\delta(q_{t+1}), \cdots, R_\delta(q))$ needs to be studied[5]. We thus follow a similar approach as in Section 3.4, and we build the *progressive score vector* $\boldsymbol{d}_{\delta,t}(q) = (\boldsymbol{d}_\delta(q_t)||\boldsymbol{d}_\delta(q_{t+1})|| \cdots ||\boldsymbol{d}_\delta(q))$ where $||$ denotes the vector concatenation operator. This vector can be used to compute the *progressive comparison vector* $\mathbf{c}_{\delta,t}(q)$ similarly as in Section 3.4. We give hereafter the distribution of the progressive score vector.

[5] It is worth noting at this point that the variable $R_\delta(q_i)$ does not verify the Markov property, and that the whole vector has to be studied.

Proposition 3.9.3. *For a CPA exploiting a number q of observations, the progressive score vector $\boldsymbol{d}_{\delta,t}(q)$ follows a multivariate normal distribution with expectation vector $(\mathrm{E}\left[d_{k,t}\right])_{k\in\mathcal{K}}$, a $|\mathcal{K}|(q-q_t)$ vector satisfying:*

$$\mathrm{E}\left[d_{k,t}\right] = \mathrm{E}\left[d_k\right]$$

and covariance matrix $(\mathrm{Cov}\left[d_{k_1,t_i}, d_{k_2,t_j}\right])_{(k_1,k_2)\in\mathcal{K}^2}$ *a* $|\mathcal{K}|(q-q_t)\times|\mathcal{K}|(q-q_t)$ *matrix, satisfying*

$$\mathrm{Cov}\left[d_{k_1,t_i}, d_{k_2,t_j}\right](q) = \left(\frac{q}{max(t_i,t_j)}\mathrm{Cov}\left[d_{k_1}, d_{k_2}\right]\right).$$

Proof. By its construction, the progressive score vector $\boldsymbol{d}_{\delta,t}(q)$ follows a multivariate normal law whose mean vector is trivially deduced from the mean vector of the score vector. To compute the covariance matrix, without loss of generality, one consider $t_j > t_i$. Since we consider an additive distinguisher, the score d_{k_2,t_j} can hence be written $d_{k_2,t_j} = d_{k_2,t_i} + \tilde{d}$, where $\tilde{d}$ is computable from the observations from t_i+1 to t_j. By assumption, these observations are independent from the t_i-th first ones. Consequently, the linearity of covariance ensures the result.

□

This proposition allows for the evaluation of the distribution of $\boldsymbol{d}_{\delta,t}(q)$, and thus for the evaluation of $\mathbf{P}(R_\delta(q_t)=1, R_\delta(q_{t+1})=1,\cdots,R_\delta(q)=1)$ for all hypotheses k_δ. We are then able to compute the confidence $c(K^{\mathcal{R}_t})$.

As an illustration, we study the case where a single intermediate ranking is taken into account, *ie.* we study the probability $\mathbf{P}(R_\delta(\frac{q}{2})=1, R_\delta(q)=1)$, and we plot in Figure 3.15 the obtained confidences.

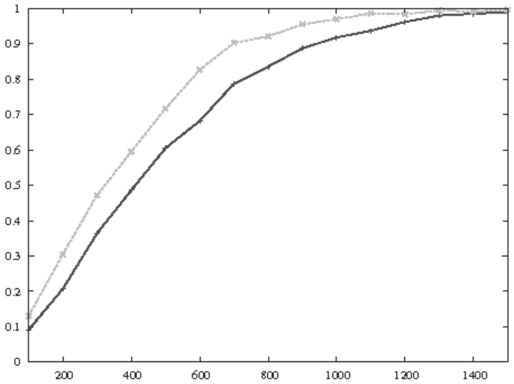


Figure 3.15: Evaluation of confidences in function of the number of measurements for $\mathcal{R}_0$ (plain dark grey), and for $\mathcal{R}_{\frac{q}{2}}$ (dashed light grey), with $\sigma = 10$.

As we can see, at any number of observations, the rule $\mathcal{R}_{\frac{q}{2}}$ actually increases the confidence in the output of an attack compared to the rule $\mathcal{R}_0$.

3.9.2 Discussion and empirical study of convergence and score rules

The accurate evaluation of the confidence level allows a side-channel attacker to assess the success or failure of a partial attack with a known margin of error. For example, and as illustrated in the previous section, applying the selection rule $\mathcal{R}_0$ for a CPA on 800 noisy observations (with Gaussian noise standard deviation equal to 10) leads to an attack failure in 18% of the cases. As a consequence, to reach a 90% confidence level, the attacker has either to perform the attack on more observations (1000 in our example), or to use an other selection rule. Indeed, different selection rules lead to different confidence levels, as they are based on different information. Though a rule based on the whole matrix $\mathcal{M}_s$ may theoretically give better results, the estimation of the confidence level in such a case would be difficult. An interesting open problem is to find an acceptable tradeoff between the computation of the involved probabilities and the accuracy of the obtained confidence.

In this section, we study two rules often observed in practice. The first one exploits the *convergence* of the best hypothesis' rank. To this end, we consider a rule $\mathcal{R}_t^\gamma$ (with $1 \leq \gamma \leq |\mathcal{K}|$) and define it as a slight variation of $\mathcal{R}_t$. The rule $\mathcal{R}_t^\gamma$ returns the best ranked key candidate after the q-th attack only if it was ranked *lower than* γ for the attack on q_t observations. As in previous section, we simulate the simple case where only the ranking obtained with an arbitrary number x of observations is taken into account. We hence experimentally estimate the confidence given by $\mathcal{R}_x^\gamma$ for all γ in Figure 3.16.

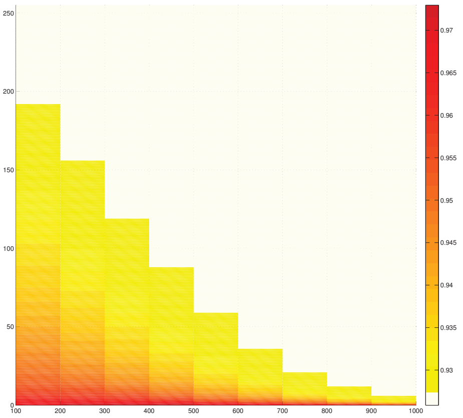


Figure 3.16: Confidence in the key ranked first after a CPA on 1000 observations with $\sigma = 10$, knowing that it was ranked below a given rank γ (in y-axis) on a smaller number of measurements q_t (in x-axis).

For example, when the final best ranked key is ranked lower than 50 using 200 messages, the confidence is around 94% (compared to 92% when using $\mathcal{R}_0$).

Eventually, the analysis conducted in this section shows that though a stabilization of the rank brings a strong confidence, its convergence can also bring some information to an adversary.

The second rule we study considers a transverse approach, by observing the last vector of scores (instead of the rank obtained from intermediate attacks). Namely, we focus on a rule outputting the best ranked candidate when the difference between its score and the score of every other hypothesis is greater than a certain value. This criterion is considered for example in [WO11a]. We simulate this rule, for several bounds, and we plot the results in Figure 3.17. It is of particular interest to note that this rule can bring a huge confidence. Indeed, if the difference using 500 observations is higher than 0.06, then the obtain confidence is around 96% (while 1000 observations would not suffice to attain this level using $\mathcal{R}_0$).

3.9.3 Learning from past attacks

The notion of confidence applied to the convergence rule hints that the results of any *intermediate* attack (that is, an attack using less than the maximum number q of measurements) gives information

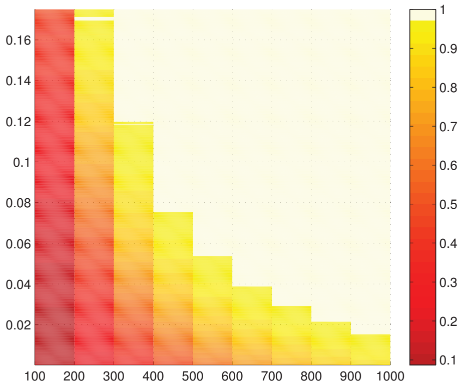


Figure 3.17: Confidence in the best ranked key after a CPA with $\sigma = 10$, on a given number of observations (in x-axis), knowing that its score is higher by a certain value (in y-axis) than every other hypothesis score.

about the key. This observation raises the question of the definition of a new unconditional selection rule taking into account the results of any intermediate attack. This subsection is the fruit of discussions on this topic with Carolyn Whitnall, following the publications of [TPR13; LPR+14].

We study here if the results of the intermediate attacks can help to design a rule that would increase the success rate of the standard rule $\mathcal{R}_0$. To this end, an attacker should use the additional information to *change* its selection, by choosing a most likely key hypothesis than the one he would have chosen by following $\mathcal{R}_0$.

We give in the following a first (negative) partial answer to this question, by studying a specific case. Particularly, we focus on a maximum-likelihood approach on the score vectors.

Score-vector based meta-attack

We showed in Section 3.3 that, for a side-channel attack using an additive distinguisher, the parameters of the distribution of the score vector is computable from the knowledge of the correct key k^*. We argued furthermore that, when k^* is known, we can use this property to accurately evaluate the success rate of the attack. An interesting observation one can make is that, when k^* is unknown, it is possible to compute the parameters for all possible distributions for every $k \in \mathcal{K}$. We propose hereafter to mount an attack based on this observation.

Formally, we mount a maximum-likelihood attack, which we will call a score-vector based meta-attack, comparing an observation against $|\mathcal{K}|$ different distributions. The observed value of our attack is the score vector $\boldsymbol{d_q}$ obtained by performing a side-channel attack with q measurements, while the hypothetic distributions are all the distributions $\mathcal{D}_q^k$ for each $k \in |\mathcal{K}|$, and the distinguisher is the maximum-likelihood. We proved in Section 3.3 that $\mathcal{D}_q^k$ is the multivariate normal law. Hence, denoting by μ_q^k and Σ_q^k respectively its expectation vector and covariance matrix, we define the score returned by this distinguisher as (see Equation 2.1):

$$\mathrm{ML}(\boldsymbol{d_q}, \mathcal{D}_q^k) = \frac{1}{\sqrt{(2\pi)^{|\mathcal{K}|} det(\Sigma_q^k)}} \, e^{-\frac{(\boldsymbol{d_q}-\boldsymbol{\mu_q^k})^\top (\Sigma_q^k)^{-1} (\boldsymbol{d_q}-\boldsymbol{\mu_q^k})}{2}}, \tag{3.38}$$

where $|\mathcal{K}|$ is the size of $\boldsymbol{d_q}$.

For the convenience of our following proofs, we also introduce the following statistical tool $\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k)$, defined as:

$$\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = (\boldsymbol{d_q} - \boldsymbol{\mu_q^k})^\top (\Sigma_q^k)^{-1} (\boldsymbol{d_q} - \boldsymbol{\mu_q^k}). \tag{3.39}$$

Problem description

In order to get a sense of the impact of intermediate attacks on the likelihood of correctness of a key hypothesis, we consider the following setting. Similarly to the setting of Section 3.9, we consider several sets of strictly increasing numbers $q_1, q_2, \cdots q_n$ of measurements. As in [Riv09], we assume a *uniform setting*, that is, that all occurrences ratio (see Equation 3.10) are equal. Note that this requirement is trivially fulfilled if the attacker is in a chosen-plaintext scenario. Then, n side-channel attacks based on the same additive distinguisher are performed, respectively using $q_1, q_2, \cdots q_n$ measurements. These attacks respectively lay n scores vector $\boldsymbol{d_{q_1}}, \boldsymbol{d_{q_2}}, \cdots, \boldsymbol{d_{q_n}}$. We then perform two score-vector based meta-attacks. The first one is performed on $\boldsymbol{d_{q_n}}$. The second one is performed on the concatenation of all scores vector $\boldsymbol{d_\mathbf{q}} = \boldsymbol{d_{q_1}} || \boldsymbol{d_{q_2}} || \cdots \boldsymbol{d_{q_n}}$[6]. (Note that the size of $\boldsymbol{d_q}$, is now $n|\mathcal{K}|$ instead of $|\mathcal{K}|$). Our goal is to compare the two distinguishers of these attacks. We will in fact prove that they are equivalent, thus hinting that the past results are useless for the construction of a good selection rule. Precisely, we will prove the following Theorem:

Theorem 3.9.4. *Let $q_1, q_2, \cdots q_n$ be strictly positive increasing integers. In the uniform setting, when q_1 is high enough, there exist a real numbers $\alpha > 0$ such that, for any hypothesis $k \in \mathcal{K}$, it holds:*

$$ML(\boldsymbol{d_q}, \mathcal{D}_q^k) = \alpha ML(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k). \tag{3.40}$$

Proof of Theorem 3.9.4

The core of the proof for this theorem relies on the following Lemma:

Lemma 3.9.5. *Let $q_1, q_2, \cdots q_n$ be strictly positive increasing integers. In the uniform setting, when q_1 is high enough, there exist a real number γ, such that, for any hypothesis $k \in \mathcal{K}$, it holds:*

$$\widehat{ML}(\boldsymbol{d_q}, \mathcal{D}_q^k) = \widehat{ML}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) + \gamma. \tag{3.41}$$

To be able to prove this, we claim the following Lemma:

Lemma 3.9.6. *In the uniform setting, for any key hypothesis k, and for any number of measurements q, the covariance matrix Σ_q^k is constant.*

Proof. Let k_1, k_2 be two key hypotheses in $\mathcal{K}$. In the uniform setting, it is proven in [Riv09] that the distributions $\mathcal{D}_q^{k_1}$ and $\mathcal{D}_q^{k_2}$ are identical up to a rearrangement of the coordinates (a $\oplus$*-rotation*) of the corresponding score vectors $\boldsymbol{d_q^{k_1}}, \boldsymbol{d_q^{k_2}}$. We denote by $d_i^{k_1}$ (resp. $d_i^{k_2}$) the i-th coordinate of $\boldsymbol{d_q^{k_1}}$ (resp. $\boldsymbol{d_q^{k_2}}$). It follows that, for any two hypotheses k, k' in $\mathcal{K}$, the equality $\mathrm{Cov}(d_{k\oplus k_1}^{k_1}, d_{k'\oplus k_1}^{k_1}) = \mathrm{Cov}(d_{k\oplus k_2}^{k_2}, d_{k'\oplus k_2}^{k_2})$ holds. Hence, the covariance matrices are identical for all considered distributions. □

[6] One can check that this concatenation also trivially follows a multivariate normal law when q_1 is large enough, thanks to the arguments of 3.3.

In the following, we hence omit the index k in the notation of the covariance matrices. Let us now study the distribution $\mathcal{D}_q^k$ of the concatenation vector $\boldsymbol{d_q^k}$ obtained when considering k as the correct hypothesis. By construction, $\mathcal{D}_q^k$ is also a multivariate normal law, and we denote its parameters such that: $\mathcal{D}_q^k = \mathcal{N}(\boldsymbol{\mu_q^k}, \Sigma_q^k)$. The following lemma allows to express those parameters from those of $\mathcal{D}_{q_n}^k$:

Lemma 3.9.7. *In the uniform setting, for any strictly increasing natural numbers $q_1, q_2, \cdots q_n$, the expectation vector $\boldsymbol{\mu_q^k}$ can be expressed as:*

$$\boldsymbol{\mu_q^k} = \boldsymbol{\mu_{q_n}^k} || \boldsymbol{\mu_{q_n}^k} || \boldsymbol{\mu_{q_n}^k} || \cdots || \boldsymbol{\mu_{q_n}^k},$$

and the covariance matrix Σ_q^k is constant for any $k \in \mathcal{K}$ and can be expressed thanks to the covariance matrix Σ_{q_n} as:

$$\Sigma_q^k = q_n \begin{pmatrix} {q_1}^{-1}\Sigma_{q_n} & {q_2}^{-1}\Sigma_{q_n} & {q_3}^{-1}\Sigma_{q_n} & \cdots & q_n^{-1}\Sigma_{q_n} \\ {q_2}^{-1}\Sigma_{q_n} & {q_2}^{-1}\Sigma_{q_n} & {q_3}^{-1}\Sigma_{q_n} & \cdots & q_n^{-1}\Sigma_{q_n} \\ {q_3}^{-1}\Sigma_{q_n} & {q_3}^{-1}\Sigma_{q_n} & {q_3}^{-1}\Sigma_{q_n} & \cdots & q_n^{-1}\Sigma_{q_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ q_n^{-1}\Sigma_{q_n} & q_n^{-1}\Sigma_{q_n} & q_n^{-1}\Sigma_{q_n} & \cdots & q_n^{-1}\Sigma_{q_n} \end{pmatrix}.$$

Proof. Since, for any k, $\boldsymbol{d_q^k} = \boldsymbol{d_{q_1}^k} || \boldsymbol{d_{q_2}^k} || \boldsymbol{d_{q_3}^k} || \cdots || \boldsymbol{d_{q_n}^k}$, the parameters of $\mathcal{D}_q^k$ can be deduced exactly as in Proposition 3.9.3. This, together with Lemma 3.9.6 lays the result. □

Consequently, we will also omit the index k in the notation of this covariance matrix. Let us now study the expression of $\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k)$:

$$\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = (\boldsymbol{d_q} - \boldsymbol{\mu_q^k})^\top (\Sigma_q)^{-1} (\boldsymbol{d_q} - \boldsymbol{\mu_q^k}). \tag{3.42}$$

We see Σ_q^{-1} as a $n \times n$ block matrix and we denote $\Sigma_q^{-1}[i, j]$ its block of coordinate (i, j). Using Lemma 3.9.7, Equation 3.42 can be rewritten as:

$$\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = \sum_{i,j} (\boldsymbol{d_{q_i}} - \boldsymbol{\mu_{q_n}^k})^\top \Sigma_q^{-1}[i, j] \, (\boldsymbol{d_{q_j}} - \boldsymbol{\mu_{q_n}^k}). \tag{3.43}$$

Let us now focus on the inversion of Σ_q. Denoting by $I_{|\mathcal{K}|}$ the $|\mathcal{K}|$-th dimension identity matrix, we deduce from Lemma 3.9.7:

$$\Sigma_q^{-1} = q_n^{-1} \begin{pmatrix} \Sigma_{q_n}^{-1} & 0 & 0 & \cdots & 0 \\ 0 & \Sigma_{q_n}^{-1} & 0 & \cdots & 0 \\ 0 & 0 & \Sigma_{q_n}^{-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \Sigma_{q_n}^{-1} \end{pmatrix} \begin{pmatrix} {q_1}^{-1} I_{|\mathcal{K}|} & {q_2}^{-1} I_{|\mathcal{K}|} & {q_3}^{-1} I_{|\mathcal{K}|} & \cdots & q_n^{-1} I_{|\mathcal{K}|} \\ {q_2}^{-1} I_{|\mathcal{K}|} & {q_2}^{-1} I_{|\mathcal{K}|} & {q_3}^{-1} I_{|\mathcal{K}|} & \cdots & q_n^{-1} I_{|\mathcal{K}|} \\ {q_3}^{-1} I_{|\mathcal{K}|} & {q_3}^{-1} I_{|\mathcal{K}|} & {q_3}^{-1} I_{|\mathcal{K}|} & \cdots & q_n^{-1} I_{|\mathcal{K}|} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ q_n^{-1} I_{|\mathcal{K}|} & q_n^{-1} I_{|\mathcal{K}|} & q_n^{-1} I_{|\mathcal{K}|} & \cdots & q_n^{-1} I_{|\mathcal{K}|} \end{pmatrix}^{-1}, \tag{3.44}$$

hence:

$$\Sigma_q^{-1} = q_n^{-1} \begin{pmatrix} \frac{q_1q_2}{q_2-q_1}\Sigma_{q_n}^{-1} & \frac{q_1q_2}{q_1-q_2}\Sigma_{q_n}^{-1} & 0 & \cdots & 0 \\ \frac{q_1q_2}{q_1-q_2}\Sigma_{q_n}^{-1} & \frac{(q_3-q_1)q_2^2}{(q_1-q_2)(q_2-q_3)}\Sigma_{q_n}^{-1} & \frac{q_2q_3}{q_2-q_3}\Sigma_{q_n}^{-1} & \ddots & 0 \\ 0 & \frac{q_2q_3}{q_2-q_3}\Sigma_{q_n}^{-1} & \frac{(q_4-q_2)q_3^2}{(q_2-q_3)(q_3-q_4)}\Sigma_{q_n}^{-1} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \frac{q_{n-1}q_n}{q_{n-1}-q_n}\Sigma_{q_n}^{-1} & \cdots & \frac{q_n^2}{q_n-q_{n-1}}\Sigma_{q_n}^{-1} \end{pmatrix} \tag{3.45}$$

Note that this matrix actually exists, since for every $i < n$, $q_i \neq q_{i+1}$. Since Σ_q is a covariance matrix, it is positive-definite. Each block $\Sigma_q^{-1}[i,j]$ can hence be written as $\alpha_{i,j}\Sigma_{q_n}^{-1}$, where $\alpha_{i,j}$ is deduced from Equation 3.45:

$$\begin{cases} \alpha_{i,i+1} = & \dfrac{q_iq_{i+1}}{q_n(q_i-q_{i+1})}, \text{for } 1 \leq i < n \\ \alpha_{i+1,i} = & \alpha_{i,i+1}, \text{for } 1 \leq i < n \\ \alpha_{i,i} = & -\alpha_{i,i+1}, \text{for } i = 1 \\ \alpha_{i,i} = & \dfrac{q_i^2(q_{i+1}-q_{i-1})}{q_n(q_{i-1}-q_i)(q_i-q_{i+1})}, \text{for } 1 < i < n \\ \alpha_{i,i} = & \dfrac{q_i^2}{q_n(q_i-q_{i-1})}, \text{for } i = n \\ \alpha_{i,j} = & 0, \text{otherwise} \end{cases} \tag{3.46}$$

Thanks to the symmetry of Σ_q^{-1}, Equation 3.43 becomes:

$$\begin{aligned} \widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = \sum_{i=1}^{n-1} & \Big(\alpha_{i,i}(\boldsymbol{d_{q_i}} - \boldsymbol{\mu^k_{q_n}})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_i}} - \boldsymbol{\mu^k_{q_n}}) + 2\alpha_{i,i+1}(\boldsymbol{d_{q_i}} - \boldsymbol{\mu^k_{q_n}})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_{i+1}}} - \boldsymbol{\mu^k_{q_n}})\Big) \\ & + \alpha_{n,n}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu^k_{q_n}})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu^k_{q_n}}) \end{aligned} \tag{3.47}$$

For any i lower than n, we now denote by Δ_i the difference between the scores vector $\boldsymbol{d_{q_i}}$ and $\boldsymbol{d_{q_n}}$: $\Delta_i = \boldsymbol{d_{q_i}} - \boldsymbol{d_{q_n}}$. Since Σ_{q_n} is symmetrical, then for any i, j lower than n, we have:

$$\begin{aligned} (\boldsymbol{d_{q_i}} - \boldsymbol{\mu^k_{q_n}})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_j}} - \boldsymbol{\mu^k_{q_n}}) \quad = \quad & \Delta_i^\top \Sigma_{q_n}^{-1}\Delta_j + (\boldsymbol{d_{q_n}} - \boldsymbol{\mu^k_{q_n}})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu^k_{q_n}}) \\ & + (\Delta_i + \Delta_j)^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu^k_{q_n}}) \qquad (3.48) \\ & \qquad (3.49) \end{aligned}$$

Hence, Equation 3.47 becomes:

$$\begin{aligned}
\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) \quad = \quad & \sum_{i=1}^{n-1} \alpha_{i,i}(\Delta_i^\top \Sigma_{q_n}^{-1} \Delta_i + \widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) + 2\Delta_i^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu_{q_n}^k})) \\
& +2\sum_{i=1}^{n-1} \alpha_{i,i+1}(\Delta_i^\top \Sigma_{q_n}^{-1} \Delta_{i+1} + \widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) + (\Delta_i + \Delta_{i+1})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu_{q_n}^k})) \\
& +\alpha_{n,n}\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) \qquad (3.50) \\
= \quad & (\alpha_{n,n} + \sum_{i=1}^{n-1}(\alpha_{i,i} + 2\alpha_{i,i+1}))\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) \\
& + \sum_{i=1}^{n-1} \Delta_i^\top \Sigma_{q_n}^{-1}(\alpha_{i,i}\Delta_i + 2\alpha_{i,i+1}\Delta_{i+1}) \\
& +2\sum_{i=1}^{n-1}((\alpha_{i,i} + \alpha_{i,i+1})\Delta_i + \alpha_{i,i+1}\Delta_{i+1})^\top \Sigma_{q_n}^{-1}(\boldsymbol{d_{q_n}} - \boldsymbol{\mu_{q_n}^k})). \qquad (3.51)
\end{aligned}$$

Let us simplify this equation. First, from Equations 3.44 and 3.45 we have:

$$\sum_{i=1}^{n-1}((\alpha_{i,i}+\alpha_{i,i+1})\Delta_i+\alpha_{i,i+1}\Delta_{i+1}) = (\alpha_{1,1}+\alpha_{1,2})\Delta_1+\alpha_{n-1,n}\Delta_n+\sum_{i=2}^{n-1}(\alpha_{i,i}+\alpha_{i,i+1}+\alpha_{i-1,i})\Delta_i. \tag{3.52}$$

Trivially, we have $\Delta_n = 0$, and it can easily be checked that $\alpha_{1,1} + \alpha_{1,2} = 0$. Let us study the last part of the sum from the expression of Σ_q. For any $1 < i < n$, we get:

$$\alpha_{i,i} + \alpha_{i,i+1} + \alpha_{i-1,i} \quad = \quad \frac{q_i^2(q_{i+1} - q_{i-1})}{q_n(q_{i-1} - q_i)(q_i - q_{i+1})} + \frac{q_i q_{i+1}}{q_n(q_i - q_{i+1})} + \frac{q_i q_{i-1}}{q_n(q_{i-1} - q_i)} = 0. \tag{3.53}$$

Consequently, Equation 3.50 becomes:

$$\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = (\alpha_{n,n} + \sum_{i=1}^{n-1}(\alpha_{i,i} + 2\alpha_{i,i+1}))\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) + \sum_{i=1}^{n-1} \Delta_i^\top \Sigma_{q_n}^{-1}(\alpha_{i,i}\Delta_i + 2\alpha_{i,i+1}\Delta_{i+1}). \tag{3.54}$$

Moreover, from Equation 3.53 it holds that $\alpha_{1,2} = -\alpha_{1,1}$ and that $\alpha_{i,i} + \alpha_{i,i+1} = -\alpha_{i-1,i}$ for $1 < i < n$. Therefore, we obtain: $\alpha_{n,n} + \sum_{i=1}^{n-1}(\alpha_{i,i} + 2\alpha_{i,i+1}) = \alpha_{n,n} + \alpha_{n-1,n} = 1$. And thus we get:

$$\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k) = \widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k) + \sum_{i=1}^{n-1} \Delta_i^\top \Sigma_{q_n}^{-1}(\alpha_{i,i}\Delta_i + 2\alpha_{i,i+1}\Delta_{i+1}). \tag{3.55}$$

It is clear that nothing except $\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k)$ depends on k in the right term of this equation. This concludes the proof of Lemma 3.9.5. □

To prove Theorem 3.9.4, we start from Lemma 3.9.5, and use properties of the exponential function. Reusing the notation γ of this lemma, we get for any hypothesis $k \in \mathcal{K}$:

$$e^{-\frac{\widehat{\mathrm{ML}}(\boldsymbol{d_q}, \mathcal{D}_q^k)}{2}} = e^{-\frac{\gamma}{2}} \cdot e^{-\frac{\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}}, \mathcal{D}_{q_n}^k)}{2}}. \tag{3.56}$$

From Lemma 3.9.7 (resp. 3.9.6), we moreover know that, for any hypothesis $k \in \mathcal{K}$, the covariance matrix $\Sigma_{q_n}^k$ (resp Σ_q^k) is constant, and can be denoted Σ_{q_n} (resp. Σ_q). Hence, multiplying Equation 3.56 by a constant, we obtain:

$$\frac{1}{\sqrt{(2\pi)^{n|\mathcal{K}|}det(\Sigma_q)}}e^{-\frac{\widehat{\mathrm{ML}}(\boldsymbol{d_q},\mathcal{D}_q^k)}{2}} = \frac{e^{-\frac{\gamma}{2}}}{\sqrt{(2\pi)^{n|\mathcal{K}|}det(\Sigma_q)}} \cdot e^{-\frac{\widehat{\mathrm{ML}}(\boldsymbol{d_{q_n}},\mathcal{D}_{q_n}^k)}{2}}. \tag{3.57}$$

Finally, this equation lays:

$$\mathrm{ML}(\boldsymbol{d_q},\mathcal{D}_q^k) = e^{-\frac{\gamma}{2}}\sqrt{(2\pi)^{(1-n)|\mathcal{K}|}\frac{det(\Sigma_{q_n})}{det(\Sigma_q)}} \cdot \mathrm{ML}(\boldsymbol{d_{q_n}},\mathcal{D}_{q_n}^k). \tag{3.58}$$

Note that nothing in the right term of this equation depends on k except for $\mathrm{ML}(\boldsymbol{d_{q_n}},\mathcal{D}_{q_n}^k)$. Moreover, since Σ_{q_n} and Σ_q are covariance matrices, their determinant are strictly positive, and hence the square root is well defined and non-zero. Since the exponential function is strictly positive on $\mathbb{R}$, Equation 3.58 finishes the proof of Theorem 3.9.4. □

3.10 Conclusion

In this chapter we have studied the methods for assessing the security of cryptographic implementations against side-channel attacks, both in the context of a known and an unknown key. In the context of a known key, we have presented a methodology to evaluate the success rate of side-channel attacks. We have shown how to apply this methodology in the particular cases of attacks based on the correlation and likelihood distinguishers. The soundness of our approach has been validated by simulations and experiments performed on different microcontrollers. Using this methodology, an evaluator can estimate the side-channel resistance of his masked cryptographic implementation at the cost of inferring a few linear regression coefficients. In the context of an unknown key, we give a rationale for the use of some empirical criteria (such as the convergence of the best hypothesis' rank towards 1) as indicators of the attack success. We hence involve the notion of confidence to allow for the accurate estimation of this success. As an avenue for further research, this work opens the new problem of the exhibition of novel selection rules allowing to efficiently and accurately evaluate the confidence in a side-channel attack while conserving an acceptable success rate.

Chapter 4

Randomness Complexity of Private Circuits for Multiplication

Après son retour de Zurich à Prague, Tomas fut pris de malaise à l'idée
que sa rencontre avec Tereza avait été le résultat de six improbables hasards.
Milan Kundera - *L'insoutenable légèreté de l'être*

Many cryptographic algorithms are vulnerable to side-channel analysis and several leakage models have been introduced to better understand these flaws. In 2003, Ishai, Sahai and Wagner introduced the d-probing security model, in which an attacker can observe at most d intermediate values during a processing. They also proposed an algorithm that securely performs the multiplication of 2 bits in this model, using only $d(d+1)/2$ random bits to protect the computation. We study the randomness complexity of multiplication algorithms secure in the d-probing model. We propose several contributions: we provide new theoretical characterizations and constructions, new practical constructions and a new efficient algorithmic tool to analyze the security of such schemes.

We start with a theoretical treatment of the subject: we propose an algebraic model for multiplication algorithms and exhibit an algebraic characterization of the security in the d-probing model. Using this characterization, we prove a linear (in d) lower bound and a quasi-linear (non-constructive) upper bound for this randomness cost. Then, we construct a new generic algorithm to perform secure multiplication in the d-probing model that only uses $d + d^2/4$ random bits.

From a practical point of view, we consider the important cases $d \leq 4$ that are actually used in current real-life implementations and we build algorithms with a randomness complexity matching our theoretical lower bound for these small-order cases. Finally, still using our algebraic characterization, we provide a new dedicated verification tool, based on information set decoding, which aims at finding attacks on algorithms for fixed order d at a very low computational cost. Part of this work has been published at the *Eurocrypt* conference [BBP+16].

4.1 Introduction

Most commonly used cryptographic algorithms are now considered secure against classical black-box attacks, when the adversary has only knowledge of their inputs or outputs. Today, it is however well known that their implementations are vulnerable to side-channel attacks, as revealed in the academic community by Kocher in 1996 [Koc96]. These attacks exploit the physical emanations of the underlying device such as the execution time, the device temperature, or the power consumption during the algorithm execution.

To thwart side-channel attacks, many countermeasures have been proposed by the community. Among them, the most widely deployed one is probably *masking* (a.k.a. secret/processing sharing) [GP99; CJRR99], which has strong links with techniques usually applied in secure multi-party computation (see e.g., [Yao82; BOGKW88]) or private circuits theory [ISW03]. For many kinds of real-life implementations, this countermeasure indeed demonstrated its effectiveness when combined with noise and processing jittering. The idea of the masking approach is to split every single *sensitive* variable/processing, which depends on the secret and on known variables, into several shares. Each share is generated uniformly at random except the last one which ensures that the combination of all the shares is equal to the initial sensitive value. This technique aims at making the physical leakage of one variable independent of the secret and thus useless for the attacker. The tuple of shares still brings information about the shared data but, in practice, the leakages are noisy and the complexity of extracting useful information increases exponentially with the number of shares, the basis of the exponent being related to the amount of noise [CJRR99].

In order to formally prove the security of masking schemes, the community has made important efforts to define leakage models that accurately capture the leakage complexity and simultaneously enable to build security arguments. In 2003, Ishai *et al.* introduced the *d-probing model* in which the attacker can observe at most d exact intermediate values [ISW03]. This model is very convenient to make security proofs but does not fit the reality of embedded devices which leak noisy functions of all their intermediate variables. In 2013, Prouff and Rivain extended the noisy leakage model [PR13], initially introduced by Chari et al. [CJRR99], to propose a new one more accurate than [ISW03] but not very convenient for security proofs. The two models [ISW03] and [PR13] were later unified by Duc, Dziembowski, and Faust [DDF14] and Duc, Faust, and Standaert [DFS15a] who showed that a security proof in the noisy leakage model can be deduced from security proofs in the d-probing model. This sequence of works shows that proving the security of implementations in the d-probing model makes sense both from a theoretical and practical point of view. An implementation secure in the d-probing model is said to satisfy the *d-privacy property* or equivalently to be *d-private* [ISW03] (or secure at order d).

It is worth noting that there is a tight link between sharing techniques, *Multi Party Computation* (MPC) and also *threshold implementations* [NRS11; BGN+14a; BGN+14b]. In particular, the study in the classical d-probing security model can be seen as a particular case of MPC with honest players. Furthermore, the threshold implementations manipulate sharing techniques with additional restrictions to thwart further hardware attacks resulting from the leakage of electronic glitches. This problem can itself be similarly seen as a particular case of MPC, with Byzantine players [LSP82].

4.1.1 Our Problem

Since most symmetric cryptographic algorithms manipulate Boolean values, the most practical way to protect them is generally to implement *Boolean sharing* (a.k.a. *masking*): namely, each sensitive intermediate result x is shared into several pieces, say $d+1$, which are manipulated by the algorithm and whose parity is equal to x. To secure the processing of a function f on a shared data, one must design a so-called *masking scheme* (or formally a *private circuit*) that describes how to build a sharing of $f(x)$ from that of x while maintaining the d-probing security.

In the context of Boolean sharing, we usually separate the protection of linear functions from that of non-linear ones. In particular, at the hardware level, any circuit can be implemented using only two gates: the linear `XOR` gate and the non-linear `AND` gate. While the protection of linear operations (e.g., XOR) is straightforward since the initial function f can be applied to each share separately, it becomes more difficult for non-linear operations (e.g., AND). In these cases, the shares cannot be manipulated separately and must generally be processed all together to compute the correct result. These values must then be further protected using additional random bits which results in an important timing overhead.

State-of-the-art solutions to implement Boolean sharing on non-linear functions [RP10; CPRR14] have focused on optimizing the computation complexity. Surprisingly, the amount of necessary random bits has only been in the scope of the seminal paper of Ishai, Sahai and Wagner [ISW03]. In this work, the authors proposed and proved a clever construction (further referred to as ISW multiplication) allowing to compute the multiplication of two shared bits by using $d(d+1)/2$ random bits, that is, half as many random bits as the straightforward solution uses. Their construction has since become a cornerstone of secure implementations [RP10; DDF14; DFS15b; RBN+15]. Even if this result is very important, the quantity of randomness remains very expensive to generate in embedded cryptographic implementations. Indeed, such a generation is usually performed using a physical generator followed by a deterministic random bit generator (DRBG). In addition of being a theoretical "chicken-and-egg" problem for this DRBG protection, in practice the physical generator has often a low throughput and the DRBG is also time-consuming. In general, for a DRBG based on a 128-bit block cipher, one call to this block cipher enables to generate 128 pseudo-random bits[1] (see [BK12]). However, one invocation of the standard AES-128 block cipher with the ISW multiplication requires as much as 30,720 random bits (6 random bytes per multiplication, 4 multiplications per S-box [RP10]) to protect the multiplications when masked at the low order $d = 3$, which corresponds to 240 preliminary calls to the DRBG.

4.1.2 Our Contributions

We analyze the quantity of randomness required to define a d-private multiplication algorithm at any order d. Given the sharings $\boldsymbol{a} = (a_i)_{0\leq i\leq d}$, $\boldsymbol{b} = (b_i)_{0\leq i\leq d}$ of two bits a and b, the problem we tackle out is to find the minimal number of random bits necessary to securely compute a sharing $(c_i)_{0\leq i\leq d}$ of the bit $c = ab$ with a d-private algorithm. We limit our scope to the construction of a multiplication based on the sum of shares' products. That is, as in [ISW03], we start with the pairwise products of a's and b's shares and we work on optimizing their sum into $d+1$ shares with as few random bits as possible. We show that this reduces to studying the randomness complexity of some particular d-private compression algorithm that securely transforms the $(d+1)^2$ shares' products into $d+1$ shares of c. In our study we make extensive use of the following theorem that gives an alternative characterization of the d-privacy:

Theorem 4.3.1 ***(informal).*** A compression algorithm is d-private if and only if there does not exist a set of ℓ intermediate results $\{p_1, \dots, p_\ell\}$ such that $\ell \leqslant d$ and $\sum_{i=1}^{\ell} p_i$ can be written as $\boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ with $\boldsymbol{M}$ being some matrix such that the all-ones vector is in the row space or in the column space of $\boldsymbol{M}$.

From this theorem, we deduce the following lower bound on the randomness complexity:

Theorems 4.4.3–4.4.4 ***(informal).*** If $d \geqslant 3$ (resp. $d = 2$), then a d-private compression algorithm for multiplication must involve at least $d+1$ random bits (resp. 2).

This theorem shows that the randomness complexity is in $\Omega(d)$. Following the probabilistic method, we additionally prove the following theorem which claims that there exists a d-private

[1] Actually, the generation of pseudo-random bits roughly corresponds to the execution of a block cipher but we should also consider the regular internal state update.

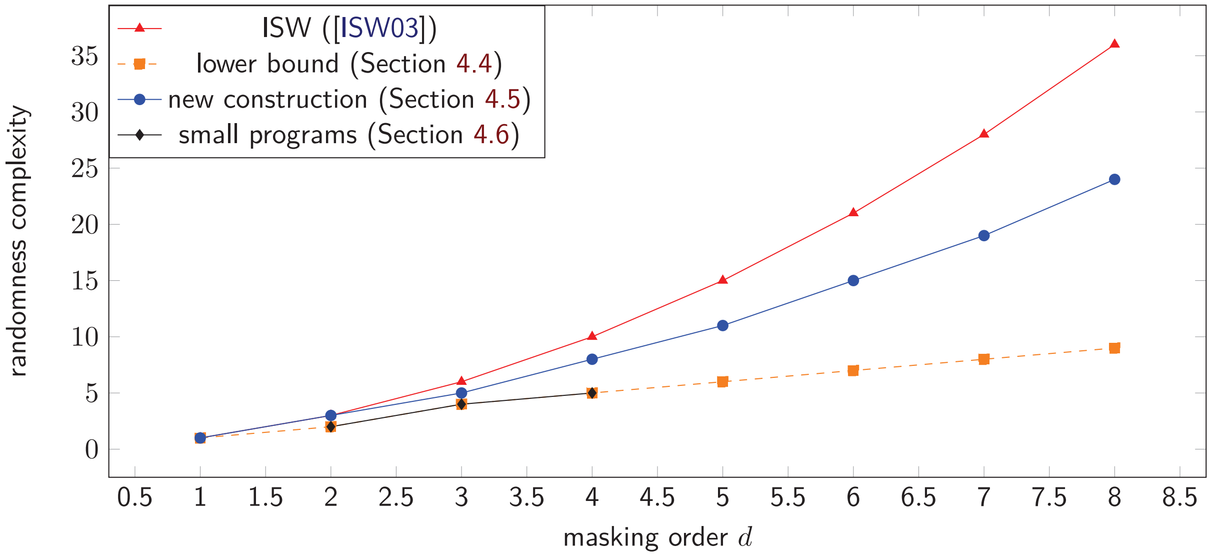


Figure 4.1: Randomness complexity of d-private multiplication algorithms

multiplication algorithm with randomness complexity $O(d \cdot \log d)$. This provides a quasi-linear upper bound $O(d \cdot \log d)$ for the randomness complexity, when $d \to \infty$.

Theorem 4.4.6 *(informal).* There exists a d-private multiplication algorithm with randomness complexity $O(d \cdot \log d)$, when $d \to \infty$.

This upper bound is non-constructive: we show that a randomly chosen multiplication algorithm (in some carefully designed family of multiplication algorithms using $O(d \cdot \log d)$ random bits) is d-private with non-zero probability. This means that there exists one algorithm in this family which is d-private.

In order to explicitly construct private algorithms with low randomness, we analyze the ISW multiplication to bring out necessary and sufficient conditions on the use of the random bits. In particular, we identify necessary chainings and we notice that some random bits may be used several times at several locations to protect more shares' products, while in the ISW multiplication, each random bit is only used twice. From this analysis, we deduce a new d-private multiplication algorithm requiring $\lfloor d^2/4 \rfloor + d$ random bits instead of $d(d+1)/2$. As a positive side-effect, our new construction also reduces the algorithmic complexity of ISW multiplication (i.e., its number of operations).

Based on this generic construction, we then try to optimize some widely used small order instances. In particular, we bring out new multiplication algorithms, for the orders $d = 2$, 3 and 4, which exactly achieve our proven linear lower bound while maintaining the d-privacy. Namely, we present the optimal multiplication algorithms for orders 2, 3 and 4 when summing the shares' products into $d+1$ shares. We formally verify their security using the tool provided in [BBD+15b]. Figure 4.1 illustrates the randomness complexity of our constructions (for general orders d and small orders) and our lower bound.

Note that while the ISW algorithm was initially given for multiplications of bits, it was later extended by Rivain and Prouff in [RP10] for any multiplication in $\mathbb{F}_{2^n}$. In the following, for the sake of simplicity, we refer to binary multiplications ($n = 1$) for our constructions, but note that all of them can also be adapted to multiplication in $\mathbb{F}_{2^n}$.

Contrary to the ISW algorithm, our new constructions are not directly composable — in the sense of Strong Non-Interferent (SNI) in [BBD+15a] — at any order. Fortunately, they can still be used in compositions instead of the ISW algorithms at carefully chosen locations. In this chapter, we thus recall the different security properties related to compositions and we show that in the AES example, our new constructions can replace half the ISW ones while preserving the d-privacy of the whole algorithm.

Finally, while the tool provided in [BBD+15b] — which is based on Easycrypt — is able to reveal potential attack paths and formally prove security in the d-probing model with full confidence, it is limited to the verification of small orders ($d = 6$ in our case). Therefore, we propose a new dedicated probabilistic verification tool, which aims at finding attacks in fixed order private circuits (or equivalently masking schemes) at a very low cost. The tool is developed in Sage (Python) [Sage] and though less generic than [BBD+15b] it is orders of magnitude faster. It relies on some heuristic assumption (i.e. it cannot be used to actually prove the security) but it usually finds attacks very swiftly for any practical order d. It makes use of information set decoding (a technique from coding theory introduced to the cryptographic community for the security analysis of the McEliece cryptosystem in [Pra62; McE78]).

4.2 Preliminaries

This section defines the notations and basic notions that we use in this chapter, but also some elementary constructions we refer to. In particular, we introduce the notion of d-private compression algorithm for multiplication and we present its only concrete instance which was proposed by Ishai, Sahai, and Wagner [ISW03].

4.2.1 Notations

For a set S, we denote by $|S|$ its cardinality, and by $s \overset{\$}{\leftarrow} S$ the operation of picking up an element s of S uniformly at random. We denote by $\mathbb{F}_q$ the finite field with q elements. Vectors are denoted by lower case bold font letters, and matrices are denoted by upper case bold font letters. All vectors are column vectors unless otherwise specified. The *kernel* (resp. the *image*) of the linear map associated to a matrix $\boldsymbol{M}$ is denoted by $\ker(\boldsymbol{M})$ (resp. $\operatorname{im}(\boldsymbol{M})$). For a vector $\boldsymbol{x}$, we denote by x_i its i-th coordinate and by $\mathtt{HW}(\boldsymbol{x})$ its Hamming weight (i.e., the number of its coordinates that are different from 0).

For any fixed $n \geq 1$, let $\boldsymbol{U}_n \in \mathbb{F}_2^{n \times n}$ denote the matrix whose coefficients $u_{i,j}$ equal 1 for all $1 \leq i, j \leq n$. Let $\boldsymbol{0}_{n,\ell} \in \mathbb{F}_2^{n \times \ell}$ denote the matrix whose coefficients are all 0. Let $\boldsymbol{u}_n \in \mathbb{F}_2^n$ denote the vector $(1, \ldots, 1)^\intercal$ and $\boldsymbol{0}_n \in \mathbb{F}_2^n$ denote the vector $(0, \ldots, 0)^\intercal$. For vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ in $\mathbb{F}_2^n$ we denote $\langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t \rangle$ the vector space generated by the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$.

We say that an expression $f(x_1, \ldots, x_n, r)$ functionally depends on the variable r if there exists $a_1, \ldots, a_n$ such that the function $r \mapsto f(a_1, \ldots, a_n, r)$ is not constant.

For an algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x_1, x_2, \ldots)$ the operation of running $\mathcal{A}$ on inputs $(x_1, x_2, \ldots)$ and letting y denote the output. Moreover, if $\mathcal{A}$ is randomized, we denote by $y \overset{\$}{\leftarrow} \mathcal{A}(x_1, x_2, \ldots; r)$ the operation of running $\mathcal{A}$ on inputs $(x_1, x_2, \ldots)$ and with uniform randomness r (or with fresh randomness if r is not specified) and letting y denote the output. The *probability density function* associated to a discrete random variable X defined over S (e.g., $\mathbb{F}_2$) is the function which maps $x \in S$ to $\mathrm{P}[\, X = x \,]$. It is denoted by $\{X\}$ or by $\{X\}_r$ if there is a need to precise the randomness source r over which the *distribution* is considered.

4.2.2 Private Circuits

We examine the privacy property in the setting of Boolean circuits and start with the definition of *circuit* and *randomized circuit* given in [ISW03]. A deterministic circuit C is a directed acyclic graph whose vertices are Boolean gates and whose edges are wires. A *randomized circuit* is a circuit augmented with random-bit gates. A random-bit gate is a gate with fan-in 0 that produces a random bit and sends it along its output wire; the bit is selected uniformly and independently of everything else afresh for each invocation of the circuit. From the two previous notions, we may deduce the following definition of a private circuit inspired from [IKL+13].

Definition 4.2.1. *[IKL+13] A* private circuit *for $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$ is defined by a triple (I, C, O), where*

- *$I\colon \mathbb{F}_2^n \to \mathbb{F}_2^{n'}$ is a randomized circuit with uniform randomness ρ and called input encoder;*
- *C is a randomized boolean circuit with input in $\mathbb{F}_2^{n'}$, output in $\mathbb{F}_2^{m'}$, and uniform randomness $r \in \mathbb{F}_2^t$;*
- *$O\colon \mathbb{F}_2^{m'} \to \mathbb{F}_2^m$ is a circuit, called output decoder.*

*We say that C is a d-*private *implementation of f with encoder I and decoder O if the following requirements hold:*

- Correctness*: for any input $w \in \mathbb{F}_2^n$, $\mathrm{P}\left[\, O(C(I(w;\rho);r)) = f(w)\,\right] = 1$, where the probability is over the randomness ρ and r;*
- Privacy*: for any $w, w' \in \mathbb{F}_2^n$ and any set P of d wires in C, the distributions $\{C_P(I(w;\rho);r)\}_{\rho,r}$ and $\{C_P(I(w';\rho);r)\}_{\rho,r}$ are identical, where $C_P(I(w;\rho);r)$ denotes the list of the d values on the wires from P.*

Remark 4.2.2. It may be noticed that the notions of d-privacy and of security in the d-probing model used, e.g., in [BBD+15b] are perfectly equivalent.

Unless noted otherwise, we assume I and O to be the following *canonical* encoder and decoder: I encodes each bit-coordinate b of its input w by a block $(b_j)_{0\leq j\leq d}$ of $d+1$ random bits with parity b, and O takes the parity of each block of $d+1$ bits. Each block $(b_j)_{0\leq j\leq d}$ is called a *sharing* of b and each b_j is called a *share* of b.

From now on, the wires in a set P used to attack an implementation are referred as the *probes* and the corresponding values in $C_P(I(w;\rho);r)$ as the *intermediate results.* To simplify the descriptions, a probe p is sometimes used to directly denote the corresponding result. A set of probes P such that the distributions $\{C_P(I(w;\rho);r)\}_{\rho,r}$ and $\{C_P(I(w';\rho);r)\}_{\rho,r}$ are *not* identical for some inputs $w, w' \in \mathbb{F}_2^n$ shall be called an *attack*. When the inputs w are clear from the context, the distribution $\{C_P(I(w;\rho);r)\}_{\rho,r}$ is simplified to $\{(p)_{p\in P}\}$.

We now introduce the notions of multiplication algorithm and of d-compression algorithm for multiplication. In this chapter, we deeply study d-private multiplication algorithms and d-private compression algorithms for multiplication.

Definition 4.2.3. *A* multiplication algorithm *is a circuit for the multiplication of* 2 *bits (i.e., with f being the function $f\colon (a,b) \in \mathbb{F}_2^2 \mapsto a \cdot b \in \mathbb{F}_2$), using the canonical encoder and decoder.*

Before moving on to the next notion, let us first introduce a new particular encoder, called *multiplicative*, which has been used in all the previous attempts to build a d-private multiplication algorithm. This encoder takes as input two bits $(a, b) \in \mathbb{F}_2^2$, runs the canonical encoder on these two bits to get $d+1$ random bits $(a_0, \dots, a_d)$ and $(b_0, \dots, b_d)$ with parity a and b respectively, and outputs the $(d+1)^2$ bits $(\alpha_{i,j})_{0 \le i,j \le d}$ with $\alpha_{i,j} = a_i \cdot b_j$. Please note that, in particular, we have $a \cdot b = (\sum_{i=0}^d a_i) \cdot (\sum_{i=0}^d b_i) = \sum_{0 \leqslant i,j \leqslant d} \alpha_{i,j}$.

Definition 4.2.4. *A* d-compression algorithm for multiplication *is a circuit for the multiplication of* 2 *bits (i.e., with* f *being the function* $f\colon (a, b) \in \mathbb{F}_2^2 \mapsto a \cdot b \in \mathbb{F}_2$*), using the canonical decoder and the multiplicative encoder. Moreover, we restrict the circuit* C *to only perform additions in* $\mathbb{F}_2$*.*

When clear from the context, we often omit the parameter d and simply say "a compression algorithm for multiplication".

Remark 4.2.5. Any d-compression algorithm for multiplication yields a multiplication algorithm, as the algorithm can start by computing $\alpha_{i,j}$ given its inputs $(a_0, \dots, a_d, b_0, \dots, b_d)$.

Proposition 4.2.6. *A multiplication algorithm* $\mathcal{B}$ *constructed from a* d*-compression algorithm for multiplication* $\mathcal{A}$ *(as in Remark 4.2.5) is* d*-private if and only if the compression algorithm* $\mathcal{A}$ *is* d*-private.*

Clearly if $\mathcal{B}$ is d-private, so is $\mathcal{A}$. However, the converse is not straightforward, as an adversary can also probe the input shares a_i and b_i in $\mathcal{B}$, while it cannot in $\mathcal{A}$. The full proof of this proposition will be given in Section 4.3.4 and is surprisingly hard: we actually use a stronger version of our algebraic characterization. In the remaining of the chapter, we focus on compression algorithms and we do not need to consider probes on the input shares a_i and b_i, which makes the notation simpler.

In the sequel, a d-compression algorithm for multiplication is denoted by $\mathcal{A}(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{r})$ with $\boldsymbol{r}$ denoting the tuple of uniform random bits used by the algorithm and with $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) denoting the vector of $d+1$ shares of the multiplication operand a (resp. b).

The purpose of the rest of this chapter is to investigate how much randomness is needed for such an algorithm to satisfy the d-privacy and to propose efficient or optimal constructions with respect to the consumption of this resource. The number of bits involved in an algorithm $\mathcal{A}(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{r})$ (i.e., the size of $\boldsymbol{r}$) is called its *randomness complexity* or *randomness cost*.

4.2.3 ISW Algorithm

The first occurrence of a d-private compression circuit for multiplication in the literature is the ISW algorithm, introduced by Ishai, Sahai, and Wagner in [ISW03]. It is described in Algorithm 2. Its randomness cost is $d(d+1)/2$.

To better understand this algorithm, let us first write it explicitly for $d = 3$:

$$\begin{aligned}
c_0 &\leftarrow \alpha_{0,0} + r_{0,1} + r_{0,2} + r_{0,3} \\
c_1 &\leftarrow \alpha_{1,1} + (r_{0,1} + \alpha_{0,1} + \alpha_{1,0}) + r_{1,2} + r_{1,3} \\
c_2 &\leftarrow \alpha_{2,2} + (r_{0,2} + \alpha_{0,2} + \alpha_{2,0}) + (r_{1,2} + \alpha_{1,2} + \alpha_{2,1}) + r_{2,3} \\
c_3 &\leftarrow \alpha_{3,3} + (r_{0,3} + \alpha_{0,3} + \alpha_{3,0}) + (r_{1,3} + \alpha_{1,3} + \alpha_{3,1}) + (r_{2,3} + \alpha_{2,3} + \alpha_{3,2})
\end{aligned}$$

where, for the security to hold, the terms are added from left to right and where the brackets indicate the order in which the operations must be performed (from d-privacy point of view, the addition is

Algorithm 2 ISW algorithm

```
Require: sharing $(\alpha_{i,j})_{0\leq i,j\leq d}$
Ensure: sharing $(c_i)_{0\leq i\leq d}$
  for $i = 0$ to $d$ do
    for $j = i+1$ to $d$ do
      $r_{i,j} \xleftarrow{\$} \mathbb{F}_2$;   $t_{i,j} \leftarrow r_{i,j}$;   $t_{j,i} \leftarrow r_{i,j} + \alpha_{i,j} + \alpha_{j,i}$
  $c_i \leftarrow \alpha_{i,i}$
  for $i = 0$ to $d$ do
    for $j = 0$ to $d$ do
      if $i \neq j$ then
        $c_i \leftarrow c_i + t_{i,j}$
```

not commutative). In particular, when the brackets gather three terms (e.g., $(r_{0,1} + \alpha_{0,1} + \alpha_{1,0})$), the attacker is allowed to probe two values from left to right (e.g., $r_{0,1}+\alpha_{0,1}$ and $(r_{0,1}+\alpha_{0,1}+\alpha_{1,0})$).

Let us now simplify the description by removing all the $+$ symbols, the assignments $c_i \leftarrow$, and defining $\hat{\alpha}_{i,j}$ as $\alpha_{i,j} + \alpha_{j,i}$ if $i \neq j$ and $\alpha_{i,i}$ if $i = j$. The ISW algorithm for $d = 3$ can then be rewritten as:

$$\begin{array}{lllllll}
\hat{\alpha}_{0,0} & r_{0,1} & & r_{0,2} & & r_{0,3} & \\
\hat{\alpha}_{1,1} & (r_{0,1} & \hat{\alpha}_{0,1}) & r_{1,2} & & r_{1,3} & \\
\hat{\alpha}_{2,2} & (r_{0,2} & \hat{\alpha}_{0,2}) & (r_{1,2} & \hat{\alpha}_{1,2}) & r_{2,3} & \\
\hat{\alpha}_{3,3} & (r_{0,3} & \hat{\alpha}_{0,3}) & (r_{1,3} & \hat{\alpha}_{1,3}) & (r_{2,3} & \hat{\alpha}_{2,3}).
\end{array}$$

Please note that the expression of $\hat{\alpha}_{i,j}$ with $i \neq j$ (*i.e.* $\alpha_{i,j} + \alpha_{j,i}$) is expanded before the actual evaluation, i.e., as in the previous representation, the sum $\alpha_{i,j} + \alpha_{j,i}$ is not evaluated beforehand but evaluated during the processing of $r_{i,j} + \hat{\alpha}_{i,j} = r_{i,j} + \alpha_{i,j} + \alpha_{j,i}$.

4.3 Algebraic Characterization

In order to reason about the required quantity of randomness in d-private compression algorithms for multiplication, we define an algebraic condition on the security and we prove that an algorithm is d-private if and only if there is no set of probes which satisfies it.

4.3.1 Matrix Notation

As our condition is algebraic, it is practical to introduce some matrix notation for our probes. We write $\boldsymbol{a} = (a_0, \ldots, a_d)^\intercal$ and $\boldsymbol{b} = (b_0, \ldots, b_d)^\intercal$ the vectors corresponding to the shares of the inputs a and b respectively. We also denote by $\boldsymbol{r} = (r_1, \ldots, r_R)^\intercal$ the vector of the random bits.

We remark that, for any probe p on a compression algorithm for multiplication, p is always an expression that can be written as a sum of $\alpha_{i,j}$'s (with $\alpha_{i,j} = a_i \cdot b_j$) and r_k's, and possibly a constant $c_p \in \mathbb{F}_2$. In other words, we can write p as

$$p = \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M_p} \cdot \boldsymbol{b} + \boldsymbol{s_p}^{\intercal} \cdot \boldsymbol{r} + c_p,$$

with $\boldsymbol{M_p}$ being a matrix in $\mathbb{F}_2^{(d+1)\times(d+1)}$ and $\boldsymbol{s_p}$ being a vector in $\mathbb{F}_2^R$. This matrix $\boldsymbol{M_p}$ and this vector $\boldsymbol{s_p}$ are uniquely defined. In addition, any sum of probes can also be written that way.

Furthermore, if $c_p = 1$, we can always sum the probe with 1 and consider $p+1$ instead of p. This does not change anything on the probability distribution we consider. Therefore, for the sake of simplicity, we always assume $c_p = 0$ in all the chapter.

4.3.2 Algebraic Condition

We now introduce our algebraic condition:

Condition 1. *A set of probes $P = \{p_1, \dots, p_\ell\}$ on a d-compression algorithm for multiplication satisfies Condition 1 if and only if the expression $f = \sum_{i=1}^{\ell} p_i$ can be written as $f = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ with $\boldsymbol{M}$ being some matrix such that $\boldsymbol{u}_{d+1}$ is in the row space or the column space of $\boldsymbol{M}$.*

As seen previously, the expression f can always be written as

$$f = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{s}^\intercal \cdot \boldsymbol{r},$$

for some matrix $\boldsymbol{M}$ and some vector $\boldsymbol{s}$. Therefore, what the condition enforces is that $\boldsymbol{s} = \boldsymbol{0}_R$ (or in other words, f does not functionally depend on any random bit) and the column space or the row space of $\boldsymbol{M}$ contains the vector $\boldsymbol{u}_{d+1}$.

A Weaker Condition. To better understand Condition 1, let us introduce a weaker condition which is often easier to deal with:

Condition 2 (weak condition)**.** *A set of probes $P = \{p_1, \dots, p_\ell\}$ on a d-compression algorithm for multiplication satisfies Condition 2 if and only if the expression $f = \sum_{i=1}^{\ell} p_i$ does not functionally depend on any r_k and there exists a map $\gamma\colon \{0, \dots, d\} \to \{0, \dots, d\}$ such that f does functionally depend on every $(\alpha_{i,\gamma(i)})_{0 \le i \le d}$ or on every $(\alpha_{\gamma(i),i})_{0 \le i \le d}$.*

This condition could be reformulated as: $f = \sum_{i=1}^{\ell} p_i$ functionally depends on either all the a_i's or all the b_i's and does not functionally depend on any r_k. It is easy to see that any set P verifying Condition 1 also verifies Condition 2.

4.3.3 Algebraic Characterization

Theorem 4.3.1. *Let $\mathcal{A}$ be a d-compression algorithm for multiplication. Then, $\mathcal{A}$ is d-private if and only if there does not exist a set $P = \{p_1, \dots, p_\ell\}$ of $\ell \le d$ probes that satisfies Condition 1. Furthermore any set $P = \{p_1, \dots, p_\ell\}$ satisfying Condition 1 is an attack.*

Please note that Theorem 4.3.1 would not be valid with Condition 2 (instead of Condition 1). Let us indeed consider the following 2-compression scheme for multiplication:

$$\begin{array}{ccccc}
\alpha_{2,0} & r_1 & \alpha_{0,0} & r_2 & \alpha_{0,2} \\
\alpha_{2,1} & r_1 & \alpha_{1,1} & r_3 & \alpha_{1,2} \\
\alpha_{1,0} & r_2 & \alpha_{0,1} & r_3 & \alpha_{2,2}
\end{array}$$

It is easy to see that the only set of probes satisfying Condition 2 is

$$P = \{\alpha_{2,0} + r_1 + \alpha_{0,0}, \quad \alpha_{2,1} + r_1 + \alpha_{1,1}\} \enspace .$$

However, this set does not satisfy Condition 1, and in fact, this compression algorithm is 2-private.

Proof. **Direction 1: Left to right.** We prove hereafter that if $\mathcal{A}$ is d-private, then there does not exist a set $P = \{p_1, \dots, p_\ell\}$ of $\ell \le d$ probes that satisfies Condition 1.

By contrapositive, let us assume that there exists a set $P = \{p_1, \dots, p_\ell\}$ of at most d probes that satisfies Condition 1. Let $\boldsymbol{M}$ be the matrix such that $f = \sum_{i=1}^{\ell} p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ and let us assume, without loss of generality, that $\boldsymbol{u}_{d+1}$ is in the vector subspace generated by the columns of $\boldsymbol{M}$. We remark that, for any $\boldsymbol{v} \in \mathbb{F}_2^{d+1}$:

$$\mathrm{P}\left[\, \boldsymbol{a}^{\intercal} \cdot \boldsymbol{v} = a \,\right] = \begin{cases} 1 & \text{when } \boldsymbol{v} = \boldsymbol{u}_{d+1} \\ \frac{1}{2} & \text{when } \boldsymbol{v} \neq \boldsymbol{u}_{d+1} \end{cases}$$

by definition of the sharing $\boldsymbol{a}$ of a (probability is taken over $\boldsymbol{a}$). Thus we have, when $a = 0$ (assuming that b is uniformly random)

$$\begin{aligned}
&\mathrm{P}\left[\, f = 0 \mid a = 0 \,\right] \\
&= \mathrm{P}\left[\, \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M} \cdot \boldsymbol{b} = 0 \mid \boldsymbol{a}^{\intercal} \cdot \boldsymbol{u}_{d+1} = 0 \,\right] \\
&= \mathrm{P}\left[\, \boldsymbol{a}^{\intercal} \cdot \boldsymbol{u}_{d+1} = 0 \mid a = 0 \text{ and } \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right] \cdot \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right] \\
&\quad + \textstyle\sum_{\boldsymbol{v} \in \mathbb{F}_2^{d+1} \setminus \{\boldsymbol{u}_{d+1}\}} \mathrm{P}\left[\, \boldsymbol{a}^{\intercal} \cdot \boldsymbol{v} = 0 \mid a = 0 \text{ and } \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{v} \,\right] \cdot \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{v} \,\right] \\
&= 1 \cdot \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right] + \textstyle\sum_{\boldsymbol{v} \in \mathbb{F}_2^{d+1} \setminus \{\boldsymbol{u}_{d+1}\}} \frac{1}{2} \cdot \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{v} \,\right] \\
&= 1 \cdot \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right] + \tfrac{1}{2}(1 - \mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right]) \\
&= \tfrac{1}{2} + \tfrac{1}{2}\mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right].
\end{aligned}$$

Similarly, when $a = 1$, we have

$$\mathrm{P}\left[\, f = 0 \mid a = 1 \,\right] = \tfrac{1}{2} - \tfrac{1}{2}\mathrm{P}\left[\, \boldsymbol{M} \cdot \boldsymbol{b} = \boldsymbol{u}_{d+1} \,\right].$$

As $\boldsymbol{u}_{d+1}$ is in the column space of $\boldsymbol{M}$, the distribution of $\{f\}$ is not the same when $a = 0$ and when $a = 1$. This implies that the distribution $\{(p_1, \ldots, p_\ell)\}$ is also different when $a = 0$ and $a = 1$. Hence $\mathcal{A}$ is not d-private.

This concludes the proof of the first implication and the fact that any set $P = \{p_1, \ldots, p_\ell\}$ satisfying Condition 1 is an attack.

Direction 2: Right to left. Let us now prove by contradiction that if there does not exist a set $P = \{p_1, \ldots, p_\ell\}$ of $\ell \leq d$ probes that satisfies Condition 1, then $\mathcal{A}$ is d-private.

Let us assume that $\mathcal{A}$ is not d-private. Then there exists an attack using a set of probes $P = \{p_1, \ldots, p_\ell\}$ with $\ell \leq d$. This is equivalent to say that there exists two inputs $(a^{(0)}, b^{(0)}) \neq (a^{(1)}, b^{(1)})$ such that the distribution $\{(p_1, \ldots, p_\ell)\}$ is not the same whether $(a, b) = (a^{(0)}, b^{(0)})$ or $(a, b) = (a^{(1)}, b^{(1)})$.

We first remark that we can consider $0 = a^{(0)} \neq a^{(1)} = 1$, without loss of generality as the $a^{(i)}$'s and the $b^{(i)}$'s play a symmetric role (and $(a^{(0)}, b^{(0)}) \neq (a^{(1)}, b^{(1)})$). Furthermore, we can always choose $b^{(0)} = b^{(1)}$, as if the distribution $\{(p_1, \ldots, p_\ell)\}$ is not the same whether $(a, b) = (0, b^{(0)})$ or $(a, b) = (1, b^{(1)})$, with $b^{(0)} \neq b^{(1)}$, then:

- it is not the same whether $(a, b) = (0, b^{(0)})$ or $(a, b) = (1, b^{(0)})$ (in which case, we could have taken $b^{(1)} = b^{(0)}$), or
- it is not the same whether $(a, b) = (1, b^{(0)})$ or $(a, b) = (1, b^{(1)})$ (in which case, we can just exchange the a's and the b's roles).

To summarize, there exists $b^{(0)}$ such that the distribution $\{(p_1, \ldots, p_\ell)\}$ is not the same whether $(a, b) = (0, b^{(0)})$ or $(a, b) = (1, b^{(0)})$.

In the sequel $b^{(0)}$ is fixed and we call a tuple $(p_1, \ldots, p_\ell)$ satisfying the previous property an *attack tuple*.

We now remark that if $\ell = 1$ or if even the distribution $\{(\sum_{i=1}^{\ell} p_i)\}$ is not the same whether $(a,b) = (0, b^{(0)})$ or $(a,b) = (1, b^{(0)})$ (i.e., $(\sum_{i=1}^{\ell} p_i)$ is an attack tuple), then it follows easily from the probability analysis of the previous proof for the other direction of the theorem, that the set P satisfies Condition 1. The main difficulty is that it is not necessarily the case that $\ell = 1$ or $(\sum_{i=1}^{\ell} p_i)$ is an attack tuple. To overcome it, we use linear algebra.

But first, let us introduce some useful notations and lemmas. We write $\boldsymbol{p}$ the vector $(p_1, \ldots, p_\ell)^\intercal$ and we say that $\boldsymbol{p}$ is an *attack vector* if and only if $(p_1, \ldots, p_\ell)$ is an attack tuple. Elements of $\boldsymbol{p}$ are polynomials in the a_i's, the b_j's and the r_k's.

Lemma 4.3.2. *If $\boldsymbol{p}$ is an attack vector and $\boldsymbol{N}$ is an invertible matrix in $\mathbb{F}_2^{\ell \times \ell}$, then $\boldsymbol{N} \cdot \boldsymbol{p}$ is an attack vector.*

Proof. This is immediate from the fact that $\boldsymbol{N}$ is invertible. Indeed, as a matrix over $\mathbb{F}_2$, $\boldsymbol{N}^{-1}$ is also a matrix over $\mathbb{F}_2$. Hence, multiplying the set of probes $\{\boldsymbol{N} \cdot \boldsymbol{p}\}$ by $\boldsymbol{N}^{-1}$ (which leads to the first set of probes $\{\boldsymbol{p}\}$) can be done by simply computing sums of elements in $\{\boldsymbol{N} \cdot \boldsymbol{p}\}$. Hence, as the distribution of $\{\boldsymbol{p}\}$ differs when $(a,b) = (0, b^{(0)})$ and $(a,b) = (1, b^{(0)})$, the same is true for the distribution $\{\boldsymbol{N} \cdot \boldsymbol{p}\}$. □

We also use the following straightforward lemma.

Lemma 4.3.3. *If $(p_1, \ldots, p_\ell)$ is an attack tuple such that the $\ell - t + 1$ random variables $(p_1, \ldots, p_t)$, p_{t+1}, $\ldots$, and p_ℓ are mutually independent, and the distributions of $(p_{t+1}, \ldots, p_\ell)$ is the same for all the values of the inputs (a,b), then $(p_1, \ldots, p_t)$ is an attack tuple.*

Let us consider the matrix $\boldsymbol{S} \in \mathbb{F}_2^{\ell \times R}$ whose coefficients $s_{i,j}$ are defined as $s_{i,j} = 1$ if and only if the expression p_i functionally depends on r_j. In other words, if we write $p_i = \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M_{p_i}} \cdot \boldsymbol{b} + \boldsymbol{s}_{\boldsymbol{p_i}}^{\intercal} \cdot \boldsymbol{r}$, the i-th row of $\boldsymbol{S}$ is $\boldsymbol{s}_{\boldsymbol{p_i}}^{\intercal}$. We can permute the random bits (i.e., the columns of $\boldsymbol{S}$ and the rows of $\boldsymbol{r}$) such that a row reduction on the matrix $\boldsymbol{S}$ yields a matrix of the form:

$$\boldsymbol{S'} = \begin{pmatrix} \boldsymbol{0}_{t,t} & \boldsymbol{0}_{t,\ell-t} \\ \boldsymbol{I}_t & \boldsymbol{S''} \end{pmatrix}.$$

Let $\boldsymbol{N}$ be the invertible matrix in $\mathbb{F}_2^{\ell \times \ell}$ such that $\boldsymbol{N} \cdot \boldsymbol{S} = \boldsymbol{S'}$. And we write $\boldsymbol{p'} = (p'_1, \ldots, p'_\ell)^\intercal = \boldsymbol{N} \cdot \boldsymbol{p}$. Then, $\boldsymbol{p'}$ is also an attack vector according to Lemma 4.3.2. In addition, for $t < i \leq \ell$, p'_i does functionally depend on r_i and no other p'_j does functionally depend on r_j (due to the shape of $\boldsymbol{S'}$). Therefore, according to Lemma 4.3.3, $(p'_1, \ldots, p'_t)$ is an attack tuple.

We remark that $(p'_1, \ldots, p'_t)$ does not functionally depend on any random bit, due to the shape of $\boldsymbol{S'}$. Therefore, for each $1 \leq i \leq t$, we can write:

$$p'_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M'_i} \cdot \boldsymbol{b},$$

for some matrix $\boldsymbol{M'_i}$.

We now need a final lemma to be able to conclude.

Lemma 4.3.4. *If $(p'_1, \ldots, p'_t)$ is an attack tuple, then there exists a vector $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$ such that $\boldsymbol{u}_{d+1}$ is in the vector space $\langle \boldsymbol{M'_1} \cdot \boldsymbol{b}^*, \ldots, \boldsymbol{M'_t} \cdot \boldsymbol{b}^* \rangle$.*

Proof. This lemma can be seen as a generalization of the probability analysis in the proof of the first direction of the theorem.

We suppose by contradiction that $(p'_1, \dots, p'_t)$ is an attack vector but there does not exist a vector $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$ such that $\boldsymbol{u}_{d+1}$ is in the vector space $\langle \boldsymbol{M'_1} \cdot \boldsymbol{b}^*, \dots, \boldsymbol{M'_t} \cdot \boldsymbol{b}^* \rangle$. Then, for any value $a^{(0)}$, any vector $\boldsymbol{b^{(0)}} \in \mathbb{F}_2^{d+1}$, and any vector $\boldsymbol{x} = (x_1, \dots, x_t)^\intercal \in \mathbb{F}_2^t$:

$$\begin{aligned}
&\mathrm{P}\left[(p'_1, \dots, p'_t) = (x_1, \dots, x_t) \;\middle|\; a = a^{(0)} \text{ and } \boldsymbol{b} = \boldsymbol{b^{(0)}} \right] \\
&= \mathrm{P}\left[(\boldsymbol{a}^{\intercal} \cdot \boldsymbol{M'_1} \cdot \boldsymbol{b^{(0)}}, \dots, \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M'_t} \cdot \boldsymbol{b^{(0)}}) = (x_1, \dots, x_t) \;\middle|\; \boldsymbol{a}^\intercal \cdot \boldsymbol{u}_{d+1} = a^{(0)} \right] \\
&= \mathrm{P}\left[\boldsymbol{a}^{\intercal} \cdot \boldsymbol{B} = \boldsymbol{x}^{\intercal} \;\middle|\; \boldsymbol{a}^{\intercal} \cdot \boldsymbol{u}_{d+1} = a^{(0)} \right],
\end{aligned}$$

where $\boldsymbol{B}$ is the matrix whose i-th column is the vector $\boldsymbol{M'_i} \cdot \boldsymbol{b^{(0)}}$. To conclude, we just need to remark that

$$\mathrm{P}\left[\boldsymbol{a}^{\intercal} \cdot \boldsymbol{B} = \boldsymbol{x}^{\intercal} \mid \boldsymbol{a}^{\intercal} \cdot \boldsymbol{u}_{d+1} = 0 \right] = \mathrm{P}\left[\boldsymbol{a}^{\intercal} \cdot \boldsymbol{B} = \boldsymbol{x}^{\intercal} \mid \boldsymbol{a}^{\intercal} \cdot \boldsymbol{u}_{d+1} = 1 \right],$$

which implies that the probability distribution of $(p'_1, \dots, p'_t)$ is independent of the value of a, which contradicts the fact the $(p'_1, \dots, p'_t)$ is an attack tuple.

To prove the previous equality, we use the fact that $\boldsymbol{u}_{d+1}$ is not in the column space of $\boldsymbol{B}$ and therefore the value of $\boldsymbol{a}^\intercal \cdot \boldsymbol{u}_{d+1}$ is uniform and independent of the value of $\boldsymbol{a}^\intercal \cdot \boldsymbol{B}$ (when $\boldsymbol{a}$ is a uniform vector in $\mathbb{F}_2^{d+1}$). □

Thanks to Lemma 4.3.4, there exists a vector $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_t)^\intercal \in \mathbb{F}_2^t$ and a vector $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$ such that

$$\left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{M'_i} \right) \cdot \boldsymbol{b}^* = \boldsymbol{u}_{d+1} \quad . \tag{4.1}$$

Let $\boldsymbol{\sigma'}$ be the vector in $\mathbb{F}_2^\ell$ defined by $\boldsymbol{\sigma'}^\intercal = \begin{pmatrix} \boldsymbol{\sigma}^\intercal & \boldsymbol{0}_{\ell-t}^\intercal \end{pmatrix} \cdot \boldsymbol{N}$. We have:

$$\boldsymbol{\sigma'}^\intercal \cdot \boldsymbol{p} = \sum_{i=1}^{t} \sigma_i \cdot p'_i = \sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{a}^\intercal \cdot \boldsymbol{M'_i} \cdot \boldsymbol{b} = \boldsymbol{a}^\intercal \cdot \left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{M'_i} \right) \cdot \boldsymbol{b} \quad . \tag{4.2}$$

Therefore, we can define the set $P' = \{p_i \mid \sigma_i = 1\}$. This set satisfies Condition 1, according to Equations (4.1) and (4.2).

This concludes the proof. □

4.3.4 Approach extension

We now propose to prove Proposition 4.2.6 by extending our algebraic condition. As explained in Section 4.2.2, we just need to prove that we can transform an attack set $P = \{p_1, \dots, p_\ell\}$ of size $\ell \leq d$ for the multiplication algorithm $\mathcal{B}$ into an attack set P' for the compression algorithm $\mathcal{A}$. The only difference is that some probes in P may be the inputs a_i or b_i, while such probes are forbidden in P'.

This is actually surprisingly very hard.

Extended Matrix Notation

We now write probes p as:

$$p = \boldsymbol{a}^\intercal \cdot \boldsymbol{M_p} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{m_{p,a}} + \boldsymbol{m_{p,b}^\intercal} \cdot \boldsymbol{b} + \boldsymbol{s_p^\intercal} \cdot \boldsymbol{r} + c_p,$$

where $\boldsymbol{M_p}$ is a matrix in $\mathbb{F}_2^{(d+1)\times(d+1)}$, $\boldsymbol{m_{p,a}}$ and $\boldsymbol{m_{m,b}}$ are two vectors in $\mathbb{F}_2^{d+1}$, and $c_p \in \mathbb{F}_2$ is a constant which is supposed to be zero in the sequel (as in Section 4.3). This notation can be extended to the sum of probes.

Notice that actually, for all the probes we consider: at most one of the matrices or vectors $\boldsymbol{M_p}$, $\boldsymbol{m_{p,a}}$, and $\boldsymbol{m_{p,b}}$ is non-zero. Furthermore the Hamming weight of $\boldsymbol{m_{p,a}}$, and $\boldsymbol{m_{p,b}}$ is at most 1. However, it is easier to deal with this slight generalization.

Extended Algebraic Condition

We now introduce our extended algebraic condition:

Condition 3. *A set of probes $P = \{p_1, \dots, p_\ell\}$ on a multiplication algorithm satisfies Condition 1 if and only if the expression $f = \sum_{i=1}^{\ell} p_i$ can be written as $f = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{m_a} + \boldsymbol{m_b}^\intercal \cdot \boldsymbol{b}$ with $\boldsymbol{M}$ being some matrix and $\boldsymbol{m_a}$ and $\boldsymbol{m_b}$ being some vectors such that $\boldsymbol{u}_{d+1}$ is in the affine space $\boldsymbol{m_a} + \operatorname{im}(\boldsymbol{M})$ or $\boldsymbol{m_b} + \operatorname{im}(\boldsymbol{M}^\intercal)$, where $\operatorname{im}(\boldsymbol{M})$ is the column space of $\boldsymbol{M}$ and $\operatorname{im}(\boldsymbol{M}^\intercal)$ is the row space of $\boldsymbol{M}$.*

Extended Algebraic Characterization

Theorem 4.3.5. *Let $\mathcal{B}$ be a multiplication algorithm constructed from a d-compression algorithm for multiplication as in Remark 4.2.5. Then, $\mathcal{B}$ is d-private if and only if there does not exist a set $P = \{p_1, \dots, p_\ell\}$ of $\ell \le d$ probes that satisfies Condition 3. Furthermore any set $P = \{p_1, \dots, p_\ell\}$ satisfying Condition 1 is an attack.*

Proof. The left-to-right direction can be proven similarly as for Theorem 4.3.1. Let us focus on the right-to-left direction.

The proof is exactly the same up to the definition of the p'_i (since what comes before that only considers the random bits in the probes and hence is similar when probes of the form a_i or b_i are taken into account), which can now be written as:

$$p'_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M'_i} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{m}'_{i,a} + \boldsymbol{m}'^{\intercal}_{i,b} \cdot \boldsymbol{b},$$

for some matrix $\boldsymbol{M'_i}$ and vectors $\boldsymbol{m'_{i,a}}$ and $\boldsymbol{m'_{i,b}}$.

We now can conclude using the following lemma, which is an extended version of Lemma 4.3.4 and whose proof is similar:

Lemma 4.3.6. *If $(p'_1, \dots, p'_t)$ is an attack tuple, then there exists a vector $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$ such that $\boldsymbol{u}_{d+1}$ is in the vector space $\langle \boldsymbol{M'_1} \cdot \boldsymbol{b}^* + \boldsymbol{m'_{1,a}}, \dots, \boldsymbol{M'_t} \cdot \boldsymbol{b}^* + \boldsymbol{m'_{t,a}} \rangle$.*

Thanks to Lemma 4.3.6, there exists a vector $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_t)^\intercal \in \mathbb{F}_2^t$ and a vector $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$, such that

$$\left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{M'_i}\right) \cdot \boldsymbol{b}^* + \left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{m'_{i,b}}\right) = \boldsymbol{u}_{d+1} \quad . \tag{4.3}$$

Let $\boldsymbol{\sigma'}$ be the vector in $\mathbb{F}_2^\ell$ defined by $\boldsymbol{\sigma'}^\intercal = \begin{pmatrix} \boldsymbol{\sigma}^\intercal & \boldsymbol{0}_{\ell-t}^\intercal \end{pmatrix} \cdot \boldsymbol{N}$. We have:

$$\boldsymbol{\sigma'}^\intercal \cdot \boldsymbol{p} = \sum_{i=1}^{t} \sigma_i \cdot p'_i = \sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{a}^\intercal \cdot \boldsymbol{M'_i} \cdot \boldsymbol{b} + \sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{a}^\intercal \cdot \boldsymbol{m}'_{i,a} + \sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{m}'^{\intercal}_{i,b} \cdot \boldsymbol{b}$$

so that:

$$\boldsymbol{\sigma}'^{\intercal} \cdot \boldsymbol{p} = \boldsymbol{a}^{\intercal} \cdot \left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{M}'_i \right) \cdot \boldsymbol{b} + \boldsymbol{a}^{\intercal} \cdot \left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{m}'_{i,a} \right) + \left(\sum_{i=1}^{t} \sigma_i \cdot \boldsymbol{m}'^{\intercal}_{i,b} \right) \cdot \boldsymbol{b} \ . \tag{4.4}$$

Therefore, we can define the set $P' = \{p_i \mid \sigma_i = 1\}$. This set satisfies Condition 3, according to Equations (4.3) and (4.4).

This concludes the proof.

□

Proof of Proposition 4.2.6

We can now prove Proposition 4.2.6.

Proof. Let us suppose by contraposition that $\mathcal{B}$ is not d-private. Thanks to Theorem 4.3.5, this means that there exists a set of probes $P = \{p_1, \ldots, p_\ell\}$ satisfying Condition 3. We suppose without loss of generality that:

$$\sum_{i=1}^{\ell} p_i = \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{a}^{\intercal} \cdot \boldsymbol{m_a} + \boldsymbol{m}_{\boldsymbol{b}}^{\intercal} \cdot \boldsymbol{b}$$

and $\boldsymbol{u}_{d+1}$ is in the affine space $\boldsymbol{m_a} + \mathrm{im}(\boldsymbol{M})$.

From the shapes of possibles probes, we know that for any i, if $m_{a,i} = 1$, then one of the probe p_j has to be a_i. Let us spit our set of probes P in three sets:

- P_1 contains all the probes p_j such that $p_j = a_i$ and $m_{a,i} = 1$;
- P_2 contains all the probes which are not of the form a_i or b_j;
- P_3 contains all the other probes.

We remark that:

$$\sum_{p \in P_1 \cup P_2} p = \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{a}^{\intercal} \cdot \boldsymbol{m_a}.$$

Let $\boldsymbol{b}^* \in \mathbb{F}_2^{d+1}$ be a vector such that $\boldsymbol{u}_{d+1} = \boldsymbol{m_a} + \boldsymbol{B} \cdot \boldsymbol{b}^*$. The Hamming weight of $\boldsymbol{m_a}$ is exactly the cardinality of P_1 and is at most d. Therefore, $\boldsymbol{b}^* \neq \boldsymbol{0}_{d+1}$ and we can arbitrarily choose $0 \leq j^* \leq d$ such that $b^*_{j^*} = 1$. Finally, let us set $P'_1 = \{\alpha_{i,j^*} \mid m_{a,i} = 1\}$. We can write

$$\sum_{p \in P'_1 \cup P_2} p = \boldsymbol{a}^{\intercal} \cdot \boldsymbol{M}' \cdot \boldsymbol{b},$$

and we have that:

$$\boldsymbol{M}' \cdot \boldsymbol{b}^* = \boldsymbol{M} \cdot \boldsymbol{b}^* + \boldsymbol{m_a} = \boldsymbol{u}_{d+1}.$$

Therefore $P'_1 \cup P_2$ is a set of probes (for the compression algorithm $\mathcal{A}$) which satisfies Condition 1.

This concludes the proof. □

4.4 Theoretical Lower and Upper Bounds

In this section, we exhibit lower and upper bounds for the randomness complexity of a d-private compression algorithm for multiplication. We first prove an algebraic result and an intermediate lemma that we then use to show that at least $d+1$ random bits are required to construct a d-private compression algorithm for multiplication, for any $d \geq 3$ (and 2 random bits are required for $d = 2$). Finally, we provide a (non-constructive) proof that for large enough d, there exists a d-private multiplication algorithm with a randomness complexity $O(d \cdot \log d)$.

4.4.1 A Splitting Lemma

We first prove an algebraic result, stated in the lemma below, that we further use to prove Lemma 4.4.2. The latter allows us to easily exhibit attacks in order to prove our lower bounds.

Lemma 4.4.1. *Let $n \geq 1$. Let $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \mathbb{F}_2^{n \times n}$ such that $\boldsymbol{M}_0 + \boldsymbol{M}_1 = \boldsymbol{U}_n$. Then, there exists a vector $\boldsymbol{v} \in \mathbb{F}_2^n$ such that:*

$$\boldsymbol{M}_0 \cdot \boldsymbol{v} = \boldsymbol{u}_n \qquad \text{or} \qquad \boldsymbol{M}_1 \cdot \boldsymbol{v} = \boldsymbol{u}_n \qquad \text{or} \qquad \boldsymbol{M}_0^\intercal \cdot \boldsymbol{v} = \boldsymbol{u}_n \qquad \text{or} \qquad \boldsymbol{M}_1^\intercal \cdot \boldsymbol{v} = \boldsymbol{u}_n \ .$$

Proof. We show the above lemma by induction on n.

Base case: for $n = 1$, $\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{U} \in \mathbb{F}_2$, so $\boldsymbol{M}_0 + \boldsymbol{M}_1 = 1$, which implies $\boldsymbol{M}_0 = 1$ or $\boldsymbol{M}_1 = 1$ and the claim immediately follows.

Inductive case: let us assume that the claim holds for a fixed $n \geq 1$. Let us consider two matrices $\boldsymbol{M}_0, \boldsymbol{M}_1 \in \mathbb{F}_2^{(n+1)\times(n+1)}$ such that $\boldsymbol{M}_0 + \boldsymbol{M}_1 = \boldsymbol{U}_{n+1}$.

Clearly, if $\boldsymbol{M}_0$ (or $\boldsymbol{M}_1$) is invertible, then the claim is true (as $\boldsymbol{u}_{n+1}$ is in its range). Then, let us assume that $\boldsymbol{M}_0$ is not invertible. Then, there exists a non-zero vector $\boldsymbol{x} \in \ker(\boldsymbol{M}_{\boldsymbol{0}})$. Now, as $\mathrm{im}(\boldsymbol{U}_{n+1}) = \{\boldsymbol{0}_{n+1}, \boldsymbol{u}_{n+1}\}$, if $\boldsymbol{U}_{n+1} \cdot \boldsymbol{x} = \boldsymbol{u}_{n+1}$, then $\boldsymbol{M}_1 \cdot \boldsymbol{x} = \boldsymbol{u}_{n+1}$ and the claim is true. Hence, clearly, the claim is true if $\ker(\boldsymbol{M}_0) \neq \ker(\boldsymbol{M}_1)$ (with the symmetric remark). The same remarks hold when considering matrices $\boldsymbol{M}_0^\intercal$ and $\boldsymbol{M}_1^\intercal$.

Hence, the only remaining case to consider is when $\ker(\boldsymbol{M}_0) \neq \{\boldsymbol{0}_{n+1}\}$, $\ker(\boldsymbol{M}_0^\intercal) \neq \{\boldsymbol{0}_{n+1}\}$ and when $\ker(\boldsymbol{M}_0) = \ker(\boldsymbol{M}_1)$ and $\ker(\boldsymbol{M}_0^\intercal) = \ker(\boldsymbol{M}_1^\intercal)$. In particular, we have $\ker(\boldsymbol{M}_0) \subseteq \ker(\boldsymbol{U}_{n+1})$ and $\ker(\boldsymbol{M}_0^\intercal) \subseteq \ker(\boldsymbol{U}_{n+1})$.

Let $\boldsymbol{x} \in \ker(\boldsymbol{M}_0)$ (and then $\boldsymbol{x} \in \ker(\boldsymbol{M}_1)$ as well) be a non-zero vector. Up to some rearrangement of the *columns* of $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$ (by permuting some columns), we can assume without loss of generality that $\boldsymbol{x} = (1, \ldots, 1, 0, \ldots, 0)^\intercal$. Let $\boldsymbol{X}$ denote the matrix $(\boldsymbol{x}, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n+1})$ where $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)^\intercal$ is the i-th canonical vector of length $n+1$, so that it has a 1 in the i-th position and 0's everywhere else.

Now, let $\boldsymbol{y} \in \ker(\boldsymbol{M}_0^\intercal)$ (and then $\boldsymbol{y} \in \ker(\boldsymbol{M}_1^\intercal)$ as well) be a non-zero vector, so $\boldsymbol{y}^\intercal \cdot \boldsymbol{M}_0^\intercal = \boldsymbol{0}_{n+1}^\intercal$. Moreover, up to some rearrangement of the *rows* of $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, we can assume that $\boldsymbol{y} = (1, \ldots, 1, 0, \ldots, 0)^\intercal$. Let $\boldsymbol{Y}$ denote the matrix $(\boldsymbol{y}, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n+1})$.

Please note that rearrangements apply to the columns in the first case and to the rows in the second case, so we can assume without loss of generality that there exists both $\boldsymbol{x} \in \ker(\boldsymbol{M}_0)$ and $\boldsymbol{y} \in \ker(\boldsymbol{M}_0^\intercal)$ with the above form and matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are well defined.

We now define the matrices $\boldsymbol{M}_0' = \boldsymbol{Y}^\intercal \cdot \boldsymbol{M}_0 \cdot \boldsymbol{X}$ and $\boldsymbol{M}_1' = \boldsymbol{Y}^\intercal \cdot \boldsymbol{M}_1 \cdot \boldsymbol{X}$. We have:

$$\boldsymbol{M}_0' = \begin{pmatrix} & \boldsymbol{y}^\intercal \\ \boldsymbol{0}_n & \boldsymbol{I}_n \end{pmatrix} \cdot \boldsymbol{M}_0 \cdot \begin{pmatrix} \boldsymbol{x} & \boldsymbol{0}_n^\intercal \\ & \boldsymbol{I}_n \end{pmatrix} = \begin{pmatrix} & \boldsymbol{y}^\intercal \\ \boldsymbol{0}_n & \boldsymbol{I}_n \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{0}_{n+1} & \boldsymbol{M}_0^{(1)} \end{pmatrix}$$

where $\boldsymbol{M}_0^{(1)}$ is the matrix extracted from $\boldsymbol{M}_0$ by removing its first column. Hence:

$$\boldsymbol{M}_0' = \begin{pmatrix} 0 & \boldsymbol{0}_n^\intercal \\ \boldsymbol{0}_n & \boldsymbol{M}_0^{(1,1)} \end{pmatrix}$$

where $\boldsymbol{M}_0^{(1,1)}$ is the matrix extracted from $\boldsymbol{M}_0$ by removing its first column and its first row. Similar equation holds for $\boldsymbol{M}_1'$ as well. Thus, it is clear that:

$$\boldsymbol{M}_0' + \boldsymbol{M}_1' = \begin{pmatrix} 0 & \boldsymbol{0}_n^\intercal \\ \boldsymbol{0}_n & \boldsymbol{U}_n \end{pmatrix} .$$

Let us consider the matrices $\boldsymbol{M}''_0$ and $\boldsymbol{M}''_1$ in $\mathbb{F}_2^{n\times n}$ that are extracted from matrices $\boldsymbol{M}'_0$ and $\boldsymbol{M}'_1$ by removing their first row and their first column (i.e., $\boldsymbol{M}''_i = \boldsymbol{M}'^{(1,1)}_i$ with the previous notation). Then, it is clear that $\boldsymbol{M}''_0 + \boldsymbol{M}''_1 = \boldsymbol{U}_n$. As matrices in $\mathbb{F}_2^{n\times n}$, by induction hypothesis, there exists $\boldsymbol{v}'' \in \mathbb{F}_2^n$ such that at least one of the 4 propositions from Lemma 4.4.1 holds. We can assume without loss of generality that $\boldsymbol{M}''_0 \cdot \boldsymbol{v}'' = \boldsymbol{u}_n$.

Let $\boldsymbol{v}' = \begin{pmatrix} 0 \\ \boldsymbol{v}'' \end{pmatrix} \in \mathbb{F}_2^{n+1}$. Then, we have:

$$\boldsymbol{M}'_0 \cdot \boldsymbol{v}' = \begin{pmatrix} 0 & \boldsymbol{0}_n^\intercal \\ \boldsymbol{0}_n & \boldsymbol{M}''_0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ \boldsymbol{v}'' \end{pmatrix} = \begin{pmatrix} \boldsymbol{0}_n \cdot \boldsymbol{v}'' \\ \boldsymbol{M}''_0 \cdot \boldsymbol{v}'' \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{u}_n \end{pmatrix}.$$

Now, let $\boldsymbol{v} = \boldsymbol{X} \cdot \boldsymbol{v}'$ and $\boldsymbol{w} = \boldsymbol{M}_0 \cdot \boldsymbol{w}$, so $\boldsymbol{Y}^\intercal \cdot \boldsymbol{w} = \boldsymbol{Y}^\intercal \cdot \boldsymbol{M}_0 \boldsymbol{X} \cdot \boldsymbol{v}' = \boldsymbol{M}'_0 \cdot \boldsymbol{v}' = \begin{pmatrix} 0 \\ \boldsymbol{u}_n \end{pmatrix}$. Moreover, as $\boldsymbol{Y}$ is invertible, $\boldsymbol{w}$ is the *unique* vector such that $\boldsymbol{Y}^\intercal \cdot \boldsymbol{w} = \begin{pmatrix} 0 \\ \boldsymbol{u}_n \end{pmatrix}$. Finally, as the vector $\boldsymbol{u}_{n+1}$ satisfies $\boldsymbol{Y}^\intercal \cdot \boldsymbol{u}_{n+1} = \begin{pmatrix} 0 \\ \boldsymbol{u}_n \end{pmatrix}$, then $\boldsymbol{w} = \boldsymbol{u}_{n+1}$, and the claim follows for $n+1$, since $\boldsymbol{v}$ satisfies $\boldsymbol{M}_0 \cdot \boldsymbol{v} = \boldsymbol{w} = \boldsymbol{u}_{n+1}$.

Conclusion: The claim follows for any $n \geq 1$, and so does Lemma 4.4.1. □

We can now easily prove the following statement that is our main tool for proving our lower bounds, as explained after its proof.

Lemma 4.4.2. *Let $\mathcal{A}$ be a d-compression algorithm for multiplication. If there exist two sets S_1 and S_2 of at most d probes such that $s_i = \sum_{p\in S_i} p$ does not functionally depend on any of the random bits, for $i \in \{0,1\}$, and such that $s_0 + s_1 = a \cdot b$, then $\mathcal{A}$ is not d-private.*

Proof. Let $\mathcal{A}$, S_0, S_1, s_0 and s_1 defined in the above statement. Then, there exists $\boldsymbol{M}_i \in \mathbb{F}_2^{(d+1)\times(d+1)}$ such that $s_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M}_i \cdot \boldsymbol{b}$, for $i \in \{0,1\}$. Furthermore, as $s_0 + s_1 = a \cdot b = \boldsymbol{a}^\intercal \cdot \boldsymbol{U}_{d+1} \cdot \boldsymbol{b}$, we have $\boldsymbol{M}_0 + \boldsymbol{M}_1 = \boldsymbol{U}_{d+1}$. Hence, via Lemma 4.4.1, there exists $\boldsymbol{v} \in \mathbb{F}_2^{d+1}$ and $i \in \{0,1\}$ such that $\boldsymbol{M}_i \cdot \boldsymbol{v} = \boldsymbol{u}_{d+1}$ or $\boldsymbol{M}_i^\intercal \cdot \boldsymbol{v} = \boldsymbol{u}_{d+1}$. This means that $\boldsymbol{u}_{d+1}$ is in the row subspace or in the column subspace of $\boldsymbol{M}_i$, and therefore, $\boldsymbol{M}_i$ satisfies Condition 1. Therefore, as $|S_i| \leq d$, applying Theorem 4.3.1, $\mathcal{A}$ is not d-private. Lemma 4.4.2 follows. □

We use the above lemma to prove our lower bounds as follows: for proving that at least $R(d)$ random bits are required in order to achieve d-privacy for a compression algorithm for multiplication, we prove that any algorithm with a lower randomness complexity is not d-private by exhibiting two sets of probes S_0 and S_1 that satisfy the requirements of Lemma 4.4.2.

4.4.2 Simple Linear Lower Bound

As a warm-up, we show that at least d random bits are required, for $d \geqslant 2$.

Theorem 4.4.3. *Let $d \geqslant 2$. Let us consider a d-compression algorithm for multiplication $\mathcal{A}$. If $\mathcal{A}$ uses only $d-1$ random bits, then $\mathcal{A}$ is not d-private.*

Proof. Let $r_1, \dots, r_{d-1}$ denote the random bits used by $\mathcal{A}$. Let $c_0, \dots, c_d$ denote the outputs of $\mathcal{A}$. Let us define $\boldsymbol{N} \in \mathbb{F}_2^{(d-1)\times d}$ as the matrix whose coefficients $n_{i,j}$ are equal to 1 if and only if c_j

functionally depends on r_i, for $1 \leqslant i \leqslant d-1$ and $1 \leqslant j \leqslant d$. Please note in particular that $\boldsymbol{N}$ does not depend on c_0.

As a matrix over $\mathbb{F}_2$ with d columns and $d-1$ rows, there is necessarily a vector $\boldsymbol{w} \in \mathbb{F}_2^d$ with $\boldsymbol{w} \neq \boldsymbol{0}_d$ such that $\boldsymbol{N} \cdot \boldsymbol{w} = \boldsymbol{0}_{d-1}$.

The latter implies that the expression $s_0 = \sum_{i=1}^d w_i \cdot c_i$ does not functionally depend on any of the r_k's. Furthermore, by correctness, we also have that $s_1 = c_0 + \sum_{i=1}^d (1 - w_i) \cdot c_i$ does not functionally depend on any of the r_k's, and $s_0 + s_1 = \sum_{i=0}^d c_i = a \cdot b$. Then, the sets of probes $S_0 = \{c_i \mid w_i = 1\}$ and $S_1 = \{c_0\} \cup \{c_i \mid w_i = 0\}$ (whose cardinalities are at most d) satisfy the requirements of Lemma 4.4.2, and then, $\mathcal{A}$ is not d-private. Theorem 4.4.3 follows. □

4.4.3 Better Linear Lower Bound

We now show that at least $d+1$ random bits are actually required if $d \geqslant 3$.

Theorem 4.4.4. *Let $d \geqslant 3$. Let us consider a d-compression algorithm for multiplication $\mathcal{A}$. If $\mathcal{A}$ uses only d random bits, then $\mathcal{A}$ is not d-private.*

Proof. Let $r_1, \ldots, r_d$ denote the random bits used by $\mathcal{A}$. Let $c_0, \ldots, c_d$ denote the outputs of $\mathcal{A}$. The proof is organized through 4 steps as follows:

1. we prove that at least one c_i (further referred to as c_0) functionally depends on at least two distinct random bits,
2. we prove that at least two shares c_0 and c_j (further referred to as c_1) both functionally depend on at least two distinct random bits,
3. we prove that at least one random bit r_1 is such that no additive share functionally depends on only r_1,
4. we exhibit an attack.

Step 1: c_0 functionally depends on at least two distinct random bits.

Let us first show that at least one of the $(c_i)_{0 \leqslant i \leqslant d}$ functionally depends on two different random bits. By contradiction, let us assume that every $(c_i)_{0 \leqslant i \leqslant d}$ functionally depends on at most 1 random bit. Then, as there are d random bits, it implies that, either one c_i does not functionally depend on any random bit, or there exist $i < j$ such that c_i and c_j functionally depend on the same random bit:

- In the first case, let us assume that c_0 does not functionally depend on any random bit. Then, by correctness, neither does $\sum_{i=1}^d c_i$. Then, $S_0 = \{c_0\}$ and $S_1 = \{c_1, \ldots, c_d\}$ satisfy the requirements of Lemma 4.4.2 and $\mathcal{A}$ is not d-private.
- In the second case, let us assume without loss of generality that c_0 and c_1 functionally depend on the same random bit (and only on this one by assumption). Then, $c_0 + c_1$ does not depend on any random bit and so does $\sum_{i=2}^d c_i$ via correctness. Then, $S_0 = \{c_0, c_1\}$ and $S_1 = \{c_2, \ldots, c_d\}$ satisfy the requirements of Lemma 4.4.2 and $\mathcal{A}$ is not d-private.

Then, we can now assume without loss of generality that c_0 functionally depends on at least two distinct random bits.

Step 2: both c_0 and c_1 functionally depend on at least two distinct random bits.

By contradiction, let us now assume that for all $1 \leq i \leq d$, c_i functionally depends on only one random bit. In order to achieve correctness, there must exist c_i, c_j with $1 \leq i < j \leq d$ that respectively depend on the first and second of the two distinct random bits on which c_0 functionally depends on. As we assume that there are d random bits in total, all the d random bits on which the c_i's functionally depend on for $i \geqslant 1$ have to be different. Hence, we can assume without loss of generality that c_i functionally depends on r_i, for $1 \leq i \leq d$. Thus, by correctness, c_0 functionally depends on all r_i for $i = 1, \dots, d$.

Then, we can simply probe the first subsum p of c_0 that functionally depends on at least 2 distinct random bits (actually, any subsum that functionally depends on at least 2 and at most $d-1$ random bits would work).

Let us denote by $I = \{i_1, i_2\}$ the set of indices corresponding to the random bits on which p functionally depends on. Then, $S_0 = \{p, c_{i_1}, c_{i_2}\}$ and $S_1 = \{c_0, p\} \cup \{c_i \mid i \in \{1, \dots, d\} \backslash \{i_1, i_2\}\}$ satisfy the requirements of Lemma 4.4.2 and $\mathcal{A}$ is not d-private.

Therefore, it is not possible that every c_i for $1 \leq i \leq d$ functionally depends on only one random bit and thus, there are at least two c_i, c_j with $i \neq j$ that functionally depend on at least 2 distinct random bits. We can assume without loss of generality that c_0 and c_1 each functionally depend on at least 2 distinct random bits.

Step 3: r_1 is such that no c_i (for $i = 0, \dots, d$) functionally depends on only r_1 (and on no other random bits).

By correctness, c_0 functionally depends on all the random bits on which $\sum_{i=1}^{d} c_i$ functionally depends on. We just proved that c_1 functionally depends on at least two distinct bits. Also, all random bits have to be used (otherwise there are at most $d-1$ bits and Theorem 4.4.3 already proves that $\mathcal{A}$ is not d-private). Consequently, each of the at most $d-2$ random bits on which c_1 does not functionally depend on have to satisfy that at least one of the c_i for $2 \leq i \leq d$ functionally depends on this random bit. Thus, it is not possible that all c_i for $i \geqslant 2$ functionally depend on only 1 random bit (otherwise, there would exist $2 \leqslant i_1 < i_2 \leqslant d$ such that c_{i_1} and c_{i_2} functionally depend only on the same random bit. Indeed, in that case, $S_0 = \{c_{i_1}, c_{i_2}\}$ and $S_1 = \{c_0\} \cup \{c_i \mid i \in \{1, \dots, d\} \backslash \{i_1, i_2\}\}$ satisfy the requirements of Lemma 4.4.2 and $\mathcal{A}$ is not d-private.

Thus, this proves that at least one of the random bits, say r_1 without loss of generality, is such that no c_i (for $i = 0, \dots, d$) functionally depends on only r_1 (and on no other random bits).

Step 4: there exists an attack with at most d probes.

Let us now consider, similarly to what we did in the proof of Theorem 4.4.3, the matrix $\boldsymbol{N} \in \mathbb{F}_2^{d \times d}$ defined as the matrix whose coefficients $n_{i,j}$ are equal to 1 if and only if c_j functionally depends on the random bit r_i, for $1 \leqslant i, j \leqslant d$. Please note once again that this matrix does not depend of c_0. In order to prevent the same kind of attack than the one we used in the proof of Theorem 4.4.3, it is clear that this matrix has to be invertible. Hence, there exists $\boldsymbol{w} \in \mathbb{F}_2^d$ such that $\boldsymbol{N} \cdot \boldsymbol{w} = \boldsymbol{e}_1$ where $\boldsymbol{e}_1$ denotes the first vector of the canonical basis of $\mathbb{F}_2^d$. Moreover, since c_0 and $\sum_{i=1}^{d} c_i$ both functionally depends on the same at least 2 distinct random bits, then by correctness, we have $\boldsymbol{w} \neq 1^d$. Also, the Hamming weight of $\boldsymbol{w}$ is at least 2, since r_1 never appears alone in an additive share, which implies that $\boldsymbol{e}_1$ is not a column in $\boldsymbol{N}$. Hence, we have $2 \leqslant \mathtt{HW}(\boldsymbol{w}) \leqslant d-1$.

To conclude the proof, we just note that $S_0 = \{r', c_0\} \cup \{c_i \mid w_i = 0\}$ and $S_1 = \{r'\} \cup \{c_i \mid w_i = 1\}$ satisfy the requirements of Lemma 4.4.2 and $\mathcal{A}$ is not d-private.

Algorithm 3 Random algorithm

```
Require: sharing $(\alpha_{i,j})_{0 \le i,j \le d}$
Ensure: sharing $(c_i)_{0 \le i \le d}$
  for $i = 1$ to $R$ do
    $r_i \overset{\$}{\leftarrow} \mathbb{F}_2$
  for $i = 0$ to $d$ do
    $c_i \leftarrow \alpha_{i,i}$
    for $j = i + 1$ to $d$ do
      $c_i \leftarrow c_i + \rho(i,j) + \alpha_{i,j} + \alpha_{j,i}$          ▷ $\rho(i,j)$ is not computed first
  $c_d \leftarrow c_d + \rho(d,d)$
```

Theorem 4.4.4 follows. □

4.4.4 (Non-Constructive) Quasi-Linear Upper Bound

We now construct a d-private compression algorithm for multiplication which requires a quasi-linear number of random bits. More precisely, we show that with non-zero probability, a random algorithm in some family of algorithms (using a quasi-linear number of random bits) is secure, which directly implies the existence of such an algorithm. Note that it is an interesting open problem (though probably difficult) to derandomize this construction.

Concretely, let d be some masking order and R be some number of random bits (used in the algorithm), to be fixed later. For $i = 0, \dots, d-1$ and $j = i+1, \dots, d$, let us define $\rho(i,j)$ as:

$$\rho(i,j) = \textstyle\sum_{k=1}^{R} X_{i,j,k} \cdot r_k$$

with $X_{i,j,k} \overset{\$}{\leftarrow} \{0,1\}$ for $i = 0, \dots, d-1$, $j = i+1, \dots, d$ and $k = 1, \dots, R$, so that $\rho(i,j)$ is a random sum of all the random bits $r_1, \dots, r_R$ where each bit appears in $\rho(i,j)$ with probability 1/2. We also define $X_{d,d,k} = \sum_{i=0}^{d-1} \sum_{j=i+1}^{d} X_{i,j,k}$ and $\rho(d,d)$ as:

$$\rho(d,d) = \textstyle\sum_{k=1}^{R} X_{d,d,k} \cdot r_k.$$

We generate a (random) algorithm as in Algorithm 3. This algorithm is correct because the sum of all $\rho(i,j)$ is equal to 0.

We point out that we use two kinds of random which should not be confused: the R fresh random bits $r_1, \dots, r_R$ used in the algorithm to ensure its d-privacy (R is what we really want to be as low as possible), and the random variables $X_{i,j,k}$ used to define a random family of such algorithms (which are "meta"-random bits). In a concrete implementation or algorithm, these latter values are fixed.

Lemma 4.4.5. *Algorithm 3 is d-private with probability at least*

$$1 - \binom{(R+3) \cdot d \cdot (d+1)/2}{d} \cdot 2^{-R}$$

over the values of the $X_{i,j,k}$'s.

Proof. In order to simplify the proof, we are going to show that, with non-zero probability, there is no set of probes $P = \{p_1, \dots, p_\ell\}$ with $\ell \le d$ that satisfies Condition 2. In particular, this implies that, with non-zero probability, there is no set of probes $P = \{p_1, \dots, p_\ell\}$ with $\ell \leq d$ that satisfies Condition 1, which, via Theorem 4.3.1, is equivalent to the algorithm being d-private.

One can only consider sets of exactly d probes as if there is a set of $\ell < d$ probes P' that satisfies Condition 2, one can always complete P' into a set P with exactly d probes by adding $d - \ell$ times the same probe on some input $\alpha_{i,j}$ such that P' initially does not depend on $\alpha_{i,j}$. That is, if $\boldsymbol{M}'$ denotes the matrix such that $\sum_{p' \in P'} p' = \boldsymbol{a} \cdot \boldsymbol{M}' \cdot \boldsymbol{b}$, one could complete P' with any $\alpha_{i,j}$ such that $m'_{i,j} = 0$, so that P, with $\sum_{p \in P} p = \boldsymbol{a} \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ still satisfies Condition 2 if P' initially satisfied the condition.

Thus, let us consider an arbitrary set of d probes $P = \{p_1, \dots, p_d\}$ and let us bound the probability that P satisfies Condition 2. Let $f = \sum_{i=1}^{d} p_i$. Let us first show that f has to contain at least one $\rho(i,j)$ (meaning that it appears an odd number of times in the sum). Let us assume the contrary, so f does not contain any $\rho(i,j)$. Every $\rho(i,j)$ appears only once in the shares (in the share c_i precisely). Then, one can assume that every probe is made on the same share. Let us assume (without loss of generality) that every probe is made on c_0. If no probe contains any $\rho(0,j)$, then clearly P cannot satisfy Condition 2 as this means that each probe contain at most one $\alpha_{0,j}$, to P cannot contain more than d different $\alpha_{0,j}$. Hence, at least one (so at least two) probe contains at least one $\rho(0,j)$. We note that every probe has one of the following form: either it is exactly a random r_k, a share $\alpha_{0,j}$, a certain $\rho(0,j)$, a certain $\rho(0,j) + \alpha_{0,j}$ or $\rho(0,j) + \alpha_{0,j} + \alpha_{j,0}$, or a subsum (starting from $\alpha_{0,0}$) of c_0. Every form gives at most one $\alpha_{0,j}$ with a new index j except probes on subsums. However, in any subsum, there is always a random $\rho(i,j)$ between $\alpha_{0,j}$ and $\alpha_{0,j+1}$ and one needs to get all the $d+1$ indices to get a set satisfying Condition 2. Then, it is clear that one cannot achieve this unless there is a $\rho(i,j)$ that does not cancel out in the sum, which is exactly what we wanted to show. Now, let $1 \leq k \leq R$ be an integer and let us compute the probability (over the $X_{i,j,k}$'s) that f contains r_k. There exists some set S of pairs (i,j), such that f is the sum of $\sum_{(i,j) \in S} X_{i,j,k} \cdot r_k$ and some other expression not containing any $X_{i,j,k} \cdot r_k$. From the previous point, S is not empty. Furthermore, as there are $d+1$ outputs $c_0, \dots, c_d$ and as there are only d probes, S cannot contain all the possible pairs (i,j), and therefore, all the random variables $X_{i,j,k}$ for $(i,j) \in S$ are mutually independent. Therefore, $\sum_{(i,j) \in S} X_{i,j,k}$ is 1 with probability $1/2$ and f functionally depends on the random r_k with probability $1/2$. As there are R possible random bits, f does not functionally depend on any r_k (and then P satisfies Condition 2) with probability $(1/2)^R$.

There are N possibles probes with

$$N \leq \frac{d \cdot (d+1)}{2} + R + (R+2) \cdot \frac{d \cdot (d-1)}{2} \leq (R+3) \cdot \frac{d \cdot (d+1)}{2},$$

as every ρ contains at most R random bits r_k. Also, there are $\binom{N}{d}$ possible sets $P = \{p_1, \dots, p_d\}$. Therefore, by union bound, the above algorithm is not secure (so there is an attack) with probability at most

$$\binom{N}{d} / 2^R \leq \binom{(R+3) \cdot d \cdot (d+1)/2}{d} \cdot 2^{-R}$$

which concludes the proof of Lemma 4.4.5. □

Theorem 4.4.6. *For some $R = O(d \cdot \log d)$, there exists a choice of $\rho(i,j)$ such that Algorithm 3 is a d-private d-compression algorithm for multiplication, when $d \to \infty$.*

We just need to remark that for some $R = O(d \cdot \log d)$, the probability that Algorithm 3 is d-private, according to Lemma 4.4.5 is non-zero.

Proof. We remark that a random algorithm as defined in Algorithm 3 and Lemma 4.4.5 is secure with probability at least

$$\begin{aligned} 1 - \binom{(R+3)\cdot d \cdot (d+1)/2}{d} \cdot 2^{-R} &\geq 1 - ((R+3)\cdot d \cdot (d+1)/2)^d \cdot 2^{-R} \\ &= 1 - 2^{d\cdot \log((R+3)\cdot d\cdot (d+1)/2)-R}. \end{aligned}$$

When this probability is greater than 0, then there exists necessarily a choice of $\rho(i,j)$ leading to a secure algorithm. This condition can be rewritten as:

$$d \cdot \log((R+3)\cdot d \cdot (d+1)/2) - R < 0.$$

Let us take $R = K \cdot d \cdot \log d - 3$ with K some constant to be fixed later. As $d \cdot (d+1)/2 \leq d^2$, we have

$$\begin{aligned} d \cdot \log((R+3)\cdot d \cdot (d+1)/2) - R &\leq d \cdot \log(K \cdot d^3 \cdot \log d) - K \cdot d \cdot \log d + 3 \\ &= d \cdot ((3-K)\cdot \log d + \log K + \log\log d) + 3 \end{aligned}$$

When K is large enough, this is always negative. Theorem 4.4.6 easily follows. □

4.5 New Construction

The goal of this section is to propose a new d-private multiplication algorithm. Compared to the construction in [ISW03], our construction halves the number of required random bits. It is therefore the most efficient existing construction of a d-private multiplication.

Some rationales behind our new construction may be found in the two following necessary conditions deduced from a careful study of the original work of Ishai, Sahai and Wagner [ISW03].

Lemma 4.5.1. *Let $\mathcal{A}(\boldsymbol{a},\boldsymbol{b};\boldsymbol{r})$ be a d-compression algorithm for multiplication. Let f be an intermediate result taking the form $f = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{s}^\intercal \cdot \boldsymbol{r}$. Let t denote the greatest Hamming weight of an element in the vector subspace generated by the rows of $\boldsymbol{M}$ or by the columns of $\boldsymbol{M}$. If $\mathtt{HW}(\boldsymbol{s}) < t-1$, then $\mathcal{A}(\boldsymbol{a},\boldsymbol{b};\boldsymbol{r})$ is not d-private.*

Proof. By definition of $\boldsymbol{s}$, the value $\boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ can be recovered by probing f and then each of the $\mathtt{HW}(\boldsymbol{s}) < t-1$ random bits on which $\boldsymbol{s}^\intercal \cdot \boldsymbol{r}$ functionally depends and by summing all these probes. Let $P_1 = \{f, p_1, \dots, p_j\}$ with $j < t-1$ denote the set of these at most $t-1$ probes. Then, we just showed that $f + \sum_{i=1}^{j} p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$.

To conclude the proof, we want to argue that there is a set of at most $d-(t-1)$ probes $P_2 = \{p'_1, \dots, p'_k\}$ such that $f + \sum_{i=1}^{j} p_i + \sum_{\ell=1}^{k} p'_\ell = \boldsymbol{a}^\intercal \cdot \boldsymbol{M'} \cdot \boldsymbol{b}$, where $\boldsymbol{M'}$ is a matrix such that $\boldsymbol{u}_{d+1}$ is in its row space or in its column space. If such a set P_2 exists, then the set of probes $P_1 \cup P_2$ (whose cardinality is at most d) satisfies Condition 1, and then $\mathcal{A}$ is not d-private, via Theorem 4.3.1.

We now use the fact that there is a vector of Hamming weight t in the row space or in the column space of $\boldsymbol{M}$. We can assume (without loss of generality) that there exists a vector $\boldsymbol{w} \in \mathbb{F}_2^{d+1}$ of Hamming weight t in the column subspace of $\boldsymbol{M}$, so that $\boldsymbol{w} = \sum_{j \in J} \boldsymbol{m}_j$, with $J \subseteq \{0,\dots,d\}$ and $\boldsymbol{m}_j$ the j-th column vector of $\boldsymbol{M}$. Let $i_1, \dots, i_{d+1-t}$ denote the indices i of $\boldsymbol{w}$ such that $w_i = 0$. Then, let $j \in J$, we claim that $P_2 = \{\alpha_{i_1,j}, \dots, \alpha_{i_{d+1-t},j}\}$ allows us to conclude the proof. Please note that all these values are probes of intermediate values of $\mathcal{A}$.

$$\begin{array}{lllllllllllll}
\hat{\alpha}_{0,0} & r_{0,1} & & r_{0,2} & & r_{0,3} & & r_{0,4} & & r_{0,5} & & r_{0,6} & \\
\hat{\alpha}_{1,1} & (r_{0,1} & \hat{\alpha}_{0,1}) & r_{1,2} & & r_{1,3} & & r_{1,4} & & r_{1,5} & & r_{1,6} & \\
\hat{\alpha}_{2,2} & (r_{0,2} & \hat{\alpha}_{0,2}) & (r_{1,2} & \hat{\alpha}_{1,2}) & r_{2,3} & & r_{2,4} & & r_{2,5} & & r_{2,6} & \\
\hat{\alpha}_{3,3} & (r_{0,3} & \hat{\alpha}_{0,3}) & (r_{1,3} & \hat{\alpha}_{1,3}) & (r_{2,3} & \hat{\alpha}_{2,3}) & r_{3,4} & & r_{3,5} & & r_{3,6} & \\
\hat{\alpha}_{4,4} & (r_{0,4} & \hat{\alpha}_{0,4}) & (r_{1,4} & \hat{\alpha}_{1,4}) & (r_{2,4} & \hat{\alpha}_{2,4}) & (r_{3,4} & \hat{\alpha}_{3,4}) & r_{4,5} & & r_{4,6} & \\
\hat{\alpha}_{5,5} & (r_{0,5} & \hat{\alpha}_{0,5}) & (r_{1,5} & \hat{\alpha}_{1,5}) & (r_{2,5} & \hat{\alpha}_{2,5}) & (r_{3,5} & \hat{\alpha}_{3,5}) & (r_{4,5} & \hat{\alpha}_{4,5}) & r_{5,6} & \\
\hat{\alpha}_{6,6} & (r_{0,6} & \hat{\alpha}_{0,6}) & (r_{1,6} & \hat{\alpha}_{1,6}) & (r_{2,6} & \hat{\alpha}_{2,6}) & (r_{3,6} & \hat{\alpha}_{3,6}) & (r_{4,6} & \hat{\alpha}_{4,6}) & (r_{5,6} & \hat{\alpha}_{5,6})
\end{array}$$

Figure 4.2: ISW construction for $d = 6$, with $\hat{\alpha}_{i,j} = \alpha_{i,j} + \alpha_{j,i}$

$$\begin{array}{lllllllllllll}
\hat{\alpha}_{0,0} & (r_{0,6} & \hat{\alpha}_{0,6}) & (r_{0,5} & \hat{\alpha}_{0,5}) & (r_{0,4} & \hat{\alpha}_{0,4}) & (r_{0,3} & \hat{\alpha}_{0,3}) & (r_{0,2} & \hat{\alpha}_{0,2}) & (r_{0,1} & \hat{\alpha}_{0,1}) \\
\hat{\alpha}_{1,1} & (r_{1,6} & \hat{\alpha}_{1,6}) & (r_{1,5} & \hat{\alpha}_{1,5}) & (r_{1,4} & \hat{\alpha}_{1,4}) & (r_{1,3} & \hat{\alpha}_{1,3}) & (r_{1,2} & \hat{\alpha}_{1,2}) & r_{0,1} & \\
\hat{\alpha}_{2,2} & (r_{2,6} & \hat{\alpha}_{2,6}) & (r_{2,5} & \hat{\alpha}_{2,5}) & (r_{2,4} & \hat{\alpha}_{2,4}) & (r_{2,3} & \hat{\alpha}_{2,3}) & r_{1,2} & & r_{0,2} & \\
\hat{\alpha}_{3,3} & (r_{3,6} & \hat{\alpha}_{3,6}) & (r_{3,5} & \hat{\alpha}_{3,5}) & (r_{3,4} & \hat{\alpha}_{3,4}) & r_{2,3} & & r_{1,3} & & r_{0,3} & \\
\hat{\alpha}_{4,4} & (r_{4,6} & \hat{\alpha}_{4,6}) & (r_{4,5} & \hat{\alpha}_{4,5}) & r_{3,4} & & r_{2,4} & & r_{1,4} & & r_{0,4} & \\
\hat{\alpha}_{5,5} & (r_{5,6} & \hat{\alpha}_{5,6}) & r_{4,5} & & r_{3,5} & & r_{2,5} & & r_{1,5} & & r_{0,5} & \\
\hat{\alpha}_{6,6} & r_{5,6} & & r_{4,6} & & r_{3,6} & & r_{2,6} & & r_{1,6} & & r_{0,6} &
\end{array}$$

Figure 4.3: First step of our new construction for $d = 6$, with $\hat{\alpha}_{i,j} = \alpha_{i,j} + \alpha_{j,i}$

Indeed, we have $f + \sum_{i=1}^{j} p_i + \sum_{k=1}^{d+1-t} \alpha_{i_k,j} = \boldsymbol{a}^\intercal \cdot \boldsymbol{M'} \cdot \boldsymbol{b}$ where all coefficients of $\boldsymbol{M'}$ are the same as coefficients of $\boldsymbol{M}$ except for coefficients in positions $(i_1, j), \ldots, (i_{d+1-t}, j)$ which are the opposite, and now $\sum_{j \in J} \boldsymbol{m}'_j = \boldsymbol{u}_{d+1}$, where $\boldsymbol{m}'_j$ is the j-th column vector of $\boldsymbol{M'}$. Lemma 4.5.1 easily follows. □

In our construction, we satisfy the necessary condition in Lemma 4.5.1 by ensuring that any intermediate result that functionally depends on t shares of a (resp. of b) also functionally depends on at least $t - 1$ random bits.

The multiplication algorithm of Ishai, Sahai and Wagner is the starting point of our construction. Before exhibiting it, we hence start by giving the basic ideas thanks to an illustration in the particular case $d = 6$. In Figure 4.2 we recall the description of ISW already introduced in Section 4.2.3.

The first step of our construction is to order the expressions $\hat{\alpha}_{i,j}$ differently. Precisely, to compute the output share c_i (which corresponds, in ISW, to the sum $r_{i,i,+} \sum_{j<i}(r_{j,i}+\hat{\alpha}_{j,i})+\sum_{j>i} r_{i,j}$ from left to right), we process $r_{i,i,+} \sum_{j<d-i}(r_{i,d-j}+\hat{\alpha}_{i,j})+\sum_{1\leqslant j\leqslant i} r_{d-j,i}$ from left to right. Of course, we also put particular care to satisfy the necessary condition highlighted by Lemma 4.5.1. This leads to the construction illustrated in Figure 4.3.

Then, the core idea is to decrease the randomness cost by reusing some well chosen random bit to protect different steps of the processing. Specifically, for any even positive number k, we show that replacing all the random bits $r_{i,j}$ such that $k = j - i$ with a fixed random bit r_k preserves the d-privacy of ISW algorithm. Note, however, that the computations then have to be performed with a slightly different bracketing in order to protect the intermediate variables which involve the same random bits. The obtained construction is illustrated in Figure 4.4.

Finally, we suppress from our construction the useless repetitions of random bits that appear at the end of certain computations. Hence, we obtain our new construction, illustrated in Figure 4.5.

Before proving that this scheme is indeed d-private, we propose a formal description in Algorithm 4. As can be seen, this new scheme involves $3d^2/2 + d(d+2)/4 + 2d$ sums if d is even and

$$\begin{array}{ccccccccccccc}
\hat{\alpha}_{0,0} & (r_{0,6} & \hat{\alpha}_{0,6} & r_5 & \hat{\alpha}_{0,5}) & (r_{0,4} & \hat{\alpha}_{0,4} & r_3 & \hat{\alpha}_{0,3}) & (r_{0,2} & \hat{\alpha}_{0,2} & r_1 & \hat{\alpha}_{0,1}) \\
\hat{\alpha}_{1,1} & (r_{1,6} & \hat{\alpha}_{1,6} & r_5 & \hat{\alpha}_{1,5}) & (r_{1,4} & \hat{\alpha}_{1,4} & r_3 & \hat{\alpha}_{1,3}) & (r_{1,2} & \hat{\alpha}_{1,2}) & r_1 & \\
\hat{\alpha}_{2,2} & (r_{2,6} & \hat{\alpha}_{2,6} & r_5 & \hat{\alpha}_{2,5}) & (r_{2,4} & \hat{\alpha}_{2,4} & r_3 & \hat{\alpha}_{2,3}) & r_{1,2} & & r_{0,2} & \\
\hat{\alpha}_{3,3} & (r_{3,6} & \hat{\alpha}_{3,6} & r_5 & \hat{\alpha}_{3,5}) & (r_{3,4} & \hat{\alpha}_{3,4}) & r_3 & & r_3 & & r_3 & \\
\hat{\alpha}_{4,4} & (r_{4,6} & \hat{\alpha}_{4,6} & r_5 & \hat{\alpha}_{4,5}) & r_{3,4} & & r_{2,4} & & r_{1,4} & & r_{0,4} & \\
\hat{\alpha}_{5,5} & (r_{5,6} & \hat{\alpha}_{5,6}) & r_5 & & r_5 & & r_5 & & r_5 & & r_5 & \\
\hat{\alpha}_{6,6} & r_{5,6} & & r_{4,6} & & r_{3,6} & & r_{2,6} & & r_{1,6} & & r_{0,6} &
\end{array}$$

Figure 4.4: Second step of our new construction for $d = 6$, with $\hat{\alpha}_{i,j} = \alpha_{i,j} + \alpha_{j,i}$

$$\begin{array}{ccccccccccccc}
\hat{\alpha}_{0,0} & (r_{0,6} & \hat{\alpha}_{0,6} & r_5 & \hat{\alpha}_{0,5}) & (r_{0,4} & \hat{\alpha}_{0,4} & r_3 & \hat{\alpha}_{0,3}) & (r_{0,2} & \hat{\alpha}_{0,2} & r_1 & \hat{\alpha}_{0,1}) \\
\hat{\alpha}_{1,1} & (r_{1,6} & \hat{\alpha}_{1,6} & r_5 & \hat{\alpha}_{1,5}) & (r_{1,4} & \hat{\alpha}_{1,4} & r_3 & \hat{\alpha}_{1,3}) & (r_{1,2} & \hat{\alpha}_{1,2}) & r_1 & \\
\hat{\alpha}_{2,2} & (r_{2,6} & \hat{\alpha}_{2,6} & r_5 & \hat{\alpha}_{2,5}) & (r_{2,4} & \hat{\alpha}_{2,4} & r_3 & \hat{\alpha}_{2,3}) & r_{1,2} & & r_{0,2} & \\
\hat{\alpha}_{3,3} & (r_{3,6} & \hat{\alpha}_{3,6} & r_5 & \hat{\alpha}_{3,5}) & (r_{3,4} & \hat{\alpha}_{3,4}) & r_3 & & & & & \\
\hat{\alpha}_{4,4} & (r_{4,6} & \hat{\alpha}_{4,6} & r_5 & \hat{\alpha}_{4,5}) & r_{3,4} & & r_{2,4} & & r_{1,4} & & r_{0,4} & \\
\hat{\alpha}_{5,5} & (r_{5,6} & \hat{\alpha}_{5,6}) & r_5 & & & & & & & & & \\
\hat{\alpha}_{6,6} & r_{5,6} & & r_{4,6} & & r_{3,6} & & r_{2,6} & & r_{1,6} & & r_{0,6} &
\end{array}$$

Figure 4.5: Application of our new construction for $d = 6$, with $\hat{\alpha}_{i,j} = \alpha_{i,j} + \alpha_{j,i}$

Algorithm 4 New construction for d-secure multiplication

```
Require: sharing $(\alpha_{i,j})_{0 \leq i,j \leq d}$
Ensure: sharing $(c_i)_{0 \leq i \leq d}$
for $i = 0$ to $d$ do                                          ▷ Random Bits Generation
    for $j = 0$ to $d - i - 1$ by 2 do
        $r_{i,d-j} \xleftarrow{\$} \mathbb{F}_2$
for $j = d - 1$ downto 1 by 2 do
    $r_j \xleftarrow{\$} \mathbb{F}_2$
for $i = 0$ to $d$ do                                          ▷ Multiplication
    $c_i \leftarrow \alpha_{i,i}$
    for $j = d$ downto $i + 2$ by 2 do
        $t_{i,j} \leftarrow r_{i,j} + \alpha_{i,j} + \alpha_{j,i} + r_{j-1} + \alpha_{i,j-1} + \alpha_{j-1,i}$;    $c_i \leftarrow c_i + t_{i,j}$
    if $i \not\equiv d \pmod 2$ then
        $t_{i,i+1} \leftarrow r_{i,i+1} + \alpha_{i,i+1} + \alpha_{i+1,i}$;    $c_i \leftarrow c_i + t_{i,i+1}$
        if $i \equiv 1 \pmod 2$ then                           ▷ Correction $r_i$
            $c_i \leftarrow c_i + r_i$
    else
        for $j = i - 1$ downto 0 do                            ▷ Correction $r_{i,j}$
            $c_i \leftarrow c_i + r_{j,i}$
```

$3(d^2-1)/2+(d+1)^2/4+3(d+1)/2$ if d is odd. In every case, it also involves $(d+1)^2$ multiplications and requires the generation of $d^2/4+d$ random values in $\mathbb{F}_2$ if d is even and $(d^2-1)/4+d$ otherwise (see Table 4.1 for values at several orders and comparison with ISW).

Proposition 4.5.2. *Algorithm 4 is d-private.*

Algorithm 4 was proven to be d-private with the verifier built by Barthe et al. [BBD+15b] up

Table 4.1: Complexities of ISW, our new d-private compression algorithm for multiplication and our specific algorithms at several orders

Complexities	Algorithm ISW	Algorithm 4	Algorithms 5, 6 and 7
Second-Order Masking			
sums	12	12	10
products	9	9	9
random bits	3	3	2
Third-Order Masking			
sums	24	22	20
products	16	16	16
random bits	6	5	4
Fourth-Order Masking			
sums	40	38	30
products	25	25	25
random bits	10	8	5
d^{th}-Order Masking			
sums	$2d(d+1)$	$\begin{cases} d(7d+10)/4 & (d \text{ even}) \\ (7d+1)(d+1)/4 & (d \text{ odd}) \end{cases}$	-
products	$(d+1)^2$	$(d+1)^2$	-
random bits	$d(d+1)/2$	$\begin{cases} d^2/4+d & (d \text{ even}) \\ (d^2-1)/4+d & (d \text{ odd}) \end{cases}$	-

to order $d = 6$.

Furthermore, we propose hereafter a pen-and-paper proof.

Proof. Inspired from simulation-based proofs of multiplications in [RP10; CPRR14], our proof consists in constructing two sets I and J of indices in $[0, d]$ of size at most d and such that the distribution of any d-tuple $(v_1, v_2, \ldots, v_d)$ of intermediate variables can be perfectly simulated from $\alpha_{I,J} = (\alpha_{i,j})_{i \in I, j \in J}$. This proves our statement as long as the cardinalities of I and J are smaller than $d+1$. We now describe the construction of I and J.

Construction of the sets I and J.

Initially, I and J are empty. We fill them in the following specific order according to the possible attacker's probes.

1. for any observed variable $\alpha_{i,i}$, add i to I and i to J.

2. for any observed variable $\alpha_{i,j}$, add i to I and j to J.

3. for any observed variable r_j, put j in I and j in J.

4. for any observed intermediate sum occurring during the computation of c_i, assign from shortest sums (in terms of number of terms) to longest sums:

   - if $i \notin I$, add i to I. Otherwise, if c_i involves corrective terms (i.e., random bits not in $t_{i,j}$), consider them successively (from left to right). For a random of the form $r_{j,i}$, if $j \notin I$, add j to I, otherwise, consider the next random. For a random of the form r_j, if $j \notin I$, add j to I. If there are no more corrective terms to consider, or if c_i does not involve corrective terms, consider the involved $t_{i,j}$ in reverse order (from right to left). Add to I the first index j that is not in I.

- if $i \notin J$, add i in J. Otherwise, if c_i involves corrective terms (i.e., random bits not in $t_{i,j}$),consider them successively (from left to right). For a random of the form $r_{j,i}$, if $j \notin J$, add j to J, otherwise, consider the next random. For a random of the form r_j, if $j \notin J$, add j to J. If there are no more corrective terms to consider, or if c_i does not involve corrective terms, consider the involved $t_{i,j}$ in reverse order (from right to left). Add to J the first index j that is not in J.

5. for any observed variable $r_{i,j}$:
   - if $i \notin I$, add i to I, otherwise add j to I.
   - if $i \notin J$, add i to J, otherwise add j to J.
6. for any observed intermediate sum t occurring during the computation of $t_{i,j}$ we distinguish two cases:
   - t is a sum of at most 3 terms[2]:
     - if $i \notin I$, add i to I, otherwise add j to I.
     - if $i \notin J$, add i to J, otherwise add j to J.
   - t is a sum of strictly more than 3 terms:
     - if $j-1 \notin I$, add $j-1$ to I. Otherwise, if $i \notin I$, add i to I, otherwise add j to I.
     - if $j-1 \notin J$, add $j-1$ to J. Otherwise, if $i \notin J$, add i to J, otherwise add j to J.

We can check that these categories cover all possible intermediate variables in our algorithm. Moreover, each observation adds at most one index in I and one index in J. With at most d probes, their cardinals hence cannot be greater than d.

Simulation phase.

Before simulating, we make the following useful observations.

(i). all variables whose expression involves $r_{i,j}$ are: $r_{i,j}$, $t_{i,j}$, c_i, c_j.

(ii). all variables whose expression involves r_{j-1} are: r_{j-1}, $t_{k,j}$, c_{j-1}, c_k, for any $k \leqslant j-2$.

(iii). all variables whose expression involves both $r_{i,j}$ and r_{j-1} are: c_i and $t_{i,j}$.

We now prove that every observed value can be perfectly simulated with the input shares whose indexes are among I and J.

1. any variable $\alpha_{i,i}$ is trivially simulated thanks to the fact that i is in I and in J.
2. any variable $\alpha_{i,j}$ is trivially simulated thanks to the fact that i is in I and j is in J.
3. any variable $r_{i,j}$ is assigned to a uniformly distributed random value, as it is the case in the real algorithm.

[2] Note that the case where it involves only one term $r_{i,j}$ has already been treated.

4. any variable r_j is assigned to a uniformly distributed random value, as it is the case in the real algorithm.

5. for any variable t of at most three terms manipulated during the computation of $t_{i,j}$:

   - if t is a sum of at most 3 terms (i.e., $t = r_{i,j} + \alpha_{i,j}$ or $t = r_{i,j} + \alpha_{i,j} + \alpha_{j,i}$), then necessarily we have $i \in I$ and $i \in J$. Moreover :
     - if $j \in I$ and $j \in J$, t can be perfectly simulated with $\alpha_{i,j}$ and $\alpha_{j,i}$ thanks to the indexes in I and J.
     - otherwise, we show that t can be assigned to a random value. In particular, we show that if t is non-random, we must have $i, j \in I$ and $i, j \in J$. The variable t involves $r_{i,j}$. As noted in Observation (i), this variable can only appear either alone, in c_i, in c_j, in another t' of less than three terms part of $t_{i,j}$, or in another t' of strictly more than three terms part of $t_{i,j}$.
       * $r_{i,j}$ appears alone: this probe involved $i \in I$ and $i \in J$, and hence the probe of t added j in I and j in J
       * $r_{i,j}$ appears in an observed c_i: this probe involved $i \in I$ and $i \in J$, and hence the probe of t added j in I and j in J
       * $r_{i,j}$ appears in an observed c_j: this probe involved $j \in I$ and $j \in J$ and hence the probe of t added i in I and i in J
       * $r_{i,j}$ appears in another observed t' of less than three terms: the probe of two variables t and t' of this kind leads to first $i \in I$ and $i \in J$ and then $j \in I$ and $j \in J$
       * $r_{i,j}$ appears in another observed t' of strictly more than three terms: in this case, t' also involves the random r_{j-1}. With Observation (ii), we know that r_{j-1} can either be observed alone, in c_{j-1}, in t'' of more than three terms part of $t_{k,j}$ or in c_k. Once again, considering r_{j-1} or c_{j-1}, and t and t', we get that $j-1, j, i \in I$ and $j-1, j, i \in J$. Considering t'' of more than three terms, or c_k, if $k = i$, we have already treated this case and we have $i, j \in I$ and $i, j \in J$, otherwise, the variable involves $r_{k,j}$. Once again thanks to Observation (i), we know exactly in which variables of the protocol $r_{k,j}$ can be involved. It can be checked that $i, j, k, j-1$ are in I and in J for each variables that are not part of $c_{k'}$ or $t_{k,j}$. Consequently, each other probe that does not imply $i, j \in I$ and $i, j \in J$ are variables of these kinds. However, each of these variables involves both r_{j-1} and $r_{k',j}$ for a certain k'. To summarize, t has been queried, which involves only $r_{i,j}$, and the only other possible variables involve r_{j-1} and $r_{\ell,j}$, where ℓ is the index of the line. Hence, the parity of the number of occurrences of r_{j-1} is different from the parity of the number of occurrences of $r_{\ell,j}$. This ensures that it is impossible to get rid of r_{j-1} and all variables $r_{\ell,j}$ at the same time. Therefore, in those cases t can be assigned to a random value.
   - if t is a sum of strictly more than 3 terms:
     - if $i, j, j-1 \in I$ and $i, j, j-1 \in J$, then t can be simulated from the indexes in I and J.
     - t involves $r_{i,j}$ and r_{j-1}. Observations (i) and (ii) provide us the variables in which these random bits are involved. For all but four cases, we trivially have $i, j, j-1 \in I$ and $i, j, j-1 \in J$. Those four cases are the queries of $(r_{i,j}, t')$ with t' part of $t_{k,j}$ and

Algorithm 5 Second-Order Compression Algorithm

```
Require: sharing $(\alpha_{i,j})_{0\leq i,j\leq 2}$
Ensure: sharing $(c_i)_{0\leq i\leq 2}$
  $r_0 \overset{\$}{\leftarrow} \mathbb{F}_2$;    $r_1 \leftarrow \mathbb{F}_2$
  $c_0 \leftarrow \alpha_{0,0} + r_0 + \alpha_{0,2} + \alpha_{2,0}$
  $c_1 \leftarrow \alpha_{1,1} + r_1 + \alpha_{0,1} + \alpha_{1,0}$
  $c_2 \leftarrow \alpha_{2,2} + r_0 + r_1 + \alpha_{1,2} + \alpha_{2,1}$
```

Algorithm 6 Third-Order Compression Algorithm

```
Require: sharing $(\alpha_{i,j})_{0\leq i,j\leq 3}$
Ensure: sharing $(c_i)_{0\leq i\leq 3}$
  $r_0 \overset{\$}{\leftarrow} \mathbb{F}_2$;    $r_1 \overset{\$}{\leftarrow} \mathbb{F}_2$;    $r_2 \overset{\$}{\leftarrow} \mathbb{F}_2$;    $r_3 \overset{\$}{\leftarrow} \mathbb{F}_2$
  $c_0 \leftarrow \alpha_{0,0} + r_0 + \alpha_{0,3} + \alpha_{3,0} + r_1 + \alpha_{0,2} + \alpha_{2,0}$
  $c_1 \leftarrow \alpha_{1,1} + r_2 + \alpha_{1,3} + \alpha_{3,1} + r_1 + \alpha_{1,2} + \alpha_{2,1}$
  $c_2 \leftarrow \alpha_{2,2} + r_3 + \alpha_{2,3} + \alpha_{3,2}$
  $c_3 \leftarrow \alpha_{3,3} + r_3 + r_2 + r_0 + \alpha_{0,1} + \alpha_{1,0}$
```

involving strictly more than three terms, (c, c'), where c and c' are part of c_i, (t', t'') with t' part of $t_{i,j}$ and t'' part of $t_{k,j}$, where t'' is assigned before t', both involving more than three terms, and finally, any other couple involving a part of c_k.

- the cases $(r_{i,j}, t')$ and (t', t'') imply the involvement of $r_{k,j}$. Thanks to Observation (i), all possible cases can be exhausted, and we obtain $i, k, j-1, j \in I$ and $i, k, j-1, j \in J$.
- the case (c, c') is particular. Indeed, we can assume that c is computed during the computation of c'. We can hence safely assign t to a random variable if this is the only case where $r_{i,j}$ and r_{j-1} have been involved.
- the query of a c, part of c_k and involving r_{j-1} involves the variable $r_{k,j}$. From Observation (i), we can exhaust the possible cases. For each of these cases except five, we have $i, j, j-1, k \in I$ and $i, j, j-1, k \in J$. The five remaining cases are $(c_j, c_j), (c_j, c_k), (r_{k,j}, c_k), (c_i, c_k), (t_{i,j}, c_k)$. With the case involving c_j, by construction we have that $r_{k,j}$ and $r_{i,j}$ appear after the addition of all the terms of the form $t_{j,\ell}$. Consequently, this expression involves the term $r_{j-1,j}$ (if $i = j-1$, $t_{i,j}$ does not exist, and if $k = j-1$, the probe of c_k assures that we have $j-1$ in I and J, hence we also get $i, j \in I$ and $i, j \in J$). Using Observation (i), we find out that the only way not to have $i, j, j-1 \in I$ and $i, j, j-1 \in I$ is to make another probe to c_j. However, this case is similar to the one we just observed: it is safe to randomly assign t. For any another case, the random $r_{k,j}$ reappears, and we must hence query another variable to get rid of it. The only possibility is to query c_k once more. Hence t can be randomly assigned.

□

4.6 Optimal Small Cases

We propose three secure compression algorithms using less random bits than the generic solution given by ISW and than our new solution for the specific small orders $d = 2$, 3 and 4. These algorithms actually use only the optimal numbers of random bits for these small quantity of probes, as proven in Section 4.4. Furthermore, since they all are dedicated to a specific order d (among 2, 3, and 4), we got use of the verifier proposed by Barthe et al. in [BBD+15b] to formally prove their correctness and their d-privacy.

Proposition 4.6.1. *Algorithms 5, 6, and 7 are correct and respectively* $2, 3$ *and* 4*-private.*

Algorithm 7 Fourth-Order Compression Algorithm

```
Require: sharing $(\alpha_{i,j})_{0\leq i,j\leq 4}$
Ensure: sharing $(c_i)_{0\leq i\leq 4}$
  $r_0 \overset{\$}{\leftarrow} \mathbb{F}_2$;   $r_1 \overset{\$}{\leftarrow} \mathbb{F}_2$;   $r_2 \overset{\$}{\leftarrow} \mathbb{F}_2$;   $r_3 \overset{\$}{\leftarrow} \mathbb{F}_2$;   $r_4 \overset{\$}{\leftarrow} \mathbb{F}_2$
  $c_0 \leftarrow \alpha_{0,0} + r_0 + \alpha_{0,1} + \alpha_{1,0} + r_1 + \alpha_{0,2} + \alpha_{2,0}$
  $c_1 \leftarrow \alpha_{1,1} + r_1 + \alpha_{1,2} + \alpha_{2,1} + r_2 + \alpha_{1,3} + \alpha_{3,1}$
  $c_2 \leftarrow \alpha_{2,2} + r_2 + \alpha_{2,3} + \alpha_{3,2} + r_3 + \alpha_{2,4} + \alpha_{4,2}$
  $c_3 \leftarrow \alpha_{3,3} + r_3 + \alpha_{3,4} + \alpha_{4,3} + r_4 + \alpha_{3,0} + \alpha_{0,3}$
  $c_4 \leftarrow \alpha_{4,4} + r_4 + \alpha_{4,0} + \alpha_{0,4} + r_0 + \alpha_{4,1} + \alpha_{1,4}$
```

Table 4.1 (Section 4.5) compares the amount of randomness used by the new construction proposed in Section 4.5 and by our optimal small algorithms. We recall that each of them attains the lower bound proved in Section 4.4.

4.7 Composition

Our new algorithms are all d-private, when applied on the outputs of a multiplicative encoder parameterized at order d. We now aim to show how they can be involved in the design of larger functions (e.g., block ciphers) to achieve a global d-privacy. In [BBD+15a], Barthe et al. introduce and formally prove a method to compose small d-private algorithms (a.k.a., *gadgets*) into d-private larger functions. The idea is to carefully refresh the sharings when necessary, according to the security properties of the gadgets. Before going further into the details of this composition, we recall some security properties used in [BBD+15a].

4.7.1 Compositional Security Notions

Before stating the new security definitions, we first need to introduce the notion of simulatability. For the sake of simplicity, we only state this notion for multiplication algorithm, but this can easily be extended to more general algorithms.

Definition 4.7.1. *A set $P = \{p_1, \ldots, p_\ell\}$ of ℓ probes of a multiplication algorithm can be* simulated *with at most t shares of each input, if there exists two sets $I = \{i_1, \ldots, i_t\}$ and $J = \{j_1, \ldots, j_t\}$ of t indices from $\{0, \ldots, d\}$ and a random function f taking as input $2t$ bits and outputting ℓ bits such that for any fixed bits $(a_i)_{0\leq i\leq d}$ and $(b_j)_{0\leq j\leq d}$, the distributions $\{p_1, \ldots, p_\ell\}$ (which implicitly depends on $(a_i)_{0\leq i\leq d}$, $(b_j)_{0\leq j\leq d}$, and the random bits used in the multiplication algorithm) and $\{f(a_{i_1}, \ldots, a_{i_t}, b_{j_1}, \ldots, b_{j_t})\}$ are identical.*

We write $f(a_{i_1}, \ldots, a_{i_t}, b_{j_1}, \ldots, b_{j_t}) = f(a_I, b_J)$.

Definition 4.7.2. *An algorithm is d-*non-interferent (or d-NI) *if and only if every set of at most d probes can be simulated with at most d shares of each input.*

While this notion might be stronger than the notion of security we used, all our concrete constructions in Sections 4.5 and 4.6 satisfy it. The proof of Algorithm 4 is indeed a proof by simulation, while the small cases in Section 4.6 are proven using the verifier by Barthe et al. in [BBD+15b], which directly proves NI.

Definition 4.7.3. *An algorithm is d-*tight non-interferent (or d-TNI) *if and only if every set of $t \leqslant d$ probes can be simulated with at most t shares of each input.*

While this notion of d-tight non-interference was assumed to be stronger than the notion of d-non-interference in [BBD+15a], we show hereafter that these two security notions are actually equivalent. In particular, this means that all our concrete constructions are also TNI.

Proposition 4.7.4. ***(d-NI ⇔ d-TNI)*** *An algorithm is d-non-interferent if and only if it is d-tight non-interferent.*

Proof. The right-to-left implication is straightforward from the definitions. Let us thus consider the left-to-right direction.

For that purpose, we first need to introduce a technical lemma. Again, for the sake of simplicity, we only consider multiplication algorithm, with only two inputs, but the proof can easily be generalized to any algorithm.

Lemma 4.7.5. *Let $P = \{p_1, \ldots, p_\ell\}$ be a set of ℓ probes which can be simulated by the sets (I, J) and also by the sets (I', J'). Then it can also be simulated by $(I \cap I', J \cap J')$.*

Proof. Let f the function corresponding to I, J and f' the function corresponding to I', J'. We have that for any bits $(a_i)_{0 \leq i \leq d}$ and $(b_j)_{0 \leq j \leq d}$, the distributions $\{p_1, \ldots, p_\ell\}$, $\{f(a_I, b_J)\}$, and $\{f'(a_{I'}, b_{J'})\}$ are identical. Therefore, f does not depend on a_i nor b_j for $i \in I \setminus I'$ and $j \in J \setminus J'$, since f' does not depend on them. Thus, P can be simulated by only shares from $I \cap I', J \cap J'$ (using the function f where the inputs corresponding to a_i and b_j for $i \in I \setminus I'$ and $j \in J \setminus J'$ are just set to zero, for example). □

We now assume that an algorithm $\mathcal{A}$ is d-NI, that is, every set of at most d probes can be simulated with at most d shares of each input. Now, by contradiction, let us consider a set P with minimal cardinality $t < d$ of probes on $\mathcal{A}$, such that it cannot be simulated by at most t shares of each input. Let us consider the sets I, J corresponding to the intersection of all sets I', J' (respectively) such that the set P can be simulated by I', J'. The sets I, J also simulate P thanks to Lemma 4.7.5. Furthermore, by hypothesis, $t < |I| \leq d$ or $t < |J| \leq d$. Without loss of generality, let us suppose that $|I| > t$.

Let i^* be an arbitrary element of $\{0, \ldots, d\} \setminus I$ (which is not an empty set as $|I| \leq d$). Let us now consider the set of probes $P' = P \cup \{a_{i^*}\}$. By hypothesis, P' can be simulated by at most $|P'| = t + 1$ shares of each input. Let I', J' two sets of size at most $t + 1$ simulating P'. These two sets also simulate $P \subseteq P'$, therefore, $I \cap I', J \cap J'$ also simulate P. Furthermore, $i^* \in I$, as all the shares a_i are independent. Since $i^* \notin I$, $|I \cap I'| \leq t$ and $I \cap I' \subsetneq I$, which contradicts the fact that I and J were the intersection of all sets I'', J'' simulating P. □

Definition 4.7.6. *An algorithm $\mathcal{A}$ is d-*strong non-interferent (or d-SNI) *if and only if for every set $\mathcal{I}$ of t_1 probes on intermediate variables (i.e., no output wires or shares) and every set $\mathcal{O}$ of t_2 probes on output shares such that $t_1 + t_2 \leqslant d$, the set $\mathcal{I} \cup \mathcal{O}$ of probes can be simulated by only t_1 shares of each input.*

The composition of two d-SNI algorithms is itself d-SNI, while that of d-TNI algorithms is not necessarily d-TNI. This implies that d-SNI gadgets can be directly composed while maintaining the d-privacy property, whereas a so-called *refreshing* gadget must sometimes be involved before the composition of d-TNI algorithms. Since the latter refreshing gadgets consume the same quantity of random values as ISW, limiting their use is crucial if the goal is to reduce the global amount of randomness.

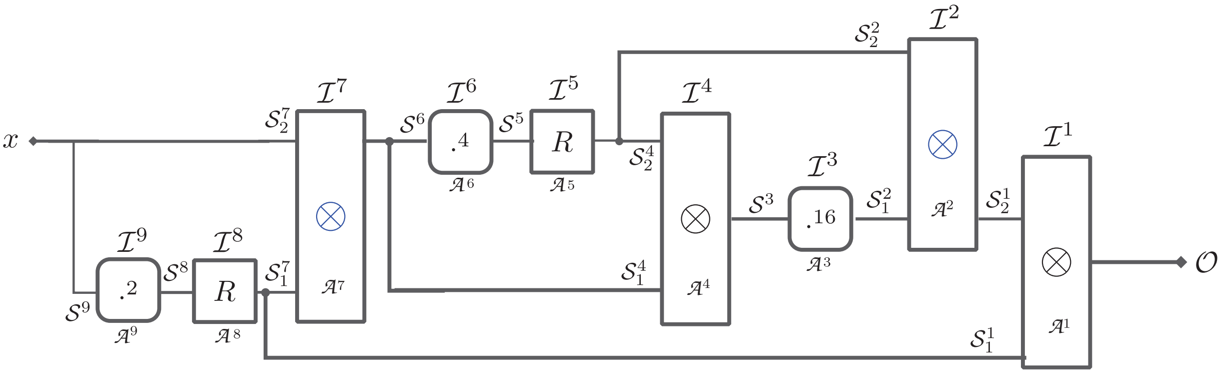


Figure 4.6: AES $\cdot^{254}$

4.7.2 Building Compositions with our New Algorithms

In [BBD+15a], the authors show that the ISW multiplication is d-SNI and use it to build secure compositions. Unfortunately, our new multiplication algorithms are d-TNI but not d-SNI. Therefore, as discussed in the previous section, they can replace only some of the ISW multiplications in secure compositions. Let us take the example of the AES inversion that is depicted in [BBD+15a]. We can prove that replacing the first ($\mathcal{A}^7$) and the third ($\mathcal{A}^2$) ISW multiplications by d-TNI multiplications (e.g., our new constructions) and moving the refreshing algorithm R in different locations preserves the strong non-interference of the inversion, while benefiting from our reduction of the randomness consumption.

The tweaked inversion is given in Figure 4.6. $\otimes$ denotes the d-SNI ISW multiplication, $\cdot^{\alpha}$ denotes the exponentiation to the power α, $\mathcal{A}^i$ refers to the i-th algorithm or gadget (indexed from left to right), R denotes the d-SNI refreshing gadget, $\mathcal{I}^i$ denotes the set of internal probes in the i-th algorithm, $\mathcal{S}_j^i$ denotes the set of shares from the j inputs of algorithm $\mathcal{A}^i$ used to simulate all further probes. Finally, x denotes the inversion input and $\mathcal{O}$ denotes the set of probes at the output of the inversion. The global constraint for the inversion to be d-SNI (and thus itself composable) is that: $|\mathcal{S}^8 \cup \mathcal{S}^9| \leqslant \sum_{1\leqslant i\leqslant 9} |\mathcal{I}^i|$, i.e., all the internal probes can be perfectly simulated with at most $\sum_{1\leqslant i\leqslant 9} |\mathcal{I}^i|$ shares of x.

Proposition 4.7.7. *The AES inversion given in Figure 4.6 with $\mathcal{A}^1$ and $\mathcal{A}^4$ being d-SNI multiplications and $\mathcal{A}^2$ and $\mathcal{A}^7$ being d-TNI multiplications is d-SNI.*

Proof. From the d-probing model, we assume that the total number of probes used to attack the inversion is limited to d, that is $\sum_{1\leqslant i\leqslant 9} |\mathcal{I}^i| + |\mathcal{O}| \leqslant d$. As in [BBD+15a], we build the proof from right to left by simulating each algorithm. Algorithm $\mathcal{A}^1$ is d-SNI, thus $|\mathcal{S}_1^1|, |\mathcal{S}_2^1| \leqslant |\mathcal{I}^1|$. Algorithm $\mathcal{A}^2$ is d-TNI, thus $|\mathcal{S}_1^2|, |\mathcal{S}_2^2| \leqslant |\mathcal{I}^1 + \mathcal{I}^2|$. As explained in [BBD+15a], since Algorithm $\mathcal{A}^3$ is affine, then $|\mathcal{S}^3| \leqslant |\mathcal{S}_1^2 + \mathcal{I}^3| \leqslant |\mathcal{I}^1 + \mathcal{I}^2 + \mathcal{I}^3|$. Algorithm $\mathcal{A}^4$ is d-SNI, thus $|\mathcal{S}_1^4|, |\mathcal{S}_2^4| \leqslant |\mathcal{I}^4|$. Algorithm $\mathcal{A}^5$ is d-SNI, thus $|\mathcal{S}^5| \leqslant |\mathcal{I}^5|$. Algorithm $\mathcal{A}^6$ is affine, thus $|\mathcal{S}^6| \leqslant |\mathcal{S}^5 + \mathcal{I}^6| \leqslant |\mathcal{I}^5 + \mathcal{I}^6|$. Algorithm $\mathcal{A}^7$ is d-TNI, thus $|\mathcal{S}_1^7|, |\mathcal{S}_2^7| \leqslant |\mathcal{S}^6 + \mathcal{S}_1^4 + \mathcal{I}^7| \leqslant |\mathcal{I}^4 + \mathcal{I}^5 + \mathcal{I}^6 + \mathcal{I}^7|$. Algorithm $\mathcal{A}^8$ is d-SNI, thus $|\mathcal{S}^8| \leqslant |\mathcal{I}^8|$. Algorithm $\mathcal{A}^9$ is affine, thus $|\mathcal{S}^9| \leqslant |\mathcal{I}^9 + \mathcal{S}^8| \leqslant |\mathcal{I}^8 + \mathcal{I}^9|$. Finally, all the probes of this inversion can be perfectly simulated from $|\mathcal{S}^9 \cup \mathcal{S}_1^7| \leqslant |\mathcal{I}^4 + \mathcal{I}^5 + \mathcal{I}^6 + \mathcal{I}^7 + \mathcal{I}^8 + \mathcal{I}^9|$ shares of x, which proves that the inversion is still d-SNI. □

From Proposition 4.7.7, our new constructions can be used to build d-SNI algorithms. In the case of the AES block cipher, half of the d-SNI ISW multiplications can be replaced by ours while preserving the whole d-SNI security.

4.8 New Automatic Tool for Finding Attacks

In this section, we describe a new automatic tool for finding attacks on compression algorithms for multiplication which is developed in Sage (Python) [Sage]. Compared to the verifier developed by Barthe *et al.* [BBD+15b] and based on Easycrypt, to find attacks in practice, our tool is not as generic as it focuses on compression algorithms for multiplication and its soundness is not perfect (and relies on some heuristic assumption). Nevertheless, it is order of magnitudes faster.

A non-perfect soundness means that the algorithm may not find an attack and can only guarantee that there does not exist an attack except with probability ε. We believe that, in practice, this limitation is not a big issue as if ε is small enough (e.g., 2^{-20}), a software bug is much more likely than an attack on the scheme. Furthermore, the running time of the algorithm depends only linearly on $\log(1/\varepsilon)$. Concretely, for all the schemes we manually tested for $d = 3, 4, 5$ and 6, attacks on invalid schemes were found almost immediately. If not used to formally prove schemes, our tool can at least be used to quickly eliminate (most) incorrect schemes, and enables to focus efforts on trying to prove "non-trivially-broken" schemes.

4.8.1 Algorithm of the Tool

From Theorem 4.3.1, in order to find an attack $P = \{p_1, \ldots, p_\ell\}$ with $\ell \leq d$, we just need to find a set $P = \{p_1, \ldots, p_\ell\}$ satisfying Condition 1. If no such set P exists, the compression algorithm for multiplication is d-private.

A naive way to check the existence of such a set P is to enumerate all the sets of d probes. However, there are $\binom{N}{d}$ such sets, with N being the number of intermediate variables of the algorithm. For instance, to achieve 4-privacy, our construction (see Section 4.6) uses $N = 81$ intermediate variables, which makes more than 2^{20} sets of four variables to test. In [BBD+15b], the authors proposed a faster way of enumerating these sets by considering larger sets which are still independent from the secret. However, their method falls short for the compression algorithms proposed here as soon as $d > 6$, as shown in Section 4.8.4. Furthermore even for $d = 3, 4, 5$, their tool takes several minutes to prove security (around 5 minutes to check security of Algorithm 4 with $d = 5$) or to find an attack for incorrect schemes, which prevent people from quickly checking the validity of a newly designed scheme.

To counteract this issue, we design a new tool which is completely different and which borrows ideas from coding theory to enumerate the sets of d or less intermediate variables. Let $\gamma_1, \ldots, \gamma_\nu$ be all the intermediate results whose expression functionally depends on at least one random and $\gamma'_1, \ldots, \gamma'_{\nu'}$ be the other intermediate results that we refer to as deterministic intermediate results ($\nu + \nu' = N$). We remark that all the $\alpha_{i,j} = a_i b_j$ are intermediate results and that no intermediate result can functionally depend on more than one shares' product $\alpha_{i,j} = a_i b_j$ without also depending on a random bit. Otherwise, the compression algorithm would not be d-private, according to Lemma 4.5.1. As this condition can be easily tested, we now assume that the only deterministic intermediate results are the $\alpha_{i,j} = a_i b_j$ that we refer to as γ'_k in the following. As an example, intermediate results of Algorithm 5 are depicted in Table 4.2.

An attack set $P = \{p_1, \ldots, p_\ell\}$ can then be separated into two sets $Q = \{\gamma_{i_1}, \ldots, \gamma_{i_\delta}\}$ and $Q' = \{\gamma_{i'_1}, \ldots, \gamma_{i'_{\delta'}}\}$, with $\ell = \delta + \delta' \leq d$. We remark that necessarily $\sum_{p \in Q} p$ does not functionally depend on any random value. Actually, we even have the following lemma:

Lemma 4.8.1. *Let $\mathcal{A}(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{r})$ be a compression algorithm for multiplication. Then $\mathcal{A}$ is d-private if and only if there does not exist a set of non-deterministic probes $Q = \{\gamma_{i_1}, \ldots, \gamma_{i_\delta}\}$ with $\delta \leqslant d$ such that $\sum_{p \in Q} p = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ where the column space or the row space of $\boldsymbol{M}$ contains a vector of Hamming weight at least $\delta + 1$.*

Table 4.2: Intermediate results of Algorithm 5

non-deterministic ($\nu = 12$)		deterministic ($\nu' = 9$)	
$\gamma_1 = a_0b_0 + r_0$	$\gamma_7 = c_1$	$\gamma'_1 = a_0b_0$	$\gamma'_6 = a_1b_0$
$\gamma_2 = a_0b_0 + r_0 + a_0b_2$	$\gamma_8 = r_1$	$\gamma'_2 = a_0b_2$	$\gamma'_7 = a_2b_2$
$\gamma_3 = c_0$	$\gamma_9 = a_2b_2 + r_1$	$\gamma'_3 = a_2b_0$	$\gamma'_8 = a_1b_2$
$\gamma_4 = r_0$	$\gamma_{10} = a_2b_2 + r_1 + r_0$	$\gamma'_4 = a_1b_1$	$\gamma'_9 = a_2b_1$
$\gamma_5 = a_1b_1 + r_1$	$\gamma_{11} = a_2b_2 + r_1 + r_0 + a_1b_2$	$\gamma'_5 = a_0b_1$	
$\gamma_6 = a_1b_1 + r_1 + a_0b_1$	$\gamma_{12} = c_2$		

Furthermore, if such a set Q exists, there exists a set $\{\gamma_{i'_1}, \dots, \gamma_{i'_{\delta'}}\}$, with $\delta + \delta' \leq d$, such that $P = Q \cup Q'$ is an attack.

Moreover, the lemma is still true when we restrict ourselves to sets Q such that there exists no proper subset $\hat{Q} \subsetneq Q$ such that $\sum_{p \in \hat{Q}} p$ does not functionally depend on any random.

Proof. The two first paragraphs of the lemma can be proven similarly to Lemma 4.5.1. Thus, we only need to prove its last part.

By contradiction, let us suppose that there exists a set Q of non-deterministic probes $Q = \{\gamma_{i_1}, \dots, \gamma_{i_\delta}\}$ such that $\sum_{p \in Q} p = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ and the column space (without loss of generality, by symmetry of the a_i's and b_i's) of $\boldsymbol{M}$ contains a vector of Hamming weight at least $\delta + 1$, but such that any subset $\hat{Q} \subsetneq Q$ where $\sum_{p \in \hat{Q}} p$ that does not functionally depend on any random. Consequently, the sum $\sum_{p \in \hat{Q}} p = \boldsymbol{a}^\intercal \cdot \hat{\boldsymbol{M}} \cdot \boldsymbol{b}$, is such that the column space (still without loss of generality) of $\hat{\boldsymbol{M}}$ does not contain any vector of Hamming weight at least $|\hat{Q}| + 1$.

First, let us set $\bar{\boldsymbol{M}} = \hat{\boldsymbol{M}} + \boldsymbol{M}$ (over $\mathbb{F}_2$), so $\sum_{p \in Q \setminus \hat{Q}} p = \boldsymbol{a}^\intercal \cdot \bar{\boldsymbol{M}} \cdot \boldsymbol{b}$, as $\sum_{p \in \hat{Q}} p + \sum_{p \in Q \setminus \hat{Q}} = \sum_{p \in Q} p = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b}$ and let $\hat{\delta} = |\hat{Q}|$ and $\bar{\delta} = |Q \setminus \hat{Q}| = \delta - \hat{\delta}$. Let also ω, $\hat{\omega}$, and $\bar{\omega}$ be the maximum Hamming weights of the vectors in the column space of $\boldsymbol{M}$, $\hat{\boldsymbol{M}}$, and $\bar{\boldsymbol{M}}$, respectively. Since $\boldsymbol{M} = \hat{\boldsymbol{M}} + \bar{\boldsymbol{M}}$, then $\omega \leq \hat{\omega} + \bar{\omega}$ and since $\omega > \delta + 1$, and $\delta = \hat{\delta} + \bar{\delta}$, then $\hat{\omega} > \hat{\delta}$ or $\bar{\omega} > \bar{\delta}$. We set $\tilde{Q} = \hat{Q}$ if $\hat{\omega} > \hat{\delta}$, and $\tilde{Q} = Q \setminus \hat{Q}$ otherwise. According to the definitions of $\hat{\delta}$ and $\bar{\omega}$, we have that $\tilde{Q} \subsetneq Q$ is such that $\sum_{p \in Q} p = \boldsymbol{a}^\intercal \cdot \tilde{\boldsymbol{M}} \cdot \boldsymbol{b}$ where the column space of $\tilde{\boldsymbol{M}}$ contains a vector of Hamming weight at least $|\tilde{Q}| + 1$. This contradicts the definition of Q and concludes the proof of the lemma. □

To quickly enumerate all the possible attacks, we first enumerate the sets $Q = \{\gamma_{i_1}, \dots, \gamma_{i_\delta}\}$ of size $\delta \leq d$ such that $\sum_{p \in Q} p$ does not functionally depend on any random bit (and no proper subset of $\hat{Q} \subsetneq Q$ is such that $\sum_{p \in \hat{Q}} p$ does not functionally depend on any random bit), using *information set decoding*, recalled in the next section. Then, for each possible set Q, we check if the column space or the row space of $\boldsymbol{M}$ (as defined in the previous lemma) contains a vector of Hamming weight at least $\delta + 1$. A naive approach would consist in enumerating all the vectors in the row space and the column space of $\boldsymbol{M}$. Our tool however uses the two following facts to perform this test very quickly in most cases:

- when $\boldsymbol{M}$ contains at most δ non-zero rows and at most δ non-zero columns, Q does not yield an attack;
- when $\boldsymbol{M}$ contains exactly $\delta + 1$ non-zero rows (resp. columns), that we assume to be the first $\delta + 1$ (without loss of generality), Q yields an attack if and only if the vector $(\boldsymbol{u}_{\delta+1}^\intercal, \boldsymbol{0}_{d-\delta}^\intercal)$ is in the row space (resp. $(\boldsymbol{u}_{\delta+1}, \boldsymbol{0}_{d-\delta})$ is in the column space) of $\boldsymbol{M}$ (this condition can be checked in polynomial time in d).

4.8.2 Information Set Decoding and Error Probability

We now explain how to perform the enumeration step of our algorithm using information set decoding. Information set decoding was introduced in the original security analysis of the McEliece cryptosystem in [Pra62; McE78] as a way to break the McEliece cryptosystem by finding small code words in a random linear code. It was further explored by Lee and Brickell in [LB88]. We should point out that since then, many improvements were proposed, e.g., in [Leo88; Ste88]. However, for the sake of simplicity and because it already gives very good results, we use the original information set decoding algorithm. Furthermore, it is not clear that the aforementioned improvements also apply in our case, as the codes we consider are far from the Singleton bound.

We assume that random bits are denoted $r_1, \ldots, r_R$. For each intermediate γ_k containing some random bit, we associate the vector $\boldsymbol{\tau} \in \mathbb{Z}_2^R$, where $\tau_i = 1$ if and only if γ_k functionally depends on the random bit r_i. We then consider the matrix $\Gamma \in \mathbb{Z}_2^{R \times \nu}$ whose k-th column is $\boldsymbol{\tau}$. For instance, for Algorithm 5, we have:

$$\Gamma = \begin{array}{c} \\ \\ \end{array}\begin{array}{cccccccccccc} \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & \gamma_9 & \gamma_{10} & \gamma_{11} & \gamma_{12} \end{array}$$

$$\Gamma = \left(\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}\right) \begin{array}{c} r_0 \\ r_1 \end{array} .$$

For every $\delta \leq d$, enumerating the sets $Q = \{\gamma_{i_1}, \ldots, \gamma_{i_\delta}\}$, such that $\sum_{p \in Q} p$ does not functionally depend on any random, consists in enumerating the vectors $\boldsymbol{x}$ of Hamming weight δ such that $\Gamma \cdot \boldsymbol{x} = \mathbf{0}$ (specifically, $\{i_1, \ldots, i_\delta\}$ are the coordinates of the non-zero components of $\boldsymbol{x}$). Furthermore, we can restrict ourselves to vector $\boldsymbol{x}$ such that no vector $\hat{\boldsymbol{x}} < \boldsymbol{x}$ satisfies $\Gamma \cdot \hat{\boldsymbol{x}} = \mathbf{0}$ (where $\hat{\boldsymbol{x}} < \boldsymbol{x}$ means that $\hat{\boldsymbol{x}} \neq \boldsymbol{x}$ and for any $1 \leq i \leq \nu$, if $x_i = 0$ then $\hat{x}_i = 0$), since we can restrict ourselves to sets Q such that no proper subset $\hat{Q} \subsetneq Q$ is such that $\sum_{p \in \hat{Q}} p$ does not functionally depend on any random bit. This is close to the problem of finding code words $\boldsymbol{x}$ of small Hamming weight for the linear code of parity matrix Γ and we show this can be solved using information set decoding.

The basic idea is the following one. We first apply a row-reduction to Γ. Let us call the resulting matrix Γ'. We remark that, for any vector $\boldsymbol{x}$, $\Gamma \cdot \boldsymbol{x} = 0$ if and only if $\Gamma' \cdot \boldsymbol{x} = 0$ and thus we can use Γ' instead of Γ in our problem. We assume in a first time that the first R columns of Γ are linearly independent (recall that the number ν of columns of Γ is much larger than its number R of rows), so that the R first columns of Γ' forms an identity matrix. Then, for any $k^* > R$, if the k^*-th column of Γ' has Hamming weight at most $d - 1$, we can consider the vector $\boldsymbol{x}$ defined as $x_{k^*} = 1$, $x_k = 1$ when $\Gamma'_{k,k^*} = 1$, and $x_k = 0$ otherwise; and this vector satisfies the conditions we were looking for: its Hamming weight is at most d and $\Gamma' \cdot \boldsymbol{x} = 0$. That way, we have quickly enumerated all the vectors $\boldsymbol{x}$ of Hamming weight at most d such that $\Gamma' \cdot \boldsymbol{x} = 0$ and with the additional property that $x_k = 0$ for all $k > R$ except for at most[3] one index k^*. Without the condition $\Gamma' \cdot \boldsymbol{x} = 0$, there are $(\nu - R + 1) \cdot \sum_{i=0}^{d-1} \binom{R}{i} + \binom{R}{d}$ such vectors, as there are $\sum_{i=0}^{d} \binom{R}{i}$ vectors $\boldsymbol{x}$ such that $\mathtt{HW}(\boldsymbol{x}) \leq d$ and $x_k = 0$ for every $k > R$, and there are $(\nu - R) \cdot \sum_{i=0}^{d-1} \binom{R}{i}$ vectors $\boldsymbol{x}$ such that $\mathtt{HW}(\boldsymbol{x}) \leq d$ and $x_k = 1$, for a single $k > R$.

In other words, using row-reduction, we have been able to check $(\nu - R + 1) \cdot \sum_{i=0}^{d-1} \binom{R}{i} + \binom{R}{d}$ possible vectors $\boldsymbol{x}$ among at most $\sum_{i=1}^{d} \binom{\nu}{i}$ vectors which could be used to mount an attack, by testing at most $\nu - R$ vectors.[4]

[3] We have seen that for one index k^*, but it is easy to see that, as the first R columns of Γ' form an identity matrix, there does not exist such vector $\boldsymbol{x}$ so that $x_k = 0$ for all $k > R$ anyway.

[4] There are exactly $\sum_{i=1}^{d} \binom{\nu}{i}$ vectors of Hamming weight at most d, but here we recall that we only consider vectors $\boldsymbol{x}$ satisfying the following additional condition: there is no vector $\hat{\boldsymbol{x}} < \boldsymbol{x}$ such that $\Gamma \cdot \hat{\boldsymbol{x}} = \mathbf{0}$. We also remark that the vectors $\boldsymbol{x}$ generated by the described algorithm all satisfy this additional condition.

Table 4.3: Complexities of exhibiting an attack at several orders

		Time to find an attack	
Order	Target Algorithm	Verifier [BBD+15b]	New tool
$d = 2$	tweaked Algorithm 5	less than 1 ms	less than 10 ms
$d = 3$	tweaked Algorithm 6	36 ms	less than 10 ms
$d = 4$	tweaked Algorithm 7	108 ms	less than 10 ms
$d = 5$	tweaked Algorithm 4	6.264 s	less than 100 ms
$d = 6$	tweaked Algorithm 4	26 min	less than 300 ms

Then, we can randomly permute the columns of Γ and repeat this algorithm. Each iteration would find an attack (if there was one attack) with probability at least $\left((\nu - R + 1) \cdot \sum_{i=0}^{d-1} \binom{R}{i} + \binom{R}{d}\right) / \sum_{i=1}^{d} \binom{\nu}{i}$. Therefore, after K iterations, the error probability is only

$$\varepsilon \leq \left(1 - \frac{(\nu - R + 1) \cdot \sum_{i=0}^{d-1} \binom{R}{i} + \binom{R}{d}}{\sum_{i=1}^{d} \binom{\nu}{i}}\right)^{K},$$

and the required number of iterations is linear with $\log(1/\varepsilon)$, which is what we wanted.

Now, we just need to handle the case when the first R columns of Γ are not linearly independent, for some permuted matrix Γ at some iteration. We can simply redraw the permutation or taking the pivots in the row-reduction instead of taking the first R columns of Γ. In both cases, this may slightly bias the probability. We make the *heuristic assumption* that the bias is negligible. To support this heuristic assumption, we remark that if we iterate the algorithm for all the permutations for which the first R columns of Γ are not linearly independent, then we would enumerate all the vectors $\boldsymbol{x}$ we are interested in, thanks to the additional condition that there is no vector $\hat{\boldsymbol{x}} < \boldsymbol{x}$ such that $\Gamma \cdot \hat{\boldsymbol{x}} = \mathbf{0}$.

4.8.3 The Tool

The tool takes as input a description of a compression algorithm for multiplication similar to the ones we used in this chapter (see Figure 4.2 for instance) and the maximum error probability ε we allow, and tries to find an attack. If no attack is found, then the scheme is secure with probability $1 - \varepsilon$. The tool can also output a description of the scheme which can be fed off into the tool in [BBD+15b].

4.8.4 Complexity Comparison

It is difficult to compare the complexity of our new tool to the complexity of the tool proposed in [BBD+15b] since it strongly depends on the tested algorithm. Nevertheless, we try to give some values for the verification time of both tools when we intentionally modify our constructions to yield an attack. From order 2 to 4, we start with our optimal constructions and we just invert two random bits in an output share c_i. Similarly, for orders 5 and 6, we use our generic construction and apply the same small modification. The computations were performed on a Intel(R) Core(TM) i5-2467M CPU @ 1.60GHz and the results are given in Table 4.3. We can see that in all the considered cases, our new tool reveals the attack in less than 300 ms while the generic verifier of Barthe et al. needs up to 26 minutes for order $d = 6$.

Chapter 5

Private Veto with Constant Randomness

Quando você diz sim para outros, certifique-se de não estar dizendo não a si mesmo.
Paulo Coelho

In this chapter, we formalize and study the problem of *private veto* protocols. A private veto protocol allows a set of players to test whether there is general agreement on a given decision while keeping each player's choice secret. We consider the problem of n honest-but-curious players with private bits $x_1, \ldots, x_n$ who wish to compute their `AND` value $x_1 \wedge \cdots \wedge x_n$ in such a way that, after the execution of the protocol, no player obtains any information about the inputs of the remaining players (other than what can be deduced from their own inputs and the `AND` value). This problem can be seen as the counterpart of the classical problem of the secure `XOR` evaluation. The study of private veto protocols can then be seen a first step towards the secure evaluation of any Boolean function, through its expression in an algebraic normal form. In distributed protocols, it is natural to consider randomness as a resource like space and time and we here study the amount of randomness needed in such private protocols. However, the problem of the randomness complexity of the secure evaluation of private veto protocols was never tackled specifically. We propose in this chapter a protocol that requires as few as 6 random bits for three players and we present a protocol for all $n \in \mathbb{N}$ with less than 26 random bits. In the particular case where n belongs to the infinite geometric progression of integers $\{3 \cdot 2^i, i \in \mathbb{N}\}$, our proposal uses only 12 random bits. The conception of our protocols is modular which allows us to present the general solution in a recursive way (with simplified correctness and privacy proofs).

5.1 Introduction

A *private veto* protocol should only reveal whether there is consensus, rather than an individual count of the number of agreeing and disagreeing players. We propose several concrete constructions for this problem and prove their security in the information-theoretic security model. We notably propose a protocol that requires as few as 6 random bits for three players and a generic protocol that constantly uses 12 random bits for an unbounded number of players (of a particular form).

5.1.1 Previous work

Secure distributed computation is the problem of cooperative evaluation of a function such that the output becomes commonly known while the inputs remain secret. More precisely, a private multi-party protocol allows a set of players to compute a function of the players' private inputs in such a way that each player learns the result but no player learns anything about another's private input.

The basic problem of designing private protocols for arbitrary functions was first solved for two players by Yao [Yao86], and for any number of players by Goldreich *et al.* [GMW87]. These solutions rely on unproven computational assumptions and Ben-Or *et al.* [BOGW88], and Chaum *et al.* [CCD88] later removed the reliance on such assumptions. In these latter works, no assumption is made about the computing power of the players and security proofs rely on information-theoretic arguments. In this setting, an adversary (passive or active) may control some corrupted players but the communication channels are assumed to be private (*i.e.* players are connected *via* ideal point-to-point authenticated and encrypted channels).

In the case of *passive* adversaries (*i.e.* when the corrupted parties are honest-but-curious) the protocols proposed in [BOGW88; CCD88] are secure against adversaries corrupting up to $t < n/2$ players. In the case of *active* adversaries, where the corrupted players may arbitrarily deviate from the protocol, the threshold is reduced from $n/2$ to $n/3$. In this chapter, we consider only the security in the information-theoretic setting against a passive adversary and the *t-privacy* property means that any passive coalition of at most t players cannot learn anything from the execution of the protocol (except what is implied by the result and its own private input). We define formally this notion of information-theoretical privacy in Section 5.2. In this honest-but-curious model enforcing the correctness constraint is usually easy, but enforcing the privacy constraint is hard.

As argued in Section 2.6, randomness plays a crucial role in several areas of computer science, including probabilistic algorithms, cryptography, and distributed computing. Random bits are hard to produce and devices that generate them, such as Geiger counters and Zener diodes are slow. It can be readily seen that, except for degenerate cases, randomness is essential to maintain privacy in multi-party computation. In order to compute a function $f : (\mathbb{Z}_p)^n \to \mathbb{Z}_p$ (with p a rational prime number) of n players' private inputs (with $n < p$), while satisfying the t-privacy, the protocols from [BOGW88; CCD88] require $\Omega(ntm\log(p))$ random bits, where m denotes the number of multiplication gates in the arithmetic circuit representing the function f. Since then, considerable effort has been devoted to reduce the number of random bits used by secure multi-party protocols.

Explicit results on the so-called *randomness complexity* of secure distributed computation are known only for the computation of the specific function $\texttt{XOR-n} : \{0,1\}^n \to \{0,1\}, (x_1, \dots, x_n) \mapsto x_1 \oplus \cdots \oplus x_n$. It can be easily seen that one random bit is necessary and sufficient to compute the XOR-n function in a 1-private way (using n rounds of communication). In [KM96], Kushilevitz and Mansour proved that $\Omega(t)$ bits are necessary, and $O(t^2 \log(n/t))$ are sufficient, for the t-private computation of the XOR-n function. A lower bound $\Omega(\log n)$ was later given by Gál and Rosén in [GR03] for any $t \geqslant 2$ (proving that the upper bound from [KM96] is tight, up to constant factors, for any fixed constant t). It is also known that there exists a 1-private protocol in only 2 communication rounds using $n-2$ random bits (other tradeoffs between randomness complexity and round complexity are given in [KR98]). Several works also analyse the randomness complexity of the secure evaluation of the XOR-n function on multiple instances and prove that one can "amortize" the randomness on several executions of the protocol [KOR03; BGP07].

Even if the properties for the XOR-n function are well-understood, very few studies show how to reduce the randomness complexity of other arbitrary functions, even simple. In particular, finding good upper bounds on the randomness complexity of useful functions such as the AND-n $: \{0,1\}^n \to \{0,1\}, (x_1, \dots, x_n) \mapsto x_1 \wedge \cdots \wedge x_n$ is of high interest. In [KOR99], Kushilevitz *et al.* proved that the class of Boolean functions that can be 1-privately evaluated by n players with a constant number of random bits are exactly those with $O(n)$-complexity size circuits. Their proof is constructive and their construction gives a solution for the AND-n function using 73 random bits (for $n \geqslant 8$ players). More precisely, their protocol requires $72m/(n-1)+1$ random bits for the private evaluation of a Boolean function that can be computed with a Boolean circuit with m binary gates. The underlying idea is to "recycle" the randomness in the protocol by distributing it to different players. Of course, this idea must be pursued with caution, since reusing randomness may hinder the security of the protocols. In [CKOR00], Canetti *et al.* then proved that t-private computations of functions with size m Boolean circuit (*ie.* m logical gates) and n inputs is possible using $O(\mathsf{poly}(t) \cdot (\log(n) + m/n))$ random bits.

In spite of these very strong results, the exact randomness complexity of simple Boolean functions is still not well-understood. In particular, the situation is more complex for the AND-n function than for the XOR-n. The analysis of the randomness complexity of this function (or equivalently of OR-n) is therefore interesting and challenging. It is the main goal of the present chapter to investigate this problem. One application of such protocols is for *private veto* [KY03] (or related "dining cryptographers" [Cha88][1]), since it allows to take decisions by some jury, which must be unanimous, without ever revealing the possible vetoing party(-ies). Another motivation for the study of the AND-n function is that they form, together with the XOR-n function, a basis for the Boolean algebra: simple and efficient protocols for their private evaluation should therefore lead to simple and efficient protocols for the private evaluation of any Boolean function.

5.1.2 Contributions

Using the seminal protocol from [BOGW88] (with the improvement proposed in [GRR98]), one obtains a 1-private protocol for the AND-3 function (with 3 players) which requires 36 random bits. In this computation, the players compute shares of the output of an arithmetic gate given shares of the input wires of that gate (using Shamir's secret sharing [Sha79]). Therefore, each player has to share his private input bit using a linear polynomial over the finite field $\mathbb{F}_4$ and each multiplication requires a re-sharing of the result with another linear polynomial over $\mathbb{F}_4$. Note that this protocol follows closely the transcoding method described in Algorithm 1 (Section 2.5.1). Surprisingly, to the best of our knowledge, this simple protocol is the most randomness-efficient that has been proposed in the literature. Our first contribution is to propose an alternative protocol with a randomness complexity reduced to 6 bits. It seems very difficult to prove a tight lower bound on this randomness complexity. Obviously no deterministic protocol can exist and we prove that protocols using less than two random bits cannot be private (Section 5.3). Applying the same ideas as for AND-3, we propose private protocols for the private evaluation of AND-n for $n \in [4, 15]$, with the improved randomness complexity given in Table 5.1. Our approach is modular and is based on protocols for private computation of several functions that are of independent interest. In particular, they allow us to present a randomness-efficient 1-private protocol for the MAJORITY3 function and we use them in our improved protocol for the AND-n function when n is large.

As mentioned above, for $n \geqslant 16$, the protocol with the lowest randomness complexity is the generic protocol from [KOR99] which requires 73 random bits. In this chapter, we present a 1-

[1] Chaum introduced the dining cryptographers problem in 1988 [Cha88]: several cryptographers want to find out whether the National Security Agency (NSA) or (exactly) one of them pays for the dinner, while respecting each other's right to make a payment anonymously. His famous protocol computes the OR-n function for arbitrary n using n random bits but only for the case where the Hamming weight of the inputs vector is 0 or 1.

n	3	4	5	6	7	8	9	10	11	12	13	14	15
[GRR98]	36	41.79	50.54	50.54	54	57.06	62.29	62.29	66.61	66.61	72	72	72
Us	6	11	13	11	15	16	15	18	18	12	17	16	16

Table 5.1: Randomness complexity for 1-private complexity of `AND-n`.

private protocol to process the `AND-n` function for all $n \in \mathbb{N}$ with less than 26 random bits. In the particular case where n belongs to the infinite geometric progression of integers $\{3 \cdot 2^i, i \in \mathbb{N}\}$, we propose an improved protocol which requires only 12 random bits. The conception of our protocols is modular which allows us to present the general protocol for any $n \in \mathbb{N}$ in a recursive way (with simplified correctness and privacy proofs). As a side effect, our approach also permits to improve the overall randomness complexity of the generic protocol from [KOR99]. We detail this improvement in Section 5.9.

5.2 Notations and Preliminaries

In this section, we formally define basic concepts that will be used thorough this chapter, and we introduce notations.

Protocols presented in this chapter are executed by a set of players $\{P_i; 1 \leq i \leq n\}$, each holding one secret bit x_i. Formally, the set of players can be seen as a set of randomized Turing machine, interacting with each others, and each associated to a *random tape*. The random variable corresponding to the random tape associated to P_i is denoted by $\boldsymbol{R}_i$, whereas a *realization* of it will be denoted $\boldsymbol{r}_i \hookleftarrow \boldsymbol{R}_i$. The set of Boolean values will be viewed as the binary field $\mathbb{F}_2$ with addition and multiplication respectively denoted by $+$ (aka `XOR` operation) and $\cdot$ (aka `AND` operation). When there is no ambiguity, the notation $\cdot$ is omitted. The vector $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$ is denoted by $\boldsymbol{x}$ and the set of Boolean functions defined over $\mathbb{F}_2^n$ is denoted by $\mathcal{B}_n$. We will denote by $x \in \boldsymbol{x}$ the fact that x is a coordinate of $\boldsymbol{x}$. A protocol run by n players $(P_i)_{i \leqslant n}$ to evaluate $f \in \mathcal{B}_n$ on $\boldsymbol{x}$ is denoted by $\mathcal{P}((P_i)_i, f)$. If there is no ambiguity on the players, the notation will be simplified in $\mathcal{P}(n, f)$.

At each step of a protocol $\mathcal{P}(n, f)$, a player P_i can read one or several random bit(s) on its random tape. Then, he may send *messages* to the other players (note that messages are sent over secure channels, *ie.* that only the intended receiver can listen to them). At the end of the protocol, a particular player may broadcast the final message $f(\boldsymbol{x})$. The *communication* $\boldsymbol{C}_i$ associated to P_i is defined as the vector of all messages received by this player during the protocol execution. Note that $\boldsymbol{C}_i$ depends on random bits, and is therefore viewed as a random variable. A realization of it will be denoted by $\boldsymbol{c_i} \hookleftarrow \boldsymbol{C_i}$.

We shall use the notation $P_i\{\ldots\}$ to define a player together with a list of values he knows. For instance, the notation $P_i\{x_i, r_i, r\}$ refers to a player P_i who knows x_i, r_i and r. More generally, for a given protocol, we shall denote by $P_i\{\boldsymbol{C}_i, \boldsymbol{R}_i, x_i\}$ the player P_i with communication $\boldsymbol{C}_i$, random tape $\boldsymbol{R}_i$ and input bit x_i. The random variable $(\boldsymbol{C}_i \mid \boldsymbol{R}_i = \boldsymbol{r}_i, X_i = x_i)$ (or $(\boldsymbol{C} \mid \boldsymbol{R} = \boldsymbol{r}, X = x)$ if there is no ambiguity on i) simply corresponds to the values taken by the communication vector when the player random bits and secret inputs are respectively fixed to $\boldsymbol{r}_i$ and x_i.

We now list classical properties which must be satisfied by a protocol in our context.

Definition 5.2.1 (Correctness)**.** *Let f be a Boolean function. A protocol aiming at evaluating f is said to be* correct *if, after its execution, each player always gets an evaluation of f in the players' secret bits.*

Definition 5.2.2 (Privacy)**.** *A protocol $\mathcal{P}((P_i\{X_i\})_{i\leqslant n}, f)$ is said to be* private with respect to the player $P\{\boldsymbol{C}, \boldsymbol{R}, X\}$ *with* $X \in \boldsymbol{X} = (X_1, \ldots, X_n)$*, if for every message* $\boldsymbol{c} \hookleftarrow \boldsymbol{C}$*, every* $\boldsymbol{r} \hookleftarrow \boldsymbol{R}$*, every vector* $\boldsymbol{x} \hookleftarrow \boldsymbol{X}$ *and every bit* $b \hookleftarrow B$*, the following equality holds:* $\Pr\left(\boldsymbol{C} = \boldsymbol{c} | \boldsymbol{R} = \boldsymbol{r}, \boldsymbol{X} = \boldsymbol{x}, f(\boldsymbol{X}) = b\right) = \Pr\left(\boldsymbol{C} = \boldsymbol{c} | \boldsymbol{R} = \boldsymbol{r}, X = x, f(\boldsymbol{X}) = b\right)$*, where the probability is taken on the supports of* $\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{X}$*. A protocol is said to be* private *if it is private with respect to all players taking part in it.*

The privacy property states that any player taking part in the protocol does not learn more information about the other player's inputs than that he can deduce from its own input x and the result of the evaluation $f(\boldsymbol{x})$. To prove the privacy of some protocols, we will also need the following property.

Definition 5.2.3 (Instance-Secrecy)**.** *A protocol $\mathcal{P}((P_i\{X_i\})_{i\leqslant n}, f)$ is said to satisfy the* instance-secrecy property with respect to the player $P\{\boldsymbol{C}, \boldsymbol{R}, X\}$ *with* $X \in \boldsymbol{X} = (X_1, \ldots, X_n)$*, if for every message* $\boldsymbol{c} \hookleftarrow \boldsymbol{C}$*, every* $\boldsymbol{r} \hookleftarrow \boldsymbol{R}$ *and every vector* $\boldsymbol{x} \hookleftarrow \boldsymbol{X}$*, the following equality holds:* $\Pr\left(\boldsymbol{C} = \boldsymbol{c} | \boldsymbol{R} = \boldsymbol{r}, \boldsymbol{X} = \boldsymbol{x}\right) = \Pr\left(\boldsymbol{C} = \boldsymbol{c} | \boldsymbol{R} = \boldsymbol{r}, X = x\right)$*, where the probability is taken on the supports of* $\boldsymbol{C}, \boldsymbol{R}, \boldsymbol{X}$*. A protocol is said to satisfy the* instance-secrecy property *if it satisfies it with respect to all players taking part in it.*

The instance-secrecy property states that a player learns nothing about the other players' secrets during the protocol execution, including the result of the protocol itself (which is the difference with the privacy property that allows each player to know the result). Indeed, several protocols built in the next sections do not directly return the evaluation of a function f, but an encoding of it. The *encoding of a variable* y is here defined as a pair $(y + r, r)$ such that y and r are independent. For protocols returning such an encoding of a function evaluation $f(\boldsymbol{x})$, the instance-secrecy property implies that no player learns $f(\boldsymbol{x})$, which itself implies that no player learns both $f(\boldsymbol{x}) + r$ and r.

In the following lemma, we exhibit a particular situation where stating the instance-secrecy property of a protocol with respect to a player is relatively easy.

Lemma 5.2.4. *Let $\mathcal{P}$ be a protocol with players $P_i\{X_i\}$. Let $P\{\boldsymbol{C}, \boldsymbol{R}, x\}$ be one these players. If the random variable $(\boldsymbol{C} \mid \boldsymbol{R} = \boldsymbol{r}, X = x)$ is independent of $\boldsymbol{X} = (X_1, \ldots, X_n)$ for any $\boldsymbol{r} \hookleftarrow \boldsymbol{R}$ and any $x \hookleftarrow X$, then the protocol satisfies the instance-secrecy property with respect to P.*

Proof. Let $\boldsymbol{C}'$ denote the random variable $(\boldsymbol{C} \mid \boldsymbol{R} = \boldsymbol{r}, X = x)$. If $\boldsymbol{C}'$ is independent of $\boldsymbol{X}$, then $P(\boldsymbol{C}' = \boldsymbol{c}' | \boldsymbol{X} = \boldsymbol{x}) = P(\boldsymbol{C}' = \boldsymbol{c}')$ for any $\boldsymbol{c}' \hookleftarrow \boldsymbol{C}'$. This directly implies that $\mathcal{P}$ satisfies the instance-secrecy property with respect to P. ◇

In our proofs, we will extensively use the following notion of *functional independence*:

Definition 5.2.5 (Functional Independence)**.** *Let n and i be positive integers such that $i \in [1; n]$ and let $f \in \mathcal{B}_n$. If f satisfies $f(x_1, \ldots, x_i, \ldots, x_n) = f(x_1, \ldots, x_i{+}1, \ldots, x_n)$ for every $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$, then f is said to be* functionally independent of x_i*, which is denoted by $f \perp x_i$. Let $\boldsymbol{x}$ be a vector composed of coordinates of $(x_1, \ldots, x_n)$. The function f is said to be functionally independent of $\boldsymbol{x}$, denoted $f \perp \boldsymbol{x}$, if it is independent of every coordinate of $\boldsymbol{x}$.*

Remark 5.2.6. The functional independence is closely related to the classic notion of *sensitivity*: a function f is functionally independent of the variable x_i if and only if f is *not* sensitive to x_i. The functional independence implies the statistical one, whereas the converse is false. Definition 5.2.5 directly extends to random variables: if a random variable Y can be described as a function depending only on variables statistically independent from a random variable X, then $X \perp Y$.

Remark 5.2.7. The notion of *functional dependence* is trivially defined as the contrary of the functional independence: if f is not functionally independent of x_i, it is functionally dependent on x_i.

In the privacy proofs done in this chapter, we will often use the following Lemmas 5.2.8 and 5.2.9 which establish a link between the notions of statistical and functional dependencies in some useful contexts.

Lemma 5.2.8. *Let $\boldsymbol{R}, \boldsymbol{X}$ and $\boldsymbol{V}$ be three random vectors such that $\boldsymbol{R}$ is uniformly distributed and independent of $\boldsymbol{X}$, and $\boldsymbol{V}$ is a function of $\boldsymbol{X}$ and $\boldsymbol{R}$. If any non-zero linear combination of coordinates of $\boldsymbol{V}$ can be written $g(\boldsymbol{X}, \boldsymbol{R}) + R$ with $R \in \boldsymbol{R}$ and $g \perp R$, then $\boldsymbol{V}$ is statistically independent of $\boldsymbol{X}$.*

Proof. Since every coordinate of $\boldsymbol{R}$ is uniform, then any variable taking the form $g(\boldsymbol{X}, \boldsymbol{R}) + R$ with $g \perp R$ is uniform. Moreover, the functional independence between g and R implies that this uniformity holds even knowing $\boldsymbol{X} = \boldsymbol{x}$ for any $\boldsymbol{x}$. We deduce that $(\boldsymbol{V} \mid \boldsymbol{X} = \boldsymbol{x})$ is uniform whatever $\boldsymbol{x}$, which implies the independence between $\boldsymbol{V}$ and $\boldsymbol{X}$. $\diamond$

Lemma 5.2.9. *Let X, Y, R be three random variables, such that R is uniformly distributed, and X and Y are functions. If $X \perp R$ and $Y \perp R$ then $X + Y + R$ is statistically independent of (X, Y).*

Proof. Since $X \perp R$ and $Y \perp R$, then $(X + Y) \perp R$. Moreover, since R is uniformly distributed, then $X + Y + R$ is also uniformly distributed, and statistically independent of (X, Y). $\diamond$

The main purpose of this chapter is to reduce the number of random bits required to evaluate some particular Boolean functions, while continuing to satisfy the privacy property. We define hereafter the *randomness complexity* of a protocol.

Definition 5.2.10 (Randomness Complexity)**.** *A protocol is said to be d-*random *if, for any execution, the total number of random bits used by the players is at most d.*

5.3 A lower bound

The private processing of the `AND-n` function has been proven to be infeasible for $n = 2$ [BOGW88]. The exhibition of a private protocol involving the minimal number of three players is therefore a natural problem.

In this section, we restrict our study to what we further design as *oblivious* protocols, *ie.* protocols where, for any execution, the players generating/reading the random bits are always the same.

This section presents the proof of the following theorem:

Theorem 5.3.1. *The minimal randomness complexity of any 1-private oblivious protocol computing the `AND-3` function is strictly higher than 2.*

We reason *ad absurdio* and distinguish two cases: at the beginning of the protocol, either both random bits are known by a single player, or are held by two different players. We denote those random bits by r_1 and r_2 respectively. Before giving the proof we start by introducing a formalisation of our problem in terms of circuit privacy [SYY99].

5.3.1 Problem Formalisation and Notations

Following the approach in [ISW03], we define a *deterministic circuit* as a directed acyclic graph whose vertices are Boolean gates and whose edges are wires. Without loss of generality, we assume that every gate has fan-in at most 2. A *randomised circuit* is a circuit augmented with random-bit gates. A *random-bit gate* is a gate with fan-in 0 that produces a uniformly distributed random bit and sends it along its output wire. The *depth* of a circuit is the length of the longest path from a circuit's input to a circuit's output. Similarly, the *depth of a gate* is the length of the longest path from a circuit's input to this gate. We define a total order on the gates of a circuit by first ordering all gates according to their depth, and then arbitrarily for each gate at the same depth. An illustration of such an ordering on a simple circuit is illustrated in Figure 5.1.

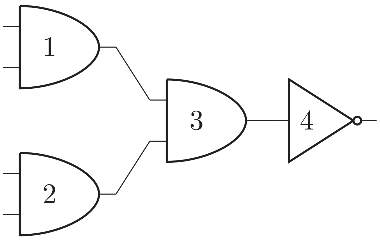


Figure 5.1: Example of circuit ordering.

We see a protocol as a randomised circuit of gates `NOT` and `AND` (together with random bit gates). Its randomness complexity is the number of random-bit gates. To analyse the privacy of the 2-random protocol for the `AND-3` function, we follow an assignment/simulation approach (as *e.g.* in [ISW03]):

- *assignment phase*; the first step of the proof consists in assigning each node/gate of the graph/circuit to a single player[2]. Then, for each edge/wire in the graph/circuit linking a node assigned to Player P_i to a node assigned to Player P_j, the edge is assigned to both P_i and P_j. Note that edges can hence be assigned to at most two players. The three input wires x_1, x_2 and x_3 of the protocol are always assigned to P_1, P_2 and P_3 respectively. For the special case of the output gate, $x_1x_2x_3$ is always assigned to all three players.

- *privacy*; a protocol is *private with respect to a player* P_i if and only if the joint distribution of all the bits on wires assigned to P_i is independent from every other player's input conditioned by the output of the protocol. A protocol is *private* if and only if it is private with respect to every player taking part in it.

5.3.2 Proof's Part 1: Both Random Bits Belong to the Same Player

Without loss of generality we assume that the wires of both random-bit gates are assigned to P_1, which implies that every random-bit gate is also assigned to P_1.

Since the circuit evaluates $x_1x_2x_3$, there exists at least one gate in the protocol that evaluates a function of degree 3 in the x_i's. We denote by $\mathbf{G}$ the first AND gate of this kind and we prove hereafter that every possible assignment of this gate breaks the privacy. To this end, we consider each possible assignment.

[2] Note that this assignment does not make sense for non-oblivious protocols, where the random bits could be generated by different players depending on the execution.

- case $\mathbf{G}$ assigned to P_1: by construction, at least one input wire of $\mathbf{G}$ functionally depends on x_2 or on x_3. The value on this wire is denoted by v: by definition, there exists a Boolean function f such that $v = f(x_1, x_2, x_3, r_1, r_2)$ and $f(x_1, x_2, x_3, r_1, r_2)$ is functionally dependent on $\{x_2, x_3\}$. Since P_1 knows $\{v, r_1, r_2, x_1\}$ (by assignment), it can then be deduced that P_1 can retrieve information on $\{x_2, x_3\}$ (by definition of the functional dependence, there exists certain r_1, r_2, x_1, such that v becomes a non constant function of $\{x_2, x_3\}$). Consequently, privacy is broken.

- case $\mathbf{G}$ assigned to P_2 (symmetric case for P_3): by construction, at least one input wire of $\mathbf{G}$ is of degree 2 in the x_i's, while the other one is of degree 1 or 2 in the x_i's, and both wires are assigned to P_2. To maintain privacy, the degree 2 wire has to be additively protected by a random bit (*ie.*, a random bit has to be used as a mask), and the other wire must either be equal to x_2 or to be additively protected by a different random bit.

  We denote by w one of these wires which functionally depends on x_2. According to what we just said, w is hence either x_2, or can be without loss of generality expressed as $f_2 \oplus R$, where f_2 functionally depends on x_2 and on no random bit, and $R \in \{r_1, r_2, r_1 \oplus r_2\}$. We denote by v the value on the other wire, which must hence functionally depend on either x_1 or x_3 (or both) and must additively involve R', with $R' \in \{r_1, r_2, r_1 \oplus r_2\} \setminus R$. Without loss of generality, in the following, we will always set $R = r_2$ and $R' = r_1$.

  By construction, and according to the inputs of $\mathbf{G}$, the algebraic normal form of the Boolean function representing the output of $\mathbf{G}$ involves the term $f_2 r_1$, where f_2 depends on x_2.

  The output of the circuit being $x_1 x_2 x_3$, there must hence be another gate in the circuit computing a Boolean function involving $f_2 r_1$. Here are the ways this can be achieved:

  1. a gate $\mathbf{G}_0$ evaluates a product between f_2 and r_1
  2. a gate $\mathbf{G}_1$ evaluates a product between f_2 and a value of the form $r_1 \oplus \cdots \oplus r_2$ and a gate $\mathbf{G}_2$ evaluates $f_2 r_2$
  3. a gate $\mathbf{G}_1$ evaluates a product between f_2 and a value of the form $\hat{v} \oplus \cdots \oplus r_1$ and a gate $\mathbf{G}_2$ evaluates a product between f_2 and $\hat{v}$
  4. a gate $\mathbf{G}_1$ evaluates a product between a value of the form $f_2 \oplus \cdots \oplus r_2$ and a value of the form $\hat{v} \oplus \cdots \oplus r_1$ and a gate $\mathbf{G}_2$ evaluates $r_1 r_2$.

  Let us study separately each of these cases. We will show that every possible assignment of $\mathbf{G}_0, \mathbf{G}_1$ or $\mathbf{G}_2$ breaks the privacy:

  1. Since one of the two input wires of $\mathbf{G}_0$ is f_2, privacy can only be maintained if $\mathbf{G}_0$ is assigned to P_2. However, if $\mathbf{G}_0$ is assigned to P_2, then by construction the other input wire is also assigned to P_2. This implies that r_1 is also assigned to P_2. The set of wires assigned to P_2 hence contains $\{f_2, r_1, w, v\}$. Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_1 or x_3 and f_v functionally depends on r_1, privacy is broken.
  2. Since both $\mathbf{G}_1$ and $\mathbf{G}_2$ have an input wire whose value is f_2, then both $\mathbf{G}_1$ and $\mathbf{G}_2$ must be assigned to P_2. Consequently, the assignment of P_2 contains $\{r_2, r_1 \oplus \cdots \oplus r_2, v\}$. Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_1 or x_3 and f_v functionally depends on r_1, privacy is broken.
  3. Since both $\mathbf{G}_1$ and $\mathbf{G}_2$ have an input wire whose value is f_2, then both $\mathbf{G}_1$ and $\mathbf{G}_2$ must be assigned to P_2. Consequently, the assignment of P_2 contains $\{r_2, \hat{v} \oplus \cdots \oplus r_1, v, \hat{v}\}$.

Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_1 or x_3 and f_v depends on r_1, privacy is broken.

4. Let us denote by v' the value $f_2 \oplus \cdots \oplus r_2$, and by respectively f_v and f'_v the Boolean functions evaluating v and v'. We focus on the player that $\mathbf{G}_1$ can be assigned to. First of all, if $\mathbf{G}_1$ is assigned to P_1, its assignment contains $\{v', r_1, r_2\}$. Since f_2 depends on x_2, hence privacy is broken. If $\mathbf{G}_1$ is assigned to P_2, hence its assignment contains $\{v', v, \hat{v} \oplus \cdots \oplus r_1\}$. Since f_v functionally depends on either x_1 or x_3 and f_v depends on r_1, privacy is broken. If $\mathbf{G}_1$ is assigned to P_3, his assignment contains $\{v', \hat{v} \oplus \cdots \oplus r_1\}$. In this case, if $\mathbf{G}_2$ is assigned to P_2, privacy is broken because f_v functionally depends on r_1 and on x_1 or x_3. If $\mathbf{G}_2$ is assigned to P_3, privacy is broken because $f_{v'}$ depends on r_2 and on x_2. Consequently, r_1r_2 can only be known by P_1. Hence no further message can be sent without breaking privacy and consequently it is impossible to compute $x_1x_2x_3$.

Hence, no oblivious protocol using only two random bits drawn by the same player and allowing for the 1-private secure computation of the AND of three players can exist.

5.3.3 Proof's part 2: Both bits held by different players

We consider that both random-bit gates (and their wires) are assigned to two different players, without loss of generality P_1 and P_2, to which are assigned respectively r_1 and r_2. Consider the first AND gate $\mathbf{G}$ which evaluates a value of degree 3 in x_i's.

- case $\mathbf{G}$ assigned to P_1 (the case is symmetric for P_2):

  As in the previous subsection, we denote by w one of the input wires of $\mathbf{G}$ which functionally depends on x_1 (w is either x_1 or can be without loss of generality expressed as $f_1 \oplus R$, where f_1 functionally depends on x_1, and $R \in \{r_1, r_2, r_1 \oplus r_2\}$); and we denote by v the value on the other wire, which must hence functionally depend on either x_2 or x_3 (or both) and additively involve R', with $R' \in \{r_1, r_2, r_1 \oplus r_2\} \setminus R$. Without loss of generality, in the following, we will always set $R = r_1$ and $R' = r_2$. Similarly, we denote by v the input of $\mathbf{G}$ that is not w. The output of $\mathbf{G}$ can be evaluated by a Boolean function. We can show similarly to the previous case, that the ANF of this function involves the term f_1r_2, where f_1 depends on x_1.

  The output of the circuit being $x_1x_2x_3$, there must hence be another gate in the circuit computing a Boolean function involving f_1r_2. Here are all the possible ways this can be achieved:

  1. a gate $\mathbf{G}_0$ evaluates a product between f_1 and r_2
  2. a gate $\mathbf{G}_1$ evaluates a product between f_1 and a value of the form $r_1 \oplus \cdots \oplus r_2$ and a gate $\mathbf{G}_2$ evaluates f_1r_1
  3. a gate $\mathbf{G}_1$ evaluates a product between f_1 and a value of the form $\hat{v} \oplus \cdots \oplus r_2$ and a gate $\mathbf{G}_2$ evaluates a product between f_1 and $\hat{v}$
  4. a gate $\mathbf{G}_1$ evaluates a product between a value of the form $f_1 \oplus \cdots \oplus r_1$ and a value of the form $\hat{v} \oplus \cdots \oplus r_2$ and a gate $\mathbf{G}_2$ evaluates r_1r_2.

  Let us study separately each of these cases. We will show that every possible assignment of $\mathbf{G}_0, \mathbf{G}_1$ or $\mathbf{G}_2$ breaks the privacy:

1. Since one of the input wires of $\mathbf{G}_0$ is f_1, privacy can only be maintained if this gate is assigned to P_1. Its assignment hence contains $\{f_1, r_2, v\}$. Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_2 or x_3 and the only random bit f_v depends on is r_2, privacy is broken.
2. Since one of the input wires of $\mathbf{G}_1$ is f_1, privacy can only be maintained if this gate is assigned to P_1. Its assignment hence contains $\{r_1, f_1, r_1 \oplus \cdots \oplus r_2, v\}$. Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_2 or x_3 and f_v depends on r_2, privacy is broken.
3. Since one of the input wires of $\mathbf{G}_1$ is f_1, privacy can only be maintained if this gate is assigned to P_1. Its assignment hence contains $\{r_1, f_1, \hat{v} \oplus \cdots \oplus r_2, v\}$. Denote by f_v the Boolean function evaluating v. Since f_v functionally depends on either x_2 or x_3 and f_v functionally depends on r_2, privacy is broken.
4. We denote by v' the value of the form $f_1 \oplus \cdots \oplus r_1$. We focus on the player that $\mathbf{G}_1$ can be assigned to. If it is assigned to P_1, then his assignment contains $\{v, v', r_1\}$. Denote by f_v the Boolean function evaluating v. P_1 obtains information on x_2 or x_3 thanks to the fact that f_v is functionally dependent on one of these variables. If it is assigned to P_2 then his assignment contains $\{v', \hat{v} \oplus \cdots \oplus r_2\}$. Hence $\mathbf{G}_2$ can only be assigned to P_3 (whose assignment hence contains $\{r_1, \hat{v} \oplus \cdots \oplus r_2\}$ and hence no further message can be assigned to any player without breaking privacy. Consequently it is impossible to compute $x_1x_2x_3$. Similarly, if $\mathbf{G}_1$ is assigned to P_3 then $\mathbf{G}_2$ can only be assigned to P_2 and for the same reason, it is impossible to compute $x_1x_2x_3$.

- case $\mathbf{G}$ assigned to P_3: Similarly, we denote by w one of the input wires of $\mathbf{G}$ which functionally depends on x_3 (w is either x_3 or can be without loss of generality expressed as $f_3 \oplus R$, where f_3 functionally depends on x_3 and on no random bit, and $R \in \{r_1, r_2, r_1 \oplus r_2\}$); and we denote by v the value on the other wire, which must hence functionally depend on either x_1 or x_2 (or both) and additively involve R', with $R' \in \{r_1, r_2, r_1 \oplus r_2\} \setminus R$. Without loss of generality, in the following, we will always set $R = r_2$ and $R' = r_1$. Similarly, we denote by v the input of $\mathbf{G}$ that is not w. The output of $\mathbf{G}$ can be evaluated by a Boolean function. We can show similarly to the previous case, that the ANF of this function involves the term f_3r_1, where f_3 functionally depends on x_3.

  The output of the circuit being $x_1x_2x_3$, there must hence be another gate in the circuit computing a Boolean function involving f_3r_1. Here are the ways this can be achieved:

  1. a gate $\mathbf{G}_0$ evaluates a product between f_3 and r_1
  2. a gate $\mathbf{G}_1$ evaluates a product between f_3 and a value of the form $r_1 \oplus \cdots \oplus r_2$ and a gate $\mathbf{G}_2$ evaluates f_3r_2
  3. a gate $\mathbf{G}_1$ evaluates a product between f_3 and a value of the form $\hat{v} \oplus \cdots \oplus r_1$ and a gate $\mathbf{G}_2$ evaluates a product between f_3 and $\hat{v}$
  4. a gate $\mathbf{G}_1$ evaluates a product between a value of the form $f_3 \oplus \cdots \oplus r_2$ and a value of the form $\hat{v} \oplus \cdots \oplus r_1$ and a gate $\mathbf{G}_2$ evaluates r_1r_2.

  Let us study separately each of these cases. We will show that every possible assignment of $\mathbf{G}_0, \mathbf{G}_1$ or $\mathbf{G}_2$ breaks the privacy:

  1. Since f_3 is an input of $\mathbf{G}$, then this gate must be assigned to P_3. Hence its assignment contains $\{v, r_1\}$, and privacy is broken because v functionally depends on x_1 or x_2.

2. Since f_3 is an input of $\mathbf{G}$, then this gate must be assigned to P_3. Hence its assignment contains $\{v, r_1, r_1 \oplus \cdots \oplus r_2, w\}$, and privacy is broken because v functionally depends on x_1 or x_2.
3. Since f_3 is an input of $\mathbf{G}$, then this gate must be assigned to P_3. Hence its assignment contains $\{v, r_1, \hat{v} \oplus \cdots \oplus r_1\}$, and privacy is broken because v functionally depends on x_1 or x_2.
4. If G_2 is assigned to a player, then this player knows both r_1 and r_2. This is the first case of our proof, and we already proved that such protocol is impossible.

Hence, no protocol using only two random bits drawn by different players and allowing for the 1-private secure computation of the AND of three players can exist.

5.4 Private Evaluation of `AND-n` for $n = 3$

To the best of our knowledge, the most random-efficient 1-private protocol for the `AND-3` function requires 36 random bits and is based on [BOGW88; GRR98].

In this section, we tackle this problem by describing a 6-random protocol allowing to process `AND-3` in a 1-private way. We also propose an alternative protocol where the result is returned in an encoded form, and therefore stays unknown to any player. It will be the cornerstone of our general protocol presented in Sections 5.6 and 5.7.

Let us recall the goal of our protocol. Three players $P_1\{x_1\}, P_2\{x_2\}, P_3\{x_3\}$ are provided with a limited number of independent[3] and uniformly distributed random bits. At each round, a player can read one or several random bit(s) on its random tape, and send one or several message(s) to other players. After the execution of the protocol, each player must know the value $\mathtt{AND}(x_1, x_2, x_3) = x_1x_2x_3$ (correctness property), but must not learn more information on (x_1, x_2, x_3) than that given by its own input x_i and $\mathtt{AND}(x_1, x_2, x_3)$ (privacy property). This section aims at describing a protocol $\mathcal{P}_{\mathtt{AND-3}}$ verifying these two properties with a randomness complexity equal to 6. This protocol is the composition of the three following protocols $\mathcal{P}^{-}_{\mathtt{AND-2}}$, $\mathcal{P}^{(1)}_{\mathtt{AND-2}}$ and $\mathcal{P}_{\mathtt{XOR}}$.

The first protocol, $\mathcal{P}^{-}_{\mathtt{AND-2}}$ allows three players to process the multiplication of two of their secrets, while satisfying the instance-secrecy property. It is 3-random and, at the end of its execution, the `AND` of the two secrets is encoded among the players who own these secrets.

Protocol $\mathcal{P}^{-}_{\mathtt{AND-2}}$

```
Require: players $P_a\{x_a, \boldsymbol{R}_a\}$, $P_b\{x_b, \boldsymbol{R}_b\}$ and $P_c\{\boldsymbol{R}_c\}$
Ensure: $P_b$ gets $x_ax_b + r$ with $r$ a random bit known by $P_c$ only

  round 1: Players $P_a$ and $P_b$ read random bits $r_1$ and $r_2$ on their respective random tapes
  $P_a$ sends $m_1 = x_a + r_1$ to $P_b$ and $r_1$ to $P_c$
  $P_b$ sends $m_2 = x_b + r_2$ to $P_a$ and $r_2$ to $P_c$
  round 2: $P_c$ reads a random bit $r$ on its random tape
  $P_c$ sends $m_3 = r_1r_2 + r$ to $P_a$
  round 3: $P_a$ sends $m_4 = m_3 + m_2x_a$ to $P_b$        ▷ $m_4 = r_1r_2 + r + (x_b + r_2)x_a$
  $P_b$ computes $m_5 = m_4 + r_2m_1$        ▷ $m_5 = x_ax_b + r$
```

[3] For the ease of exposition, we assume that the players are provided with random tapes. However, we do not consider the model in which parties receive a sample from a predetermined joint distribution where random strings are correlated as in [IKM+13]. When two players have the same random bit r in their random tape, we actually assume that one player has picked r uniformly at random and has then sent it to the other player.

We describe hereafter two other protocols $\mathcal{P}^{(1)}_{\texttt{AND-2}}$ and $\mathcal{P}_{\texttt{XOR}}$. The first one is very close to $\mathcal{P}^{-}_{\texttt{AND-2}}$; the main difference is that one of the secrets which must be multiplied by the protocol comes in an encoded form at input. It is also 3-random and outputs a refreshed encoding of the AND of the encoded input and the other secret. The protocol $\mathcal{P}_{\texttt{XOR}}$ allows n players to privately process the addition of two of their secrets. It is well-known that it is impossible to privately compute the addition of $n \geqslant 3$ inputs deterministically but this is not the case for two inputs. The simple solution that we propose for $\mathcal{P}_{\texttt{XOR}}$ is indeed deterministic. It will be used each time we shall need to recover and broadcast a value owned in an encoded form by a set of players.

We can now present our new 6-random protocol allowing three players to process the multiplication of their respective secrets. We present two versions of it: the first one provides the players with an encoding of the evaluation, whereas the second one allows all of them to know the evaluation itself. Before executing the protocols, the three players P_1, P_2 and P_3 are respectively provided with the random tapes $\boldsymbol{R}_1$, $\boldsymbol{R}_2 \cup \boldsymbol{R}'_2$ and $\boldsymbol{R}_3 \cup \boldsymbol{R}'_3$. Random tapes $\boldsymbol{R}_i$ (resp. $\boldsymbol{R}'_i$) will be used when executing protocol $\mathcal{P}^{-}_{\texttt{AND-2}}$ (resp. $\mathcal{P}^{(1)}_{\texttt{AND-2}}$). Before the execution of the protocol, each player fills his random tape such that: $\boldsymbol{R}_1 \leftarrow \{r_1\}$, $\boldsymbol{R}_2 \leftarrow \{r_2\}$, $\boldsymbol{R}'_2 \leftarrow \{r'_1\}$, $\boldsymbol{R}_3 \leftarrow \{r\}$, $\boldsymbol{R}'_3 \leftarrow \{r'_2, r''\}$. This implies that each player has initially generated the random bits he is going to use during the protocol. This assumption will be useful in the next section as it will enable the reuse of some random bits when executing several protocols in parallel or sequentially. At the end of this section we propose another protocol that will be useful in the next section. This protocol is very close to $\mathcal{P}^{(1)}_{\texttt{AND-2}}$: it allows three players to process the encoding of the multiplication of two encoded secrets. The protocol is 5-random. The correctness of all protocols can be directly checked thanks to the comments inserted in the algorithms descriptions.

Protocol $\mathcal{P}^{(1)}_{\texttt{AND-2}}$

```
Require: players $P_a\{x_a + r', \boldsymbol{R}_a\}$, $P_b\{x_b, r', \boldsymbol{R}_b\}$ and $P_c$
Ensure: $P_a$ gets $x_a x_b + r''$ with $r''$ a random bit known by $P_b$ only

    round 1: Players $P_a$ and $P_b$ read a random bit $r'_1$ and $r'_2$ on their tapes
    $P_a$ sends $m_1 = (x_a + r') + r'_1$ to $P_b$ and $r'_1$ to $P_c$
    $P_b$ sends $m_2 = x_b + r'_2$ to $P_a$ and $r'_2$ to $P_c$
    round 2: $P_b$ reads a random bit $r''$ on its random tape
    $P_b$ sends $m_3 = (m_1 + r')x_b + r''$ to $P_c$
                                                        ▷ $m_3 = (x_a + r'_1)x_b + r''$
    round 3: $P_c$ sends $m_4 = m_3 + r'_1 r'_2$ to $P_a$
                                                        ▷ $m_4 = x_a x_b + r'_1(x_b + r'_2) + r''$
    $P_a$ computes $m_5 = m_4 + r'_1 m_2$
                                                        ▷ $m_5 = x_a x_b + r''$
```

Protocol $\mathcal{P}_{\texttt{XOR}}$

Require: a list of players $(P_i)_{i \leq n}$ containing $P_a\{x_a\}$ and $P_b\{x_b\}$.
Ensure: all players get $x_a + x_b$.

```
  round 1: Player $P_a$ sends $m_1 = x_a$ to $P_b$
  Player $P_b$ computes $m_2 = m_1 + x_b$                    ▷ $m_2 = x_a + x_b$
  round 2: Player $P_b$ broadcasts $m_2$
                     ▷ we get $(P_1\{x_a + x_b\}, \cdots, P_n\{x_a + x_b\})$
```

Protocol $\mathcal{P}^{-}_{\texttt{AND-3}}$

Require: players $P_1\{x_1, \boldsymbol{R}_1\}$, $P_2\{x_2, \boldsymbol{R}_2 \cup \boldsymbol{R}'_2\}$ and $P_3\{x_3, \boldsymbol{R}'_3\}$
Ensure: P_1 gets $x_1x_2x_3 + r''$, where r'' is a random bit only known by P_3

```
  Protocol $\mathcal{P}^{-}_{\texttt{AND-2}}$ for $(P_1\{x_1, \boldsymbol{R}_1\}, P_2\{x_2, \boldsymbol{R}_2\}, P_3)$
                     ▷ we get $(P_2\{x_2, x_1x_2 + r\}, P_3\{x_3, r\})$
  Protocol $\mathcal{P}^{(1)}_{\texttt{AND-2}}$ for $(P_2\{x_1x_2 + r, \boldsymbol{R}'_2\}, P_3\{x_3, r, \boldsymbol{R}'_3\}, P_1)$
                     ▷ we get $(P_2\{x_1x_2x_3 + r''\}, P_3\{r''\})$
```

Protocol $\mathcal{P}_{\texttt{AND-3}}$

Require: players $P_1\{x_1, \boldsymbol{R_1}\}$, $P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R_2^\star}\}$ and $P_3\{x_3, \boldsymbol{R_3^\star}\}$
Ensure: P_1, P_2, P_3 get $x_1x_2x_3$

```
  Protocol $\mathcal{P}^{-}_{\texttt{AND-3}}$ for $P_1\{x_1, \boldsymbol{R_1}\}$, $P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R_2^\star}\}$ and $P_3\{x_3, \boldsymbol{R_3^\star}\}$
                     ▷ we get $(P_2\{x_1x_2x_3 + r''\}, P_3\{r''\})$
  Protocol $\mathcal{P}_{\texttt{XOR}}$ for $P_2\{x_1x_2x_3 + r''\}$, $P_3\{r''\}$ and $P_1$
                     ▷ we get $(P_1\{x_1x_2x_3\}, P_2\{x_1x_2x_3\}, P_3\{x_1x_2x_3\})$
```

The following theorem states the instance-secrecy of Protocol $\mathcal{P}^{-}_{\texttt{AND-3}}$ and the privacy of Protocol $\mathcal{P}_{\texttt{AND-3}}$.

Theorem 5.4.1. *Protocol $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property and Protocol $\mathcal{P}_{\texttt{AND-3}}$ is private.*

Proof. We start by proving the instance-secrecy of Protocol $\mathcal{P}^{-}_{\texttt{AND-3}}$. To this end, we first need to introduce the two following useful lemmas:

Lemma 5.4.2. *Let $P_i\{\boldsymbol{C_i}, \boldsymbol{R}_i\}$, $i \in \{a, b, c\}$, be a player taking part in $\mathcal{P}^{-}_{\texttt{AND-2}}$. Every non-zero linear combination of $\boldsymbol{C_i}$ coordinates can be written $g(X_a, X_b, \boldsymbol{R}_a, \boldsymbol{R}_b, \boldsymbol{R}_c) + R^\star$ where $R^\star \in (\boldsymbol{R}_a \cup \boldsymbol{R}_b \cup \boldsymbol{R}_c) \backslash \boldsymbol{R}_i$ and $g \perp R^\star$.*

Proof. We prove the lemma statement for the three players taking part in $\mathcal{P}^{-}_{\texttt{AND-2}}$. By construction we have $\boldsymbol{C_a} = (X_b + R_2, R_1R_2 + R)$, $\boldsymbol{C_b} = (X_a + R_1, R_1R_2 + X_a(X_b + R_1) + R)$ and $\boldsymbol{C_c} = (R_1, R_2)$ (where the players' secrets and random tapes content are viewed as random variables, and are therefore denoted with capital letters). Moreover, the content of the random tapes of the 3 players are $\{R_1\}$, $\{R_2\}$ and $\{R\}$ respectively. It can be checked that they are disjoint. Any linear combination of $\boldsymbol{C_a}$ coordinates involving the second coordinate can be written as $g(\cdots) + R$ with $g \perp R$ and $R \in \boldsymbol{R}_c$. The remaining non-zero combination can be written as $g(\cdots) + R_2$ with $g \perp R_2$ and $R_2 \in \boldsymbol{R}_b$. Any linear combination involving the second coordinate of $\boldsymbol{C_b}$ is of the form $g(\cdots) + R$ with $g \perp R$ and $R \in \boldsymbol{R}_c$. The remaining combination is already of the form $g(\cdots) + R_1$, with $R_1 \in \boldsymbol{R}_a$. Finally, any linear combination of $\boldsymbol{C_c}$ coordinates can be written as $g(\cdots) + R_i$ with $R_i \in \boldsymbol{R}_a \cup \boldsymbol{R}_b$. ◇

Lemma 5.4.3. *Let $P_i\{\boldsymbol{C_i}, \boldsymbol{R}_i\}$, $i \in \{a, b, c\}$, be a player taking part in $\mathcal{P}^{(1)}_{\texttt{AND-2}}$. Every non-zero linear combination of $\boldsymbol{C_i}$ coordinates can be written $g(X_a, X_b, \boldsymbol{R}_a, \boldsymbol{R}_b) + R^\star$ where $R^\star \in (\boldsymbol{R}_a \cup \boldsymbol{R}_b)\backslash\boldsymbol{R}_i$ and $g \perp R^\star$.*

Proof. For players P_a and P_c, the proof is similar as for Lemma 5.4.2 (after replacing R_1, R_2 and R by R_1', R_2' and R'' respectively). For player P_b, we have $\boldsymbol{C_b} = (X_a + R' + R_1')$ which is already of the form $g(\cdots) + R_1'$ with $g \perp R_1'$ and $R_1' \in \boldsymbol{R}_a$. The lemma statement is hence correct. ◇

Based on Lemmas 5.4.2 and 5.4.3, we can now prove the instance-secrecy of $\mathcal{P}^{-}_{\texttt{AND-3}}$.

Proof. Let us check the instance-secrecy property with respect to each player attending to $\mathcal{P}^{-}_{\texttt{AND-3}}$. Note that every value appearing during the protocol is replaced by the corresponding random variable (to enable independence proofs). We moreover denote by $\boldsymbol{C_i}$ the set of messages received by P_i during $\mathcal{P}^{-}_{\texttt{AND-3}}$.

- $P_1\{\boldsymbol{C_1}, \boldsymbol{R}_1\}$ plays the role of the first player in $\mathcal{P}^{-}_{\texttt{AND-2}}$, and of the third player in $\mathcal{P}^{(1)}_{\texttt{AND-2}}$. During $\mathcal{P}^{-}_{\texttt{AND-3}}$, he only receives messages coming from the execution of these two protocols. The sets of random tapes used in $\mathcal{P}^{-}_{\texttt{AND-2}}$ and $\mathcal{P}^{(1)}_{\texttt{AND-2}}$ are respectively $\boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \boldsymbol{R}_3 = \{R_1, R_2, R\}$ and $\boldsymbol{R}_2' \cup \boldsymbol{R}_3' = \{{R_1}', {R_2}', R''\}$, which are disjoint. Applying Lemmas 5.4.2 and 5.4.3, we get that any non-zero linear combination of $\boldsymbol{C_1}$ coordinates takes the form $g_1(X_1, X_2, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3) + g_2(X_1X_2 + R, X_3, \boldsymbol{R}_2', \boldsymbol{R}_3') + R_1^\star + R_2^\star$ where $R_1^\star \in \boldsymbol{R}_2 \cup \boldsymbol{R}_3$ (resp. $R_2^\star \in \boldsymbol{R}_2' \cup \boldsymbol{R}_3'$), and $g_1 \perp R_1^\star$ (resp. $g_2 \perp R_2^\star$). Since $\boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \boldsymbol{R}_3$ and $\boldsymbol{R}_2' \cup \boldsymbol{R}_3'$ are disjoint, we have $g_1 \perp R_2^\star$ and $g_2 \perp R_1^\star$, from which we deduce $(g_1 + g_2) \perp R_1^\star$ and $(g_1 + g_2) \perp R_2^\star$. As a consequence, any non-zero linear combination of $\boldsymbol{C_1}$ coordinates takes the form of a sum $g(X_1, X_2, X_3, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3, \boldsymbol{R}_2', \boldsymbol{R}_3') + R^\star$ such that $R^\star \in \boldsymbol{R}_2 \cup \boldsymbol{R}_3 \cup \boldsymbol{R}_2' \cup \boldsymbol{R}_3'$ and $g \perp R^\star$. Hence, for every $x_1 \hookleftarrow X_1$ and every $r_1 \hookleftarrow \boldsymbol{R}_1$, the set of messages $\boldsymbol{C_1}$ is independent of (X_2, X_3) (by Lemma 5.2.8) which implies that $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property with respect to P_1 (Lemma 5.2.4).

- $P_2\{\boldsymbol{C_2}, \boldsymbol{R}_2 \cup \boldsymbol{R}_2'\}$ plays the role of the second player in $\mathcal{P}^{-}_{\texttt{AND-2}}$, and of the first player in $\mathcal{P}^{(1)}_{\texttt{AND-2}}$. During $\mathcal{P}^{-}_{\texttt{AND-3}}$, he only receives messages coming from the execution of these two protocols. The same reasoning as for P_1 shows that any non-zero linear combination of $\boldsymbol{C_2}$ coordinates coming only from $\mathcal{P}^{-}_{\texttt{AND-2}}$ and $\mathcal{P}^{(1)}_{\texttt{AND-2}}$ takes the form of a sum $g(X_1, X_2, X_3, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3, \boldsymbol{R}_2', \boldsymbol{R}_3') + R^\star$ such that $R^\star \in \boldsymbol{R}_1 \cup \boldsymbol{R}_3'$ and $g \perp R^\star$. We hence conclude that for every $x_2 \hookleftarrow X_2$ and every $(r_2, r_1') \hookleftarrow (\boldsymbol{R}_2, \boldsymbol{R}_2')$, the set of messages $\boldsymbol{C_2}$ is independent of (X_1, X_3) (by Lemma 5.2.8) which implies that $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property with respect to P_2 (Lemma 5.2.4).

- $P_3\{\boldsymbol{C_3}, \boldsymbol{R}'_3\}$ plays the role of the third player in $\mathcal{P}^{-}_{\texttt{AND-2}}$, and of the second player in $\mathcal{P}^{(1)}_{\texttt{AND-2}}$. During $\mathcal{P}^{-}_{\texttt{AND-3}}$, he only receives messages coming from the execution of these two protocols. The same reasoning as for P_1 shows that any non-zero linear combination of $\boldsymbol{C_3}$ coordinates coming only from $\mathcal{P}^{-}_{\texttt{AND-2}}$ and $\mathcal{P}^{(1)}_{\texttt{AND-2}}$ takes the form of a sum $g(X_1, X_2, X_3, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}_3, \boldsymbol{R}'_2, \boldsymbol{R}'_3) + R^\star$ such that $R^\star \in \boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \boldsymbol{R}'_2$ and $g \perp R^\star$. Eventually, we get that for every $x_3 \hookleftarrow X_3$ and every $(r, r'_2, r'') \hookleftarrow (\boldsymbol{R}_3, \boldsymbol{R}'_3)$, the set of messages $\boldsymbol{C_3}$ is independent of (X_1, X_2) (by Lemma 5.2.8) which implies that $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property with respect to P_3 (Lemma 5.2.4).

We eventually conclude that $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property. ◇

We now prove the privacy of Protocol $\mathcal{P}_{\texttt{AND-3}}$.

Proof. By construction, any message received by a player running $\mathcal{P}_{\texttt{AND-3}}$ is either a message received in $\mathcal{P}^{-}_{\texttt{AND-3}}$ or in $\mathcal{P}_{\texttt{XOR}}$. Moreover, $\mathcal{P}^{-}_{\texttt{AND-3}}$ satisfies the instance-secrecy property with respect to the three players. During $\mathcal{P}_{\texttt{XOR}}$, each player receives a single message. P_1 receives $X_1X_2X_3$. As this variable is functionally independent of any element of the random tapes at input of $\mathcal{P}_{\texttt{AND-3}}$, this player learns nothing new except the protocol output: $\mathcal{P}_{\texttt{AND-3}}$ is private with respect to P_1. Player P_3 receives $X_1X_2X_3 + R''$ with R'' in his random tape $\boldsymbol{R}'_3$: he then learns $X_1X_2X_3$. As shown in the proof of the instance-secrecy of $\mathcal{P}^{-}_{\texttt{AND-3}}$, all messages received by him during $\mathcal{P}^{-}_{\texttt{AND-3}}$ take the form $g(X_1, X_2, X_3, \boldsymbol{R}_1, \boldsymbol{R}_2, \boldsymbol{R}'_2, \boldsymbol{R}'_3) + R^\star$ with $R^\star \in \boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \boldsymbol{R}'_2$ and $g \ \perp R^\star$. We therefore deduce that the sum of any such message with $X_1X_2X_3 + R''$ is still in the form $g(\cdots) + R^\star$ with $R^\star \in \boldsymbol{R}_1 \cup \boldsymbol{R}_2 \cup \boldsymbol{R}'_2$ and $g \perp R^\star$. This implies that P_3 learns nothing about the secrets of the other players, except the final result $X_1X_2X_3$. Player P_2 receives $X_1X_2X_3$ and he knows $X_1X_2X_3 + R''$. He then deduces $X_1X_2X_3$ and R''. Among all the messages he received during $\mathcal{P}^{-}_{\texttt{AND-3}}$, only $X_1X_2X_3 + R'_1(X_3 + R'_2) + R''$ is also protected by R'' (any other combination of received messages takes the form $g(\cdots) + R^\star$ with $R^\star \in (\boldsymbol{R}_1 \cup \boldsymbol{R}'_3)\backslash\{R''\}$. The sum of this message with $X_1X_2X_3 + R''$ gives $R'_1(X_3 + R'_2)$ which is independent of X_3 (even knowing $R'_1 \in \boldsymbol{R}'_2$) as long as R'_2 is unknown. The only other message received by P_2 which functionally depends on R'_2 is $X_3 + R'_2$, but it can be checked that this gives no additional information. We eventually deduce that P_2 learns nothing more than $X_1X_2X_3$ and X_2: $\mathcal{P}_{\texttt{AND-3}}$ is private with respect to P_2. ◇

□

```
Protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$
Require: Players $P_a\{x_a + r, \boldsymbol{R_a}\}, P_b\{x_b + r, \boldsymbol{R_b}\}, P_c\{r\}$.
Ensure: $P_c$ gets $x_a x_b + r^\star$ with $r^\star$ a random bit known by $P_b$ only.

  round 1: $P_a$ and $P_b$ respectively read random bits $r_1, r_2$ and $r_3, r_4$ on their random tapes
  $P_a$ sends $m_1 = x_a + r + r_1 + r_2$ to $P_c$, $r_1$ to $P_b$ and $r_2$ to $P_c$
  $P_b$ sends $m_2 = x_b + r + r_3 + r_4$ to $P_c$, $r_3$ to $P_a$ and $r_4$ to $P_c$

  round 2: $P_c$ sends $m_3 = m_2 + r$ to $P_a$ and $m_4 = m_1 + r$ to $P_b$        ▷ $m_3 = x_b + r_3 + r_4$,
  $m_4 = x_a + r_1 + r_2$
  round 3: $P_b$ reads a random bit $r^\star$ on its random tape
  $P_b$ sends $m_5 = m_4(x_b + r) + r_1(r_3 + r_4) + r^\star$ to $P_a$▷ $m_5 = (x_a + r_1 + r_2)(x_b + r) + r_1(r_3 + r_4) + r^\star$
  round 4: $P_a$ sends $m_6 = m_5 + m_3(r_1 + r_2) + r_2 r_3$ to $P_c$
                                                  ▷ $m_6 = x_a x_b + r_2 r_4 + r(x_a + r_1 + r_2) + r^\star$
  round 5: $P_c$ computes $m_7 = m_6 + r_2 r_4 + r m_4$                         ▷ $m_7 = x_a x_b + r^\star$
```

Interestingly, it is also possible to privately evaluate the MAJORITY function for three players. In dimension 3, this function may be defined $\texttt{MAJORITY}(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$, or equivalently $\texttt{MAJORITY}(x_1, x_2, x_3) = (x_1 + x_2)(x_1 + x_3) + x_1$. We describe herafter a private protocol for the evaluation of this function.

```
Protocol $\mathcal{P}_{\texttt{MAJORITY3}}$
Require: Players $P_1\{x_1, \boldsymbol{R_1}\}, P_2\{x_2, \boldsymbol{R_2}\}, P_3\{x_3, \boldsymbol{R_3}\}$
Ensure: All players get $\texttt{MAJORITY3}(x_1, x_2, x_3)$
    Player $P_1$ reads a random bit $r$ on its random tape
    $P_1$ sends $m_1 = x_1 + r$ to $P_2$ and to $P_3$
    Protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ for $(P_2\{x_1 + x_2 + r, \boldsymbol{R_2}\}, P_3\{x_1 + x_3 + r, \boldsymbol{R_3}\}, P_1\{r\})$
                                          ▷ we get $(P_1\{(x_1 + x_2)(x_1 + x_3) + r^\star\}, P_3\{r^\star\})$
    $P_3$ sends $r^\star$ to $P_1$
     $P_1$ computes $m_2 = (x_1 + x_2)(x_1 + x_3) + r^\star + r^\star + x_1$
                                                  ▷ $m_2 = (x_1 + x_2)(x_1 + x_3) + x_1$
    $P_1$ broadcasts $m_2$
```

The correctness of $\mathcal{P}_{\texttt{MAJORITY3}}$ can be checked thanks to the comments inserted in the description. The privacy may moreover be deduced from the forthcoming Lemma 5.6.5.

5.5 Private Evaluation of AND-n for $n = 4, 5$

We follow the same approach as in Section 5.4 to build protocols $\mathcal{P}_{\texttt{AND-4}}$ and $\mathcal{P}_{\texttt{AND-5}}$ enabling the private evaluation of AND-n for $n = 4, 5$. Their respective randomness complexity is 11 and 13.

To this end, we introduce new protocols $\mathcal{P}^{(3)}_{\texttt{AND2}}$, $\mathcal{P}^{-}_{\texttt{AND4}}$, $\mathcal{P}^{(4)}_{\texttt{AND2}}$ and $\mathcal{P}^{-}_{\texttt{AND5}}$.

The following protocol $\mathcal{P}^{(3)}_{\texttt{AND2}}$ allows three players to output the encoding of the multiplication of two values, one of them being itself encoded. This protocol is 4-random.

```
Protocol $\mathcal{P}^{(3)}_{\texttt{AND2}}$
Require: players $P_a\{x_a + r, \boldsymbol{R_a}\}, P_b\{x_b, \boldsymbol{R_b}\}, P_c\{r, \boldsymbol{R_c}\}$
Ensure: $P_a$ gets $x_a x_b + r'$ with $r'$ a random bit known by $P_b$ only
    round 1: Players $P_a$ and $P_b$ read a random bit $r_1$ and $r_2$ on their respective random tapes
    $P_a$ sends $m_1 = x_a + r + r_1$ to $P_c$ and $r_1$ to $P_b$
    $P_b$ sends $m_2 = x_b + r_2$ to $P_a$ and $r_2$ to $P_c$

    round 2: Player $P_c$ reads a random bit $r_3$ on its random tape
    $P_c$ sends $m_3 = m_1 + r + r_3$ to $P_b$ and $r_3$ to $P_a$          ▷ $m_3 = x_a + r_1 + r_3$
    round 3: Player $P_b$ reads a random bit $r'$ on its random tape
    $P_b$ sends $m_4 = m_3 x_b + r_1 r_2 + r'$ to $P_c$          ▷ $m_4 = x_a x_b + r_1(x_b + r_2) + r_3 x_b + r'$
    round 4: $P_c$ sends $m_5 = m_4 + r_2 r_3$ to $P_a$          ▷ $m_5 = x_a x_b + (x_b + r_2)(r_1 + r_3) + r'$
    round 5: $P_a$ computes $m_6 = m_5 + (r_1 + r_3)(x_b + r_2)$          ▷ $m_6 = x_a x_b + r'$
```

The following sequence of two protocols $\mathcal{P}^{-}_{\texttt{AND4}}$ and $\mathcal{P}_{\texttt{AND4}}$ allows four players to privately compute the multiplication of their secret inputs. The four players P_1, P_2, P_3 and P_4 are respectively provided with the random tapes $\boldsymbol{R}_1$, $\boldsymbol{R}_2 \cup \boldsymbol{R}'_2$, $\boldsymbol{R}_3 \cup \boldsymbol{R}'_3$ and $\boldsymbol{R}'_4$. Random tapes $\boldsymbol{R}_i$ (resp. $\boldsymbol{R}'_i$) will be used when executing protocol $\mathcal{P}^{-}_{\texttt{AND-2}}$ (resp. $\mathcal{P}^{(3)}_{\texttt{AND-2}}$). Before the execution of the protocol, each player fills his random tape such that: $\boldsymbol{R}_1 \leftarrow \{r_1, r'\}$, $\boldsymbol{R}_2 \leftarrow \{r'_1, r_2, r\}$, $\boldsymbol{R}_3 \leftarrow \{r'_2, r''\}$, $\boldsymbol{R}'_2 \leftarrow \{r''_1\}$, $\boldsymbol{R}'_4 \leftarrow \{r''_2, r'''\}$ and $\boldsymbol{R}'_3 \leftarrow \{r''_3\}$. These protocols are 11-random.

```
Protocol $\mathcal{P}^{-}_{\texttt{AND4}}$
Require: players $P_1\{x_1, \boldsymbol{R_1}\}$,$P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2}\}$,$P_3\{x_3, \boldsymbol{R_3} \cup \boldsymbol{R'_3}\}$, $P_4\{x_4, \boldsymbol{R'_4}\}$
Ensure: $P_2$ gets $x_1 x_2 x_3 x_4 + r'''$ with $r'''$ a random bit known by only $P_4$
    Protocol $\mathcal{P}^{-}_{\texttt{AND3}}$ for $(P_1\{x_1, \boldsymbol{R_1}\}, P_2\{x_2, \boldsymbol{R_2}\}, P_3\{x_3, \boldsymbol{R_3}\})$
                                        ▷ we get $(P_2\{x_1 x_2 x_3 + r''\}, P_3\{r''\})$
    Protocol $\mathcal{P}^{(3)}_{\texttt{AND2}}$ for $(P_2\{x_1 x_2 x_3 + r'', \boldsymbol{R'_2}\}, P_4\{x_4, \boldsymbol{R'_4}\}, P_3\{r'', \boldsymbol{R'_3}\})$
                                        ▷ we get $(P_2\{x_1 x_2 x_3 x_4 + r'''\}, P_4\{r'''\})$
```

```
Protocol $\mathcal{P}_{\texttt{AND4}}$
Require: players $P_1\{x_1, \boldsymbol{R_1}\}$,$P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2}\}$,$P_3\{x_3, \boldsymbol{R_3} \cup \boldsymbol{R'_3}\}$, $P_4\{x_4, \boldsymbol{R'_4}\}$
Ensure: $P_1, P_2, P_3, P_4$ get $x_1 x_2 x_3 x_4$

  Protocol $\mathcal{P}^{-}_{\texttt{AND4}}$ for $P_1\{x_1, \boldsymbol{R_1}\}$, $P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2}\}$ and $P_3\{x_3, \boldsymbol{R_3} \cup \boldsymbol{R'_3}\}, P_4\{x_4, \boldsymbol{R'_4}\}$
                                        ▷ we get $(P_2\{x_1 x_2 x_3 x_4 + r'''\}, P_3\{r'''\})$
  Protocol $\mathcal{P}_{\texttt{XOR}}$ for $P_2\{x_1 x_2 x_3 x_4 + r'''\}$, $P_3\{r'''\}$ , $P_1$ and $P_4$
                    ▷ we get $(P_1\{x_1 x_2 x_3 x_4\}, P_2\{x_1 x_2 x_3 x_4\}, P_3\{x_1 x_2 x_3 x_4\}, P_4\{x_1 x_2 x_3 x_4\}\})$
```

The following protocol $\mathcal{P}^{(4)}_{\texttt{AND2}}$ allows three players to compute the multiplication of two encoded secrets. This protocol is 5-random.

```
Protocol $\mathcal{P}^{(4)}_{\texttt{AND2}}$
Require: players $P_a\{x_a + r, \boldsymbol{R_a}\}, P_b\{x_b + r', \boldsymbol{R_b}\}, P_c\{r, r'\}$
Ensure: $P_c$ gets $x_a x_b + r''$ with $r''$ a random bit known by $P_b$ only
    round 1: $P_a$ and $P_b$ respectively read random bits $r_1, r_2$ and $r_3, r_4$ on their random tapes
    $P_a$ sends $m_1 = x_a + r + r_1 + r_2$ to $P_c$, $r_1$ to $P_b$ and $r_2$ to $P_c$
    $P_b$ sends $m_2 = x_b + r' + r_3 + r_4$ to $P_c$, $r_3$ to $P_a$ and $r_4$ to $P_c$
    round 2: $P_c$ sends $m_3 = m_2 + r$ to $P_a$ and $m_4 = m_1 + r'$ to $P_b$
                                        ▷ $m_3 = x_b + r_3 + r_4$, $m_4 = x_a + r_1 + r_2$
    round 3: $P_b$ reads a random bit $r''$ on its random tape
    $P_b$ sends $m_5 = m_4(x_b + r') + r_1(r_3 + r_4) + r''$ to $P_a$
                                        ▷ $m_5 = (x_a + r_1 + r_2)(x_b + r') + r_1(r_3 + r_4) + r''$
    round 4: $P_a$ sends $m_6 = m_5 + m_3(r_1 + r_2) + r_2 r_3$ to $P_c$
                                        ▷ $m_6 = x_a x_b + r_2 r_4 + r'(x_a + r_1 + r_2) + r''$
    round 5: $P_c$ computes $m_7 = m_6 + r_2 r_4 + r m_4$      ▷ $m_7 = x_a x_b + r''$
```

The following sequence of two protocols $\mathcal{P}^{-}_{\texttt{AND5}}$ and $\mathcal{P}_{\texttt{AND5}}$ allow five players to privately compute the multiplication of their secrets. The five players P_1, P_2, P_3, P_4 and P_5 are respectively provided with the random tapes $\boldsymbol{R}_1 \cup \boldsymbol{R}'_1$, $\boldsymbol{R}_2 \cup \boldsymbol{R}''_2$, $\boldsymbol{R}''_3$, $\boldsymbol{R}_4 \cup \boldsymbol{R}'_4 \cup \boldsymbol{R}''_4$ and $\boldsymbol{R}_5$. Random tapes $\boldsymbol{R}_i$ will be used for the first execution of protocol $\mathcal{P}^{-}_{\texttt{AND-2}}$, random tapes $\boldsymbol{R}'_i$ will be used for the second execution and random tapes $\boldsymbol{R}''_i$ will be used for $\mathcal{P}^{(3)}_{\texttt{AND-2}}$. Before the execution of the protocol, each player fills his random tape such that: $\boldsymbol{R}_1 = \{r_1\}$ $\boldsymbol{R}_2 = \{r_2, r\}$ $\boldsymbol{R}_4 = \{r_1\}$ $\boldsymbol{R}_5 = \{r_2, r'\}$ $\boldsymbol{R}'_1 = \{r'_1, r'_2\}$ $\boldsymbol{R}'_4 = \{r'_3, r'_4, r''\}$ $\boldsymbol{R}''_2 = \{r''_1\}$ $\boldsymbol{R}''_3 = \{r''_2, r'''\}$ $\boldsymbol{R}''_4 = \{r''_3\}$. These protocols are 13-random.

```
Protocol $\mathcal{P}^{-}_{\texttt{AND5}}$
Require: players $P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R'_1}\}$,$P_b\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2} \cup \boldsymbol{R''_2}\}$,$P_3\{x_3, \boldsymbol{R'_3}\}, P_4\{x_4, \boldsymbol{R_4} \cup \boldsymbol{R'_4}\}, P_5\{x_5, \boldsymbol{R_5}\}$
Ensure: $P_2$ gets $x_1 x_2 x_3 x_4 x_5 + r'''$ with $r'''$ a random bit known by only $P_3$
    Protocol $\mathcal{P}^{-}_{\texttt{AND2}}$ for $(P_1\{x_1, \boldsymbol{R_1}\}, P_2\{x_2, \boldsymbol{R_2}\}, P_3)$
                                        ▷ we get $(P_1\{x_1 x_2 + r\}, P_2\{r\})$
    Protocol $\mathcal{P}^{-}_{\texttt{AND2}}$ for $(P_4\{x_4, \boldsymbol{R_4}\}, P_5\{x_5, \boldsymbol{R_5}\}, P_3)$
                                        ▷ we get $(P_4\{x_4 x_5 + r'\}, P_5\{r'\})$
    $P_5$ sends $r'$ to $P_2$
    Protocol $\mathcal{P}^{(4)}_{\texttt{AND2}}$ for $(P_1\{x_1 x_2 + r, \boldsymbol{R'_1}\}, P_4\{x_4 x_5 + r', \boldsymbol{R'_4}\}, P_2\{r, r'\})$
                                        ▷ we get $(P_2\{x_1 x_2 x_4 x_5 + r''\}, P_4\{r''\})$
    Protocol $\mathcal{P}^{(3)}_{\texttt{AND2}}$ for $(P_2\{x_1 x_2 x_4 x_5 + r'', \boldsymbol{R''_2}\}, P_3\{x_3, \boldsymbol{R''_3}\}, P_4\{r'', \boldsymbol{R''_4}\})$
                                        ▷ we get $(P_2\{x_1 x_2 x_3 x_4 x_5 + r'''\}, P_3\{r'''\})$
```

```
Protocol $\mathcal{P}_{\mathtt{AND5}}$
Require: players $P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R'_1}\}$,$P_b\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2} \cup \boldsymbol{R''_2}\}$,$P_3\{x_3, \boldsymbol{R'_3}\}$, $P_4\{x_4, \boldsymbol{R_4} \cup \boldsymbol{R'_4}\}$, $P_5\{x_5, \boldsymbol{R_5}\}$
Ensure: $P_1, P_2, P_3, P_4, P_5$ get $x_1x_2x_3x_4x_5$

    Protocol $\mathcal{P}^{-}_{\mathtt{AND5}}$ for $(P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R'_1}\}$, $P_2\{x_2, \boldsymbol{R_2} \cup \boldsymbol{R'_2} \cup \boldsymbol{R''_2}\}$, $P_3\{x_3, \boldsymbol{R'_3}\}$, $P_4\{x_4, \boldsymbol{R_4} \cup \boldsymbol{R'_4}\}$,$P_5\{x_5, \boldsymbol{R_5}\})$
                                        ▷ we get $(P_2\{x_1x_2x_3x_4x_5 + r'''\}, P_3\{r'''\})$
    Protocol $\mathcal{P}_{\mathtt{XOR}}$ for $(P_2\{x_1x_2x_3x_4x_5 + r'''\}$, $P_3\{r'''\}$,$P_1$,$P_4$,$P_5)$
                                        ▷ we get
    $(P_1\{x_1x_2x_3x_4x_5\}, P_2\{x_1x_2x_3x_4x_5\}, P_3\{x_1x_2x_3x_4x_5\}, P_4\{x_1x_2x_3x_4x_5\}, P_5\{x_1x_2x_3x_4x_5\})$
```

The proofs proposed in Section 5.4 can be easily adapted to argue that $\mathcal{P}^{(3)}_{\mathtt{AND2}}$, $\mathcal{P}^{(4)}_{\mathtt{AND2}}$, $\mathcal{P}^{-}_{\mathtt{AND4}}$ and $\mathcal{P}^{-}_{\mathtt{AND5}}$ satisfy the instance-secrecy property, and that $\mathcal{P}_{\mathtt{AND4}}$ and $\mathcal{P}_{\mathtt{AND5}}$ are private.

5.6 Private Evaluation of AND-n for any $n = 3 \cdot 2^j$

The AND function has a *linear size circuit* (indeed the AND of n bits can be computed by a circuit of n binary AND gates). This observation and the analysis in [KOR99] imply that this function can be privately evaluated by any arbitrarily high number n of players with a d-random protocol, where d stays constant with respect to n. Nonetheless, a straightforward adaptation of [KOR99] leads to a 73-random protocol. In this section, we aim at improving this randomness complexity.

For such a purpose, we deduce a new protocol $\mathcal{P}_{\mathtt{AND\text{-}n}}$ from the protocols $\mathcal{P}^{-}_{\mathtt{AND3}}$, $\mathcal{P}^{(2)}_{\mathtt{AND2}}$ and $\mathcal{P}_{\mathtt{XOR}}$ introduced in the previous section.

5.6.1 Core Idea

The description of the new protocol $\mathcal{P}_{\mathtt{AND\text{-}n}}$ slightly differs depending on the number of players n taking part in it. The simplest version requires that n takes the form 3×2^j with $j \in \mathbb{N}$. In what follows, we present the protocol in this ideal case. Also, since the case $n = 3$ has already been addressed in Section 5.4 (Protocol $\mathcal{P}_{\mathtt{AND\text{-}3}}$), n is assumed here to be strictly greater than 3 (*ie.* $n = 3 \times 2^j$ with $j \geqslant 1$). Adaptations to deal with any number n of players are described further in Section 5.7.

The first case we have to consider is the case of $n = 6$ players. A naive idea is to divide these players into two sets of 3 players and make them perform the $\mathcal{P}^{-}_{\mathtt{AND3}}$ protocol in parallel before computing the product of the encoded results. It is indeed possible to use the same random bits in the two parallel executions of the protocol while maintaining the privacy. We then use the protocol $\mathcal{P}^{(2)}_{\mathtt{AND2}}$ (with new random bits) to obtain the result (see Figure 5.2), thus building an efficient protocol which is 11-random.

One can generalize this idea recursively for any integer n of the form 3×2^j by first performing in parallel 2^j runs of the protocol $\mathcal{P}^{-}_{\mathtt{AND3}}$ and then multiplying the results using a complete binary tree of depth j where each node runs a $\mathcal{P}^{-}_{\mathtt{AND2}}$ protocol (using three players). This idea further gives a way to 1-privately evaluate the AND-n function with $n = 3 \cdot 2^j$ player in $11 + (j - 1) = O(\log(n))$ random bits. To achieve constant randomness complexity, we must remark that it is possible to recycle the random bits used at each level of the tree without breaking the privacy of the protocol. For $n = 12$, it is also possible to reuse some random bits from the first level in the final second level by a careful selection of the players involved at each level (see Figure 5.3). We obtain a 12-random protocol for the AND-n function for $n = 12$.

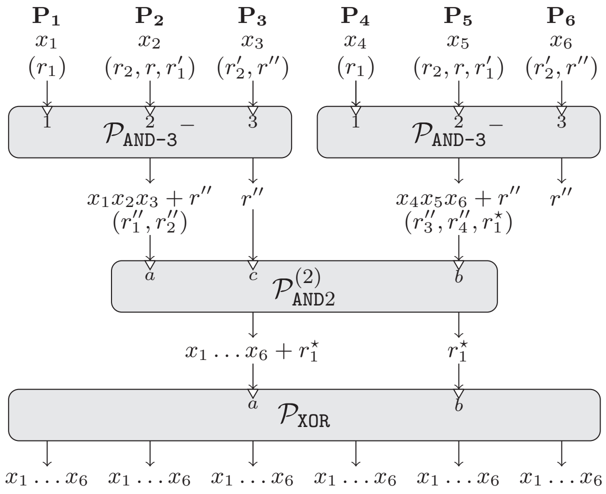

$\mathbf{P_1}$ x_1 (r_1)
$\mathbf{P_2}$ x_2 (r_2, r, r_1')
$\mathbf{P_3}$ x_3 (r_2', r'')
$\mathbf{P_4}$ x_4 (r_1)
$\mathbf{P_5}$ x_5 (r_2, r, r_1')
$\mathbf{P_6}$ x_6 (r_2', r'')
1 2 3 $\mathcal{P}^-_{\texttt{AND-3}}$
1 2 3 $\mathcal{P}^-_{\texttt{AND-3}}$
$x_1x_2x_3 + r''$ (r_1'', r_2'')
r''
$x_4x_5x_6 + r''$ $(r_3'', r_4'', r_1^\star)$
r''
a c b $\mathcal{P}^{(2)}_{\texttt{AND2}}$
$x_1 \dots x_6 + r_1^\star$
$r_1^\star$
a b $\mathcal{P}_{\texttt{XOR}}$
$x_1 \dots x_6$ $x_1 \dots x_6$ $x_1 \dots x_6$ $x_1 \dots x_6$ $x_1 \dots x_6$ $x_1 \dots x_6$

Figure 5.2: Description of the protocol $\mathcal{P}_{\texttt{AND-n}}$ for $n = 6$

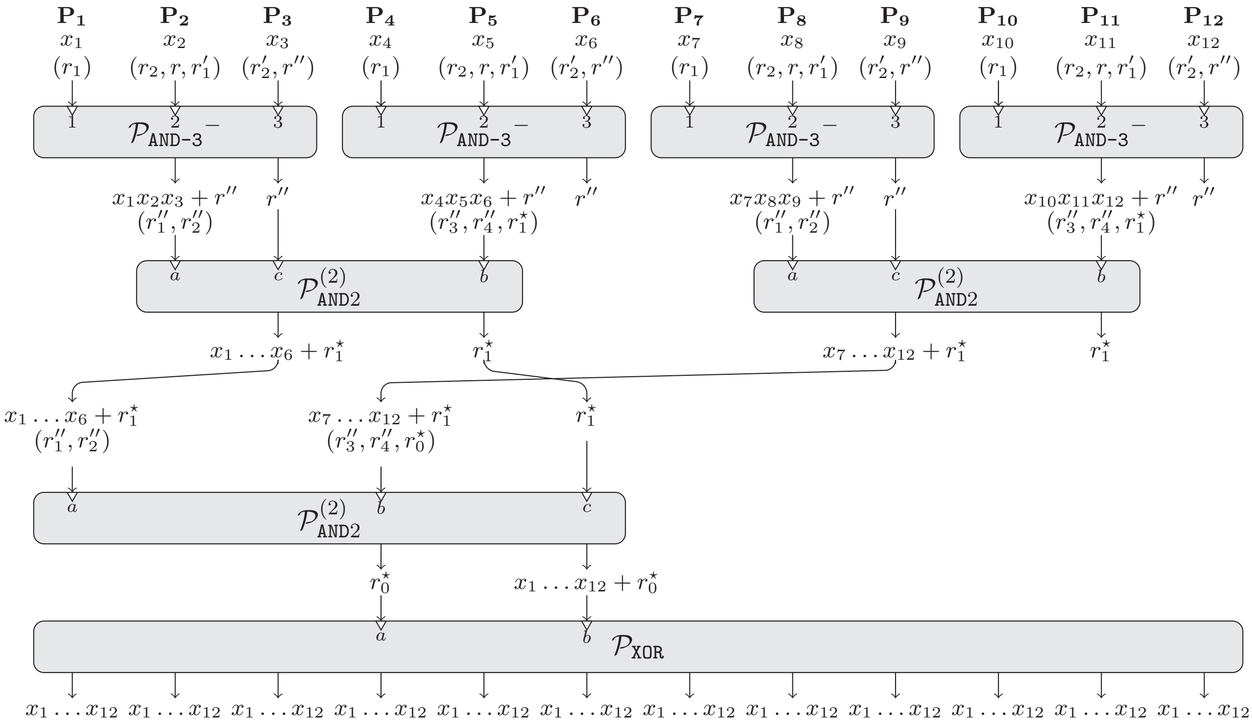

$\mathbf{P_1}$ x_1 (r_1)
$\mathbf{P_2}$ x_2 (r_2, r, r_1')
$\mathbf{P_3}$ x_3 (r_2', r'')
$\mathbf{P_4}$ x_4 (r_1)
$\mathbf{P_5}$ x_5 (r_2, r, r_1')
$\mathbf{P_6}$ x_6 (r_2', r'')
$\mathbf{P_7}$ x_7 (r_1)
$\mathbf{P_8}$ x_8 (r_2, r, r_1')
$\mathbf{P_9}$ x_9 (r_2', r'')
$\mathbf{P_{10}}$ x_{10} (r_1)
$\mathbf{P_{11}}$ x_{11} (r_2, r, r_1')
$\mathbf{P_{12}}$ x_{12} (r_2', r'')
1 2 3 $\mathcal{P}^-_{\texttt{AND-3}}$
$x_1x_2x_3 + r''$ (r_1'', r_2'')
r''
$x_4x_5x_6 + r''$ $(r_3'', r_4'', r_1^\star)$
r''
$x_7x_8x_9 + r''$ (r_1'', r_2'')
r''
$x_{10}x_{11}x_{12} + r''$ $(r_3'', r_4'', r_1^\star)$
r''
a c b $\mathcal{P}^{(2)}_{\texttt{AND2}}$
$x_1 \dots x_6 + r_1^\star$
$r_1^\star$
$x_7 \dots x_{12} + r_1^\star$
$r_1^\star$
$x_1 \dots x_6 + r_1^\star$ (r_1'', r_2'')
$x_7 \dots x_{12} + r_1^\star$ $(r_3'', r_4'', r_0^\star)$
$r_1^\star$
a b c $\mathcal{P}^{(2)}_{\texttt{AND2}}$
$r_0^\star$
$x_1 \dots x_{12} + r_0^\star$
a b $\mathcal{P}_{\texttt{XOR}}$
$x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$ $x_1 \dots x_{12}$

Figure 5.3: Description of the protocol $\mathcal{P}_{\texttt{AND-n}}$ for $n = 12$

The final idea is that there is no need to use a fresh random bit for each level of the tree. Indeed, one can use one bit for nodes at even depth and another bit for nodes at odd depth, thus obtaining a 12-random protocol for all such n. In order to transform this intuition into a concrete protocol, we have to check that it is always possible to find *teams* of 3 players to perform the $\mathcal{P}^-_{\texttt{AND2}}$ protocols and that the reuse of the random bits do not hinder the security of the protocol.

This check, as long as the correctness and privacy proofs, are done in Section 5.6.2. To enable these analyses, the next section presents our new protocol in a formal (recursive) way.

5.6.2 Formal Description

Because our construction is recursive, we need two versions of the protocol: the first version (used in the recursion) is called $\mathcal{P}^{-}_{\texttt{AND-n}}$ and provides the players with an encoding of the evaluation, whereas the second version $\mathcal{P}_{\texttt{AND-n}}$ allows all of them to know the evaluation result (it will be used only once after the recursion). Moreover, as the number of random bits must be kept constant w.r.t. n, we allow the players to work with copies of the same limited sets of random bits during the protocol execution. To grant this possibility, before the execution $\mathcal{P}^{-}_{\texttt{AND-n}}$ and $\mathcal{P}_{\texttt{AND-n}}$, all the players P_i start by running a protocol enabling to generate all the needed random bits and to distribute them to the appropriate players. After this initial protocol, each player P_i is provided with two random tapes $\boldsymbol{R}_i$ and $\boldsymbol{R}^\star_i$ defined such that $\boldsymbol{R}_i = \mathcal{E}_{i \bmod 3}$ and

$$\boldsymbol{R}^\star_i = \begin{cases} \mathcal{F}_1 & \text{if } i \equiv 1, 2 \pmod 6 \\ \mathcal{F}^1_2 & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is odd} \\ \mathcal{F}^0_2 & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is even} \\ \mathcal{F}^1_2 & \text{if } i \equiv 5 \pmod 6 \\ \mathcal{F}_0 & \text{otherwise} \end{cases}, \tag{5.1}$$

with $\mathcal{E}_1 = \{r_1\}$, $\mathcal{E}_2 = \{r_2, r, r'_1\}$, $\mathcal{E}_0 = \{r'_2, r''\}$, $\mathcal{F}_0 = \emptyset$, $\mathcal{F}_1 = \{r''_1, r''_2\}$, $\mathcal{F}^0_2 = \{r''_3, r''_4, r^\star_0\}$ and $\mathcal{F}^1_2 = \{r''_3, r''_4, r^\star_1\}$.

The n players $(P_i\{x_i, \boldsymbol{R}_i \cup \boldsymbol{R}^\star_i\})_{1 \leqslant i \leqslant n}$ can now process the following recursive protocols $\mathcal{P}^{-}_{\texttt{AND-n}}$ and $\mathcal{P}_{\texttt{AND-n}}$, where players indexation is done thanks to two functions τ and τ' defined on $\mathbb{N}^2$ by:

$$\tau(t,n) = \begin{cases} (t+5, t+3) & \text{if } n = 12 \\ (t + \frac{n}{4} - 2, t + \frac{n}{4}) & \text{otherwise} \end{cases} \quad \text{and} \quad \tau'(t,n) = \begin{cases} (t+11, t+9) & \text{if } n = 12 \\ (t + \frac{3n}{4} - 2, t + \frac{3n}{4}) & \text{otherwise} \end{cases}.$$

Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$

Require: Players $P_{t+1}\{x_{t+1}, \boldsymbol{R}_{t+1} \cup \boldsymbol{R}^\star_{t+1}\}, \cdots, P_{t+n}\{x_{t+n}, \boldsymbol{R}_{t+n} \cup \boldsymbol{R}^\star_{t+n}\}$ with n and t of the form $3 \cdot 2^j$ for $j \geqslant 1$.

Ensure: Player $P_{t+\frac{n}{2}}$ gets $\prod_{l=1}^{n} x_{t+l} + r^\star_{\overline{k}}$. The random bit $r^\star_{\overline{k}}$ is known by other players but not by $P_{t+\frac{n}{2}}$.

```
if $n = 6$ then                                                              ▷ $n = 6$ players
    Protocol $\mathcal{P}^{-}_{\texttt{AND3}}$ for $(P_{t+1}\{x_{t+1}, \boldsymbol{R}_{t+1}\}, P_{t+2}\{x_{t+2}, \boldsymbol{R}_{t+2}\}, P_{t+3}\{x_{t+3}, \boldsymbol{R}_{t+3}\})$
                                       ▷ we get $(P_{t+2}\{x_{t+1}x_{t+2}x_{t+3} + r''\}, P_{t+3}\{r''\})$
    Protocol $\mathcal{P}^{-}_{\texttt{AND3}}$ for $(P_{t+4}\{x_{t+4}, \boldsymbol{R}_{t+4}\}, P_{t+5}\{x_{t+5}, \boldsymbol{R}_{t+5}\}, P_{t+6}\{x_{t+6}, \boldsymbol{R}_{t+6}\})$
                                       ▷ we get $(P_{t+5}\{x_{t+4}x_{t+5}x_{t+6} + r''\}, P_{t+6}\{r''\})$
    Protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ for $(P_{t+2}\{x_{t+1}x_{t+2}x_{t+3}+r'', \boldsymbol{R}^\star_{t+2}\}, P_{t+5}\{x_{t+4}x_{t+5}x_{t+6}+r'', \boldsymbol{R}^\star_{t+5}\}, P_{t+3}\{r''\})$
                                       ▷ we get $(P_{t+5}\{\prod_{l=1}^{6} x_{t+l} + r^\star_1\}, P_{t+3}\{r^\star_1\})$

else                                                                         ▷ $n > 6$ players
    Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $(P_{t+1}\{x_{t+1}, \boldsymbol{R}_{t+1} \cup \boldsymbol{R}^\star_{t+1}\}, \cdots, P_{t+\frac{n}{2}}\{x_{t+\frac{n}{2}}, \boldsymbol{R}_{t+\frac{n}{2}} \cup \boldsymbol{R}^\star_{t+\frac{n}{2}+1}\})$
                                       ▷ we get $(P_d\{\prod_{l=1}^{\frac{n}{2}} x_{t+l} + r^\star_k\}, P_e\{r^\star_k\})$, with $(d, e) = \tau(t, n)$
    Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $(P_{t+\frac{n}{2}+1}\{x_{t+\frac{n}{2}+1}, \boldsymbol{R}_{t+\frac{n}{2}+1} \cup \boldsymbol{R}^\star_{t+\frac{n}{2}+1}\}, \cdots, P_{t+n}\{x_{t+n}, \boldsymbol{R}_{t+n} \cup \boldsymbol{R}^\star_{t+n}\})$
                                       ▷ we get $(P_f\{\prod_{l=\frac{n}{2}}^{n} x_{t+l} + r^\star_k\}, P_g\{r^\star_k\})$, with $(f, g) = \tau'(t, n)$
    $a \leftarrow t + \frac{n}{2} - 5$, $b \leftarrow t + \frac{n}{2} - 2$ and $c \leftarrow t + \frac{n}{2}$
    $P_d$ sends $\prod_{l=1}^{\frac{n}{2}} x_{t+l} + r^\star_k$ to $P_a$, $P_f$ sends $\prod_{l=\frac{n}{2}}^{n} x_{t+l} + r^\star_k$ to $P_b$, and $P_e$ sends $r^\star_k$ to $P_c$
    Protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ for $(P_a\{\prod_{l=1}^{\frac{n}{2}} x_{t+l} + r^\star_k, \boldsymbol{R}^\star_{\boldsymbol{a}}\}, P_b\{\prod_{l=\frac{n}{2}}^{n} x_{t+l} + r^\star_k, \boldsymbol{R}^\star_{\boldsymbol{b}}\}, P_c\{r^\star_k, \boldsymbol{R}^\star_c\})$
                                       ▷ we get $P_c\{\prod_{l=1}^{n} x_{t+l} + r^\star_{\overline{k}}\}$ with $r^\star_{\overline{k}} \in \boldsymbol{R}^\star_b$
```

Notation. In Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$, the index k satisfies $\overline{k} = \log_2(\frac{n}{3}) \bmod 2$ by construction of the random tapes.

Protocol $\mathcal{P}_{\texttt{AND-n}}$

Require: n players $P_1\{x_1, \boldsymbol{R}_1 \cup \boldsymbol{R}^\star_1\}, \cdots, P_n\{x_n, \boldsymbol{R}_n \cup \boldsymbol{R}^\star_n\}$ with n in the form $3 \cdot 2^j$ for $j \geqslant 1$.

Ensure: All the players get $\prod_{i=1}^{n} x_i$

```
Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $(P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R^\star_1}\}, \cdots, P_n\{x_n, \boldsymbol{R_n} \cup \boldsymbol{R^\star_n}\})$
                                       ▷ we get $P_{\frac{n}{2}}\{\prod_{i=1}^{n} x_i + r^\star_{\overline{k}}\}$ with $r^\star_{\overline{k}} \in \boldsymbol{R}^\star_{\frac{n}{2}-2}$
Protocol $\mathcal{P}_{\texttt{XOR}}$ for $(\cdots, P_{\frac{n}{2}-2}\{r^\star_{\overline{k}}\}, \cdots, P_{\frac{n}{2}}\{\prod_{i=1}^{n} x_i + r^\star_{\overline{k}}\}, \cdots)$
                                       ▷ we get $(P_1\{\prod_{i=1}^{n} x_i\}, \cdots, P_n\{\prod_{i=1}^{n} x_i\})$
```

The recursive protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ has been built such that each player executes $\mathcal{P}^{-}_{\texttt{AND-3}}$ only once, and protocol $\mathcal{P}^{(2)}_{\texttt{AND-2}}$ at most once. In the following, we show that it is indeed the case, and we use this property to prove the protocol correctness.

According to our description, the protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $n = 3 \cdot 2^j$ for some integer $j \geqslant 1$, splits the n players into 2^{j-1} teams made of 6 players. Among these 6 players, three players run the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ to obtain an encoded form of the `AND` of their 6 entries. These players, with indices in residue classes 2, 3 and 5 modulo 6, run the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ with random tapes $\mathcal{F}_1 = \{r''_1, r''_2\}$, $\mathcal{F}_0 = \emptyset$ and

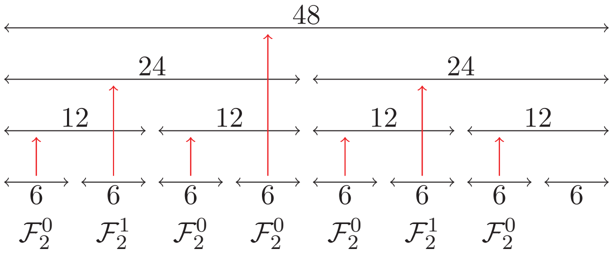


Figure 5.4: Illustration of the teams involved in $\mathcal{P}^{(2)}_{\texttt{AND2}}$ runs in $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $n = 48$

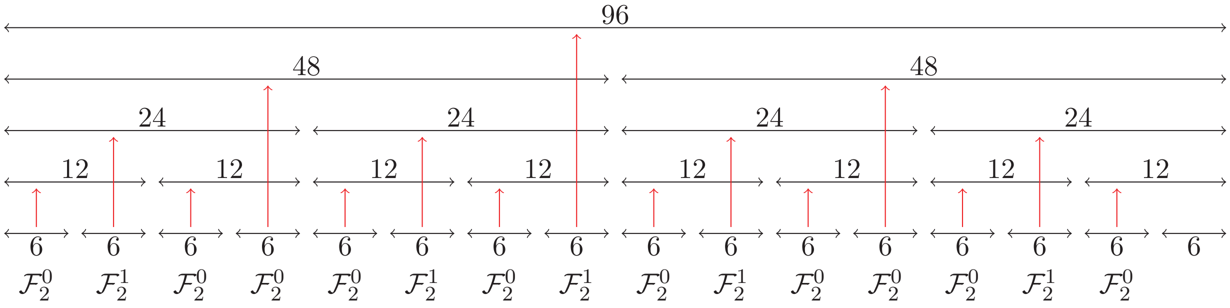


Figure 5.5: Illustration of the teams involved in $\mathcal{P}^{(2)}_{\texttt{AND2}}$ runs in $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $n = 96$

$\mathcal{F}_2^1 = \{r_3'', r_4'', r_1^\star\}$ (respectively). The three other players will be involved later in another execution of the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ in the recursive evaluation of the binary tree.

The three *fresh* players in teams with odd indices are involved in protocols that compute an encoded form of the AND-value of 12 private inputs (*i.e.* the first involved in Step 15 of protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ or equivalently in the maximum depth of the binary tree). These players, with indices in residue classes 1, 4 and 6 modulo 12, run the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ with random tapes $\mathcal{F}_1 = \{r_1'', r_2''\}$, $\mathcal{F}_2^0 = \{r_3'', r_4'', r_0^\star\}$ and $\mathcal{F}_0 = \emptyset$ (respectively). The players in teams with even indices run the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ later in the recursion. For instance, the teams with indices in residue class 2 modulo 4 are involved in the next level of the recursion (*i.e.* the one which computes the AND-value of 24 private inputs). Figures 5.4 and 5.5 illustrate the teams involved in each execution of the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ following this idea. They indicate for each team which random tape among $\mathcal{F}_2^0$ and $\mathcal{F}_2^1$ is actually used in the execution of the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$.

We now prove the correctness of $\mathcal{P}_{\texttt{AND-n}}$. For such a purpose, we prove that protocols $\mathcal{P}^{-}_{\texttt{AND3}}$ and $\mathcal{P}^{(2)}_{\texttt{AND2}}$ are always run with a triplet of players that have the required random tapes.

Proposition 5.6.1. *For any number of players $n = 3 \times 2^j$ running $\mathcal{P}^{-}_{\texttt{AND-n}}$, the players P_{t+1}, P_{t+2}, P_{t+3}, P_{t+4}, P_{t+5}, P_{t+6} with $t \in \{6i; i \in [0; \frac{n}{6} - 1]\}$ possess the required random tapes to run $\mathcal{P}^{-}_{\texttt{AND3}}$.*

Proof. At each *terminal point* of the recursion, Protocol $\mathcal{P}^{-}_{\texttt{AND3}}$ is executed by a triplet of players which is either $(P_{t+1}, P_{t+2}, P_{t+3})$ or $(P_{t+4}, P_{t+5}, P_{t+6})$ with $t \in \{6i; i \in [0; \frac{n}{6} - 1]\}$. By construction, each player P_{t+j} runs $\mathcal{P}^{-}_{\texttt{AND3}}$ with the random set $\boldsymbol{R}_{t+j}$ defined such that $\boldsymbol{R}_{t+j} \doteq \mathcal{E}_{t+j \bmod 3}$ (see the definition of the $\boldsymbol{R}_i$). Since, by construction, t equals 0 modulo 3, then the players P_{t+1}, P_{t+2}, P_{t+3} (resp. P_{t+4}, P_{t+5}, P_{t+6}) hold respectively the random sets $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_0$. Consequently, the two processings of $\mathcal{P}^{-}_{\texttt{AND3}}$ can be performed, leading to the expected encoding $(x_{t+1}x_{t+2}x_{t+3} + r'', r'')$ (resp. $(x_{t+4}x_{t+5}x_{t+6} + r'', r'')$). ◇

Proposition 5.6.2. *For any number of players $n = 3 \times 2^j$ running $\mathcal{P}^{-}_{\textit{AND-n}}$, the triplet of players $P_{t+2}, P_{t+3}, P_{t+5}$ with $t \in \{6i; i \in [0; \frac{n}{6} - 1]\}$ possesses the required random tapes to run the protocol $\mathcal{P}^{(2)}_{\textit{AND2}}$ at Step 6.*

Proof. At each *terminal point* of the recursion, Protocol $\mathcal{P}'_{\texttt{AND2}}$ is run by a triplet of players $(P_{t+2}, P_{t+3}, P_{t+5})$ such that $t \in \{6i; i \in [0; \frac{n}{6} - 1]\}$. Since t is by construction a multiple of 6, we have that $t+2 = 2 \bmod 6, t+3 = 3 \bmod 6, t+5 = 5 \bmod 6$. Thus, according to the definition of $\boldsymbol{R}^\star_i$ (see Relation Equation (5.1)), players $P_{t+2}, P_{t+5}, P_{t+3}$ indeed possess the random sets $\mathcal{F}_1, \mathcal{F}^1_2, \mathcal{F}_0$. Consequently, they can perform $\mathcal{P}^{(2)}_{\texttt{AND2}}$, leading to the encoding $(\prod_{l=1}^{6} x_{t+l} + r^\star_1, r^\star_1)$. ◇

Proposition 5.6.3. *For any number of players $n = 3 \times 2^j$ running $\mathcal{P}^{-}_{\textit{AND-n}}$, the protocol $\mathcal{P}^{(2)}_{\textit{AND2}}$ at Step 15 is always executed by a triplet of players P_a, P_b and P_c who possesses the required random tapes.*

Proof. Let us prove the lemma statement for each level ℓ of the recursion (level 0 corresponding to the first call to $\mathcal{P}^{-}_{\texttt{AND-n}}$ and level $j-2$ corresponding to the last call to $\mathcal{P}^{-}_{\texttt{AND-n}}$ for a number of players strictly greater than 6). By construction, for each such level, Protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 15 is executed by a triplet of players (P_a, P_b, P_c) such that $a = t + \frac{n}{2^{\ell+1}} - 5$, $b = t + \frac{n}{2^{\ell+1}} - 2$ and $c = t + \frac{n}{2^{\ell+1}}$ with $\ell \in [0; j-2]$ and $t \in \{\frac{n}{2^\ell} i; i \in [0, 2^\ell - 1]\}$. The execution of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 15 leads to an encoding of the form $(\prod_{l=1}^{\frac{n}{2^\ell}} x_{t+l} + r^\star_{\overline{k}}, r^\star_{\overline{k}})$, with $k = \log_3(\frac{n}{3 \cdot 2^\ell}) - 1 \bmod 2$, if the players P_a, P_b, P_c respectively have the random sets $\mathcal{F}_1$, $\mathcal{F}^{\overline{k}}_2$, and $\mathcal{F}_0$. Since $\frac{n}{2^{\ell+1}}$ and $\frac{n}{2^\ell}$ are multiples of 6 as long as $\ell \leqslant j-2$, the indices a and c satisfy $a = 1 \bmod 6$ and $c = 0 \bmod 6$. Therefore, by definition of $\boldsymbol{R}^\star_i$ (see Equation (5.1)), players P_a and P_c respectively possess $\mathcal{F}_1, \mathcal{F}_0$. Eventually, by construction, b equals 4 modulo 6 and satisfies $\frac{b-4}{6} + 1 = \frac{t}{6} + \frac{n}{6 \cdot 2^{\ell+1}}$ that is $\frac{b-4}{6} + 1 = \frac{n}{3 \cdot 2^{\ell+2}}(2i+1)$ for some $i \in [0, 2^\ell - 1]$. The parity of the 2-valuation of $\frac{b-4}{6}$ is hence that of $\log_2(\frac{n}{3 \cdot 2^{\ell+2}})$, which exactly corresponds to $\overline{k}$. Eventually, by construction of $\boldsymbol{R}^\star_b$ (see Relation Equation (5.1)), P_b possesses the random set $\mathcal{F}^{\overline{k}}_2$. We conclude that players P_a, P_b, P_c possess the required random sets to perform $\mathcal{P}^{(2)}_{\texttt{AND2}}$.

◇

Propositions 5.6.1, 5.6.2 and 5.6.3 imply that $\mathcal{P}_{\texttt{AND-n}}$ is correct.

These results allow us to state the following theorem.

Theorem 5.6.4. *Protocol $\mathcal{P}^{-}_{\textit{AND-n}}$ satisfies the instance-secrecy property and Protocol $\mathcal{P}_{\textit{AND-n}}$ is private.*

Proof. We start by proving the instance-secrecy of Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$. To this end, we introduce the following useful Lemmas 5.6.5 and 5.6.6.

Lemma 5.6.5. *Let $P_i\{\boldsymbol{C_i}, \boldsymbol{R}_i\}$ be a player taking part in $\mathcal{P}^{(2)}_{\mathit{AND2}}$. Every non-zero linear combination of coordinates of $\boldsymbol{C_i}$ can be written as $g(X_a, X_b, \boldsymbol{R_a}, \boldsymbol{R_b}, \boldsymbol{R_c}) + R^\star$ where $R^\star \in (\boldsymbol{R_a} \cup \boldsymbol{R_b} \cup \boldsymbol{R_c}) \setminus \boldsymbol{R_i}$ and $g \perp R^\star$.*

Proof. We prove the lemma statement for the three players taking part in $\mathcal{P}^{(2)}_{\texttt{AND2}}$. By construction we have $\boldsymbol{C_a} = (R_3, X_b{+}R_3{+}R_4, (X_a{+}R_1{+}R_2)(X_b{+}R){+}R_1(R_3{+}R_4){+}R^\star)$, $\boldsymbol{C_b} = (R_1, X_a{+}R_1{+}R_2)$ and $\boldsymbol{C_c} = (X_a{+}R{+}R_1{+}R_2, R_2, X_b{+}R{+}R_3{+}R_4, R_4, X_aX_b{+}R_2R_4{+}R(X_a{+}R_1{+}R_2){+}R^\star)$ (where the player's secrets and random tapes content are viewed as random variables, and are therefore denoted by capital letters). Moreover, the random tapes of the 3 players are $\{R_1, R_2\}, \{R_3, R_4, R^\star\}$ and $\emptyset$ respectively. They are disjoint. Any linear combination of $\boldsymbol{C_a}$ coordinates involving the third coordinate can be written as $g(\cdots) + R^\star$, where $g(\cdots) \perp R^\star$ and $R^\star \in \boldsymbol{R_b}$. Any other combination involving the second coordinate can be written $g(\cdots)+R_4$, with $g \perp R_4$ and $R_4 \in \boldsymbol{R_b}$. The remaining non-zero combination takes the form $g(\cdots) + R_3$ with $g \perp R_3$ and $R_3 \in \boldsymbol{R_b}$. This concludes the proof for player P_a. Any linear combination of coordinates of $\boldsymbol{C_b}$ involving its second coordinate can be written $g(\cdots) + R_2$ with $g \perp R_2$ and $R_2 \in \boldsymbol{R_a}$, and the only remaining combination can be written $g(\cdots) + R_1$ with $g \perp R_1$ and $R_1 \in \boldsymbol{R_a}$. The lemma statement hence holds for player P_b. Finally, any linear combination of coordinates of $\boldsymbol{C_c}$ involving the last coordinate can be written as $g(\cdots) + R^\star$ with $g \perp R^\star$, $R^\star \in \boldsymbol{R_b}$. Any other combination involving the first (resp. third) coordinate can be written respectively $g(\cdots) + R_1$ (resp. $g(\cdots) + R_3$), with $g \perp R_1$ (resp. $g \perp R_3$) and $R_1 \in \boldsymbol{R_a}$ (resp. $R_3 \in \boldsymbol{R_b}$). Any other combination involving its second (resp. fourth) coordinate takes the form $g(\cdots) + R_2$ (resp. $g(\cdots) + R_4$), with $g \perp R_2$ (resp. $g \perp R_4$) and $R_2 \in \boldsymbol{R_a}$ (resp. $R_4 \in \boldsymbol{R_b}$). ◇

Lemma 5.6.6. *During the execution of $\mathcal{P}^{-}_{\mathit{AND\text{-}n}}$, any player executes $\mathcal{P}^{(2)}_{\mathit{AND2}}$ at most once.*

Proof. By assumption in $\mathcal{P}^{-}_{\texttt{AND-n}}$, the number of players n verifies $n = 3 \cdot 2^j$ for some positive integer j. Due to the recursive construction of our protocol, every player P_i such that $i = 2, 3, 5 \bmod 6$ executes $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 6, when $\mathcal{P}^{-}_{\texttt{AND-n}}$ reaches a terminal point. During the other level of the recursions (not terminal), executions of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ (at Step 15) are performed by players of indices $a = t + \frac{n}{2^{\ell+1}} - 5$, $b = t + \frac{n}{2^{\ell+1}} - 2$ and $c = t + \frac{n}{2^{\ell+1}}$ with $\ell \in [0; j-2]$ and $t \in \{\frac{n}{2^\ell} i; i \in [0, 2^\ell - 1]\}$. For such values ℓ and t, it can be checked that $t + \frac{n}{2^{\ell+1}}$ is a multiple of 6 which implies $a \equiv 1 \pmod 6$, $b \equiv 4 \pmod 6$ and $c \equiv 0 \pmod 6$. This shows that no player can participate to the execution of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at both Steps 6 and 15 of the recursion. Moreover, for $x \in \{1, 4, 6\}$, $\ell \in [0; j-2]$ and $i \in [0, 2^\ell - 1]$, it can be checked that the function $x, \ell, i \mapsto \frac{n}{2^{\ell+1}}(2i+1) + x - 6$ is injective, which implies that no player can perform $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 15 more than once during the recursion. ◇

We can now prove the instance-secrecy of $\mathcal{P}^{-}_{\texttt{AND-n}}$.

Proof. We reason by induction.

Basis. We assume $n = 6$. In this case, players P_1, P_4 and P_6 run no other protocol than $\mathcal{P}^{-}_{\texttt{AND3}}$ (and only once) and hence only receive messages during this execution. Moreover, $\mathcal{P}^{-}_{\texttt{AND3}}$ satisfies the instance-secrecy property with respect to these players. For any other player P, all received messages are either received during $\mathcal{P}^{-}_{\texttt{AND3}}$ or $\mathcal{P}^{(2)}_{\texttt{AND2}}$. Lemma 5.6.5 ensures that any non-zero linear combination involving only messages of $\mathcal{P}^{-}_{\texttt{AND3}}$ or of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ takes the form $g(\cdots) + R$ with R not in the random tape of P and $g \perp R$. Moreover, no message sent in $\mathcal{P}^{-}_{\texttt{AND3}}$ functionally depends on any random bit used in $\mathcal{P}^{(2)}_{\texttt{AND2}}$ (they indeed both use disjoint triplets of random tapes). Thus, Lemma 5.2.9 ensures that every linear combination of messages received during $\mathcal{P}^{(2)}_{\texttt{AND2}}$ and $\mathcal{P}^{-}_{\texttt{AND3}}$ takes the form $g(\cdots) + R$ with R not in the random tape of P and $g \perp R$. Lemma 5.2.8 thus implies that the communication vector received by any player $P\{X_i\}$ is independent of $(X_1, \cdots, X_6)|X_i = x_i$ for any $x_i \in \mathbb{F}_2$, which itself implies that $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property for $n = 6$ (Lemma 5.2.4).

Induction. We assume that (H1) $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property for an integer n such that $n = 3 \times 2^j$, and (H2) every non-zero linear combination of coordinates of the communication vector $\boldsymbol{C_i^{(n)}}$ received by a player $P_i\{\boldsymbol{R}_i\}$ during $\mathcal{P}^{-}_{\texttt{AND-n}}$ can be written as $g(\cdots) + R$ with $R \notin \boldsymbol{R}_i \cup \boldsymbol{R}_i^\star$ and $g \perp R$.

We consider now the execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $2n$ players. By construction, the communication $\boldsymbol{C_i^{(2n)}}$ received by each player $P_i\{\boldsymbol{R}_i\}$ can be written as $(\boldsymbol{C_i^{(n)}}, \boldsymbol{C_i'}, \boldsymbol{C_i''})$, where $\boldsymbol{C_i^{(n)}}$ is the communication received by P_i during the execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ for n players (Step 9 or Step 11), $\boldsymbol{C_i''}$ is the communication received during the execution of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 15, and $\boldsymbol{C_i'}$ is the communication received between these two calls (Step 14). Note that, for all but three players, we have $\boldsymbol{C_i'} = \boldsymbol{C_i''} = 0$. Thus, by hypothesis, $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to all these players and their communication vectors $\boldsymbol{C_i^{(2n)}}$ satisfy Hypothesis (H2) for $2n$ instead of n.

For any of the three remaining players P_i with $i \in \{a, b, c\}$, we deduce from Lemma 5.6.6 and $\boldsymbol{C_i''} \neq 0$ that no message in $\boldsymbol{C_i^{(n)}}$ has been received during the execution of $\mathcal{P}^{(2)}_{\texttt{AND2}}$. As a consequence, no message in $\boldsymbol{C_i^{(n)}}$ functionally depends on $\boldsymbol{R}_i^\star$ (the random tape used by P_i to execute $\mathcal{P}^{(2)}_{\texttt{AND2}}$). The single message in $\boldsymbol{C_i'}$ can be written $g'(\cdots) + R_k^\star$. Moreover, by construction, indices a, b and c respectively equal $n-5$, $n-2$ and n. The same reasoning as in the proof of Proposition 5.6.3 then shows that $\boldsymbol{R}_a^\star = \mathcal{F}_1$, $\boldsymbol{R}_b^\star = \mathcal{F}_2^{\overline{k}}$ and $\boldsymbol{R}_c^\star = \mathcal{F}_0$. For every $i \in \{a, b, c\}$, we deduce that the single message in $\boldsymbol{C_i'}$ is functionally independent of the random tapes $\boldsymbol{R}_a^\star$, $\boldsymbol{R}_b^\star$ and $\boldsymbol{R}_c^\star$ used in $\mathcal{P}^{(2)}_{\texttt{AND2}}$. Together with Lemma 5.6.5, this independency implies that any non-zero linear combination of $\boldsymbol{C_i'}$ with elements in $\boldsymbol{C_i''}$ takes the form $g(\cdots) + R$ with $R \in (\boldsymbol{R}_a^\star \cup \boldsymbol{R}_b^\star \cup \boldsymbol{R}_c^\star) \backslash (\boldsymbol{R}_i^\star)$ and $g \perp R$. Since no message in $\boldsymbol{C_i^{(n)}}$ functionally depends on $\boldsymbol{R}_a^\star \cup \boldsymbol{R}_b^\star \cup \boldsymbol{R}_c^\star$, we eventually deduce from Hypothesis (H2) that any non-zero linear combination of messages from $\boldsymbol{C_i^{(2n)}}$ can be written as $g(\cdots) + R$ with $R \in (\boldsymbol{R}_a \cup \boldsymbol{R}_a^\star \cup \boldsymbol{R}_b \cup \boldsymbol{R}_b^\star \cup \boldsymbol{R}_c \cup \boldsymbol{R}_c^\star) \backslash (\boldsymbol{R}_i \cup \boldsymbol{R}_i^\star)$ and $g \perp R$. Thanks to Lemma 5.2.8, we hence conclude that $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to the three players P_a, P_b and P_c and that their communication vectors $\boldsymbol{C_i^{(2n)}}$ satisfy Hypothesis (H2) for $2n$ instead of n.

We have shown that $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property for $2n$ players if it satisfies the instance-secrecy property for n in the form $3 \cdot 2^j$. As the instance-secrecy property is true for $n = 6$ we can conclude that the proposition statement is true for any n in the form $3 \cdot 2^j$ with $j \geqslant 1$. $\diamond$

We now prove the privacy of Protocol $\mathcal{P}_{\texttt{AND-n}}$.

Proof. By construction, any message received by a player running $\mathcal{P}_{\texttt{AND-n}}$ is either a message received in $\mathcal{P}^{-}_{\texttt{AND-n}}$ or in $\mathcal{P}_{\texttt{XOR}}$. Moreover, $\mathcal{P}^{-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to every player. During $\mathcal{P}_{\texttt{XOR}}$, each player receives a single message. All players but $P_{\frac{n}{2}}$ and $P_{\frac{n}{2}-2}$, who played respectively the roles of P_c and P_b during the last call to $\mathcal{P}^{(2)}_{\texttt{AND2}}$, receive $\prod_{i=1}^n X_i$ and learn nothing new except the protocol output.

- Player $P_{\frac{n}{2}-2}$ receives $\prod_{i=1}^n X_i + R^\star_{\overline{k}}$ and knows $R^\star_{\overline{k}}$. He then deduced $\prod_{i=1}^n X_i$ but learns nothing new: $\mathcal{P}_{\texttt{AND-n}}$ is private w.r.t $P_{\frac{n}{2}-2}$.

- Player $P_{\frac{n}{2}}$ receives $\prod_{i=1}^n X_i$ and knows $\prod_{i=1}^n X_i + R^\star_{\overline{k}}$. He then deduces $R^\star_{\overline{k}}$ who is used to secure communications during executions of $\mathcal{P}^{(2)}_{\texttt{AND2}}$. Due to Lemma 5.6.6, $P_{\frac{n}{2}}$ ran the latter protocol only once. Hence, his communication vector contains a single message which functionally depends on $R^{\star}_{\bar{k}}$, which is $\prod_{i=1}^{n} X_i + R''_2 R''_4 + R^{\star}_{k}(\prod_{i=1}^{n/2} X_i + R''_1 + R''_2) + R^{\star}_{\bar{k}}$. The sum of this message with $\prod_{i=1}^{n} X_i + R^{\star}_{\bar{k}}$ gives $R''_2 R''_4 + R^{\star}_{k}(\prod_{l=1}^{n/2} X_i + R''_1 + R''_2)$. Since $P_{\frac{n}{2}}$ also received R''_2 and R''_4 during $\mathcal{P}^{(2)}_{\texttt{AND2}}$, he can deduce the value $\prod_{i=1}^{n/2} X_i + R''_1$. The latter one is still protected by R''_1 which is unknown to $P_{\frac{n}{2}}$. Among the other messages received by $P_{\frac{n}{2}}$ during $\mathcal{P}^{(2)}_{\texttt{AND2}}$, only $\prod_{i=1}^{n/2} X_i + R''_1 + R''_2$ functionally depends on R''_1, however this message brings no additional information. We therefore deduce that $P_{\frac{n}{2}}$ learns nothing more than his secret and the final result: $\mathcal{P}_{\texttt{AND-n}}$ is private w.r.t. $P_{\frac{n}{2}}$.

Eventually, we get that $\mathcal{P}_{\texttt{AND-n}}$ is private with respect to all the players, which concludes the proof. $\diamond$

$\square$

5.7 Private AND evaluation with any number of players

In this section, we describe the protocol $\mathcal{P}^{\star}_{\texttt{AND-n}}$ allowing the private multiplication of any number n of input bits. It completes the protocol $\mathcal{P}_{\texttt{AND-n}}$ presented in Sections 4 for any number n of players taking the form 3×2^j. As Section 5.5 addresses the particular cases $n = 4, 5$, we focus hereafter on the case $n > 6$.

Protocol $\mathcal{P}^{\star}_{\texttt{AND-n}}$ mainly consists in two separate instances of $\mathcal{P}^{-}_{\texttt{AND-n}}$ for two sets of players of same cardinal n' defined as the largest integer of the form 3×2^j which is smaller than n. The first instance of $\mathcal{P}^{-}_{\texttt{AND-n}}$ is performed by players $(P_1, \cdots, P_{n'})$ and it returns an encoding of $x_1 \cdots x_{n'}$. Since n' and n satisfy $2n' > n$, the second execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ involves not only the remaining players $(P_{n'+1}, \cdots, P_n)$, but also *dummy* players (holding a public input 1) who are actually personified by players involved in the first instance. For the sake of clarity, we assume that each dummy player $(P_{n+1}, \cdots, P_{2n'})$ is in fact simulated by the corresponding player $P_{n-n'+i}$. Eventually, the second execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ returns an encoding of $x_{n'}, \cdots, x_n$. A further execution of $\mathcal{P}^{(4)}_{\texttt{AND2}}$ therefore suffices to provide players with an encoding of $x_1 \cdots x_n$. Finally, running $\mathcal{P}_{\texttt{XOR}}$ ensures that every player get the expected result. Each player P_i is provided with the random tapes $\boldsymbol{R}_i$, $\boldsymbol{R}^{\star}_i$ and $\boldsymbol{R}'_i$. Random tapes $\boldsymbol{R}_i$ and $\boldsymbol{R}^{\star}_i$ will be used for the executions of $\mathcal{P}^{-}_{\texttt{AND-n}}$ and random tapes $\boldsymbol{R}'_i$ will be used for $\mathcal{P}^{(4)}_{\texttt{AND2}}$. Before running the protocol, players fill their random tapes such that:

$$\boldsymbol{R}_i = \begin{cases} \{r_1\} & \text{if } i \equiv 1 \pmod 3 \text{ and } i \leq n', \quad \{s_1\} \quad \text{if } i \equiv 1 \pmod 3 \text{ and } i > n' \\ \{r_2, r, r'_1\} & \text{if } i \equiv 2 \pmod 3 \text{ and } i \leq n', \quad \{s_2, s, s'_1\} \quad \text{if } i \equiv 2 \pmod 3 \text{ and } i > n' \\ \{r'_2, r''\} & \text{if } i \equiv 0 \pmod 3 \text{ and } i \leq n', \quad \{s'_2, s''\} \quad \text{if } i \equiv 0 \pmod 3 \text{ and } i > n' \end{cases}$$

$$\boldsymbol{R}^{\star}_i = \begin{cases} \{r''_1, r''_2\} & \text{if } i \equiv 1, 2 \pmod 6 \text{ and } i \leq n' \\ \{r''_3, r''_4, r^{\star}_1\} & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is odd and } i \leq n' \\ \{r''_3, r''_4, r^{\star}_0\} & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is not odd and } i \leq n' \\ \{r''_3, r''_4, r^{\star}_1\} & \text{if } i \equiv 5 \pmod 6 \text{ and } i \leq n' \\ \{s''_1, s''_2\} & \text{if } i \equiv 1, 2 \pmod 6 \text{ and } i > n' \\ \{s''_3, s''_4, s^{\star}_1\} & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is odd and } i > n' \\ \{s''_3, s''_4, s^{\star}_0\} & \text{if } i \equiv 4 \pmod 6 \text{ and the 2-adic valuation of } \frac{i-4}{6} + 1 \text{ is not odd and } i > n' \\ \{s''_3, s''_4, s^{\star}_1\} & \text{if } i \equiv 5 \pmod 6 \text{ and } i > n' \\ \emptyset & \text{otherwise} \end{cases}$$

$$\boldsymbol{R'_i} = \begin{cases} \{r''_1, r''_2\} & \text{if } i = n' - 5 \\ \{r''_3, r''_4, r^\star\} & \text{if } i = n' - 2 \\ \emptyset & \text{otherwise} \end{cases}$$

We hence get the following 26-random protocol:

Protocol $\mathcal{P}^{\star-}_{\texttt{AND-n}}$

```
Require: $n$ players $P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R^\star_1} \cup \boldsymbol{R'_1}\}, \cdots, P_n\{x_n, \boldsymbol{R_n} \cup \boldsymbol{R^\star_n} \cup \boldsymbol{R'_n}\}$
Ensure: Player $P_{n'}$ gets $\prod_{i=1}^n x_i + r^\star$ where $r^\star$ is a random bit known by$P_{n'-2}$ only

  Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $(P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R^\star_1}\}, \cdots, P_{n'}\{x_{n'} \cup \boldsymbol{R_{n'}}, \boldsymbol{R^\star_{n'}}\})$
                                      ▷ we get $(P_{\frac{n'}{2}}\{x_1 \cdots x_{n'} + r^\star_k\}, P_{\frac{n'}{2}-2}\{r^\star_k\})$
  Protocol $\mathcal{P}^{-}_{\texttt{AND-n}}$ for $(P_{n'+1}\{x_{n'+1}, \boldsymbol{R}_{n'+1} \cup \boldsymbol{R}^\star_{n'+1}\}, \cdots, P_{2n'}\{1, \boldsymbol{R}_{2n'} \cup \boldsymbol{R}_{2n'\star}\})$
                                      ▷ we get $(P_{\frac{3n'}{2}}\{x_{n'+1} \cdots x_n + s^\star_k\}, P_{\frac{3n'}{2}-2}\{s^\star_k\})$
  $P_{\frac{n'}{2}}$ sends $x_1 \cdots x_{n'} + r^\star_k$ to $P_{n'-5}$
  $P_{\frac{3n'}{2}}$ sends $x_{n'+1} \cdots x_n + s^\star_k$ to $P_{n'-2}$
  $P_{\frac{n'}{2}-2}$ sends $r^\star_k$ to $P_{n'}$
  $P_{\frac{3n'}{2}}$ sends $s^\star_k$ to $P_{n'}$
  Protocol $\mathcal{P}^{(4)}_{\texttt{AND2}}$ for $(P_{n'-5}\{x_1 \cdots x_{n'} + r^\star_k, \boldsymbol{R}'_{n'-5}\}, P_{n'-2}\{x_{n'+1} \cdots x_n + s^\star_k, \boldsymbol{R}'_{n'-2}\}, P_{n'}\{r^\star_k, s^\star_k\})$
                                      ▷ we get $(P_{n'}\{\prod_{i=1}^n x_i + r^\star\}, P_{n'-2}\{r^\star\})$
```

Protocol $\mathcal{P}^{\star}_{\texttt{AND-n}}$

```
Require: $n$ players $P_1\{x_1, \boldsymbol{R}_1 \cup \boldsymbol{R}^\star_1\}, \cdots, P_n\{x_n, \boldsymbol{R}_n \cup \boldsymbol{R}^\star_n\}$ with $n$ in the form $3 \cdot 2^j$ for $j \geqslant 1$.
Ensure: All the players get $\prod_{i=1}^n x_i$

  Protocol $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ for $(P_1\{x_1, \boldsymbol{R_1} \cup \boldsymbol{R^\star_1} \cup \boldsymbol{R'_1}\}, \cdots, P_n\{x_n, \boldsymbol{R_n} \cup \boldsymbol{R^\star_n} \cup \boldsymbol{R'_n}\})$
                                      ▷ we get $P_{n'}\{\prod_{i=1}^n x_i + r^\star\}$ with $r^\star \in \boldsymbol{R'_{n'-2}}$
  Protocol $\mathcal{P}_{\texttt{XOR}}$ for $(\cdots, P_{\frac{n'}{2}-2}\{r^\star\}, \cdots, P_{n'}\{\prod_{i=1}^n x_i + r^\star\}, \cdots)$
                                      ▷ we get $(P_1\{\prod_{i=1}^n x_i\}, \cdots P_n\{\prod_{i=1}^n x_i\})$
```

The correctness of $\mathcal{P}^{\star}_{\texttt{AND-n}}$ can be checked thanks to the comments inserted in the protocol description. We now state the following theorem:

We then deduce the following theorem:

Theorem 5.7.1. *Protocol $\mathcal{P}^{\star-}_{\textit{AND-n}}$ satisfies the instance-secrecy property and Protocol $\mathcal{P}^{\star}_{\textit{AND-n}}$ is private for any number of players.*

Proof. We start by proving the instance-secrecy of Protocol $\mathcal{P}^{\star}_{\texttt{AND-n}}$. To this end, we introduce the following lemma:

Lemma 5.7.2. *In protocol $\mathcal{P}_{\textit{AND-n}}$ with $n = 3 \times 2^j$ and $j \geqslant 1$, the players P_{n-5}, P_{n-2} and P_n never execute $\mathcal{P}^{(2)}_{\textit{AND2}}$.*

Proof. Since $n = 3 \times 2^j$, we have $n-5 = 1 \bmod 6$, $n-2 = 4 \bmod 6$ and $n = 0 \bmod 6$. Thus, none of these players can execute $\mathcal{P}^{(2)}_{\texttt{AND2}}$ during a terminal point of $\mathcal{P}^{-}_{\texttt{AND-n}}$. As previously stated, when not in a terminal point, executions of $\mathcal{P}^{(2)}_{\texttt{AND2}}$ are performed by players of indices $a = t + \frac{n}{2^{\ell+1}} - 5, b = t + \frac{n}{2^{\ell+1}} - 2$ and $c = t + \frac{n}{2^{\ell+1}}$ with $\ell \in [0, j-2]$ and $t \in \{\frac{n}{2^\ell} i; i \in [0, 2^\ell - 1]\}$. Thus, players P_{n-5}, P_{n-2} and P_n perform $\mathcal{P}^{(2)}_{\texttt{AND2}}$ at Step 15 if and only if the equation $t + \frac{n}{2^{\ell+1}} = n$ has a solution. However, due to the expression of t, it can be checked that this equation is equivalent to $2i + 1 = 2^{\ell+1}$, which accepts no solution. Consequently, players P_{n-5}, P_{n-2}, P_n do not perform $\mathcal{P}^{(2)}_{\texttt{AND2}}$. ◇

We can now prove the instance-secrecy of Protocol $\mathcal{P}^{\star}_{\texttt{AND-n}}$:

Proof. By construction, the communication $\boldsymbol{C_i^{(n)}}$ received by each player P_i can be written as $(\boldsymbol{C_i^{(1)}}, \boldsymbol{C_i^{(2)}}, \boldsymbol{C_i^{(3)}})$, where $\boldsymbol{C_i^{(1)}}$ is the communication received during the first execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ with n' players, $\boldsymbol{C_i^{(2)}}$ is the communication received during the second execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ with n' players, and $\boldsymbol{C_i^{(3)}}$ is the communication received during the execution of $\mathcal{P}^{(4)}_{\texttt{AND2}}$. Note that, for all players but $P_{n'-5}, P_{n'-2}, P_{n'}$, we have $\boldsymbol{C_i^{(3)}} = 0$.

Let us first recall that n' in $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ corresponds to the largest value of the form $3 \cdot 2^j$ which is smaller than n. We denote by I_1 (resp. I_2) the union of intervals $[1, n-n'] \cup [n'+1, n]$ (resp. the interval $[n-n'+1, n']$). The set $\{n'-5, n'-2, n'\}$ is moreover denoted by I_3.

By construction, players P_i with index $i \in I_1$ participated in a single execution of $\mathcal{P}^{-}_{\texttt{AND-n}}$ (and thus $\boldsymbol{C_i^{(n)}}$ equals either $\boldsymbol{C_i^{(1)}}$ if $i \in [1, n-n']$ or $\boldsymbol{C_i^{(2)}}$ if $i \in [n'+1, n]$). The remaining players with $i \in I_2$ participated in both executions.

- In the case $i \in I_1 \backslash I_3$, we deduce from Lemma 5.2.8 that any non-zero linear combination of $\boldsymbol{C_i^{(n)}}$ coordinates can be written as $g(\cdots) + R$ with $g \perp R$ and $R \in (\bigcup_{l=1}^{n'} (\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})) \backslash (\boldsymbol{R_i} \cup \boldsymbol{R_i^\star})$ or $R \in (\bigcup_{l=n'+1}^{2n'} (\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})) \setminus (\boldsymbol{R_i} \cup \boldsymbol{R_i^\star})$. We hence deduce from Lemma 5.2.8 that $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to players P_i with $i \in I_1 \backslash I_3$.

- In the case $i \in I_2 \backslash I_3$, the players P_i have not only their random tapes $\boldsymbol{R_i} \cup \boldsymbol{R_i^\star}$ but also the random tapes $\boldsymbol{R_{n'+i}} \cup \boldsymbol{R^\star_{n'+i}}$ (to play the role of the dummy players $P_{n'+i}$). By construction, no message sent in $\boldsymbol{C_i^{(2)}}$ is functionally dependent of any random bit in $\bigcup_{l=1}^{n'} (\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})$ (involved in $\boldsymbol{C_i^{(1)}}$), hence any non-zero linear combination of $\boldsymbol{C_i^{(n)}}$ can be written as $g(\cdots) + R$ with $g \perp R$ and $R \in (\bigcup_{l=1}^{2n'} (\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})) \backslash (\boldsymbol{R_i} \cup \boldsymbol{R_i^\star} \cup \boldsymbol{R_{n'+i}} \cup \boldsymbol{R^\star_{n'+i}})$. We deduce from Lemma 5.2.8 that $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to players P_i with $i \in I_2 \backslash I_3$.

- For players P_i with $i \in I_3$ (who also played the roles of the dummy players with indices in $\{2n'-5, 2n'-2, 2n'\}$ during the second call to $\mathcal{P}^{-}_{\texttt{AND-n}}$), Lemma 5.7.2 imply that every non-zero linear combination of $\boldsymbol{C_i^{(1)}}$ coordinates can be written as $g(\cdots) + R$ with $R \in (\bigcup_{l=1}^{n'} (\boldsymbol{R_l})) \setminus \boldsymbol{R_i}$ and $g \perp R$. Similarly, every non-zero linear combination of $\boldsymbol{C_i^{(2)}}$ coordinates can be written as $g(\cdots) + R$, with $R \in (\bigcup_{l=n'+1}^{2n'} (\boldsymbol{R_l})) \setminus \boldsymbol{R_{n'+i}}$ and $g \perp S$. Moreover, we have that any non-linear combination of $\boldsymbol{C_i^{(3)}}$ coordinates takes the form $g(\cdots) + R$, with $R \in (\bigcup_{l=1}^{2n'} (\boldsymbol{R'_l})) \setminus \boldsymbol{R'_i}$ and $g \perp R$. Since every message in $\boldsymbol{C_i^{(1)}}$ or $\boldsymbol{C_i^{(2)}}$ is functionally independent of any element in $\bigcup_{l=1}^{2n'} \boldsymbol{R'_l}$, we have that every non-zero linear combination involving a coordinate of $\boldsymbol{C_i^{(3)}}$ can be written as $g(\cdots) + R'$, with $R' \in (\bigcup_{l=1}^{2n'} (\boldsymbol{R'_l})) \setminus (\boldsymbol{R'_i})$ and $g \perp R'$. Finally, since no message sent in $\boldsymbol{C_i^{(2)}}$ is functionally dependent on any random bit in $\bigcup_{l=1}^{n'} (\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})$, we get that any non-zero linear combination of $\boldsymbol{C_i^{(n)}}$ can be written as $g(\cdots) + R$ with $g \perp R$ and

$R \in (\bigcup_{l=1}^{2n'}(\boldsymbol{R_l} \cup \boldsymbol{R_l^\star})) \setminus (\boldsymbol{R_i} \cup \boldsymbol{R_i^\star} \cup \boldsymbol{R_{n'+i}} \cup \boldsymbol{R_{n'+i}^\star} \cup \boldsymbol{R_i'})$. Thanks to Lemma 5.2.8, we hence conclude that $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property with respect to P_i with $i \in I_3$.

We eventually conclude that $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property. Since $\mathcal{P}^{\star-}_{\texttt{AND-n}}$ satisfies the instance-secrecy property, the proof of the privacy of $\mathcal{P}_{\texttt{AND-n}}$ follows. ◇

□

According to the definition of our random tapes, $\mathcal{P}^{\star}_{\texttt{AND-n}}$ uses 26 random bits.

5.8 Direct Extension of the Results

All the protocols described in this paper can be straightforwardly extended to privately evaluate the multiplication in any finite ring $\mathcal{R}$. The randomness complexity of our protocols is multiplied by the binary logarithm of the cardinality of $\mathcal{R}$.

Proposition 5.8.1. *Let $\mathcal{R}$ be a finite ring with multiplication law denoted by $\times$. Let n be a positive integer and let $\texttt{MUL-n}$ denotes the function $(x_1, \cdots, x_n) \in \mathcal{R} \mapsto x_1 \times \cdots \times x_n$. Then, for every n, there exists a private protocol $\mathcal{P}(n, \texttt{MUL-n})$ with randomness complexity equal to $12 \cdot \log_2(\#\mathcal{R})$ if n takes the form $3 \cdot 2^j$ or $26 \cdot \log_2(\#\mathcal{R})$ otherwise.*

To the best of our knowledge, the protocol described in [KOR99] cannot be adapted to this general case. Our proposal is hence of particular relevance in this context.

The construction of [KOR99] is however still relevant, as it considers a much larger class of functions. We propose in Section 5.9 an improvement to its randomness complexity.

5.9 Diminution of the random complexity of the KOR protocol

In this section, we adapt the protocol proposed [KOR99] and show how its random complexity can be straightforwardly improved by our results.

We start by giving a very high level description of the approach.

The n players $P_1, \ldots, P_n$ taking part in the protocol from [KOR99] are divided into two groups $\Delta_1 = \{P_1, \cdots, P_{\lfloor \frac{n}{2} \rfloor}\}$, and $\Delta_2 = \{P_{\lfloor \frac{n}{2} \rfloor + 1}, \cdots, P_n\}$. The player P_1 plays the role of a *randomness player*. That is, P_1 reads every random bit used in the protocol and distributes some of them to the other players. Players in $\Delta_1 \setminus \{P_1\}$ are further separated in $k \simeq \frac{n}{6}$ *teams* $\{T_1, \cdots, T_k\}$ of three players. We denote by A_i, B_i, C_i the three players forming the i-th team T_i.

The function evaluated by the n players is considered as a circuit taking as inputs each player's secret, as well as their negations. The circuit is assumed to be composed of only `AND` and `OR` gates of fan-in 2 and arbitrary fan-out. The number of gates in this circuit is denoted m.

At the beginning of the protocol, P_1 reads a random bit r_0 and sends it to each player in Δ_1. Then, each player $P_i \in \Delta_1$ sends $x_i + r_0$ to $P_{i + \lfloor \frac{n}{2} \rfloor}$. The idea of the protocol is to use the teams $T_1, \cdots, T_k$ to *simulate* the m gates of the circuit. Specifically, each team will be responsible for the simulation of at most $\frac{m}{k}$ gates. The protocol ensures that each time a team is evaluating a function $c = f(a, b)$, the players A_i and B_i should hold respectively $a \oplus r_d$ and $b \oplus r_d$ where r_d is a random bit known by P_1 only. Through a particular interaction between A_i, B_i, C_i and P_1, the protocol ensures that C_i gets the value $c \oplus s_d$, where s_d is a random bit known by P_1 only. When the final gate has been simulated, the output of the evaluated function is then computed with the protocol $\mathcal{P}_{\texttt{XOR}}$.

The randomness complexity of this protocol is limited by the fact that, each team evaluating its j-th gate, will use the same set $\boldsymbol{R}_j$ of random bits. Therefore, the random complexity of this protocol is exactly $(\alpha + 2) \times \frac{m}{k} + 1$, where α is the number of random bits used to simulate a gate. Since $k \approx \frac{n}{6}$, then for each function for which the number of gates m is a linear function of the number of players n, this randomness complexity is constant. In the protocol given in [KOR99], we have $\alpha = 10$. Note that in the particular case of the `AND` function, we have $m = n - 1$, and therefore the protocol needs $6 \times 12 + 1 = 73$ random bits.

The results we presented in Section 4 allow for the reduction of the number α, and therefore for the reduction of the random complexity of such a protocol for any function which can be evaluated with a linear circuit. Indeed, for evaluating an `AND` gate, players A, B, C just need to perform the protocol $\mathcal{P}^{(2)}_{\texttt{AND2}}$ with random tapes provided by P_1. The evaluation of any `OR` gate is trivial thanks to De Morgan's theorem: instead of performing $\mathcal{P}^{(2)}_{\texttt{AND2}}$ with inputs $a + r_d, b + r_d$, players perform it with inputs $a + r_d + 1, b + r_d + 1$, ensuring the return of $c + r_d + 1$, which gives $c + r_d$. Since $\mathcal{P}^{(2)}_{\texttt{AND2}}$ is 5-random, we get that $\alpha = 5$.

Therefore, each function f can be securely computed with $42 \times \frac{m}{n-1} + 1$ random bits, where m is the number of gates of the circuit evaluating f, and n is the number of input variables.

5.10 Conclusion

This chapter opens the study of the exact randomness complexity of private veto protocols. In addition to its intrinsic interest, the analysis of this problem has already found applications in other private protocols (as shown in Sections 5.8 and 5.9) and raises several questions for future research.

For 1-private protocols, the tantalizing open problem is naturally to provide a non-trivial lower bound on the randomness complexity of the evaluation of the `AND-n` function. Even for three players, it seems very difficult to prove that a protocol requires more than 3 random bits. For $t \geqslant 2$, the results from [KM96] and [GR03] imply that $\Omega(\max(t, \log(n)))$ random bits are necessary for the t-private computation of the `AND-n` function. The most random-efficient t-private protocol for $t \geqslant 2$ requires $O(\mathsf{poly}(t) \log(n))$ random bits [CKOR00]. It is an interesting open problem to develop new techniques for proving improved lower bounds and/or to find better constructions than those proposed here (for $t = 1$). Finally, finding tradeoffs between randomness complexity and round complexity for private veto protocols (on one or several instances) also deserves further attention.

Chapter 6

Conclusion and Perspectives

During these three years, we have addressed several issues concerning embedded cryptography. In this manuscript, we chose to present three distinct but related topics.

First, we introduced a novel methodology allowing to estimate the success rate of side-channel attacks, without even having to perform them. This methodology could help evaluators and designers to get a strong sense of the security of their products, in a limited time. We also gave some rationales about the confidence one can have in the results of such attacks when they are actually performed. Part of these results have been presented at the CHES conference ([TPR13; LPR+14]).

Then, we tackled the problem of the randomness complexity of the multiplication. Our approach studied the design of private circuits allowing to compute the product of two shared variables. We deeply analyzed this problem, and offered bounds on such a complexity, as well as concrete proven constructions. We also studied the composition of these algorithms, allowing for the design of circuits implementing whole cryptographic functions. These results could help designers to achieve secure implementations, while maintaining correct performance. Part of these results have been presentend at the Eurocrypt conference ([BBP+16]).

Finally, we addressed the construction of 1-private veto protocols, in the honest but curious model. We gave lower and upper bounds on the randomness complexity of such protocols. We then used several small protocols as basic bricks to modularly construct secure solutions to this problem. Our ideas allow for the secure computation of more complex functions, such as the majority, or field products. This work is unpublished.

We also tackled other topics, that are not presented in this manuscript.

Just before the official beginning of this thesis, we addressed at the *Fault Diagnosis and Tolerance in Cryptography conference* (FDTC) ([LRT12]) the issue of fault attacks. Fault attacks consists in the perturbation of a device while it is performing a cryptographic algorithm, in order to obtain erroneous ciphertexts. These erroneous ciphertexts are then compared with the corresponding expected correct ciphertexts, thanks to a statistical treatment, allowing to retrieve the secret key. The main contribution of our paper was to present several attacks against countermeasures supposed to thwart these attacks, proving their inefficiency by using simple mathematical properties. We then proposed several other constructions to prevent such attacks.

We continued with the subject of fault attacks at the FDTC conference ([FJLT13]). This time, we demonstrated how it is possible to actually attack cryptographic algorithms with faults, without the knowledge of the expected correct ciphertexts. We validated our approach against AES. These results show the inefficiency of a family of protocol countermeasures which prevented the same plaintext to be encrypted twice.

Finally, we provided at the PKC conference ([BCTV16]) a toolbox based on analytic combinatorics to ease some public-key cryptanalysis methods based on Coppersmith's algorithm. The use of this toolbox allows to easily recover known results, sometimes even allowing more accurate estimations of their complexities.

All those subjects are still opened for further interesting and exciting researches.

Bibliography

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. “An Implementation of DES and AES, Secure against Some Attacks”. In: *CHES 2001*. Ed. by Çetin Kaya Koç, David Naccache, and Christof Paar. Vol. 2162. LNCS. Springer, Heidelberg, May 2001, pp. 309–318 (cit. on p. 28).

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. “Template Attacks in Principal Subspaces”. In: *CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. LNCS. Springer, Heidelberg, Oct. 2006, pp. 1–14 (cit. on p. 45).

[Bau04] Arthur O. Bauer. *Some aspects of military line communication as deployed by the German armed forces prior to 1945*. 2004 (cit. on p. 19).

[BBD+15a] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. *Compositional Verification of Higher-Order Masking: Application to a Verifying Masking Compiler*. Cryptology ePrint Archive, Report 2015/506. http://eprint.iacr.org/2015/506. 2015 (cit. on pp. 31, 71, 94–96).

[BBD+15b] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. “Verified Proofs of Higher-Order Masking”. In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 457–485. DOI: 10.1007/978-3-662-46800-5_18 (cit. on pp. 31, 70–72, 89, 93, 94, 97, 100).

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. “Randomness Complexity of Private Circuits for Multiplication”. In: *Eurocrypt*. Springer, 2016. DOI: http://doi.org/10.1007/978-3-662-49896-5_22 (cit. on pp. 67, 133).

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation Power Analysis with a Leakage Model”. In: *CHES 2004*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. LNCS. Springer, Heidelberg, Aug. 2004, pp. 16–29 (cit. on pp. 23, 45).

[BCTV16] Fabrice Benhamouda, Céline Chevalier, Adrian Thillard, and Damien Vergnaud. “Easing Coppersmith Methods Using Analytic Combinatorics: Applications to Public-Key Cryptography with Weak Pseudorandomness”. In: (2016), pp. 36–66 (cit. on p. 133).

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. “Inner Product Masking Revisited”. In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 486–510. DOI: 10.1007/978-3-662-46800-5_19 (cit. on pp. 28, 32).

[BFGV12] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. "Theory and Practice of a Leakage Resilient Masking Scheme". In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 758–775. DOI: 10.1007/978-3-642-34961-4_45 (cit. on p. 28).

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. "On the Cost of Lazy Engineering for Masked Software Implementations". In: (2014), pp. 64–81. DOI: 10.1007/978-3-319-16763-3_5. URL: http://dx.doi.org/10.1007/978-3-319-16763-3_5 (cit. on p. 29).

[BGN+14a] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. "A More Efficient AES Threshold Implementation". In: *AFRICACRYPT 14*. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, Heidelberg, May 2014, pp. 267–284. DOI: 10.1007/978-3-319-06734-6_17 (cit. on p. 68).

[BGN+14b] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. "Higher-Order Threshold Implementations". In: *ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, Heidelberg, Dec. 2014, pp. 326–343. DOI: 10.1007/978-3-662-45608-8_18 (cit. on p. 68).

[BGP07] Carlo Blundo, Clemente Galdi, and Giuseppe Persiano. "Low-randomness constant-round private XOR computations". In: *Int. J. Inf. Sec.* 6.1 (2007), pp. 15–26 (cit. on p. 102).

[BK12] Elaine B. Barker and John M. Kelsey. *SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. Tech. rep. Gaithersburg, MD, United States, 2012 (cit. on pp. 34, 69).

[BOGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. "Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions". In: *20th ACM STOC*. ACM Press, May 1988, pp. 113–131 (cit. on p. 68).

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 1–10 (cit. on pp. 30, 102, 103, 106, 111).

[Cac97] Cachin. "Entropy Measures and Unconditional Security in Cryptography". PhD thesis. 1997 (cit. on p. 44).

[CC06] Hao Chen and Ronald Cramer. "Algebraic Geometric Secret Sharing Schemes and Secure Multi-Party Computations over Small Fields". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Vol. 4117. LNCS. Springer, Heidelberg, Aug. 2006, pp. 521–536 (cit. on p. 30).

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. "Differential Power Analysis in the Presence of Hardware Countermeasures". In: *CHES 2000*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. LNCS. Springer, Heidelberg, Aug. 2000, pp. 252–263 (cit. on p. 33).

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. "Multiparty Unconditionally Secure Protocols (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 11–19 (cit. on pp. 30, 102).

[CCG+07] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. “Secure Computation from Random Error Correcting Codes”. In: *EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. LNCS. Springer, Heidelberg, May 2007, pp. 291–310 (cit. on p. 29).

[CEM] CEM. *Common Methodology for Information Technology Security Evaluation*. URL: `http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R3.pdf` (cit. on p. 11).

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. “Higher-Order Masking Schemes for S-Boxes”. In: *FSE 2012*. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, Heidelberg, Mar. 2012, pp. 366–384 (cit. on p. 31).

[CGPZ16] Jean-Sébastien Coron, Aurélien Greuet, Emmanuel Prouff, and Rina Zeitoun. “Faster Evaluation of SBoxes via Common Shares”. In: *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. Lecture Notes in Computer Science. Springer, 2016, pp. 498–514. ISBN: 978-3-662-53139-6. DOI: `10.1007/978-3-662-53140-2_24`. URL: `http://dx.doi.org/10.1007/978-3-662-53140-2_24` (cit. on p. 31).

[Cha88] David Chaum. “The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability”. In: *Journal of Cryptology* 1.1 (1988), pp. 65–75 (cit. on p. 103).

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. “Towards Sound Approaches to Counteract Power-Analysis Attacks”. In: *CRYPTO’99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 398–412 (cit. on pp. 25, 28, 68).

[CK14] Omar Choudary and Markus G. Kuhn. *Template Attacks on Different Devices*. Cryptology ePrint Archive, Report 2014/459. `http://eprint.iacr.org/2014/459`. 2014 (cit. on p. 22).

[CKOR00] Ran Canetti, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. “Randomness versus Fault-Tolerance”. In: *Journal of Cryptology* 13.1 (2000), pp. 107–142 (cit. on pp. 103, 131).

[CPR12] Jean-Sebastien Coron, Emmanuel Prouff, and Thomas Roche. “On the use of Shamir’s secret sharing against side-channel analysis”. In: (2012) (cit. on p. 31).

[CPRR14] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. “Higher-Order Side Channel Security and Mask Refreshing”. In: *FSE 2013*. Ed. by Shiho Moriai. Vol. 8424. LNCS. Springer, Heidelberg, Mar. 2014, pp. 410–424. DOI: `10.1007/978-3-662-43933-3_21` (cit. on pp. 69, 90).

[CPRR15] Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. “Algebraic Decomposition for Probing Security”. In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 742–763. DOI: `10.1007/978-3-662-47989-6_36` (cit. on p. 31).

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. “Template Attacks”. In: *CHES 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. LNCS. Springer, Heidelberg, Aug. 2003, pp. 13–28 (cit. on pp. 22, 45, 51).

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. “Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-Channel Countermeasures”. In: *CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. LNCS. Springer, Heidelberg, Sept. 2014, pp. 170–187. DOI: 10.1007/978-3-662-44709-3_10 (cit. on p. 31).

[CSM+14] Kaushik Chakraborty, Sumanta Sarkar, Subhamoy Maitra, Bodhisatwa Mazumdar, Debdeep Mukhopadhyay, and Emmanuel Prouff. *Redefining the Transparency Order*. Cryptology ePrint Archive, Report 2014/367. http://eprint.iacr.org/2014/367. 2014 (cit. on p. 46).

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. “Unifying Leakage Models: From Probing Attacks to Noisy Leakage”. In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 423–440. DOI: 10.1007/978-3-642-55220-5_24 (cit. on pp. 26, 68, 69).

[Def85] Department of Defense. *Trusted Computer System Evaluation Criteria*. 1985. URL: http://csrc.nist.gov/publications/secpubs/rainbow/std001.txt (cit. on p. 9).

[Des] *DES (Data Encryption Standard) Review at Stanford University*. URL: http://www.toad.com/des-stanford-meeting.html (cit. on p. 15).

[DFS15a] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. “Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device”. In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 401–429. DOI: 10.1007/978-3-662-46800-5_16 (cit. on pp. 26, 68).

[DFS15b] Stefan Dziembowski, Sebastian Faust, and Maciej Skorski. “Noisy Leakage Revisited”. In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Heidelberg, Apr. 2015, pp. 159–188. DOI: 10.1007/978-3-662-46803-6_6 (cit. on pp. 26, 69).

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. “Leakage-Resilient Cryptography”. In: *49th FOCS*. IEEE Computer Society Press, Oct. 2008, pp. 293–302 (cit. on p. 24).

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. *Univariate Side Channel Attacks and Leakage Modeling*. Cryptology ePrint Archive, Report 2011/302. http://eprint.iacr.org/2011/302. 2011 (cit. on pp. 24, 38, 55).

[EC90] Commission of the European Communities. *Information Technology Security Evaluation Criteria*. 1990. URL: http://www.ssi.gouv.fr/uploads/2015/01/ITSEC-uk.pdf (cit. on p. 9).

[Eck85] Wim Van Eck. *Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?* 1985 (cit. on p. 19).

[Fis20] R.A. Fisher. “A mathematical examination of the methods of determining the accuracy of an observation by the mean error, and by the mean square error”. In: *Mon. Notices Roy. Astron. Society* (1920) (cit. on p. 22).

[Fis22] R.A. Fisher. “On the mathematical foundations of theoretical statistics”. In: *Philosophical Transactions of the Royal Society* (1922) (cit. on pp. 22, 38).

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. "Fault Attacks on AES with Faulty Ciphertexts Only". In: (2013), pp. 108–118 (cit. on p. 133).

[FLD12] Yunsi Fei, Qiasi Luo, and A. Adam Ding. "A Statistical Model for DPA with Novel Algorithmic Confusion Analysis". In: *CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. LNCS. Springer, Heidelberg, Sept. 2012, pp. 233–250 (cit. on pp. 38, 46).

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. "Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases". In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 135–156 (cit. on p. 25).

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. "Parity, Circuits, and the Polynomial-Time Hierarchy". In: *Mathematical Systems Theory* 17.1 (1984), pp. 13–27. DOI: 10.1007/BF01744431. URL: http://dx.doi.org/10.1007/BF01744431 (cit. on p. 25).

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. "Homomorphic Evaluation of the AES Circuit". In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 850–867 (cit. on p. 31).

[GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. "Templates vs. Stochastic Methods". In: *CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. LNCS. Springer, Heidelberg, Oct. 2006, pp. 15–29 (cit. on p. 45).

[GM11] Louis Goubin and Ange Martinelli. "Protecting AES with Shamir's Secret Sharing Scheme". In: *CHES 2011*. Ed. by Bart Preneel and Tsuyoshi Takagi. Vol. 6917. LNCS. Springer, Heidelberg, 2011, pp. 79–94 (cit. on p. 31).

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *19th ACM STOC*. Ed. by Alfred Aho. ACM Press, May 1987, pp. 218–229 (cit. on p. 102).

[GP99] Louis Goubin and Jacques Patarin. "DES and Differential Power Analysis (The "Duplication" Method)". In: *CHES'99*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. LNCS. Springer, Heidelberg, Aug. 1999, pp. 158–172 (cit. on pp. 28, 68).

[GPP+16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Adi Shamir, and Eran Tromer. "Physical key extraction attacks on PCs". In: *Commun. ACM* 59.6 (2016), pp. 70–79. DOI: 10.1145/2851486. URL: http://doi.acm.org/10.1145/2851486 (cit. on p. 9).

[GPPT15] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. "Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation". In: *CHES 2015*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. LNCS. Springer, Heidelberg, Sept. 2015, pp. 207–228. DOI: 10.1007/978-3-662-48324-4_11 (cit. on p. 9).

[GPPT16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. "ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs". In: *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*. 2016, pp. 219–235 (cit. on p. 9).

[GPQ10] Laurie Genelle, Emmanuel Prouff, and Michaël Quisquater. “Secure Multiplicative Masking of Power Functions”. In: *ACNS 10*. Ed. by Jianying Zhou and Moti Yung. Vol. 6123. LNCS. Springer, Heidelberg, June 2010, pp. 200–217 (cit. on p. 28).

[GPS14] Vincent Grosso, Emmanuel Prouff, and François-Xavier Standaert. “Efficient Masked S-Boxes Processing - A Step Forward -”. In: *AFRICACRYPT 14*. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. LNCS. Springer, Heidelberg, May 2014, pp. 251–266. DOI: 10.1007/978-3-319-06734-6_16 (cit. on p. 31).

[GPT15] Daniel Genkin, Itamar Pipman, and Eran Tromer. “Get your hands off my laptop: physical side-channel key-extraction attacks on PCs - Extended version”. In: *J. Cryptographic Engineering* 5.2 (2015), pp. 95–112 (cit. on p. 9).

[GPW+04] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. “Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs”. In: *CHES 2004*. Ed. by Marc Joye and Jean-Jacques Quisquater. Vol. 3156. LNCS. Springer, Heidelberg, Aug. 2004, pp. 119–132 (cit. on p. 15).

[GR03] Anna Gál and Adi Rosén. “Lower bounds on the amount of randomness in private computation”. In: *35th ACM STOC*. ACM Press, June 2003, pp. 659–666 (cit. on pp. 102, 131).

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. “Simplified VSS and Fact-Track Multiparty Computations with Applications to Threshold Cryptography”. In: *17th ACM PODC*. Ed. by Brian A. Coan and Yehuda Afek. ACM, 1998, pp. 101–111 (cit. on pp. 103, 104, 111).

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. “RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis”. In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 444–461. DOI: 10.1007/978-3-662-44371-2_25 (cit. on pp. 9, 19).

[GT03] Jovan Dj. Golic and Christophe Tymen. “Multiplicative Masking and Power Analysis of AES”. In: *CHES 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. LNCS. Springer, Heidelberg, Aug. 2003, pp. 198–212 (cit. on p. 28).

[Har96] C. Harpes. “Cryptanalysis of iterated block ciphers”. In: *ETH Series in Information Processing* 7 (1996) (cit. on p. 38).

[HNI+06] Naofumi Homma, Sei Nagashima, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. “High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching”. In: *CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. LNCS. Springer, Heidelberg, Oct. 2006, pp. 187–200 (cit. on p. 21).

[HS14] Michael Hutter and Jörn-Marc Schmidt. *The Temperature Side Channel and Heating Fault Attacks*. Cryptology ePrint Archive, Report 2014/190. http://eprint.iacr.org/2014/190. 2014 (cit. on p. 19).

[IKL+13] Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. “Robust Pseudorandom Generators”. In: *ICALP 2013, Part I*. Ed. by Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg. Vol. 7965. LNCS. Springer, Heidelberg, July 2013, pp. 576–588. DOI: 10.1007/978-3-642-39206-1_49 (cit. on p. 72).

[IKM+13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. “On the Power of Correlated Randomness in Secure Computation”. In: *TCC 2013*. Ed. by Amit Sahai. Vol. 7785. LNCS. Springer, Heidelberg, Mar. 2013, pp. 600–620. DOI: 10.1007/978-3-642-36594-2_34 (cit. on p. 111).

[ISO] ISO. *Information technology – Security techniques – Evaluation criteria for IT security*. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=50341 (cit. on p. 10).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. “Private Circuits: Securing Hardware against Probing Attacks”. In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 463–481 (cit. on pp. 24, 26, 31, 34, 68–73, 87, 107).

[Jol86] I. T. Jolliffe. *Principal Component Analysis*. 1986 (cit. on p. 21).

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. “Differential Power Analysis”. In: *CRYPTO’99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 388–397 (cit. on pp. 19, 24, 33).

[KL51] S. Kullback and R.A. Leibler. “On information and sufficiency”. In: *Annals of Mathematical Statistics*. 1951, pp. 79–86 (cit. on p. 22).

[KM96] Eyal Kushilevitz and Yishay Mansour. “Randomness in Private Computations”. In: *15th ACM PODC*. Ed. by James E. Burns and Yoram Moses. ACM, Aug. 1996, pp. 181–190 (cit. on pp. 102, 131).

[Koc96] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *CRYPTO’96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Heidelberg, Aug. 1996, pp. 104–113 (cit. on pp. 9, 19, 24, 27, 67).

[KOR03] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. “Amortizing Randomness in Private Multiparty Computations”. In: *SIAM J. Discrete Math.* 16.4 (2003), pp. 533–544 (cit. on p. 102).

[KOR99] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. “Characterizing Linear Size Circuits in Terms of Pricacy”. In: *J. Comput. Syst. Sci.* 58.1 (1999), pp. 129–136 (cit. on pp. 103, 104, 119, 130, 131).

[KPP+06] Sandeep Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. “Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker”. In: *CHES 2006*. Ed. by Louis Goubin and Mitsuru Matsui. Vol. 4249. LNCS. Springer, Heidelberg, Oct. 2006, pp. 101–118 (cit. on p. 15).

[KR98] Eyal Kushilevitz and Adi Rosén. “A Randomness-Rounds Tradeoff in Private Computation”. In: *SIAM J. Discrete Math.* 11.1 (1998), pp. 61–80 (cit. on p. 102).

[KY03] Aggelos Kiayias and Moti Yung. “Non-interactive Zero-Sharing with Applications to Private Distributed Decision Making”. In: *FC 2003*. Ed. by Rebecca Wright. Vol. 2742. LNCS. Springer, Heidelberg, Jan. 2003, pp. 303–320 (cit. on p. 103).

[LB88] Pil Joong Lee and Ernest F. Brickell. “An Observation on the Security of McEliece’s Public-Key Cryptosystem”. In: *EUROCRYPT’88*. Ed. by C. G. Günther. Vol. 330. LNCS. Springer, Heidelberg, May 1988, pp. 275–280 (cit. on p. 99).

[Leo88] Jeffrey S. Leon. “A probabilistic algorithm for computing minimum weights of large error-correcting codes”. In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 1354–1359 (cit. on p. 99).

[Lib] Joint Interpretation Library. *Application of Attack Potential to Smartcards.* URL: http://www.sogis.org/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v2-9.pdf (cit. on p. 15).

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". In: *CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. LNCS. Springer, Heidelberg, Sept. 2014, pp. 35–54. DOI: 10.1007/978-3-662-44709-3_3 (cit. on pp. 37, 61, 133).

[LRT12] Victor Lomné, Thomas Roche, and Adrian Thillard. "On the Need of Randomness in Fault Attack Countermeasures - Application to AES". In: (2012), pp. 85–94 (cit. on p. 133).

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (1982), pp. 382–401 (cit. on p. 68).

[Man04] Stefan Mangard. "Hardware Countermeasures against DPA? A Statistical Analysis of Their Effectiveness". In: *CT-RSA 2004*. Ed. by Tatsuaki Okamoto. Vol. 2964. LNCS. Springer, Heidelberg, Feb. 2004, pp. 222–235 (cit. on p. 38).

[Mas93] James L. Massey. "Minimal Codewords and Secret Sharing". In: *Proceedings of the 6th Joint Swedish-Russian International Workshop on Information Theory*. 1993, pp. 276–279 (cit. on p. 29).

[Mas94] J. Massey. "Guessing and Entropy". In: *IEEE International Symposium on Information Theory* (1994) (cit. on p. 44).

[McE78] Robert J McEliece. "A public-key cryptosystem based on algebraic coding theory". In: *DSN progress report* 42.44 (1978), pp. 114–116 (cit. on pp. 71, 99).

[MMC+02] S.W. Moore, R.D. Mullins, P.A. Cunningham, R.J. Anderson, and G.S. Taylor. "Improving smart card security using self-timed circuits". In: *ASYNC* 211 (2002) (cit. on p. 33).

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smartcards*. Ed. by Springer. 2007 (cit. on p. 25).

[MR04] Silvio Micali and Leonid Reyzin. "Physically Observable Cryptography (Extended Abstract)". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Heidelberg, Feb. 2004, pp. 278–296 (cit. on p. 24).

[MSH+04] F Mace, Francois-Xavier Standaert, I Hassoune, Jean-Jacques Quisquater, and Jean-Damien Legat. "A dynamic current mode logic to counteract power analysis". In: *DCIS Conference on Design Of Circuits and Integrated Systems* 11 (2004), pp. 186–191 (cit. on p. 33).

[NP33] J. Neyman and E.S. Pearson. "On the problem of the most efficient tests of statistical hypotheses." In: *Philosophical Transactions of the Royal Society of London, series A, Containing Papers of a Mathematical or Physical Character* (1933) (cit. on p. 40).

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. "Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches". In: *Journal of Cryptology* 24.2 (Apr. 2011), pp. 292–321 (cit. on p. 68).

[NSA] National Security Agency NSA. *TEMPEST: A Signal Problem* (cit. on p. 19).

[NSGD11] M. Nassar, Y. Souissu, S. Guilley, and J.-L. Danger. "Rank Correction: A New Side-Channel Approach for Secret Key Recovery". In: *InfoSecHiComNet* (2011) (cit. on p. 40).

[Pea95] Karl Pearson. "Notes on regression and inheritance in the case of two parents". In: *Proceedings of the Royal Society of London*. 1895, pp. 240–242 (cit. on p. 22).

[PM05] Thomas Popp and Stefan Mangard. "Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints". In: *CHES 2005*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. LNCS. Springer, Heidelberg, 2005, pp. 172–186 (cit. on p. 33).

[PR13] Emmanuel Prouff and Matthieu Rivain. "Masking against Side-Channel Attacks: A Formal Security Proof". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 142–159. DOI: 10.1007/978-3-642-38348-9_9 (cit. on pp. 25, 26, 68).

[Pra62] E. Prange. "The use of information sets in decoding cyclic codes". In: *Information Theory, IRE Transactions on* 8.5 (Sept. 1962), pp. 5–9. ISSN: 0096-1000. DOI: 10.1109/TIT.1962.1057777 (cit. on pp. 71, 99).

[PRB10] Emmanuel Prouff, Matthieu Rivain, and Régis Bévan. *Statistical Analysis of Second Order Differential Power Analysis*. Cryptology ePrint Archive, Report 2010/646. http://eprint.iacr.org/2010/646. 2010 (cit. on pp. 45, 47).

[Pro05] Emmanuel Prouff. "DPA Attacks and S-Boxes". In: *FSE 2005*. Ed. by Henri Gilbert and Helena Handschuh. Vol. 3557. LNCS. Springer, Heidelberg, Feb. 2005, pp. 424–441 (cit. on p. 46).

[QS01] Jean-Jacques Quisquater and David Samyde. "Electromagnetic analysis(EMA): Measures and counter-measures for smart cards". In: 2140 (2001), pp. 200–210 (cit. on p. 19).

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. "Consolidating Masking Schemes". In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 764–783. DOI: 10.1007/978-3-662-47989-6_37 (cit. on pp. 29, 69).

[RCC+08] Denis Réal, Cécile Canovas, Jessy Clédière, M'hamed Drissi, and Frédéric Valette. "Defeating classical Hardware Countermeasures: a new processing for Side Channel Analysis". In: (2008), pp. 1274–1279. DOI: 10.1109/DATE.2008.4484854. URL: http://dx.doi.org/10.1109/DATE.2008.4484854 (cit. on p. 21).

[Riv09] Matthieu Rivain. "On the Exact Success Rate of Side Channel Analysis in the Gaussian Model". In: *SAC 2008*. Ed. by Roberto Maria Avanzi, Liam Keliher, and Francesco Sica. Vol. 5381. LNCS. Springer, Heidelberg, Aug. 2009, pp. 165–183 (cit. on pp. 38, 40, 43, 44, 46, 47, 62).

[RP10] Matthieu Rivain and Emmanuel Prouff. "Provably Secure Higher-Order Masking of AES". In: *CHES 2010*. Ed. by Stefan Mangard and François-Xavier Standaert. Vol. 6225. LNCS. Springer, Heidelberg, Aug. 2010, pp. 413–427 (cit. on pp. 31, 34, 69, 70, 90).

[RV13] Arnab Roy and Srinivas Vivek. “Analysis and Improvement of the Generic Higher-Order Masking Scheme of FSE 2012”. In: *CHES 2013*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. LNCS. Springer, Heidelberg, Aug. 2013, pp. 417–434. DOI: 10.1007/978-3-642-40349-1_24 (cit. on p. 31).

[Sage] The Sage Developers. *Sage Mathematics Software (Version 6.8)*. http://www.sagemath.org. 2015. DOI: 10.5281/zenodo.28513 (cit. on pp. 71, 97).

[Sha48] Claude E. Shannon. “A Mathematical Theory of Communication”. In: 27.3 (1948), pp. 379–423 (cit. on p. 26).

[Sha79] Adi Shamir. “How to Share a Secret”. In: *Communications of the Association for Computing Machinery* 22.11 (Nov. 1979), pp. 612–613 (cit. on pp. 28, 103).

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. “A Stochastic Model for Differential Side Channel Cryptanalysis”. In: *CHES 2005*. Ed. by Josyula R. Rao and Berk Sunar. Vol. 3659. LNCS. Springer, Heidelberg, 2005, pp. 30–46 (cit. on pp. 23, 45, 55).

[SMY06] Francois-Xavier Standaert, Tal G. Malkin, and Moti Yung. *A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks (extended version)*. Cryptology ePrint Archive, Report 2006/139. http://eprint.iacr.org/2006/139. 2006 (cit. on p. 44).

[SPRQ06] FX. Standaert, Eric Peeters, Gael Rouvroy, and Jean-Jacques Quisquater. “An Overview of Power Analysis Attacks Against Field Programmable Gate Arrays”. In: *Proceedings of the IEEE* 94 (Feb. 2006) (cit. on p. 38).

[Sta11] Francois-Xavier Standaert. “Leakage Resilient Cryptography: a Practical Overview”. In: *Proceedings of the ECRYPT Workshop on Symmetric Encryption* (2011) (cit. on p. 25).

[STC07] Rinne S., Eisennbarth T., and Paar C. “Performance Analysis of Contemporary Lightweight Block Ciphers on 8-bit Microcontrollers.” In: *SPEED Software Performance Enhancement for Encryption and Decryption* (2007) (cit. on p. 15).

[Ste88] Jacques Stern. “A method for finding codewords of small weight”. In: *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*. Ed. by Gérard D. Cohen and Jacques Wolfmann. Vol. 388. Lecture Notes in Computer Science. Springer, 1988, pp. 106–113 (cit. on p. 99).

[SYY99] Tomas Sander, Adam Young, and Moti Yung. “Non-Interactive CryptoComputing For NC1”. In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 554–567 (cit. on p. 106).

[TPR13] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. “Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack”. In: *CHES 2013*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. LNCS. Springer, Heidelberg, Aug. 2013, pp. 21–36. DOI: 10.1007/978-3-642-40349-1_2 (cit. on pp. 37, 61, 133).

[Tre68] Harry L. Van Trees. “Detection, Estimation, and Modulation Theory”. In: *John Wiley & Sons* (1968) (cit. on p. 22).

[VGRS11] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. *An optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks*. Cryptology ePrint Archive, Report 2011/610. http://eprint.iacr.org/2011/610. 2011 (cit. on p. 40).

[WO11a] Carolyn Whitnall and Elisabeth Oswald. “A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework”. In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 316–334 (cit. on pp. 40, 60).

[WO11b] Carolyn Whitnall and Elisabeth Oswald. *A Fair Evaluation Framework for Comparing Side-Channel Distinguishers*. Cryptology ePrint Archive, Report 2011/403. http://eprint.iacr.org/2011/403. 2011 (cit. on p. 34).

[Yao82] Andrew Chi-Chih Yao. “Protocols for Secure Computations (Extended Abstract)”. In: *23rd FOCS*. IEEE Computer Society Press, Nov. 1982, pp. 160–164 (cit. on p. 68).

[Yao86] Andrew Chi-Chih Yao. “How to Generate and Exchange Secrets (Extended Abstract)”. In: *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167 (cit. on p. 102).

Résumé

Les cryptosystèmes sont présents dans de nombreux appareils utilisés dans la vie courante, tels que les cartes à puces, ordiphones, ou passeports. La sécurité de ces appareils est menacée par les attaques par canaux auxiliaires, où un attaquant observe leur comportement physique pour obtenir de l'information sur les secrets manipulés. L'évaluation de la résilience de ces produits contre de telles attaques est obligatoire afin de s'assurer la robustesse de la cryptographie embarquée. Dans cette thèse, nous exhibons une méthodologie pour évaluer efficacement le taux de succès d'attaques par canaux auxiliaires, sans avoir besoin de les réaliser en pratique. En particulier, nous étendons les résultats obtenus par Rivain en 2009, et nous exhibons des formules permettant de calculer précisément le taux de succès d'attaques d'ordre supérieur. Cette approche permet une estimation rapide de la probabilité de succès de telles attaques. Puis, nous étudions pour la première fois depuis le papier séminal de Ishai, Sahai et Wagner en 2003 le problème de la quantité d'aléa nécessaire dans la réalisation sécurisée d'une multiplication de deux bits. Nous fournissons des constructions explicites pour des ordres pratiques de masquage, et prouvons leur sécurité et optimalité. Finalement, nous proposons un protocole permettant le calcul sécurisé d'un veto parmi un nombre de joueurs arbitrairement grand, tout en maintenant un nombre constant de bits aléatoires. Notre construction permet également la multiplication sécurisée de n'importe quel nombre d'éléments d'un corps fini.

Abstract

Cryptosystems are present in a lot of everyday life devices, such as smart cards, smartphones, set-top-boxes or passports. The security of these devices is threatened by side-channel attacks, where an attacker observes their physical behavior to learn information about the manipulated secrets. The evaluation of the resilience of products against such attacks is mandatory to ensure the robustness of the embedded cryptography. In this thesis, we exhibit a methodology to efficiently evaluate the success rate of side-channel attacks, without the need to actually perform them. In particular, we build upon a paper written by Rivain in 2009, and exhibit explicit formulaes allowing to accurately compute the success rate of high-order side-channel attacks. We compare this theoretical approach against practical experiments. This approach allows for a quick assessment of the probability of success of any attack based on an additive distinguisher. We then tackle the issue of countermeasures against side- channel attacks. To the best of our knowledge, we study for the first time since the seminal paper of Ishai, Sahai and Wagner in 2003 the issue of the amount of randomness in those countermeasures. We improve the state of the art constructions and show several constructions and bounds on the number of random bits needed to securely perform the multiplication of two bits. We provide specific constructions for practical orders of masking, and prove their security and optimality. Finally, we propose a protocol allowing for the private computation of a secure veto among an arbitrary large number of players, while using a constant number of random bits. Our construction also allows for the secure multiplication of any number of elements of a finite field.

Mots Clés

canaux auxiliaires, cryptographie, aléa, calcul multi-parties, masquage

Keywords

side-channel, cryptography, randomness, multi-party computation, masking